Volume 123, Issues 1-2, Pages 1-531 (1 November 2000)



		View: Citations	Go
1. 🗖	Iterative solution of linear systems in the 20th century, Pages 1-33 Yousef Saad and Henk A. van der Vorst SummaryPlus   Full Text + Links   PDF (188 K)		
2.	<b>Eigenvalue computation in the 20th century</b> , <i>Pages 35-65</i> Gene H. Golub and Henk A. van der Vorst <u>SummaryPlus</u>   <u>Full Text + Links</u>   <u>PDF (181 K)</u>		
3.	<i>QR</i> -like algorithms for eigenvalue problems, <i>Pages 67-83</i> David S. Watkins <u>SummaryPlus</u>   <u>Full Text + Links</u>   <u>PDF (130 K)</u>		
4.	The ubiquitous Kronecker product, <i>Pages 85-100</i> Charles F. Van Loan SummaryPlus   Full Text + Links   PDF (130 K)		
5. 🔲	Preconditioning eigenvalues and some comparison of solvers, Pages 101-1 Ronald B. Morgan SummaryPlus   Full Text + Links   PDF (150 K)	15	
6.	For tridiagonals Treplace T with LDL <sup>t</sup> , Pages 117-130 Beresford N. Parlett SummaryPlus   Full Text + Links   PDF (306 K)		
7. 🗖	An overview of relative sin $\Theta$ theorems for invariant subspaces of comple Ilse C. F. Ipsen SummaryPlus   Full Text + Links   PDF (148 K)	x matrices, Pages 131-153	
8.	The trace minimization method for the symmetric generalized eigenvalue Ahmed Sameh and Zhanye Tong SummaryPlus   Full Text + Links   PDF (202 K)	problem, Pages 155-175	
9.	Successive overrelaxation (SOR) and related methods, <i>Pages 177-199</i> A. Hadjidimos SummaryPlus   Full Text + Links   PDF (181 K)		
10. 📘	On asynchronous iterations, <i>Pages 201-216</i> Andreas Frommer and Daniel B. Szyld SummaryPlus   Full Text + Links   PDF (128 K)		
11. 📘	Iterative methods for large continuation problems, Pages 217-240 D. Calvetti and L. Reichel SummaryPlus   Full Text + Links   PDF (458 K)		

	C. Brezinski, M. Redivo-Zaglia and H. Sadok SummaryPlus   Full Text + Links   PDF (157 K)	
13.	Analysis of acceleration strategies for restarted minimal residual methods, <i>Pages 261-292</i> Michael Eiermann, Oliver G. Ernst and Olaf Schneider SummaryPlus   Full Text + Links   PDF (250 K)	
14.	Refining an approximate inverse, Pages 293-306 Robert Bridson and Wei-Pai Tang SummaryPlus   Full Text + Links   PDF (102 K)	
15. 🗖	Scalable preconditioned conjugate gradient inversion of vector finite element mass matrices, Pages 307-321 Joe Koning, Garry Rodrigue and Dan White SummaryPlus   Full Text + Links   PDF (429 K)	
16. 🗖	Robust multigrid methods for nonsmooth coefficient elliptic linear systems, Pages 323-352 Tony F. Chan and W. L. Wan SummaryPlus   Full Text + Links   PDF (275 K)	
17. 🗖	The Rook's pivoting strategy, <i>Pages 353-369</i> George Poole and Larry Neal SummaryPlus   Full Text + Links   PDF (241 K)	
18.	Numerical methods in control, Pages 371-394 Volker Mehrmann and Hongguo Xu Abstract   PDF (169 K)	
19. 🗖	Krylov-subspace methods for reduced-order modeling in circuit simulation, Pages 395-421 Roland W. Freund Abstract   PDF (320 K)	
20.	<b>Tikhonov regularization and the L-curve for large discrete ill-posed problems</b> , <i>Pages 423-446</i> D. Calvetti, S. Morigi, L. Reichel and F. Sgallari SummaryPlus   Full Text + Links   PDF (527 K)	
21.	Symbiosis between linear algebra and optimization, <i>Pages 447-465</i> Dianne P. O'Leary SummaryPlus   Full Text + Links   PDF (124 K)	
22.	Some computational problems arising in adaptive optics imaging systems, <i>Pages 467-487</i> Robert J. Plemmons and Victor P. Pauca SummaryPlus   Full Text + Links   PDF (264 K)	
23.	Numerical linear algebra algorithms and software, Pages 489-514 Jack J. Dongarra and Victor Eijkhout SummaryPlus   Full Text + Links   PDF (158 K)	
24.	The impact of high-performance computing in the solution of linear systems: trends and problems, <i>Pages 515-530</i> Iain S. Duff SummaryPlus   Full Text + Links   PDF (193 K)	
25.	Index, Page 531 Unknown PDF (40 K)	

SummaryPlus | Full Text + Links | PDF (37 K)



JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

Journal of Computational and Applied Mathematics 123 (2000) ix-xii

www.elsevier.nl/locate/cam

#### Foreword

# Numerical Analysis 2000 Vol. III: Linear Algebra

With the year 2000 being elected "The World Mathematical Year", the Journal of Computational and Applied Mathematics decided to publish a series of volumes dedicated to various disciplines of applied mathematics and numerical analysis. The series received the ambitious title "Numerical Analysis in the 20th Century" and contains seven volumes of which the present one is devoted to "Linear Algebra".

From the early days of scientific computing, numerical linear algebra has been driven by the necessity to be able to solve linear systems, to solve eigenproblems, and to understand the meaning of the results. Because many of these problems have to be solved repeatedly in other computational problems, the algorithms have to be robust and as fast as possible. This has led to much activity, and other than only developing algorithms on demand, the involved research has been equally intellectually challenging as in other sciences. The behavior of algorithms under rounding errors was a great source of inspiration for the further development of perturbation theory.

Also, the possibility and the necessity to solve larger problems has led to algorithms for the reduction of the information to lower dimensional subspaces. The theories of iterative methods have been pushed forward by curiosity-driven research as well as by strong pressure from applications.

Numerical analysis and numerical linear algebra in particular, have strongly contributed to the giant leaps that could be made, in scientific computing in recent decades. The scientific problems that can be solved nowadays are bigger by many orders of magnitude than those that could be solved, say, some 30 years ago. Roughly, half of the increased computational power must be attributed to improved algorithms, and the other half has been made possible by the increase of computational speeds in hardware. This impressive increase in scale of computation has led to more complicated applications and this in turn has led to more complicated numerical linear algebra problems, such as Kronecker structures, highly nonnormal matrices, ill-posed problems, nonlinear eigenproblems, etc.

At this point in time, we can conclude that numerical linear algebra has reached a certain level of maturity. There is a solid theoretical basis for the study of various phenomena and the theory is still in flux. There have been times, not so long ago, when leading researchers believed that the theory for this area was more or less complete and that further progress in scientific computing was simply a matter of scaling. Simply stated: one had only to increase the order of the problem and to implement the well-known algorithms efficiently on modern computers. It has turned out that this was a too simple and too pessimistic point of view. Not only have we seen new challenging

problems, but also the rapidly growing problem sizes led to new algorithms. Even parallel processing led to new classes of problems, such as domain decomposition techniques.

Research in numerical linear algebra is active as ever before and witnesses many new developments. As a consequence, we collected in this volume some survey and/or tutorial papers, that illustrate the current high level of knowledge and progress, as well as papers with emerging or promising ideas. In order to cover most research areas of linear algebra, we invited leading researchers to submit a paper in their respective field of interest. Our initial list of invitations contained 30 names, well spread over the areas that we considered as being most representative for Numerical Linear Algebra. The response was quite impressive as can be seen below. In hindsight with respect to our original list, the only major topics missing on the final list are error analysis, perturbation analysis, and the Krylov methods for eigenproblems. Impressive progress has been made in these areas and we wish to mention Wilkinson's book "The Algebraic Eigenvalue Problem" as a landmark in numerical linear algebra.

All papers have been refereed in the usual way, and it was certainly a surprise for us that the whole process could be completed almost as scheduled. The involvement of the leading researchers in numerical linear algebra is nicely illustrated by the observation that most of those who could not find time to write a contribution helped us in the refereeing process.

This volume starts with two historical surveys, one on iterative methods for linear systems, by Y. Saad and H. van der Vorst, and the other on eigenproblems, written by G. Golub and H. van der Vorst.

These two surveys cover the major developments that have taken place in the twentieth century. The reader may find interesting details on how the major algorithmic developments evolved in time. The two papers contain many references, which may serve as starting points for further research in these areas (including the "missing" topics mentioned before).

The papers in this volume can be roughly subdivided into the following groups:

### 1. Eigenproblems (including SVD)

The papers in this group reflect established as well as current research. The QR methods represent a success story in numerical linear algebra. In these methods, we see various ideas that click together in an algorithm that leads to very fast and robust algorithms. D. Watkins presents new views on QR-like methods for the eigenproblem, which leads to a better understanding of how the various approaches are related.

C. Van Loan shows that Kronecker products do occur in many contexts of linear algebra and he gives a nice survey on the topic. Preconditioning for eigenproblems, that is the idea to solve nearby easier problems in order to get good starting values for iterative solution procedures, is currently an important research topic. This aspect is taken care of in a contribution by R. Morgan. An important problem, related to stability, is how to compute the eigenvalues of tridiagonal matrices. B. Parlett gives a brief motivation for the new class of tridiagonal eigensolvers and shows that the key feature here is to represent them as a product of two bidiagonals.

I. Ipsen has contributed a paper on the separation of invariant subspaces of complex matrices. This is an interesting problem, in particular when classical eigenvector computations are suspect because of small mutual angles. An alternative approach for studying sensitivity in eigenproblems, the theory of pseudo-spectra, is not represented in our collection of papers. We refer to the discussion in the Golub-van der Vorst paper for further background and information. The Jacobi–Davidson method is a relatively new branch in the tree of eigenproblem solvers. The underlying idea permits to attack non-standard eigenproblems, such as polynomial eigenproblems. A. Sameh and Z. Tong show that a variant of the trace minimization algorithm is related to the Jacobi–Davidson method and they present a numerical comparison.

#### 2. Linear systems

This has traditionally been the core business of numerical linear algebra, with more emphasis on iterative approaches during the last few decades of the past century. The current issues are represented by the following contributions. A. Hadjidimos highlights the SOR methods, which have played a dominant role in iterative solution approaches for a long time. D. Szyld and A. Pfrommer consider asynchronous iteration methods, inspired by parallel processing possibilities.

Iterative techniques, based on the block-Lanczos algorithm, for the computation of solution paths for continuation problems are discussed by D. Calvetti and L. Reichel.

Two different views on the Lanczos method are discussed in a paper by C. Brezinski, M. Redivo–Zaglia, and H. Sadok: the matrix approach and the formal orthogonal polynomial approach. This leads to convenient treatment of breakdowns in the two-sided Lanczos method for unsymmetric linear systems.

Minimal residual methods (including GMRES) are powerful tools for the iterative solution of large linear systems. A common approach is to restart them when the storage requirements or CPU-time per iteration becomes too high. M. Eiermann, O. Ernst, and O. Schneider present an analysis for efficient restarting techniques.

Preconditioning has always been an essential ingredient for many iterative methods. In the 1990s, the concept of sparse approximate inverses became popular. R. Bridson and W.-P. Tang consider refinement techniques for this way of preconditioning. This includes symbolic factorization algorithms, reorderings, and blocking techniques.

Parallel aspects of the popular conjugate gradients method, for problems related to finite element discretization techniques, are discussed in a paper by G. Rodrigue, J. Koning and D. White.

Many of the large-scale linear problems originate from PDEs, and the study of such systems has significant overlap with research in numerical analysis. Multigrid methods is one particular area where insights from linear algebra and analysis merge fruitfully. T. Chan and J. Wan survey robust multigrid methods for elliptic PDEs with non-smooth coefficients. They highlight how to recover the usual multigrid efficiency for this more difficult class of problems.

The paper by G. Poole and L. Neal on pivoting strategies for direct linear solution methods goes back to the basics. These pivoting strategies and relevant for exploiting the possibilities of modern computer architectures.

## 3. Miscellaneous problems

As we have stated before, numerical linear algebra plays an important role in many other research fields and scientific applications. V. Mehrmann and H. Xu give a compact survey of some key numerical linear algebra problems of control theory and discuss the new developments in the area.

R. Freund describes how to use Krylov subspace methods for generating reduced-order models of linear electric circuits.

D. Calvetti, S. Morigi, L. Reichel and F. Sgallari present existing and new iterative methods for the determination of the Tikhonov regularization parameter for classes of ill-posed problems.

D. O'Leary's paper gives a good impression on how numerical linear algebra has intruded other research areas. She discusses, in particular, the role of linear algebra in Optimization.

Imaging problems give rise to large linear systems for reconstruction from ray tracing information. Computational problems, related to image reconstruction in Adaptive Optics Imaging, are discussed in a paper by R. Plemmons and V. Pauca.

4. Software

Numerical linear algebra has a long tradition in high-quality software. This started with the famous Wilkinson–Reinsch collection, which formed the basis for well-known packages such as EISPACK, LINPACK, and, more recently, LAPACK and ScaLAPACK. This has been very important for the current popularity and influence of our research area. J. Dongarra and V. Eijkhout present an overview of the linear algebra algorithms for which mature software is available.

Modern computer architectures have had a significant impact on the design of linear algebra software, and the linear algebra algorithms have, in turn, influenced the design of computer architectures. Think, for instance, of the famous LINPACK benchmark. I. Duff discusses the trends and current problems related to high-performance computing.

We would like to thank all the people who have contributed to the successful completion of this volume: Luc Wuytack for taking the initiative and for inviting us to be the editors, the authors for their contributions and, last but not least, the referees for their careful reading and constructive criticisms.

Apostolos Hadjidimos Department of Mathematics University of Crete, GR-714 09 Heraklion Greece

Henk van der Vorst Mathematisch Instituut, Universiteit Utrecht Budapestlaan 6, NL-3584 CD Utrecht The Netherlands E-mail address: vorst@math.uu.nl

Paul Van Dooren Department of Mathematical Engineering Université Catholique de Louvain Av. Lemaitre 4, B-1348 Louvain la Neuve Belgium



Journal of Computational and Applied Mathematics 123 (2000) 1-33

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.nl/locate/cam

## Iterative solution of linear systems in the 20th century

Yousef Saad<sup>a,1</sup>, Henk A. van der Vorst<sup>b,\*</sup>

<sup>a</sup>Department of Computer Science and Engineering, University of Minnesota, Minneapolis, USA <sup>b</sup>Department of Mathematics, Utrecht University, P.O. Box 80.010, 3508 TA Utrecht, Netherlands

Received 23 January 2000; received in revised form 2 March 2000

#### Abstract

This paper sketches the main research developments in the area of iterative methods for solving linear systems during the 20th century. Although iterative methods for solving linear systems find their origin in the early 19th century (work by Gauss), the field has seen an explosion of activity spurred by demand due to extraordinary technological advances in engineering and sciences. The past five decades have been particularly rich in new developments, ending with the availability of large toolbox of specialized algorithms for solving the very large problems which arise in scientific and industrial computational models. As in any other scientific area, research in iterative methods has been a journey characterized by a chain of contributions building on each other. It is the aim of this paper not only to sketch the most significant of these contributions during the past century, but also to relate them to one another. © 2000 Elsevier Science B.V. All rights reserved.

*Keywords:* ADI; Krylov subspace methods; Multigrid; Polynomial acceleration; Preconditioning; Relaxation methods; SOR; Sparse approximate inverse

#### 1. Introduction

Numerical linear algebra is an exciting field of research and much of this research has been triggered by a problem that can be posed simply as: given  $A \in \mathscr{C}^{m \times n}$ ,  $b \in \mathscr{C}^m$ , find solution vector(s)  $x \in \mathscr{C}^n$  such that Ax = b. Many scientific problems lead to the requirement to solve linear systems of equations as part of the computations. From a pure mathematical point of view, this problem can be considered as being solved in the sense that we explicitly know its solution in terms of determinants. The actual computation of the solution(s) may however lead to severe complications, when carried out in finite precision and when each basic arithmetic operation takes finite time. Even

<sup>\*</sup> Corresponding author.

E-mail addresses: saad@cs.umn.edu (Y. Saad), vorst@math.uu.nl (H.A. van der Vorst).

<sup>&</sup>lt;sup>1</sup>Work supported by NSF/CCR and by the Minnesota Supercomputer Institute.

the "simple" case when n = m and A is nonsingular, which is a trivial problem from a mathematical point of view, may become very complicated, from a computational point of view, and may even turn out to be impossible.

The traditional way to solve a nonsingular linear system is to employ Gaussian elimination, and, with all its enhancements, to overcome numerical instabilities. This process can be carried out in  $\mathcal{O}(n^3)$  basic floating point operations (additions and multiplications, assuming n = m). Many applications lead to linear systems with a large n (where the notion of "large" depends, of course, on the capacity of the available computer), and it became soon evident that one has to exploit specific properties of the A at hand in order to make solution of the system feasible. This has led to variants of Gaussian elimination in which the nonzero structure of A is exploited, so that multiplications with zero result are avoided and that savings in computer storage could be realized.

Another direction of approach was based on the solution of a nearby linear system, with a matrix that admits a computationally inexpensive process (in terms of computing time and computer storage), and to embed this in an iterative process. Both approaches aim at making the impossible possible, and for the novice in this field this may seem to be just a collection of clever programming tricks: "in principle solving the problem is well understood but one has to be well organized to make the computational process a little faster". For this novice it will certainly come as a big surprise that a whole, still incomplete, mathematical framework had to be developed with deep and elegant results. As a result, relevant systems could be solved many orders of magnitude faster (and also often more accurate) than by a straightforward Gaussian elimination approach. In this paper, we will sketch the developments and progress that has taken place in the 20th century with respect to iterative methods alone. As will be clear, this subfield could not evolve in isolation, and the distinction between iterative methods and Gaussian elimination methods is sometimes artificial – and overlap between the two methodologies is significant in many instances. Nevertheless, each of the two has its own dynamics and it may be of interest to follow one of them more closely.

It is likely that future researchers in numerical methods will regard the decade just passed as the beginning of an era in which iterative methods for solving large linear systems of equations started gaining considerable acceptance in real-life industrial applications. In looking at past literature, it is interesting to observe that iterative and direct methods have often been in competition for solving large systems that arise in applications. A particular discovery will promote a given method from one camp only to see another discovery promote a competing method from the other camp. For example, the 1950s and 1960s saw an enormous interest in relaxation-type methods – prompted by the studies on optimal relaxation and the work by Young, Varga, Southwell, Frankel and others. A little later, sparse direct methods appeared that were very competitive – both from the point of view of robustness and computational cost. To this day, there are still applications dominated by direct solvers, it was sometimes thought that these would eventually be replaced by iterative solvers, in all applications. However, the superior robustness of direct solvers prevented this. As computers have become faster, very large problems are routinely solved by methods from both camps.

Iterative methods were, even halfway in the 20th century, not always viewed as promising. For instance, Bodewig [23, p. 153], in 1956, mentioned the following drawbacks of iterative methods: nearly always too slow (except when the matrix approaches a diagonal matrix), for most problems

3

they do not converge at all, they cannot easily be mechanised<sup>2</sup> and so they are more appropriate for computing by hand than with machines, and do not take advantage of the situation when the equations are symmetric. The only potential advantage seen was the observation that Rounding errors do not accumulate, they are restricted to the last operation. It is noteworthy that Lanczos' method was classified as a direct method in 1956.

The penetration of iterative solvers into applications has been a slow process that is still ongoing. At the time of this writing for example, there are applications in structural engineering as well as in circuit simulation, which are dominated by direct solvers.

This review will attempt to highlight the main developments in iterative methods over the past century. It is clear that a few pages cannot cover an exhaustive survey of 100 years of rich developments. Therefore, we will emphasize the ideas that were successful and had a significant impact.

Among the sources we used for our short survey, we would like to mention just a few that are notable for their completeness or for representing the thinking of a particular era. The books by Varga [188] and Young [205] give a complete treatise of iterative methods as they were used in the 1960s and 1970s. Varga's book has several excellent historical references. These two masterpieces remained the handbooks used by academics and practitioners alike for three decades. Householder's book [102] contains a fairly good overview of iterative methods – specifically oriented towards projection methods. Among the surveys we note the outstanding booklet published by the National Bureau of Standards in 1959 which contains articles by Rutishauser [150], Engeli [68] and Stiefel [170]. Later Birkhoff [21], who supervised David Young's Ph.D. thesis in the late 1940s, wrote an excellent historical perspective on the use of iterative methods as he experienced them himself from 1930 to 1980. The more recent literature includes the books by Axelsson [7], Brezinski [29], Greenbaum [88], Hackbusch [97], and Saad [157], each of which has a slightly different perspective and emphasis.

#### 2. The quest for fast solvers: a historical perspective

Iterative methods have traditionally been used for the solution of large linear systems with diagonally dominant sparse matrices. For such systems the methods of Gauss–Jacobi and Gauss–Seidel could be used with some success, not so much because of the reduction in computational work, but mainly because of the limited amount of memory that is required. Of course, reduction of the computational work was also a serious concern, and this led Jacobi (1846) to apply plane rotations to the matrix in order to force stronger diagonal dominance, giving up sparsity. Jacobi had to solve many similar systems in the context of eigenvalue computations; his linear systems were rather small: of order 7.

In this century, simple iterative methods were predominantly applied for solving discretized elliptic self-adjoint partial differential equations, together with a local parameter for accelerating the iteration process. The first and simplest of these methods in Richardson's method [146]. Actually, this method

<sup>&</sup>lt;sup>2</sup> This remark was removed from the second edition (in 1959); instead Bodewig included a small section on methods for automatic machines [24, Chapter 9]. The earlier remark was not as puzzling as it may seem now, in view of the very small memories of the available electronic computers at the time. This made it necessary to store intermediate data on punched cards. It required a regular flow of the computational process, making it cumbersome to include techniques with row interchanging.

4

was later viewed as a polynomial method and many authors have sought to optimize it by selecting its parameters so that the iteration polynomials became the Chebyshev polynomials; this was work done in the period 1950–1960 by Young, Lanczos and others. In the second half of this decade it became apparent that using the explicit three-term recurrence relation between Chebyshev polynomials, which led to three-term recurrence iteration methods (rather than the classical methods that are two-term iterations), were numerically superior in terms of stability [87].

The acceleration of the slightly more difficult to analyze Gauss-Seidel method led to point successive overrelaxation techniques introduced simultaneously by Frankel [78] and by Young [203]. It was shown, for rather simple Dirichlet problems, that a suitably chosen relaxation parameter could lead to drastic improvements in convergence. Young showed that these improvements could be expected for a larger class of matrices, characterized by his property A [203]. Successive overrelaxation methods, and numerous variants, became extremely popular and were the methods of choice in computer codes for large practical problems, such as nuclear reactor diffusion, oil reservoir modeling and weather prediction. Although their popularity has been overshadowed later, around after 1980, by more powerful techniques, they are still used in some applications either as the main iterative solution method or in combination with recent techniques (e.g. as smoothers for multigrid or as preconditioners for Krylov methods). The successive over-relaxation (SOR) methods made it possible to solve efficiently systems within the order of 20,000 unknowns by 1960 [188], and by 1965 systems of the order of 100,000 could be solved in problems related to eigenvalue computations in nuclear diffusion codes. The success of the SOR methods has led to a rich theory for iterative methods; this could be used fruitfully for the analysis of later methods as well. In particular, many methods, including SOR, could be viewed as simple Richardson iterations for specific splittings of the matrix of the linear system.

In 1955, Peaceman and Rachford [141] suggested a splitting that was motivated by the observation that the matrix for a three-point finite difference stencil for a one-dimensional second-order PDE is tridiagonal and this system can easily be solved. Their suggestion was to view the five-point finite difference approximation for a two-dimensional problem as the direct sum of two one-dimensional approximations. This led to an iteration in which alternatingly a tridiagonal associated with one of the two directions was split off, and this became popular as the alternating direction iteration (ADI). With the inclusion of iteration parameters, that steered the inclusion of a diagonal correction to the iteration matrices, the resulting ADI iterations could be tuned into a very effective method. Varga [188] gives a good overview of the theory for understanding ADI methods. He, as well as Birkhoff [21] mentions that ADI was initially derived as a by-product of numerical methods for parabolic equations (the correction to the diagonal was motivated by the effect of the time derivative in these methods). Sheldon and Wachspress, in 1957, gave an early proof for the convergence of ADI for fixed parameters [192]. Wachspress discusses these ADI methods in his book [193] and considers also other grid-oriented acceleration techniques. One of these techniques exploits approximations obtained on coarser grids and can be viewed as a primitive predecessor to multigrid.

The first half of the century begins also with simple local projection methods, in which one attempts to solve a set of equations by solving each separate equation by a correction that is small in some norm. These methods could be used for over- or underdetermined linear systems, such as those that arise in tomography problems. This has led to the methods of Cimmino [44] and Kaczmarz [106], which were later identified as instances of Gauss–Jacobi and or Gauss–Seidel for related systems with  $A^{T}A$  or  $AA^{T}$ . Modern variants of these methods, under the name of ART and

5

SIRT are very popular, for instance in medical and seismic tomography. ART and SIRT can be related to SOR and Block SOR. Spakman and Nolet [168] report on the solution of 292, 451 by 20,070 systems related to structures of the upper earth mantle, with these methods (and with LSQR).

The second half of the century was marked by the invention (paper published in 1952) of the conjugate gradient method by Hestenes and Stiefel [101] and the Lanczos algorithm for linear systems [117]. This started the era of Krylov iterative methods. Initially, these were not viewed as truly iterative techniques, but rather as direct solution algorithms since they terminated in exact arithmetic in fewer than n steps, if n is the order of the matrix (see, for instance, Householder's book where conjugate gradients is discussed in the chapter on direct methods [102, Chapter 5.7]). Hestenes and Stiefel already recognized that the method behaves as an iterative method, in the sense that the norm of the residual often decreases quite regularly, and that this might lead for some systems to acceptable approximations for the solution within n steps. A little earlier, papers by Lanczos [115] and by Arnoldi [2] had addressed the issue of transforming a matrix into simpler form for the purpose of diagonalizing it. These four papers together set the foundations of many methods that were developed later.

A famous publication by Engeli et al. [69] considered the method as a truly iterative process and showed that in rounding precision arithmetic, the conjugate gradient method did not terminate in the expected number of iteration steps (equal to at most the order of the matrix). This was shown for a matrix of order 64, a discretized biharmonic problem. Convergence occurred only after a few hundred steps. Notwithstanding this apparent failure, the method appeared later in the famous Wilkinson and Reinsch collection [202] as a kind of memory-friendly direct technique. It was mentioned that actual convergence might occur only after m iterations, where m could be 3 up to five times the order of the matrix. Because of this not well-understood behavior in rounded arithmetic, the method did not make it to the first universal linear algebra package LINPACK (mid-1970s). In the early to mid-1960s it became clear that the convergence of the conjugate gradient method depends on the distribution of the eigenvalues of the matrix, and not so much on the order of the matrix, as was, for example, explained in a paper by Kaniel [109]. Daniel [50,51] studied the conjugate gradient method as an iterative method for the minimization of functionals in (infinite dimensional) Hilbert spaces. This is a natural consequence of the observation that conjugate gradients, like other Krylov subspace methods, requires the action of the matrix as a linear operator and does not exploit the actual representation of the matrix (that is, the method does not require knowledge of the individual entries of the matrix). Also, Daniel expressed concerns about the convergence behavior of the method in finite precision, and he discussed modifications with guaranteed convergence [51, p. 134]. Note also that much of the convergence theory developed for the conjugate gradient and the Lanczos methods was almost invariably set in the context of operators on infinite-dimensional spaces, see, for example [109].

It was Reid [145] who suggested to use the conjugate gradient method again as an iterative technique, but now for large sparse linear systems arising in the discretization of certain PDEs. Soon after this, the notion of preconditioning (already proposed in the Hestenes and Stiefel paper) became quite popular. Thus, the incomplete Choleski decompositions of Meijerink and van der Vorst [125] led to the ICCG process, which became the de facto iterative solver for SPD systems.

Hence, it took about 25 years for the conjugate gradient method to become the method of choice for symmetric positive-definite matrices (the incomplete Choleski decompositions were shown to exist for M matrices). A good account of the first 25 years of the history of the CG method was given by Golub and O'Leary [86].

The unsymmetric variants of the Krylov methods required a similar amount of time to mature. The late 1960s and early 1970s, saw the roots for such methods. Techniques named ORTHODIR, ORTHOMIN, FOM, and others, were introduced but in their original formulations, these methods suffered from breakdowns and numerical instabilities. The GMRES variant, introduced by Saad and Schultz [158], was designed to avoid these undesirable features and became the de facto standard for unsymmetric linear systems. However, it suffered from the disadvantage of requiring increasing computational resources for increasing numbers of iterations. Bi-CG, the unsymmetric variant of conjugate gradients, did not have these disadvantages. The method, based on the unsymmetric Lanczos method (1952), was introduced by Fletcher in 1976 [76], but it is mathematically equivalent to a technique that had already been described in Lanczos' paper. Bi-CG, however, suffered from other practical problems, known as breakdowns of the first and second kind, which prevented early success. Moreover, the occurrence of nonorthogonal transformations led to much suspicion among numerical analysts. Nevertheless, the method became quite popular in a variant known as CGS (Sonneveld, 1984) [166] which, for virtually equal cost could essentially apply Bi-CG twice, leading often to a twice as fast convergence, but also amplifying the problems of Bi-CG. In the 1980s. Parlett and co-authors [140] and later Freund and Nachtigal [81] have shown how to repair the deficiencies in the Bi-CG method so that rather reliable software could be constructed. More recently, we have seen hybrids of the Bi-CG and GMRES approaches, with Bi-CGSTAB [186] as one of the most popular ones.

Originally, the usage of iterative methods was restricted to systems related to elliptic partial differential equations, discretized with finite difference techniques. Such systems came from oil reservoir engineering, weather forecasting, electronic device modeling, etc. For other problems, for instance related to various finite element modeling, practitioners preferred the usage of direct solution techniques, mainly efficient variants of Gaussian elimination, because of the lack of robustness of iterative methods for large classes of matrices. Until the end of the 1980s almost none of the big commercial packages for finite element problems included iterative solution techniques. Simon [164] presented results, obtained for matrices of the order of 55,000, for direct solution techniques. On the then fastest supercomputers, this required in the order of a few minutes of computing time. He claimed that direct sparse solvers would remain the method of choice for irregularly structured problems. Although this is certainly true if the structure of the matrix allows for an efficient elimination process, it became clear that for many PDE-related problems, the complexity of the elimination process increased too much to make realistic three-dimensional modeling feasible. Irregularly structured finite element problems of order 1,000,000, as foreseen by Simon, may be solved by direct methods - given a large enough computer (memory wise) but at tremendous cost and difficulty. However, some of them can be solved with iterative techniques, if an adequate preconditioning can be constructed. In the last decade of this century, much effort was devoted to the identification of effective preconditioners for classes of matrices. For instance, Pomerell [142] in 1994 reports on successful application of preconditioned Krylov methods for very ill-conditioned unstructured finite element systems of order up to 210,000 that arise in semiconductor device modeling.

While using iterative methods still requires know-how, skill, and insight, it can be said that enormous progress has been made for their integration in real-life applications. Still, linear systems arising from many relevant problems, for instance large electric and electronic circuits, are not easy to solve in an efficient and reliable manner by iterative methods. Steady progress is being made but the field as a whole can still be viewed as being in its infancy.

#### 3. Relaxation-based methods

The Gauss–Seidel iteration was the starting point for the successive over-relaxation methods which dominated much of the literature on iterative methods for a big part of the second half of this century. The method was developed in the 19th century, originally by Gauss in the mid-1820s and then later by Seidel in 1874 (see references in [102]). In fact, according to Varga [188], the earliest mention on iterative methods is by Gauss (1823).

Indeed, on December 26, 1823, Gauss writes a letter to Gerling, in which he describes an iterative technique for the accurate computation of angles occurring in geodesy [84, p. 278]. The corrections for the four angles in a quadrangle, determined by four church towers, were computed from a singular linear systems of four equations with four unknowns (the singularity comes from the observation that the four angles sum up to 360°). The technique that Gauss describes is what we now know as the Gauss–Seidel algorithm. The order of processing of the equations was determined by the unknown that helped to reduce the residual most. Gauss recognized that the singularity of the system led to convergence to the solution modulo a vector in the null space, for which he could easily make a correction. The three pages of his letter are full of clever tricks. He concludes by recommending the new method to Gerling, arguing that the method is self correcting, and that one can easily determine how far to go and then ends his letter with the remark that the computations were a pleasant entertainment for him. He said that one could do this even half asleep, or one could think of other things during the computations. In view of this remark it may hardly be a surprise that the method became so popular in the era of electronic computing.

The method as it was developed in the 19th century was a relaxation technique, in which relaxation was done by "hand". It was therefore natural to eliminate the largest components, see for example [55,118]. This method is referred to as Nekrasov's method in the Russian literature [130]. Referring to the more modern method in which relaxation was done in a cyclic manner, Forsythe is quoted as having stated that "the Gauss–Seidel method was not known to Gauss and not recommended by Seidel", see [102, p. 115].

However, the blossoming of overrelaxation techniques seems to have been initiated by the Ph.D. work of David Young [203]. Young introduced important notions such as *consistent ordering* and *property A*, which he used for the formulation of an elegant theory for the convergence of these methods. Generalizations of Young's results to other relevant classes of matrices were due to Varga, who published his book on Matrix Iterative Analysis in 1962. For decades to come this book was the standard text for iterative methods for linear systems. It covered important notions such as *regular splittings*, a rather complete theory on *Stieltjes* and *M-matrices*, and a treatment of semi-iterative methods, including the Chebyshev semi-iteration method. The latter method, analyzed by Golub and Varga [87], also became more widely known, especially in the period when inner products were relatively expensive.

The accelerated Gauss–Seidel methods have motivated important developments in the theory of matrix linear algebra. In particular, relevant properties for *M*-matrices, introduced by Ostrowski [135], were uncovered and convergence results for so-called *regular splittings*, introduced by Varga [189] were established. A cornerstone in the convergence theory was the theorem of Stein–Rosenberg (1948) [169] which proved relations between the asymptotic rates of convergence for the successive overrelaxation methods, including the Gauss–Seidel method, and the Gauss–Jacobi method. The concept of *irreducibility* of a matrix, a natural property for grid-oriented problems, helped to extend

results for strongly diagonally dominant matrices to matrices for which the strict diagonal dominance inequality is required to hold only for one single equation at least. Another important notion is the concept of cyclic matrices: an irreducible matrix with k eigenvalues of modulus  $\rho(A)$  is said to be of index k. Varga [188] gives a good overview of the relevant theory and the implications of this concept for iterative methods. It has a close relationship with Young's property A [188, p. 99], and provides the basis for the convergence theory of the SOR methods. Sufficient conditions for the convergence of the SOR methods were given by theorems of Ostrowski [136] and Reich [144]. Lower bounds for the spectral radius of the SOR iteration matrix were derived by Kahan [107]. This together provided the basis for a theory for iterative methods, published in Varga's book [188] from which many new methods emerged. Later, in the 1970s major part of this theory served well in the development of preconditioners for Krylov methods.

The following is a quotation from Varga's book (page 1) "As an example of the magnitude of problems that have been successfully solved on digital computers by cyclic iterative methods, the Bettis Atomic Power laboratory of the Westinghouse Electric Corporation had in daily use in 1960 a two-dimensional program which would treat as a special case, Laplacean-type matrix equations of order 20,000". So the state of the art in 1960 was a  $20,000 \times 20,000$  Laplace equation.

In the late 1960s and early 1970s a number of methods appeared in which the order of relaxation was not prescribed or even deterministic. These were appropriately termed "chaotic" or "asynchronous" relaxations. It was established that if a variable is relaxed an infinite number of times, the global method would always converge for any order in which the relaxation takes place. A few of the main contributions were by Chazan and Miranker [41], Miellou [128], Robert [147] and Robert et al. [148]. These methods were motivated by parallelism and were essentially ahead of their time for this reason.

#### 4. Richardson and projection methods

Another line of development started with Richardson's method [146].

$$x_{k+1} = x_k + \omega r_k = (I - \omega A)x_k + \omega b,$$

which can be viewed as a straightforward iteration associated with the splitting A = K - R, with  $K = (1/\omega)I$ ,  $R = (1/\omega)I - A$ . Here  $r_k$  is the residual vector of the current iterate:

$$r_k = b - Ax_k$$
.

For the residual at the (k + 1)th step, one obtains

$$r_{k+1} = (I - \omega A)^{k+1} r_0 = P_{k+1}(A) r_0,$$

where  $P_{k+1}(A)$  is a k + 1 degree polynomial in A, with  $P_{k+1}(t) = (1 - t)^{k+1}$ . It is easy to see that for symmetric positive-definite matrices the process will converge for  $\omega$  in the open interval  $0 < \omega < 2/\lambda_{\text{max}}$  where  $\lambda_{\text{max}}$  is the largest eigenvalue of A. In addition the best  $\omega$  is known to be  $2/(\lambda_{\min} + \lambda_{\max})$ , see, e.g., [188,157] for details.

The original Richardson iteration is readily generalized by taking a different  $\omega = \omega_k$  for each iteration, which leads to the generalized Richardson iteration

$$x_{k+1} = x_k + \omega_k r_k. \tag{1}$$

The sequence of  $\omega_k$ s can be selected in a number of different ways. Note that the residual vector after step k + 1 is given by

$$r_{k+1} = (I - \omega_k A)(I - \omega_{k-1} A) \cdots (I - \omega_0 A)r_0,$$
<sup>(2)</sup>

which shows that we can obtain any desired polynomial  $P_{k+1}$  with the property that  $P_{k+1}(0) = 1$ , by selecting its roots as the iteration parameters in (1). This process is referred to as polynomial acceleration of the simple splitting for A that we gave above. It was studied by, among others, Young [204], Shortley [162], and Lanczos [116]. By 1962 it was not considered competitive, since, as quoted from [188, p. 159]: "*Richardson's method has the disadvantage of being numerically unstable*".

In fact, the Chebyshev semi-iteration method [87] can, in exact arithmetic, be obtained from these polynomial accelerated methods, by choosing the acceleration parameters in successive Richardson iterations properly, but this approach is unstable. In the Chebyshev semi-iteration method one exploits the three term recurrence relation for Chebyshev polynomials, which leads to a stable three term recurrence iterative method. The main problem with these Chebyshev methods is that one needs fairly accurate information about extremal eigenvalues, since these define the interval for the Chebyshev polynomials.

The method of steepest descent which is attributed to Cauchy (1847) is also of the form (1). Kantorovitch later considered the method in a 1945 paper [110] that appeared in the Russian Doklady Akademii Nauk SSSR. In this case, the scalar  $\omega_k$  is selected so as to minimize the quadratic form

$$J(x) = \frac{1}{2}(Ax, x) - (b, x)$$

in the direction of  $r_k$ .

In the 1950s and 1960s other matrix splittings were suggested as basis for iteration methods. We mentioned before one such splitting, namely the ADI method of Peaceman and Rachford [141]. In 1968, Stone [171] proposed the strongly implicit procedure, which is, in fact, a simple Richardson iteration with a series of splittings of the matrix. The idea, for a five-point finite difference stencil, is to factor the matrix in a lower triangular matrix and an upper triangular matrix each with a three-point stencil (as in incomplete LU with no fill-in). The factors are chosen in such a way that the fill-in is spread over the seven-point stencil of the product matrix. This is different from the Kendall-Dupont-Rachford [60] decomposition, where the fill-in is compensated by a correction to the diagonal in such a way that the sum of the elements of the error matrix equals zero for each row. Convergence in the SIP method is achieved by a set of iteration parameters (up to 18) for different distributions of the elements in the error matrix. The choice of the parameters is motivated by a Fourier analysis. Each value of the iteration parameter kills some components in the error (but may lead to increase in other components). Successful application of SIP requires to apply a special order of the iteration parameters, and for each value the decomposition has to be carried out from top to bottom and next from bottom to top. The SIP method gained quite some popularity in oil reservoir simulation and groundwater simulation problems, but its usage seem to have declined in favor of the Krylov methods. The Dupont-Kendall-Rachford splitting was proposed to be used in combination with Chebyshev polynomial acceleration.

In 1937 and 1938 two papers were published on methods that can be termed 'row-projection methods' (or column projection methods). These methods proposed by Kaczmarz [106] and Cimmino

[44] were also based on one-dimensional corrections:

$$x_{k+1} = x_k + \omega_k a_{i,.},\tag{3}$$

where  $a_{i,.} = A^{T}e_{i}$  is the *i*th row vector of *A*. These rows are cycled through from 1 to *n*. Here  $\omega_{k}$  is selected so that the *i*th component of  $r_{k+1}$  becomes zero. Because  $r_{k+1} = r_{k} - \omega_{k}AA^{T}e_{i}$ , it is easy to see that this method is mathematically equivalent to the Gauss–Seidel iteration applied to the normal equations

$$AA^{\mathrm{T}}y = b, \quad x = A^{\mathrm{T}}y.$$

The method proposed by Cimmino was the Jacobi equivalent of this approach. It is also possible to define similarly a Gauss–Seidel approach for the normal equations

$$A^{\mathrm{T}}Ax = b$$

as was noted by Björck and Elfving [22], and this would correspond to taking directions  $d_k$  along the columns of the matrix. This class of methods regained interest in the 1970s and 1980s with the work of Björck and Elfving [22], Tanabe [174], and later Kamath and Sameh [108] and Bramley and Sameh [25].

However, one-dimensional projections methods of a different type, based on very general definitions of norms were very popular in the later 1950s. Here, we mention the work of Gastinel among others. Gastinel's approach [83] consisted of defining generating vectors for norms. Consider an arbitrary vector norm  $\phi$  (for example the norm  $\|.\|_1$ ). Gastinel defines the vector v which realizes the norm of a vector r in the sense

$$(v,r) = \phi(r).$$

For example, for the 1-norm, the components of v can be defined as  $v_i = \text{sign}(e_i^T r)$ , where r = b - Ax is the current residual vector. This vector v is chosen to do an orthogonal projection step. The method can be shown to converge for any nonsingular matrix.

#### 5. Second-order and polynomial acceleration

An important observation regarding all acceleration methods of the form (1) is that their residuals take the form (2), so there is room for improvement to the scheme if successive iterates are considered.

In 1950 Frankel [78] proposed an acceleration method which used a three-term recurrence of the form

$$x_{k+1} = x_k + \beta_k \delta_k, \quad \delta_k = r_k - \alpha_k \delta_{k-1}.$$

This "second-order Richardson process" is initialized by  $\delta_{-1} = r_{-1} = 0$ . Frankel's method [78] uses constant coefficients and results in a residual polynomial which is a combination of Chebyshev polynomials of the first and second kind.

Naturally Chebyshev polynomials should give rise to optimal-like behavior and a number of authors discovered, rediscovered, or enhanced the method at various times. The paper by Flanders and Shortley [75] showed how to use Chebyshev polynomials for eigenvalue calculations. Later

Shortley [162] adapted this method for the solution of linear systems. In the Russian literature, Gavurin [85] also introduced the idea independently in 1950. In 1954, Young [204] proposed a method which amounted to compounding (or restarted) Chebyshev iterations. However the method was in the form (1) – which is unstable. Young gave some remedies to the process which consisted of reordering the roots  $\omega_k$  before applying the polynomial.

In the particular case where A is symmetric positive definite the eigenvalues are located in an interval  $[\lambda_{\min}, \lambda_{\max}]$ . The best residual polynomial  $1 - \lambda s(\lambda)$  in this case is a shifted and scaled Chebyshev polynomial of the first kind, and its three-term recurrence results in a simple three-term recurrence for the approximate solution, see, e.g., [157].

Using a different approach altogether, Lanczos in a 1952 paper [116] discusses the use of certain polynomials that are optimal in a least-squares sense. This paper, which was by and large overlooked by researchers, made a number of contributions. Its focus is on symmetric positive-definite matrices – for general matrices the author resorts to the normal equations. One of the main ideas proposed is to consider the problem of approximating the delta function in the interval [0, 1] which contains all eigenvalues (after the matrix is scaled by its largest eigenvalue). He then transforms the variable using the change of variables  $x = (1 - \cos \theta)/2$ . Now a least-squares approximation to the delta function is sought in the trigonometric basis. This leads to the so-called Dirichlet kernel whose solution is well known

$$P_k(\theta) = \frac{\sin(k + \frac{1}{2})\theta}{(k + \frac{1}{2})\sin\theta}$$
$$= \frac{1}{k + \frac{1}{2}} \left[\frac{1}{2} + \cos\theta + \cos 2\theta + \dots + \cos k\theta\right]$$

To avoid the high oscillations around discontinuities, the so-called Gibbs phenomenon, Lanczos suggested a strategy due to Fejer.

Later, a remarkable paper by Stiefel gave a fairly complete view on similar ideas revolving around least-squares polynomials [170]. The above paper by Lanczos was not referenced by Stiefel. It is only in 1983 that the idea of using least-squares polynomials resurfaced in force again, motivated essentially by parallelism and vector processing. Earlier in 1979 a paper by Dubois et al. [58] suggested using simple Neumann series expansion of the matrix. In 1976 Axelsson addressed the problem of computing good polynomials when the spectrum is located in two intervals, and he was followed later in 1980 by deBoor and Rice [54] who showed how to compute the best min–max polynomial in this situation and the more general situation of multiple intervals. The least-squares alternative considered by Johnson et al. in [104] was for a single interval, assuming that A is symmetric positive definite. In other words, we need to solve

Find  $s \in \Pi_k$  that minimizes:

$$\|1 - \lambda s(\lambda)\|_{w}, \tag{4}$$

where w is some weight function on the interval  $(\lambda_{\min}, \lambda_{\max})$ , and  $\|.\|_w$  is the  $L_2$ -norm associated with the corresponding inner product. Because the distribution of eigenvalues matters more than condition numbers for the preconditioned conjugate gradient method, the authors observed in [104]

that least-squares polynomials tend to perform better than those based on the uniform norm, because they result in a better clustering of the spectrum. Moreover, Lanczos [116] and Rutishauser [150] already noted that the eigenvalue estimates need not be accurate: in fact, it suffices to use the simple bounds that are provided by Gershgorin's theorem. Further experiments in [151] did confirm that in some cases the least-squares polynomial over the Gershgorin interval, may perform as well as the infinity norm polynomial over [ $\lambda_{\min}, \lambda_{\max}$ ]. Note that this is only a minor advantage of least-squares polynomials since effective adaptive procedures exist to compute  $\lambda_{\min}, \lambda_{\max}$ ; see [98] for symmetric problems and [123,66] for nonsymmetric problems. We should add that the observations made in [104,151], and the simplicity of a method that bypasses eigenvalue estimates, have made least-squares polynomials more popular for polynomial preconditionings.

In the more general nonsymmetric case the interval (or union of intervals) that contains the spectrum is to be replaced by a more complex continuum E in  $\mathbb{C}$ , which ideally would contain the eigenvalues of the matrix A. Several choices have been used for E. The first idea, proposed by Manteuffel in 1977–1978 [122,123], is to use an ellipse E that encloses an approximate convex hull of the spectrum, or more precisely, the field of values of A. Then the shifted and scaled Chebyshev polynomials are optimal or nearly optimal and the use of these polynomials leads again to an attractive three-term recurrence. He exploited the fact that an unaccurate guess of extremal eigenvalues leads to either divergence or very slow convergence, in which the eigenvectors corresponding to the unidentified extremal eigenvalues play a dominant role. After a few iterations these directions can be identified and the parameters for the Chebyshev iteration polynomials can be adjusted. Although superseded by the parameter-free Krylov iteration methods, the Chebyshev methods are still of interest on computer platforms where the inner products are relatively expensive. They can be used in combination with Krylov methods, either as polynomial-type preconditioners in order to damp dominating parts of the spectrum, or to continue the iteration with the eigenvalue guesses that can be obtained from the Krylov methods (the Ritz values).

A second alternative is to use a polygon H that contains  $\sigma(A)$  [160,152]. A notable advantage of using polygons is that they may better represent the shape of an arbitrary spectrum. The polynomial is not explicitly known but it may be computed by a Remez algorithm. As in the symmetric case an alternative is to use some weighted  $L_2$ -norm instead of the infinity norm. Saylor and Smolarski used a discrete norm on the polygon [160]. Saad [152] used an  $L_2$ -norm associated with Chebyshev weights on the edges of the polygon and expressed the best polynomial as a linear combination of Chebyshev polynomials associated with the ellipse of smallest area containing H.

Yet another attractive possibility, with polygons instead of ellipses, proposed by Fischer and Reichel [74] is to avoid the problem of best approximation altogether and interpolate the function 1/z with a polynomial at the Fejer points of *E*, i.e., the points  $e^{2ji\pi/k}$ ,  $j=0,\ldots,k$  that are conformally mapped from the unit circle to *H*. This is known to be an asymptotically optimal process. There are numerous publications related to this approach and the use of Faber polynomials; see the references in [74].

#### 6. Krylov subspace methods: the first period

In the early 1950s a number of new methods appeared that dramatically changed the landscape of iterative methods. In separate contributions Lanczos [117] and Hestenes and Stiefel [101] propose

in effect different versions of what is now known as the conjugate gradient method. Amazingly, Hestenes and Stiefel actually discovered the same method independently.<sup>3</sup> The method proposed by Lanczos is, for symmetric positive-definite matrices, mathematically equivalent to the conjugate gradient method, but it was described for the general case of nonsymmetric matrices.

There is no doubt that the origin of this class of methods was deeply rooted in approximation theory and, in particular in orthogonal polynomials. The ideas behind "gradient methods" as this class of methods was referred to, are based on some kind of global minimization. For instance, for positive-definite symmetric A, the CG method minimizes the so-called A-norm:  $||x_i - x||_A^2 \equiv$  $(x_i - x, A(x_i - x))$ , for  $x_i$  that are in the Krylov subspace  $K_i(A, r_0) \equiv \{r_0, \dots A^{i-1}r_0\}$ . For some PDE problems this norm is known as the energy norm, which has physical relevance. Another interpretation of the gradient methods is that the residual is orthogonal to the space of previously generated residuals, or some related space. Both interpretations are useful for the formulation of methods as well as for the analysis. A very useful consequence from the Krylov subspace basis is that  $x_i$  can be expressed as a polynomial in A of degree i - 1, acting on  $r_0$ . The minimization interpretation makes it possible to bound the error for CG by replacing the "CG-polynomial" by easier to analyze polynomials, for instance a Chebyshev polynomial. This leads to the well-known upper bound [109,50,47,4]

$$\|x_{i} - x\|_{A} \leq 2\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^{i} \|x_{0} - x\|_{A}$$
(5)

for symmetric positive-definite matrices, in which  $\kappa = \lambda_{max}(A)/\lambda_{min}(A)$ . This upper bound describes well the convergence behavior for matrices A of which the eigenvalues are distributed rather homogeneously. For more uneven distributions one can obtain bounds by making more subtle choices for the approximating polynomials, for instance, products of suitably chosen Chebyshev polynomials [4]. These choices do not reveal the superlinear convergence that is often observed for CG and also for other Krylov subspace methods. The notion of superlinear convergence refers to the observation that the convergence of CG accelerates in the course of the process. Proofs for superlinear convergence had been given already in the early 1950s [111,99], but these did not reveal that the superlinear behavior may take place in early phases of the iteration process; they were qualitative rather than quantitative. Concus et al. [47] related this convergence behavior to the Krylov subspace approximations, by stating that "the extremal eigenvalues are approximated especially well (by the Ritz values corresponding to the Krylov subspace) as CG proceeds, the iteration then behaving as if the corresponding eigenvectors are not present (thus leading to a smaller "effective" condition number in (5), which might then explain the faster convergence". In 1986, this was proven in a quantitative

<sup>&</sup>lt;sup>3</sup> The anecdote told at the recent "Conference on preconditioning methods for sparse matrix problems in industrial applications" held in Minneapolis, by Emer. Prof. Marvin Stein, the post-doc who programmed the algorithm for M. Hestenes the first time, is that Stiefel was visiting UCLA at the occasion of a conference in 1951. Hestenes, then a faculty member at UCLA, offered to demonstrate this effective new method to Stiefel, in the evening after dinner. Stiefel was impressed by the algorithm. After seeing the deck of cards he discovered that this was the same method as the one he had developed independently in Zurich. Stiefel also had an assistant, by the name of Hochstrasser, who programmed the method.

way [179] and it was shown that the relevant eigenvalues needed to be approximated only in a modest degree by Ritz values, for an acceleration to set in.

As was mentioned in Section 2, the actual behavior of CG, in finite precision arithmetic, was initially not quite well understood. Several attempts to analyze this have been made, but it was in the early 1990s that this problem was satisfactorily explained. Greenbaum and Strakoš showed that CG in finite precision could be regarded as the exact process applied to an expanded system that is closely related to the given system, and the matrix of the expanded system has almost multiple exterior eigenvalues when the orthogonality in the process is lost. This helps explain why finite precision leads to a delay in the iteration process, but does not prevent the algorithm to deliver accurate approximate solutions (for symmetric positive-definite systems). For details, see [89,172], and [88, Chapter 4].

Surprisingly, it took some time before the ideas of the CG were generalized to other classes of matrices. Paige and Saunders [137], in 1975, showed that the underlying Lanczos method, could also be used for efficient implementations of related approaches for symmetric indefinite matrices. In MINRES, the norm of the residual  $||Ax_i - b||_2$  is minimized, and this required a more careful approach in order to avoid breakdowns. Conjugate gradient can be interpreted as a process in which A is projected to the Krylov subspace (in fact, the Lanczos approach), which leads to a tridiagonal matrix T of low dimension. The actual approximation  $x_i$  is determined by solving a small linear system with T, and this is done with LU without pivoting. This leads to the elegant short recurrences in CG. For indefinite matrices, the LU factorization could lead to numerical difficulties.

Paige and Saunders circumvented this by employing a QR decomposition of T, which leads again to regular short recurrences, making the MINRES method attractive because of its minimal overhead and its economy of storage. It may come as a small wonder that one can also minimize  $||x_i - x||_2$ , without knowledge of the solution x. Paige and Saunders accomplished this in their SYMMLQ, by restricting  $x_i$  to  $AK_i(A, r_0)$ . The advantage of SYMMLQ over MINRES appears to be in less sensitivity to ill-conditioning of A, the price one has to pay is that SYMMLQ often takes more steps. A slight disadvantage is also that although the method minimizes the norm of the error, the value of this norm is not known and the only practical information one has is the norm of the residual.

In 1976, Concus and Golub [45] and Widlund [200] came up with the idea of splitting a matrix into its symmetric and nonsymmetric parts and using the symmetric part as a preconditioner. With the proper inner product, the resulting algorithm corresponds to an iteration with a skew-Hermitian matrix – and therefore a three-term recurrence – CG-like – algorithm (called CGW) can be formulated.

The Bi-CG method, proposed in 1976 by Fletcher [76], is actually an implementation of the two-sided Lanczos process, which was suggested by Lanczos in 1952. In Bi-CG, the residual is constructed to be orthogonal to a Krylov subspace generated with  $A^{T}$  and some vector  $s_0$ . Initially, many numerical analysts were very skeptical of the Bi-CG, mainly because of the various breakdown situations that may occur. Also, Bi-CG did not minimize any norm of interest and the convergence behavior can be very irregular. The fact that the underlying two-sided Lanczos process works with nonorthogonal projections led to serious doubts on the usefulness of the method. A good example of this concern is in [201, pp. 394, 395], where the two-sided Lanczos method (viewed as a finite method for reducing a nonsymmetric matrix to tridiagonal form) is commented on: "… we may well have to pay a heavy price in terms of numerical instability…" and "… it is difficult to think of any reason why we should use Lanczos' method in preference of Householder's". Wilkinson's

analysis is, of course, still valid. However in the context of solving large sparse linear systems, we have learned to make the two-sided Lanczos method and the Bi-CG into useful solution techniques thanks to a number of enhancements and a better understanding of these processes.

#### 7. Krylov subspace methods: the second period

Of course, the success of Krylov methods for symmetric matrices has inspired the construction of similar methods for unsymmetric matrices. The classical Lanczos method leads to a tridiagonal system, the projection of A with respect to the Krylov subspace. Factorization of this tridiagonal matrix as the product of a lower and an upper bidiagonal matrix, leads to the coupled two-term recurrences as in Bi-CG. As said before, Bi-CG suffered from several breakdown conditions. One breakdown, associated with the indefiniteness of the implicitly generated projected tridiagonal system, can be cured by admitting  $2 \times 2$  blocks along the diagonal of one of the factors. This requires the combination of two successive iteration steps, which explains the name Composite step Bi-CG [12]. A similar idea had been used much earlier by Luenberger, in order to make the conjugate gradient algorithm robust for symmetric-indefinite matrices [119]. The other breakdown, a more serious one, arises when the bi-orthogonalization process leads to a zero inner product of the two new vectors in the Krylov subspace and its adjoint space (that is the Krylov subspace, generated in the two-sided Lanczos process, with  $A^{T}$ ). Likewise, a near breakdown should also be avoided since it may lead to inaccuracies in the approximated solution. This breakdown can be cured with a look-ahead strategy, first proposed by Parlett et al. [140]. The idea is to expand the Krylov subspaces by two vectors simultaneously, and to make the new vectors block bi-orthogonal with respect to the similarly expanded adjoint space. Parlett et al. considered only look-aheads of length two, but a few years later, around 1990, the idea was picked up almost simultaneously by a number of other researchers who generalized it to look-aheads of arbitrary length. The most well known of these approaches were those published by Gutknecht and co-authors [95], Joubert [105], Parlett [139], Freund and Nachtigal [80], and Brezinski and co-authors [31,32]. In the latter work, the look-ahead strategy was related to the theory of orthogonal polynomials and referred to as recursive zoom technique. The connection between orthogonal polynomials and the Lanczos algorithms (and also the  $\varepsilon$ -algorithm) is discussed in [28]. This has proved to be very useful for getting more insight in the Lanczos and two-sided Lanczos algorithms. It also has helped to construct breakdown free variants of the hybrid Bi-CG algorithms, for details on this see [29].

Curing the breakdowns in Bi-CG was important, but there were other aspects as well that motivated further research. The convergence behavior of Bi-CG is usually quite irregular, in the sense that the norms of successive residuals can behave erratically. This motivated Freund and Nachtigal [80] to propose an algorithm in which the projected overdetermined tridiagonal system is solved in a least-squares sense. Since the basis vectors for the Krylov subspace, generated by the two-sided Lanczos process, are in general not orthogonal, this approach does not lead to a minimum residual approximate solution (as with MINRES), and this inspired for the name quasi-minimum residual (QMR). The full QMR method includes a look-ahead strategy, but it became also popular without it, since the first breakdown condition is cured by the least-squares solution of the tridiagonal system. For a template for this simplified QMR, see [13].

The other clear disadvantage in the basic two-sided Lanczos method was the necessity to construct two Krylov subspaces:  $K^{i}(A; r_{0})$ , and  $K^{i}(A^{T}; s_{0})$ . Of the two vector bases generated only one is exploited for the solution, the other can be regarded as an auxiliary set used to generate the inner products needed to generate the bi-orthogonal basis. Sonneveld [166] made the clever observation that the operations with  $A^{\rm T}$  could be reformulated to operations with A, and these operations can be used for a further reduction of the residual norm. Whereas the previous approaches, look-ahead, composite-step, and QMR, help to cure deficiencies of the Lanczos method, they do not lead to essential faster convergence. However, Sonneveld's trick can lead to much faster convergence, for practically the same costs per iteration. The idea is based on the observation that the residual  $r_i$ can be expressed formally as  $r_i = p_i(A)r_0$ , with  $p_i$  a polynomial of degree *i*. Likewise, the shadow residual in the adjoint space can be expressed as  $s_i = p_i(A^T)s_0$ . The iteration coefficients for Bi-CG are computed from inner-products such as  $(r_i, s_i)$ , and such an inner product can be rewritten formally, as  $(p_i(A)r_0, p_i(A^T)s_0) = (p_i^2(A)r_0, s_0)$ . This observation leads to an algorithm that generates  $\tilde{x}_i \in$  $K^{2i}(A; r_0)$ , for which the corresponding residual  $\tilde{r}_i$  can be expressed as  $\tilde{r}_i = p_i^2(A)r_0$ . For the situation where Bi-CG delivers a residual  $r_i$  (= $p_i(A)r_0$ ) that is small compared with  $r_0$ , one may conclude that  $p_i(A)$  has transformed  $r_0$  into a small vector, and hopefully, if we apply  $p_i(A)$  twice, then this leads to a double reduction. Indeed, the resulting method, conjugate gradients squared (CGS), often leads to a convergence about twice as fast as Bi-CG. This algorithm is also referred to as Bi-CGS, which is actually more appropriate. The downside of the squaring of  $p_i$  is that the convergence of CGS is usually much more irregular than for Bi-CG. This can lead to serious accuracy problems in finite precision arithmetic; we will come back to this aspect later.

Soon after the discovery of the CGS method, it was recognized that the operations with  $A^{T}$  could also be transformed to other polynomials in A. The first idea in this direction was Bi-CGSTAB [186], in which Bi-CG was combined with minimum residual steps of degree one. This led to a convergence that is rather smooth as well as faster than Bi-CG and it gave rise to many other hybrids. Gutknecht suggested to combine 2i Bi-CG steps with *i* times a minimum residual method of degree 2. This was generalized by Sleijpen and Fokkema [165] to Bi-CGSTAB( $\ell$ ). The same principles can also be applied to QMR, and the analogue of CGS led to TFQMR [79]. The analogue of Bi-CGSTAB is QMRSTAB, suggested by Chan et al. [38]. Zhang [207] describes more general product methods based on Bi-CG. His framework includes the previously described methods, but also admits hybrid variants in which one can shift from CGS to Bi-CGSTAB at some iteration step. This principle admits further possibilities for reducing the residual in some norm. An interesting variant of CGS has been suggested by Fokkema et al. [77]. Here, the polynomial  $p_i^2$  that generates the residuals, is replaced by the product  $p_i \tilde{p}_i$ , where  $\tilde{p}_i$  is the polynomial that corresponds to a 'nearby' Bi-CG process. The principle can be used to help reduce severe irregularities in the convergence, while the quadratically reduced errors in important eigenvector directions are still realized. According to the authors, this is an advantage in the context of iterative solutions of Jacobian systems in a Newton process for nonlinear systems of equations. Similar ideas were also considered by Brezinski and Redivo Zaglia [30]. Their approach is to compute approximations by two different methods and to combine the two results in an effort to get a better approximation. For some methods, the combined method can be executed at reduced costs, that is some of the matrix vector products can be used for both methods. For a detailed overview of this approach see [29, Chapter 5].

A different direction is to try to minimize the norm of the residual over all vectors in the Krylov subspace, similar to the MINRES approach for symmetric A. A number of methods were

proposed that achieved this goal, among them ORTHODIR [103], GCR [62], ORTHOMIN [190], and Axelsson's method [5], but many of these methods suffered from some numerical instability. An early specimen of this approach was suggested by Khabaza in 1963 [112]. He proposed a method which, in exact arithmetic, leads to the same iterands as GMRES(m). However, he used the defining vectors  $r_0, Ar_0, \ldots, A^{m-1}r_0$  as the basis vectors, without further orthogonalization. The observation that Khabaza considered matrices with a condition number of 200 as being ill-conditioned, may serve as an illustration for the numerical problems that were encountered.

GMRES [158] developed in 1986, was mathematically equivalent to these techniques but soon it came to be preferred because of its better numerical behavior and its lower cost, both in terms of memory and arithmetic. An advantage of GMRES is its guarantee to compute the approximate solution with minimum residual norm, but the price to be paid is that the overhead costs per iteration, that is the computations other than the matrix vector product, increase linearly with the iteration count. Also, all basis vectors for the Krylov subspace have to be stored. The obvious solution seems to be to restart after a convenient number of iterations, before the costs for computation and storage become prohibitive. This is known as restarted GMRES, or simply GMRES(m). The disadvantage of this approach is that it decreases the robustness of the method since convergence is no longer guaranteed. Moreover, by restarting the convergence behavior may become very slow and one runs the risk to miss the faster convergence that might have occurred for a larger value of *m*. For that reason, researchers have tried to find ways to reduce the number of iteration steps, other than by preconditioning, or even in addition to preconditioning. One idea is to try to improve the preconditioner with updates from the Krylov subspace. This has been suggested first by Eirola and Nevanlinna [61]. Their approach leads to iterative methods that are related to Broyden's method [35], which is a Newton-type method. For specific but obvious choices, one recovers a method that is equivalent to GMRES. The Broyden method can be obtained from this update-approach if we do not restrict ourselves to Krylov subspaces. See [191] for a discussion on the relation of these methods.

The idea of preconditioning is to approximate  $A^{-1}p$  for vectors p generated by the iteration method. One could do this in a different way for every iteration step, for instance, by incorporating information from previous iteration steps in the preconditioner, or by approximating  $A^{-1}p$  by some iteration method again. The updated preconditioners cannot be applied immediately to GMRES, since the preconditioned operator now changes from step to step, and we are not forming a regular Krylov subspace. However, we can still minimize the residual over the new subspace. The idea of variable preconditioning has been exploited in this sense, by different authors. Axelsson and Vassilevski [11] have proposed a Generalized Conjugate Gradient method with variable preconditioning, Saad [155] has proposed a scheme very similar to GMRES, called Flexible GMRES (FGMRES), and Van der Vorst and Vuik [187] have published a scheme called GMRESR. FGMRES has received more attention, possibly because it is fairly easy to implement: only the update directions in GMRES have to be preconditioned, and each update may be preconditioned differently. This means that only one line in the GMRES algorithm has to be adapted. The price to be paid is that the method is no longer robust; it may break down. GMRESR and the generalized conjugate gradient method produce, in exact arithmetic, the same results, but GMRESR is numerically more stable and more efficient. In GMRESR the residual vectors are preconditioned and if this gives a further reduction then GMRESR does not breakdown. This gives slightly more control over the method in comparison with FGMRES. In most cases though the results are about the same.

Other methods that proved to be very useful include the LSQR method, suggested in 1982 by Paige and Saunders [138]. LSQR is a clever implementation of the Lanczos method that leads to a factorization of the tridiagonal reduced matrix for  $A^TA$ . This is often the method of choice for overdetermined or underdetermined systems; it minimizes the norm of the residual over the Krylov subspace generated with  $A^TA$ . For square systems the method is not so effective, unless one finds a good preconditioner, since the convergence behavior of LSQR is dictated by the square of the condition number of the system involved. The condition number, however, does not always give a good indication for the behavior of Krylov methods; Nachtigal et al. describe examples of matrices for which the singular values may predict the convergence behavior (much) better [129]. In extreme cases, GMRES with A may converge much slower than LSQR, or in fact, any method based on the normal equations. LSQR may also be viewed, in exact arithmetic, as CG applied to the normal equations. Applying CG in this fashion, however, will result in poorer stability. Craig's method [138] is a Krylov method which also works with  $A^TA$  and in which the error  $||x_i - x||_2$  is minimized, without computing the value of this norm.

Finally, we mention that Weiss, in the early 1990s, gave generalizations of SYMMLQ for unsymmetric matrices. These methods are known as GMERR methods [196,198]. It may have an advantage to have a method in which the norm of the error is minimized, but since this is done over a different subspace, namely  $A^{T}K_{i}(A^{T}, r_{0})$ , it is not clear yet when this leads to advantages over, for example, GMRES, in terms of efficiency or stability.

Convergence results for Krylov methods in the non-Hermitian case were established following essentially similar ideas as for the CG algorithm, see [157, Chapter 6.11]; [88, Chapter 3] for overviews. However, this simple analysis which was given, for example, for GMRES [158] was soon viewed as insufficient. The traditional bounds on the norm of the error or the residual are expressed in terms of eigenvalues of A and the condition number of the eigenvector matrix. For highly nonnormal matrices this does not always lead to informative results. Embree, in his thesis [67], describes situations for which the field of values of A, or its pseudospectra, are used for understanding the observed convergence behavior of GMRES (see also [88, Chapter 3]).

Many attempts to get a better understanding of the behavior of GMRES were made. Work by Brown [34] and later by Cullum and Greenbaum [49] established relations between certain methods (GMRES and FOM, and then BiCG and QMR). Greenbaum and Strakos [90] showed a number of interesting properties of GMRES – in particular they characterize all linear systems that have the same GMRES convergence behavior. In a later paper, Greenbaum et al. [91] established that essentially any convergence behavior is possible for the same spectrum.

One fundamental question that was asked in the early 1980s was whether a short recurrence iterative process could be found that was also optimal for non-Hermitian matrices. Indeed, it was known how to generate short-term recurrence algorithms for nonsymmetric matrices (e.g. the Bi-CG) but these do not verify obvious optimality properties. On the other hand the optimal processes that were known required long recurrences (e.g., GMRES). The answer to this question was given by the excellent paper by Faber and Manteuffel in 1984, and alas is was a negative one [71]: short-term solution algorithms that are also optimal can essentially be devised only for a restricted class of matrices; for all practical purposes, these matrices are either hermitian or skew-hermitian. An essentially equivalent result had been published by Voevodin, just one year before Faber and Manteuffel [180].

For the Bi-CG and related methods, very little theory is available on convergence. An attempt on this was to introduce variable metrics, see [14] and the survey by Weiss [197].

#### 8. Accelerators are not enough: preconditioning methods

The convergence of iterative methods depends on spectral properties of the matrix of the linear system and in order to improve these properties one often transforms the linear system by a suitable linear transformation. This process is known as preconditioning.

We do not know for sure who coined the term 'preconditioning' first - it may have been Turing (according to Golub and O'Leary [86]) or Forsythe (see below). Regardless, the idea was known quite early on. Cesari, in 1937 [37], proposed to *multiply* a given system Ax = b with a polynomial P(A), in an attempt to speed up the convergence of the Richardson iteration (see also [23, p. 156]) for a discussion on this; in this reference the Richardson process is referred to as Mises' iteration -Cesari calls it the von Mises' iteration). In the 1952 paper by Lanczos [116] the notion of polynomial preconditioning is clearly defined: "The construction of the inverse matrix is equivalent to a linear transformation which transforms the given matrix into the unit matrix. The unit matrix can be conceived as the extreme case of a well-conditioned matrix whose eigenvalues are all 1. We will ask for much less if we merely demand the transformation of the original system whose dispersion is moderate". Lanczos then states that the goal of the procedure is to "reduce the initial skewness" of the system, not bring about the exact solution. Forsythe in his report on this paper in the mathematical reviews (review MR 16-751, 1955) does employ the term "preconditioning" explicitly. Polynomial preconditioning is also clearly mentioned in the review paper in Stiefel [170] - in 1959. Hestenes in 1956 [100], viewed the conjugate gradient method as an acceleration technique for suitable matrix splittings. His formulation of the algorithm is equivalent with preconditioned conjugate gradients. Finally, we mention Faddeev and Faddeeva, who used the term "preparing" in their 1963 book [72, p. 188] (a translation of the Russian text of 1960) for transforming a system Ax = b to KAx = Kb, with K such that KA is close to a unit matrix.

Modern preconditioning methods started in the late 1960s and early 1970s. Evans [70] used the term *preconditioning* explicitly for the acceleration of SSOR by the Chebyshev iteration. However, this combination had already been studied by Sheldon in 1955 [161]. In 1972, Axelsson [3] proposed to use the SSOR method as a preconditioner for the conjugate gradient method. The incomplete Cholesky decompositions (Meijerink and van der Vorst, 1974, 1977), became quite popular and led to the ICCG process [125]. Concus, Golub and O'Leary [47] wrote an influential paper on the usage and effect of preconditioning for the CG method.

#### 8.1. Incomplete factorizations

Preconditioning as we know it today refers mostly to approximate or incomplete factorizations of the coefficient matrix. Some of the early publications on such factorizations that are often cited include Buleev [36], Varga [189] and Oliphant [132]. Later in the 1960s a few other procedures were developed specifically for matrices arising from finite difference approximations to elliptic operators, these include the work by Dupont et al. [60]. In 1977, Meijerink and Van der Vorst introduced the more general incomplete LU factorization [125]. The paper suggests that the combination of this "preconditioning" and the conjugate gradient method could lead to a robust and very fast combination.

Similar ideas were in existence before. However, the paper provided a much needed stimulus to the whole area of iterative methods.

The Dupont-Kendall-Rachford splitting can be viewed as an incomplete LU factorization with zero fill-in, in which the elimination errors are compensated by corrections to the diagonal of the decomposition. In 1977, this procedure was generalized by Gustafsson [94] in 1978, as a modified form of the incomplete LU factorizations: MILU.

Several developments marked the years that followed. Two distinct ways of developing incomplete factorization preconditioners with improved accuracy were developed. The first approach is based on a symbolic factorization view, i.e., it only requires the nonzero structure of the matrix to determine which fill-ins to drop. A method proposed by Watts [195] for irregular sparse matrices attributes a "level of fill" recursively to each fill-in element from the levels of fill-in of its parents, in the Gaussian elimination process. Then each fill-in that is introduced and whose level exceeds a certain threshold is dropped. In practice for M-matrices, the higher the fill-in the lower the level. The second common approach is to modify a given direct solver by including a dropping rule, based on the numerical size of the fill-ins introduced [82,134,53,52,208,206].

Although the relation between the size of the dropped elements and the number of iterations required to achieve convergence is far from being understood, *on the average* dropping small elements is more likely to produce a better quality preconditioner than dropping large elements. However, experience reveals that this is not always true. Another drawback of the level-of-fill approach is that it is difficult to predict the amount of fill-in that will be generated.

The number of variations that can be found on incomplete factorization preconditioner is truly astounding and we will not attempt to list them all. It suffices to say that there were variants developed for specific architectures. (e.g., Twisted Factorizations), or for specific applications (e.g., element-by-element preconditioners), or to exploit specific features of the equations (e.g., block factorizations), among other classes. See [57] for an overview of these preconditioners, specially in view of their implementation for high-speed computers.

One of the interesting recurring themes in preconditioning methods is whether or not reordering the matrix prior to applying the ILU factorization can be helpful. Two early papers examined this carefully and concluded rather negatively. The first is a paper by Simon [163] who considered large nonsymmetric linear systems. For the systems he considered he concluded that standard techniques used for sparse direct solvers were not too helpful for use in preconditioners based on level-of-fill. Immediately following this was a paper by Duff and Meurant [59] which concluded, similarly, that ICCG does not in general benefit in any significant manner form reordering. These studies were limited to certain types of reorderings and certain types of preconditioners. It is now known [19] that certain reorderings, such as Reverse Cuthill McKee are beneficial in preconditioning methods, in particular with some form of dropping strategy. The beneficial impact of well-chosen fill-ins was already demonstrated in [59] for some orderings. What seems to be also clear is that the best approaches for direct solvers.

Since ILU and IC factorizations were the most popular preconditioners, at least in a sequential environment, many attempts have been made to improve them, for instance by including more fill [126], by modifying the diagonal of the ILU factorization in order to force rowsum constraints [94,8,7,131,181,64], or by changing the ordering of the matrix [183,184]. A set of experiments with respect to the effects of ordering is contained in [59].

Saad [153] proposed a few variants on the incomplete LU approach for the matrix A, one of which is in fact an incomplete LQ decomposition. In this approach it is not necessary to form the matrix Q explicitly, and it turns out that the lower triangular matrix L can be viewed as the factor of an incomplete Cholesky factorization of the matrix  $A^{T}A$ . This can be exploited in the preconditioning step, avoiding the use of Q. The second approach was to introduce partial pivoting in ILU, which appears to have some advantages for convection-dominated problems. This approach was further improved by including a threshold technique for fill-in as is done in the ILUT algorithm, see [157, p. 287].

Another major step forward, for important classes of problems, was the introduction of block variants of incomplete factorizations [176,46,6], and modified variants of them [46,6,120]. It was observed, by Meurant, that these block variants were more successful for discretized two-dimensional problems than for three-dimensional problems, unless the two-dimensional' blocks in the latter case were solved accurately. For discussions and analysis on ordering strategies, in relation to modified (block) incomplete factorizations, see [127,121].

#### 8.2. Parallel preconditioners

Parallel preconditioners were discussed as early as with the first appearance of vector and parallel computers. It soon became apparent that the standard ILU-based preconditioners which were just becoming quite popular, were also very sequential in nature and had either to be replaced or implemented differently. The first ideas that were promoted or discussed were based on approximating the LU-solves by means of Neuman expansions in the L and U solves [182] as well as from the start by approximating the inverse of the original matrix by the Neuman series expansion of its inverse [58]. This gave rise to a number of papers on "polynomial preconditioners". The survey paper [154] gives an account of the state of the art toward the end of the 1980s and it can be seen that polynomial preconditioners figured prominently in the article. Another approach – termed "level-scheduling" or "wavefront" approach, was to unravel parallelism from the forward and backward solves. Because of sparsity, many equations can be solved at the same time in several levels during the forward and the backward solves – and a technique known in graph theory as "topological sorting" allows to determine these levels [1,15,20,159,184,185].

However, these two methods were soon viewed as having a limited potential. Level scheduling has limited parallelism and the first and last (smallest) levels were small enough to cause bottlenecks. A number of strategies could be used to improve the situation however. Polynomial preconditioners faced more serious challenges. Their performance relative to existing alternatives was not too convincing, especially for small number of processors. In addition, it is difficult to find a good polynomial in the case of indefinite matrices. Current interest in these techniques has all but vanished. This is a case where good mathematically based methods are not enough to overcome an inherent limitation of a given approach.

Red-black ordering is an obvious approach to improve parallel properties for well-structured problems, but experimental results were disappointing [59] so it was avoided. If carefully done though, they can lead to significant gains in efficiency. Elman and Golub [65] suggested such an approach, in which Red-Black ordering was combined with a reduced system technique. The idea is simply to eliminate the red points, and construct an ILU for the reduced system of black points. Recently, DeLong and Ortega [56] and Saad [156] suggested carrying out a few steps of red-black ordered SOR as a preconditioner for GMRES and Bi-CGSTAB. The key to success in these cases seems to be a combined effect of fast convergence of SOR for red-black ordering, and the ability of the Krylov subspace to remove stagnations in convergence behavior associated with a few isolated eigenvalues of the preconditioned matrix.

Another stream of ideas for deriving parallel preconditioning methods came from domain decomposition-type methods. Such methods were in existence in the partial differential equations (PDE) literature already in a different form, see, e.g., the survey paper [40]. Though domain decomposition methods were motivated by parallel computing it appeared that the approach could be used with success also for the construction of sequential preconditioners. Domain decomposition has been used for problems that arise from discretization of a PDE over a given domain. The idea is to split the given domain into subdomains, and to solve the discretized PDEs over each subdomain separately. The main problem is to find proper boundary conditions along the interior boundaries of the subdomains. Domain decomposition is used in an iterative fashion and usually the interior boundary conditions are based upon information on the approximate solution of neighboring subdomains that is available from a previous iteration step.

It was shown by Chan and Goovaerts [39] that domain decomposition can actually lead to improved convergence rates, provided the number of domains is not too large. A splitting of the matrix with overlapping sub-blocks along the diagonal, which can be viewed as a splitting of the domain, if the matrix is associated with a discretized PDE and has been ordered properly, was suggested by Radicati and Robert [143]. They suggested to construct incomplete factorizations for the sub-blocks. These sub-blocks are then applied to corresponding parts of the vectors involved, and some averaging was applied on the overlapping parts. A more sophisticated domain-oriented splitting was suggested in [194], for SSOR and MILU decompositions, with a special treatment for unknowns associated with interfaces between the sub-domains.

The isolation of sub-blocks was done by Tang [175] in such a way that the sub-blocks correspond to subdomains with proper internal boundary conditions. In this case it is necessary to modify the sub-blocks of the original matrix such that the sub-blocks could be interpreted as the discretizations for subdomains with Dirichlet and Neumann boundary conditions in order to force some smoothness of the approximated solution across boundaries. In [173] this was further improved by requiring also continuity of cross-derivatives of the approximate solution across boundaries. The local fine tuning of the resulting interpolation formulae for the discretizations was carried out by local Fourier analysis. It was shown that this approach could lead to impressive reductions in numbers of iterations for convection dominated problems.

Note that domain decomposition methods for general sparse linear systems became successful at the same time as the machines for which they were designed (distributed memory, MIMD computers) were gaining importance. Currently, most of the parallel iterative solvers packages utilize essentially DD-type preconditioners.

For an overview of parallel preconditioners, and guidelines for their efficient implementation, see [57].

#### 8.3. Multilevel preconditioners

Methods based on multilevel techniques, such as multigrid, have been popular for solving certain types of PDEs [96]. They are often designed specifically for problems arising from PDEs with regular

meshes. Algebraic multilevel solution methods were developed as an attempt to extend the scope of these methods [149]. Clearly, nothing can prevent the use of these techniques as preconditioners for Krylov solvers. Since the multigrid method is viewed as optimal, its users have often avoided to use an accelerator such as GMRES or BICGSTAB to accelerate it. A study by Oosterlee and Washio [133] did, however, indicate that such a combination could be beneficial and lead to a much more robust solver than a multigrid solver alone.

Recently, a class of preconditioners that tended to close the gap between multilevel methods and preconditioned Krylov methods drew much attention. It was discovered that a multigrid-inspired ordering can be very effective for discretized diffusion–convection equations, leading in some cases to almost grid-independent speeds of convergence [177,178], see also [52]. These preconditioners can be viewed also from the angle of ILU factorization combined with a reordering as in the ILUM strategy, see [157, p. 371]. This type of approach can be fairly robust and scale well with problem size, unlike other ILU preconditioners.

In earlier related work, Axelsson and Vassilevski developed a method which was later referred to as AMLI [9,10] that is based on a set of nested finite element grids. The equations associated with the finer mesh are reduced (approximately) and the process is repeated to a number of levels until the coarsest mesh is reached.

It is interesting to note that currently, this general approach offers an excellent potential for providing a global method that can encompass most of the successful approaches for solving linear systems. By restricting the number of levels to one and performing the factorization accurately, one obtains a direct solver. A standard ILU solver can also be obtained by dropping fill-in.

#### 8.4. Sparse approximate inverses

Many researchers and practitioners became aware of an important and damaging phenomenon in ILU techniques. An ILU factorization can be an accurate approximation to the original matrix but it can yield a very ill-conditioned factorization [181]. This phenomenon of instability of the LU factors was analyzed in particular by Elman [63]. This weakness of ILU factorizations, coupled with their sequential nature, spurred researchers to consider radical alternatives. The approximate inverse methods which were first proposed in the late 1970s [16] were in this category. It is only with the advent of massive parallel processing that such methods were considered as serious contenders of the now standard ILU methods [93]. A flurry of publications followed this work and the work by Kolotilina and Yeremin [113,114]. To cite just a few, [48,92] define strategies for determining the best pattern for the inverse, [43,42,18,17] define alternative schemes. While at the beginning, these preconditioning methods were received with much skepticism, it is fair to say that substantial progress has been made and a number of recent papers reported that approximate inverse schemes can often be competitive with ILU factorization methods – even in a sequential environment.

One idea for constructing an approximate inverse is to find a sparse matrix M such that ||AM - I|| is small for some convenient norm. Kolotilina and Yeremin [114] presented an algorithm in which the inverse was delivered in factored form, which has the advantage that singularity of M is avoided. In [48] an algorithm is presented which uses the 1-norm for the minimization. We also mention Chow and Saad [43], who use GMRES for the minimization of  $||AM - I||_F$ . Drop-tolerance strategies are applied to limit the amount of fill-in allowed. The approach can also be used to correct explicitly some given implicit approximation, such as a given ILU decomposition.

An elegant approach was suggested by Grote and Huckle [92]. They also attempt to minimize the F-norm, which is equivalent to the Euclidean norm for the errors in the columns  $m_i$  of M

$$||AM - I||_F^2 = \sum_{i=1}^n ||Am_i - e_i||_2^2.$$

Based on this observation they derive an algorithm that produces the sparsity pattern for the most error-reducing elements of M. This is done in steps, starting with a diagonal approximation, each steps adds more nonzero entries to M, and the procedure is stopped when the norms are small enough or when memory requirements are violated.

#### 9. Multigrid methods

As was mentioned above, among the earliest preconditioning methods were the simple relaxation schemes since these have, historically, been quite popular techniques. Thus Krylov subspace methods were viewed as methods for accelerating such techniques. Another powerful way of accelerating relaxation techniques is to use multigrid - or multilevel methods. Although we have given little emphasis to these methods in this survey, they are nevertheless important methods which can give rise to very efficient solvers, actually of optimal complexity in some cases. The main observation of multigrid techniques is based on a Fourier analysis of the residual (or error) vector of a sequence of iterates that are generated by a scheme such as Jacobi or Gauss-Seidel. This means that these residual vectors are analyzed in the eigen-basis associated with the iteration matrix M – assuming that Mhas a complete set of eigenvectors. In the case of Jacobi, the observation is that the components associated with the largest eigenvalues (in the original matrix) will decrease rapidly. However, those associated with the smallest eigenvalues will converge much more slowly. As a result after a few steps, the "high-frequency" components may have converged while the "low-frequency" components may have made very little progress in comparison. To correct this situation, researchers developed methods that used several grids. The simplest idea is to use two meshes one fine and one that is coarser, where the fine mesh can be viewed as the result of refining the coarse one. The iteration initially takes place on the fine mesh. After a few steps, the residual is *projected* onto the coarse mesh, by some form of restriction. Let  $A_{2h}$  be the matrix for the problem on the coarse mesh and  $r_{2h}$  this projected residual. The system  $A_{2h}\delta = r_{2h}$  is then solved on the coarse mesh by means of a few steps of relaxation. This is called a correction step. The vector  $\delta$  is then extrapolated into the finer mesh and the result is added as a correction to the iterate on the fine mesh.

An early paper describing essentially such an idea can be traced back to 1935, when Southwell [167] discusses a "group-relaxation" scheme for a two-level setting. It is clear that we do not have to stop at two levels of meshes. Much later Fedorenko [73] described the first true multigrid technique – which employs more than two grids. The idea laid dormant for some time until Achi Brandt published a series of articles, the first of which in 1972 [26]. The paper [27] provided the needed analysis to boost this class of techniques. Many variants of multigrid methods have been developed and the literature is perhaps richer than that of Krylov subspace methods. The excellent "frontiers in applied mathematics" [124] volume published in 1987 listed already 607 references. A number of excellent books have been written on multigrid. For a quick tutorial see [33]. More complete texts

include Hackbusch [96], and Wesseling [199]. The volume [124] contains an excellent collection of articles on multigrid and algebraic multigrid.

It is often asked what is the best method to use: preconditioned Krylov subspace methods or a multigrid approach? Users of iterative techniques are really split in two camps: those who use exclusively multigrid methods and those who use exclusively (preconditioned) Krylov subspace methods. Combination of the two methods have been advocated however, see Section 8.3, but this is an exception rather than a rule. When multigrid techniques work, they can be extremely efficient – far more so than preconditioned Krylov methods. However, their efficiency relies essentially on the inter-level restriction, and prolongation operators, the choice of which will vary from one application to the next. Such efficiencies can be achieved for regular meshes and for smooth elliptic PDEs. Standard multigrid methods cannot be applied without the existence of an underlying mesh – hence its major limitation.

This led to the development of algebraic multi-grid (AMG) initiated by Ruge and Stuben [149]. AMG was defined for algebraic systems – in the same manner as general sparse linear systems solvers – by defining restriction and prolongation operators algebraically. The overall success of AMG, which is derived based on an underlying PDE problem, has been somewhat limited.

#### 10. Outlook

It is rather difficult to predict what the future will bring in the area of iterative methods. However, it is almost certain that the usage of these methods will increase substantially in the application areas. This is partly due to the impact of parallel architectures. Direct methods are more complex to implement in parallel than are iterative methods. Also it is clear that problem sizes are increasing to the point of making direct solvers exceedingly expensive – both in terms of memory and arithmetic costs. One ray of hope for those problems that are hard to solve by iterative techniques, is to combine techniques from direct and iterative solution technologies. As the communities from direct and iterative solvers are getting to learn each other's tricks, the distinction between the two methodologies is getting to be blurred and this results in better, more robust, methods. Indeed, if memory is the only difficulty with direct solvers, it may be possible to find preconditioners that are far more accurate than current ones – but which use moderate amounts of memory.

#### Acknowledgements

We wish to thank our colleages Michele Benzi, Claude Brezinski, Mark Embree, Gerard Meurant, Nick Trefethen, and Gerard Sleijpen, for their careful reading of the text and for adding useful information. Michele Benzi brought the early (and not very well-known) paper of Cesari, on polynomial preconditioning, to our attention.

#### References

 E.C. Anderson, Y. Saad, Solving sparse triangular systems on parallel computers, Internat. J. High Speed Comput. 1 (1989) 73–96.

- [2] W.E. Arnoldi, The principle of minimized iteration in the solution of the matrix eigenvalue problem, Quart. Appl. Math. 9 (1951) 17–29.
- [3] O. Axelsson, A generalized SSOR method, BIT 12 (1972) 443-467.
- [4] O. Axelsson, Solution of linear systems of equations: iterative methods, in: V.A. Barker (Ed.), Sparse Matrix Techniques, Springer, Berlin, 1977, Copenhagen, 1976, pp. 1–51.
- [5] O. Axelsson, Conjugate gradient type-methods for unsymmetric and inconsistent systems of linear equations, Linear Algebra Appl. 29 (1980) 1–16.
- [6] O. Axelsson, A general incomplete block-factorization method, Linear Algebra Appl. 74 (1986) 179-190.
- [7] O. Axelsson, Iterative Solution Methods, Cambridge University Press, New York, 1994.
- [8] O. Axelsson, G. Lindskog, On the eigenvalue distribution of a class of preconditioning methods, Numer. Math. 48 (1986) 479–498.
- [9] O. Axelsson, P. Vassilevski, Algebraic multilevel preconditioning methods, I, Numer. Math. 56 (1989) 157-177.
- [10] O. Axelsson, P. Vassilevski, Algebraic multilevel preconditioning methods, II, SIAM J. Numer. Anal. 27 (6) (1990) 1569–1590.
- [11] O. Axelsson, P.S. Vassilevski, A black box generalized conjugate gradient solver with inner iterations and variable-step preconditioning, SIAM J. Matrix Anal. Appl. 12 (4) (1991) 625–644.
- [12] R.E. Bank, T.F. Chan, An analysis of the composite step biconjugate gradient method, Numer. Math. 66 (1993) 295–319.
- [13] R. Barrett, M. Berry, T.F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H. van der Vorst, Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, SIAM, Philadelphia, PA, 1994.
- [14] T. Barth, T. Manteuffel, Variable metric conjugate gradient methods, in: Advances in Numerical Methods for Large Sparse Sets of Linear Equations, Number 10, Matrix Analysis and Parallel Computing, PCG 94, Keio University, Yokohama, Japan, 1994, pp. 165–188.
- [15] D. Baxter, J. Saltz, M.H. Schultz, S.C. Eisenstat, K. Crowley, An experimental study of methods for parallel preconditioned Krylov methods, in: Proceedings of the 1988 Hypercube Multiprocessors Conference, Pasadena, CA, January 1988, pp. 1698–1711.
- [16] M.W. Benson, P.O. Frederickson, Iterative solution of large sparse linear systems arising in certain multidimensional approximation problems, Utilitas Math. 22 (1982) 127–140.
- [17] M. Benzi, J. Marín, M. Tůma, A two-level parallel preconditioner based on sparse approximate inverses, in: D.R. Kincaid, A.C. Elster (Eds.), Iterative Methods in Scientific Computation, II, IMACS, New Brunswick, NJ, 1999, pp. 167–178.
- [18] M. Benzi, C.D. Meyer, M. Tůma, A sparse approximate inverse preconditioner for the conjugate gradient method, SIAM J. Sci. Comput. 17 (1996) 1135–1149.
- [19] M. Benzi, D.B. Szyld, A. van Duin, Orderings for incomplete factorization preconditioning of nonsymmetric problems, SIAM J. Sci. Comput. 20 (1999) 1652–1670.
- [20] H. Berryman, J. Saltz, W. Gropp, R. Mirchandaney, Krylov methods preconditioned with incompletely factored matrices on the CM-2, J. Partial Distrib. Comput. 8 (1990) 186–190.
- [21] G. Birkhoff, Solving elliptic problems: 1930–1980, in: M.H. Schultz (Ed.), Elliptic Problem Solver, Academic Press, New York, 1981, pp. 17–38.
- [22] A. Björck, T. Elfving, Accelerated projection methods for computing pseudo-inverse solutions of systems of linear equations, BIT 19 (1979) 145–163.
- [23] E. Bodewig, Matrix Calculus, North-Holland, Amsterdam, 1956.
- [24] E. Bodewig, Matrix Calculus (second revised and enlarged edition), North-Holland, Amsterdam, 1959.
- [25] R. Bramley, A. Sameh, Row projection methods for large nonsymmetric linear systems, SIAM J. Sci. Statist. Comput. 13 (1992) 168–193.
- [26] A. Brandt, Multi-level adaptive technique (MLAT) for fast numerical solutions to boundary problems, in: H. Cabannes, R. Temam (Eds.), Proceedings of the Third International Conference on Numerical Methods in Fluid Mechanics, Paris, 1972, Springer, Berlin, 1973, pp. 82–89.
- [27] A. Brandt, Multi-level adaptive solutions to boundary value problems, Math. Comp. 31 (1977) 333–390.
- [28] C. Brezinski, Padé-Type Approximation and General Orthogonal Polynomials, Birkhäuser, Basel, 1980.
- [29] C. Brezinski, Projection Methods for Systems of Equations, North-Holland, Amsterdam, 1997.

- [30] C. Brezinski, M. Redivo-Zaglia, Look-ahead in bicgstab and other product methods for linear systems, BIT 35 (1995) 169–201.
- [31] C. Brezinski, M. Redivo-Zaglia, H. Sadok, Avoiding breakdown and near-breakdown in Lanczos type algorithms, Numer. Algorithms 1 (1991) 261–284.
- [32] C. Brezinski, M. Redivo-Zaglia, H. Sadok, A breakdown-free Lanczos' type algorithm for solving linear systems, Numer. Math. 63 (1992) 29–38.
- [33] W.L. Briggs, A Multigrid Tutorial, SIAM, Philadelphia, PA, 1987.
- [34] P.N. Brown, A theoretical comparison of the Arnoldi and GMRES algorithms, SIAM J. Sci. Statist. Comput. 12 (1991) 58–78.
- [35] G.C. Broyden, A new method of solving nonlinear simultaneous equations, Comput. J. 12 (1969) 94-99.
- [36] N.I. Buleev, A numerical method for the solution of two-dimensional and three-dimensional equations of diffusion, Math. Sb. 51 (1960) 227–238 (in Russian).
- [37] L. Cesari, Sulla risoluzione dei sistemi di equazioni lineari per approssimazioni successive, Atti Accad. Naz. Lincei. Rend. Cl. Sci. Fis. Mat. Nat. Ser. 6a 25 (1937) 422–428.
- [38] T.F. Chan, E. Gallopoulos, V. Simoncini, T. Szeto, C.H. Tong, A quasi-minimal residual variant of the Bi-CGSTAB algorithm for nonsymmetric systems, SIAM J. Sci. Comput. 15 (1994) 338–347.
- [39] T.F. Chan, D. Goovaerts, A note on the efficiency of domain decomposed incomplete factorizations, SIAM J. Sci. Statist. Comput. 11 (1990) 794–803.
- [40] T.F. Chan, D. Resasco, A framework for the analysis of domain decomposition preconditioners, in: Proceedings of the First International Symposium on Domain Decomposition Methods for Partial Differential Equations, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1988, pp. 217–230.
- [41] D. Chazan, M. Miranker, Chaotic relaxation, Linear Algebra Appl. 2 (1969) 199-222.
- [42] E. Chow, Y. Saad, Approximate inverse techniques for block-partitioned matrices, SIAM J. Sci. Comput. 18 (1997) 1657–1675.
- [43] E. Chow, Y. Saad, Approximate inverse preconditioners via sparse-sparse iterations, SIAM J. Sci. Comput. 19 (1998) 995–1023.
- [44] G. Cimmino, Calcolo approssimato per le soluzioni dei sistemi di equazioni lineari, Ric. Sci. Progr. Tecn. Econom. Naz. 9 (1938) 326–333.
- [45] P. Concus, G.H. Golub, A generalized conjugate gradient method for nonsymmetric systems of linear equations, in: R. Glowinski, J.L. Lions (Eds.), Computing Methods in Applied Sciences and Engineering, Springer, New York, 1976, pp. 56–65.
- [46] P. Concus, G.H. Golub, G. Meurant, Block preconditioning for the conjugate gradient method, SIAM J. Sci. Statist. Comput. 6 (1985) 220–252.
- [47] P. Concus, G.H. Golub, D.P. O'Leary, A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations, in: J.R. Bunch, D.J. Rose (Eds.), Sparse Matrix Computations, Academic Press, New York, 1976, pp. 309–332.
- [48] J.D.F. Cosgrove, J.C. Díaz, A. Griewank, Approximate inverse preconditioning for sparse linear systems, Internat. J. Comput. Math. 44 (1992) 91–110.
- [49] J. Cullum, A. Greenbaum, Relations between Galerkin and norm-minimizing iterative methods for solving linear systems, SIAM J. Matrix Anal. Appl. 17 (1996) 223–247.
- [50] J.W. Daniel, The conjugate gradient method for linear and nonlinear operator equations, SIAM J. Numer. Anal. 4 (1967) 10–26.
- [51] J.W. Daniel, The Approximate Minimization of Functionals, Prentice-Hall, Englewood Cliffs, NJ, 1971.
- [52] E.F. D'Azevedo, F.A. Forsyth, W.P. Tang, Ordering methods for preconditioned conjugate gradient methods applied to unstructured grid problems, SIAM J. Matrix Anal. Appl. 13 (1992) 944–961.
- [53] E.F. D'Azevedo, F.A. Forsyth, W.P. Tang, Towards a cost effective ILU preconditioner with high level fill, BIT 31 (1992) 442–463.
- [54] C. de Boor, J.R. Rice, Extremal polynomials with applications to Richardson iteration for indefinite systems, SIAM J. Sci. Statist. Comput. 3 (1982) 47–57.
- [55] R. Dedekind, Gauss in seiner Vorlesung über die Methode der kleisten Quadrate, Gesammelte Math. Werke 2 (1901) 293–306.
- [56] M.A. DeLong, J.M. Ortega, SOR as a preconditioner, Appl. Numer. Math. 18 (1995) 431-440.

- [57] J.J. Dongarra, I.S. Duff, D.C. Sorensen, H.A. van der Vorst, Numerical Linear Algebra for High-Performance Computers, SIAM, Philadelphia, PA, 1998.
- [58] P.F. Dubois, A. Greenbaum, G.H. Rodrigue, Approximating the inverse of a matrix for use on iterative algorithms on vectors processors, Computing 22 (1979) 257–268.
- [59] I.S. Duff, G.A. Meurant, The effect of ordering on preconditioned conjugate gradients, BIT 29 (1989) 635-657.
- [60] T. Dupont, R. Kendall, H. Rachford, An approximate factorization procedure for solving self-adjoint elliptic difference equations, SIAM J. Numer. Anal. 5 (1968) 559–573.
- [61] T. Eirola, O. Nevanlinna, Accelerating with rank-one updates, Linear Algebra Appl. 121 (1989) 511-520.
- [62] S.C. Eisenstat, H.C. Elman, M.H. Schultz, Variational iterative methods for nonsymmetric systems of linear equations, SIAM J. Numer. Anal. 20 (1983) 345–357.
- [63] H.C. Elman, A stability analysis of incomplete LU factorizations, Math. Comp. 47 (1986) 191-217.
- [64] H.C. Elman, Relaxed and stabilized incomplete factorizations for non-self-adjoint linear systems, BIT 29 (1989) 890–915.
- [65] H.C. Elman, G.H. Golub, Line iterative methods for cyclically reduced discrete convection-diffusion problems, SIAM J. Sci. Statist. Comput. 13 (1992) 339–363.
- [66] H.C. Elman, Y. Saad, P. Saylor, A hybrid Chebyshev Krylov subspace algorithm for solving nonsymmetric systems of linear equations, SIAM J. Sci. Statist. Comput. 7 (1986) 840–855.
- [67] M. Embree, Convergence of Krylov subspace methods for non-normal matrices, Ph.D. Thesis, Oxford University Computing Laboratory, Oxford, UK, 1999.
- [68] M. Engeli, Overrelaxation and related methods, Refined Iterative Methods for Computation of the Solution and the Eigenvalues of Self-Adjoint Boundary Value Problems, Institute of Applied Mathematics, Zürich, Birkhäuser, Basel, 1959, pp. 79–107.
- [69] M. Engeli, T. Ginsburg, H. Rutishauser, E. Stiefel, Refined Iterative Methods for Computation of the Solution and the Eigenvalues of Self-Adjoint Boundary Value Problems, Birkhäuser, Basel, 1959.
- [70] D.J. Evans, The use of pre-conditioning in iterative methods for solving linear equations with symmetric positive definite matrices, J. Inst. Math. Appl. 4 (1968) 295–314.
- [71] V. Faber, T. Manteuffel, Necessary and sufficient conditions for the existence of a conjugate gradient method, SIAM J. Numer. Anal. 21 (1984) 352–361.
- [72] D.K. Faddeev, V.N. Faddeeva, Computational Methods of Linear Algebra, Freeman and Company, San Francisco, 1963.
- [73] R.P. Fedorenko, On the speed of convergence of an iteration process, USSR Comput. Math. Math. Phys. 4 (1964) 227–235.
- [74] B. Fischer, L. Reichel, A stable Richardson iteration method for complex linear systems, Numer. Math. 54 (1988) 225–241.
- [75] D.A. Flanders, G. Shortley, Numerical determination of fundamental modes, J. Appl. Phys. 21 (1950) 1322–1328.
- [76] R. Fletcher, Conjugate gradient methods for indefinite systems, in: G.A. Watson (Ed.), Proceedings of the Dundee Biennal Conference on Numerical Analysis 1974, Springer, New York, 1975, pp. 73–89.
- [77] D.R. Fokkema, G.L.G. Sleijpen, H.A. van der Vorst, Generalized conjugate gradient squared, J. Comput. Appl. Math. 71 (1994) 125–146.
- [78] S. Frankel, Convergence rates of iterative treatments of partial differential equations, MTAC (1950) 65–75.
- [79] R.W. Freund, A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems, SIAM J. Sci. Comput. 14 (2) (1993) 470–482.
- [80] R.W. Freund, N.M. Nachtigal, An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices, Part 2, Technical Report 90.46, RIACS, NASA Ames Research Center, 1990.
- [81] R.W. Freund, N.M. Nachtigal, QMR: a quasi-minimal residual method for non-Hermitian linear systems, Numer. Math. 60 (1991) 315–339.
- [82] K. Gallivan, A. Sameh, Z. Zlatev, A parallel hybrid sparse linear system solver, Comput. Systems Eng. 1 (2-4) (1990) 183–195.
- [83] N. Gastinel, Analyse Numérique Linéaire, Hermann, Paris, 1966.
- [84] C.F. Gauss, Werke, Band IX, Teubner, Leipzig, 1903.
- [85] M.K. Gavurin, Application of the best approximation polynomial to improving the convergence of iterative methods, Uspehi Mat. Nauk. 5(3) (1950) 156–160 (in Russian).

- [86] G.H. Golub, D.P. O'Leary, Some history of the conjugate gradient and Lanczos algorithms: 1948–1976, SIAM Rev. 31 (1989) 50–102.
- [87] G.H. Golub, R.S. Varga, Chebyshev semi iterative methods successive overrelaxation iterative methods and second order Richardson iterative methods, Numer. Math. 3 (1961) 147–168.
- [88] A. Greenbaum, Iterative Methods for Solving Linear Systems, SIAM, Philadelphia, PA, 1997.
- [89] A. Greenbaum, Z. Strakoš, Predicting the behavior of finite precision Lanczos and conjugate gradient computations, SIAM J. Matrix Anal. Appl. 13 (1992) 121–137.
- [90] A. Greenbaum, Z. Strakoš, Matrices that generate the same Krylov residual spaces, in: G. Golub, M. Luskin, A. Greenbaum (Eds.), Recent Advances in Iterative Methods, IMA Volumes in Mathematics and Its Applications, Vol. 60, Springer, New York, 1994, pp. 95–119.
- [91] A. Greenbaum, V. Ptak, Z. Strakoš, Any nonincreasing convergence curve is possible for GMRES, SIAM J. Matrix Anal. Appl. 17 (1996) 465–469.
- [92] M. Grote, T. Huckle, Parallel preconditioning with sparse approximate inverses, SIAM J. Sci. Comput. 18 (1997) 838–853.
- [93] M. Grote, H.D. Simon, Parallel preconditioning and approximate inverses on the connection machine, in: R.F. Sincovec, D.E. Keyes, L.R. Petzold, D.A. Reed (Eds.), Parallel Processing for Scientific Computing, Vol. 2, SIAM, Philadelphia, PA, 1992, pp. 519–523.
- [94] I. Gustafsson, A class of first order factorization methods, BIT 18 (1978) 142-156.
- [95] M.H. Gutknecht, A completed theory of the unsymmetric Lanczos process and related algorithms, Part I, SIAM J. Matrix Anal. Appl. 13 (1992) 594–639.
- [96] W. Hackbusch, Multi-Grid Methods and Applications, Springer, New York, 1985.
- [97] W. Hackbusch, Iterative Solution of Large Linear Systems of Equations, Springer, New York, 1994.
- [98] A.L. Hageman, D.M. Young, Applied Iterative Methods, Academic Press, New York, 1981.
- [99] R.M. Hayes, Iterative methods of solving linear systems on Hilbert space, National Bureau of Standards, Appl. Math. Ser. 39 (1954) 71–103.
- [100] M.R. Hestenes, The conjugate-gradient method for solving linear systems, in: Sympos. Appl. Math., Numerical Analysis, Vol. VI, McGraw-Hill, New York, 1956, pp. 83–102.
- [101] M.R. Hestenes, E.L. Stiefel, Methods of conjugate gradients for solving linear systems, J. Res. Nat. Bur. Standards Section B 49 (1952) 409–436.
- [102] A.S. Householder, Theory of Matrices in Numerical Analysis, Blaisdell Publishing Company, Johnson, CO, 1964.
- [103] K.C. Jea, D.M. Young, Generalized conjugate-gradient acceleration of nonsymmetrizable iterative methods, Linear Algebra Appl. 34 (1980) 159–194.
- [104] O.G. Johnson, C.A. Micchelli, G. Paul, Polynomial preconditionings for conjugate gradient calculations, SIAM J. Numer. Anal. 20 (1983) 362–376.
- [105] W.D. Joubert, Generalized conjugate gradient and Lanczos methods for the solution of nonsymmetric systems of linear equations, Ph.D. Thesis, University of Texas, Austin, Center for Numerical Analysis, Austin, TX, 1990
- [106] S. Kaczmarz, Angenäherte Auflösung von Systemen linearer Gleichungen, Bull. Internat. Acad. Pol. Sci. Lett. A 35 (1937) 355–357.
- [107] W. Kahan, Gauss-Seidel methods of solving large systems of linear equations, Ph.D. Thesis, University of Toronto, 1958.
- [108] C. Kamath, A. Sameh, A projection method for solving nonsymmetric linear systems on multiprocessors, Parallel Comput. 9 (1988/89) 291–312.
- [109] S. Kaniel, Estimates for some computational techniques in linear algebra, Math. Comp. 20 (1966) 369–378.
- [110] L. Kantorovitch, On an efficient method for solving optimization problems for quadratic functionals, DAN SSSR 48(7) (1945) 7–8 (in Russian).
- [111] W. Karush, Convergence of a method of solving linear problems, Proc. Amer. Math. Soc. 3 (1952) 839-851.
- [112] I.M. Khabaza, An iterative least-square method suitable for solving large sparse matrices, Comput. J. 6 (1963) 202–206.
- [113] L.Yu. Kolotilina, A.Yu. Yeremin, On a family of two-level preconditionings of the incomplete block factorization type, Soviet J. Numer. Anal. Math. Modelling 1 (1986) 293–320.
- [114] L.Yu. Kolotilina, A.Yu. Yeremin, Factorized sparse approximate inverse preconditionings I, Theory, SIAM J. Matrix Anal. Appl. 14 (1993) 45–58.
- [115] C. Lanczos, An iteration method for the solution of the eigenvalue problem of linear differential and integral operators, J. Res. Nat. Bur. Standards 45 (1950) 255–282.
- [116] C. Lanczos, Chebyshev polynomials in the solution of large-scale linear systems, Toronto Symposium on Computing Techniques (1952) 124–133.
- [117] C. Lanczos, Solution of systems of linear equations by minimized iterations, J. Res. Nat. Bur. Standards 49 (1952) 33-53.
- [118] H. Liebmann, Die angenäherte harmonischer Funktionen und konformer Abbildungen (nach Ideen von Boltzmann und Jacobi), S. B. Math. Nat. Kl. Bayerischen Akad. Wiss. München (1918) 385–416.
- [119] D.G. Luenberger, Hyperbolic pairs in the method of conjugate gradients, SIAM J. Appl. Math. 17 (1979) 1263– 1267.
- [120] M.M. Magolu, Modified block-approximate factorization strategies, Numer. Math. 61 (1992) 91-110.
- [121] M.M. Magolu, Ordering strategies for modified block incomplete factorizations, SIAM J. Sci. Comput. 16 (1995) 378–399.
- [122] T.A. Manteuffel, The Tchebychev iteration for nonsymmetric linear systems, Numer. Math. 28 (1977) 307-327.
- [123] T.A. Manteuffel, Adaptive procedure for estimation of parameter for the nonsymmetric Tchebychev iteration, Numer. Math. 28 (1978) 187–208.
- [124] S.F. Mc Cormick (Ed.), Multigrid Methods, SIAM, Philadelphia, PA, 1987.
- [125] J.A. Meijerink, H.A. van der Vorst, An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix, Math. Comp. 31 (137) (1977) 148–162.
- [126] J.A. Meijerink, H.A. van der Vorst, Guidelines for the usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems, J. Comput. Phys. 44 (1981) 134–155.
- [127] G. Meurant, Computer Solution of Large Linear Systems, North-Holland, Amsterdam, 1999.
- [128] J.C. Miellou, Algorithmes de relaxation chaotiques á retard, RAIRO R-1 (1975) 55-82.
- [129] N.M. Nachtigal, S.C. Reddy, L.N. Trefethen, How fast are nonsymmetric matrix iterations?, SIAM J. Matrix Anal. Appl. 13 (1992) 778–795.
- [130] P.A. Nekrasov, Determining unknowns using the least squares method when the number of unknowns is large, Mat. Sb. 12 (1884) 189–204 (in Russian).
- [131] Y. Notay, DRIC: a dynamic version of the RIC method, Numer. Linear Algebra Appl. 1 (1994) 511-532.
- [132] T.A. Oliphant, An extrapolation process for solving linear systems, Quart. Appl. Math. 20 (1962) 257-267.
- [133] C.W. Oosterlee, T. Washio, An evaluation of parallel multigrid as a solver and a preconditioner for singularly perturbed problems, SIAM J. Sci. Statist. Comput. 19 (1991) 87–110.
- [134] O. Osterby, Z. Zlatev, Direct Methods for Sparse Matrices, Springer, New York, 1983.
- [135] A.M. Ostrowski, Uber die Determinanten mit überwiegender Hauptdiagonale, Comment. Math. Helv. 10 (1937) 69–96.
- [136] A.M. Ostrowski, On the linear iteration procedures for symmetric matrices, Rend. Mat. Appl. 14 (1954) 140–163.
- [137] C.C. Paige, M.A. Saunders, Solution of sparse indefinite systems of linear equations, SIAM J. Numer. Anal. 12 (1975) 617–629.
- [138] C.C. Paige, M.A. Saunders, LSQR: an algorithm for sparse linear equations and sparse least squares, ACM Trans. Math. Software 8 (1982) 43–71.
- [139] B.N. Parlett, Reduction to tridiagonal form and minimal realizations, SIAM J. Matrix Anal. Appl. 13 (1992) 567–593.
- [140] B.N. Parlett, D.R. Taylor, Z.S. Liu, A look-ahead Lanczos algorithm for nonsymmetric matrices, Math. Comp. 44 (1985) 105–124.
- [141] D. Peaceman, H. Rachford, The numerical solution of elliptic and parabolic differential equations, J. SIAM 3 (1955) 28–41.
- [142] C. Pommerell, Solution of large unsymmetric systems of linear equations, Ph.D. Thesis, Swiss Federal Institute of Technology, Zürich, 1992.
- [143] G. Radicati di Brozolo, Y. Robert, Parallel conjugate gradient-like algorithms for solving sparse nonsymmetric systems on a vector multiprocessor, Parallel Comput. 11 (1989) 223–239.
- [144] E. Reich, On the convergence of the classical iterative method of solving linear simultaneous equations, Ann. Math. Statist. 20 (1949) 448–451.

- [145] J.K. Reid, On the method of conjugate gradients for the solution of large sparse systems of linear equations, in: J.K. Reid (Ed.), Large Sparse Sets of Linear Equations, Academic Press, New York, 1971, pp. 231–254.
- [146] L.F. Richardson, The approximate arithmetical solution by finite differences of physical problems involving differential equations with an application to the stresses to a masonry dam, Philos. Trans. Roy. Soc. London Ser. A 210 (1910) 307–357.
- [147] F. Robert, Contraction en norme vectorielle: convergence d'itérations chaotiques, Linear Algebra Appl. 13 (1976) 19–35.
- [148] F. Robert, M. Charnay, F. Musy, Iterations chaotiques série-parallèle pour les équations non linéaires de point fixe, Apl. Math. 20 (1975) 1–38.
- [149] A. Ruge, K. Stüben, Algebraic multigrid, in: S. McCormick (Ed.), Multigrid Methods, Frontiers in Applied Mathematics, Vol. 3, SIAM, Philadelphia, PA, 1987 (Chapter 4).
- [150] H. Rutishauser, Theory of gradient methods, Refined Iterative Methods for Computation of the Solution and the Eigenvalues of Self-Adjoint Boundary Value Problems, Institute of Applied Mathematics, Zurich, Birkhäuser, Basel, 1959, pp. 24–49.
- [151] Y. Saad, Practical use of polynomial preconditionings for the conjugate gradient method, SIAM J. Sci. Statist. Comput. 6 (1985) 865–881.
- [152] Y. Saad, Least squares polynomials in the complex plane and their use for solving sparse nonsymmetric linear systems, SIAM J. Numer. Anal. 24 (1987) 155–169.
- [153] Y. Saad, Preconditioning techniques for indefinite and nonsymmetric linear systems, J. Comput. Appl. Math. 24 (1988) 89–105.
- [154] Y. Saad, Krylov subspace methods on supercomputers, SIAM J. Sci. Statist. Comput. 10 (1989) 1200–1232.
- [155] Y. Saad, A flexible inner-outer preconditioned GMRES algorithm, SIAM J. Sci. Statist. Comput. 14 (1993) 461–469.
- [156] Y. Saad, Highly parallel preconditioners for general sparse matrices, in: G. Golub, M. Luskin, A. Greenbaum (Eds.), Recent Advances in Iterative Methods, IMA Volumes in Mathematics and Its Applications, Vol. 60, Springer, New York, 1994, pp. 165–199.
- [157] Y. Saad, Iterative Methods for Sparse Linear Systems, PWS Publishing, New York, 1996.
- [158] Y. Saad, M.H. Schultz, GMRES: a generalized minimal residual algorithm for solving a nonsymmetric linear systems, SIAM J. Sci. Statist. Comput. 7 (1986) 856–869.
- [159] J. Saltz, R. Mirchandaney, K. Crowley, Run-time paralellization and scheduling of loops, IEEE Trans. Comput. 40 (1991) 603–612.
- [160] P.E. Saylor, D.C. Smolarski, Computing the roots of complex orthogonal kernel polynomials, SIAM J. Sci. Statist. Comput. 9 (1988) 1–13.
- [161] J.W. Sheldon, On the numerical solution of elliptic difference equations, MTAC 9 (1955) 101–112.
- [162] G.H. Shortley, Use of Tschebyscheff-polynomial operators in the solution of boundary value problems, J. Appl. Phys. 24 (1953) 392–396.
- [163] H.D. Simon, Incomplete LU preconditioners for conjugate gradient type iterative methods, in: Proceedings of the SPE 1985 Reservoir Simulation Symposium, Dallas, TX, Society of Petroleum Engineers of AIME, Paper No 13533, 1988, pp. 302–306.
- [164] H.D. Simon, Direct sparse matrix methods, in: J.C. Almond, D.M. Young (Eds.), Modern Numerical Algorithms for Supercomputers, Center for High Performance Computing, The University of Texas, Austin, 1989, pp. 325–344.
- [165] G.L.G. Sleijpen, D.R. Fokkema, BICGSTAB(ℓ) for linear equations involving unsymmetric matrices with complex spectrum, ETNA 1 (1993) 11–32.
- [166] P. Sonnoveld, CGS: a fast Lanczos-type solver for nonsymmetric linear systems, SIAM J. Sci. Statist. Comput. 10 (1989) 36–52.
- [167] R.V. Southwell, Stress calculation in frameworks by the method of systematic relaxation of constraints, Proc. Roy. Soc. London Ser. A 151 (1935) 56–95.
- [168] W. Spakman, G. Nolet, Imaging algorithms, accuracy and resolution in delay time tomography, in: N.J. Vlaar, N. Nolet, M. Wortel, S. Cloetingh (Eds.), Mathematical Geophysics: A Survey of Recent Developments in Seismology and Geodynamics, Reidel, Dordrecht, 1987, pp. 155–188.
- [169] P. Stein, R.L. Rosenberg, On the solution of linear simultaneous equations by iteration, J. London Math. Soc. 23 (1948) 111–118.

- [170] E.L. Stiefel, Kernel polynomials in linear algebra and their applications, U.S. Nat. Bur. Standards Appl. Math. Ser. 49 (1958) 1–24.
- [171] H.L. Stone, Iterative solution of implicit approximations of multidimensional partial differential equations, SIAM J. Numer. Anal. 5 (1968) 530–558.
- [172] Z. Strakoš, On the real convergence rate of the conjugate gradient method, Linear Algebra Appl. 154/156 (1991) 535–549.
- [173] K.H. Tan, Local coupling in domain decomposition, Ph.D. Thesis, Utrecht University, Utrecht, The Netherlands, 1995.
- [174] K. Tanabe, Projection method for solving a singular system of linear equations and its applications, Numer. Math. 17 (1971) 203–214.
- [175] W.P. Tang, Generalized Schwarz splitting, SIAM J. Sci. Statist. Comput. 13 (1992) 573-595.
- [176] R.R. Underwood, An approximate factorization procedure based on the block Cholesky factorization and its use with the conjugate gradient method, Technical Report Tech Report, General Electric, San Jose, CA, 1976.
- [177] A. van der Ploeg, Preconditioning for sparse matrices with applications, Ph.D. Thesis, Department of Mathematics, University of Groningen, 1995.
- [178] A. van der Ploeg, E.F.F. Botta, F.W. Wubs, Nested grids ILU-decomposition (NGILU), J. Comput. Appl. Math. 66 (1996) 515–526.
- [179] A. van der Sluis, H.A. van der Vorst, The rate of convergence of conjugate gradients, Numer. Math. 48 (1986) 543–560.
- [180] V.V. Voevodin, The problem of a non-selfadjoint generalization of the conjugate gradient method has been closed, USSR Comput. Math. Math. Phys. 23 (1983) 143–144.
- [181] H.A. van der Vorst, Iterative solution methods for certain sparse linear systems with a non-symmetric matrix arising from PDE-problems, J. Comput. Phys. 44 (1981) 1–19.
- [182] H.A. van der Vorst, A vectorizable variant of some ICCG methods, SIAM J. Statist. Sci. Comput. 3 (1982) 350–356.
- [183] H.A. van der Vorst, Large tridiagonal and block tridiagonal linear systems on vector and parallel computers, Parallel Comput. 5 (1987) 45–54.
- [184] H.A. van der Vorst, High performance preconditioning, SIAM J. Sci. Statist. Comput. 10 (1989) 1174–1185.
- [185] H.A. van der Vorst, ICCG and related method for 3-D problems on vector computers, in: D. Truhlar (Ed.), Workshop on Practical Iterative Methods for Large Scale Computations, Minneaopolis, MN, October 23–25, 1988, Comput. Phys. Comm. 53 (1989).
- [186] H.A. van der Vorst, Bi-CDSTAB: a fast and smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems, SIAM J. Sci. Statist. Comput. 13 (1992) 631–644.
- [187] H.A. van der Vorst, C. Vuik, GMRESR: a family of nested GMRES methods, Numer. Linear Algebra Appl. 1 (1994) 369–386.
- [188] R.S. Varga, Matrix Iterative Analysis, Prentice-Hall, Englewood Cliffs, NJ, 1962.
- [189] R.S. Varga, Factorization and normalized iterative methods, in: R.E. Langer (Ed.), Boundary Problems in Differential Equations, University of Wisconsin Press, Madison, 1960, pp. 121–142.
- [190] P.K.W. Vinsome, ORTHOMIN: an iterative method solving sparse sets of simultaneous linear equations, Proceedings of the Fourth Symposium on Reservoir Simulation, Society of Petroleum Engineers of AIME, 1976, pp. 149–159.
- [191] C. Vuik, H.A. van der Vorst, A comparison of some GMRES-like methods, Linear Algebra Appl. 160 (1992) 131–162.
- [192] E.L. Wachspress, CURE, a generalized two-space-dimension multigroup coding for the IBM-704, Technical Report KAPL-1724, Knolls Atomic Power Laboratory, Schenectady, New York, 1957.
- [193] E.L. Wachspress, Iterative Solution of Elliptic Systems and Applications to the Neutron Equations of Reactor Physics, Prentice-Hall, Englewood Cliffs, NJ, 1966.
- [194] T. Washio, K. Hayami, Parallel block preconditioning based on SSOR and MILU, Numer. Linear Algebra Appl. 1 (1994) 533–553.
- [195] J.W. Watts III, A conjugate gradient truncated direct method for the iterative solution of the reservoir simulation pressure equation, Soc. Petroleum Eng. J. 21 (1981) 345–353.
- [196] R. Weiss, Error-minimixing Krylov subspace methods, SIAM J. Sci. Comput. 15 (1994) 511-527.

- [197] R. Weiss, A theoretical overview of Krylov subspace methods, in: W. Schönauer, R. Weiss (Eds.), Special Issue on Iterative Methods for Linear Systems, Applied Numerical Methods, 1995, pp. 33–56.
- [198] R. Weiss, Parameter-Free Iterative Linear Solvers, Akademie, Berlin, 1996.
- [199] P. Wesseling, An Introduction to Multigrid Methods, Wiley, Chichester, 1992.
- [200] O. Widlund, A Lanczos method for a class of nonsymmetric systems of linear equations, SIAM J. Numer. Anal. 15 (1978) 801–812.
- [201] J.H. Wilkinson, The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, 1965.
- [202] J.H. Wilkinson, C. Reinsch, Handbook of Automatic Computation, Linear Algebra, Vol. II, Springer, New York, 1971.
- [203] D.M. Young, Iterative methods for solving partial differential equations of elliptic type, Ph.D. Thesis, Harvard University, Cambridge, MA, USA, 1950.
- [204] D.M. Young, On Richardson's method for solving linear systems with positive definite matrices, J. Math. Phys. 32 (1954) 243–255.
- [205] D.M. Young, Iterative Solution of Large Linear Systems, Academic Press, New York, 1971.
- [206] D.P. Young, R.G. Melvin, F.T. Johnson, J.E. Bussoletti, L.B. Wigton, S.S. Samant, Application of sparse matrix solvers as effective preconditioners, SIAM J. Sci. Statist. Comput. 10 (1989) 1186–1199.
- [207] S.-L. Zhang, GPBi-CG: generalized product-type methods based on Bi-CG for solving nonsymmetric linear systems, SIAM J. Sci. Comput. 18 (1997) 537–551.
- [208] Z. Zlatev, Use of iterative refinement in the solution of sparse linear systems, SIAM J. Numer. Anal. 19 (1982) 381-399.



Journal of Computational and Applied Mathematics 123 (2000) 35-65

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.nl/locate/cam

## Eigenvalue computation in the 20th century

Gene H. Golub<sup>a, 1</sup>, Henk A. van der Vorst<sup>b,\*</sup>

<sup>a</sup>SCCM, Stanford University, Stanford, USA

<sup>b</sup>Department of Mathematics, Utrecht University, P.O. Box 80.010, 3508 TA, Utrecht, The Netherlands

Received 5 March 2000

#### Abstract

This paper sketches the main research developments in the area of computational methods for eigenvalue problems during the 20th century. The earliest of such methods dates back to work of Jacobi in the middle of the 19th century. Since computing eigenvalues and vectors is essentially more complicated than solving linear systems, it is not surprising that highly significant developments in this area started with the introduction of electronic computers around 1950. In the early decades of this century, however, important theoretical developments had been made from which computational techniques could grow. Research in this area of numerical linear algebra is very active, since there is a heavy demand for solving complicated problems associated with stability and perturbation analysis for practical applications. For standard problems, powerful tools are available, but there still remain many open problems. It is the intention of this contribution to sketch the main developments of this century, especially as they relate to one another, and to give an impression of the state of the art at the turn of our century. (c) 2000 Elsevier Science B.V. All rights reserved.

#### MSC: 65F15; 65N25

*Keywords:* QR; QZ; Lanczos' method; Arnoldi's method; Jacobi–Davidson; Power iteration; RKS; Jacobi's method; SVD; Perturbation theory

## 1. Sources

Numerical linear algebra is a very active field of research. Many problems are challenging of themselves, and in addition, much of scientific computing depends critically in one way or another on numerical linear algebra algorithms. Not only do the more classical scientific computational models for physical or engineering problems depend on linear algebra kernels, but many modern applications, such as information retrieval and image restoration, profit from numerical linear algebra

<sup>\*</sup> Corresponding author.

E-mail addresses: golub@sccm.stanford.edu (G.H. Golub), vorst@math.uu.nl (H.A. van der Vorst).

<sup>&</sup>lt;sup>1</sup> The work of this author was completed under a grant from the DOE: DE-FG03-97ER35330.

results. These factors have motivated numerical linear algebra research throughout the entire 20th century.

The field has blossomed, especially since the introduction of the modern computer, roughly from the early 1950s. This is evident from the large number of scientific journals in which articles in this area appear: *SIAM J. on Matrix Analysis and Applications (SIMAX), Linear Algebra and its Applications (LAA), Numerical Linear Algebra with Applications (NLAA),* are completely devoted to this specialty. Articles on numerical linear algebra, theoretical as well as applied, regularly appear in journals such as *BIT, SIAM J. Numerical Analysis, SIAM J. on Scientific Computing, J. on Computational and Applied Mathematics, J. Applied Numerical Mathematics, Numerische Mathematik, Numerical Algorithms, Mathematics of Computation, Parallel Computing, ACM Transactions on Mathematical Software, Computing, J. Inst. Math. Applic., SIAM Review, IMA J. Num. Anal., and several others in more application oriented directions, such as <i>J. Computational Physics* and engineering journals. And from, for instance, the bibliography in Golub and Van Loan's book [51], one can see how many papers are referenced from these and other sources. A quick glance through the contents of the average 60 papers per year in SIMAX shows that roughly 40% of the papers are associated with eigenvalue problem research, and it is likely that this holds more or less for the many papers per year that focus on numerical linear algebra.

This makes any attempt to write a complete overview on the research on computational aspects of the eigenvalue problem a hopeless task. It also serves as an excuse for the incompleteness in the current overview. We have tried to highlight what seemed most important from our point of view. We have included references to main sources, and we have made a personally colored selection of references to more specialized details. Instead of trying to give an overview of all sorts of different approaches that have been followed to solve aspects of eigenproblems, we will try to emphasize the history of those methods that, in our opinion, still play a role. Our aim is to consider the algorithmic developments from a historical point of view and to indicate how the recent powerful techniques are the result of many smaller steps. This will also help to show how many of the algorithms are interrelated; we hope not to get lost in sidesteps. The reader who is interested in methods that have played a role but that are at present no longer considered to be on the main track, is referred to Householder's and Wilkinson's books [64,154]. In addition, Parlett [100] gives interesting historical information on older methods that still have some significance from a theoretical point of view.

In order to be active in this area of research, or to be informed about special aspects, then one might be interested in our main sources:

- Wilkinson: The Algebraic Eigenvalue Problem [154].
- Householder: The Theory of Matrices in Numerical Analysis [64].
- Wilkinson and Reinsch: The Handbook [158].
- Parlett: The Symmetric Eigenvalue Problem [100].
- Stewart and Sun: Matrix Perturbation Theory [129].
- Watkins: Fundamentals of Matrix Computations [150].
- Golub and Van Loan: Matrix Computations [51].
- Chatelin: Spectral Approximation of Linear Operators [18].
- Saad: Numerical Methods for Large Eigenvalue Problems [116].
- Demmel: Applied Numerical Linear Algebra [28].
- Trefethen and Bau: Numerical Linear Algebra [137].

- Arpack Guide (Lehoucq, Sorensen and Yang) [81].
- Dongarra et al.: Numerical Linear Algebra for High Performance Computers [31].
- Bai's paper in Numerical Lin. Alg. with Appl. [6].
- Watkin's paper in SIAM Review [151].
- Dongarra and Walker on Software [33].
- Wilkinson: State of the Art overview [157].
- van der Vorst and Golub: State of the Art paper [50].

By regularly examining the dedicated numerical linear algebra journals, one should be able to trace most of the relevant and interesting papers for further investigations and research.

It should be noted that we have concentrated on algebraic eigenvalue problems in this paper. For eigenvalue problems related to, for instance, PDEs, one may use methods that exploit the nature of the PDE or the expected behaviour of the solution. We have not considered such specialized techniques (of which multigrid is a good example).

### 2. Introduction

The eigenvalue problem for square matrices A, that is the determination of nontrivial solutions of  $Ax = \lambda x$ , is a central topic in numerical linear algebra. It is inherently nonlinear and this leads to many computational problems. Computation of the eigenvalues  $\lambda$  via the explicit construction of the characteristic equation

$$\det(A - \lambda I) = 0$$

is, except for very special cases, not an option since the coefficients of the characteristic equation cannot be computed from determinant evaluations in a numerically stable way. And even if the characteristic equation could be determined accurately, then the computation of its roots, in finite precision, may be highly unstable since small perturbations in the coefficients may lead to large perturbations of the roots. The numerical computation of the associated eigenvectors and generalized eigenvectors is even more delicate, in particular when eigenvectors of A make small angles with each other. In the limiting case, when the matrix is defective, A can be reduced to the Jordan canonical form, but arbitrary small perturbations in A may yield a nondefective matrix. This leads to many challenging numerical questions, which give rise to the central problem: how can we compute eigenvalues and eigenvectors in an efficient manner and how accurate are they?

In fact, this was already recognized by Jacobi, who, in 1846, computed the eigenvalues of symmetric matrices by rotating the matrix to a strongly diagonally dominant one. We will return to this later, since Jacobi's techniques are still relevant and have led to popular and powerful algorithms.

Another longstanding method that is of great significance and serves as the basis for many algorithms is the Power iteration. The method is based on the idea that if a given vector is repeatedly applied to a matrix, and is properly normalized, then ultimately, it will lie in the direction of the eigenvector associated with the eigenvalues which are largest in absolute value. The rate of convergence for the Power iteration depends on the ratio of the second largest eigenvalue (in absolute value) to the largest eigenvalue (in absolute value) and for many applications this leads to unacceptably slow convergence. The method can be problematic if one wants to compute a number of extremal eigenvalues. The Power iteration is still in use, but most frequently as (implicit) part of more efficient techniques, e.g., Krylov methods, inverse iteration, QR-method.

What becomes clear is that all these methods are of an iterative nature, and this is necessarily the case, since if there were a method of computing the eigenvalues of an *n*th-order matrix in a finite number of computations, depending only on n, then this would be in contradiction with the fundamental theorem of Abel–Ruffini (and also a well-known result in Galois theory) that no such algorithm exists for the computation of the roots of a general polynomial of degree greater than 4. Hence, an algorithm for a matrix with a general structure (that is, not a diagonal matrix or a triangular matrix or alike) is necessarily iterative and the problem is to identify iterative algorithms which have a fast rate of convergence and lead to accurate results.

In solving an eigenvalue problem there are a number of properties that need be considered. These greatly affect the choice of algorithm. We list below a number of questions that an investigator needs to consider in solving a particular problem.

- Is the matrix real or complex?
- What special properties does the matrix have?
  - symmetric,
  - Hermitian,
  - o skew symmetric,
  - unitary.
- Structure?
  - band,
  - o sparse,
  - o structured sparseness,
  - Toeplitz.
- Eigenvalues required?
  - o largest,
  - smallest in magnitude,
  - o real part of eigenvalues negative,
  - o sums of intermediate eigenvalues.

As well as the standard eigenproblem, there are a variety of more complicated eigenproblems, for instance  $Ax = \lambda Bx$ , and more generalized eigenproblems like  $Ax + \lambda Bx + \lambda^2 Cx = 0$ , higher-order polynomial problems, and nonlinear eigenproblems. All these problems are considerably more complicated than the standard eigenproblem, depending on the operators involved. However, as the standard eigenproblem has become better understood, in a numerical sense, progress has been made in the other problems and we will consider developments in solving these problems.

## 3. Canonical forms

The standard approach for the numerical solution of the eigenproblem is to reduce the operators involved to some simpler form, that yields the eigenvalues and eigenvectors directly, for instance, diagonal form. The idea is that the transformation be made with orthogonal operators as often as possible, in order to reduce the effects of perturbations.

The easiest situation is the symmetric case: for a real symmetric matrix, there exists an orthogonal matrix Q, so that  $Q^{T}AQ = D$ , where D is a diagonal matrix. The diagonal elements of D are the eigenvalues of A, the columns of Q are the corresponding eigenvectors of A.

Unsymmetric matrices do not in general have an orthonormal set of eigenvectors, and may not have a complete set of eigenvectors, but they can be transformed unitarily to Schur form:

 $Q^*AQ = R$ ,

in which R is upper triangular. In fact, the symmetric case is a special case of this Schur decomposition, since a symmetric triangular matrix is clearly diagonal. Apart from the ordering of the eigenvalues along the diagonal of R and the sign of each column of Q, the matrix Q is unique. Van Dooren [146] has described an algorithm for the orthogonal transformation of Q, so that the eigenvalues appear in prescribed order along the diagonal of R. If the eigenvalues are distinct then there exists a nonsingular matrix X (in general not orthogonal), that transforms A to diagonal form

 $X^{-1}AX = D.$ 

An unsymmetric matrix can be transformed to Jordan form by a nonsingular matrix X. This Jordan matrix may have upper bidiagonal blocks along the diagonal. Each of these blocks has identical eigenvalues and the upper bidiagonal elements are equal, and most often set to 1. The numerical computation of the Jordan form is highly unstable, since a small perturbation suffices to obtain a matrix with different eigenvalues (and possibly a complete eigenvector system). Small angles between (some of) the eigenvectors reveal that A is close to a matrix that is similar to a nondiagonal Jordan form. For a discussion on how to compute elementary Jordan blocks (with the help of the singular value decomposition), see the 1976 paper by Golub and Wilkinson [52].

Just as the Jordan canonical form describes the eigenstructure of a matrix, the Kronecker form does this for matrix pencil  $A - \lambda B$ , even for rectangular A and B. For details on this we refer to papers by Wilkinson [155,156,143], and Kågström [66]. The latter has also developed software for the computation of the Kronecker structure [67]. Wilkinson, in his discussion on the progress made in the period 1976–1986 in eigenvalue computations [157], noted that the Jordan canonical and Kronecker canonical forms were largely regarded as irrelevant by numerical analysts because of their ill-posedness. He even stated: "Many felt that I should have ignored the Jordan canonical form in the Algebraic Eigenvalue Problem [154] and I had misgivings about including a discussion of it". Since the 1970s, this has changed, and contributions have been made by many, including Demmel, Beelen, Van Dooren, Chaitin-Chatelin, Edelman, Kågström, Nichols, Kautsky, Golub, and Wilkinson. Although serious attempts have been undertaken for the computation of the Kronecker canonical form, by for instance Kågström and Van Dooren, this still needs further research. Also the computation of invariant subspaces of highly nonnormal matrices is still in its infancy, notwithstanding useful contributions by, for instance, Chaitin-Chatelin et al. [17,12] and Lee [80]. For recent references, see [4]. Van Dooren described, in papers published in 1981, how the Kronecker form can be used in system control problems (input-output systems) [145,144].

Related to eigendecompositions is the singular value decomposition. Let A be a real  $m \times n$  matrix, then there exists an orthogonal  $m \times m$  matrix U and an orthogonal  $n \times n$  matrix V, such that

 $U^{\mathrm{T}}AV = \mathrm{diag}(\sigma_1,\ldots,\sigma_p),$ 

with  $p = \min\{m, n\}$ , and  $\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_p \ge 0$ . The values  $\sigma_i$  are the *singular values* of A, the columns  $v_i$  of V are the *right singular vectors* and the columns  $u_i$  of U are the *left singular vectors*. The number of nonzero singular values is equal to the rank of the matrix A. The singular value decomposition (SVD) plays an important role in numerical linear algebra, for instance in the solution of underdetermined or overdetermined linear systems.

#### 4. Perturbation theorems

Perturbation theorems play a very essential role in computational processes for eigenproblems. As we have remarked, eigensolvers are essentially iterative processes and many of them rest on the principle of reducing a matrix to a special/simpler form, either diagonal or upper triangular. One has then to decide when a matrix is sufficiently close to the appropriate ultimate form. It is important to know what the approximation errors imply about the desired eigen information. A modern treatment of perturbation theory for a variety of eigenproblems is given in the book by Stewart and Sun [129]. We will restrict ourselves here to what we regard as some of the most relevant results in this area.

Many theoretical results rest on the famous Gershgorin Disc Theorem, which states that the eigenvalues of a matrix  $A = (a_{ij})$  are located in the union of circles with centre  $a_{ii}$  and radius  $\sum_{j \neq i} |a_{ij}|$ . This theorem first appeared in a classic paper by Gerschgorin in 1931 [44]. A very useful refinement shows that if a group of *s* discs  $D_i$  is isolated from the others, then this group contains precisely *s* eigenvalues. In particular, if one disc is isolated then this disc contains one eigenvalue. This particular case is of great interest, since it can be used for stopping criteria in actual computations.

Wilkinson [154] discussed the application of Gerschgorin's theorem to various situations. He mentioned the discussion of useful extensions by Taussky in her 1949 paper [131]: Varga acknowledged Taussky's paper in his work on solving systems of linear equations by iterative methods (cf. [148]). An important extension, using block matrices, of the Gerschgorin Theorem was given by Feingold and Varga [38].

The eigenvalues depend continuously on the elements of A and if the *i*th eigenvalue  $\lambda_i$  is distinct, then it is even differentiable. In this case one can carry through a first-order perturbation analysis (cf. [129, p. 185]). This leads to the observation that if a matrix A is perturbed by  $\delta A$ , then the perturbation to  $\lambda_i$  is in first order of terms of  $\delta A$  given by

$$\delta\lambda_i = \frac{1}{y_i^* x_i} y_i^* \delta A x_i,$$

where  $x_i$ , and  $y_i$  are the normalized right and left eigenvectors, respectively, corresponding to  $\lambda_i$ , and  $y_i^*$  denotes the complex conjugate of  $y_i$ . The factor  $1/y_i^*x_i$  is referred to as the *condition number* of the *i*th eigenvalue. The Bauer–Fike result (1960) [9], which is actually one of the more famous refinements of Gershgorin's theorem, makes this more precise: the eigenvalues  $\tilde{\lambda}_j$  of  $A + \delta A$  lie in discs  $B_i$  with centre  $\lambda_i$ , and radius  $n(||\delta A||_2/|y_i^*x_i|)$  (for normalized  $x_i$  and  $y_i$ ).

The Courant–Fischer minimax theorem is the basis of many useful results. For a symmetric matrix A with ordered eigenvalues  $\lambda_n \leq \cdots \leq \lambda_2 \leq \lambda_1$  it states that the eigenvalues are the stationary values of the *Rayleigh quotients*:

$$\lambda_k = \max_{\dim(S) = k} \min_{0 \neq y \in S} \frac{y^* A y}{y^* y},$$

for k = 1, 2, ..., n. Some important consequences are the following.

For symmetric matrices, Weyl in 1912 proved an important property for symmetric perturbations  $\delta A$ :

$$|\lambda_i - \lambda_i| \leq \|\delta A\|_2.$$

In fact, Weyl gave even more detailed results in terms of the eigenvalues of A and of  $\delta A$ : let the eigenvalues of  $\delta A$  be denoted by  $\eta_n \leq \cdots \leq \eta_2 \leq \eta_1$ , then the eigenvalues  $\tilde{\lambda}_i$  of A + E satisfy

$$\lambda_k + \eta_n \leqslant \tilde{\lambda}_k \leqslant \lambda_k + \eta_1.$$

These inclusion formulas were later refined to, in Parlett's terminology, a blizzard of results, independently obtained by Kahan  $(1957)^2$  and Weinberger (1974) [153]. See Parlett's book [100, Chapter 10.6] for an exposé of these results. It is interesting to note that by 1990 the theory had evolved to such an extent that Weyl's result and the Kato–Temple results could be left as exercises in Stewart and Sun's book [129, p. 210–211]. This illustrates a rich and powerful framework.

Another important property that plays a big role in iterative (projection type) algorithms, is the interlace property. Let  $A_r$  denote the leading  $r \times r$  minor of A, with eigenvalues  $\lambda_i^{(r)}$ , then

$$\lambda_{r+1}^{(r+1)} \leqslant \lambda_r^{(r)} \leqslant \lambda_r^{(r+1)} \leqslant \cdots \leqslant \lambda_2^{(r+1)} \leqslant \lambda_1^{(r)} \leqslant \lambda_1^{(r+1)}$$

An important result, that underlies the powerful *divide and conquer* method, comes from rank-one perturbations. If  $B = A + \tau cc^{T}$ , with  $||c||_{2} = 1$ , and real  $\tau \ge 0$ , then the *i*th eigenvalue of *B* is in the interval  $[\lambda_{i}, \lambda_{i-1}]$ , for  $\tau \le 0$  it is in  $[\lambda_{i+1}, \lambda_{i}]$ . In either case, there exist nonnegative  $\alpha_{1}, \ldots, \alpha_{n}$  with  $\sum_{i} \alpha_{i} = 1$ , such that  $\lambda_{i}(B) = \lambda_{i} + \alpha_{i}\tau$ .

Further details and results can be found in most books on the (numerical) eigenproblem; in particular Wilkinson's book [154] is a great source. A good overview of these results and similar results for invariant subspaces is given in [51]. From a result formulated as the Kato–Temple theorem, one can obtain sharp bounds for the Rayleigh quotients for symmetric matrices. This rests on work of Temple (1933) [132] and Kato (1949) [70]; more extended work in this direction has been done by Davis and Kahan (1970) [27] (see also [18, p. 46], [28, Chapter 5.2]). In these results the *gap* for the *i*th eigenvalue plays an essential role:  $gap(i, A) \equiv \min_{j \neq i} |\lambda_j - \lambda_i|$ . A small gap indicates a sensitive eigenvector. In particular, let x denote a normalized vector with Rayleigh quotient  $\beta = x^T Ax$ , and residual  $r = Ax - \beta x$ . Then there is a (normalized) eigenvector  $q_i$ , corresponding to  $\lambda_i$ , for which

$$|\lambda_i - \beta| \leqslant \frac{\|r\|_2^2}{\operatorname{gap}(i, A)}, \qquad \sin(\theta) \leqslant \frac{\|r\|_2}{\operatorname{gap}(i, A)},$$

where  $\theta$  denotes the angle between x and  $q_i$ . These results show the superior quality of a Rayleigh quotient from a given subspace. It is exploited in modern iterative methods, such as the Lanczos algorithm, but it is also essential in the QR algorithm. Related to the perturbation analysis for Rayleigh quotients is work of Kaniel (1966) [69] for errors in the Ritz approximations computed in the Lanczos process. For a comprehensive discussion of this, see [100, Chapter 12].

<sup>&</sup>lt;sup>2</sup> According to Parlett [100, p. 203], who also mentions unpublished (?) results of Weinberger (1959).

#### 5. Jacobi's method

For our discussion of the Jacobi method, we have used the following sources: [154, Chapter 5], [100, Chapter 9], [51, Chapter 8.4]. The Jacobi method which was originally proposed in 1846 [65], reduces a real symmetric matrix to diagonal form by a sequence of plane rotations. Jacobi, however, did not use the method to full convergence, as was done in the 20th century, but combined it with an iterative process (Gauss-Jacobi iterations) for the missing component of the desired eigenvector (for which he took a unit vector as an initial guess). Actually, Jacobi's technique may be viewed as a form of preconditioning for the Gauss-Jacobi iteration that he also used to solve linear least-squares systems. This has escaped the attention of most researchers that were active with Jacobi's method; the exception seems to be Bodewig [2, pp. 280-287]. The preconditioning part of the method, as an iterative technique to diagonalize a matrix, was reinvented in 1949 by Goldstine et al. and published in a manuscript. After Ostrowski had pointed out that this was actually a rediscovery of Jacobi's method,<sup>3</sup> the adapted manuscript was published only in 1959 [46].<sup>4</sup> According to Wilkinson [154, p. 343] Jacobi's method was already being used independently on desk computers at the National Physical Laboratory in 1947. From 1950 on, the method got much attention. In the classical process, the maximal off-diagonal element is annihilated, and this guarantees convergence. Since it is a time-consuming process to determine the maximal element after each rotation, cyclic procedures were suggested, (cf. [54]). Later, threshold strategies were developed in order to avoid wasting time in trying to annihilate tiny elements [105]. Quadratic convergence for the cyclic Jacobi algorithm was proven, under various assumptions, by Henrici (1958) [59], Schönhage (1961) [119], Wilkinson (1962) [154], and van Kempen (1966) [147]. This rate of convergence sets in after a number of sweeps (that is (n-1)n/2 elementary rotations), but there is no rigorous bound on the number of sweeps required to achieve a specified accuracy. Brent and Luk [14] argued that this number is proportional to log(n), which is in line with Parlett's remark [100, p. 181] that after three or four sweeps through all the off-diagonal elements convergence is usually very rapid.

The success of the Jacobi method for diagonalizing a symmetric matrix by orthogonal similarity transformations inspired many investigators to find a similar method for nonsymmetric matrices. It was quickly realized that the Schur form was the appropriate decomposition. John Greenstadt, in 1955 [53], was one of the earliest investigators to develop such a method (indeed, Greenstadt made von Neumann aware of this canonical form). Unfortunately, these earliest attempts were not successful. The QR method, that gained more popularity somewhat later, can be viewed, however, as a Jacobi like method, since it can produce the Schur form via a sequence of similarity transformations composed of rotations.

Rutishauser made an Algol60 implementation of Jacobi's process, as a contribution to the Wilkinson Reinsch collection [158]. In the 1960s, the popularity of the method declined, because of the growing popularity first of the Givens method and slightly later, the Householder method: these latter two methods first reduced the matrix to tridiagonal form and then used an efficient procedure for computing the eigenvalues of the tridiagonal matrix. Interest in the Jacobi returned with the advent

<sup>&</sup>lt;sup>3</sup> Michele Benzi brought to our attention that this story is narrated on p. 294 in Goldstine's book *The Computer from Pascal to von Neumann*, Princeton University Press, 1972; it is also nicely covered in the Master Thesis of Anjet den Boer [30] on the history of Jacobi's method.

<sup>&</sup>lt;sup>4</sup> Parlett [100, p. 184] dates the reinvention in 1946, by Bargmann et al. [7], but this is presumably a misprint.

of parallel computers, starting with a paper of Sameh (1971) [118], and followed by others in the 1980s. Variants of Jacobi's method were proposed; we mention the extensions for normal matrices, by Paardekooper in 1971 [96], and to nonnormal matrices by Eberlein, in 1970 [36]. The latter process was also part of the Wilkinson and Reinsch collection (the Algol60 procedure *eigen*). In order to improve data locality for distributed memory computers, block Jacobi methods were suggested, see Bischof [10] for a discussion on the influence of the solution of the subproblems on the overall process.

Another interesting feature of the Jacobi method is its superior behaviour with respect to accuracy. Wilkinson analysed this and showed that the relative error in the eigenvalue approximations is eventually reduced to the order of the condition number of A times machine precision. This was perfected in 1992, by Demmel and Veselic [29], who showed that for symmetric positive-definite matrices, the condition number of A could be replaced by that of the matrix symmetrically scaled by the diagonal. If one is satisfied with less accuracy, then for large-scale computations, Jacobi's method is no longer regarded as competitive, not even for modern parallel computers.

#### 6. Power method

For our discussion of the Power method, we have borrowed material from Householder's book [64]. The Power method, for general square matrices, is the simplest of all the methods for solving for eigenvalues and eigenvectors. The basic idea is to multiply the matrix A repeatedly by a well-chosen starting vector, so that the component of that vector in the direction of the eigenvector with largest eigenvalue in absolute value is magnified relative to the other components. Householder called this Simple Iteration, and attributed the first treatment of it to Müntz (1913). Bodewig [2, p. 250] attributes the power method to von Mises [149], and acknowledges Müntz for computing approximate eigenvalues from quotients of minors of the *explicitly* computed matrix  $A^k$ , for increasing values of k. For a careful analytic treatment of the Power method, Householder acknowledged work by Ostrowski and Werner Gautschi; the reader can find a fairly complete treatment in Wilkinson's book [154] together with the proper references. The speed of convergence of the Power iteration depends on the ratio of the second largest eigenvalue (in absolute value) to the largest eigenvalue (in absolute value). In many applications this ratio can be close to 1 - this has motivated research to improve the efficiency of the Power method. It is interesting that the most effective variant is the inverse Power method, in which one works with the matrix  $(A - \mu I)^{-1}$ , and this variant was proposed as late as 1944 by Wielandt (Wielandt's fractional iteration). Wielandt also proposed continuing the process after the largest eigenvalue has converged, by working with the deflated matrix  $A - \lambda v v^*$ , for which  $\lambda, v$  is the computed eigenpair (with  $||v||_2 = 1$ ), associated with the largest eigenvalue in magnitude. (The deflation procedure outlined here is for symmetric matrices. For unsymmetric matrices it is necessary to work with at least two vectors; the choice of one of the vectors may not be unique.) This is called *implicit deflation*; another possibility is to keep the iteration vectors orthogonal to the computed eigenvector(s): explicit deflation. A compact description and analysis of these deflation techniques was given by Parlett [100]. The Power method and the Inverse Power method, in their pure form are no longer competitive methods even for the computation of a few eigenpairs, but they are still of interest since they are explicitly or implicitly part of most modern methods such as the QR method, and the methods of Lanczos and Arnoldi. These methods evolved in some way or another from the Power method and some of the techniques that were suggested as improvements to the Power method are still in use as acceleration techniques for modern iterative methods. One of these ideas is to work with polynomials of A, with the purpose of damping unwanted parts of the spectrum.

Another possibility is working with properly updated shifts  $\mu$  in the inverse process and, in particular, if one takes the Rayleigh quotient with the most recent vector as a shift, then one obtains the *Rayleigh quotient iteration*. According to Parlett [100, p. 71], Lord Rayleigh used in the 1870s a less powerful technique: he did a single shift-and-invert step with a Rayleigh quotient for an eigenvector approximation, but with a unit vector as the right-hand side. (This saves the refactoring of the matrix  $(A - \mu I)$  at each iteration.) The modern RQI, in which one takes the most current eigenvector approximation as the right-hand side, leads to very fast convergence. Ostrowski, in a series of six papers [95], studied the convergence properties for variance of RQI for the symmetric and unsymmetric case. He was able to establish cubic convergence in both cases under various circumstances (in the unsymmetric case for a properly generalized Rayleigh quotient). These results are essential for the understanding of modern iterative techniques that are based on (approximate) shift-and-invert strategies (for example, the Jacobi–Davidson method, see below).

A step forward was to work with a set of independent vectors in order to find a number of eigenvectors, instead of the deflation procedure suggested by Wielandt. A problem with the Power method is the determination of eigenvalues that have equal modulus, for instance, finding a conjugate pair of eigenvalues of a real unsymmetric matrix. It is therefore quite natural to work with a couple of independent vectors: this was first suggested in 1937 by Horst [62]. The next step that seems logical, in hindsight, is to force the vectors to be independent. This was initially done (cheaply) by Gaussian transformations by Bauer [8] in 1957, and led to Treppeniteration. If the set of vectors is denoted as a matrix  $L_s$  (an *n* by *s* unit lower-triangular matrix), then one forms  $AL_s$  and factors the resulting matrix, by Gaussian transformations, as  $L_{s+1}R_{s+1}$ . If the eigenvalues are distinct, then the  $s \times s$  upper triangular matrix  $R_{s+1}$  converges, for increasing s, to a matrix whose eigenvalues are those of the dominant subspace on its diagonal. Rutishauser [111] made the important observation that if we factor A as A = LR (again L unit lower triangular), then the similar matrix  $L^{-1}AL = L^{-1}LRL = RL$ . He proposed decomposing RL again, and repeating this process in an iterative fashion. This R also converges to an upper triangular matrix, and L is a unit matrix. This is the LR method of Rutishauser. The correspondence between Treppeniteration and LR is that if we start Treppeniteration with a unit full matrix, then in exact arithmetic we obtain the same matrices R in the process. For an efficient implementation, the matrix A is first reduced to an upper Hessenberg matrix. The LR method maintains this form throughout the process, and this makes LR computationally very attractive. Rutishauser's observation that permuting the factors of the matrix is equivalent to performing a similarity transformation was a key step. Wilkinson [154, p. 485] commented: "In my opinion its development (i.e. of LR) is the most significant advance which has been made in connection with the eigenvalue problem since the advent of automatic computers". However, Bauer's technique could be applied to a smaller set of starting vectors and it does not modify the matrix A. For this reason, in the words of Householder [64, p. 195], it is self-correcting. This seems to imply that Treppeniteration leads to more accurate results.

Since orthogonal reduction techniques often evidence superior stability properties, it became apparent that the LR factorization should be replaced by a QR factorization. This leads to one of the most popular and powerful methods of our time for eigenvalue problems: the QR method for

computing all of the eigenvalues and associated eigenvectors of a dense symmetric matrix. (In fact, the QR method has essential enhancements that make the method really powerful; we will discuss this in another section.)

With the number of vectors less than n, this Power method in combination with QR orthogonalization is known as the *Simultaneous Iteration method*; Rutishauser studied this method in 1969 [112], see also [113]. Its convergence behaviour for general unsymmetric matrices was studied by Stewart [126] in 1976. Stewart also developed a subroutine, based on simultaneous iteration, for the computation of a dominant invariant subspace. This routine, SRRIT [127], was further improved in 1992, and made available for general use through Netlib [5].

The collection of vectors generated by the Power method define Krylov subspaces of increasing dimension. This motivated Krylov to try to determine the characteristic polynomial of a matrix by inspecting the dependence of a full set of these vectors. This procedure may fail because the system of equations is highly ill-conditioned but this can be repaired by orthogonalizing each new vector to the previous vectors and applying A onto the last constructed vector. This iteration process is known as the Lanczos method for symmetric matrices, and Arnoldi's method for unsymmetric matrices. We will discuss these Krylov methods below.

Our presentation might suggest that the Krylov methods have overshadowed the Simultaneous Iteration method, and for most situations this is indeed the case. Parlett, however, [100, p. 289] described situations where Simultaneous Iteration is still competitive. For instance, if we can store only a limited number of n-vectors in fast memory, or if the relative gap between the desired eigenvalues and the others is great, then Simultaneous Iteration is very useful.

#### 7. Reduction algorithms

Early computational techniques, other than Jacobi's famous but slowly converging diagonalization method, and the unsatisfactory Power method with its many restrictions, attempted to exploit the fact that every matrix satisfies its characteristic equation. To this end, Krylov suggested in 1931 [73], using the vectors  $x, Ax, A^2x, \ldots$ , generated by the Power method, to determine the coefficients of the characteristic equation. This was not successful, because, as we have learned from Wilkinson's analysis [154] the roots of a polynomial may vary widely with only tiny perturbations to the coefficients of the polynomial. Even rounding the exact coefficients in floating point arithmetic may destroy much accuracy in many of the roots. Although Krylov's method failed, his name is still attached to the subspace generated by the Power method.

There is yet another reason for the failure of Krylov's method in finite precision arithmetic: the vectors generated by the Power method tend to converge in the direction of the eigenvectors associated with the dominating eigenvalues. Hence, the computed vectors for the subspace necessarily yield a very ill-conditioned basis. Checking mutual dependence of this basis, as is required in order to construct the characteristic polynomial, is an almost impossible task.

An early attempt to reduce the matrix A to a form that lends itself better for solving the characteristic equation was suggested by Hessenberg [60]. He suggested to compute a modified Krylov basis by making a set of basis vectors for the Krylov subspace, orthogonal to a given test-set, for instance the canonical basis vectors. This led to a reduction of A to upper Hessenberg form. This technique is very close to the techniques by Lanczos and Arnoldi. In 1950, Lanczos [78] suggested building a basis for the Krylov subspace in a more stable way by orthogonalizing the vectors as they are constructed. The idea was to immediately orthogonalize the new vector for the Krylov subspace with respect to the already existing orthonormal basis. The difference with Hessenberg's approach is that Lanczos (and slightly later also Arnoldi) took the Krylov vectors themselves for the test-set.

The new vector for expansion is created by applying A to the latest orthogonal basis vector. For symmetric matrices A, this leads to a three-term recurrence relation between the basis vectors  $v_j$  and in exact arithmetic this can be formulated as

$$AV_j = V_jT_j + \gamma_j v_{j+1} \mathbf{e}_{j+1}^{\mathrm{T}}$$

Obviously, this recursion must terminate for some  $j \leq n$ , in which case  $V_i$  forms the basis for an invariant subspace of A, and the eigenvalues of the tridiagonal matrix  $T_i$  are the eigenvalues of A with respect to this invariant subspace. This algorithm is equivalent to the well-known algorithm of the Dutch mathematician Stieltjes for generating orthogonal polynomials by a three-term recurrence relationship. Lanczos also proposed a reduction process for unsymmetric matrices A, the so-called two-sided Lanczos process. In this process two sets of basis vectors are constructed, one for the Krylov subspace with A and one for a Krylov subspace with  $A^{T}$ . By requiring biorthogonality of the two sets, the two bases can be used for reduction of A to tridiagonal form. This form has suffered from many drawbacks. Not only are the reduction matrices nonorthogonal, a suspect property, but the algorithm also suffers from various break-down conditions. (The basic problem lies in the fact that the measure generated by the initial vectors is not nonnegative). The symmetric variant did not become popular in the 1950s, since it was soon recognized that rounding errors could spoil the process dramatically. Wilkinson [154] showed that the Lanczos algorithm is highly (forward) unstable and there seemed to be no way of stabilizing the process other than re-orthogonalizing the generated vectors. He showed this process is comparable to the methods of Householder or Givens (proposed in the late fifties), but the latter are more economical. He then concluded: "it is difficult to think of any reason why we should use Lanczos' method in preference to Householder's". This illustrates that the Lanczos method was commonly viewed as a direct reduction method at that time, and from that point of view Wilkinson's remarks were quite correct.

At about the same time as Lanczos, Arnoldi (1951) [1] gave a reduction algorithm for unsymmetric matrices. This was basically the same algorithm as Lanczos' algorithm for symmetric matrices, with the difference that each new basis vector had to be orthogonalized with respect to all previous basis vectors. In this way A is reduced by an orthogonal similarity transformation to upper Hessenberg form. Arnoldi's method suffers far less from numerical instability, depending on how well the orthogonalization process is carried out. But the method is more expensive than the Householder reduction, making it less attractive, as a direct method, for large (dense) matrices.

A very important notion was the recognition that matrices could be reduced, by orthogonal transformations, in a finite number of steps, to some special reduced form that lends itself more efficiently to further computations. In particular, a symmetric matrix can be reduced to tridiagonal form by Jacobi-rotations, provided that these rotations are restricted to annihilate entries of A outside its tridiagonal part. This was suggested by Givens in 1954 [45], and in this connection the Jacobi-rotations are also called Givens rotations. A few years later, Householder, in 1958 [63], discovered that complete columns of A could be reduced to zero, outside the tridiagonal part, by the more efficient Householder reflections. These are well-chosen orthogonal rank-one updates of the form

$$\left(I-\frac{2}{v^{\mathrm{T}}v}vv^{\mathrm{T}}\right);$$

these are discussed in [139]. The Householder method has become the method of choice for the reduction of matrices to tridiagonal form on serial computers. Thus for eigenproblems, a symmetric matrix can be reduced by orthogonal similarity transformations to tridiagonal form and unsymmetric matrices can be transformed to upper Hessenberg form.

By 1960, the eigenvalue problem for a symmetric tridiagonal matrix was solved by using the Sturm sequence property for successive subdeterminants.<sup>5</sup> The corresponding eigenvectors were solved by inverse iteration. The whole process is described in Givens' papers. A complete and thorough analysis for the Givens and Householder reductions and for the use of the Sturm sequences, is given in Wilkinson's book, which was the numerical linear algebra bible (Old Testament) for a long time.

As we have already shown in the section on the Power method, the QR method is, for determining the complete set of eigenvalues and eigenvectors, a superior technique. At the time that Wilkinson's book appeared, the blossoming of the QR method had just begun. Wilkinson devoted much attention to this method, but not as the method of choice for symmetric problems. We quote from Parlett [100, p. 172]: "Yet it was not invented until 1958–1959 and was not appreciated until the mid-1960s. The key idea came from Rutishauser with his construction of a related algorithm called LR in 1958". Whereas Wilkinson's book was the reference for eigenvalue problems in the period 1960–1980, after 1980, Parlett's book *The Symmetric Eigenvalue Problem* became the main source, at least for symmetric problems. Comparison of the two books clearly shows the progress made in this field.

The use of the QR (a mathematical equivalent is the QL algorithm) algorithm began with the work of Francis, [40] who recognized in 1961-1962 that a QR iteration maps a Hessenberg matrix to a Hessenberg matrix again, and this makes the process economical and also adds to stability since the zero elements need not be computed. Furthermore, Francis cleverly implicitly used origin shifts, and these can be carried out very economically for Hessenberg matrices. Kublanovskaja, in 1961 [74], also independently discovered the same process, but did not employ the invariance of the Hessenberg form. She deeply understood the mathematical aspects of the algorithm but was less concerned with the important computational details. The inclusion of Wilkinson shifts eventually makes the process very efficient, and for these shifts it can be proved that, for symmetric matrices, the process does not fail. The order of convergence for symmetric matrices is cubic (see, for instance, [61]), while for unsymmetric matrices it is quadratic [51,28]. These results rest on work of Ostrowski carried out in connection with the shift-and-inverse Power method (the RQI method, see that section). For a treatment of modern implementations of the QR method see [51, Chapter 7] or Demmel's book [28, Section 4.4.5]. These implementations incorporate techniques developed in the 1990s, such as (multiple) implicit shifts. This implicit shift technique leads to a rank-one perturbation of the Hessenberg structure, and this perturbation can be removed in its turn by a technique that is known as chasing the bulge: the perturbation (bulge) is chased down (and out of) the Hessenberg matrix with (double) Givens transformations. These chasing techniques were analysed in 1991 by Watkins and Elsner [152]. An important and complete overview of the practical QR algorithm can be found in [151].

<sup>&</sup>lt;sup>5</sup> This shows that working with the characteristic equation, if not explicitly constructed, is not a bad idea in some situations.

With respect to the relation between the QR and the Lanczos algorithms, we note the following. Lanczos' method focusses on one particular starting vector; QR starts with a full orthogonal basis and keeps it orthogonal through the Power iterations; the inclusion of shifts does not destroy the structure of the Hessenberg matrix. With Lanczos' method, a shift only makes sense in damping unwanted parts of the spectrum, but one cannot vary the shift during the process.

By 1970 the standard numerical eigenproblem, for dense matrices of not too large order, could be regarded as essentially solved and research shifted to larger problems and other eigenproblems. The next important problem to consider was the generalized eigenproblem  $Ax - \lambda Bx = 0$ . An obvious approach is to transform this to a standard eigenproblem by inverting either A or B, or to work with more complicated transformations, such as the *Cayley Transform*:  $(A - \sigma B)^{-1}(A - \tau B)$ . These approaches share the disadvantage that matrices A and B are not treated in the same way, which is most obvious from the simple transformation  $B^{-1}A$ . This leads to problems if B is singular or ill-conditioned, but as Stewart (1980) has pointed out, this does not necessarily mean that the given eigenproblem is ill-conditioned because if A is well-conditioned then the pencil  $B - \mu A$  may be well behaved (small perturbations in A and B lead to small perturbations in  $\mu$ . In Stewart's analysis the matrices are treated symmetrically; in particular he suggests considering the pencil  $\sigma A - \tau B$ , and regarding multiples of  $(\sigma, \tau)$ , for which the determinant  $|\sigma A - \tau B|$  vanishes, as generalized eigenpairs; see [129, Chapter VI] for more details. This approach, of course, requires a different reduction and this is accomplished by the Generalized Schur Decomposition, proposed by Moler and Stewart in 1973. This says that for arbitrary square A and B there exist unitary Q and Z such that  $Q^*AZ = T$ and  $Q^*BZ = S$  are upper triangular. For real matrices, the arithmetic can be kept real, but then the reduced matrices are quasi-triangular (that is  $2 \times 2$  nonzero blocks along the diagonal may occur). Moler and Stewart [91] also proposed a stable algorithm to accomplish the reduction to (quasi) triangular form and this is known as the OZ algorithm. Major modern software packages include software for the QZ algorithm. For perturbation analysis, we refer to Stewart and Sun [129].

After the 1970s, the eigenproblem for dense matrices of moderate order seemed to be solved and further improvements were not expected, especially for symmetric dense matrices. However, with the ever increasing demand for higher efficiency and/or better accuracy, things changed from time to time. In 1981, Cuppen [25] proposed a divide and conquer algorithm for the solution of the eigenproblem for symmetric tridiagonal matrices. The idea was to split the tridiagonal matrix in two blocks, each of half the original size, plus a rank-one update. Cuppen showed how the eigenproblems for each of the blocks could be combined for the original full problem by exploiting the rank-one update property, which led to the solution of a secular equation. Initially, this approach was not seen as a competitive algorithm by itself for general matrices of modest dimensions, although Cuppen recognized that his algorithm was asymptotically (much) faster than QR. Further investigations by others were made on account of promising parallel properties. A major problem was that the original algorithm suffered from instabilities, especially for the eigenvectors belonging to close eigenvalues. Some scaling problems were rectified by Dongarra and Sorensen in 1987 [32], but the "right" implementation, according to Demmel [28, Section 5.3.3] was not discovered until 1992 and published in 1995, by Gu and Eisenstat [55]. Meanwhile, software for this algorithm found its way into LAPACK and ScaLAPACK. As stated by Demmel again, the divide and conquer approach is now the fastest algorithm for computing all eigenvalues and eigenvectors of a symmetric matrix of order larger than 25; this also holds true for nonparallel computers. If the subblocks are of order greater than 25, then they are further reduced; else, the QR algorithm is used for computing the eigenvalues and eigenvectors of the subblock. For a full treatment of the modern variant of the divide and conquer method, we refer to Demmel's book [28]. A recent discussion on parallel implementation aspects of this method can be found in [134].

There are still niches for other methods for dense symmetric matrices. Wilkinson advocated the bisection method for tridiagonal matrices if only a small subset of the eigenvalues is wanted. Inverse iteration may then be used to determine the corresponding eigenvectors. For the bisection method, based on the Sturm sequence property, the reader may find classical material in [154], a modern treatment of the inverse iteration (considered suspect by many because of the ill-conditioning of the shifted matrix) can be found in [100].

#### 8. Iterative methods

Soon after its introduction in 1952, it was recognized that the Lanczos method was not a panacea for eigenproblems. The method showed strange behaviour, because of rounding errors, and in an influential paper by Engeli et al. in 1959 [37], it was shown by careful experiments that the theoretical finite termination within *n* steps, had no practical meaning. For a discretized biharmonic problem of order 64, they observed that hundreds of steps where necessary in order to obtain the 64 eigenvalues (together with extraneous other values). Wilkinson also analyzed the method and showed that it was forward unstable, which seemed to mark more or less the end of the Lanczos method. It was Paige, who showed in 1971 [97] that the Lanczos method could be used in a truly iterative way in order to obtain correct eigenvalue information. The crux of his analysis is that the observed loss of orthogonality in the Lanczos process, the source of all problems in the method, marked the convergence of an eigenpair; and most remarkably, it did not prevent convergence of other eigenpairs. This loss of orthogonality, by the re-introduction of components of the converged eigenvector to the process, led to duplication of the converged eigenpair in the reduced tridiagonal matrix. The main effect on the convergence of the left eigenpairs seemed to be some delay in the process in exact computation. Paige's analysis spurred much activity in this field and eventually the Lanczos method became a powerful tool and the method of choice for large sparse symmetric matrices, from 1980 on.

We mention the following major steps that led to improvements in the method, and to a better understanding. Parlett and Scott [103] proposed removing the delaying effects on convergence by a *selective* orthogonalization process for the Lanczos vectors. Also, the determination of converged eigenvalues became easier to identify by efficiently computing upper bounds for the residual of an eigenpair. Kaniel (1966) [69] derived upper bounds for the error in an eigenvalue approximation (the so-called *Ritz values*, the eigenvalues of the reduced tridiagonal matrix). These upper bounds have no direct practical implication, since they are in terms of unknown quantities associated with the matrix, such as gaps between eigenvalues relative to the span of the eigenvalues. However, these upper bounds are, in a general sense, rather sharp and can be used for the study of the convergence behaviour of the Ritz values. Later, Saad (1980) [114] refined these upper bounds. The convergence behaviour of Ritz values can be quite irregular; a temporary (almost) stagnation of the process can take place (also called *misconvergence*). Many of these effects were studied carefully, and explained, in a paper by Van der Sluis and Van der Vorst [142], through a rather complicated model for the Ritz values. This model, however, seemed to be necessary in order to show all the intricate dependencies in the apparently simple Lanczos process. In their paper, it is also shown that the convergence behaviour of the Ritz values is superlinear. No matter how irregular the rate of convergence may seem, on average it becomes faster as the iteration proceeds. Parlett also studied the so-called (occasional) misconvergence of the Ritz values [101]. His model is simpler, but does not explain all effects, such as the possible length of the stagnation phase. An algorithm ready for implementation was published by Parlett and Reid [102]; this algorithm computes upper bounds for the errors in the Ritz values, and exploits the fact that we have to compute these for successive tridiagonal matrices. We are not aware of a publicly available implementation, although we know that the process has been implemented for local use.

Note that, strictly mathematically speaking, the Lanczos process is finite and thus it is not correct to use a notion such as "convergence" or even "superlinear convergence". However, in finite precision and for large values of the order of the matrix, the Ritz values will become close to an eigenvalue and for practical purposes the method behaves like a truly convergent process.

In our discussion of the Power method, we showed how the use of a set of starting vectors arose quite naturally. This is also possible for the Lanczos (and Arnoldi) method, and this approach leads to block and banded Lanczos (and Arnoldi) methods [47,141,108].

For unsymmetric matrices it took longer for similar methods to gain popularity. An influential paper, that helped to promote Arnoldi's method as a useful tool, was published by Saad [115]. The Arnoldi method, for orthogonal reduction to upper Hessenberg form, was not only too expensive if one wanted to know only a few eigenpairs, it also suffered from poor convergence for specific eigenvalue distributions. Well-known is the Saad-Schultz example [117], which is a permuted identity matrix. The method leads to trivial approximations after the first n-1 steps, and after n steps all eigenpairs suddenly appear. This however, is at a much higher cost than for Householder's reduction. For this reason, the unsymmetric Lanczos process, also referred to as the two-sided Lanczos method, received some attention. Initially, the method was notorious for its break-down possibilities, its behaviour in finite precision arithmetic, and the fact that the reduction operators to tridiagonal form are nonorthogonal. Cullum and Willoughby, in 1986 [24], presented a code based on the two-sided Lanczos method, in which they solved a number of practical problems; this included a clever trick for identifying the spurious eigenvalues due to rounding errors. The code gained some popularity, for instance for plasma-physics eigenvalue computations [21]. Parlett and co-workers [104] introduced the concept of "look-ahead", mainly in order to improve the numerical stability of the process. The look-ahead strategy, introduced in order to prevent breakdown, was further perfected by Freund and Nachtigal, in 1996 [41]. They published a code based on quasi-minimization of residuals, and included look-ahead strategies, in which most of the original Lanczos problems were repaired (but the non-orthogonal reductions were still there). Gutknecht [57] published a thorough theoretical overview of the two-sided Lanczos algorithm and exploited its relation to Padé approximations. This gave a better understanding of look-ahead strategies and the convergence behaviour of the method (in the context of solving linear systems). Block variants of the two-sided Lanczos process were discussed in Day's Ph.D. thesis in 1993; for a further description of the algorithms see [3].

Almost simultaneously, there were efforts to make the Arnoldi method more practical. We mention firstly polynomial preconditioning, discussed extensively in Saad's book [116], which damps unwanted parts of the spectrum, and secondly, sophisticated restarting strategies. The method becomes effective for matrices for which shift-and-invert operations can be applied for given vectors. But the many (increasingly expensive) iterations for relevant problems were a bottleneck. A real breakthrough for the Arnoldi method was realized by Sorensen [125], in 1991, with the so-called *Implicit Restart Technique*. This is a clever technique by which unwanted information can be filtered away from the process. This leads to a reduced subspace with a basis, for which the matrix still has a Hessenberg form, so that Arnoldi's process can be continued with a subspace (rather than with a single vector as with the more classical restart techniques).

The Arnoldi iteration procedure is often carried out with the shift-and-invert approach. For instance, when solving the generalized eigenproblem  $Ax = \lambda Bx$ , the method is applied to the operator  $(A - \sigma B)^{-1}B$ . One step further is the *Cayley* transform  $(A - \sigma B)^{-1}(A - \tau B)$ , which can be used for emphasizing eigenvalues (near  $\sigma$ ) and for damping of eigenvalues (near  $\tau$ ). Both techniques require expensive operations with an inverted operator, but the advantage is much faster convergence. Meerbergen and Roose [89] considered the use of inexact Cayley transforms, realized by a few steps of an iterative method, for Arnoldi's method. This technique bears a close relation to polynomial preconditioning. Ruhe [110] considered a more general shift-and-invert transform, and so-called *Rational Krylov Subspace (RKS) Method*:

$$(\delta_j A - \gamma_j B)^{-1} (\sigma_j A - \rho_j B),$$

in which the coefficients may be different for each iteration step j. It has been shown that by generating a subspace with this operator, the given problem can be reduced to a small projected generalized system

$$(\zeta K_{i,i} - \eta L_{i,i})s = 0,$$

where  $K_{j,j}$  and  $L_{j,j}$  are upper Hessenberg matrices of dimension *j*. This small system may be solved by the QZ algorithm in order to obtain approximate values for an eigenpair. The parameters in RKS can be chosen to obtain faster convergence to interior eigenvalues. For a comparison of RKS and Arnoldi, see [110,109].

In 1975, Davidson, a chemist, suggested an iterative method that had the idea of projection on a subspace in common with the Arnoldi method, but with the subspace chosen differently. Motivated by the observation that the matrices in his relevant applications were (strongly) diagonally dominant, Davidson computed the Ritz pairs (the eigenpairs of the projected matrix), computed the residual  $r = (A - \theta I)z$  for a pair of interest  $(\theta, z)$ , and proposed expanding the subspace with the vector  $(D_A - \theta I)^{-1}r$  (after proper orthogonalization with respect to the current subspace). The matrix  $D_A$  denotes the diagonal of the matrix A. For diagonally dominant matrices this approximates, in some sense, inverse iteration with Rayleigh quotient shifts. The Davidson method [26] became quite popular for certain applications. Although other approximations were suggested (see, e.g., [93]), its convergence behaviour for nondiagonal dominant matrices, or for poor initial starting vectors, was far from guaranteed. It was also puzzling that the "optimal" expansion  $(A - \theta I)^{-1}r$ , optimal from the inverse iteration point of view, led to the vector z, so that the method stagnated. As a consequence, Davidson's method was not able to find the eigenvalues of a diagonal matrix, a very unsatisfactory situation. The method was not very well understood by numerical analysts and as a consequence we find very little reference to it in the numerical analysis literature. Only after 1990, is there some serious analysis [20], almost simultaneously with a successful improvement of the method.

In 1996, Sleijpen and van der Vorst [121] suggested restricting the expansion of the current subspace to vectors from the space orthogonal to z, which restored a largely forgotten technique

used by Jacobi (in 1846). Jacobi took an appropriate unit vector for z, and attempted to find the missing component to make it an eigenvector from the space spanned by the remaining n-1 unit vectors. He did this by solving a correction equation for the matrix shifted by  $\theta$  and then restricted the correction to the subspace orthogonal to the chosen unit vector. This correction equation was solved by Gauss–Jacobi iterations and after each two iterations, Jacobi updated the value for  $\theta$ . Sleippen and van der Vorst suggested updating z as well and using the update vector for a subspace. This new Jacobi-Davidson method combined Davidson's idea of taking a different subspace with Jacobi's idea of restricting the search of an update to  $z^{\perp}$ . We refer to Sleijpen and van der Vorst [121] for details. A variant of the technique, in which the correction is approximately solved with one single invert step with a preconditioner, was suggested by Olsen et al. in 1990 [94]. An exact solution of the correction equation, or an approximate solution of high accuracy, leads to cubic convergence for a properly selected sequence of  $\theta$ 's if A is symmetric, and to quadratic convergence in the unsymmetric case. In the following years, it became clear how to efficiently implement the method with preconditioning [122] and how it could be applied to various other eigenproblems, amongst which are generalized eigenprobelms [39] and quadratic eigenproblems [123]. Thus, the transformation of these generalized eigenproblems to standard forms can be avoided. The Jacobi-Davidson method is attractive for large sparse eigenproblems for which shift-and-invert operations are too expensive, and for more unusual eigenproblems. The problem of identifying effective preconditioners for the correction matrix for larger classes of matrices is still largely open.

It is well-known that subspace methods lead to eigenvalue approximations that tend to converge towards exterior eigenvalues and that approximations for interior eigenvalues are difficult to obtain. In principle, it is easy to obtain these by working with  $A^{-1}$ , but this may be expensive. It is also possible to obtain eigenvalue approximations that converge (slowly) to the eigenvalues of A closest to the origin, from the subspaces generated by A. We explain this for the Arnoldi process. The Arnoldi process leads to

$$AV_m = V_{m+1}H_{m+1,m},$$

where  $H_{m+1,m}$  is an upper Hessenberg matrix with m + 1 rows and m columns. This means that we have a basis for the space with basis vectors  $Av_j$  and, using the above relation, this basis can be easily transformed into an orthogonal basis. This orthogonal basis can be used for the projection of  $A^{-1}$ , and multiplication by the inverse can be avoided, since all basis vectors have a factor A in common. The exterior eigenvalues of the projected  $A^{-1}$ , that is the inverses of interior eigenvalues of A, converge (slowly) to the exterior eigenvalues of  $A^{-1}$ . This way of approximating interior eigenvalues has received some attention in the 1990s. In [98], these eigenvalue approximations, in connection with the related Lanczos process, were called Harmonic Ritz values, and some nice relations for Harmonic Ritz values for symmetric indefinite matrices are given in that paper. Harmonic Ritz values had already been studied from a different viewpoint by other authors. Freud [42] studied them as the zeros of the GMRES and MINRES iteration polynomials. Morgan [92] had observed that the Harmonic Ritz values and vectors are very suitable for restarting purposes if one wants to compute interior eigenvalues with subspaces of restricted dimension. In [121,120], the Harmonic Ritz values are considered in connection with the Jacobi–Davidson process for the selection of proper shifts.

## 9. Related topics

#### 9.1. The singular value decomposition

The singular value decomposition plays an essential role in many situations in which we want to decide between relevant and less relevant information. In addition to traditional applications such as regularization of linear least-squares problems, and the determination of the numerical rank of a matrix, there are applications to the reduction of information for images and information retrieval from large data bases. A headline contribution to the SIAM News by Berry and Dongarra, summer 1999, showed that the SVD can even help to reduce work in the organization of a conference. Interesting and unusual applications are also shown in a paper by Moler and Morrison [90]. A modern treatment of the numerical aspects of the SVD can be found in Golub and Van Loan's textbook [51] and we have taken most of our information from that source. Demmel [28] gives a good treatment of the important implementational aspects of the SVD.

Since the square of the singular values and the right singular vectors are the eigenpairs for the matrix  $A^{T}A$  (for the left singular vectors this holds with  $AA^{T}$ ), it is not surprising that the numerical treatment of the SVD has many relationships to algorithms for symmetric eigenproblems. This is most visible in Demmel's book, where the discussion on modern SVD algorithms almost parallels the symmetric eigenvalue problem discussion [28, p. 211 and 241]. We note here that working directly with either  $A^{T}A$  or  $AA^{T}$  is not satisfactory for stability and complexity reasons, and this makes separate treatment of the numerical SVD necessary.

The origins of the singular value decomposition go back to the late 19th century, with work of Beltrami in 1873. Stewart [128] gave a historic overview of the SVD. The important numerical developments on the SVD started with work of Golub and Kahan in 1965 [49]. This led first to a contribution to the famous ACM-collection as Algorithm 358 [15] and later to the basis of the EISPACK and LAPACK routines in the Wilkinson–Reinsch collection [158, pp. 1334–1351]. The key trick in the numerical computation of the SVD is, instead of tridiagonalizing  $A^{T}A$ , to bidiagonalize the matrix A. Then, with the bidiagonal reduced forms obtained by orthogonal transformations, one can make variants of QR, divide and conquer, and bisection techniques. The choice between these techniques can be made as above for the symmetric eigenproblem. In particular, for matrices of order larger than 25, the divide and conquer approach is currently regarded as the fastest option [28, p. 241].

### 9.2. Nonlinear eigenproblems and related problems

Standard and generalized eigenproblems arise, for instance, in the study of conservative mechanical systems, governed by Lagrange's equations of small free motion. According to Rogers [106], Rayleigh (in 1873) could not apply his technique for nonconservative systems (systems with a damping term). The well-known technique for the numerical solution of the resulting quadratic eigenproblem is to rewrite it as a generalized eigenproblem [43,77]. That is

 $\lambda^2 M x + \lambda C x + K x = 0$ 

is equivalent to

 $Az = \lambda Bz$ 

with

$$A = \begin{bmatrix} 0 & I \\ -K & -C \end{bmatrix}, \quad B = \begin{bmatrix} I & 0 \\ 0 & M \end{bmatrix}, \quad z = \begin{bmatrix} x \\ \lambda x \end{bmatrix}.$$

Of course, algorithms and theory for the generalized eigenproblem can be used directly, but the unsatisfactory aspect of this is that we have to double the dimension of our spaces. Also, if the generalized eigenproblem is solved approximately by some iteration technique, then it is not straight forward to reduce the information from the double-dimensioned space to the original space. Duffin [34] seemed to be the first to generalize the Rayleigh-Ritz principle for the quadratic eigenproblem for symmetric K, C, and M, and A positive definite. Since then, the quadratic eigenproblem, and higher-order polynomial problems, have received attention in the numerical literature. Rogers [106], in 1964, considered a more general quadratic eigenproblem and used minimax principles for the investigation of such problems. In the late 1960s and early 1970s algorithms that avoid the linearization step appeared. We mention the work of Lancaster [77], Kublanovskaja [75,76], and Ruhe [107]. The suggested algorithms are mostly variants of Newton's method. More recently, in 1995, Guo et al. [56] described several iterative methods, that can be regarded as a fixed point iteration combined with the Lanczos method and a (simplified) Newton iteration. A backward error analysis for more general polynomial eigenproblems was given by Tisseur (1998 [133]), and a perturbation analysis for quadratic problems was published by Sun in 1999 [130]. In a paper that appeared in 1996, Sleijpen et al. [120] showed that the Jacobi-Davidson method could be applied in order to reduce a given polynomial problem in an *n*-dimensional space to a similar problem in a much lower *m*-dimensional space. The problem in the lower dimensional space can then be attacked by any of the previously mentioned approaches. For a quadratic equation from an accoustic problem, it was shown how this approach led successfully and efficiently to the desired eigenpair, for matrices of the order of about 240,000 [123]. For higher polynomial eigenproblems there has not been much experience to date. Bai [6] mentioned the need for algorithms for fifth-order polynomial eigenproblems in a review paper that appeared in 1995. Quite recently, in 1998, Heeg [58] showed how the Jacobi-Davidson approach could be successfully applied to fourth-order polynomial eigenproblems with complex matrices in the study of instabilities of attachment-line flows for airfoils.

Other than these polynomial eigenproblems, there is a wide variety of problems that are associated with the determination of invariant subspaces. Since the standard and generalized eigenproblems became more familiar and more or less routinely solvable, the more difficult problems received more attention. Among these problems we mention the following; the *Procrustes problem*: minimize  $||AY - YB||_F$  for given A and B over the manifold  $Y^*Y = I$ ; the determination of a nearest Jordan structure, and the problem of determining a simultaneous Schur decomposition for a set of perturbed matrices (under the assumption that the unperturbed matrices have a simultaneous Schur form). A nice overview of such problems, as well as software for the numerical solution is described by Edelman and Lippert [83].

#### 9.3. Pseudospectra

Eigenvalues are often used as a source of information on stability or convergence, and the question arises as to the validity of the information gained from these values. For example, during the 1990s it was realized (see, e.g., [138]) that eigenvalues alone do not govern the instability and transition to

turbulence of high Reynolds number fluid flows as had previously been thought. Many authors have studied the problem of sensitivity of the eigenvalues with respect to perturbations, see for instance [154,129,19]. These studies are usually related to perturbations caused by rounding errors, and not so much by the relevance of the eigenvalues due to the particular representation of a given problem, for instance the choice of basis.

Around 1987, Trefethen [140] began to emphasize this aspect of eigencomputations. He propagated the idea of inspecting the pseudospectra of a matrix as a relatively simple means for determining the significance of a particular part of the spectrum, without getting involved in complicated matters such as angles between eigenvectors or eigenspaces.<sup>6</sup> The definition of the pseudospectrum  $\Lambda_{\varepsilon}(A)$  for a matrix A is directly related to perturbations:

$$\Lambda_{\varepsilon}(A) \equiv \{ z \in \mathbb{C} : z \in \Lambda(A + E) \text{ for some } E \text{ with } \|E\| \leq \varepsilon \}.$$

The pseudospectra are usually shown graphically as a set of level curves for various values of  $\varepsilon$ . The level curves, or contour integrals, are more apparent from the original definition of the  $\varepsilon$ -pseudospectrum, in terms of the norm of the resolvent  $(zI - A)^{-1}$ :

$$\Lambda_{\varepsilon}(A) \equiv \{ z \in \mathbb{C} \colon ||(zI - A)^{-1}|| \ge \varepsilon^{-1} \}$$

with the convention  $||(zI - A)^{-1}|| = \infty$  for  $z \in A(A)$ .

For symmetric matrices, the pseudospectra of A is a collection of discs around the eigenvalues of A (note that the perturbation E need not be symmetric). For unsymmetric matrices the pseudospectra can be any collection of curves around the set of eigenvalues of A. These level curves may give information that is hidden by the information provided by the eigenvalues themselves. For instance, when studying stability of integration methods for systems of ODEs, or in bifurcation problems, the eigenvalues may be in a proper region, for instance, in the left-half plane, while the level curves, even for small values of  $\varepsilon$ , may intersect with the right-half plane. This suggests that it may be necessary to ask further questions about the problem. On the other hand, the pseudospectra may not tell the full story. For instance, the sensitivity problems may be due to a single pair of ill-conditioned eigenvectors for which the more global level curves are too pessimistic. It may be the case that it is not realistic to assume equal perturbations for all matrix entries, but nevertheless the pseudospectra focus attention on critical places in the spectrum. A nice introduction to the relevance of pseudospectra is given in [135], where the pseudospectra are actually computed and discussed for a number of matrices.

Due to the nature of computing pseudospectra, this useful tool is often restricted to matrices of relatively moderate size and one has to be careful in generalizing the insights gained from smaller problems to larger problems which are similar. More recently, tools have become available for computing the pseudospectrum of large sparse matrices. Carpraux et al. [16] proposed an algorithm for computing the smallest singular value of zI - A, that is based on Davidson's method using ILU preconditioning. Lui [84] (see also [13]) suggested using the Lanczos method in combination with continuation techniques. This is a plausible approach, since we need to do the computation for many values of z, well-distributed over the region of interest, in order to obtain a complete picture of the pseudospectra. We know that currently such tools are being used for the analysis of instability problems of large sets of ODEs, related to climate modelling, but results have not yet been published.

<sup>&</sup>lt;sup>6</sup> The notion had been defined by others earlier, the earliest of them, according to Trefethen, being Landau [79].

Valuable information on pseudospectra can also be derived from the Arnoldi iteration obtained, for instance, with ARPACK [81]. For an up-to-date overview on pseudospectra results, see Trefethen's 1999 paper [136].

#### 9.4. Homotopy methods

The subspace methods that we have discussed above are often applied in combination with shiftand-invert operations. This means that if one wants to compute eigenvalues close to a value  $\sigma$ , then the methods are applied to  $(A - \sigma I)^{-1}$ . As we have seen, the Jacobi–Davidson method can be interpreted as an inexact shift-and-invert method, since the invert step is usually approximated by a few steps of some convenient preconditioned inner iteration method.

Another method related to these inexact shift-and-invert approaches is the homotopy approach which has received some attention in the 1990s. The idea is to compute some of the eigenvalues of a perturbed matrix A + E, when the eigenvalues of A are known, or can be relatively easily computed. In order to this we use the homotopy H(t) = A + tE,  $0 \le t \le 1$ . If eigenpairs of  $H(t_0)$ are known, then they are used as approximations for those of  $H(t_0 + \Delta t)$ . These approximations are improved by a convenient subspace iteration (cf. [85]). Rayleigh quotient iterations are used for symmetric A and E (see references in [85] for earlier work on homotopy for eigenproblems). For the Rayleigh quotient iteration one needs to solve systems like  $(H(t_0 + \Delta t) - \lambda I)y = x$ , where  $(\lambda, x)$ represents the current approximation for an eigenpair of  $H(t_0 + \Delta t)$ . In the context of large sparse matrices, it may be undesirable to do this with a direct solver, and in [85] the system is solved with SYMMLQ [99]. Of course, one could restrict oneself to only a few steps with SYMMLQ, and then try to accelerate the inexact Rayleigh quotient steps, as is done in the Jacobi-Davidson method. This indicates relations between these different approaches, but as far as we know, these relations have not yet been explored. In [85], it is observed that SYMMLQ may have difficulty in converging for the nearly singular system  $(H(t_0 + \Delta t) - \lambda I)y = x$ , and it is suggested that the situation would be improved by applying the Rayleigh quotient iteration to the approximately deflated matrix  $H(t_0 + \Delta t) + xx^T$ . (The term approximately deflated is used to indicate that x is only an approximation to the desired eigenvector.) Note that similar deflation procedures are incorporated in the Jacobi–Davidson process. The whole procedure is repeated for successive increments  $\Delta t$ , until the final value t = 1 is reached. In [85] an elegant approach is followed for the selection of the step size  $\Delta t$ .

The homotopy approach lends itself quite naturally to situations where the matrix A varies in time, or where it varies as a linearization of a nonlinear operator, as in bifurcation problems. Another example of an interesting problem is the Schrödinger eigenvalue problem [85]

$$-\Delta u + f u = \lambda u$$

in the unit square in two dimensions with homogeneous Dirichlet boundary conditions. With the usual finite difference approximations on a uniform grid, this leads to the discrete Laplacian for  $-\Delta u$ , for which we know the eigensystem.

In [159], the homotopy approach is used for symmetric generalized eigenproblems, very much along the lines sketched above. The application to real unsymmetric eigenproblems is considered in [86].

In [82], the homotopy approach is suggested as a means of realizing a divide and conquer method for unsymmetric eigenproblems, as an alternative for the solution of the secular equation for symmetric problems (which cannot be used for unsymmetric problems).

#### 9.5. Miscellaneous

Subspace methods such as Simultaneous Iteration, RQI, Lanczos, Arnoldi, and Davidson for large sparse systems are in general more powerful in combination with shift-and-invert techniques. This requires the solution of large sparse linear systems, since these methods only need operations with, for instance,  $(A - \sigma I)$  in order to compute matrix vector products  $(A - \sigma I)^{-1}v$ . This has led to the question of how accurately these operations have to be carried out in order to maintain a reasonable rate of convergence. For instance, for the Power method, this approximation technique, or preconditioning, can be described as follows for the standard eigenproblem  $Ax = \lambda x$ . For each iteration one computes, for the current iteration vector  $x^{(i)}$  the Rayleigh quotient  $\mu^{(i)} = x^{(i)*}Ax^{(i)}/x^{(i)*}x^{(i)}$ . Instead of computing  $x^{(i+1)}$  with  $Ax^{(i)}$ , the defect  $r = Ax^{(i)} - \mu^{(i)}x^{(i)}$  is computed and this defect is multiplied by the preconditioner K. The new vector  $x^{(i+1)}$  is then computed as a normalized linear combination of  $x^i$  and Kr. Note the resemblance to Davidson's approach.

The convergence of this basic algorithm, including its use for the generalized eigenproblem, has been studied quite extensively in the Russian literature, starting in about 1980 in the work of D'yakonov [35] and others. Knyazev [72] gave an excellent overview of the work in the Russian literature on *preconditioned eigensolvers*. The study of these preconditioned iteration techniques is relevant also to the understanding of inexact, or preconditioned, forms of shift-and-invert Lanczos (proposed by Morgan and Scott [93]), inexact Arnoldi (work by Meerbergen [87]) and Davidson variants, including the Jacobi–Davidson method. A presentation of various iteration techniques including preconditioning was given by Knyazev in [71]. Smit [124, Chapter 4] studied the effect of approximate inverses on the convergence of the RQI method. For a discussion on inexact Krylov methods, see Meerbergen and Morgan [88]. Preconditioning in relation to the Jacobi–Davidson method is discussed in [122]. Note that in all these algorithms, the preconditioner is used for the computation of a promising new direction vector; the given eigenproblem is untouched. This is different from the situation with linear solution methods, where preconditioning is used to transform the given system to one that can be handled more efficiently.

## 10. Software

The history of reliable high-quality software for numerical linear algebra started with the book edited by Wilkinson and Reinsch, the *Handbook for Automatic Computation, Vol. 2, Linear Algebra*, published in 1971. This book contained a number of articles that had appeared previously in Numerische Mathematik, which described state-of-the-art algorithms for the solution of linear systems and eigenproblems. All these articles contained implementations in Algol60. Most of these algorithms are still alive albeit in other languages. Algol60 was a computer language that gained some popularity in academia, mostly in Europe, but it was not as fast as Fortran on most machines and it did not gain a foothold in the slowly emerging large industrial codes (the majority of which were written in Fortran or even in assembler language). For this reason, groups in the USA started the development of two influential software packages LINPACK and EISPACK in the early 1970s. These packages started as transcriptions of the major part of the Wilkinson and Reinsch collection: LINPACK covered the numerical solution of linear systems; EISPACK concentrated on eigenvalue problems. The most prominent omissions from these packages were iterative solution methods: the

conjugate gradient method was not included in LINPACK and the Jacobi method was not included in EISPACK. At that time the Lanczos and Arnoldi methods were not even considered as candidates, because they were viewed either as direct methods, and in that sense not competitive with the then available methods, or as iterative methods that could not be safely automated [68]. The Lanczos method was only considered, in that time, as a safe iterative method provided one did complete re-orthogonalization (cf. [48]).

The "Wilkinson and Reinsch" procedures can also be viewed as prototypes for eigenvalue routines in the bigger software packages NAG and IMSL, and in the widely available software package MATLAB. EISPACK was replaced in 1995 by LAPACK, in the words of Golub and Van Loan (1996): "LAPACK stands on the shoulders of two other packages (viz: LINPACK and EISPACK) that are milestones in the history of software development". A more recent development along these lines in ScaLAPACK [11] which aims to provide close to optimal software for modern parallel computers.

We have already mentioned MATLAB in passing; the impact of this computing environment in the scientific computing world has been tremendous. MATLAB provides state-of-the-art software for all sorts of numerical linear algebra computations and has become the de facto standard for coding and testing in the 1990s. Its impact on the development of this field can hardly be overestimated.

As already indicated, in 1970 there were few robust and well-understood iterative methods available, and mainly for that reason, these methods were not included in the packages constructed then. They are still not available in their successors, with the exception of MATLAB (since 1998, MATLAB has had iterative methods for eigenproblems available). It was soon clear that the powerful "direct" methods, based on reduction to some special form, had their limitations for the large sparse matrices that occur in the modelling of realistic stability problems, and there was a heavy demand for methods that could, at least partially, handle big problems. Cullum and Willoughby filled this gap, in 1985, with their software based on the two-sided Lanczos procedure [22–24]. Of course, they realized the intricacies of the Lanczos method, and they advocated the use of iterated Rayleigh quotient steps for improvement of the information of the Lanczos output. They also provided software for this updating step (their software was available some years before 1985, but its publication in book form took place in that year).

Freund and Nachtigal proposed in 1992, a variant of the two-sided Lanczos process that improved convergence properties using a quasi minimization step; they also included sophisticated look-ahead facilities. The QMR method could also be used for eigenvalue computations, and they provided software through Netlib for this purpose, see QMRPACK [41]. Algorithms and software for adaptive block-variants of the two-sided Lanczos algorithm (ABLE) have been described by Bai et al. [3].

Because of the improvements made to the Arnoldi method, in particular the implicit restart technique, it became feasible to exploit the attractive orthogonal reduction properties in an efficient manner. This was realized in the ARPACK software, for which the User's guide was published, by Lehoucq et al. in 1998 [81]. The package was in existence and available to users a few years earlier. At present, ARPACK seems to be the default choice for large sparse eigenproblems, provided that either it is possible to implement shift-and-invert operations efficiently, or that unwanted parts of the spectrum can be damped by a fixed (polynomial) preconditioner. The parallel implementation of ARPACK is referred to as P\_ARPACK, it is portable across a wide range of distributed memory platforms. The parallelism is mainly in the matrix vector products and the user has full control over this trough the reverse communication principle. For more details, see [81,31]. Kågström et al. have provided software for the generalized upper staircase Schur form. The software, called GUPTRI, can be used for the analysis of singular pencils  $A - \lambda B$ . This reduction admits more general cases than the QZ decomposition, for instance A and B may be singular. This software has been described in [4]. See also this reference for pointers to recent work by Kågström.

For a thorough overview of modern algorithms, software and pointers to sources for further information see [4].

## 11. Epilogue

For symmetric matrices, the eigenproblem is relatively simple, due to the existence of a complete orthogonal eigensystem, and the fact that all eigenvalues are real. These properties are exploited in the most efficient numerical methods, and the symmetric eigenproblem may be considered as solved: for small matrices  $n \leq 25$  we have the QR method, one of the most elegant numerical techniques produced in the field of numerical analysis; for larger matrices (but smaller than a few thousand), we have a combination of divide and conquer with QR techniques. For the largest matrices, there is the Lanczos method, which in its pure form is strikingly simple but which conceals so many nice and attractive properties. All methods have a relatively long history. The theoretical aspects of the computations (convergence and perturbation theory) are relatively well understood.

For unsymmetric matrices the picture is less rosy. Unfortunately, it is not always possible to diagonalize an unsymmetric matrix, and even if it is known that all eigenvalues are distinct, then it may be numerically undesirable to do this. The most stable methods seem to be based on the Schur factorization, that is for each  $n \times n$  matrix A, there exists an orthogonal Q, so that

$$Q^*AQ = R,$$

in which R is upper triangular. Apart from permutations and signs, the matrix Q is unique. The Schur factorization reveals much of the eigenstructure of A: its diagonal elements are the eigenvalues of A and the off-diagonal elements of R indicate how small the angles between eigenvectors may be. For matrices not too large, QR is the method of choice, but for larger matrices the picture is less clear. Modern variants of the Arnoldi method seem to be the first choice at the moment, and, if approximations are available, the Jacobi–Davidson method may be attractive. There is still a lot to investigate: if the matrix is nonnormal, that is, if the eigenvectors do not form a unitary set, then what kind of meaningful information can we extract from a given matrix (invariant subspaces, angles between subspaces, distance to Jordan canonical form), and how can we compute this as accurately as possible? Much has been done, but even more remains to be done.

There are even more open problems as eigenproblems become more complicated: generalized eigenproblems, polynomial eigenproblems, nonlinear eigenproblems, etc.

Looking back over the past century, we see that the solution of the eigenproblem has given up some of its mysteries through the work of many devoted and highly talented researchers. Novices in the field should be aware that the modern algorithms, even the apparently simple ones, are the result of many independent "small" steps. The fact that many of these steps can now be regarded as "small" illustrates how theory has kept up the pace with computational practice, so that new developments can find their place in an expanding but still elegant framework. Astonishingly much has been achieved, both computationally and theoretically, in a concerted effort, but much more remains to be unravelled. It is our firm belief that eigenproblem research will remain a lively and useful area of research for a long time to come. We hope that this overview will help to motivate young researchers to make their contributions to solving pieces of the gigantic puzzles that remain.

#### Acknowledgements

We wish to thank our colleagues Michele Benzi, Jan Brandts, Mark Embree, Karl Meerbergen, Gerard Sleijpen, Nick Trefethen, and David Watkins, for their suggestions for improvements and corrections.

#### References

- [1] W.E. Arnoldi, The principle of minimized iteration in the solution of the matrix eigenproblem, Quart. Appl. Math. 9 (1951) 17–29.
- [2] Z. Bai, D. Day, Q. Ye, ABLE: An adaptive block Lanczos method for non-Hermitian eigenvalue problems, Technical Report Research Report 95-04, University of Kentucky, Lexington, KY, 1995.
- [3] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, H. van der Vorst, Templates for the Solution of Algebraic Eigenvalue Problems: a Practical Guide, SIAM, Philadelphia, PA, 1999.
- [4] Z. Bai, G.W. Stewart, SRRIT A FORTRAN subroutine to calculate the dominant invariant subspace of a nonsymmetric matrix, Technical Report 2908, Department of Computer Science, University of Maryland, 1992.
- [5] Z. Bai, Progress in the numerical solution of the nonsymmetric eigenvalue problem, Numer. Linear Algebra Appl. 2 (1995) 219–234.
- [6] V. Bargmann, C. Montgomery, J. von Neumann, Solution of linear systems of high order, Technical Report, Princeton, Institute for Advanced Study, 1946.
- [7] F.L. Bauer, Das Verfahren der Treppeniteration und verwandte Verfahren zur Lösung algebraischers Eigenwertprobleme, Z. Angew. Math. Phys. 8 (1957) 214–235.
- [8] F.L. Bauer, C.T. Fike, Norms and exclusion theorems, Numer. Math. 2 (1960) 137-141.
- [9] C.H. Bischof, The two-sided block Jacobi method on hypercube architectures, in: M.T. Heath (Ed.), Hypercube Multiprocessors, SIAM, Philadelphia, PA, 1987.
- [10] L. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, R. Whaley, ScaLAPACK User's Guide, SIAM, Philadelphia, PA, 1997.
- [11] E. Bodewig, Matrix Calculus, North-Holland, Amsterdam, 1956.
- [12] T. Braconnier, F. Chatelin, V. Frayssé, The influence of large nonnormality on the quality of convergence of iterative methods in linear algebra, Technical Report TR-PA-94/07, CERFACS, Toulouse, 1994.
- [13] T. Braconnier, N.J. Higham, Computing the field of values and pseudospectra using the Lanczos method with continuation, BIT 36 (1996) 422–440.
- [14] R.P. Brent, F.T. Luk, The solution of singular value and symmetric eigenvalue problems on multiprocessor arrays, SIAM J. Sci. Statist. Comput. 6 (1985) 69–84.
- [15] P.A. Businger, G.H. Golub, Algorithm 358: Singular value decomposition of a complex matrix, Comm. Assoc. Comp. Mach. 12 (1969) 564–565.
- [16] J.F. Carpraux, J. Erhel, M. Sadkane, Spectral portrait for non-Hermitian large sparse matrices, Computing 53 (1994) 301–310.
- [17] F. Chaitin-Chatelin, Is nonnormality a serious difficulty? Technical Report TR-PA-94/18, CERFACS, Toulouse, 1994.
- [18] F. Chatelin, Spectral Approximation of Linear Operators, Academic Press, New York, 1983.
- [19] F. Chatelin, Eigenvalues of Matrices, Wiley, Chichester, 1993; Masson, Paris, 1988.
- [20] M. Crouzeix, B. Philippe, M. Sadkane, The Davidson method, SIAM J. Sci. Comput. 15 (1994) 62-76.
- [21] J. Cullum, W. Kerner, R. Willoughby, A generalized nonsymmetric Lanczos procedure, Comput. Phys. Comm. 53 (1989) 19–48.

- [22] J. Cullum, R.A. Willoughby, Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Vol. I, Birkhaüser, Boston, MA, 1985.
- [23] J. Cullum, R.A. Willoughby, Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Vol. II, Birkhaüser, Boston, MA, 1985.
- [24] J. Cullum, R.A. Willoughby, A practical procedure for computing eigenvalues of large sparse nonsymmetric matrices, in: J. Cullum, R.A. Willoughby (Eds.), Large Scale Eigenvalue Problems, Amsterdam, North-Holland, 1986, pp. 193–240.
- [25] J.J.M. Cuppen, A divide and conquer method for the symmetric eigenproblem, Numer. Math. 36 (1981) 177–195.
- [26] E.R. Davidson, The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real symmetric matrices, J. Comput. Phys. 17 (1975) 87–94.
- [27] C. Davis, W. Kahan, The rotation of eigenvectors by a perturbation-iii, Soc. Ind. Appl. Math. J. Num. Anal. 7 (1970) 1–46.
- [28] J.W. Demmel, Applied Numerical Linear Algebra, SIAM, Philadelphia, PA, 1997.
- [29] J.W. Demmel, K. Veselić, Jacobi's method is more accurate than QR, SIAM J. Matrix Anal. Appl. 13 (1992) 1204–1246.
- [30] A. den Boer, De Jacobi-methode van 1845 tot 1990, Master Thesis, Department of Mathematics, Utrecht University, 1991.
- [31] J.J. Dongarra, I.S. Duff, D.C. Sorensen, H.A. van der Vorst, Numerical Linear Algebra for High-Performance Computers, SIAM, Philadelphia, PA, 1998.
- [32] J.J. Dongarra, D.C. Sorensen, A fully parallel algorithm for the symmetric eigenvalue problem, SIAM J. Sci. Statist. Comput. 8 (1987) 139–154.
- [33] J.J. Dongarra, D.W. Walker, Software libraries for linear algebra computations on high performance computers, SIAM Rev. 37 (1995) 151–180.
- [34] R.J. Duffin, A minimax theory for overdamped networks, J. Rational Mech. Anal. 4 (1955) 221-233.
- [35] E.G. D'yakonov, Iterative methods in eigenvalue problems, Math. Notes 34 (1983) 945–953.
- [36] P.J. Eberlein, Solution to the complex eigenproblem by a norm-reducing Jacobi-type method, Numer. Math. 14 (1970) 232–245.
- [37] M. Engeli, T. Ginsburg, H. Rutishauser, E. Stiefel, Refined Iterative Methods for Computation of the Solution and the Eigenvalues of Self-Adjoint Boundary Value Problems, Birkhäuser, Basel/Stuttgart, 1959.
- [38] D.G. Feingold, R.S. Varga, Block diagonally dominant matrices in generalizations of the Gerschgorin circle theorem, Pacific J. Math. 12 (1962) 1241–1250.
- [39] D.R. Fokkema, G.L.G. Sleijpen, H.A. van der Vorst, Jacobi–Davidson style QR and QZ algorithms for the reduction of matrix pencils, SIAM J. Sci. Comput. 20 (1996) 94–125.
- [40] J.G.F. Francis, The QR transformation: a unitary analogue to the LR transformation, Parts I and II Comput. J. 4 1961 265–272, 332–345.
- [41] R.W. Freund, N.M. Nachtigal, QMRPACK: a package of QMR algorithms, ACM Trans. Math. Software 22 (1996) 46–77.
- [42] R.W. Freund, Quasi-kernel polynomials and their use in non-Hermitian matrix iterations, J. Comput. Appl. Math. 43 (1992) 135–158.
- [43] F. Gantmacher, The Theory of Matrices, Vols. I and II, Chelsea, New York, 1959.
- [44] S. Gerschgorin, Über die Abgrenzung der Eigenwerte einer Matrix, Izv. Akad. Nauk SSSR, Ser. Fiz.-Mat. 6 (1931) 749–754.
- [45] W. Givens, Numerical computation of the characteristic values of a real symmetric matrix, Oak Ridge National Lab., ORNL-1574, 1954.
- [46] H.H. Goldstine, F.J. Murray, J. von Neumann, The Jacobi method for real symmetric matrices, J. Assoc. Comput. Mach. 6 (1959) 176–195.
- [47] G.H. Golub, Some uses of the Lanczos algorithm in numerical linear algebra, in: J.J.H. Miller (Ed.), Topics in Numerical Analysis, Academic Press, New York, 1973, pp. 173–184.
- [48] G.H. Golub, Sparse matrix computations: Eigenvalues and linear equations, Seminaries Iria, Inst. Rech. D'Informatique et d'Automatique, Rocquencourt, 1975, pp. 117–140.
- [49] G.H. Golub, W. Kahan, Calculating the singular values and pseudo-inverse of a matrix, SIAM J. Numer. Anal. 2 (1965) 205–224.

- [50] G.H. Golub, H.A. van der Vorst, 150 Years old and still alive: Eigenproblems, in: I.S. Duff, G.A. Watson (Eds.), The State of the Art in Numerical Analysis, Clarendon Press, Oxford, 1997, pp. 93–119.
- [51] G.H. Golub, C.F. Van Loan, Matrix Computations, The Johns Hopkins University Press, Baltimore, 1996.
- [52] G.H. Golub, J.H. Wilkinson, Ill-conditioned eigensystems and the computation of the Jordan canonical form, SIAM Rev. 18 (1976) 578–619.
- [53] J. Greenstadt, A method for finding roots of arbitrary matrices, Math. Tables Aids Comput. 9 (1955) 47-52.
- [54] R.T. Gregory, Computing eigenvalues and eigenvectors of a symmetric matrix on the ILLIAC, Math. Tables Aids Comput. 7 (1953) 215–220.
- [55] M. Gu, S.C. Eisenstat, A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem, SIAM J. Matrix Anal. Appl. 16 (1995) 172–191.
- [56] J.-S. Guo, W.-W. Lin, C.-S. Wang, Numerical solutions for large sparse eigenvalue problems, Linear Algebra Appl. 225 (1995) 57–89.
- [57] M.H. Gutknecht, A completed theory of the unsymmetric Lanczos process and related algorithms, Part I, SIAM J. Matrix Anal. Appl. 13 (1992) 594–639.
- [58] R. Heeg, Stability and transition of attachment-line flow, Ph.D. Thesis, Universiteit Twente, Enschede, The Netherlands, 1998.
- [59] P. Henrici, On the speed of convergence of cyclic and quasicyclic Jacobi methods for computing the eigenvalues of Hermitian matrices, SIAM J. Appl. Math. 6 (1958) 144–162.
- [60] K. Hessenberg, Auflösung linearer Eigenwertaufgaben met Hilfe der Hamilton-Cayleyschen Gleichung, Ph.D. Thesis, Technische Hochschule, Darmstadt, 1940.
- [61] W. Hoffmann, B.N. Parlett, A new proof of global convergence for the tridiagonal QL algorithm, SIAM J. Numer. Anal. 15 (1978) 929–937.
- [62] P. Horst, A method of factor analysis by means of which all coordinates of the factor matrix are given simultaneously, Psychometrika 2 (1937) 225–236.
- [63] A.S. Householder, Unitary triangularization of a nonsymmetric matrix, J. ACM 5 (1958) 339–342.
- [64] A.S. Householder, The Theory of Matrices in Numerical Analysis, Dover, New York, 1964.
- [65] C.G.J. Jacobi, Über ein leichtes Verfahren, die in der Theorie der Säcularstörungen vorkommenden Gleichungen numerisch aufzulösen, J. Reine Angew. Math. 30 (1846) 51–94.
- [66] B. Kågström, The generalized singular value decomposition and the general  $A \lambda B$  problem, BIT 24 (1985) 568–583.
- [67] B. Kågström, RGSVD: an algorithm for computing the Kronecker structure and reducing subspaces of singular  $A \lambda B$  pencils, SIAM J. Sci. Statist. Comput. 7 (1986) 185–211.
- [68] W. Kahan, B.N. Parlett, How far should you go with the Lanczos process, in: J.R. Bunch, D.J. Rose (Eds.), Sparse Matrix Computations, Academic Press, New York, 1976.
- [69] S. Kaniel, Estimates for some computational techniques in linear algebra, Math. Comput. 20 (1966) 369–378.
- [70] T. Kato, On the upper and lower bounds of eigenvalues, J. Phys. Soc. Japan 4 (1949) 334–339.
- [71] A.V. Knyazev, Preconditioned eigensolvers an oxymoron?, Electron. Trans. Numer. Anal. 7 (1998) 104–123.
- [72] A. Knyazev, Preconditioned eigensolvers, in: Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, H. van der Vorst (Eds.), Templates for the Solution of Algebraic Eigenvalue Problems: a Practical Guide, SIAM, Philadelphia, PA, 2000, in press.
- [73] A.N. Krylov, O cislennom resenii uravnenija, kotorym v techniceskih voprasah opredeljajutsja castoty malyh kolebanii material'nyh sistem, Izv. Akad. Nauk SSSR. Ser. Fiz.-Mat. 4 (1931) 491–539.
- [74] V.N. Kublanovskaja, On some algorithms for the solution of the complete eigenvalue problem, Zh. Vych. Mat. 1 (1961) 555–570.
- [75] V.N. Kublanovskaja, On an application of Newton's method to the determination of eigenvalues of  $\lambda$ -matrices, Soviet Math. Dokl. 10 (1969) 1240–1241.
- [76] V.N. Kublanovskaja, On an approach to the solution of the generalized latent value problem for λ-matrices, SIAM J. Numer. Anal. 7 (1970) 532–537.
- [77] P. Lancaster, Lambda-Matrices and Vibrating Systems, Pergamon Press, Oxford, UK, 1969.
- [78] C. Lanczos, An iteration method for the solution of the eigenvalue problem of linear differential and integral operators, J. Res. Natl. Bur. Stand 45 (1950) 225–280.

- [79] H.J. Landau, On Szegö's eigenvalue distribution theory and non-Hermitian kernels, J. Anal. Math. 28 (1975) 335–357.
- [80] S.L. Lee, A practical upper bound for departure from normality, SIAM J. Matrix Anal. Appl. 16 (1995) 462-468.
- [81] R.B. Lehoucq, D.C. Sorensen, C. Yang, ARPACK User's Guide Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods, SIAM, Philadelphia, PA, 1998.
- [82] T.-Y. Li, Z. Zeng, Homotopy-determinant algorithm for solving nonsymmetric eigenvalue problems, Math. Comput. 59 (1992) 483–502.
- [83] R. Lippert, A. Edelman, Nonlinear eigenvalue problems with orthogonality constraints, in: Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, H. van der Vorst (Eds.), Templates for the Solution of Algebraic Eigenvalue Problems: a Practical Guide, SIAM, Philadelphia, PA, 2000, in press.
- [84] S.H. Lui, Computation of pseudospectra by continuation, SIAM J. Sci. Comput. 18 (1997) 565-573.
- [85] S.H. Lui, G.H. Golub, Homotopy method for the numerical solution of the eigenvalue problem of self-adjoint partial differential operators, Numer. Algorithms 10 (1995) 363–378.
- [86] S.H. Lui, H.B. Keller, W.C. Kwok, Homotopy method for the large sparse real nonsymmetric eigenvalue problem, SIAM J. Matrix Anal. Appl. 18 (1997) 312–333.
- [87] K. Meerbergen, Robust methods for the calculation of rightmost eigenvalues of nonsymmetric eigenvalue problems, Ph.D. Thesis, Katholieke Universiteit Leuven, Leuven, Belgium, 1996.
- [88] K. Meerbergen, R. Morgan, Inexact methods, in: Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, H. van der Vorst (Eds.), Templates for the Solution of Algebraic Eigenvalue Problems: a Practical Guide, SIAM, Philadelphia, PA, 1999.
- [89] K. Meerbergen, D. Roose, The restarted Arnoldi method applied to iterative linear system solvers for the computation of rightmost eigenvalues, SIAM J. Matrix Anal. Appl. 18 (1997) 1–20.
- [90] C.B. Moler, D. Morrison, Singular value analysis of cryptograms, Amer. Math. Monthly 90 (1983) 78-87.
- [91] C.B. Moler, G.W. Stewart, An algorithm for generalized matrix eigenvalue problems, SIAM J. Numer. Anal. 10 (1973) 241–256.
- [92] R.B. Morgan, Computing interior eigenvalues of large matrices, Linear Algebra Appl. 154–156 (1991) 289–309.
- [93] R.B. Morgan, D.S. Scott, Generalizations of Davidson's method for computing eigenvalues of sparse symmetric matrices, SIAM J. Sci. Statist. Comput. 7 (3) (1986) 817–825.
- [94] J. Olsen, P. Jørgensen, J. Simons, Passing the one-billion limit in full configuration-interaction (FCI) calculations, Chem. Phys. Lett. 169 (1990) 463–472.
- [95] A.M. Ostrowski, On the convergence of the Rayleigh quotient iteration for the computation of characteristic roots and vectors I–VI, Arch. Rational. Mesh. Anal. I 1:233–241, II 2:423–428, III 3:325–340, IV 3:341–347, V 3:472– 481, VI 4:153–165, 1958–1959.
- [96] M.H.C. Paardekooper, An eigenvalue algorithm for skew symmetric matrices, Numer. Math. 17 (1971) 189-202.
- [97] C.C. Paige, The computation of eigenvalues and eigenvectors of very large matrices, Ph.D. Thesis, University of London, 1971.
- [98] C.C. Paige, B.N. Parlett, H.A. van der Vorst, Approximate solutions and eigenvalue bounds from Krylov subspaces, Numer. Linear Algebra Appl. 2 (2) (1995) 115–134.
- [99] C.C. Paige, M.A. Saunders, Solution of sparse indefinite systems of linear equations, SIAM J. Numer. Anal. 12 (1975) 617–629.
- [100] B.N. Parlett, The Symmetric Eigenvalue Problem, Prentice-Hall, Englewood Cliffs, NJ, 1980.
- [101] B.N. Parlett, Misconvergence in the Lanczos algorithm, in: M.G. Cox, S. Hammarling (Eds.), Reliable Numerical Computation, Clarendon Press, Oxford, UK, 1990, pp. 7–24 (Chapter 1).
- [102] B.N. Parlett, J.K. Reid, Tracking the progress of the Lanczos algorithm for large symmetric eigenproblems, IMA J. Numer. Anal. 1 (1981) 135–155.
- [103] B.N. Parlett, D.S. Scott, The Lanczos algorithm with selective orthogonalization, Math. Comput. 33 (1979) 217–238.
- [104] B.N. Parlett, D.R. Taylor, Z.A. Liu, A look-ahead Lanczos algorithm for unsymmetric matrices, Math. Comput. 44 (1985) 105–124.
- [105] D.A. Pope, C. Tompkins, Maximizing functions of rotations: experiments concerning speed of diagonalization of symmetric matrices using Jacobi's method, J. ACM 4 (1957) 459–466.
- [106] E.H. Rogers, A minimax theory for overdamped systems, Arch. Rational Mech. Anal. 19 (1964) 89-96.
- [107] A. Ruhe, Algorithms for the nonlinear eigenvalue problem, SIAM J. Numer. Anal. 10 (1973) 674-689.

- [108] A. Ruhe, Implementation aspects of band Lanczos algorithms for computation of eigenvalues of large sparse symmetric matrices, Math. Comput. 33 (1979) 680–687.
- [109] A. Ruhe, The rational Krylov algorithm for nonsymmetric eigenvalue problems. iii. complex shifts for real matrices, BIT 34 (1994) 165–176.
- [110] A. Ruhe, Rational Krylov algorithms for nonsymmetric eigenvalue problems. ii. Matrix pairs, Linear Algebra Appl. 197/198 (1994) 283–295.
- [111] H. Rutishauser, Solution of eigenvalue problems with the LR-transformation, Nat. Bur. Standards Appl. Math. Ser. 49 (1958) 47–81.
- [112] H. Rutishauser, Computational aspects of F.L. Bauer's simultaneous iteration method, Numer. Math. 13 (1969) 4–13.
- [113] H. Rutishauser, Simultaneous iteration method for symmetric matrices, in: J.H. Wilkinson, C. Reinsch (Eds.), Handbook for Automatic Computation, Vol. 2, Linear Algebra, Springer Verlag, Heidelberg, 1971, pp. 284–302.
- [114] Y. Saad, On the rates of convergence of the Lanczos and the block Lanczos methods, SIAM J. Numer. Anal. 17 (1980) 687–706.
- [115] Y. Saad, Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices, Linear Algebra Appl. 34 (1980) 269–295.
- [116] Y. Saad, Numerical Methods for Large Eigenvalue Problems, Manchester University Press, Manchester, UK, 1992.
- [117] Y. Saad, M.H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Statist. Comput. 7 (1986) 856–869.
- [118] A. Sameh, On Jacobi and Jacobi-like algorithms for a parallel computer, Math. Comput. 25 (1971) 579–590.
- [119] A. Schönhage, Zur Konvergenz des Jacobi-Verfahrens, Numer. Math. 3 (1961) 374-380.
- [120] G.L.G. Sleijpen, J.G.L. Booten, D.R. Fokkema, H.A. van der Vorst, Jacobi–Davidson type methods for generalized eigenproblems and polynomial eigenproblems, BIT 36 (1996) 595–633.
- [121] G.L.G. Sleijpen, H.A. van der Vorst, A Jacobi–Davidson iteration method for linear eigenvalue problems, SIAM J. Matrix Anal. Appl. 17 (1996) 401–425.
- [122] G.L.G. Sleijpen, H.A. van der Vorst, E. Meijerink, Efficient expansion of subspaces in the Jacobi–Davidson method for standard and generalized eigenproblems, Electron. Trans. Numer. Anal. 7 (1998) 75–89.
- [123] G.L.G. Sleijpen, H.A. van der Vorst, M.B. van Gijzen, Quadratic eigenproblems are no problem, SIAM News 29 (1996) 8–9.
- [124] P. Smit, Numerical analysis of eigenvalue algorithms based on subspace iterations, Ph.D. Thesis, Tilburg University, The Netherlands, 1997.
- [125] D.C. Sorensen, Implicit application of polynomial filters in a k-step Arnoldi method, SIAM J. Matrix Anal. Appl. 13 (1) (1992) 357–385.
- [126] G.W. Stewart, Simultaneous iteration for computing invariant subspaces of non-Hermitian matrices, Numer. Math. 25 (1976) 123–136.
- [127] G.W. Stewart, SRRIT: a Fortran subroutine to calculate the dominant invariant subspace of a nonsymmetric matrix, Technical Report TR 154, Department of Computer Science, University of Maryland, 1978.
- [128] G.W. Stewart, On the early history of the singular value decomposition, SIAM Rev. 35 (1993) 551-566.
- [129] G.W. Stewart, Ji-Guang Sun, Matrix Perturbation Theory, Academic Press, San Diego, CA, 1990.
- [130] J.G. Sun, Perturbation analysis of quadratic eigenvalue problems, Report UNINF 99.01, Dept. Comp. Science, Umeå University, Sweden, 1999.
- [131] O. Taussky, A recurring theorem on determinants, Amer. Math. Monthly 56 (1949) 672-676.
- [132] G. Temple, The accuracy of Rayleigh's method of calculating the natural frequencies of vibrating systems, Proc. Roy. Soc. A 211 (1952) 204–224.
- [133] F. Tisseur, Backward error and condition of polynomial eigenvalue problems, Technical Report NA-332, University of Manchester, Manchester, UK, 1998.
- [134] F. Tisseur, J. Dongarra, A parallel divide and conquer algorithm for the symmetric eigenvalue problem on distributed memory architectures, SIAM J. Sci. Comput. 20 (1999) 2223–2236.
- [135] L.N. Trefethen, Pseudospectra of matrices, in: D.F. Griffiths, G.A. Watson (Eds.), Numerical Analysis 1991, Longman, 1992, pp. 234–266.
- [136] L.N. Trefethen, Computation of pseudospectra, Acta Numer. 8 (1999) 247-295.
- [137] L.N. Trefethen, D. Bau, III, Numerical Linear Algebra, SIAM, Philadelphia, PA, 1997.

- [138] L.N. Trefethen, A.E. Trefethen, S.C. Reddy, T.A. Driscoll, Hydrodynamic stability without eigenvalues, Science 261 (1993) 578–584.
- [139] L.N. Trefethen, M.R. Trummer, An instability phenomenon in spectral methods, SIAM J. Numer. Anal. 24 (1987) 1008–1023.
- [140] H.W. Turnbull, A.C. Aitken, Introduction to the Theory of Canonical Matrices, Dover Publications, New York, 1961.
- [141] R.R. Underwood, An iterative block Lanczos method for the solution of large sparse symmetric eigenproblems, Technical Report STAN-CS-75-495, Department of Computer Science, Stanford University, Stanford, CA, 1975.
- [142] A. van der Sluis, H.A. van der Vorst, The convergence behavior of Ritz values in the presence of close eigenvalues, Linear Algebra Appl. 88/89 (1987) 651–694.
- [143] P. Van Dooren, The computation of Kronecker's canonical form of a singular pencil, Linear Algebra Appl. 27 (1979) 103–140.
- [144] P. Van Dooren, The generalized eigenstructure problem in linear system theory, IEEE Trans. Automat. Control AC 26 (1981) 111–128.
- [145] P. Van Dooren, A generalized eigenvalue approach of solving Ricatti equations, SIAM J. Sci. Statist. Comput. 2 (1981) 121–135.
- [146] P. Van Dooren, Algorithm 590, DUSBSP AND EXCHQZZ: FORTRAN subroutines for computing deflating subspaces with specified spectrum, ACM Trans. Math. Software 8 (1982) 376–382.
- [147] H.P.M. van Kempen, On quadratic convergence of the special cyclic Jacobi method, Numer. Math. 9 (1966) 19-22.
- [148] R.S. Varga, Matrix Iterative Analysis, Prentice-Hall, Englewood Cliffs, NJ, 1962.
- [149] R. von Mises, H. Pollaczek-Geiringer, Praktische Verfahren der Gleichungsauflösung, Z. Angew. Math. Mech. 9 (1929) 58–77, 152–164.
- [150] D.S. Watkins, Fundamentals of Matrix Computations, Wiley, New York, 1991.
- [151] D.S. Watkins, Some perspectives on the eigenvalue problem, SIAM Rev. 35 (1993) 430-471.
- [152] D.S. Watkins, L. Elsner, Chasing algorithms for the eigenvalue problem, SIAM J. Matrix Anal. Appl. 12 (1991) 374–384.
- [153] H.F. Weinberger, Variational Methods for Eigenvalue Approximation, SIAM, Philadelphia, 1974.
- [154] J.H. Wilkinson, The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, 1965.
- [155] J.H. Wilkinson, Linear differential equations and Kronecker's canonical form, in: C. de Boor, G.H. Golub (Eds.), Recent advances in Numerical Analysis, Academic Press, New York, 1978, pp. 231–265.
- [156] J.H. Wilkinson, Kronecker's canonical form and the QZ algorithm, Linear Algebra Appl. 28 (1979) 285-303.
- [157] J.H. Wilkinson, Eigenvalue problems, in: A. Iserles, M.J.D. Powell (Eds.), The State of the Art in Numerical Analysis, Clarendon Press, Oxford, 1987, pp. 1–39.
- [158] J.H. Wilkinson, C. Reinsch (Eds.), Handbook for Automatic Computation, Vol. 2, Linear Algebra, Springer, Heidelberg, 1971.
- [159] T. Zhang, K.H. Law, G.H. Golub, On the homotopy method for perturbed symmetric generalized eigenvalue problems, SIAM J. Numer. Anal. 35 (1) (1998) 300–319.



JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

Journal of Computational and Applied Mathematics 123 (2000) 67-83

www.elsevier.nl/locate/cam

# QR-like algorithms for eigenvalue problems

David S. Watkins

Department of Mathematics, Washington State University, Pullman, WA 99164-3113, USA

Received 18 May 1999; received in revised form 12 November 1999

#### Abstract

In the year 2000 the dominant method for solving matrix eigenvalue problems is still the QR algorithm. This paper discusses the family of GR algorithms, with emphasis on the QR algorithm. Included are historical remarks, an outline of what GR algorithms are and why they work, and descriptions of the latest, highly parallelizable, versions of the QR algorithm. Now that we know how to parallelize it, the QR algorithm seems likely to retain its dominance for many years to come. © 2000 Elsevier Science B.V. All rights reserved.

MSC: 65F15

Keywords: Eigenvalue; QR algorithm; GR algorithm

### 1. Introduction

 $\tilde{G}$  Since the early 1960s the standard algorithms for calculating the eigenvalues and (optionally) eigenvectors of "small" matrices have been the *QR* algorithm [28] and its variants. This is still the case in the year 2000 and is likely to remain so for many years to come. For us a *small* matrix is one that can be stored in the conventional way in a computer's main memory and whose complete eigenstructure can be calculated in a matter of minutes without exploiting whatever sparsity the matrix may have had. If a matrix is small, we may operate on its entries. In particular, we are willing to perform similarity transformations, which will normally obliterate any sparseness the matrix had to begin with.<sup>1</sup>

If a matrix is not small, we call it *large*. The boundary between small and large matrices is admittedly vague, but there is no question that it has been moving steadily upward since the dawn of the computer era. In the year 2000 the boundary is around n = 1000, or perhaps a bit higher.

E-mail address: watkins@wsu.edu (D.S. Watkins).

<sup>&</sup>lt;sup>1</sup>However, we are not averse to seeking to preserve and exploit certain other structures (e.g. symmetry) by choosing our transforming matrices appropriately.
Eigenvalue problems come in numerous guises. Whatever the form of the problem, the QR algorithm is likely to be useful. For example, for generalized eigenvalue problems  $Ax = \lambda Bx$ , the method of choice is a variant of the QR algorithm called QZ. Another variant of QR is used to calculate singular value decompositions (SVD) of matrices. The QR algorithm is also important for solving large eigenvalue problems. Most algorithms for computing eigenvalues of large matrices repeatedly generate small auxiliary matrices whose eigensystems need to be computed as a subtask. The most popular algorithms for this subtask are the QR algorithm and its variants.

# 1.1. QR past and present

In this paper we discuss the family of *GR* algorithms, which includes the *QR* algorithm. The subject was born in the early 1950s with Rutishauser's quotient-difference algorithm [43,44] which he formulated as a method for calculating the poles of a meromorphic function. He then reformulated it in terms of matrix operations and generalized it to the *LR* algorithm [45].<sup>2</sup> The *QR* algorithm was published by Kublanovskaya [37] and Francis [28] in 1961. The Francis paper is particularly noteworthy for the refinements it includes. The double-shift implicit *QR* algorithm laid out there is only a few details removed from codes that are in widespread use today.

And what codes are in use today? By far the most popular tool for matrix computations is Matlab. If you use Matlab to compute your eigenvalues, you will use one of its four QR-based computational kernels. Each of these is just a few refinements removed from codes in the public-domain software packages EISPACK [46] and LINPACK [20]. In particular, the algorithm for computing eigenvalues of real, nonsymmetric matrices is just the Francis double-shift QR algorithm with some modifications in the shift strategy.

A newer public-domain collection is LAPACK [25], which was designed to perform well on vector computers, high-performance work stations, and shared-memory parallel computers. It also has a double-shift implicit QR code, which is used on matrices (or portions of matrices) under  $50 \times 50$ . For larger matrices a multishift QR code is used.

For many years the QR algorithm resisted efforts to parallelize it. The prospects for a massively parallel QR algorithm for distributed memory parallel computers were considered dim. The pessimism was partly dispelled by van de Geijn and Hudson [48], who demonstrated the first successful highly parallel QR code. However, their code relies on an unorthodox distribution of the matrix over the processors, which makes it hard to use in conjunction with other codes. Subsequently, Henry [33] wrote a successful parallel QR code that uses a standard data distribution. This is an implicit double-shift code that performs the iterations in pipeline fashion. This code is available in ScaLAPACK [26], a collection of matrix computation programs for distributed-memory parallel computers.

On the theoretical side, the first proof of convergence of the LR algorithm (without pivoting or shifts of origin) was provided by Rutishauser [45]. His proof was heavily laden with determinants, in the style of the time. Wilkinson [61] proved convergence of the unshifted QR algorithm using matrices, not determinants. Wilkinson [62,40] also proved global convergence of a shifted QR algorithm on symmetric, tridiagonal matrices. Della Dora [18] introduced a family of GR algorithms

<sup>&</sup>lt;sup>2</sup> Amazingly the quotient-difference algorithm has had a recent revival. Fernando and Parlett [27,41] introduced new versions for finding singular values of bidiagonal matrices and eigenvalues of symmetric, tridiagonal matrices.

and proved a general convergence theorem (unshifted case). In [59] a more general family of GR algorithms was introduced, and general convergence theorems for shifted GR algorithms were proved.

#### 1.2. Contents

This paper provides an overview of the family of GR algorithms, with emphasis on the QR case. The properties of the various QR implementations are discussed. We begin by introducing the family of GR algorithms in Section 2. These are iterative methods that move a matrix toward upper-triangular form via similarity transformations. We discuss the convergence of GR algorithms briefly. In Section 3 we show how to implement GR algorithms economically as bulge-chasing procedures on Hessenberg matrices. In Sections 4 and 5 we discuss multishift and pipelined QR algorithms, respectively.

Section 6 discusses the generalized eigenvalue problem  $Av = \lambda Bv$  and GZ algorithms, which are generalizations of GR algorithms. Particularly important among the GZ algorithms are the QZalgorithms. These are normally implemented implicitly, as bulge chasing algorithms. However, in Section 7, we discuss a completely different class of explicit QZ algorithms. These attempt to divide and conquer the problem by splitting it apart on each iteration. They are highly parallelizable and may play a significant role in parallel eigensystem computations in the future.

# 2. GR algorithms

Let A be an  $n \times n$  real or complex matrix whose eigenvalues we seek. GR algorithms [59] are iterative methods that begin with a matrix  $A_0$  similar to A (e.g.  $A_0 = A$ ) and produce a sequence  $(A_m)$  of similar matrices. All GR algorithms have the following form. Given the iterate  $A_m$ , the next iterate  $A_{m+1}$  is produced as follows. First a spectral transformation function  $f_m$  is somehow chosen. At this point the only requirement on  $f_m$  is that the matrix  $f_m(A_m)$  be well defined. Thus  $f_m$  could be a polynomial, rational function, exponential function, or whatever. The next step is to decompose  $f_m(A_m)$  into a product

$$f_m(A_m) = G_{m+1}R_{m+1},$$
(2.1)

where  $G_{m+1}$  is nonsingular and  $R_{m+1}$  is upper triangular. There are number of ways to do this; the symbol G stands for *general* or *generic*. The final step of the iteration is to use  $G_{m+1}$  in a similarity transformation to produce the next iterate:

$$A_{m+1} = G_{m+1}^{-1} A_m G_{m+1}. ag{2.2}$$

If the f's and G's are chosen well (and perhaps even if they are not), the sequence of similar matrices, all of which have the same eigenvalues, will converge rapidly to a block upper triangular form

 $\begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix},$ 

thereby splitting the problem into two smaller eigenvalue problems with matrices  $A_{11}$  and  $A_{22}$ . After O(n) such splittings, the eigenvalue problem has been solved.

Some variants are the RG algorithms, in which the order of factors in (2.1) is reversed, and the GL and LG algorithms, in which lower triangular matrices are used. All of these families have isomorphic convergence theories. In practice, some of these variants do come in handy here and there, but we will focus for the most part on the GR case.

A particular *GR* algorithm is determined by how the spectral transformation functions  $f_m$  are chosen and how the transforming matrices  $G_{m+1}$  are specified. Let us first discuss choices of *G*.

If each  $G_{m+1}$  is required to be unitary, then the symbol Q is used instead of G, the decomposition becomes  $f_m(A_m) = Q_{m+1}R_{m+1}$ , and the algorithm is called a QR algorithm. The requirement that  $Q_{m+1}$  be unitary implies that the factors in the decomposition are nearly uniquely determined. This is the most popular choice of G. Expositions on the QR algorithm can be found in numerous books [30,52,61].

If each  $G_{m+1}$  is required to be unit lower triangular, that is, lower triangular with ones on the main diagonal, then the symbol *L* is used, the decomposition becomes  $f_m(A_m) = L_{m+1}R_{m+1}$ , and the algorithm is called an *LR* algorithm. The *LR* decomposition is unique if it exists, but not every matrix has an *LR* decomposition. This means that the choice of  $f_m$  must be restricted in such a way that  $f_m(A_m)$  has an *LR* decomposition. The algorithm is unstable; difficulties arise when  $f_m$  are chosen so that  $f_m(A_m)$  is close to a matrix that has no *LR* decomposition. Stability can be improved markedly by the introduction of pivoting (row and column interchanges). Wilkinson's book [61] discusses *LR* algorithms in detail.

Other examples are the HR [9,10] SR [12,13], and BR [29] algorithms. The H stands for hyperbolic, the S for symplectic, and the B for balancing, band-reducing, bulge-chasing algorithm.

Now let us consider some ways of choosing the functions  $f_m$ . We call them spectral transformation functions because it is their job to transform the spectrum of the matrix in order to accelerate convergence. We also refer to  $f_m$  as the function that *drives* the *m*th iteration. The simplest spectral transformation functions are polynomials, and the simplest useful polynomials have degree one. If we take  $f(z) = z - \mu$ , then we have  $f(A) = A - \mu I$ . Such a choice gives us a *simple* or *single GR* step with *shift*  $\mu$ . The quadratic choice  $f(z) = (z - \mu)(z - v)$  gives a *double GR* step with *shifts*  $\mu$  and v. A double step is worth two single steps. The standard *QR* codes for real matrices (dating back to Francis [28]) take double steps with either real  $\mu$  and v or complex  $v = \bar{\mu}$ . This keeps the computations real. The multishift *QR* algorithm [2] takes  $f(z) = (z - \mu_1)(z - \mu_2) \cdots (z - \mu_p)$ , where p can be as big as one pleases in principle. In practice, roundoff errors cause problems if p is taken much bigger than six.

A more exotic choice would be a rational function such as

$$f(z) = \frac{(z - \mu)(z - \bar{\mu})}{(z + \mu)(z + \bar{\mu})}$$

This is the sort of f that is used to drive the Hamiltonian QR algorithm of Byers [15,16]. The more general use of rational spectral transformation functions is discussed in [57].

An even more exotic choice would be a characteristic function for the unit disk:

$$f(z) = \begin{cases} 1 & \text{if } |z| < 1, \\ 0 & \text{if } |z| > 1. \end{cases}$$
(2.3)

This is a simple function to describe, but how does one calculate f(A)? For now we just remark that there are good rational approximations. For example, if k is a large integer, the rational function

$$f(z) = \frac{1}{z^k + 1}$$

approximates the characteristic function quite well away from the circle |z| = 1.

#### 2.1. Factors that affect the convergence rate

The convergence theory of GR algorithms was discussed by Watkins and Elsner [59] and summarized in [55]. There are two factors affecting the convergence of the algorithm: the choice of  $f_m$  and the choice of  $G_m$ . Let  $\hat{G}_m = G_1 G_2 \cdots G_m$ , the product of the transforming matrices for the first m steps. If the condition numbers  $\kappa(\hat{G}_m)$  grow with m, convergence can be degraded or prevented. On the other hand, it is the role of the  $f_m$  to promote or accelerate convergence. For starters let us suppose that the same f is used on every iteration. If  $\lambda_1, \lambda_2, \ldots, \lambda_n$  are the eigenvalues of A, then  $f(\lambda_1), f(\lambda_2), \dots, f(\lambda_n)$  are the eigenvalues of f(A). Suppose they are numbered so that  $|f(\lambda_1)| \ge |f(\lambda_2)| \ge \cdots \ge |f(\lambda_n)|$ . Then the ratios

$$\rho_k = |f(\lambda_{k+1})/f(\lambda_k)|, \quad k = 1, \dots, n-1$$

are what determine the asymptotic convergence rate. These ratios all satisfy  $0 \le \rho_k \le 1$ . The closer to zero they are, the better. The underlying mechanism is subspace iteration [14,34,42,51,59].

Let us consider the effect of the kth ratio  $\rho_k$ . Suppose  $\rho_k < 1$ , and let  $\hat{\rho}_k$  be any number satisfying  $\rho_k < \hat{\rho}_k < 1$ . Partition the iterates  $A_m$  into blocks

$$A_m = \begin{bmatrix} A_{11}^{(m)} & A_{12}^{(m)} \\ A_{21}^{(m)} & A_{22}^{(m)} \end{bmatrix},$$

where  $A_{11}^{(m)}$  is  $k \times k$ . Then, under mild assumptions, there exists a constant C such that

 $||A_{21}^{(m)}|| \leq C\kappa(\hat{G}_m)\hat{\rho}_{\perp}^m$  for all m.

Thus  $A_m$  approaches block upper triangular form if  $\kappa(\hat{G}_m)\hat{\rho}_k^m \to 0$ . If there is a bound K such that  $\kappa(\hat{G}_m) \leq K$  for all m, then convergence is linear with ratio  $\rho_k = |f(\lambda_{k+1})/f(\lambda_k)|$ . Even if  $\kappa(\hat{G}_m)$  is unbounded, there still can be convergence if the growth is not too fast.

So far we have assumed that f is held fixed. Varying f makes the convergence analysis harder, but (with rare exceptions) it pays off in accelerated convergence. Successful shift strategies are (with rare exceptions) able to choose  $f_m$  so that  $f_m(A) \to f(A)$ , where f is a function such that  $\rho_k = 0$  for some k. This yields superlinear convergence. A simple shift strategy that normally yields quadratic convergence is discussed below.

Let us reconsider choices of G in light of the convergence theory. Clearly, the objective is to make the transforming matrices as well conditioned as possible. This is true also from the point of view of stability, since the condition numbers  $\kappa(\hat{G}_m)$  govern the stability of the algorithm as well. From this viewpoint the QR algorithms are obviously best, as they guarantee  $\kappa_2(\hat{Q}_m) = 1$  for all m. No such guarantees exist for any of the other GR algorithms, which explains why the QR algorithms are by far the most popular. In certain special circumstances (e.g. Hamiltonian problems) there exist (non-QR) GR algorithms that are very fast (O(n) work per iteration instead of O(n<sup>2</sup>)) because they are able to exploit the structure. In those circumstances one may be willing to trade the stability guarantee for speed. But then one must always be alert to the danger of instability. In this paper we will focus mainly on *QR* algorithms.

We now reconsider choices of f in light of the convergence theory. The simplest and most common choice is the polynomial

$$f(z) = (z - \mu_1)(z - \mu_2) \cdots (z - \mu_p).$$

The best we can do is to take the shifts  $\mu_1, \ldots, \mu_p$  to be eigenvalues of A. Then f(A) has p zero eigenvalues, so

$$\frac{f(\lambda_{n-p+1})}{f(\lambda_{n-p})} = 0.$$
(2.4)

Such a good ratio implies very rapid convergence. Indeed, after just one iteration we get

$$A_{1} = \begin{bmatrix} A_{11}^{(1)} & A_{12}^{(1)} \\ 0 & A_{22}^{(1)} \end{bmatrix},$$
(2.5)

where  $A_{22}^{(1)}$  is  $p \times p$  and has  $\mu_1, \ldots, \mu_p$  as its eigenvalues.<sup>3</sup> The catch is that we do not normally have the eigenvalues available to use as shifts. However, after a few iterations we might well have some good approximations, and we can use these as shifts. If all p shifts are excellent approximations to eigenvalues, then the ratio in (2.4) will be close to zero, and convergence to a form like (2.5) will be achieved in a few iterations. Subsequent iterations can be applied to the submatrix  $A_{11}$  with a new set of shifts.

Normally new shifts are chosen on each iteration. The most common strategy is to take the shifts (on the *m*th iteration) to be the eigenvalues of the lower right-hand  $p \times p$  submatrix  $A_{22}^{(m)}$ . In other words,  $f_m$  is taken to be the characteristic polynomial of  $A_{22}^{(m)}$ . Global convergence is not guaranteed, but the local convergence rate is normally quadratic and can even be cubic if the matrices satisfy certain symmetry properties [59].

A few words about global convergence are in order. The unitary circulant shift matrix  $C_n$  exemplified by the  $4 \times 4$  case

$$C_4 = \begin{bmatrix} 1 & & 1 \\ 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}$$

is invariant under QR iterations with zero shifts, as is any unitary matrix. The shift strategy described in the previous paragraph gives zero shifts, as long as p < n. Thus the algorithm fails to converge when applied to  $C_n$ . Even worse things can happen; in some cases the shifts can wander chaotically [5]. The standard cure for these problems is to use *exceptional shifts* (for example, random shifts)

<sup>&</sup>lt;sup>3</sup> This result ignores the effect of roundoff errors. In practice, the (2,1) block of (2.5) will not be exactly zero, and usually it will not be small enough to allow a safe deflation of the problem.

if many iterations have passed with no progress. The point of this strategy is to knock the matrix away from any dangerous areas. It is not foolproof [17], but it has worked well over the years. Nevertheless, a shift strategy that is provably globally convergent (and converges quadratically on almost all matrices) would be welcome.

The only class of matrices for which global convergence has been proved is that of Hermitian tridiagonal matrices, provided that the Wilkinson shift strategy is used [40]. The Wilkinson strategy takes p = 1; the lone shift is the eigenvalue of the  $2 \times 2$  lower right-hand submatrix that is closer to  $a_{nn}$ .

### 3. Implicit implementations of GR algorithms

For most of the choices of f that we have considered, the cost of calculating f(A) is high. For this and other reasons, most implementations of GR algorithms find a way to perform the iterations without calculating f(A) explicitly. Usually, the first column of f(A) is all that is needed. This section shows how to do it when f is a polynomial.

If we wish to use an implicit GR algorithm, we must first transform the matrix to a condensed form. The best known such form is upper Hessenberg, but there are others. For example, any Hermitian matrix can be put into tridiagonal form, and so can almost any other square matrix [61], although the stability of the transformation comes into question for non-Hermitian matrices. For unitary matrices there are several condensed forms, including the Schur parameter pencil [1,11,53] and the double staircase form [7,53]. For Hamiltonian matrices there are both Hessenberg-like and tridiagonal-like forms [12,39]. Implicit GR algorithms can be built on all of these forms, but for simplicity we will restrict our attention to upper Hessenberg form.



A matrix *A* is in *upper Hessenberg* form if  $a_{ij}=0$  whenever i > j+1. Every matrix can be transformed stably to upper Hessenberg form by a unitary similarity transformation [30,52,61]. There are also various useful nonunitary reductions to Hessenberg form, and these will play a role in what follows. The general plan of all of these reduction algorithms is that they first introduce zeros in the first column, then the second column, then the third column, and so on.

An upper Hessenberg matrix A is in *proper* upper Hessenberg form if  $a_{j+1,j} \neq 0$  for  $j=1,\ldots,n-1$ . If a matrix is not in proper upper Hessenberg form, we can divide its eigenvalue problem into independent subproblems for which the matrices *are* proper upper Hessenberg.

Suppose A is a proper upper Hessenberg matrix, and we wish to perform an iteration of a multishift GR algorithm:

$$f(A) = GR,\tag{3.1}$$

$$\hat{A} = G^{-1}AG,\tag{3.2}$$

where f is a polynomial of degree p:  $f(A) = (A - \mu_1 I) \cdots (A - \mu_p I)$ . Since we are considering only a single iteration, we have dropped the subscripts to simplify the notation. There is no need to calculate f(A); it suffices to compute the first column, which is

$$x = (A - \mu_1 I) \cdots (A - \mu_p I) e_1.$$

Since A is upper Hessenberg, only the first p + 1 entries of x are nonzero, and x can be computed in  $O(p^3)$  flops. This is negligible if  $p \ll n$ .

The implicit *GR* iteration is set in motion by building a nonsingular matrix  $\tilde{G}$  that has its first column proportional to x and looks like an identity matrix except for the  $(p+1) \times (p+1)$  submatrix in the upper left-hand corner. There are many ways to do this; for example,  $\tilde{G}$  can be a Householder reflector.  $\tilde{G}$  is then used to perform a similarity transformation  $A \to \tilde{G}^{-1}A\tilde{G}$ , which disturbs the upper Hessenberg form; the transformed matrix has a bulge, the size of which is proportional to p, the degree of the iteration.



The rest of the iteration consists of returning the matrix to upper Hessenberg form by any one of the standard reduction algorithms. As the columns are cleared out one by one, new nonzero entries are added to the bottom of the bulge, so the bulge is effectively chased from one end of the matrix to the other.



Hence, these algorithms are called *bulge-chasing* algorithms. Once the bulge has been chased off of the bottom of the matrix, the iteration is complete.

Let G denote the product of all of the transforming matrices applied during the iteration, so that the entire similarity transformation is  $\hat{A} = G^{-1}AG$ . Watkins and Elsner [58] showed that no matter what kind of transforming matrices are used, G satisfies p(A) = GR for some upper-triangular R. Thus the procedure just outlined effects a GR iteration (3.1), (3.2) implicitly. It follows that the GR convergence theory [59] is applicable to all algorithms of this type.

Let us consider some of the possibilities. If  $\tilde{G}$  and all of the bulge-chasing transformations are unitary, then G is unitary, so a QR iteration is performed. This is by far the most popular choice. If, on the other hand, all of the transformations are elementary lower triangular (Gaussian elimination) transformations (without pivoting), then G is unit lower triangular, and an LR iteration is performed. For stability one can perform a row interchange to maximize the pivot before each elimination. This is how one implements the LR algorithm with pivoting. Unless the matrix has some special structure that one wishes to preserve (e.g. symmetric, Hamiltonian), there is no reason to insist that all of the transforming matrices be of the same type. Haag and Watkins [31] have developed bulge-chasing algorithms that mix unitary and Gaussian elimination transformations.

#### 4. Performance of multishift QR algorithms

We now confine our attention to the OR algorithm, although this restriction is by no means necessary. In principle we can perform multishift QR steps of any degree p. What is a good choice of p in practice? Historically, the first choice was p = 1, and this is still popular. The most widely used QR codes for real symmetric matrices and for complex non-Hermitian matrices make this choice. Another early choice that is still popular is p=2, which allows the use of complex shifts on real matrices without going outside the real number field. That was Francis's reason for inventing the double-shift algorithm. Descendents of Francis's code are still in widespread use in Matlab, EISPACK, LAPACK, and elsewhere, as we have already mentioned. For many years 1 and 2 were the only choices of p that were used. The structure of certain types of matrices [16] causes their eigenvalues to come in sets of four (e.g.  $\lambda$ ,  $\overline{\lambda}$ ,  $-\lambda$ ,  $-\overline{\lambda}$ ). For these matrices the choice p = 4 is obviously in order. The use of large values of p was first advocated by Bai and Demmel [2]. This seemed like an excellent idea. If one gets, say, thirty shifts from the lower right hand  $30 \times 30$ submatrix and uses them for a *QR* step of degree p = 30, then one has to chase a  $30 \times 30$  bulge. This is like doing 30 steps at a time, and it entails a lot of arithmetic. Since the computations are quite regular, they can be implemented in level-2 (or possibly level-3) BLAS [21,22] thereby enhancing performance on modern vector, cache-based, or parallel computers.

Unfortunately, the multishift QR algorithm does not perform well if the degree p is taken too large. This empirical fact is at odds with the convergence theory and came as a complete surprise. Some experiments of Dubrulle [24] showed that the problem lies with roundoff errors. If p shifts are chosen, they can be used to perform either one QR iteration of degree p (chasing one big bulge) or p/2 iterations of degree two (chasing p/2 small bulges). In principle, the two procedures should yield the same result. Dubrulle showed that in practice they do not: The code that chases many small bulges converges rapidly as expected, while the one that chases fewer large bulges goes nowhere. The difference is entirely due to roundoff errors.

We were able to shed some light on the problem by identifying the mechanism by which information about the shifts is transmitted through the matrix during a bulge chase [56]. The shifts are



Fig. 1. Pipelined QR steps.

used only at the very beginning of the iteration, in the computation of the vector x that is used to build the transforming matrix that creates the bulge. The rest of the algorithm consists of chasing the bulge; no further reference to the shifts is made. Yet good shifts are crucial to the rapid convergence of the algorithm. In the case of multishift QR, convergence consists of repeated deflation of (relatively) small blocks off of the bottom of the matrix. The good shifts are supposed to accelerate these deflations. Thus the information about the shifts must somehow be transmitted from the top to the bottom of the matrix during the bulge chase, but how? In [56] we demonstrated that the shifts are transmitted as eigenvalues of a certain matrix pencil associated with the bulge. When pis large, the eigenvalues of this *bulge pencil* tend to be ill conditioned, so the shift information is not represented accurately. The shifts are *blurred*, so to speak. The larger p is, the worse is the blurring. When p=30, it is so bad that the shifts are completely lost. The algorithm functions as if random shifts had been applied. From this perspective it is no longer a surprise that multshift QRperforms poorly when p=30.

The multishift idea has not been abandoned. The main workhorse in LAPACK [25] for solving nonsymmetric eigenvalue problems is a multishift QR code. In principle this code can be operated at any value of p, but p = 6 has been chosen for general use. At this value the shift blurring is slight enough that it does not seriously degrade convergence, and a net performance gain is realized through the use of Level 2 BLAS.

# 5. Pipelined QR algorithm

Through Dubrulle's experiments it became clear that one can perform a QR iteration of degree 30, say, by chasing 15 bulges of degree 2. This works well because the shifts are not blurred at all when p = 2. Once we have set one bulge in motion, we can start the next bulge as soon as we please; there is no need to wait for completion of the first bulge chase. Once we have set the second bulge in motion, we can start the third, and so on. In this way we can chase all 15 (or however many) bulges simultaneously in pipeline fashion.

Imagine a matrix that is really large and is divided up over many processors of a distributed memory parallel computer. If the bulges are spread evenly, as shown in Fig. 1, a good many processors can be kept busy simultaneously.

The idea of pipelining QR steps is not new. For example, it has been considered by Heller and Ipsen [32], Stewart [47], van de Geijn [49,50], and Kaufman [36], but the idea did not catch on right away because nobody thought of changing the shift strategy. For bulges of degree two, the standard strategy is to take as shifts the two eigenvalues of the lower right-hand  $2 \times 2$  submatrix. The entries of this submatrix are among the last to be computed in a QR step, for the bulge is chased from top to bottom. If one wishes to start a new step before the bulge for the current step has reached the bottom of the matrix, one is forced to use old shifts because the new ones are not available yet. If one wants to keep a steady stream of, say, 15 bulges running in the pipeline, one is obliged to use shifts that are 15 iterations out of date, so to speak. The use of such "stale" shifts degrades the convergence rate significantly.

But now we are advocating a different strategy [54]: Choose some even number p (e.g. 30) and get p shifts by computing the eigenvalues of the lower right-hand  $p \times p$  matrix. Now we have enough shifts to chase p/2 bulges in pipeline fashion without resorting to out-of-date shifts. This strategy works well. It is used in ScaLAPACK's parallel QR code [33] for nonsymmetric eigenvalue problems.

Numerous improvements are possible. For example, the arithmetic could be performed more efficiently if the bulges were chased in (slightly blurred) packets of six instead of two. Another possibility is to chase tight clusters of small bulges, as in recent work of Braman, Byers, and Mathias [8]. As a cluster of bulges is chased through a segment of the matrix, the many small transforming matrices generated from the bulge chases can be accumulated in a larger orthogonal matrix, which can then be applied using level 3 BLAS [21]. A price is paid for this: the total number of flops per iteration is roughly doubled. The payoffs are that operations implemented in level 3 BLAS are easily parallelized and allow modern cache-based processors to operate at near top speed. Another innovation of [8] is the introduction of a more aggressive deflation strategy (and accompanying shift strategy) that allows the algorithm to terminate in fewer iterations. These innovations appear to have a good chance for widespread acceptance in time.

## 6. Generalized eigenvalue problem

Matrix eigenvalue problems frequently present themselves as generalized eigenvalue problems involving a matrix pair (A,B), which is also commonly presented as a matrix pencil  $A - \lambda B$ . A nonzero vector v is an eigenvector of the matrix pencil with associated eigenvalue  $\lambda$  if

 $Av = \lambda Bv.$ 

v is an eigenvector with eigenvalue  $\infty$  if Bv = 0. The generalized eigenvalue problem reduces to the standard eigenvalue problem in the case B = I. In analogy with the standard eigenvalue problem we easily see that  $\lambda$  is a finite eigenvalue of the pencil if and only if det $(A - \lambda B) = 0$ . In contrast with the standard eigenvalue problem, the *characteristic polynomial* det $(A - \lambda B)$  can have degree less than n. This happens whenever B is a singular matrix. A pencil is *singular* if its characteristic polynomial is identically zero. In this case every  $\lambda$  is an eigenvalue. A pencil that is not singular is called *regular*.

The QZ algorithm of Moler and Stewart [38] is a generalization of the QR algorithm that can be used to solve generalized eigenvalue problems for regular pencils. This is just one of a whole family

of GZ algorithms [60]. A good implementation of a GZ algorithm will perform well, regardless of whether the *B* matrix is singular or not. However, it is much easier to explain how GZ algorithms work when *B* is nonsingular, so we shall make that assumption. One iteration of a GZ algorithm transforms a pencil  $A - \lambda B$  to a strictly equivalent pencil  $\hat{A} - \lambda \hat{B}$  as follows: a spectral transformation function *f* is chosen, then GR decompositions of  $f(AB^{-1})$  and  $f(B^{-1}A)$  are computed:

$$f(AB^{-1}) = GR, \quad f(B^{-1}A) = ZS.$$
 (6.1)

G and Z are nonsingular, and R and S are upper triangular. The nonsingular matrices G and Z are used to effect the equivalence transformation:

$$\hat{A} = G^{-1}AZ, \quad \hat{B} = G^{-1}BZ.$$
 (6.2)

If B = I, then we may take G = Z in (6.1), in which case the GZ iteration reduces to a GR iteration.

Recombining Eq. (6.2) we see immediately that

$$\hat{A}\hat{B}^{-1} = G^{-1}(AB^{-1})G$$
, and  $\hat{B}^{-1}\hat{A} = Z^{-1}(B^{-1}A)Z$ . (6.3)

Eqs. (6.1) and (6.3) together show that an iteration of the GZ algorithm effects GR iterations on  $AB^{-1}$  and  $B^{-1}A$  simultaneously. It follows then from the GR convergence theory that if we iterate this process with good choices of spectral transformation functions, both  $AB^{-1}$  and  $B^{-1}A$  will normally converge rapidly to block upper triangular form. It is shown in [60] that the A and B matrices converge individually (at the same rate as  $AB^{-1}$  and  $B^{-1}A$ ) to block triangular form

$$\begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix} - \lambda \begin{bmatrix} B_{11} & B_{12} \\ 0 & B_{22} \end{bmatrix},$$

thus breaking the problem into two smaller problems involving the pencils  $A_{11} - \lambda B_{11}$  and  $A_{22} - \lambda B_{22}$ .

These iterations are expensive unless can find an economical way to perform the equivalence transformation (6.2) without explicitly calculating  $B^{-1}$  (which may not exist), much less  $f(AB^{-1})$  or  $f(B^{-1}A)$ . This is done by performing an initial transformation to a condensed form, usually Hessenberg-triangular form. By this we mean that A is made upper Hessenberg and B upper triangular. (Thus  $AB^{-1}$  and  $B^{-1}A$  are both upper Hessenberg.) Then the GZ step is effected by a process that chases bulges through A and B. The bulges are first formed by a transformation  $G_1$  whose first column is proportional to the first column of  $f(AB^{-1})$ . This can be computed cheaply if f is a polynomial of degree  $p \ll n$ , since  $AB^{-1}$  is upper Hessenberg. It can be done without explicitly assembling  $B^{-1}$ , and it has a reasonable interpretation even if  $B^{-1}$  does not exist. Once the bulges have been formed, the rest of the iteration consists of a sequence of transformations that return the matrices. It is similar to the GR bulge-chasing process, but there are extra details. See [30,60], or the original Moler–Stewart paper [38].

The type of GZ iteration that the bulge chase effects depends on what kinds of transformations are used to do the chasing. For example, if all transformations are unitary, a QZ step results. If Gaussian elimination transformations (with pivoting) are used, an iteration of the LZ algorithm [35] results. Other examples are the SZ algorithm for symplectic butterfly pencils [6], and the HZ algorithm for pencils of the form  $T - \lambda D$ , where T is symmetric and D is diagonal with  $\pm 1$  entries on the main diagonal. This is a reformulation of the HR algorithm for matrices of the form DT ( $=D^{-1}T$ ).

Surely the most heavily used GZ code is the one in Matlab. This is a single-shift (p=1) implicit QZ algorithm that uses complex arithmetic. The original QZ algorithm of Moler and Stewart [38] used p = 2 for real matrices, following Francis. The QZ codes in LAPACK use either p = 1 or p = 2, depending on whether the shifts are real or complex.

As far as we know, no parallel QZ code has been written so far. The various approaches that have been tried for QR can also be applied to QZ. For example, one can take p > 2 and chase larger bulges [60], but this is more difficult to implement than in the QR case. Shift blurring is also a problem if p is too large. The idea of chasing many small bulges in pipeline fashion should work as well for QZ as it does for QR.

Once the QZ algorithm is finished, the pencil will have been reduced to upper triangular form or nearly triangular form. For simplicity let us suppose the form is triangular. Then the eigenvalues are the quotients of the main diagonal entries:  $\lambda_i = a_{ii}/b_{ii}$ . If  $a_{ii} \neq 0$  and  $b_{ii} = 0$  for some *i*, this signifies an infinite eigenvalue. If  $a_{ii} = 0$  and  $b_{ii} = 0$  for some *i*, the pencil is singular. In that case the other  $a_{jj}/b_{jj}$  signify nothing, as they can take on any values whatsoever [63]. Singular pencils have fine structure that can be determined by the staircase algorithm of Van Dooren [23]. See also the code GUPTRI of Demmel and Kågström [19].

#### 7. Divide-and-conquer algorithms

To round out the paper we consider a completely different class of algorithm that has been under development in recent years [3,4]. They are not usually viewed as GZ algorithms, but that is what they are. They are explicit GZ algorithms; that is, they actual compute  $f(AB^{-1})$  and  $f(B^{-1}A)$  and their GR decompositions explicitly. They require more computation than a conventional implicit GZ algorithm does, but the computations are of types that can be implemented using level 3 BLAS. They also have a divide-and-conquer aspect. Thus, algorithms of this type have a chance of becoming the algorithms of choice for parallel solution of extremely large, dense eigenvalue problems.

Let D be a subset of the complex plane (e.g. a disk) that contains some, say k, of the eigenvalues of the pencil  $A - \lambda B$ . Ideally  $k \approx n/2$ . Let  $f = \chi_D$ , the characteristic function of D. Thus f(z) is 1 if  $z \in D$  and 0 otherwise. If we then perform a GZ iteration (6.1, 6.2) driven by this f, the resulting pencil normally has the form

$$\hat{A} - \lambda \hat{B} = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix} - \lambda \begin{bmatrix} B_{11} & B_{12} \\ 0 & B_{22} \end{bmatrix},$$
(7.1)

where  $A_{11} - \lambda B_{11}$  is  $k \times k$  and carries the eigenvalues that lie within *D*. Thus in one (expensive) iteration we divide the problem into two subproblems, which are of about equal size if  $k \approx n/2$ . A few such divisions suffice to conquer the problem.

It is easy to see why the split occurs. Let  $\mathscr{S}_d$  and  $\mathscr{S}_r$  be the invariant subspaces of  $B^{-1}A$  and  $AB^{-1}$ , respectively, associated with the eigenvalues that lie in D. Then  $(\mathscr{S}_d, \mathscr{S}_r)$  is a deflating pair for the pencil, i.e.,  $A\mathscr{S}_d \subseteq \mathscr{S}_r$  and  $B\mathscr{S}_d \subseteq \mathscr{S}_r$ . Since f is the characteristic function of D,  $f(B^{-1}A)$  and  $f(AB^{-1})$  are spectral projectors onto  $\mathscr{S}_d$  and  $\mathscr{S}_r$ , respectively. When a decomposition  $f(AB^{-1})=GR$ 

is performed, the upper-triangular matrix R normally has the form

$$R = \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix},$$

where  $R_{11}$  is  $k \times k$  and nonsingular, because  $f(AB^{-1})$  has rank k. We can be sure of obtaining in this form if we introduce column pivoting in the GR decomposition:  $f(AB^{-1}) = GR\Pi$ , where R has the desired form and  $\Pi$  is a permutation matrix. This guarantees that the first k columns of G form a basis for  $\mathscr{S}_r$ , the range of  $f(AB^{-1})$ . If we likewise introduce pivoting into the decomposition of  $f(B^{-1}A)$ , we can guarantee that the first k columns of Z are a basis of  $\mathscr{S}_d$ . Thus, if we replace (6.1) by

$$f(AB^{-1}) = GR\Pi, \quad f(B^{-1}A) = ZSP,$$
(7.2)

where  $\Pi$  and P are suitable permutation matrices, then the transformation (6.2) will result in the form (7.1), because  $\mathscr{G}_d$  and  $\mathscr{G}_r$  are deflating subspaces.

This type of GZ algorithm yields a deflation on each iteration. In order to implement it, we need to be able to calculate  $f(AB^{-1})$  and  $f(B^{-1}A)$  for various types of regions D. Various iterative methods have been put forward. The main method discussed in [3] can be applied to an arbitrary disk D. The size and location of the disk are determined by a preliminary transformation. Therefore we can take D to be the unit disk without loss of generality. The iterative method described in [3] has the effect that if one stops after j iterations, one uses instead of f the rational approximation

$$f_j(z) = \frac{1}{1+z^{2^j}}.$$

Even for modest values of j this approximation is excellent, except very near the unit circle.

The matrices  $f_j(AB^{-1})$  and  $f_j(B^{-1}A)$  are computed without ever forming  $B^{-1}$ ; the algorithm operates directly on A and B. The major operations in the iteration are QR decompositions and matrix-matrix multiplications, which can be done in level 3 BLAS. In the decomposition (7.2) the matrices G and Z are taken to be unitary for stability, so this is actually a QZ algorithm. The algorithm works even if B is singular. See [3] for many more details.

Since the iterations that compute  $f_j(AB^{-1})$  and  $f_j(B^{-1}A)$  are expensive, one prefers not to perform too many of them. Difficulties arise when there is an eigenvalue on or very near the circle that divides D from its complement. The iterations may fail to converge or converge too slowly. The remedy is to move the disk and restart the iterations. Once the projectors and their QR decompositions have been computed, the transformation (6.2) does not deliver exactly the form (7.1). The (2,1) block will not quite be zero in practice, because of roundoff errors and because the projectors have been calculated only approximately. If  $||A_{21}||$  or  $||B_{21}||$  is too big, the iteration must be rejected. Again the remedy is to move the disk and try again. Because the iterations are so expensive, one cannot afford to waste too many of them.

An experimental divide-and-conquer code (that uses a different iteration from the one discussed here) is available as a prototype code from ScaLAPACK.

# References

 G.S. Ammar, W.B. Gragg, L. Reichel, On the eigenproblem for orthogonal matrices, in: Proceedings 25th IEEE Conference on Decision and Control, Athens, New York, 1986, pp. 1063–1066.

- [2] Z. Bai, J. Demmel, On a block implementation of the Hessenberg multishift QR iteration, Int. J. High Speed Comput. 1 (1989) 97–112.
- [3] Z. Bai, J. Demmel, M. Gu, Inverse free parallel spectral divide and conquer algorithms for nonsymmetric eigenproblems, Numer. Math. 76 (1997) 279–308.
- [4] Z. Bai et al., The spectral decomposition of nonsymmetric matrices on distributed memory parallel computers, Technical Report CS-95-273, University of Tennessee, 1995.
- [5] S. Batterson, J. Smillie, Rayleigh quotient iteration for nonsymmetric matrices, Math. Comp. 55 (1990) 169– 178.
- [6] P. Benner, H. Fassbender, D.S. Watkins, Two connections between the *SR* and HR eigenvalue algorithms, Linear Algebra Appl. 272 (1998) 17–32.
- [7] B. Bohnhorst, Ein Lanczos-ähnliches Verfahren zur Lösung des unitären Eigenwertproblems, Ph.D. Thesis, University of Bielefeld, 1993.
- [8] K. Braman, R. Byers, R. Mathias, The multi-shift *QR* algorithm: aggressive deflation, maintaining well focused shifts, and level 3 performance. Manuscript, 1999.
- [9] M.A. Brebner, J. Grad, Eigenvalues of  $Ax = \lambda Bx$  for real symmetric matrices A and B computed by reduction to pseudosymmetric form and the HR process, Linear Algebra Appl. 43 (1982) 99–118.
- [10] A. Bunse-Gerstner, An analysis of the HR algorithm for computing the eigenvalues of a matrix, Linear Algebra Appl. 35 (1981) 155–173.
- [11] A. Bunse-Gerstner, L. Elsner, Schur parameter pencils for the solution of the unitary eigenproblem, Linear Algebra Appl. 154–156 (1991) 741–778.
- [12] A. Bunse Gerstner, V. Mehrmann, A symplectic *QR*-like algorithm for the solution of the real algebraic Riccati equation, IEEE Trans. Auto. Control AC-31 (1986) 1104–1113.
- [13] A. Bunse-Gerstner, V. Mehrmann, D.S. Watkins, An *SR* algorithm for Hamiltonian matrices based on Gaussian elimination, Methods Oper. Res. 58 (1989) 339–358.
- [14] H.J. Buurema, A geometric proof of convergence for the *QR* method, Ph.D. Thesis, University of Groningen, Netherlands, 1970.
- [15] R. Byers, Hamiltonian and symplectic algorithms for the algebraic Riccati equation, Ph.D. Thesis, Cornell University, 1983.
- [16] R. Byers, A Hamiltonian QR algorithm, SIAM J. Sci. Stat. Comput. 7 (1986) 212–229.
- [17] D. Day, How the *QR* algorithm fails to converge and how to fix it. Mathematics of Numerical Analysis: Real Number Algorithms, August 1995.
- [18] J. Della-Dora, Numerical linear algorithms and group theory, Linear Algebra Appl. 10 (1975) 267-283.
- [19] J. Demmel, B. Kågström, GUPTRI. NETLIB, 1991. http://www.netlib.org/.
- [20] J.J. Dongarra, J.R. Bunch, C.B. Moler, G.W. Stewart, LINPACK Users' Guide, SIAM, Philadelphia, 1979.
- [21] J.J. Dongarra, J. Du Croz, I. Duff, S. Hammarling, A set of level 3 basic linear algebra subprograms, ACM Trans. Math. Software 16 (1990) 1–17.
- [22] J.J. Dongarra, J. Du Croz, S. Hammarling, R. Hanson, An extended set of Fortran basic linear algebra subprograms, ACM Trans. Math. Software 14 (1988) 1–17.
- [23] P. van Dooren, The computation of Kronecker's canonical form of a singular pencil, Linear Algebra Appl. 27 (1979) 103–140.
- [24] A.A. Dubrulle, The multishift *QR* algorithm is it worth the trouble?, Technical Report G320-3558x, IBM Corp., Palo Alto, 1991.
- [25] E. Anderson et al., LAPACK Users' Guide, SIAM, Philadelphia, 2nd Edition, 1995. http://www.netlib.org/ lapack/lug/lapack\_lug.html.
- [26] L.S. Blackford et al., ScaLAPACK Users' Guide, SIAM, Philadelphia, 1997. http://www.netlib.org/scalapack/ slug/scalapack\_slug.html.
- [27] K.V. Fernando, B.N. Parlett, Accurate singular values and differential qd algorithms, Numer. Math. 67 (1994) 191–229.
- [28] J.G.F. Francis, The QR transformation, Parts I and II, Computer J. 4 1961 265-272, 332-345.
- [29] G.A. Geist, G.W. Howell, D.S. Watkins, The BR eigenvalue algorithm, SIAM J. Matrix Anal. Appl. 20 (1999) 1083–1098.

- [30] G.H. Golub, C.F. Van Loan, Matrix Computations, Johns Hopkins University Press, Baltimore, 3rd Edition, 1996.
- [31] J.B. Haag, D.S. Watkins, *QR*-like algorithms for the nonsymmetric eigenvalue problem, ACM Trans. Math. Software 19 (1993) 407–418.
- [32] D.E. Heller, I.C.F. Ipsen, Systolic networks for orthogonal decompositions, SIAM J. Sci. Stat. Comput. 4 (1983) 261–269.
- [33] G. Henry, D. Watkins, J. Dongarra, A parallel implementation of the nonsymmetric *QR* algorithm for distributed memory architectures, Technical Report LAPACK Working Note 121, University of Tennessee, 1997.
- [34] A.S. Householder, The Theory of Matrices in Numerical Analysis, Dover Books, 1964.
- [35] L.C. Kaufman, The LZ algorithm to solve the generalized eigenvalue problems, SIAM J. Numer. Anal. 11 (1974) 997–1024.
- [36] L.C. Kaufman, A parallel *QR* algorithm for the symmetric tridiagonal eigenvalue problem, J. Parallel and Distributed Computing 23 (1994) 429–434.
- [37] V.N. Kublanovskaya, On some algorithms for the solution of the complete eigenvalue problem, USSR Comput. Math. Math. Phys. 3 (1961) 637–657.
- [38] C.B. Moler, G.W. Stewart, An algorithm for generalized matrix eigenvalue problems, SIAM J. Numer. Anal. 10 (1973) 241–256.
- [39] C. Paige, C. Van Loan, A Schur decomposition for Hamiltonian matrices, Linear Algebra Appl. 41 (1981) 11–32.
- [40] B.N. Parlett, The Symmetric Eigenvalue Problem, Prentice-Hall, Englewood Cliffs, New Jersey, 1980. Reprinted by SIAM, 1997.
- [41] B.N. Parlett, The new qd algorithms, Acta Numerica (1995) 459–491.
- [42] B.N. Parlett, W.G. Poole, A geometric theory for the *QR*, *LU*, and power iterations, SIAM J. Numer. Anal. 10 (1973) 389–412.
- [43] H. Rutishauser, Der Quotienten-Differenzen-Algorithmus, Z. Angew. Math. Physik 5 (1954) 233-251.
- [44] H. Rutishauser, Der Quotienten-Differenzen-Algorithmus, Mitt. Inst. Angew. Math. ETH, Vol. 7, Birkhäuser, Birkhäuser, Basel, 1957.
- [45] H. Rutishauser, Solution of eigenvalue problems with the LR-transformation, Nat. Bur. Standards Appl. Math. Series 49 (1958) 47–81.
- [46] B.T. Smith et al., Matrix Eigensystem Routines EISPACK Guide, Springer, Berlin, 2nd Edition, 1976.
- [47] G.W. Stewart, A parallel implementation of the QR algorithm, Parallel Comput. 5 (1987) 187–196.
- [48] R. van de Geijn, D.G. Hudson, An efficient parallel implementation of the nonsymmetric *QR* algorithm, in: Proceedings of the Fourth Conference on Hypercube Concurrent Computers and Applications, 1989.
- [49] R.A. van de Geijn, Implementing the *QR* algorithm on an array of processors, Ph.D. Thesis, University of Maryland, 1987. Department of Computer Science TR-1897.
- [50] R.A. van de Geijn, Deferred shifting schemes for parallel QR methods, SIAM J. Matrix Anal. Appl. 14 (1993) 180–194.
- [51] D.S. Watkins, Understanding the QR algorithm, SIAM Rev. 24 (1982) 427-440.
- [52] D.S. Watkins, Fundamentals of Matrix Computations, Wiley, New York, 1991.
- [53] D.S. Watkins, Some perspectives on the eigenvalue problem, SIAM Rev. 35 (1993) 430-471.
- [54] D.S. Watkins, Shifting strategies for the parallel QR algorithm, SIAM J. Sci. Comput. 15 (1994) 953–958.
- [55] D.S. Watkins, QR-like algorithms an overview of convergence theory and practice, in: J. Renegar, M. Shub, S. Smale (Eds.), The Mathematics of Numerical Analysis, Lectures in Applied Mathematics, Vol. 22, American Mathematical Society, Providence, RI, 1996.
- [56] D.S. Watkins, The transmission of shifts and shift blurring in the *QR* algorithm, Linear Algebra Appl. 241–243 (1996) 877–896.
- [57] D.S. Watkins, Bulge exchanges in algorithms of QR type, SIAM J. Matrix Anal. Appl. 19 (1998) 1074–1096.
- [58] D.S. Watkins, L. Elsner, Chasing algorithms for the eigenvalue problem, SIAM J. Matrix Anal. Appl. 12 (1991) 374–384.
- [59] D.S. Watkins, L. Elsner, Convergence of algorithms of decomposition type for the eigenvalue problem, Linear Algebra Appl. 143 (1991) 19–47.
- [60] D.S. Watkins, L. Elsner, Theory of decomposition and bulge-chasing algorithms for the generalized eigenvalue problem, SIAM J. Matrix Anal. Appl. 15 (1994) 943–967.

- [61] J.H. Wilkinson, The Algebraic Eigenvalue Problem, Clarendon Press, Oxford University, 1965.
- [62] J.H. Wilkinson, Global convergence of tridiagonal *QR* algorithm with origin shifts, Linear Algebra Appl. 1 (1968) 409–420.
- [63] J.H. Wilkinson, Kronecker's canonical form and the QZ algorithm, Linear Algebra Appl. 28 (1979) 285–303.



JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

Journal of Computational and Applied Mathematics 123 (2000) 85-100

www.elsevier.nl/locate/cam

# The ubiquitous Kronecker product

Charles F. Van Loan<sup>1</sup>

Department of Computer Science, Cornell University, Ithaca, New York 14853, USA

Received 26 October 1999

#### Abstract

The Kronecker product has a rich and very pleasing algebra that supports a wide range of fast, elegant, and practical algorithms. Several trends in scientific computing suggest that this important matrix operation will have an increasingly greater role to play in the future. First, the application areas where Kronecker products abound are all thriving. These include signal processing, image processing, semidefinite programming, and quantum computing. Second, sparse factorizations and Kronecker products are proving to be a very effective way to look at fast linear transforms. Researchers have taken the Kronecker methodology as developed for the fast Fourier transform and used it to build exciting alternatives. Third, as computers get more powerful, researchers are more willing to entertain problems of high dimension and this leads to Kronecker products whenever low-dimension techniques are "tensored" together. © 2000 Elsevier Science B.V. All rights reserved.

#### 1. Basic properties

If  $B \in \mathbb{R}^{m_1 \times n_1}$  and  $C \in \mathbb{R}^{m_2 \times n_2}$ , then their *Kronecker product*  $B \otimes C$  is an  $m_1 \times n_1$  block matrix whose (i, j) block is the  $m_2 \times n_2$  matrix  $b_{ij}C$ . Thus,

$$\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \otimes \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} = \begin{bmatrix} b_{11}c_{11} & b_{11}c_{12} & b_{11}c_{13} & b_{12}c_{11} & b_{12}c_{12} & b_{12}c_{23} \\ b_{11}c_{21} & b_{11}c_{22} & b_{11}c_{23} & b_{12}c_{21} & b_{12}c_{22} & b_{12}c_{23} \\ b_{11}c_{31} & b_{11}c_{32} & b_{11}c_{33} & b_{12}c_{31} & b_{12}c_{32} & b_{12}c_{33} \\ b_{21}c_{11} & b_{21}c_{12} & b_{21}c_{13} & b_{22}c_{11} & b_{22}c_{12} & b_{22}c_{13} \\ b_{21}c_{21} & b_{21}c_{22} & b_{21}c_{23} & b_{22}c_{21} & b_{22}c_{22} & b_{22}c_{23} \\ b_{21}c_{31} & b_{21}c_{32} & b_{21}c_{33} & b_{22}c_{31} & b_{22}c_{31} & b_{22}c_{32} & b_{22}c_{33} \\ b_{21}c_{31} & b_{21}c_{32} & b_{21}c_{33} & b_{22}c_{31} & b_{22}c_{32} & b_{22}c_{33} \\ b_{21}c_{31} & b_{21}c_{32} & b_{21}c_{33} & b_{22}c_{31} & b_{22}c_{32} & b_{22}c_{33} \\ b_{21}c_{31} & b_{21}c_{32} & b_{21}c_{33} & b_{22}c_{31} & b_{22}c_{32} & b_{22}c_{33} \\ b_{21}c_{31} & b_{21}c_{32} & b_{21}c_{33} & b_{22}c_{31} & b_{22}c_{32} & b_{22}c_{33} \\ b_{21}c_{31} & b_{21}c_{32} & b_{21}c_{33} & b_{22}c_{31} & b_{22}c_{32} & b_{22}c_{33} \\ b_{21}c_{31} & b_{21}c_{32} & b_{21}c_{33} & b_{22}c_{31} & b_{22}c_{32} & b_{22}c_{33} \\ b_{21}c_{31} & b_{21}c_{32} & b_{21}c_{33} & b_{22}c_{31} & b_{22}c_{32} & b_{22}c_{33} \\ b_{21}c_{31} & b_{21}c_{32} & b_{21}c_{33} & b_{22}c_{31} & b_{22}c_{32} & b_{22}c_{33} \\ b_{21}c_{31} & b_{21}c_{32} & b_{21}c_{33} & b_{22}c_{31} & b_{22}c_{32} & b_{22}c_{33} \\ b_{21}c_{31} & b_{21}c_{32} & b_{21}c_{33} & b_{22}c_{31} & b_{22}c_{31} & b_{22}c_{32} & b_{22}c_{33} \\ b_{21}c_{31} & b_{21}c_{32} & b_{22}c_{31} & b_{22}c_{31} & b_{22}c_{32} & b_{22}c_{33} \\ b_{21}c_{31} & b_{31}c_{32} & b_{31}c_{32} & b_{32}c_{31} & b_{32}c_{32} & b_{32}c_{32} & b_{32}c_{33} \\ b_{31}c_{31} & b_{31}c_{32} & b_{31}c_{32} & b_{32}c_{31} & b_{32}c_{32} & b_{32}c_{32} & b_{32}c_{32} & b_{32}c_{33} \\ b_{31}c_{31} & b_{31}c_{32} & b_{31}c_{32} & b_{31}c_{32} & b_{32}c_{31} & b_{31}c_{32} & b_{31}c_{32} & b_{31}c_{32} & b_{32}c_{31}$$

The basic properties of the Kronecker product are quite predictable:

$$(B \otimes C)^{\mathrm{T}} = B^{\mathrm{T}} \otimes C^{\mathrm{T}},$$
  
 $(B \otimes C)^{-1} = B^{-1} \otimes C^{-1}$ 

<sup>&</sup>lt;sup>1</sup> Supported in part by the NSF contract CCR-9901988. *E-mail address:* cv@cs.cornell.edu (C.F. Van Loan).

 $(B \otimes C)(D \otimes F) = BD \otimes CF,$  $B \otimes (C \otimes D) = (B \otimes C) \otimes D.$ 

Of course, the indicated products and inverses must exist for the second and third identities to hold.

The entries of  $B \otimes C$  and  $C \otimes B$  consist of all possible products of a *B*-matrix entry with a *C*-matrix entry and this raises the possibility that these two Kronecker products are related by a permutation. The permutation involved is in fact the *perfect shuffle*. If *p* and *q* are positive integers and r = pq, then the (p,q) perfect shuffle is the  $r \times r$  matrix

$$S_{p,q} = \begin{bmatrix} I_r(1:q:r,:) \\ I_r(2:q:r,:) \\ \vdots \\ I_r(q:q:r,:) \end{bmatrix}$$
(1)

where  $I_r$  is the  $r \times r$  identity. (The well-known "colon notation" used in MATLAB to designate submatrices is being used here.) In effect, the matrix-vector product  $S_{p,q}x$  takes the "card deck" x, splits it into p piles of length-q each, and then takes one card from each pile in turn until the deck is reassembled. It can be shown that if  $B \in \mathbb{R}^{m_1 \times n_1}$  and  $C \in \mathbb{R}^{m_2 \times n_2}$ , then

$$C \otimes B = S_{m_1, m_2}(B \otimes C)S_{n_1, n_2}^{\mathrm{T}}$$

Henderson and Searle [36] survey the numerous connections between the Kronecker product and the perfect shuffle. Additional observations about the Kronecker product may be found in [30,14,37,63]. Henderson et al. [35] look at the operation from the historical point of view.

Particularly important in computational work are the issues that surround the exploitation of structure and the application of matrix factorizations. By and large, a Kronecker product inherits structure from its factors. For example,

	( nonsingular		( nonsingular	)
	lower(upper) triangular		lower(upper) triangular	
if B and C are $\langle$	banded	$\rangle$ , then $B \otimes C$ is $\langle$	block banded	<pre></pre>
	symmetric		symmetric	
	positive definite		positive definite	
	stochastic		stochastic	
	Toeplitz		block Toeplitz	
	permutations		a permutation	
	orthogonal		orthogonal	J

With respect to factorizations, the LU-with-partial-pivoting, Cholesky, and QR factorizations of  $B \otimes C$  merely require the corresponding factorizations of *B* and *C*:

$$B \otimes C = (P_B^{\mathrm{T}} L_B U_A) \otimes (P_C^{\mathrm{T}} L_C U_C) = (P_B \otimes P_C)^{\mathrm{T}} (L_B \otimes L_C) (U_B \otimes U_C),$$
  

$$B \otimes C = (G_B G_B^{\mathrm{T}}) \otimes (G_C G_C^{\mathrm{T}}) = (G_B \otimes G_C) (G_B \otimes G_C)^{\mathrm{T}},$$
  

$$B \otimes C = (Q_B R_B) \otimes (Q_C R_C) = (Q_B \otimes Q_C) (R_B \otimes R_C).$$

The same is true for the singular value and Schur decompositions if we disregard ordering issues. In contrast, the CS and QR-with-column pivoting factorizations of  $B \otimes C$  do not have simple relationships to the corresponding factorizations of B and C. (The matrix factorizations and decompositions mentioned in this paper are all described in [29]).

In Kronecker product work, matrices are sometimes regarded as vectors and vectors are sometimes "made into" into matrices. To be precise about these reshapings we use the vec operation. If  $X \in \mathbb{R}^{m \times n}$ , then vec(X) is an  $nm \times 1$  vector obtained by "stacking" X's columns. If C, X, and B are matrices and the product  $CXB^{T}$  is defined, then it is not hard to establish the following equivalence:

$$Y = CXB^{\mathrm{T}} \equiv y = (B \otimes C)x \tag{2}$$

where x = vec(X) and y = vec(Y). Henderson and Searle [36] thoroughly discuss the vec/Kronecker product connection.

A consequence of the above properties is that linear systems of the form  $(B \otimes C)x = f$  can be solved *fast*. For example, if  $B, C \in \mathbb{R}^{m \times m}$ , then x can be obtained in  $O(m^3)$  flops via the LU factorizations of B and C. Without the exploitation of structure, an  $m^2 \times m^2$  system would normally require  $O(m^6)$  flops to solve.

As with any important mathematical operation, the Kronecker product has been specialized and modified to address new and interesting applications. Rauhala [58] presents a theory of "array algebra" that applies to certain photogrammetric problems. See also [62]. Regalia and Mitra [60] have used the Kronecker product and various generalizations of it to describe a range of fast unitary transforms. A sample generalization that figures in their presentation is

$$\{A_1,\ldots,A_m\}``\otimes " B = \begin{bmatrix} A_1 \otimes B(1,:) \\ A_2 \otimes B(2,:) \\ \vdots \\ A_m \otimes B(m,:) \end{bmatrix}$$

where  $A_1, \ldots, A_m$  are given matrices of the same size and B has m rows.

Another generalization, the *strong Kronecker product*, is developed in [61] and supports the analysis of certain orthogonal matrix multiplication problems. The strong Kronecker product of an  $m \times p$  block matrix  $B = (B_{ij})$  and a  $p \times n$  block matrix  $C = (C_{ij})$  is an  $m \times n$  block matrix  $A = (A_{ij})$  where  $A_{ij} = B_{i1} \otimes C_{1j} + \cdots + B_{ip} \otimes C_{pj}$ .

Kronecker product problems arise in photogrammetry [59], image processing [34], computer vision [47], and system theory [6]. They surface in the analysis of generalized spectra [2], stochastic models [57], and operator theory [64]. They have even found their way into the analysis of chess endgames [65].

To make sense of the "spread" of the Kronecker product, we have organized this paper around a few important families of applications. These include the linear matrix equation problem, fast linear transforms, various optimization problems, and the idea of preconditioning with Kronecker products.

#### 2. Matrix equations

To researchers in numerical linear algebra, the most familiar problem where Kronecker products arise is the *Sylvester* matrix equation problem. Here we are given  $F \in \mathbb{R}^{m \times m}$ ,  $G \in \mathbb{R}^{n \times n}$ , and  $C \in \mathbb{R}^{m \times n}$  and seek  $X \in \mathbb{R}^{m \times n}$  so that  $FX + XG^{T} = C$ . Linear systems of this variety play a central role in

control theory [13], Poisson equation solving [18], and invariant subspace computation [29]. In light of Eq. (2), the act of finding X is equivalent to solving the  $mn \times mn$  linear system

$$(I_n \otimes F + G \otimes I_m) \operatorname{vec}(X) = \operatorname{vec}(C).$$

The Lyapunov problem results if F = G, a very important special case.

One family of effective methods for these problems involve reducing F and G to Hessenberg or triangular form via orthogonal similarity transformations [5,27].

The more general matrix equation  $F_1XG_1^T + F_2XG_2^T = C$  can be handled using the generalized Schur decomposition as discussed by Gardiner et al. [24]. However, these factorization approaches break down for the general Sylvester matrix equation problem [49,74].

$$F_1 X G_1^{\mathrm{T}} + \dots + F_p X G_p^{\mathrm{T}} = C \equiv (G_1 \otimes F_1 + \dots + G_p \otimes F_p) \operatorname{vec}(X) = \operatorname{vec}(C).$$
(3)

Related to this are linear systems where the matrix of coefficients has blocks that are themselves Kronecker products, e.g.,

$$\begin{bmatrix} F_{11} \otimes G_{11} \cdots F_{1p} \otimes G_{1p} \\ \vdots & \ddots & \vdots \\ F_{p1} \otimes G_{p1} \cdots F_{pp} \otimes G_{pp} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_p \end{bmatrix} = \begin{bmatrix} c_1 \\ \vdots \\ c_p \end{bmatrix}.$$
(4)

(Clearly the dimensions of the matrices  $F_{ij}$  and  $G_{ij}$  have to "make sense" when compared to the dimensions of the vectors  $x_i$  and  $c_i$ .) This is equivalent to a system of generalized Sylvester equations:

$$\sum_{j=1}^{p} F_{ij} X_j G_{ij}^{\mathrm{T}} = C_i, \quad i = 1 : p$$

where  $vec(X_i) = x_i$  and  $vec(C_i) = c_i$  for i = 1 : p. We can solve a linear system Ax = b fast if A is a Kronecker product. But fast solutions seem problematical for (4).

A problem of this variety arises in conjunction with the generalized eigenproblem  $M - \lambda N$  where important subspace calculations require the solution of a system of the form

$$\begin{bmatrix} I_n \otimes A & -B^{\mathrm{T}} \otimes I_m \\ I_n \otimes D & -E^{\mathrm{T}} \otimes I_m \end{bmatrix} \begin{bmatrix} \operatorname{vec}(R) \\ \operatorname{vec}(L) \end{bmatrix} = \begin{bmatrix} \operatorname{vec}(C) \\ \operatorname{vec}(F) \end{bmatrix}$$

Here the matrices  $A, D \in \mathbb{R}^{m \times m}$ ,  $B, E \in \mathbb{R}^{n \times n}$ , and  $C, F \in \mathbb{R}^{m \times n}$  are given and the matrices  $L, R \in \mathbb{R}^{m \times n}$  are sought [42].

Another area where block systems arise with Kronecker product blocks is semidefinite programming. There has been an explosion of interest in this area during the last few years due largely to the applicability of interior point methods; see [69,71]. An important feature of these methods is that they frequently require the solution of linear systems that involve the symmetric Kronecker product  $\otimes$ . For symmetric  $X \in \mathbb{R}^{n \times n}$  and arbitrary  $B, C \in \mathbb{R}^{n \times n}$  this operation is defined by

$$(B \otimes C)$$
svec $(X) =$  svec $\left(\frac{1}{2}(CXB^{T} + BXC^{T})\right)$ 

where the "svec" operation is a normalized stacking of X's subdiagonal columns, e.g.,

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix} \Rightarrow \operatorname{svec}(X) = [x_{11}, \sqrt{2}x_{21}, \sqrt{2}x_{31}, x_{22}, \sqrt{2}x_{32} & x_{33}]^{\mathrm{T}};$$

$$\begin{bmatrix} 0 & A^{1} & I \\ A & 0 & 0 \\ Z \otimes I & 0 & X \otimes I \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = \begin{bmatrix} r_{d} \\ r_{p} \\ r_{c} \end{bmatrix}$$

# 3. Least squares

Least squares problems of the form

$$\min ||(B \otimes C)x - b||$$

can be efficiently solved by computing the QR factorizations (or SVDs) of *B* and *C*; [21,22]. Barrlund [4] shows how to minimize  $||(A_1 \otimes A_2)x - f||$  subject to the constraint that  $(B_1 \otimes B_2)x = g$ , a problem that comes up in surface fitting with certain kinds of splines.

Coleman et al. [11] describe an interesting least-squares problem that arises in segmentation analysis. It is the minimization of

$$\left\| W\left( \begin{bmatrix} I_n \otimes D_m \\ D_n \otimes I_m \\ \lambda I \end{bmatrix} x - \begin{bmatrix} b_1 \\ b_2 \\ 0 \end{bmatrix} \right) \right\|_2$$
(5)

where W is diagonal, and the D matrices are upper bidiagonal. The scaling matrix W and the  $\lambda I$  block seem to rule out obvious fast factorization approaches.

On the other hand, LS problems of the form

$$\min \left\| \begin{bmatrix} B_1 \otimes C_1 \\ B_2 \otimes C_2 \end{bmatrix} x - b \right\|_2$$
(6)

can be solved fast by computing the generalized singular value decomposition of the pairs  $(B_1, B_2)$ and  $(C_1, C_2)$ :

$$B_{1} = U_{1B}D_{1B}X_{B}^{T} \qquad B_{2} = U_{2B}D_{2B}X_{B}^{T}, C_{1} = U_{1C}D_{1C}X_{C}^{T} \qquad C_{2} = U_{2C}D_{2C}X_{C}^{T}$$

Here, the U's are orthogonal, the D's are diagonal, and the X's are nonsingular. With these decompositions the matrices in (6) can be transformed to diagonal form since

$$\begin{bmatrix} B_1 \otimes C_1 \\ B_2 \otimes C_2 \end{bmatrix} = \begin{bmatrix} U_{1B} \otimes U_{2B} & 0 \\ 0 & U_{1C} \otimes U_{2C} \end{bmatrix} \begin{bmatrix} D_{1B} \otimes D_{2B} \\ D_{1C} \otimes D_{2C} \end{bmatrix} X_B^{\mathsf{T}} \otimes X_C^{\mathsf{T}}.$$

The solution of the converted problem is straightforward and the overall cost of the procedure is essentially the cost of the two generalized SVDs.

The total least squares (TLS) problem is another example of just how little it takes to stifle the easy exploitation Kronecker products in a matrix computation. A TLS solution to  $(B \otimes C)x \approx b$  requires the computation of the smallest singular value and the associated left and right singular vectors of the augmented matrix

$$M = [B \otimes C \mid b].$$

If  $B \in \mathbb{R}^{m_1 \times n_1}$  and  $C \in \mathbb{R}^{m_2 \times n_2}$ , then the SVD of  $B \otimes C$  costs  $O(m_1n_1^2 + m_2n_2^2)$  while the SVD of M appears to require  $O((m_1m_2)(n_1n_2)^2)$ . However, in this case the special structure of M permits the fast calculation of the required minimum singular triple via the coupling of condition estimation ideas with the QR factorization of  $B \otimes C$ .

# 4. Tensoring low-dimension ideas

Tensor product "ideas" in approximation and interpolation also lead to Kronecker product problems. In these multidimensional situations the "overall" method involves repetition of the same 1-dimensional idea in each coordinate direction. For example, if 1-dimensional quadrature rules of the form  $\int f(x) dx \approx \sum w_i f(x_i) \equiv w^T f(x)$  are applied in the *x*, *y*, and *z* directions to the triple integral

$$I = \int_{a_1}^{b_1} \int_{a_2}^{b_2} \int_{a_3}^{b_3} g(x, y, z) \,\mathrm{d}x \,\mathrm{d}y \,\mathrm{d}z,$$

then we obtain

$$I \approx \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} \sum_{k=1}^{n_z} w_i^{(x)} w_j^{(y)} w_k^{(z)} g(x_i, y_j, z_k) = (w^{(x)} \otimes w^{(y)} \otimes w^{(z)})^{\mathrm{T}} g(x \otimes y \otimes z)$$

where  $x \in \mathbb{R}^{n_x}$ ,  $y \in \mathbb{R}^{n_y}$ , and  $z \in \mathbb{R}^{n_z}$  are vectors of abscissa values and  $g(x \otimes y \otimes z)$  designates the vector of values obtained by evaluating g at each component of  $x \otimes y \otimes z$ . For further details about this kind of multidimensional problem [15,16]. The computationally oriented papers by Pereyra and Scherer [55] and de Boor [17] are motivated by these tensor product applications and are among the earliest references that discuss how to organize a Kronecker product calculation.

The ability to solve problems with increasingly high dimension because of very powerful computers partially explains the heightened profile of the Kronecker product in scientific computing. Interest in higher-order statistics is a good example. Roughly speaking, second-order statistics revolve around the expected value of  $xx^{T}$  where x is a random vector. In higher-order statistics the calculations involve the "cumulants"  $x \otimes x \otimes \cdots \otimes x$ . (Note that  $vec(xx^{T}) = x \otimes x$ .) [67,1]. Related Kronecker product computations arise in Volterra filtering as presented in [54,53]. A collection of very high dimensional Kronecker product problems that arise in statistical mechanics and quantum mechanics is discussed [63].

# 5. Fast transforms

Kronecker products and various generalizations are what "drive" many fast transform algorithms. Consider the fast Fourier transform (FFT) with  $n = 2^t$ . If  $P_n$  is the *bit reversal permutation* 

$$P_n = S_{2,n/2}(I_2 \otimes S_{2,n/4}) \cdots (I_{n/4} \otimes S_{2,2})$$

and  $\omega_n = \exp(-2\pi i/n)$ , then the discrete Fourier transform (DFT) matrix  $F_n = (\omega_n^{pq}) \in \mathbb{C}^{n \times n}$  can be factored as  $F_n = A_t \cdots A_1 P_n$  where

$$A_q = I_r \otimes \begin{bmatrix} I_{L/2} & \Omega_{L/2} \\ I_{L/2} & -\Omega_{L/2} \end{bmatrix},$$

with  $L = 2^q$ , r = n/L,  $\Omega_{L/2} = \text{diag}(1, \omega_L, \dots, \omega_L^{L/2-1})$ , and  $\omega_L = \exp(-2\pi i/L)$ . Based on this "sparse" factorization of the DFT matrix we obtain the Cooley–Tukey FFT framework for computing the DFT  $y = F_n x = A_1 \cdots A_1 P_n x$ :

$$x \leftarrow P_n x$$
  
for  $k = 1 : t$   
 $x \leftarrow A_q x$   
end  
 $y \leftarrow x$ 

Tolimieri et al. [70] and Van Loan [72] have shown how the organization of the FFT is clarified through the "language" of Kronecker products. Different FFT algorithms correspond to different factorizations of  $F_n$ . The value of this point of view is that it unifies the literature and exposes simple connections between seemingly disparate algorithms. For example, the Gentleman–Sande FFT framework results by taking transposes in  $F_n = A_t \cdots A_1 P_n$  and noting that  $F_n$  and  $P_n$  are symmetric:

```
for k = 1 : t

x \leftarrow A_q^{\mathsf{T}} x

end

y \leftarrow P_n x
```

The evolution of FFT ideas and algorithms would have proceeded much more rapidly and with greater clarity from the famous 1965 Cooley–Tukey paper onwards had the Kronecker notation been more actively employed.

Huang et al. [39] have developed a parallel programming methodology that revolves around the Kronecker product. Related papers include Johnson et al. [41], and Granata et al. [31,32]. Pitsianis [56] built a "Kronecker compiler" that permits the user to specify algorithms in a Kronecker product language. The compiler is based on a set of term rewriting rules that translate high-level, Kronecker-based matrix descriptions of an algorithm into any imperative language such as C, MATLAB or Fortran. The efficiency of the automatically generated code is shown to be excellent.

The Kronecker product methodology extends beyond the FFT and the related sine/cosine transforms. Indeed, Regalia and Mitra [60] have used the Kronecker product and various generalizations of it to describe a range of fast unitary transforms; See also [40] for Kronecker presentations of the Walsh–Hadamard, slant, and Hartley transforms. Strohmer [66] uses a Kronecker product framework to develop factorizations for the Gabor frame operator while Fijany and Williams [23] do the same thing with quantum wavelet transforms. Kumar et al. [48] use Kronecker product ideas to develop a memory-efficient implementation of the Strassen matrix multiply.

Kronecker products and sparse factorizations are as central to fast wavelet transforms as they are to the FFT. Consider the Haar wavelet transform  $y = W_n x$  where  $n = 2^t$ . The transform matrix  $W_n$  is defined by

$$W_n = \begin{bmatrix} W_m \otimes \begin{pmatrix} 1 \\ 1 \end{pmatrix} \mid I_m \otimes \begin{pmatrix} 1 \\ -1 \end{pmatrix} \end{bmatrix} \quad n = 2m$$

with  $W_1 = [1]$ . It can be shown that

$$W_n = S_{2,m}(W_2 \otimes I_m) \begin{bmatrix} W_m & 0 \\ 0 & I_m \end{bmatrix} \quad n = 2m$$

and from this "splitting" it is possible to show that  $W_n = H_t \cdots H_1$  where

$$H_{q} = \begin{bmatrix} S_{2,L_{*}} & 0 \\ 0 & I_{n-L} \end{bmatrix} \begin{bmatrix} W_{2} \otimes I_{L_{*}} & 0 \\ 0 & I_{n-L} \end{bmatrix} \quad L = 2^{q}, \ L_{*} = L/2.$$

The fast Haar transform then proceeds as a sequence of matrix-vector products, i.e.,  $x \leftarrow H_1 x$ ,  $x \leftarrow H_2 x$ , ...,  $x \leftarrow H_t x$ .

More complicated wavelets require more sophisticated Kronecker manipulations in order to derive the underlying sparse factorization. For example, the n = 4 transform matrix for the Daubechies wavelet is given by

$$D_4 = \begin{bmatrix} c_0 & c_1 & c_2 & c_3 \\ c_3 & -c_2 & c_1 & -c_0 \\ c_2 & c_3 & c_0 & c_1 \\ c_1 & -c_0 & c_3 & -c_2 \end{bmatrix} \quad \text{where } \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \frac{1}{4\sqrt{2}} \begin{bmatrix} 1 + \sqrt{3} \\ 3 + \sqrt{3} \\ 3 - \sqrt{3} \\ 1 - \sqrt{3} \end{bmatrix}.$$

It is easy to verify that  $D_4$  is orthogonal. The n = 8 version is given as follows:

$$D_8 = egin{bmatrix} c_0 & c_1 & c_2 & c_3 & 0 & 0 & 0 & 0 \ \hline c_3 & -c_2 & c_1 & -c_0 & 0 & 0 & 0 & 0 \ \hline 0 & 0 & c_0 & c_1 & c_2 & c_3 & 0 & 0 \ \hline 0 & 0 & 0 & 0 & c_0 & c_1 & c_2 & c_3 \ \hline 0 & 0 & 0 & 0 & c_0 & c_1 & c_2 & c_3 \ \hline 0 & 0 & 0 & 0 & c_3 & -c_2 & c_1 & -c_0 \ \hline c_2 & c_3 & 0 & 0 & 0 & 0 & c_0 & c_1 \ c_1 & -c_0 & 0 & 0 & 0 & 0 & c_3 & -c_2 \ \end{bmatrix}$$

which clearly has a replicated block structure. It is possible to describe this structure quite elegantly with a generalized Kronecker product. It is then a straightforward exercise to obtain a splitting that relates  $D_8$  to  $D_4$  and more generally,  $D_n$  to  $D_{n/2}$ . From the splitting one can then derive the sparse factorization associated with the underlying fast transform [23].

It is almost always the case that behind every fast linear transform is a sparse, Kronecker-based, factorization of the transform matrix. Notable exceptions are the recent and very interesting fast algorithms whose complexity has the form cn or  $cn \log n$  where c is a constant that depends on the precision required. Examples include the fast Gauss transform of Greengard and Strain [33] and the non-uniformly spaced FFT of Dutt and Rokhlin [19]. It is interesting to conjecture whether these algorithms can be described in terms of some approximate sparse, Kronecker-based factorization of the underlying transform matrix.

Fast transforms often require various matrix transpositions of the data and these operations also submit to Kronecker product descriptions. For example, it follows from (1) that if  $A \in \mathbb{R}^{m \times n}$  and  $B = A^{T}$ , then  $\text{vec}(B) = S_{n,m} \cdot \text{vec}(A)$ . It turns out that different "multi-pass" transposition algorithms correspond to different factorizations of the underlying perfect shuffle. If

$$S_{n,m} = \Gamma_t \cdots \Gamma_1 \tag{7}$$

then  $B = A^{T}$  can be computed with t passes through the data as follows:

 $a = \operatorname{vec}(A)$ for k = 1 : t $a \leftarrow \Gamma_k a$ end Define  $B \in \mathbb{R}^{n \times m}$  by  $\operatorname{vec}(B) = a$ .

The idea is to choose a factorization (7) so that the data motion behind the operation kth pass, i.e.,  $a \leftarrow \Gamma_k a$ , is in harmony with the architecture of the underlying memory hierarchy.

To illustrate this factorization idea, it can be shown that if m = pn, then  $S_{n,m} = \Gamma_2 \Gamma_1$  where

$$\Gamma_1 = S_{n, p} \otimes I_n,$$
  
$$\Gamma_2 = I_p \otimes S_{n, n}.$$

The first pass  $b^{(1)} = \Gamma_1 \text{vec}(A)$  corresponds to a block transposition while  $b^{(2)} = \Gamma_2 b^{(1)}$  carries out the transposition of the blocks. For example, if p = 4 and

$$A = \begin{bmatrix} A_1 \\ A_2 \\ A_3 \\ A_4 \end{bmatrix}, \quad A_i \in \mathbb{R}^{n \times n}$$

then the  $\Gamma_1$  update leaves us with

$$B^{(1)} = [A_1 | A_2 | A_3 | A_4].$$

During the  $\Gamma_2$  update the individual blocks are transposed yielding

$$B = B^{(2)} = [A_1^{\mathrm{T}} | A_2^{\mathrm{T}} | A_3^{\mathrm{T}} | A_4^{\mathrm{T}}].$$

See [72] for more details about factorizations and matrix transposition.

# 6. The nearest Kronecker product problem

Suppose  $A \in \mathbb{R}^{m \times n}$  is given with  $m = m_1 m_2$  and  $n = n_1 n_2$ . For these integer factorizations the nearest Kronecker product (NKP) problem involves minimizing

$$\phi(B,C) = ||A - B \otimes C||_F \tag{8}$$

where  $B \in \mathbb{R}^{m_1 \times n_1}$  and  $C \in \mathbb{R}^{m_2 \times n_2}$ . Van Loan and Pitsianis [73] show how to solve the NKP problem using the singular value decomposition of a permuted version of A. This result is central to much of the research proposal and so we use a small example to communicate the main idea. Suppose  $m_1 = 3$  and  $n_1 = m_2 = n_2 = 2$ . By carefully thinking about the sum of squares that define  $\phi$  we see that

$$\phi(B,C) = \left\| \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \\ a_{51} & a_{52} & a_{53} & a_{54} \\ a_{61} & a_{62} & a_{63} & a_{64} \end{bmatrix} - \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} \otimes \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \right\|_{F}$$
$$= \left\| \begin{bmatrix} a_{11} & a_{21} & a_{12} & a_{22} \\ a_{31} & a_{41} & a_{32} & a_{42} \\ a_{51} & a_{61} & a_{52} & a_{62} \\ a_{13} & a_{23} & a_{14} & a_{24} \\ a_{33} & a_{43} & a_{34} & a_{44} \\ a_{53} & a_{63} & a_{54} & a_{64} \end{bmatrix} - \begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \\ b_{12} \\ b_{22} \\ b_{32} \end{bmatrix} \begin{bmatrix} c_{11} & c_{21} & c_{12} & c_{22} \end{bmatrix} \right\|_{F}$$

Denote the preceeding  $6 \times 4$  matrix by  $\mathscr{R}(A)$  and observe that

$$\mathscr{R}(A) = \begin{bmatrix} \operatorname{vec}(A_{11})^{\mathrm{T}} \\ \operatorname{vec}(A_{21})^{\mathrm{T}} \\ \operatorname{vec}(A_{31})^{\mathrm{T}} \\ \operatorname{vec}(A_{12})^{\mathrm{T}} \\ \operatorname{vec}(A_{22})^{\mathrm{T}} \\ \operatorname{vec}(A_{32})^{\mathrm{T}} \end{bmatrix}.$$

It follows that

$$\phi(B,C) = ||\mathscr{R}(A) - \operatorname{vec}(B)\operatorname{vec}(C)^{\mathrm{T}}||_{F}$$

and so the act of minimizing  $\phi$  is equivalent to finding a nearest rank-1 matrix to  $\mathscr{R}(A)$ . The nearest rank-1 matrix problem has a well-known SVD solution [29]. In particular, if

$$U^{\mathrm{T}}\mathscr{R}(A)V = \Sigma \tag{9}$$

is the SVD of  $\mathcal{R}(A)$ , then optimum B and C are defined by

$$\operatorname{vec}(B_{\operatorname{opt}}) = \sqrt{\sigma_1} U(:, 1) \quad \operatorname{vec}(C_{\operatorname{opt}}) = \sqrt{\sigma_1} V(:, 1).$$

The scalings are arbitrary. Indeed, if  $B_{opt}$  and  $C_{opt}$  solve the NKP problem and  $\alpha \neq 0$ , then  $\alpha \cdot B_{opt}$  and  $(1/\alpha) \cdot C_{opt}$  are also optimal.

In general, if  $A = (A_{ij})$  is an  $m_1 \times n_1$  block matrix with  $m_2 \times n_2$  blocks, then

$$\tilde{A} = \mathscr{R}(A) \in \mathbb{R}^{m_1 n_1 \times m_2 n_2} \Rightarrow \tilde{A}(i + (j-1)m_1, :) = \operatorname{vec}(A_{ij})^{\mathrm{T}}, \quad i = 1 : m_1, \ j = 1 : n_1.$$

If  $\mathscr{R}(A)$ ) has rank  $\tilde{r}$  and SVD (9), then

$$A = \sum_{k=1}^r \sigma_k U_k \otimes V_k$$

where  $vec(U_k) = U(:,k)$  and  $vec(V_k) = V(:,k)$  for  $k = 1 : \tilde{r}$ . We refer to this as the *Kronecker Product SVD* (KPSVD) of A associated with the integer factorizations  $m = m_1m_2$  and  $n = n_1n_2$ . Note that for these integers,

$$A_r = \sum_{k=1}^r \sigma_k U_k \otimes V_k \quad r \leqslant \tilde{r}$$
<sup>(10)</sup>

is the closest matrix to A (in the Frobenius norm) that is the sum of r Kronecker products.

If A is large and sparse and r is small, then the Lanzcos SVD iteration of Golub, Luk, and Overton [26] can effectively be used to compute the singular vectors of  $\Re(A)$  from which we can build the optimal Kronecker product factors. An implementation is available in [12] and some preliminary experience with the method is discussed in [73,56].

Certain situations permit one to "cut corners" in the above SVD computation when solving the NKP problem:

- If A is the sum of p Kronecker products as in the generalized Sylvester equation problem (3), then  $\operatorname{rank}(\mathscr{R}(A)) \leq p$ .
- If A is an  $n_1 \times n_1$  block Toeplitz matrix with  $n_2 \times n_2$  Toeplitz blocks, then it is not hard to show that the rank of  $\Re(A)$  is less than min $\{2n_1 + 1, 2n_2 + 1\}$  [51].

In each of these situations the matrix  $\mathcal{R}(A)$  is rank deficient.

#### 7. Other NKP problems

For  $A \in \mathbb{R}^{n \times n}$  with  $n = n_1 n_2$  we refer to the problem of minimizing

$$\psi(B,C) = ||A(B \otimes C) - I_n||_F \quad B \in \mathbb{R}^{n_1 \times n_1}, \ C \in \mathbb{R}^{n_2 \times n_2}$$
(11)

as the *inverse nearest Kronecker product problem*. This problem does not have an explicit SVD solution and so we must approach it as a structured nonlinear least squares problem. It can be shown that

$$\operatorname{vec}(A(B \otimes C)) = (I_n \otimes A)\operatorname{vec}(B \otimes C) = (I_n \otimes A)P(\operatorname{vec}(B) \otimes \operatorname{vec}(C))$$

where P is the  $n^2 \times n^2$  permutation matrix defined by

$$P = I_{n_1} \otimes S_{n_1,n_2} \otimes I_{n_2}$$

Thus, minimizing  $\psi$  is equivalent to minimizing the 2-norm of

$$F(B,C) = (I_n \otimes A)P(\operatorname{vec}(B) \otimes \operatorname{vec}(C)) - \operatorname{vec}(I_n).$$

The Jacobian of this function is  $(I_n \otimes A)P[(I_n \otimes \text{vec}(C)) \text{ vec}(B) \otimes I_n)]$ . Having exposed these structures we see that there are several ways to approach the inverse NKP problem. Since it is a separable least-squares problem, the variable projection methods discussed in [28,46] or the ideas in [3] are applicable.

The NKP problem has a multiple factor analog in which we try to approximate  $A \in \mathbb{R}^{m \times n}$  with a matrix of the form  $C_1 \otimes \cdots \otimes C_p$ . In particular, if  $m = m_1 \cdots m_p$  and  $n = n_1 \cdots n_p$ , then we seek  $C_i \in \mathbb{R}^{m_i \times n_i}$ , i = 1 : p so that  $\phi(C_1, \dots, C_p) = ||A - C_1 \otimes \cdots \otimes C_p||_F$  is minimized. Closed-form SVD solutions do not appear to be possible if p > 2. The inverse NKP problem also has a multiple factor generalization. If A is structured, then it is sometimes the case that the B and C matrices that solve the NKP problem are similarly structured. For example, if A is symmetric and positive definite, then the same can be said of  $B_{opt}$  and  $C_{opt}$  (if properly normalized). Likewise, if A is nonnegative, then the optimal B and C can be chosen to be nonnegative. These and other structured NKP problems are discussed in [73,56], but a number of interesting open questions remain.

Suppose  $A \in \mathbb{R}^{n \times n}$  and that  $n = m^2$ . A trace minimization problem that we are aware of requires the minimization of  $||A - \alpha B \otimes B||_F$  where  $B \in \mathbb{R}^{m \times m}$  and  $\alpha = \pm 1$ . This leads to a nearest symmetric rank-1 problem of the form

$$\min_{a,b} ||\mathscr{R}(A) - \alpha b b^{\mathrm{T}}||_{F}$$

A related problem arises in neural networks [38]. Given  $A \in \mathbb{R}^{n \times n}$  with  $n = m^2$ , find  $B \in \mathbb{R}^{m \times m}$  so that  $||A - B \otimes B^T||_F$  is minimized. This leads to the minimization of

$$||\mathscr{R}(A) - \operatorname{vec}(B)\operatorname{vec}(B^{\mathrm{T}})^{\mathrm{T}}||_{F} = ||\mathscr{R}(A)S_{m,m} - \operatorname{vec}(B)\operatorname{vec}(B)^{\mathrm{T}}||_{F}.$$

The linearly constrained NKP problem

$$\min_{\substack{F^{\mathrm{T}} \operatorname{vec}(B)=r\\G^{\mathrm{T}} \operatorname{vec}(C)=t}} ||A - B \otimes C||_{I}$$

leads to a linearly constrained nearest rank-1 problem

$$\min_{\substack{F^{\mathrm{T}}b=r\\G^{\mathrm{T}}c=t}} ||\tilde{A} - bc^{\mathrm{T}}||_{F}$$

where  $\tilde{A} = \Re(A)$ ,  $b = \operatorname{vec}(B)$  and  $c = \operatorname{vec}(C)$ . We suppress the dimensions of the matrices involved and just assume that the linear constraint equations are underdetermined. Following Golub [25] we compute the QR factorizations

$$F = Q_F \begin{bmatrix} R_F \\ 0 \end{bmatrix} \qquad G = Q_G \begin{bmatrix} R_G \\ 0 \end{bmatrix}$$

for then the problem transforms to

$$\min_{b_{2},c_{2}} \left\| \begin{bmatrix} \tilde{A}_{11} - b_{1}c_{1}^{\mathsf{T}} & \tilde{A}_{12} - b_{1}c_{2}^{\mathsf{T}} \\ \tilde{A}_{21} - b_{2}c_{1}^{\mathsf{T}} & \tilde{A}_{22} - b_{2}c_{2}^{\mathsf{T}} \end{bmatrix} \right\|_{F}$$

where

$$Q_F^{\mathrm{T}}b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \quad Q_G^{\mathrm{T}}c = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}, \quad Q_F^{\mathrm{T}}\tilde{A}Q_G = \begin{bmatrix} \tilde{A}_{11} & \tilde{A}_{12} \\ \tilde{A}_{21} & \tilde{A}_{22} \end{bmatrix}$$

and  $R_F^{T}b_1 = r$  and  $R_G^{T}c_1 = t$ . Thus, we are led to the minimization of the function

$$\mathbf{f}(b_2,c_2) = || ilde{A}_{22} - b_2 c_2^{\mathrm{T}}||_F^2 + || ilde{A}_{12} - b_1 c_2^{\mathrm{T}}||_F^2 + || ilde{A}_{21} - b_2 c_1^{\mathrm{T}}||_F^2$$

If r and t are zero, then  $b_1$  and  $c_1$  are zero and we are left with a reduced version of the nearest rank-1 matrix problem. This homogeneous situation arises when we wish to impose sparsity constraints on the B and C matrices or when we require the optimizing B and/or C to have circulant, Toeplitz, Hankel, or some other structure of that type. In the nonhomogeneous case we are again confronted with a bilinear least-square problem. (The inhomogeneous problem would arise, for example, if we required the B and C matrices to have columns that sum to one.)

# 8. Preconditioners

In recent years the "culture" of fast transforms and Kronecker products has found its way into the linear system solving area through the design of effective preconditioners. Chan [8] proposed solving large Toeplitz systems Tx = b using a circulant preconditioner *C* that minimizes  $||C - T||_F$ . The product of a circulant matrix and a vector can be carried out in  $O(n \log n)$  flops using the FFT; See also [9] and [10].

Similar in spirit is the design of Kronecker product preconditioners. The idea is to approximate the matrix of coefficients A with a Kronecker product  $B \otimes C$  and to use  $B \otimes C$  as a preconditioner noting that linear systems of the form  $(B \otimes C)z = r$  can be solved fast. One method for generating the Kronecker factors B and C is to minimize  $||A - B \otimes C||_F$ . Other approaches tailored to applications in image restoration have been offered by Nagy [51], Kamm and Nagy [43–45] and Thirumalai [68]; See also [20,52,7].

The success of many numerical methods hinge on efficient linear equation solving and this in turn often requires finding the "right" preconditioner for the coefficient matrix A. To be effective, the preconditioner M must "capture the essence" of A and have the property that systems of the form Mz = r are easily solved.

The idea of setting M to be the nearest Kronecker product to A is studied in [73]. When applied to a model problem (Poisson's equation on a rectangle) the results compared favorably with the best alternatives, e.g., the incomplete Cholesky preconditioner. This work can be extended by (a) looking at 3D problems where the resulting linear system is the sum of three Kronecker products and (b) considering non-uniform mesh settings where the linear system is the sum of a few "near" Kronecker products.

### 9. Conclusion

Our goal in this paper is to point to the widening use of the Kronecker product in numerical linear algebra. Research in this area will heighten the profile of the Kronecker product throughout the field of matrix computations and will make it easier for researchers to spot Kronecker "opportunities" in their work. This phenomena is not without precedent. The development of effective algorithms for the QR and SVD factorizations turned many " $A^TA$ " problems into least-square/singular-value calculations. Likewise, with the development of the QZ algorithm [50] engineers with "standard" eigenproblems of the form  $B^{-1}Ax = \lambda x$  came to approach them as a generalized eigenproblems of the form  $Ax = \lambda Bx$ . The point we are making is that if an infrastructure of effective Kronecker-product algorithms is built, then Kronecker product problems will "come out of the woodwork."

## References

- T.F. Andre, R.D. Nowak, B.D. Van Veen, Low rank estimation of higher order statistics, IEEE Trans. Signal Process. 45 (1997) 673–685.
- [2] H.C. Andrews, J. Kane, Kronecker matrices, computer implementation, and generalized spectra, J. Assoc. Comput. Mach. 17 (1970) 260–268.
- [3] R.H. Barham, W. Drane, An algorithm for least squares estimation of nonlinear parameters when some of the parameters are linear, Technometrics 14 (1972) 757–766.

- [4] A. Barrlund, Efficient solution of constrained least squares problems with Kronecker product structure, SIAM J. Matrix Anal. Appl. 19 (1998) 154–160.
- [5] R.H. Bartels, G.W. Stewart, Solution of the equation AX + XB = C, Comm. ACM 15 (1972) 820–826.
- [6] J.W. Brewer, Kronecker products and matrix calculus in system theory, IEEE Trans. Circuits Systems 25 (1978) 772–781.
- [7] D. Calvetti, L. Reichel, Application of ADI iterative methods to the image restoration of noisy images, SIAM J. Matrix Anal. Appl. 17 (1996) 165–174.
- [8] T.F. Chan, An optimal circulant preconditioner for Toeplitz systems, SIAM J. Sci. Statist. Comput. 9 (1988) 766–771.
- [9] T.F. Chan, J.A. Olkin, Preconditioners for Toeplitz-block matrices, Numer. Algorithms 6 (1993) 89-101.
- [10] R. Chan, X.-Q. Jin, A family of block preconditioners for block systems, SIAM J. Sci. Statist. Comput. 13 (1992) 1218–1235.
- [11] T.F. Coleman, Y. Li, A. Mariano, Separation of pulmonary nodule images using total variation minimization, Technical Report, Cornell Theory Center, Ithaca, New York, 1998.
- [12] J. Cullum, R.A. Willoughby, Lanczos Algorithms for Large Sparse Symmetric Eigenvalue Computations, Vol. I (Theory), II (Programs), Birkhauser, Boston, 1985.
- [13] K. Datta, The matrix equation XA BX = R and its applications, Linear Algebra Appl. 109 (1988) 91–105.
- [14] M. Davio, Kronecker products and shuffle algebra, IEEE Trans. Comput. c-30 (1981) 116–125.
- [15] P. Davis, P. Rabinowitz, Numerical Integration, Blaisdell, Waltham, MA, 1967.
- [16] C. de Boor, A Practical Guide to Splines, Springer, New York, 1978.
- [17] C. de Boor, Efficient computer manipulation of tensor products, ACM Trans. Math. Software 5 (1979) 173-182.
- [18] F.W. Dorr, The direct solution of the discrete poisson equation on a rectangle, SIAM Rev. 12 (1970) 248–263.
- [19] A. Dutt, V. Rokhlin, Fast fourier transforms for nonequispaced data, SIAM J. Sci. Comput. 14 (1993) 1368–1398.
- [20] L. Eldin, I. Skoglund, Algorithms for the regularization of Ill-conditioned least squares problems with tensor product structure and applications to space-variant image restoration, Technical Report LiTH-Mat-R-82-48, Department of Mathematics, Linkoping University, Sweden, 1982.
- [21] D.W. Fausett, C.T. Fulton, Large least squares problems involving Kronecker products, SIAM J. Matrix Anal. 15 (1994) 219–227.
- [22] D.W. Fausett, C.T. Fulton, H. Hashish, Improved parallel QR method for large least squares problems involving Kronecker products, J. Comput. Appl. Math. (1997).
- [23] A. Fijany, C.P. Williams, Quantum wavelet transforms: fast algorithms and complete circuits, Technical Report 9809004, Los Alamos National Laboratory, Los Alamos, New Mexico, 1998.
- [24] J. Gardiner, M.R. Wette, A.J. Laub, J.J. Amato, C.B. Moler, Algorithm 705: a FORTRAN-77 software package for solving the Sylvester matrix equation  $AXB^{T} + CXD^{T} = E$ , ACM Trans. Math. Software 18 (1992) 232–238.
- [25] G.H. Golub, Some modified eigenvalue problems, SIAM Rev. 15 (1973) 318-344.
- [26] G.H. Golub, F. Luk, M. Overton, A block Lanzcos method for computing the singular values and corresponding singular vectors of a matrix, ACM Trans. Math. Software 7 (1981) 149–169.
- [27] G.H. Golub, S. Nash, C. Van Loan, A Hessenberg-Schur method for the matrix problem AX + XB = C, IEEE Trans. Automat. Control AC-24 (1979) 909–913.
- [28] G.H. Golub, V. Pereya, The differentiation of pseudoinverses and nonlinear least squares problems whose variables separate, SIAM J. Numer. Anal. 10 (1973) 413–432.
- [29] G.H. Golub, C. Van Loan, Matrix Computations, 3rd Edition, Johns Hopkins University Press, Baltimore, MD, 1996.
- [30] A. Graham, Kronecker Products and Matrix Calculus with Applications, Ellis Horwood, Chichester, England, 1981.
- [31] J. Granata, M. Conner, R. Tolimieri, Recursive fast algorithms and the role of the tensor product, IEEE Trans. Signal Process. 40 (1992) 2921–2930.
- [32] J. Granata, M. Conner, R. Tolimieri, The tensor product: a mathematical programming language for FFTs and other fast DSP operations, IEEE SP Mag. (January 1992) 40–48.
- [33] L. Greengard, J. Strain, The fast Gauss transform, SIAM J. Sci. Statist. Comput. 12 (1991) 79-94.
- [34] S.R Heap, D.J. Lindler, Block iterative restoration of astronomical images with the massively parallel processor Proceedings of the First Aerospace Symposium on Massively Parallel Scientific Computation, 1986, pp. 99–109.
- [35] H.V. Henderson, F. Pukelsheim, S.R. Searle, On the history of the Kronecker product, Linear Multilinear Algebra 14 (1983) 113–120.

- [36] H.V. Henderson, S.R. Searle, The vec-permutation matrix, the vec operator and Kronecker products: a review, Linear Multilinear Algebra 9 (1981) 271–288.
- [37] R.A. Horn, C.A. Johnson, Topics in Matrix Analysis, Cambridge University Press, New York, 1991.
- [38] R.M. Hristev, private communication, 1998.
- [39] C-H. Huang, J.R. Johnson, R.W. Johnson, Multilinear algebra and parallel programming, J. Supercomput. 5 (1991) 189–217.
- [40] A.K. Jain, Fundamentals of Digital Image Processing, Prentice-Hall, Englewood Cliffs, NJ, 1989.
- [41] J. Johnson, R.W. Johnson, D. Rodriguez, R. Tolimieri, A methodology for designing, modifying, and implementing fourier transform algorithms on various architectures, Circuits Systems Signal Process. 9 (1990) 449–500.
- [42] B. Kagstrom, Perturbation analysis of the generalized Sylvester equation (AR LB, DR LE) = (C, F), SIAM J. Matrix Anal. Appl. 15 (1994) 1045–1060.
- [43] J. Kamm, J.G. Nagy, Kronecker product and SVD approximations in image restoration, Linear Algebra Appli. (1998a), to appear.
- [44] J. Kamm, J.G. Nagy, Kronecker product and SVD approximations for separable spatially variant blurs, SMU Mathematics Technical Report 98-04, Dallas, TX, 1998b.
- [45] J. Kamm, J.G. Nagy, Optimal kronecker product approximation of block Toeplitz matrices, SMU Mathematics Technical Report 98-05, Dallas, TX, 1998c.
- [46] L. Kaufman, A variable projection method for solving separable nonlinear least squares problems, BIT 15 (1975) 49–57.
- [47] B. Klaus, P. Horn, Robot Vision, MIT Press, Cambridge, MA, 1990.
- [48] B. Kumar, C.H. Huang, J. Johnson, R.W. Johnson, P. Sadayappan, A tensor product formulation of Strassen's matrix multiplication algorithm with memory reduction, Seventh International Parallel Processing Symposium, 1993, pp. 582–588.
- [49] P. Lancaster, Explicit solution of linear matrix equations, SIAM Rev. 12 (1970) 544-566.
- [50] C.B. Moler, G.W. Stewart, An algorithm for generalized matrix eigenvalue problems, SIAM J. Numer. Anal. 10 (1973) 241–256.
- [51] J.G. Nagy, Decomposition of block Toeplitz matrices into a sum of Kronecker products with applications in image processing, SMU Mathematics Technical Report 96-01, Dallas, TX, 1996.
- [52] J.G. Nagy, D.P. O'Leary, Restoring images degraded by spatially-variant blur, SIAM J. Sci. Comput. 19 (1998) 1063–1082.
- [53] R.D. Nowak, Penalized least squares estimation of higher order statistics, IEEE Trans. Signal Process. 46 (1998) 419–428.
- [54] R.D. Nowak, B. Van Veen, Tensor product basis approximations for Volterra filters, IEEE Trans Signal Process. 44 (1996) 36–50.
- [55] V. Pereyra, G. Scherer, Efficient computer manipulation of tensor products with applications to multidimensional approximation, Math. Comp. 27 (1973) 595–604.
- [56] N.P. Pitsianis, The Kronecker product in approximation, fast transform generation, Ph.D. Thesis, Department of Computer Science, Cornell University, 1997.
- [57] J.H. Pollard, On the use of the direct matrix product in analyzing certain stochastic population models, Biometrika 53 (1966) 397–415.
- [58] U.A. Rauhala, Introduction to array algebra, Photogrammetric Eng. Remote Sensing 46 (2) (1980) 177–182.
- [59] U.A. Rauhala, D. Davis, K. Baker, Automated DTM validation and progressive sampling algorithm of finite element array relaxation, Photogrammetric Eng. Remote Sensing 55 (1989) 449–465.
- [60] P.A. Regalia, S. Mitra, Kronecker products, unitary matrices, and signal processing applications, SIAM Rev. 31 (1989) 586–613.
- [61] J. Seberry, X-M. Zhang, Some orthogonal matrices constructed by strong Kronecker product multiplication, Austral. J. Combin. 7 (1993) 213–224.
- [62] R.A. Snay, Applicability of array algebra, Rev. Geophys. Space Phys. 16 (1978) 459-464.
- [63] W-H. Steeb, Matrix Calculus and Kronecker Product with Applications and C++ Programs, World Scientific Publishing, Singapore, 1997.
- [64] F. Stenger, Kronecker product extensions of linear operators, SIAM J. Numer. Anal. 5 (1968) 422-435.
- [65] L. Stiller, Multilinear algebra and chess endgames, in: Computational Games, Vol. 29, MSRI Publications.

- [66] T. Strohmer, Numerical algorithms for discrete gabor expansions, in: H.G. Feichtinger, T. Strohmer (Eds.), Gabor Analysis and Algorithms, Birkhauser, Basel, 1998, pp. 267–294.
- [67] A. Swami, J. Mendel, Time and lag recursive computation of cumulants from a state-space model, IEEE Trans. Automat. Control 35 (1990) 4–17.
- [68] S. Thirumalai, High performance algorithms to solve Toeplitz and block Toeplitz matrices, Ph.D. Thesis, University of Illinois, Urbana, IL, 1996.
- [69] M.J. Todd, On search directions in interior point methods for semidefinite programming, Optim. Methods Software, (1998), to appear.
- [70] R. Tolimieri, M. An, C. Lu, Algorithms for Discrete Fourier Transform and Convolution, Springer, New York, 1989.
- [71] L. Vandenberghe, S. Boyd, Semidefinite Programming, SIAM Rev. 38 (1996) 27-48.
- [72] C. Van Loan, Computational Frameworks for the Fast Fourier Transform, SIAM Publications, Philadelphia, PA, 1992.
- [73] C. Van Loan, N.P. Pitsianis, Approximation with Kronecker products, in: M.S. Moonen, G.H. Golub (Eds.), Linear Algebra for Large Scale and Real Time Applications, Kluwer Publications, Dordrecht, 1992, pp. 293–314.
- [74] W.J. Vetter, Vector structures, solutions of linear matrix equations, Linear Algebra Appl. 10 (1975) 181–188.



Journal of Computational and Applied Mathematics 123 (2000) 101-115

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.nl/locate/cam

# Preconditioning eigenvalues and some comparison of solvers

Ronald B. Morgan

Mathematics Department, Baylor University, Waco, TX 76798-7328, USA

Received 30 June 1999; received in revised form 17 December 1999

#### Abstract

Preconditioning techniques are discussed for symmetric eigenvalue problems. The methods Davidson, Jacobi–Davidson, Rayleigh quotient iteration, and preconditioned Lanczos are considered. Some relationships are given between these different approaches, and some experimental comparisons are done. Jacobi–Davidson appears to be efficient in both expense and storage. A hybrid method may be helpful for the case of a poor initial vector. © 2000 Elsevier Science B.V. All rights reserved.

MSC: 65F15; 15A18

Keywords: Eigenvalues; Preconditioning; Davidson's method; Jacobi-Davidson

## 1. Introduction

Finding eigenvalues is an important task in scientific computation. There are many applications in physics, chemistry, and engineering. These include computing energy levels of atoms, finding vibrational states of molecules, and determining how buildings will vibrate during earthquakes. Frequently scientists wish to know some eigenvalues of very large matrices. For such problems, Krylov subspace methods are well known. The Lanczos algorithm [9,22] is a Krylov subspace method for symmetric problems. For nonsymmetric matrices, the methods are Arnoldi [1,26,35] and nonsymmetric Lanczos [9,26].

For large systems of linear equations, preconditioning is an important technique for improving the spectrum. While it is not as straightforward, preconditioning can also be used for eigenvalue problems. Methods that use preconditioning are no longer strictly Krylov methods, although they are generally still related. In cases where an effective, inexpensive preconditioner is available, preconditioning can significantly improve the convergence and provide a better method.

E-mail address: ronald\_morgan@baylor.edu (R.B. Morgan).

<sup>0377-0427/00</sup>/\$ - see front matter © 2000 Elsevier Science B.V. All rights reserved. PII: S0377-0427(00)00395-2

Preconditioning of eigenvalue problems is only partly developed. However, work has been done for some time in Russia; see [8] for a summary and references. Also, Ruhe [23–25] used SOR and the conjugate gradient method to find eigenvalues in the 1970s. Around the same time, quantum chemists, including Davidson [4], developed correction methods for their large symmetric matrices. In 1986, in [19], Davidson's method was viewed as a diagonal preconditioning method, and it was generalized for arbitrary preconditioners. For more on Davidson's method, see [18,16,3,37,38]. The Jacobi–Davidson method [31] was developed in 1996. The inexact rational Krylov method [10] and truncated RQ [36] are recent approaches that are related to Jacobi–Davidson.

This paper discusses preconditioning methods and does some fairly simple comparisons. The focus is on the methods more than on the preconditioners (not much has been done on effectiveness of various preconditioners for eigenvalue problems, but see [33,21,32]). We consider only a few methods and only the symmetric case. The methods are the generalized Davidson (GD) method [19], preconditioned Lanczos (PL) [20], the Rayleigh quotient iteration (RQI) with preconditioned conjugate gradient solution of the linear equations [11,39], and Jacobi–Davidson (JD) [31] with also the preconditioned conjugate gradient method. New implementations of PL and RQI are also given: vectors in the outer loops are saved and the Rayleigh–Ritz procedure [22] is applied.

One motivation for this paper is to see how the recent JD method compares with the others. Also, we wanted to see if the robustness of PL could be improved with the addition of the outer Rayleigh–Ritz; if it could compete better with GD, in terms of total number of iterations. And finally, the more robust implementation of RQI is given to make the comparisons with it more fair. It is desired that the discussion and the comparisons in this paper provide some insights into these methods. However, further comparisons of preconditioning methods are definitely needed.

Section 2 describes the methods that will be considered and gives the new modifications. Section 3 has discussion, and Section 4 has experiments.

# 2. Description of methods

We consider the eigenvalue problem  $Az = \lambda z$ , where A is a symmetric matrix. However, the algorithms listed below all have nonsymmetric versions. Instead of PL, preconditioned Arnoldi [13] can be used. JD and RQI just need a nonsymmetric iterative linear equations solver [27,28,7,40,2] in the inner loop.

The GD method generates a subspace with the preconditioned operator  $M^{-1}(A - \theta I)$ , where  $\theta$  is an approximate eigenvalue and M is an approximation to  $A - \theta I$ . It uses the Rayleigh-Ritz procedure to extract approximate eigenvectors from the subspace. We quickly describe the Rayleigh-Ritz procedure [22,26]. Let V be an orthonormal matrix with columns spanning the desired subspace. The Rayleigh-Ritz procedure finds eigenpairs  $(\theta, s)$  of the small matrix  $V^{T}AV$ . The  $\theta$ 's are approximate eigenvalues, called Ritz values. The approximate eigenvectors or Ritz vectors are y = Vs.

## Algorithm 1. Generalized Davidson's method

- Begin with k orthonormal starting vectors  $v_1, v_2, \ldots, v_k$ .
- For j = k, k + 1, ... do

- 1. Apply the Rayleigh–Ritz procedure to the subspace  $\text{Span}\{v_1, v_2, \dots, v_j\}$ . Let the best approximation to the eigenpair of interest be  $(\theta, y)$ , with y normalized. Choose a preconditioner M which may either be fixed or may depend on  $\theta$ .
- 2. Find the residual vector for y,  $r = (A \theta I)y$ . If  $||r|| \leq \text{TOL}$ , accept that eigenpair, otherwise continue.
- 3. Compute  $w_i = M^{-1}r$ . Orthonormalize  $w_i$  against  $v_1, \ldots, v_i$  to form  $v_{i+1}$ .

In Davidson's original method,  $M = D - \theta I$ , where D is the diagonal matrix with the same main diagonal as A.

When *j* becomes too large, the Rayleigh–Ritz expense can be prohibitive. Then Davidson's method needs to be restarted. For instance, in step 1 we can add: If j = j max, pick the *k* best approximate eigenvectors, orthonormalize them to give  $v_1, \ldots, v_k$ , and let j = k.

The PL method takes GD's operator  $M^{-1}(A - \theta I)$ , but requires M to be a positive definite preconditioner. Then  $M^{-1}(A - \theta I)$  can be symmetrized and the Lanczos algorithm applied. There is an outer loop that updates the approximate eigenpair.

#### Algorithm 2. Preconditioned Lanczos

- Choose a starting vector  $y_0$ . Let  $\theta_0 = y_0^T A y_0 / y_0^T y_0$ .
- For  $j = 0, 1, 2, \dots$  do
  - 1. Choose a SPD preconditioner  $M_j$  for  $A \theta_j I$ , and factor as  $M_j = L_j L_j^{\mathrm{T}}$ .
  - 2. Apply the Lanczos method to  $W_j = L_j^{-1}(A \theta_j I)L_j^{-T}$  with initial vector  $L_j y_j$  and with stopping criterion  $rn < -v_j$ , where  $v_j$  is the smallest Ritz value of  $W_j$  and rn is the associated residual norm. When the Lanczos loop has ended, let  $w_j$  be the normalized Ritz vector that corresponds to  $v_j$ .
  - 3. Compute  $y_{j+1} = L^{-T}w_j$ , which is an approximate eigenvector of A, and its Rayleigh quotient  $\theta_{j+1} = \theta_j + v_j/y_{j+1}^T y_{j+1}$ .
  - 4. Find the residual vector for  $y_{j+1}$ ,  $r = (A \theta_{j+1}I)y_{j+1}/||y_{j+1}||$ . If  $||r|| \leq \text{TOL}$ , accept that eigenpair, otherwise continue.

When several eigenvalues are being sought, an approximation to the next one is calculated by finding the Ritz vector corresponding to the second Ritz value of  $W_j$ , and then multiplying it by  $L^{-T}$  (this will be the new  $y_0$  vector after the current eigenpair is accepted). This is done when ||r|| is first less than TOL<sup>2/3</sup>. While computing the second and subsequent eigenvalues, the ones already determined are shifted out of the way. For the *l*th eigenvalue, replace  $A - \theta_j I$  in step 2 with  $A - \theta_j I + \gamma z_1 z_1^T + \cdots + \gamma z_{l-1} z_{l-1}^T$ , for  $\gamma$  a value such that  $\lambda_1 + \gamma$  is moved beyond the eigenvalues of interest.

Some expense can be saved by not checking the convergence of the inner Lanczos loop at every step. Also, a test can be used to terminate the Lanczos loop early when convergence is near [20]. When the  $M_j$  inner product is used in the Lanczos loop, there is no need to factor  $M_j$  [12]. The Krylov basis is then  $M_j$  orthogonal. This is similar to how the preconditioned conjugate gradient method is implemented.

As was suggested in the conclusion of [20], we now modify PL in order to make it more robust. The outer loop now applies a small Rayleigh–Ritz procedure to the last few approximate eigenvectors that have been developed.

Algorithm 3. Preconditioned Lanczos with an outside Rayleigh-Ritz projection

- Same as Preconditioned Lanczos except for:
  - 3. Compute  $x=L^{-T}w_j$  and put it in a set of x vectors. If a new  $y_0$  vector is computed (see the discussion in the paragraph after the PL algorithm), immediately add it also to the set of x vectors. Apply the Rayleigh–Ritz procedure with matrix A to the subspace spanned by the x vectors. Let  $(\theta_{i+1}, y_{i+1})$  be the smallest Ritz pair from this outside Rayleigh–Ritz.
  - 4. Find the residual vector for  $y_{j+1}$ ,  $r = (A \theta_{j+1}I)y_{j+1}/||y_{j+1}||$ . If  $||r|| \leq \text{TOL}$ , accept that eigenpair and let the new  $y_0$  be the second approximate eigenvector from the Rayleigh-Ritz procedure in 3, otherwise continue.

We abbreviate this method as PL-RR.

Similar to the discussion after the GD algorithm, once the subspace of x's becomes too large, it can be restarted, retaining the best approximate eigenvectors. The same is true for the Rayleigh–Ritz procedures in the RQI and JD algorithms that are given next. We modify the simple RQI algorithm by adding the outside Rayleigh–Ritz.

## Algorithm 4. Rayleigh quotient iteration

- Same as Generalized Davidson except for:
  - 3. Solve  $(A \theta I)w_j = y$ , using a stable conjugate gradient method such as SYMMLQ, preconditioned by a SPD matrix M.

For the stopping test in SYMMLQ, we use improvement in residual norm of  $\min\{0.01, ||r||^2\}$ . For JD, a different system of linear equations is solved.

# Algorithm 5. Jacobi-Davidson

- Same as Generalized Davidson except for:
  - 3. Solve  $(I yy^T)(A \theta I)(I yy^T)w_j = r$ , using a preconditioned conjugate gradient method. In the preconditioning step, CG uses the inverse of  $(I - yy^T)M(I - yy^T)$ .

See [31,33,29] for JD implementation details, including how to avoid most applications of  $(I - yy^{T})$ . We use regular preconditioned CG instead of SYMMLQ, because it is stable in our tests. The CG loop uses stopping test of relative residual norm less than  $2^{-j}$  [29]. As in PL, a test is used to terminate the inner loop early when convergence is near. Specifically, the  $2^{-j}$  test is replaced by min(0.5,  $0.5 \times 10^{\log_{10}(TOL) - \log_{10}(||r||)})$  if this quantity is larger than  $2^{-j}$ .

For computing the second and subsequent eigenvalues, the previous ones can be deflated by using  $(I - yy^{T})(I - QQ^{T})(A - \theta I)(I - QQ^{T})(I - yy^{T})$ , where Q is the orthonormal matrix whose columns are the converged eigenvectors, in place of  $(I - yy^{T})(A - \theta I)(I - yy^{T})$  [29]. This is used in some of our experiments.

# 3. Discussion of methods

3.1. The operator  $M^{-1}(A - \theta I)$ 

All of the preconditioning methods discussed here use essentially the same operator in their inner loop. This operator is  $M^{-1}(A - \theta I)$ , with  $\theta$  an approximate eigenvalue. GD has this operator with  $\theta$


Fig. 1. Spectrum without preconditioning from Example 3.1.

changing every iteration. With PL, the symmetric version of this operator is used during the Lanczos run, and  $\theta$  is fixed for each run. RQI has the matrix  $M^{-1}(A - \theta I)$  in its inner loop of preconditioned conjugate gradients. The same is true for JD, except that the current approximate eigenvector is deflated out of this operator.

Since these methods are all similar in their core operators, we might expect that they would give similar results. This is not necessarily the case. However, it is safe to say that they all share the same limitation. They can only be as effective as the preconditioned operator  $M^{-1}(A - \theta I)$  allows them to be. For the preconditioning to be worthwhile, the spectrum of this operator needs to be a significant improvement over that of the original operator A. Let  $(\lambda, z)$  be the desired eigenpair. If  $\theta$  is approximately equal to  $\lambda$ , the important operator is  $M^{-1}(A - \lambda I)$ . This operator has eigenvalue 0 with eigenvector z. So it has the correct eigenvector, and a major question is whether the eigenvalue 0 is better separated from the rest of the spectrum than  $\lambda$  was in the spectrum of A.

We give a couple of examples of how the spectrum of A can be changed by the preconditioning. The matrices are nonsymmetric, even though the focus of this paper is on symmetric problems. This makes the examples more general, and also the spectral plots are more interesting with complex eigenvalues. The first example shows that eigenvalue preconditioning can be effective. Then a case is given where preconditioning does not work well.

**Example 3.1.** The first matrix has diagonal elements 1, 2, 3, ..., 100, and all other entries distributed normally with mean 0 and standard deviation 1. The smallest eigenvalue of *A* is 0.2787. We look at the spectrum of *A* and the spectrum of  $N = (D - \theta I)^{-1}(A - \theta I)$ , where *D* is the diagonal matrix with the main diagonal of *A*. Also  $\theta = 0.28$ , which is accurate to two decimal places. See Figs. 1 and 2 for the plots. It is clear that the preconditioned spectrum of *N* is an improvement over that of *A*. The



Fig. 2. Spectrum with preconditioning from Example 3.1.

desired eigenvalue of N has much better separation relative to the entire spectrum. We give the ratio of the distance to the closest eigenvalue with the distance to the farthest eigenvalue. A smaller ratio can mean the eigenvalue is more difficult to find, although there other factors such as the positioning of the eigenvalues and the degree of nonnormality. The ratio is 0.04 for the smallest eigenvalue of A compared to 0.28 for the eigenvalue of N near zero. This is a significant improvement.

**Example 3.2.** The second matrix is the same as the first except the diagonal elements are also random. Unlike in Example 3.1, the diagonal of the matrix is not a good approximation to the entire matrix. So we use a larger portion of the matrix as preconditioner. We let P be a band matrix with 49 diagonals of A, so  $p_{ij} = a_{ij}$  if |i - j| < 25 and otherwise  $p_{ij} = 0$ . In Figs. 3 and 4, we give the spectrums of A and of  $N = (P - \theta I)^{-1}(A - \theta I)$ . Here  $\theta = -11.16$  is an approximation to the leftmost eigenvalue of A, -11.1633. There is very little improvement in the relative separation. For both A and N, the ratio of distances to closest and farthest eigenvalues is about 0.16. It is interesting that in this example, the inaccuracy of  $\theta$  is magnified and N's smallest eigenvalue is not very near to zero. We can get some improvement by using a preconditioner with 99 diagonals of A. The ratio of distances to closest and farthest eigenvalues becomes 0.27, but this is with 75% of the entries of the matrix used in the preconditioner.

The results in Example 3.2 are not really surprising. It is already known for linear equations that preconditioning a random matrix is difficult. Some structure is needed for effective preconditioning. It is interesting that some theoretical results can be established for preconditioning eigenvalues, just as they can for linear equations. See [17,12] for results about convergence with modified incomplete factorization versus incomplete factorization for matrices from Poisson's equation, and for



Fig. 3. Random matrix from Example 3.2, no preconditioning.



Fig. 4. Random matrix, 49 diagonals preconditioning.

supporting experiments that show the advantage of modified incomplete factorization as the problem size increases.

Next, the differences between the methods are discussed. First, we compare GD with the other approaches, then discuss JD, compare RQI and JD, and finally give relationships and comparisons for JD and PL.

# 3.2. The GD method

GD changes its approximate eigenvalue at every iteration. This is an advantage in that it always uses the best information available. Based on this, it seems that GD will converge in less iterations. Another advantage for GD is that an SPD preconditioner is not required. However, it has the disadvantage of requiring more overhead expenses. Unlike the other methods, it cannot use an efficient Lanczos or CG algorithm. So it may require more CPU time. Restarting reduces the overhead expense but can also slow convergence. For difficult problems that need large subspaces, restarted GD can even end up needing more iterations than other methods. In cases where it is expensive to apply the matrix–vector product and the preconditioner, GD's overhead is not so important, and it may be the best method.

# 3.3. Jacobi–Davidson

The ideal JD operator in the outer loop is  $((I - yy^T)(A - \theta I)(I - yy^T))^{-1}(I - yy^T)(A - \theta I)(I - yy^T)$ . We discuss what the deflation of y accomplishes. Without deflation, this ideal operator become the identity matrix and is useless. Normally, an approximation to the inverse is used, so the deflation may not be needed. However, it is reassuring to know that the method does not break down if an approximate inverse is too good. And there is a case where we can expect to have a good approximation. That is when the application of  $(A - \theta I)^{-1}$  is accomplished by solving linear equations with an iterative method. Of course, this is the approach of JD.

# 3.4. RQI and JD

The versions of RQI and JD that we used here are similar in that they both have a variant of the preconditioned conjugate gradient method in their inner iteration. However, JD has an advantage which we discuss quickly here. For more, see [10], where the shift-and-invert transform is compared to the Cayley transform. Let  $(\lambda, z)$  be the desired eigenpair, and suppose that  $\theta$  is converging to  $\lambda$ . Assume that  $(A - \theta I)^{-1}$  is being approximated. Then this approximation does not necessarily have an eigenvector converging to z, unless the accuracy increases as  $\theta$  becomes more accurate. This greater accuracy corresponds to solving the linear equations in RQI to an increasing degree of accuracy (it may theoretically be possible to have an eigenvector converging to z without increasing the accuracy, with a different choice of starting vectors for the linear solver). Meanwhile, the approximation to  $((I - yy^T)(A - \theta I)(I - yy^T)(A - \theta I)(I - yy^T)(A - \theta I)(I - yy^T))^{-1}$  does not improve. So in the JD method, the degree of accuracy needed in the solution of the linear equations is not so crucial.

JD also has the small eigenvalue removed from the conjugate gradient loop. This may improve convergence and stability compared to RQI.

# 3.5. PL and JD

We now consider PL and JD including how they relate to each other. The PL inner loop is inaccurate because of the effect of the preconditioning (for  $\theta$  not yet equal to an eigenvalue,  $M^{-1}(A - \theta I)$  does not have the same eigenvector as A). This necessitates restarting the loop and the restarting slows down the convergence. It does appear that the stopping test in the Lanczos loop is effective and minimizes the loss. Even with the early termination of the inner loop, the convergence is asymptotically quadratic with respect to the outer loop. The quadratic convergence indicates that the inner loop iteration can run longer as the method proceeds. So the efficiency compared to GD increases as it goes along.

Now with JD, the inner loop solves linear equations instead of an eigenvalue problem. So some effort potentially could be wasted if it goes into improving the linear equation solution in a way that does not eventually benefit the eigenvalue problem. And as with PL, some efficiency may be lost due to the restarts of the inner iteration. However, JD can give asymptotic cubic convergence with proper residual tolerances [6]. So when it is near convergence, its inner loop can be solved longer and still get a benefit.

There actually is a close equivalence between the problems solved in the inner loops of PL and JD. It is well known for an SPD matrix that these two tasks are approximately equivalent: (1) solving linear equations with the conjugate gradient method, (2) using the Lanczos algorithm to compute an eigenvalue added to the spectrum at zero (see, for example, [30]). They are equivalent in the sense that the same polynomial will be effective for both problems. This polynomial needs to be large at zero and small over the spectrum of A (the CG polynomial also needs to be normalized to one at zero). The comparison between PL and JD is somewhat similar, but instead of computing an eigenvalue added to the spectrum, an eigenvalue is removed from the linear equations spectrum. We look at the asymptotic case with  $\theta$  equal to the eigenvalue  $\lambda$ , with eigenvector z. For PL, we need a polynomial large at the zero eigenvalue of  $M^{-1}(A - \lambda I)$  and small at the other eigenvalues. For JD, the operator in the inner loop is  $((I - zz^T)M(I - zz^T))^{-1}(I - zz^T)(A - \lambda I)(I - zz^T)$ . This operator has the same spectrum as in the PL inner loop. The linear equations problem has zero removed from the spectrum, because of the deflation in this operator and the fact that the right-hand side for the linear equations is orthogonal to y (and asymptotically to z). So we need a polynomial that is one at zero and small over the rest of the spectrum of the JD inner loop operator. This is equivalent to the polynomial for PL. So asymptotically, the two methods solve problems of similar difficulty in their inner loops.

It is argued in [20] that PL is likely to converge to the smallest eigenvalue, although initial convergence can be very slow if there is neither a good starting vector nor an initial estimate for the smallest eigenvalue. Meanwhile, JD can get hung up and converge to the wrong eigenvalue. The inner loop of PL spreads out the spectrum of  $M^{-1}(A - \lambda I)$  and can compute a number of approximate eigenvectors (although only one or two are desired and generally only one is an accurate approximation to an eigenvector of A), while JD goes after just one vector in solving its linear equations. This one vector actually may improve approximations to several eigenpairs. Nevertheless, it seems that PL may have an advantage initially. See Example 4.3. On the other hand, in this

situation of a difficult start, some other method might be considered initially. A hybrid approach could begin with application of the Lanczos algorithm with either operator A or  $M^{-1}$ .

A major advantage for JD is in storage. PL must save all of the Lanczos vectors generated in its inner loop in order to form the Lanczos Ritz vector. But since JD has the conjugate gradient method in its inner loop, the storage demands are minimal. And the number of vectors saved for the outer Rayleigh–Ritz procedure can be kept small if necessary.

# 4. Experiments

In these tests, the number of matrix-vector products (mvps) is listed. An application of the preconditioner accompanies each mvp. CPU time on a Vax is also given. However, the algorithms are not necessarily implemented optimally, so limited attention should be paid to the timings. For instance, the small eigenvalue problems in all methods are solved with Eispack routines [34]. So the solution could perhaps be more efficient, particularly for GD which solves similar problems at every iteration. The Lanczos algorithm does not use partial reothogonalization. For the Rayleigh–Ritz procedures in all methods, basis vectors are reorthogonalized when the vector's norm drops by 90% during the orthogonalization.

The GD method is restarted when subspaces reach dimension 20. JD has subspaces of maximum size 10 in its outer loop. GD generally needs larger subspaces than JD, because it builds its subspace with a weaker preconditioner than JD uses for its outer subspace. PL–RR restarts the outer Rayleigh–Ritz after an eigenvector is computed. To reduce the expense, just two eigenpairs are computed in the inner Lanczos loop of the PL methods. Also, the convergence in the inner loop is checked for the first seven Lanczos steps and then only at multiples of five.

**Example 4.1.** For testing, we choose a matrix A of dimension 5000. It is tridiagonal with entries  $1, 2, 3, \ldots, 5000$  on the main diagonal and with 0.5's in all the superdiagonal and subdiagonal positions. The starting vector has elements chosen randomly from the interval (-1, 1). Two preconditioners of different accuracy are used. Both are fixed diagonal matrices. First  $M = \text{Diag}(1.1, 1.2, 1.3, \ldots, 500.9, 501)$  is a good preconditioner, then  $M = \text{Diag}(1.002, 1.004, 1.006, \ldots, 10.998, 11)$  is a mediocre preconditioner. It is implicitly assumed in these preconditioners that we have 0.0 as an estimate for the desired eigenvalues. So we are using M = D - 0.0I instead of  $D - \theta I$ . Both PL methods use  $\gamma = 100$  to shift already converged eigenvectors [20]. We let JD use the initial estimate of 0.0 in place of  $\theta$  for the first few outer iterations, because the poor starting vector gives a bad  $\theta$ . This is switched back once  $\theta$  improves (distance from 0.0 less than the eigenvalue residual norm). For this example, deflation of converged eigenvectors is not used in JD, because it does not significantly improve convergence.

The smallest eigenvalue and eigenvector are found first. Then the smallest five eigenvalues are computed. Tables 1 and 2 give the results with the two preconditioners. We see that GD converges in the fewest iterations, but costs more.

There appears to be some inherent loss in going from GD to the double iteration of PL, at least for the good preconditioner case. PL–RR is not a big improvement upon PL. However, for the case of five eigenvalues with the mediocre preconditioner, PL–RR does come close to GD in mvps.

	1 eigenvalue		5 eigenvalues	
	mvps	cpu time	mvps	cpu time
GD	26	1.2	87	4.9
PL	41	0.40	183	2.2
PL-RR	41	0.41	190	2.8
RQI	36	0.46	308	3.6
JD	38	0.47	150	2.6

Table 1 Test matrix, good preconditioner

Table 2 Test matrix, mediocre preconditioner

	1 eigenvalue		5 eigenvalues	
	mvps	cpu time	mvps	cpu time
GD	164	8.6	641	36.2
PL	207	2.1	812	9.6
PL-RR	213	1.8	698	8.0
RQI	237	2.1	1428	11.4
JD	230	1.6	869	8.2

RQI is the slowest method if several eigenvalues are computed. Performance of JD and PL is fairly similar. However, as mentioned earlier, JD uses less storage than PL. When using the poorer preconditioner, PL builds subspaces as large as 85 in the inner Lanczos loop. For tougher problems, even larger subspaces would be needed.

To show that GD can be the best in terms of time, we do an experiment with a similar, but less sparse matrix: 200 elements of value 0.1 are added to each row. We do not give a table, but do compare GD with PL for the good preconditioner. GD takes less time to compute 5 eigenvalues, 13.4 versus 21.8 s.

**Example 4.2.** In this example we demonstrate that the implementation does matter for a preconditioned method. Simply having the operator  $M^{-1}(A - \theta I)$  does not guarantee effectiveness. RQI had some difficulty in the previous example when computing several eigenvalues, but we seek a more dramatic example. One of the simplest preconditioning methods for eigenproblems is to compute  $M^{-1}(A - \theta I)y$  as in Davidson's method, but use it as a correction to y instead of applying Rayleigh–Ritz (see [8] for some other simple preconditioning methods). So this simple iteration starts with y and computes a new approximate eigenvector  $y - M^{-1}(A - \theta I)y$ , where  $\theta$  is the Rayleigh quotient of y. This new approximation then becomes y. This method is tested on the matrix from Example 1 with the good preconditioner. The starting vector has first two components 100 and -50 and the rest random on (-1, 1). The true eigenvector is  $(0.91, -0.41, 0.095, -0.015, 0.002, ...)^{T}$ . The method did not converge. This is not surprising, since divergence can be expected if any eigenvalues of  $I - M^{-1}(A - \lambda_1 I)$  are greater than one in magnitude.

	(Random)	(10, random)	(100,-50, random)
	mvps	mvps	mvps
PL-RR	250	149	76
JD	-	_	39

Table 3 Test matrix, good prec., different starting vectors

Next, we make the method a little more complicated. After  $M^{-1}(A - \theta I)y$  is computed, a 2 × 2 Rayleigh–Ritz procedure is applied to combine y and  $M^{-1}(A - \theta I)y$ . So this is equivalent to GD with maximum size subspace of two. This approach did give convergence, even with the random starting vector from Example 4.1. With the good preconditioner, the first eigenvalue was computed with 85 mvps and 1.1 cpu s. This compares to 26 mvps and 1.2 s for GD with subspaces of dimension 20. With the mediocre preconditioner, there is a much bigger difference. The 2 × 2 method uses 3663 mvps and 44 s compared to 164 mvps and 8.6 s for GD (and only 1.6 s for JD). This is similar to large Krylov subspaces being better than either power method or dimension two Krylov subspaces. And large Krylov subspaces are especially important for tough problems.

**Example 4.3.** We compare PL–RR to JD with the assumption that there is neither an eigenvalue estimate available nor an extremely accurate eigenvector approximation. Three starting vectors are used. The first has all entries random on (-1,1). The second has first component 10 and the others random. The third has first entries 100 and -50, as mentioned in the previous example. Even though this last vector is better, its Rayleigh quotient is 293.3, not near the desired eigenvalue. The PL inner loop is limited to 20 steps, but a better stopping test is needed for the Lanczos loop in this situation of a poor starting vector. JD does not use 0.0 in place of  $\theta$  initially, as in the earlier tests. The good preconditioner from Example 4.1 is used, except it is shifted by  $\theta$ , and absolute values are taken of negative elements. The results are given in Table 3. JD does not converge for the poorer vectors. It gets hung up searching for wrong eigenvalues. The same thing can happen to GD and RQI. PL separates out the eigenvectors in its inner loop and does not focus so much on the eigenvector nearest the current  $\theta$  (see the discussion in Section 3.5).

**Example 4.4.** We next look at a situation where JD works better than PL. A bad preconditioner is chosen. It actually has a detrimental effect. PL approximately computes an eigenvector of  $M^{-1}(A - \theta I)$ . So if that eigenvector is distorted away from the eigenvector of A by the preconditioner, the method may have trouble until  $\theta$  is very near  $\lambda$ . For JD, poor preconditioning may slow the solution of the linear equations, but does not change the vector that the method is trying to compute. To elaborate on this, we note that if both PL and JD solve their inner loops to full accuracy, JD is not affected by the distortion of the preconditioner. PL is affected.

We let the preconditioner be  $M = LL^{T}$ , where L is lower bidiagonal with 0.95's on the main diagonal and 1's on the subdiagonal. The matrix is the same as in the earlier examples, but because of the bad preconditioning, we use n = 200 instead of 5000. All inner loops are limited to 200 iterations. The starting vector is all random. To find the smallest eigenvalue, JD uses 1164 matrix–vector products. PL requires 3436. The bad preconditioning does have a stronger effect on PL.

	1 eigenvalue		5 eigenvalues	
	mvps	cpu time	mvps	cpu time
GD	42	0.7	182	2.4
PL	66	0.4	467	1.5
PL-RR	66	0.4	365	1.5
RQI	99	0.5	616	2.1
JD	52	0.4	351	1.6

Table 4 Sherman1 matrix, inc. fact. preconditioner

Table 5

Sherman1 matrix, diagonal preconditioner

	1 eigenvalue		5 eigenvalues	
	mvps	cpu time	mvps	cpu time
GD	528	5.3	2080	20.4
PL	473	1.2	2073	6.4
PL-RR	432	1.3	1537	4.7
RQI	614	1.7	3107	7.0
JD	522	1.4	1541	4.2

**Example 4.5.** Next, we run tests similar to those in Example 4.1, but with the matrix Sherman1 from the Harwell–Boeing collection [5]. The matrix is dimension 1000. The smallest eight eigenvalues are 0.00032, 0.00102, 0.00111, 0.00151, 0.00192, 0.00205, 0.00209 and 0.00210, and the largest is 5.04. So some of the small eigenvalues are close together. A good preconditioner is incomplete factorization [14] of *A* with no fill-in, and a mediocre preconditioner is diagonal preconditioning. Note that with incomplete factorization, only one factorization is done, and the matrix is not shifted before the factoring. So the estimated value 0.0 is assumed to be known for the smallest eigenvalues. JD again uses that estimate initially in place of  $\theta$ . Here JD does deflate out converged eigenvectors. The PL methods use  $\gamma = 0.01$  to remove converged eigenvectors. The results are given in Tables 4 and 5. Note that PL–RR and JD use similar numbers of matrix–vector products. Both require less iterations than GD for finding five eigenvalues with diagonal preconditioning. PL–RR performs significantly better than PL, so the outside Rayleigh–Ritz seems worthwhile. The RQI algorithm here works much better in this situation of close eigenvalues than the algorithm used in [20] (some eigenvalues were skipped). The outside Rayleigh–Ritz loop makes a big difference.

# 5. Conclusion

We have discussed that preconditioning methods are generally related by their use of the same operator,  $M^{-1}(A - \theta I)$ . So convergence rates are often similar, but the implementation can make a difference.

We give some conclusions about the methods compared in this paper. GD is expensive compared to the other methods when matrix-vector products are cheap. JD and RQI require the least memory.

PL-RR is likely to use the most memory. JD and PL-RR usually have similar performance for the examples discussed here. However, PL-RR is more sensitive to a bad preconditioner than JD. And JD is less robust than PL-RR when the starting vector is poor.

We cannot recommend just one method as being best. However, JD is reliable and efficient if the starting vector is good. And an initial subspace can be computed for JD with another method. PL–RR is a possible initial method, but perhaps better would be a Lanczos method using operator A or  $M^{-1}$ , since PL is not immune to initial problems [15, pp. 99–101]. We also note that GD is likely best if the matrix–vector product is quite expensive.

While this paper deals with the symmetric case, much of the discussion carries over for nonsymmetric problems. It would be interesting to continue the comparisons for that case, and for interior eigenvalue problems and generalized eigenvalue problems.

### Acknowledgements

The author wishes to thank Henk van der Vorst for inviting this paper. Also thanks to Henk and Zhaojun Bai for their work on the Eigenvalue Templates project which helped lead to this paper. The anonymous referees understood the methods in this paper very well. They made many suggestions for improving the paper.

#### References

- W.E. Arnoldi, The principle of minimized iterations in the solution of the matrix eigenvalue problem, Quart. Appl. Math. 9 (1951) 17–29.
- [2] R. Barrett, M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H. van der Vorst, Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, SIAM, Philadelphia, PA, 1994.
- [3] M. Crouzeix, B. Philippe, M. Sadkane, The Davidson method, SIAM J. Sci. Comput. 15 (1994) 62-76.
- [4] E.R. Davidson, The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices, J. Comput. Phys. 17 (1975) 87–94.
- [5] I.S. Duff, R.G. Grimes, J.G. Lewis, Sparse matrix test problems, ACM Trans. Math. Software 15 (1989) 1-14.
- [6] S.C. Eisenstat, H.F. Walker, Choosing the forcing terms in an inexact Newton method, SIAM J. Sci. Comput. 17 (1996) 16–32.
- [7] R.W. Freund, N.M. Nachtigal, QMR: a Quasi-minimal residual method for non-Hermitian linear systems, Numer. Math. 60 (1991) 315–339.
- [8] A. Knyazev, Preconditioning eigensolvers, in: Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide, SIAM, 2000, to appear.
- [9] C. Lanczos, An iterative method for the solution of the eigenvalue problem of linear differential and integral operators, J. Res. Nat. Bur. Standards 45 (1950) 255–282.
- [10] R.B. Lehoucq, K. Meerbergen, Using generalized Cayley transformations within an inexact rational Krylov sequence method, SIAM J. Matrix Anal. Appl. 20 (1998) 131–148.
- [11] J.G. Lewis, Algorithms for Sparse Matrix Eigenvalue Problems, Ph.D. Thesis, Stanford University, Stanford, CA, 1977.
- [12] K. Meerbergen, R.B. Morgan, Inexact methods, in: Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide. SIAM, 2000, to appear.
- [13] K. Meerbergen, D. Roose, The restarted Arnoldi method applied to iterative linear system solvers for the computation of rightmost eigenvalues, SIAM J. Matrix Anal. Appl. 18 (1997) 1–20.

- [14] J.A. Meijerink, H.A. van der Vorst, An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix, Math. Comput. 31 (1977) 148–162.
- [15] R.B. Morgan, Preconditioning Eigenvalue Problems, Ph.D. Thesis, University of Texas, Austin, TX, 1986.
- [16] R.B. Morgan, Davidson's method and preconditioning for generalized eigenvalue problems, J. Comput. Phys. 89 (1990) 241–245.
- [17] R.B. Morgan, Theory for preconditioning eigenvalue problems, Proceedings of the Copper Mountain Conference on Iterative Methods, 1990.
- [18] R.B. Morgan, Generalizations of Davidson's method for computing eigenvalues of large nonsymmetric matrices, J. Comput. Phys. 101 (1992) 287–291.
- [19] R.B. Morgan, D.S. Scott, Generalizations of Davidson's method for computing eigenvalues of sparse symmetric matrices, SIAM J. Sci. Statist. Comput. 7 (1986) 817–825.
- [20] R.B. Morgan, D.S. Scott, Preconditioning the Lanczos algorithm for sparse symmetric eigenvalue problems, SIAM J. Sci. Comput. 14 (1993) 585–593.
- [21] R.B. Morgan, M. Zeng, Harmonic projection methods for large non-symmetric eigenvalue problems, Numer. Linear. Algebra. Appl. 5 (1998) 33–55.
- [22] B.N. Parlett, The Symmetric Eigenvalue Problem, Prentice-Hall, Englewood Cliffs, NJ, 1980.
- [23] A. Ruhe, SOR-methods for the eigenvalue problem with large sparse matrices, Math. Comput. 28 (1974) 695–710.
- [24] A. Ruhe, Iterative eigenvalue algorithms based on convergent splittings, J. Comput. Phys. 19 (1975) 110-120.
- [25] A. Ruhe, T. Wiberg, The method of conjugate gradients used in inverse iteration, BIT 12 (1972) 543-554.
- [26] Y. Saad, Numerical Methods for Large Eigenvalue Problems, Halsted Press, New York, NY, 1992.
- [27] Y. Saad, Iterative Methods for Sparse Linear Systems, PWS Publishing, Boston, MA, 1996.
- [28] Y. Saad, M.H. Schultz, GMRES: a generalized minimum residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Statist. Comput. 7 (1986) 856–869.
- [29] G. Sleijpen, H. van der Vorst, Jacobi–Davidson methods, in: Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide, SIAM, 2000, to appear.
- [30] G.L.G. Sleijpen, A. van der Sluis, Further results on the convergence behavior of conjugate-gradients and Ritz values, Linear Algebra Appl. 246 (1996) 233–278.
- [31] G.L.G. Sleijpen, H.A. van der Vorst, A Jacobi–Davidson iteration method for linear eigenvalue problems, SIAM J. Matrix Anal. Appl. 17 (1996) 401–425.
- [32] G.L.G. Sleijpen, H.A. van der Vorst, Preconditioning for eigenvalue correction equations. Technical Report in preparation, Utrecht University, Utrecht, The Netherlands, 1999.
- [33] G.L.G. Sleijpen, H.A. van der Vorst, E. Meijerink, Efficient expansion of subspaces in the Jacobi–Davidson method for standard and generalized eigenvalue problems, ETNA 7 (1998) 75–89.
- [34] B.T. Smith, J.M. Boyle, Y. Ikebe, B.C. Klema, C.B. Moler, Matrix Eigensystems Routines: EISPACK Guide, Springer, New York, NY, 1970.
- [35] D.C. Sorensen, Implicit application of polynomial filters in a k-step Arnoldi method, SIAM J. Matrix Anal. Appl. 13 (1992) 357–385.
- [36] D.C. Sorensen, C. Yang, A truncated RQ iteration for large scale eigenvalue calculations, SIAM J. Matrix Anal. Appl. 19 (1998) 1045–1073.
- [37] A. Stathopoulos, Y. Saad, C. Fischer, Robust preconditioning of large, sparse, symmetric eigenvalue problems, J. Comput. Appl. Math. 64 (1995) 197–215.
- [38] A. Stathopoulos, Y. Saad, K. Wu, Dynamic thick restarting of the Davidson, and the implicitly restarted Arnoldi methods, SIAM J. Sci. Comput. 19 (1998) 227–245.
- [39] D.B. Szyld, Criteria for combining inverse and Rayleigh quotient iteration, SIAM J. Numer. Anal. 6 (1988) 1369– 1375.
- [40] H.A. van der Vorst, Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems, SIAM J. Sci. Statist. Comput. 12 (1992) 631–644.



JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

Journal of Computational and Applied Mathematics 123 (2000) 117-130

www.elsevier.nl/locate/cam

# For tridiagonals T replace T with $LDL^{t}$

Beresford N. Parlett

Mathematics Department and Computer Science Division, EECS Department, University of California, Berkeley, CA 94720, USA

Received 21 July 1999; received in revised form 10 November 1999

#### Abstract

The same number of parameters determine a tridiagonal matrix T and its triangular factors L, D and U. The mapping  $T \rightarrow LDU$  is not well defined for all tridiagonals but, in finite precision arithmetic, L, D and U determine the entries of T to more than working precision. For the solution of linear equations  $LDU\mathbf{x} = \mathbf{b}$  the advantages of factorization are clear. Recent work has shown that LDU is also preferable for the eigenproblem, particularly in the symmetric case. This essay describes two of the ideas needed to compute eigenvectors that are orthogonal without recourse to the Gram–Schmidt procedure when some of the eigenvalues are tightly clustered. In the symmetric case we must replace T, or a translate of T, by its triangular factors  $LDL^{t}$ . © 2000 Elsevier Science B.V. All rights reserved.

## 1. Introduction and representations

This essay needs justification because it examines a problem that has been considered as solved for several decades: the symmetric eigenproblem for dense matrices. Section 2 describes the methods that have been used with satisfaction for a long time. However in 1995 there was a little trouble in Paradise. A team of computational chemists at Pacific Northwest Laboratories found that certain problems of order 1000–2000 were taking much longer than expected using the best available software. On investigation it turned out that in a three-stage process the middle part, which should have been negligible, was consuming 80% of the time. Further probing showed that 95% of the eigenvalues were judged by the program to be in a tight cluster (they all agreed to four or more decimals) and so the Gram–Schmidt orthonormalizing process was invoked to make sure that the computed eigenvectors were indeed orthonormal to working accuracy. Because the cluster was so large what is normally an  $O(n^2)$  process, for  $n \times n$  matrices, turned into an  $O(n^3)$  marathon, see [11,8]. This incident provoked some interesting lines of thought. The conservative view would cite the inherent limitations of working in fixed precision arithmetic and would argue that very difficult

E-mail address: parlett@math.berkeley.edu (B.N. Parlett).

calculations should take more effort. This conservative view could be amplified and made quite persuasive. One central fact is that when the matrix elements are known to working precision, say 8 or 16 decimal places, then the eigenvalues and eigenvectors inherit a level of uncertainty that sets a limit on how accurately they can be computed. Indeed the closer some eigenvalues cluster together the less well determined are their eigenvectors. In the limit, for a multiple eigenvalue, it is only the eigenspace that is defined, there is no distinguished basis of eigenvectors. Consequently, extra measures, such as the Gram–Schmidt process, must be invoked to ensure that the program returns orthogonal eigenvectors for tight clusters of eigenvalues.

A different reaction to the 1995 revelation is to wonder whether there is a way to wriggle out of these difficult situations and to attain the following ambitious goal: given a  $n \times n$  real symmetric tridiagonal matrix T compute its eigenvalues and then send each eigenvalue, with a copy of T, to its own processor. Each processor computes its eigenvector, all at the same time, and the outputs turn out to be orthogonal to working precision without the need to check. That would be nice!

When the eigenvalues are nearly uniformly spaced in the spectrum then current methods can realize the goal. What might we do when several eigenvalues agree to 4 or 8 or 12 decimals?

There is a method, developed by Dhillon and me from 1996 to 1999, and software to implement it, but several new ideas are needed to justify the whole procedure and only one or two themes will be described in this essay. Section 4 shows the method in action on a  $4 \times 4$  example. Before launching into more detail it is helpful to recall two key facts. First, eigenvectors are invariant under translation (or shifting)  $T \rightarrow T - \sigma I$ . Second, there is no loss in assuming that the next-to-diagonal entries (i, i+1) and (i+1, i) do not vanish,  $i=1,2,\ldots,n-1$ . In that case the true eigenvalues are distinct and the eigenvectors are well defined even though some eigenvalues may be equal to working precision. This is a subtle property of the tridiagonal form. Thus, there is a basis of eigenvectors even when some eigenvalues appear multiple to working precision. We can aim to compute extremely accurate eigenvectors and then orthogonality would follow automatically since the 'true' eigenvectors have this property.

We now describe the first of the new themes. The radical new goal is to compute an approximate eigenvector  $\mathbf{x}$ ,  $||\mathbf{x}|| = 1$ , for a given approximate eigenvalue  $\hat{\lambda}$  with the *relative residual* property

$$||T\mathbf{x} - \mathbf{x}\hat{\lambda}|| = O(n\varepsilon)|\hat{\lambda}|, \quad \text{not just } O(n\varepsilon||T||), \tag{1}$$

where  $\varepsilon$  is the roundoff unit and we regard two normalized vectors **u** and **v** as orthogonal to working precision if

$$|\boldsymbol{u}^{t}\boldsymbol{v}| = \mathcal{O}(n\varepsilon). \tag{2}$$

We use big O notation to hide some modest constant between 1 and 100. Unfortunately (1) is not achievable for the simple reason that  $\lambda$  is not always defined to high relative accuracy by *T*. Here  $\lambda$  is the eigenvalue of *T* closest to  $\hat{\lambda}$ . This means that small relative uncertainty in *T*'s entries may cause large relative uncertainties in tiny eigenvalues. A simple but mild example of this phenomenon is a Toeplitz matrix  $a + b(N + N^t)$  where *N* is the  $n \times n$  Jordan block with eigenvalue 0. For n = 4,

$$N = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

When b=1, a=-2 we obtain the second difference matrix whose eigenvalues lie in (-4,0). The one closest 0 is  $-4\sin^2(\pi/4n) \approx -(\pi/2n)^2$ . Take  $n=10^3$  and change a from -2 to  $-2(1-10^{-8})$  to find that  $\lambda_{\min}$  changes to  $\lambda_{\min}(1-\frac{4}{5}10^{-2}+O(10^{-4}))$ , a large relative change. The example is mild because the eigenvalues are not severely clustered. More complicated examples show this phenomenon of large relative changes in the smallest eigenvalues for small values of n, say n = 4, see [7] and Section 4.

In order to achieve (1) it is not only necessary that each small eigenvalue, such as  $\lambda$ , be determined to high relative accuracy (by *T*) but we must find an algorithm that will approximate  $\lambda$  by  $\hat{\lambda}$  to high relative accuracy. If, for example,  $|\lambda| = 10n\varepsilon$  then the residual norm in (1) must achieve the very small value  $O(n^2\varepsilon^2)$ .

Although there are special classes of tridiagonals that do define their small eigenvalues to high relative accuracy, see [2], the property fails most of the time.

In conclusion (1) seems to be an unattainable goal. At this low point we must emphasize, briefly, why (1) is so desirable. There is a well-known error bound that is, in addition, a fairly realistic estimate of the error (angle) between the x achieving (1) and the true eigenvector s for  $\lambda$ , the eigenvalue closest to  $\hat{\lambda}$ . This result is not restricted to tridiagonals, see [18] for a proof.

**Theorem 1.** Let T be real symmetric,  $T\mathbf{s} = \mathbf{s}\lambda$ , where  $\lambda$  is the eigenvalue closest to  $\hat{\lambda}$ . For any  $\mathbf{x}$ ,  $||\mathbf{x}|| = 1$ , and any  $\hat{\lambda}$ ,

$$\begin{aligned} \sin| \angle (\mathbf{x}, \mathbf{s}) &| \leq \frac{||T\mathbf{x} - \mathbf{x}\hat{\lambda}||}{\operatorname{gap}(\hat{\lambda})}, \\ \operatorname{gap}(\hat{\lambda}) &= |\mu - \hat{\lambda}|, \ \mu \text{ is the eigenvalue } (\neq \lambda) \text{ closest to } \hat{\lambda}. \end{aligned}$$

If (1) holds then the theorem assures us that

$$\sin|\angle(\mathbf{x},\mathbf{s})| \leq \frac{\mathcal{O}(n\varepsilon)|\hat{\lambda}|}{|\mu - \hat{\lambda}|} \approx \frac{\mathcal{O}(n\varepsilon)}{\operatorname{relgap}(\hat{\lambda})},$$

where

relgap 
$$(\lambda) = \frac{|\lambda - \mu|}{|\lambda|}$$

If (1) holds then

$$rac{|\hat{\lambda}-\mu|}{|\hat{\lambda}|}pprox rac{|\lambda-\mu|}{|\lambda|}.$$

For example, if ||T|| = 1,  $\lambda = 10^{-18}$  and  $\mu = 10^{-19}$  then relgap ( $\lambda$ ) = 0.9 and x is a very accurate approximation to s. For more details see [16].

The message here is that if (1) can be achieved then very accurate eigenvectors can be produced for all eigenvalues with large *relative* gaps. The next link in the chain of ideas is the simple observation that *relative* gaps may be increased by a suitable shift of origin whereas absolute separation between eigenvalues is invariant since  $|(\lambda - \sigma) - (\mu + \sigma)| = |\lambda - \mu|$ .

Now we return to (1). The big new idea is to get rid of T! To convey the idea in a simple way consider the case when T is positive definite and so T permits a Cholesky decomposition

$$T = LL^{t}, \tag{3}$$

where L is lower bidiagonal. In 1967 Kahan proved that all the eigenvalues of  $LL^t$  are determined to high relative accuracy by the entries of L, not those of T. Today there are easy proofs of this result, see [9,10], and there is more than one way to compute the eigenvalues to this high accuracy. Of itself this result does not guarantee (1) but it is an essential element. That is the first theme. A method we shall not describe here, see [15], permits (1) to be achieved with  $LL^T$  replacing T.

Now we turn to a different theme. As suggested by the earlier remarks on relative gaps in the spectrum it may be necessary to shift the origin and use triangular factorization

$$T - \tau I = LL^{t} - \tau I = L^{(1)}D_{+}L^{(1)^{t}},$$
(4)

where  $L^{(1)}$  is a unit lower bidiagonal matrix containing the multipliers and  $D_+$  is a diagonal matrix holding the 'pivots'. There is no general statement on how well the entries of  $L^{(1)}$  and  $D_+$  determine the eigenvalues of  $L^{(1)}D_+L^{(1)^t}$  but the results in [17] show that for most values of  $\tau$  these factors  $L^{(1)}$  and  $D_+$  do give high relative accuracy for the small eigenvalues. There is nothing sacred in factoring from top to bottom. We can use as well a factorization from bottom to top:

$$T - \tau I = LL^{t} - \tau I = U^{(1)} D_{-} U^{(1)^{t}},$$
(5)

where  $U^{(1)}$  is a unit upper bidiagonal matrix and  $D_{-}$  is diagonal. In fact, there is a whole family of n twisted factorization of  $T - \tau I$  and they all use the same number of parameters, namely 2n - 1, see [17].

The implication of the preceding remarks is that we can compute very accurate eigenvectors if we can find representations, such as  $L^{(1)}D_+L^{(1)^t}$ , that define their small eigenvalues to high relative accuracy. Recall that each shift changes the eigenvalues. However, one new representation will not (usually) suffice. We will need several representations, such as in (4), for different values of  $\tau$ . We will compute a subset of eigenvectors for each representation. This raises a new difficulty. When we change from one representation to another, say  $\hat{L} D \hat{L}^t = LDL^t - \tau I$ , we wonder whether the inevitable roundoff errors in computing  $\hat{L}$  and  $\hat{D}$  from L, D, and  $\tau$  will break the link between the eigenvectors computed from L and D to those computed from  $\hat{L}$  and  $\hat{D}$ . Fortunately, the recently discovered differential stationary qd algorithms, see [12], give a way to switch between representations and preserve high relative accuracy. That is the second theme and extends to nonsymmetric tridiagonals and is the topic of Section 3. Section 4 shows the new method in action on a difficult  $4 \times 4$  example and Section 5 shows what extends to the nonsymmetric case and what still needs to be done.

# 2. The classical methods

As soon as digital computers became available to scientists around 1950 the search was begun for eigenvalue algorithms that were robust when executed in finite precision arithmetic. In 1954, very early in the game, Wallace Givens came up with a method for a symmetric matrix A that has stood with little change for over 40 years. The defects of using the characteristic polynomial were quickly grasped. No one would like to meet a polynomial of degree 1000 on a dark night. It is extremely

volatile and prone to blow up under the slightest provocation. A promising alternative is to employ explicit similarity transformations until A turns into  $\Lambda$  diagonal. In principle, an infinite sequence of similarities is needed to reach  $\Lambda$  and that brings on tricky questions of when to stop.

Givens proposed a compromise between the two approaches (explicit similarities and the characteristic polynomial) given above. The method has three distinct stages.

*Phase* 1: Reduce A to tridiagonal T by a *finite sequence* of plane rotations designed to eliminate one nontridiagonal entry at a time and preserve all previously created zero entries. Thus,

$$T = G_s^* \cdots G_1^* A G_1 \cdots G_s = F^* A F \tag{6}$$

where

$$s = \binom{n-2}{2}$$

and G alters only two columns. The G's are accumulated into F and this phase costs  $O(n^3)$  operations.

*Phase* 2: Apply 'bisection' to any given interval to find all, or some eigenvalues of T to full precision (relative to ||T||) or less. The tool is Sylvester's Inertia theorem applied to Gaussian elimination without interchanges. Let  $T - \tau I = LDL^t$ . Sylvester's Inertia theorem says the number of eigenvalues less than  $\tau$  equals the number of negative entries on D's diagonal. Once an interval contains a single eigenvalue bisection may be continued until a designated number of correct digits is obtained. The cost of each factorization is 2n operations, and so the cost of computing k eigenvalues is O(kn).

In his original technical report Givens did not invoke Sylvester's Inertia theorem nor triangular factorization. Instead he used a more complicated mechanism with a three-term recurrence and Sturm sequences but the two approaches are equivalent in exact arithmetic but Givens had to worry about over/underflow.

In order to compute the eigenvector belonging to a computed eigenvalue  $\hat{\lambda}$  Givens solved  $(T - \hat{\lambda}I)\mathbf{x} = \mathbf{e}_n$ , where  $\mathbf{e}_j$  is the *j*th column of the identity matrix *I*. This was the least successful feature of his method. Any fixed right-hand side will lead to trouble on some matrices. We now know that it is important to choose the right-hand side carefully, see [15] for more details. Again the cost for  $\mathbf{x}$  is O(n) so Phase 2 is an O(kn) process for k eigenpairs. As indicated in Section 1 numerical orthogonality depends on the separation of the eigenvalues and a Gram–Schmidt post-processing has to be available.

*Phase* 3: Let  $T = SAS^t$ . If the eigenvectors Z of A are wanted then S is mapped into Z via Z = FS. This is an  $O(n^3)$  process. F need not be found explicitly but can be represented by the sequence of G's given in (6). If only k eigenvectors are wanted the cost reduces to  $O(kn^2)$ .

Finally, Givens produced an error analysis in fixed-point arithmetic showing that the computed eigenvalues were the exact eigenvalues of a matrix close to T or A. This was one of the earliest instances of a 'backward' error analysis: the computed quantities solve exactly a nearby problem. It is worth emphasizing that a backward error analysis is not possible for all algorithms.

There is little to add for the task of computing a subset of eigenpairs. There is an alternative to Phase 2 when all eigenvalues are wanted. The QR algorithm, see [13,5], is applied to T yielding  $A = R_p^* \cdots R_1^* TR_1 \cdots R_p$  where each  $R_i$  is a plane rotation G and p is the number of iterations used in the QR algorithm. Then  $S = R_1 \cdots R_p$  and this accumulation of plane rotations produces an S that is

orthogonal to working accuracy however close the eigenvalues may be. The price for this desirable property is an  $O(n^3)$  algorithm for the spectral factorization of *T*. Since Phases 1 and 3 are  $O(n^3)$  what is wrong with having Phase 2 also  $O(n^3)$ ? Answer: the constant behind O is too big.

# 3. Changing representations

In this section we consider tridiagonals that are not necessarily symmetric. Instead of  $T - \sigma I = LDL^{t}$  we will have  $T - \tau I = LU$ . We normalize our matrices in a way that would destroy symmetry. Any tridiagonal with nonzero off-diagonal entries is said to be *unreduced*. Any unreduced tridiagonal is diagonally similar to one with all super-diagonal entries equal to 1;  $\Delta T \Delta^{-1} = J$ . We designate such matrices by J and note that the number of free parameters in J is 2n - 1 for  $n \times n$  cases. If J permits triangular factorization, J = LU, then we write

$$L = \operatorname{bidiag} \begin{pmatrix} 1 & 1 & 1 & . & 1 & 1 \\ e_1 & e_2 & . & . & e_{n-1} \end{pmatrix},$$
$$U = \operatorname{bidiag} \begin{pmatrix} 1 & 1 & . & 1 & 1 \\ q_1 & q_2 & . & . & q_{n-1} & q_n \end{pmatrix}.$$

An attractive feature of this notation is that UL is also a J-matrix and that feature is exploited later in the section.

Section 1 emphasized the advantages of exploiting the shift invariance of eigenvectors. Suppose that L and U determine well the eigenvalues of J. When we need a new representation, for  $J - \sigma I$  say, we must compute  $\overset{\circ}{L}$  and  $\overset{\circ}{U}$  satisfying

$$\overset{\circ}{L}\overset{\circ}{U}=J-\sigma I=LU-\sigma I$$

There are (at least) two ways to compute  $\overset{\circ}{L}$  and  $\overset{\circ}{U}$  from L, U, and  $\sigma$ . The first is called the *stationary* qd-algorithm *stqds* by Rutishauser, see [22].

The algorithm can be derived by equating entries on each side of  $\overset{\circ}{L} \overset{\circ}{U} = LU - \sigma I$  in the appropriate order.

stqds(
$$\sigma$$
):  $\hat{q}_1 = q_1 - \sigma$   
for  $i = 1, n - 1$  do  
 $\hat{e}_i = e_i q_i / \hat{q}_i$   
 $\hat{q}_{i+1} = e_i + q_{i+1} - \sigma - \hat{e}_i$   
end for

Unfortunately, when executed in finite precision arithmetic, this algorithm is not accurate enough to connect one representation to the other by making small *relative* changes to the parameters q and e. There is more discussion of this point later in this section. Fortunately, there is an alternative implementation. It is easy to miss and Rutishauser never published it and seems to have discovered it only in the last two years of his life. The alternative was found independently of Rutishauser by Fernando and Parlett as recently as 1991 and in another context, see [12]. It is called the *differential* 

stationary qd algorithm and so the old name is prefixed with a little d.

dstqds(
$$\sigma$$
):  
 $s_1 = -\sigma$   
for  $i = 1, n - 1$  do  
 $\stackrel{\circ}{q_i} = s_i + q_i$   
 $\stackrel{\circ}{e_i} = e_i q_i / \stackrel{\circ}{q_i}$   
 $s_{i+1} = s_i q_i / \stackrel{\circ}{q_i} - \sigma$   
end for  
 $\stackrel{\circ}{q_n} = s_n + q_n$ 

The auxiliary variable is called  $s_i$  and the new value  $s_{i+1}$  may be written over the old  $s_i$ . The essential property of the new algorithm is that it enjoys *mixed relative stability*. What does this mean?

Let  $\overset{\circ}{L}$  and  $\overset{\circ}{U}$  now denote the bidiagonal matrices actually computed by dstqds( $\sigma$ ) in the computer. Then there exist special tiny end-figure changes to the entries of L, U,  $\overset{\circ}{L}$ ,  $\overset{\circ}{U}$  giving new matrices  $\overline{L}$ ,  $\overline{U}$ ,  $\overline{L}$ ,  $\overline{U}$ ,  $\overline{L}$ ,  $\overline{U}$ , respectively, such that

$$\tilde{L}\tilde{U} = \bar{L}\bar{U} - \sigma I$$

*exactly.* The necessary change in most of the entries is two units (bits) in the last place held (i.e. in the last digit) and none exceeds four. Thus, the eigenvectors of  $\tilde{L}\tilde{U}$  are identical to those of  $\bar{L}\bar{U}$  and we only have to make sure that  $\hat{L}$ ,  $\hat{U}$  determine the (small) eigenvalues of  $\hat{L}\tilde{U}$  together with the associated eigenvectors to high relative accuracy. In addition L, U must also determine those same eigenvectors to high relative accuracy. Symmetry is not essential.

It should be mentioned that when the original matrix is symmetric then there is a minor variation of dstqds that uses, not L and U but L and D where  $LDL^t$  is the matrix in question. The same stability results hold with minor variations in the details, see [7,16]. That relative stability property of dstqds is the second ingredient in the method for computing accurate eigenvectors. It permits us to relate the eigenvectors computed from different representations to the eigenvectors of one single matrix  $LL^t$  or  $LDL^t$ .

There is an essential component of the new method that has not been discussed so far. How can one calculate the eigenvalues of  $LL^t$  or  $LDL^t$  to high relative accuracy? In the symmetric case there is a variant of the well-known bisection algorithm, see [5], that may be used. This technique is effective for refining eigenvalues already known to good accuracy but is slow as a general tool. There is a much faster method, discovered in 1991/1992 by Fernando and Parlett, see [12], that computes the eigenvalues of  $LL^t$  using the ideas in this section.

If J = LU then the LR transform of J is  $\hat{J} = UL$ . When shifts are incorporated we obtain

LR(
$$\tau$$
): factor  $J - \tau I = LU$ ,  
form  $\hat{J} = UL + \tau I$ .

Thus  $\hat{J} = L^{-1}(J - \tau I)L + \tau I = L^{-1}JL$ .

The LR algorithm consists of iterating the LR transform with well chosen shifts. It was presented by Rutishauser in 1957, see [21], along with a proof of its surprising convergence property. For simplicity take all shifts to be zero. Then if J has eigenvalues with distinct absolute values and if the factorization does not fail then, very slowly, the q-values tend to the eigenvalues in monotonic decreasing order, i.e.,  $q_n$  tends to the eigenvalue closest to 0. The e-values tend to zero.

Rutishauser implemented the LR transform so that  $\hat{J}$  overwrote J with no explicit reference to L and U. Today the LR algorithm is remembered, if at all, as the algorithm that led to the celebrated QR algorithm which is a little slower than LR but never breaks down and is backward stable.

With our preference for L, U over J we let  $\hat{J} = \hat{L}\hat{U}$  and want to compute  $\hat{L}$  and  $\hat{U}$ , without reference to J from  $\hat{L}\hat{U} = UL - \tau I$ . This may be accomplished by the qds algorithm that was discovered by Rutishauser in 1953/54, see [20], some years before he saw that qds was equivalent to LR in exact arithmetic.

qds(
$$\tau$$
):  $\hat{q}_1 = q_1 + e_1 - \tau$   
for  $i = 1, n - 1$  do  
 $\hat{e}_i = e_i * q_{i+1}/\hat{q}_i$   
 $\hat{q}_{i+1} = q_{i+1} + e_{i+1} - \tau - \hat{e}_i$   
end for

This is not the most accurate implementation. There is a *differential* form of  $qds(\tau)$  that was discovered by Fernando and Parlett as late as 1991, see [12].

dqds(
$$\tau$$
):  
 $p_1 = q_1 - \tau$   
for  $i = 1, n - 1$  do  
 $\hat{q}_i = p_i + e_i$   
 $\hat{e}_i = e_i * q_{i+1}/\hat{q}_i$   
 $p_{i+1} = p_i * q_{i+1}/\hat{q}_i - \tau$   
end for  
 $\hat{q}_n = p_n$ 

An examination of both algorithms shows that the shift is not restored;  $\hat{J} = \hat{L}\hat{U} = UL - \tau I$ . Thus all eigenvalues have been reduced by  $\tau$ . This feature has advantages although it is troublesome to people familiar with the QR algorithm. It becomes necessary to keep a running sum  $\sigma$  of all shifts used in order to recover the original eigenvalues. In practice, with dqds, the program checks when  $e_{n-1}$  and  $q_n$  are both negligible and then records  $\sigma$  as an eigenvalue and reduces the order *n* by 1.

The advantage of dqds over qds is that it enjoys high mixed relative stability even in the presence of element growth. Small end-figure changes to the input L, U and to the output  $\hat{L}$ ,  $\hat{U}$  give an exact transformation and this feature permits all the eigenvalues to be computed to high relative accuracy in the positive case. When the original J comes from a positive definite symmetric matrix T, via  $J = \Delta T \Delta^{-1}$ , then all q's and e's will be positive. The shifts must be chosen carefully and details can be found in [12,19]. The latest implementation is in LAPACK, see [1], and is almost as fast as the root free QR method that computes eigenvalues with errors  $O(\varepsilon ||T||)$ , not  $O(\varepsilon |\lambda|)$ .

Apart from its use in computing eigenvalues the  $dqds(\sigma)$  transform plays a role in computing 'twisted' factorizations of tridiagonals that are needed in the course of computing eigenvectors. We omit this material and refer the reader to [15,16].

#### 4. A small example

The ideas sketched in Section 1 may be illustrated on a  $4 \times 4$  matrix. This matrix is similar to one used by Dhillon in his dissertation [7]. It is contrived to produce an intermediate representation  $(LDL^t)$  that looks bad because it suffers severe element growth,  $||L|| = 10^7 ||T||$ . Nevertheless the representation is good enough for its special purpose.

Let  $\varepsilon$  denote the roundoff unit for Matlab (=2×10<sup>-16</sup>) and let  $\eta$ := $\sqrt{\varepsilon}$ . The tridiagonal T is given by

diagonal =  $(1 + \eta, 1 - 2\eta, 1 + 3\eta, 1 + 2\eta)$ ,

off-diagonal =  $(\sqrt{2}/2, \sqrt{2}/2, \eta)$ .

The eigenvalues are, approximately,

$$-\varepsilon, 1+\frac{4}{3}\eta, 1+\frac{8}{3}\eta, 2+\varepsilon,$$

Off-diag	Diag	Eigenvalues
7.071067811865476e - 01	1.000000014901161e + 00	2.00000000000001e + 00
7.071067811865476e - 01	9.999999701976776e – 01	1.00000040339034e + 00
1.490116119384766e - 08	1.000000044703484e + 00	1.00000019265610e + 00
0	1.000000029802322e + 00	-6.890205972143757e - 16

Matlab has no trouble computing an orthonormal set of eigenvectors because it uses the QR algorithm. We ignore the extreme eigenvalues and focus on the pair close to 1 whose separation is  $\frac{4}{3}\eta = O(\sqrt{\epsilon})$ .

First we performed standard inverse iteration using a good starting vector. Each computed vector  $\mathbf{x}$  has an excellent residual norm;  $||T\mathbf{x} - \mathbf{x}\lambda|| = O(\varepsilon)$ . The dot product between them is

$$O(\sqrt{\varepsilon}) = \frac{O(\varepsilon)}{gap} = \frac{O(\varepsilon)}{4/3\eta}$$

as expected by standard theory, see Section 1. This is not good enough.

Next, we pursued a simple variation of inverse iteration. Since our two eigenvalues agree to eight decimals we may translate T to T - I and find that the shifted eigenvalues have no digits in common. We try inverse iteration again, using T - I, and find an improvement. The dot product is  $10^{-10}$  instead of  $10^{-8}$ . This is not good enough. The calculations are shown in Fig 1.

Before proceeding to the central idea of new representations we discuss the starting vector for inverse iteration. The last entry in the two eigenvectors we seek is dominant and, in order to keep the example simple, we choose a special multiple of  $e_4:=(0, 0, 0, 1)^t$  as the starting vector in all cases.

This special multiple simplifies the calculations. In each case we factor a nearly singular matrix

$$\overset{\circ}{L}\overset{\circ}{D}\overset{\circ}{L}^{\mathrm{t}}-\tau I=LDL^{\mathrm{t}}$$

and the approximate eigenvector x is computed from

 $LDL^{t} \mathbf{x} = \mathbf{e}_{4} \mu.$ 

Eigenvalues			
$\lambda$ (for T)		$\mu$ (for $T-I$ )	
-6.890205972143757e-16		-1.000000000000000e+00	
1.000000019265610e+00		1.926561034787264e-08	
1.000000040339034e+00		4.033903460463116e-08	
2.000000000000001e+00		1.000000000000000000000000000000000000	
	Factor $T - \lambda_2 I$		
multipliers	pivots	Eigenvector	
-1.620151311728313e+08	-4.364449024407691e-09	7.071067783870038e-01	
6.172263002603381e-09	$1.145619979071370\mathrm{e}{+08}$	4.364449007128170e-09	
7.071067783870034e-01	2.107342433887993e-08	-7.071067783870034e-01	
0	8.343291570375477e-17	1.000000000000000000000000000000000000	
$\ (T - \lambda)\ $	$\ x_2I x\  = 8.34329157037548$	83e - 17	
	Factor $(T-I) - \mu_2 I$		
multipliers	pivots	Eigenvector	
-1.620151263612359e+08	-4.364449154024987e-09	7.071067870854733e-01	
6.172263185909920e-09	1.145619945048253e + 08	4.364449190434709e-09	
7.071067870854730e-01	2.107342407964534e-08	-7.071067870854730e-01	
0	$-1.758016782764627\mathrm{e}{\text{-}16}$	1.000000000000000000000000000000000000	
$\ ((T-I))\ $	$-\mu_2 I) x \  = 1.37599565703$	5863e - 16	
	Factor $T - \lambda_3 I$		
multipliers	pivots	Eigenvector	
-2.779740182130631e+07	-2.543787314124302e-08	-7.071067932881644e-01	
3.597458519427228e-08	1.965573132721291e+07	-2.543787357659366e-08	
-7.071067932881619e-01	-2.107342389479081e-08	7.071067932881619e-01	
0	3.606562560959997e-16	1.000000000000000000000000000000000000	
Factor $(T-I) - \mu_3 I$			
multipliers	pivots	Eigenvector	
-2.779740152676420e+07	-2.543787341078350e-08	-7.071067751996087e-01	
3.597458557546010e-08	$1.965573111894018e{+}07$	-2.543787319540585e-08	
-7.071067751996061e-01	-2.107342443387178e-08	7.071067751996061e-01	
0	$-1.784247126666214e{-}16$	1.000000000000000000000000000000000000	
dot prod from $\lambda$ 's: -1.315511610755493e-08			
dot prod from $\mu$ 's: 1.244713221382199e-10			

Fig. 1. Inverse iteration.

We chose  $\mu = D_{4,4}$ . Since L is unit lower triangular  $Le_4 = e_4$  and our choice of  $\mu$  yields  $L^t x = e_4$ . Backsolving yields

$$\boldsymbol{x} = \begin{pmatrix} -\ell_1 \ell_2 \ell_3 \\ +\ell_2 \ell_3 \\ -\ell_3 \\ 1 \end{pmatrix},$$

where  $\ell_i = L(i+1,i)$ , i = 1, 2, 3. Thus  $||\mathbf{x}||_2 > 1$  and the accuracy of  $\mathbf{x}$  is completely determined by the accuracy of L. In exact arithmetic

$$||(\overset{\circ}{L}\overset{\circ}{D}\overset{\circ}{L}^{\mathsf{t}}-\tau I)\boldsymbol{x}||=||LDL^{\mathsf{t}}\boldsymbol{x}||=|D_{4,4}|$$

and, when  $\tau$  is very accurate, then  $D_{4,4}$  can be  $O(\varepsilon |\tau|)$ . The roundoff errors in computing  $(\ell_2 \ell_3)$  and  $\ell_1(\ell_2 \ell_3)$  make negligible difference. We ignore them here and refer to [16] for the way those roundoff errors may be dealt with in general.

In the figures that follow we exhibit the nontrivial entries in L, D, and x for various cases.

First, we compared inverse iteration on T and T - I, see Fig. 1. The computed vectors are not orthogonal to working accuracy.

Next, we abandon T and T - I and take as our representation  $L_1D_1L_1^t = T - I$ . This looks bad because of element growth.

Lower part of $L_1$	Diagonal of $D_1$
$\begin{array}{r} 4.745313281212578e + 07 \\ -2.107342425544699e - 08 \\ 2.50000000000001e - 01 \\ 0 \end{array}$	$\begin{array}{r} 1.490116119384766e-08\\ -3.35544320000004e+07\\ 5.960464477539061e-08\\ 2.607703208923340e-08\end{array}$

 $||(T-I) - L_1 D_1 L_1^t|| = 0.$ 

The computed product  $L_1D_1L_1^{t}$  turned out to equal T - I exactly.

We computed the eigenvalues v of  $L_1D_1L_1^t$  by bisection but never formed the product  $L_1D_1L_1^t$ . For each sample  $\tau$  we computed  $L_1D_1L_1^t - \tau I = LDL^t$  using the differential form of the stationary qd algorithm (Section 3) and counted the number of negative diagonal entries in D.

The eigenvalues  $v_2$ ,  $v_3$  differ from the  $\mu_2$ ,  $\mu_3$  computed by Matlab from T - I in their last eight digits. This is because  $L_1D_1L_1^t$  defines its two tiny eigenvalues to high relative accuracy despite the element growth. The large eigenvalues are not so well represented. There is a precise 'relative condition number', greater than or equal to one, that measures the relative change in an eigenvalue due to tiny relative changes in the parameters in  $L_1$  and  $D_1$ . The condition numbers of our two eigenvalues  $v_2$  and  $v_3$  are less than 3 whereas the condition for the two extreme eigenvalues, near -1 and +1, are about  $10^8$ .

Fig. 2 shows the difference made by using  $v_2$  and  $v_3$  instead of  $\mu_2$  and  $\mu_3$  to obtain new factorizations. Notice how the last pivot changes from  $10^{-16}$  to  $10^{-23}$ . The improved eigenvalues coupled with the high accuracy of the differential stationary qd algorithm combine to correct the lower halves of the entries of the *L* factors and so give fully accurate eigenvectors of  $L_1D_1L_1^t$ . The computations are shown in Fig. 2.

We must mention that in this example T - I also defines its two small eigenvalues to high relative accuracy. If we discard Matlab's eigenvalues  $\mu_2$  and  $\mu_3$  and use bisection we get  $v_2$  and  $v_3$  instead. If we then use these eigenvalues in inverse iteration starting from  $e_4$  we get the same excellent eigenvectors as in the new method. In this case the diagonal entries of T - I have the same exponent as the two small eigenvalues and the subtraction in the shift is done exactly. The point is that in general the standard representation does not define the small eigenvalues to this high relative accuracy. This example shows that element growth in the factorization does not stop the small eigenvalues being well determined by the triangular factors.

#### 5. Unsymmetric case

This case needs more attention. Real matrices may have complex eigenvalues and, of more concern, some eigenvalues may be extremely sensitive to small changes in the matrix while others may be

Eigenvalues				
$\mu$ (for $T-I$ )		$\nu$ (for $L_1 D_1 L_1^t$ )		
-1.000000000000000000000000000000000000		-1.00000000000000000e+00		
1.926561034787264e-08		1.926561025997181e-08		
4.033903460463116e-08		4.033903451541880e-08		
1.000000000000000000000000000000000000		1.000000000000000000000000000000000000		
	Factor $L_1 D_1 L_1^t - \nu_2 I$			
multipliers	pivots	Eigenvector		
-1.620151296242512e+08	-4.364449066124159e-09	7.071067811865487e-01		
6.172263061599378e-09	1.145619968121254e+08	4.364449066124166e-09		
7.071067811865482e-01	2.107342425544699e-08	-7.071067811865482e-01		
0	-1.985233470127266e-23	1.000000000000000000000000000000000000		
	Factor $(T-I) - \mu_2 I$			
multipliers	pivots	Eigenvector		
-1.620151263612359e+08	-4.364449154024987e-09	7.071067870854733e-01		
6.172263185909920e-09	1.145619945048253e+08	4.364449190434709e-09		
7.071067870854730e-01	2.107342407964534e-08	-7.071067870854730e-01		
0	$-1.758016782764627\mathrm{e}{\text{-}16}$	1.000000000000000000000000000000000000		
	Factor $L_1 D_1 L_1^t - \nu_3 I$			
multipliers	pivots	Eigenvector		
-2.779740162425157e+07	-2.543787332157115e-08	-7.071067811865480e-01		
-7.071067811865458e-01	-2.107342425544707e-08	7.071067811865458e-01		
0	-1.323488980084844e-23	1.000000000000000000000000000000000000		
Factor $(T - I) - \mu_3 I$				
multipliers	pivots	Eigenvector		
-2.779740152676420e+07	-2.543787341078350e-08	-7.071067751996087e-01		
3.597458557546010e-08	1.965573111894018e+07	$-2.543787319540585\mathrm{e}{-08}$		
-7.071067751996061e-01	-2.107342443387178e-08	7.071067751996061e-01		
0	0 -1.784247126666214e-16 1.00000000000000e+0			
dot prod from $\mu$ 's: 1.244713221382199e-10				
dot prod from $\nu$ 's: -6.661338147750939e-16				

Fig. 2. New method.

robust. Unsymmetric tridiagonal matrices arise as output from the (two-sided) Lanczos algorithm applied to a large sparse general matrix. The lack of a good tridiagonal eigensolver to complete the calculation has hindered the acceptance of the unsymmetric Lanczos algorithms for large sparse problems.

It is not easy to find an algorithm that preserves both the eigenvalues and the tridiagonal form. None of the current methods is, in addition, backward stable. A method is backward stable if the computed eigenvalues are exact for some matrix close to the original one.

One of the earliest methods was the LR algorithm described Section 3. In 1978 came the HR algorithm of Bunse–Gerstner, see [3,4]. In 1992 came XHR from Parlett and Liu, see [14]. In 1996 and 1998 came two related methods by Uhlig called DQR and IQR PWK, see [24,23]. All of these methods work with a tridiagonal matrix.

The ideas described in Sections 1 and 3 suggest that the triangular factors might be a preferable representation to their product even in the unsymmetric case. If this is so, at least in important special cases, then we are fortunate that we have an algorithm at hand, namely dqds, that avoids the loss of information inherent in explicitly forming the product of bidiagonals.

The nice high mixed relative stability property mentioned in Section 3 extends without change to this case. Two practical lessons have been learned in working with the LR algorithm. First, by doubling the storage (from 2n to 4n cells) a transformation may be computed and then either accepted or rejected. Thus, unsatisfactory transformations merely waste a little time. They may be discarded and a better shift invoked. Second, the standard simple shift strategies based on asymptotic properties need to be supplemented with sophisticated choices in the early stages of the process.

The motivation for the method described in Sections 1 and 3 was to compute orthogonal eigenvectors. Of course, this is out of the question in the unsymmetric case because the eigenvectors need not be orthogonal. Recall that in the symmetric case we achieved orthogonality *indirectly* by attempting to compute accurate eigenvectors. This goal we can retain. The reason for hope is that the high relative accuracy property of dstqds and dqds is independent of symmetry. When LU defines its small eigenvalues to high relative accuracy then we should achieve (1), the small relative residual property.

A prototype implementation of dqds algorithm for eigenvalues entirely in complex arithmetic has been used with excellent results by David Day in building a nonsymmetric Lanczos algorithm for large sparse matrices. See [6].

There is room for more investigation on this topic.

#### References

- E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, D. Sorensen, LAPACK Users' Guide, 2nd Edition, SIAM, Philadelphia, 1995, 324pp.
- [2] J. Barlow, J.W. Demmel, Computing accurate eigensystems of scaled diagonally dominant matrices, SIAM J. Num. Anal. 27 (1990) 762–791.
- [3] A. Bunse-Gerstner, Berechnung der Eigenwerte einer Matrix mit dem *HR*-Verfahren (German and English summary), Numerische Behandlung von Eigenwertaufgaben, Band 2 (Tagung, Tech. Univ. Clausthal, Clausthal, 1978), pp. 26
   –39, Internat. Schriftenreihe Numer. Math., 43, Birkhauser, Basel-Boston, MA, 1979.
- [4] A. Bunse-Gerstner, An analysis of the *HR* algorithm for computing the eigenvalues of a matrix, Linear Algebra Appl. 35 (1981) 155–173.
- [5] J. Demmel, Applied Numerical Algebra, SIAM, Philadelphia, 1997.
- [6] D. Day, An efficient implementation of the nonsymmetric Lanczos algorithm, SIAM J. Matrix Anal. Appl. 18 (1997) 566–589.
- [7] I.S. Dhillon, A new  $O(n^2)$  algorithm for the symmetric tridiagonal eigenvalue/eigenvector problem, Ph.D. Thesis, Computer Science Division, Department of Electrical Engineering and Computer Science, University of California, Berkeley, California, May 1997. Also available as Computer Science Division Technical Report No. UCB//CSD-97-971.
- [8] I.S. Dhillon, G.I. Fann, B.N. Parlett, Application of a new algorithm for the symmetric eigenproblem to computational chemistry, in: Proceedings of the Eigth SIAM Conference on Parallel Processing for Scientific Computing, Minneapolis, MN, SIAM, Philadelphia, March 1997.
- [9] S. Eisenstat, I.C.F. Ipsen, Relative perturbation bounds for eigenspaces and singular vector subspaces, in: J.G. Lewis (Ed.), Proceedings of the Fifth SIAM Conference on Applied Linear Algebra, SIAM, Philadelphia, 1994, pp. 62–65.
- [10] S. Eisenstat, I.C.F. Ipsen, Relative perturbation techniques for singular value problems, SIAM J. Numer. Anal. 32 (1995) 1972–1988.
- [11] G.I. Fann, R.J. Littlefield, Performance of a fully parallel dense real symmetric eigensolver in quantum chemistry applications, Proceedings of High Performance Computing'95, Simulation MultiConference, The Society for Computer Simulation, 1995.
- [12] K.V. Fernando, B.N. Parlett, Accurate singular values and differential qd algorithms, Numer. Math. 67 (1994) 191– 229.

- [13] G.J.F. Francis, The QR Transformation, Parts I and II, Comput. J. 4 (1961-62), 265-271, 332-345.
- [14] Z.S. Liu, On the extended HR algorithm, Ph. D. Thesis, Technical Report CPAM-564 (August 1992), Center for Pure and Applied Math., Univ. of California, Berkeley.
- [15] B.N. Parlett, I.S. Dhillon, Fernando's solution to Wilkinson's problem: an application of double factorization, Linear Algebra Appl. 267 (1997) 247–279.
- [16] I.S. Dhillon, B.N. Parlett, Orthogonal eigenvectors and relative gaps, submitted to SIAM J. Matrix Anal. Appl.
- [17] B.N. Parlett, I.S. Dhillon, Relatively robust representations for symmetric tridiagonal matrices, Linear Algebra Appl. 309 (2000) 121–151.
- [18] B.N. Parlett, The Symmetric Eigenvalue Problem, 2nd Edition, SIAM, Philadelphia, 1998, 398pp.
- [19] B.N. Parlett, O.A. Marques, An implementation of the dqds algorithm (Positive case), Linear Algebra Appl. 309 (2000) 217–259.
- [20] H. Rutishauser, Der quotienten-differenzen-algorithmus, Z. Angew. Math. Phys. 5 (1954) 233-251.
- [21] H. Rutishauser, Solution of eigenvalue problems with the LR-transformation, Nat. Bur. Standards Appl. Math. Ser. 49 (1958) 47–81.
- [22] H. Rutishauser, Lectures on Numerical Mathematics, Birkhäuser, Boston, 1990.
- [23] F. Uhlig, A fast and simple method to compute the eigenvalues of general complex tridiagonal matrices, preprint, Olga Tausssky-Todd Memorial Meeting, June 1996.
- [24] F. Uhlig, The DQR algorithm, basic theory, convergence, and conditional stability, Numer. Math. 76 (1997) 515–553.



Journal of Computational and Applied Mathematics 123 (2000) 131-153

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.nl/locate/cam

# An overview of relative $\sin \Theta$ theorems for invariant subspaces of complex matrices $\stackrel{\text{\tiny{$\stackrel{\sim}{$}}}}{\to}$

Ilse C.F. Ipsen\*

Center for Research In Scientific Computation, Department of Mathematics, North Carolina State University, P.O. Box 8205, Raleigh, NC 27695-8205, USA

Received 28 May 1999; received in revised form 15 September 1999

#### Abstract

Relative perturbation bounds for invariant subspaces of complex matrices are reviewed, with emphasis on bounding the sines of the largest principal angle between two subspaces, i.e.  $\sin \Theta$  theorems. The goal is to provide intuition, as well as an idea for why the bounds hold and why they look the way they do. Relative bounds have the advantage of being better at exploiting structure in a perturbation than absolute bounds. Therefore the reaction of subspaces to relative perturbations can be different than to absolute perturbations. In particular, there are certain classes of relative perturbations to which subspaces of indefinite Hermitian matrices can be more sensitive than subspaces of definite matrices. © 2000 Elsevier Science B.V. All rights reserved.

MSC: 15-02; 15A18; 15A42; 65F15; 65F35

Keywords: Invariant subspace; Eigenspace; Relative bound; Eigenvalues; Grading; Scaling

# 1. Introduction

The goal is to assess the quality of perturbed invariant subspaces of complex matrices. Of interest is a new class of perturbation bounds, called relative perturbation bounds. Relative bounds are better at exploiting structure in a perturbation than absolute bounds. In particular, relative bounds can be sharper than traditional bounds when the perturbations arise from numerical errors of certain computational methods. The following example illustrates what we mean by relative bounds.

<sup>\*</sup> This research was supported in part by NSF grant DMS-9714811.

<sup>\*</sup> Corresponding author.

E-mail address: ipsen@math.ncsu.edu (I.C.F. Ipsen); http://wwwy.ncsu.edu/~ipsen/info.html

Example 1.1 (Ipsen [18, Example 1]). Suppose

$$A \equiv \begin{pmatrix} a & & \\ & b & \\ & & c \end{pmatrix},$$

is a complex diagonal matrix of order 3 with distinct eigenvalues a, b, and c; and

$$A + E_1 \equiv \begin{pmatrix} a & \varepsilon & \varepsilon \\ & b & \varepsilon \\ & & c \end{pmatrix}$$

is a perturbed matrix with the same eigenvalues as A. We want to compare the eigenvectors of A and  $A + E_1$  associated with eigenvalue c. The matrix A has  $(0\ 0\ 1)^T$  as an eigenvector<sup>1</sup> associated with c, while  $A + E_1$  has

$$\left(\frac{\varepsilon}{c-a}\left(\frac{\varepsilon}{c-b}+1\right) \quad \frac{\varepsilon}{c-b} \quad 1\right)^{\mathrm{T}}$$

The difference between these two eigenvectors depends on  $\varepsilon/(c-a)$  and  $\varepsilon/(c-b)$ . This suggests that the angle between the two vectors can be bounded in terms of

$$||E_1||/\min\{|c-a|, |c-b|\},$$
(1.1)

as  $||E_1|| = \mathcal{O}(|\varepsilon|)$ .

Now consider the perturbed matrix

$$A+E_2\equiv \begin{pmatrix} a & a\varepsilon & a\varepsilon \\ & b & b\varepsilon \\ & & c \end{pmatrix},$$

with the same eigenvalues as A. Again, compare eigenvectors of A and  $A + E_2$  associated with eigenvalue c. An eigenvector of  $A + E_2$  associated with eigenvalue c is

$$\left(\frac{\varepsilon a}{c-a}\left(\frac{\varepsilon b}{c-b}+1\right)\quad\frac{\varepsilon b}{c-b}\quad 1\right).$$

The difference between the eigenvectors of A and  $A + E_2$  depends on  $\varepsilon a/(c-a)$  and  $\varepsilon b/(c-b)$ . This suggests that their angle can be bounded in terms of

$$||A^{-1}E_2||/\min\left\{\frac{|c-a|}{|a|}, \frac{|c-b|}{|b|}\right\},\tag{1.2}$$

as  $||A^{-1}E_2|| = \mathcal{O}(|\varepsilon|).$ 

Bound (1.1) is a traditional, absolute bound and  $\min\{|c-a|, |c-b|\}$  is an absolute eigenvalue separation, while (1.2) is a relative bound and  $\min\{|c-a|/|a|, |c-b|/|b|\}$  is a relative eigenvalue separation.  $\Box$ 

The absolute bound contains ||E|| and an absolute separation, while the relative bound contains  $||A^{-1}E||$  and a relative separation. This means, the absolute bound measures sensitivity with regard to perturbations E, while the relative bound measures sensitivity with regard to perturbations  $A^{-1}E$ .

<sup>&</sup>lt;sup>1</sup> The superscript T denotes the transpose.

The sensitivity to absolute perturbations is determined by an absolute separation, while the sensitivity to relative perturbations is determined by a relative separation.

There are other ways to construct relative bounds, by taking advantage of structure in the perturbation. The estimates provided by absolute and relative bounds can be very different. Which bound to use depends on the particular matrix and perturbation. One does not know yet in general which type of bound gives the best result for a given matrix and perturbation.

One advantage of relative perturbation bounds is that they can explain why some numerical methods are much more accurate than the traditional, absolute bounds would predict. That is because the errors caused by these methods can be expressed as small, relative perturbations. Specifically for the computation of eigenvectors, numerical methods that deliver high relative accuracy include:

- Inverse iteration for real symmetric scaled diagonally dominant matrices [1, Section 11] and real symmetric positive-definite matrices [8, Section 5].
- Two-sided Jacobi methods for real symmetric positive-definite matrices [8, Section 3].
- QR algorithms for real symmetric tridiagonal matrices with zero diagonal [6, Sections 5 and 6].
- Cholesky factorization followed by SVD of Cholesky factor for scaled diagonally dominant tridiagonals [1, Section 10] and for symmetric positive-definite matrices [7, Section 12]; [8, Section 4.3]; [23].
- Shifted Cholesky factorization followed by inverse iteration for real symmetric tridiagonal matrices [9, Section 5]; [10]; [28, Section 1].

Relative bounds are better at exploiting structure in perturbations than absolute bounds. For instance, from the point of view of absolute bounds there is no need to distinguish between definite and indefinite Hermitian matrices when it comes to sensitivity of invariant subspaces. However, from the point of view of relative bounds subspaces of indefinite Hermitian matrices can be more sensitive to perturbations than those of definite matrices for certain classes of perturbations, see Sections 3.3-3.6.

# 1.1. Overview

This article is a successor to the survey on relative perturbation bounds for eigenvalues and singular values [19] and a previous review [29]. Here we review relative perturbation bounds for invariant subspaces. Due to space limitations the emphasis is on bounding the sines of the largest principal angle between two subspaces, i.e.  $\sin \Theta$  theorems. Some information can get lost by focussing on an angle. For instance,  $\sin \Theta$  theorems give no information about the accuracy of individual eigenvector components. Such bounds on individual components are derived, for instance, in [1, Section 7]; [8, Section 2]; [25, Section 3]; [23, Theorem 3.3]; [22, Theorem 4].

The goal is to provide intuition, as well as an idea for why the bounds hold and why they look the way they do. We present and derive relative as well as absolute bounds to show that there is nothing inherently special about relative bounds. Sometimes relative bounds are even implied by absolute bounds, hence they are not necessarily stronger than absolute bounds.

Relative bounds have been derived in the context of two different perturbation models:

• Additive perturbations (Section 3) represent the perturbed matrix as A + E. Bounds for the following matrix types are presented: general (Section 3.1), diagonalizable (Section 3.2), Hermitian

positive-definite (Section 3.3), graded Hermitian positive-definite (Section 3.4), Hermitian indefinite (Section 3.5), and graded Hermitian indefinite (Section 3.6).

• *Multiplicative* perturbations (Section 4) represent the perturbed matrix as  $D_1AD_2$ , where  $D_1$  and  $D_2$  are nonsingular matrices. Bounds are presented for diagonalizable (Section 4.1) and Hermitian matrices (Section 4.2).

# 1.2. Notation

Individual elements of a matrix A are denoted by  $a_{ij}$ . We use two norms: the two-norm

$$||A||_2 = \max_{x \neq 0} \frac{||Ax||_2}{||x||_2}$$
 where  $||x||_2 \equiv \sqrt{x^*x}$ 

and the superscript \* denotes the conjugate transpose; and the Frobenius norm

$$||A||_{\mathrm{F}} = \sqrt{\sum_{i,j} |a_{ij}|^2}.$$

The norm  $\|\cdot\|$  stands for both, Frobenius and two-norm. The identity matrix of order *n* is

$$I = \begin{pmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{pmatrix} = (e_1 \dots e_n)$$

with columns  $e_i$ .

For a complex matrix Y, range(Y) denotes the column space,  $Y^{-1}$  is the inverse (if it exists) and  $Y^{\dagger}$  the Moore–Penrose inverse. The two-norm condition number with respect to inversion is  $\kappa(Y) \equiv ||Y||_2 ||Y^{\dagger}||_2$ .

#### 2. The problem

Let A be a complex square matrix. A subspace  $\mathscr{S}$  is an *invariant subspace* of A if  $Ax \in \mathscr{S}$  for every  $x \in \mathscr{S}$  [15, Section 1.1]; [34, Section I.3.4]. Applications involving invariant subspaces are given in [15].

Let  $\hat{\mathscr{S}}$  be a perturbed subspace. The distance between the exact space  $\mathscr{S}$  and the perturbed space  $\hat{\mathscr{S}}$  can be expressed in terms of  $||P\hat{P}||$ , where P is the orthogonal projector onto  $\mathscr{S}^{\perp}$ , the orthogonal complement of  $\mathscr{S}$ , while  $\hat{P}$  is the orthogonal projector onto  $\hat{\mathscr{S}}$  [18, Section 2]. When  $\mathscr{S}$  and  $\hat{\mathscr{S}}$  have the same dimension, the singular values of  $P\hat{P}$  are the sines of the principal angles between  $\mathscr{S}$  and  $\hat{\mathscr{S}}$  [16, Section 12.4.3]; [34, Theorem I.5.5]. Therefore, we set

$$\sin \Theta \equiv P\hat{P}.$$

We present absolute and relative bounds for  $\|\sin \Theta\|$ , where  $\|\cdot\|$  is the two-norm or the Frobenius norm.

### 3. Additive perturbations for invariant subspaces

The perturbed subspace  $\mathscr{S}$  is interpreted as an exact subspace of a perturbed matrix A + E. Relative and absolute bounds on  $\|\sin \Theta\|$  are presented for the following matrix types: general, diagonalizable, Hermitian positive-definite, graded Hermitian positive-definite, Hermitian indefinite and graded Hermitian indefinite.

#### 3.1. General matrices

Absolute and relative bounds for invariant subspaces of complex square matrices are presented. The bounds make no reference to subspace bases and provide a unifying framework for subsequent bounds. They also illustrate that relative bounds exist under the most general of circumstances.

We start with the absolute bound. Define the absolute separation between A and A+E with regard to the spaces  $\mathscr{S}$  and  $\hat{\mathscr{S}}$  by

abssep 
$$\equiv$$
 abssep $_{\{A,A+E\}} \equiv \min_{\|Z\|=1,PZ\hat{P}=Z} \|PAZ - Z(A+E)\hat{P}\|$ 

where P is the orthogonal projector onto  $\mathscr{S}^{\perp}$ , and  $\hat{P}$  is the orthogonal projector onto  $\hat{\mathscr{S}}$ . The absolute bound below holds for any square matrix.

**Theorem 3.1** (Ipsen [18, Theorem 3.1]). If abssep > 0 then

 $\|\sin \Theta\| \leq \|E\|/\text{abssep}_{\{A,A+E\}}.$ 

**Proof.** From -E = A - (A + E) follows

$$-PE\hat{P} = PA\hat{P} - P(A+E)\hat{P}.$$

Since  $\mathscr{S}^{\perp}$  is an invariant subspace of  $A^*$ , the associated projector P satisfies PA = PAP. Similarly,  $(A + E)\hat{P} = \hat{P}(A + E)\hat{P}$ . Hence

$$-PE\hat{P} = PA\sin\Theta - \sin\Theta(A+E)\hat{P}$$

and  $\sin \Theta = P \sin \Theta \hat{P}$  implies

$$||E|| \ge ||PE\hat{P}|| \ge \text{abssep}||\sin \Theta||. \qquad \Box$$

Thus, the subspace  $\mathscr{S}$  is insensitive to absolute perturbations E if the absolute separation is large. Now we derive the corresponding relative bound. Define the relative separation between A and A + E with regard to the spaces  $\mathscr{S}$  and  $\hat{\mathscr{S}}$  by

$$\operatorname{relsep} \equiv \operatorname{relsep}_{\{A,A+E\}} \equiv \min_{\|Z\|=1, PZ\hat{P}=Z} \|PA^{-1}(PAZ - Z(A+E)\hat{P})\|_{\mathcal{P}}$$

where P is the orthogonal projector onto  $\mathscr{S}^{\perp}$ , and  $\hat{P}$  is the orthogonal projector onto  $\hat{\mathscr{S}}$ . The relative bound below holds for any nonsingular matrix.

# **Theorem 3.2** (Ipsen [18, Theorem 3.2]). If A is nonsingular and relsep > 0 then $\|\sin \Theta\| \leq \|A^{-1}E\|/\text{relsep}_{\{A,A+E\}}.$

**Proof.** From  $-A^{-1}E = I - A^{-1}(A + E)$  follows

$$-PA^{-1}E\hat{P} = P\hat{P} - PA^{-1}(A+E)\hat{P} = \sin\Theta - PA^{-1}(A+E)\hat{P}$$

Again, using the fact that PA = PAP and  $(A + E)\hat{P} = \hat{P}(A + E)\hat{P}$  one obtains

$$-PA^{-1}E\hat{P} = \sin \Theta - PA^{-1}\sin \Theta(A+E)\hat{P}$$
$$= PA^{-1}PA\sin \Theta - PA^{-1}\sin \Theta(A+E)\hat{P}$$
$$= PA^{-1}(PA\sin \Theta - \sin \Theta(A+E)\hat{P})$$

and  $\sin \Theta = P \sin \Theta \hat{P}$  implies

 $||A^{-1}E|| \ge ||PA^{-1}E\hat{P}|| \ge \operatorname{relsep}||\sin \Theta||.$   $\Box$ 

Thus, the subspace  $\mathscr{S}$  is insensitive to relative perturbations  $A^{-1}E$  if the relative separation is large. The derivation of the relative bound is very similar to the derivation of the absolute bound. In this sense, there is nothing special about a relative bound.

When the perturbed subspace has dimension one, the absolute bound implies the relative bound.

**Theorem 3.3** (Ipsen [18, Theorem 3.3]). If  $\hat{\mathscr{S}}$  has dimension one then Theorem 3.1 implies Theorem 3.2.

**Proof.** Since  $\hat{\mathscr{S}}$  has dimension one,  $\hat{X}$  consists of only one column, and  $\hat{B}$  is a scalar. Hence one can write  $(A + E)\hat{x} = \hat{\lambda}\hat{x}$ . Using  $\hat{P} = \hat{x}\hat{x}^*/\hat{x}^*\hat{x}$  and  $PZ\hat{P} = Z$ , Theorem 3.1 can be expressed as

 $\|\sin \Theta\| \leq \|E\|/\text{abssep} \text{ where abssep} = \min_{\|Z\|=1} \|P(A - \hat{\lambda}I)Z\|$ 

and Theorem 3.2 as

 $\|\sin\Theta\| \leq \|A^{-1}E\|/\text{relsep}$  where  $\text{relsep} = \min_{\|Z\|=1} \|PA^{-1}(A - \hat{\lambda}I)Z\|.$ 

The idea is to write  $(A + E)\hat{x} = \hat{\lambda}\hat{x}$  as  $(\tilde{A} + \tilde{E})\hat{x} = \hat{x}$ , where  $\tilde{A} \equiv \hat{\lambda}A^{-1}$ , and  $\tilde{E} \equiv -A^{-1}E$ . Note that  $\tilde{A}$  and  $\tilde{A} + \tilde{E}$  are associated with the same projectors P and  $\hat{P}$ , respectively, as A and A + E.

Theorem 3.1 implies Theorem 3.2 because applying the absolute bound to  $(\tilde{A} + \tilde{E})\hat{x} = 1 \cdot \hat{x}$  yields the relative bound. In particular, the norm in abssep is

$$||P(\tilde{A} - 1 \cdot I)Z|| = ||P(\hat{\lambda}A^{-1} - I)Z|| = ||PA^{-1}(A - \hat{\lambda}I)Z||,$$

which is equal to the norm in relsep.  $\Box$ 

Since the relative bound is derived by means of the absolute bound one cannot necessarily conclude that relative perturbation bounds are stronger than absolute bounds. However, there are particular matrices and classes of perturbations, where relative bounds can be much sharper than absolute bounds. **Example 3.1** (*Ipsen* [18, *Example 2*]). Let k > 0 and

$$A = \begin{pmatrix} 10^{-k} & & \\ & 2 \cdot 10^{-k} & \\ & & 10^k \end{pmatrix}.$$

Suppose  $\mathscr{S} = \operatorname{range}(1 \ 0 \ 0)^{T}$  is approximated by the subspace associated with the smallest eigenvalue  $\hat{\lambda} = 10^{-k}$  of

$$A + E = \begin{pmatrix} 10^{-k} & & \\ \epsilon 10^{-k} & 2 \cdot 10^{-k} & \\ \epsilon 10^{k} & \epsilon 10^{k} & 10^{k} \end{pmatrix},$$

where  $\varepsilon > 0$ . In this case,

$$\mathscr{S}^{\perp} = \operatorname{range} \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad P = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The absolute bound contains

$$P(A - \hat{\lambda}I) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 10^{-k} & 0 \\ 0 & 0 & 10^{k} - 10^{-k} \end{pmatrix}.$$

Hence, in the two-norm abssep  $\approx 10^{-k}$ . Since  $||E||_2 \approx \varepsilon 10^k$ , the absolute bound is

 $\|\sin \Theta\|_2 \leq \|E\|_2/\text{abssep} \approx \varepsilon 10^{2k}.$ 

In contrast, the relative bound contains

$$PA^{-1}(A - \hat{\lambda}I) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 1 - 10^{-2k} \end{pmatrix}.$$

Hence, in the two-norm relsep  $\approx 1$ . Since  $||A^{-1}E||_2 \approx \varepsilon$ , the relative bound is

 $\|\sin\Theta\|_2 \leq \|A^{-1}E\|_2/\text{relsep} \approx \varepsilon.$ 

In this case the relative bound is sharper by a factor of  $10^{2k}$  than the absolute bound.  $\Box$ 

In general, it is not known, under which circumstances a relative bound is better than an absolute bound, and which type of relative bound is the tightest for a given matrix and perturbation.

So far, we have considered bounds between two subspaces that make no reference to any basis. From a computational point of view, however, this may not be useful. This is why from now on we express subspace bounds in terms of specified bases. Such bounds turn out to be weaker, as they are derived by bounding from below abssep in Theorem 3.1 and relsep in Theorem 3.2.

Let Y and  $\hat{X}$  be respective bases for  $\mathscr{S}^{\hat{\perp}}$  and  $\hat{\mathscr{S}}$ , that is,

$$Y^*A = \Lambda Y^*$$
 where  $\mathscr{S}^{\perp} = \operatorname{range}(Y)$ 

and

$$(A+E)\hat{X} = \hat{X}\hat{A}$$
 where  $\hat{\mathscr{G}} = \operatorname{range}(\hat{X})$ 

for some matrices  $\Lambda$  and  $\hat{\Lambda}$ . This means, the eigenvalues of  $\hat{\Lambda}$  are the eigenvalues associated with the perturbed subspace  $\hat{\mathscr{S}}$ , while the eigenvalues of  $\Lambda$  are the eigenvalues associated with the exact subspace in which we are *not* interested, because the associated left subspace is orthogonal to the desired subspace  $\mathscr{S}$ . However, the separation between the eigenvalues of  $\Lambda$  and  $\hat{\Lambda}$  determines the quality of the perturbed subspace  $\hat{\mathscr{S}}$ . Denote by  $\kappa(Y) \equiv ||Y||_2 ||Y^{\dagger}||_2$  the two-norm condition number with respect to inversion. The absolute bound in Theorem 3.1 can be weakened to [18, (4.2)]

$$\|\sin\Theta\| \leq \kappa(Y)\kappa(\hat{X}) \|E\|/\text{abssep}(\Lambda,\hat{\Lambda}), \tag{3.1}$$

where

abssep
$$(\Lambda, \hat{\Lambda}) \equiv \min_{\|Z\|=1} \|\Lambda Z - Z \hat{\Lambda}\|.$$

When A is nonsingular, the relative bound in Theorem 3.2 can be weakened to [18, (4.3)]

$$\|\sin\Theta\| \leq \kappa(Y)\kappa(\hat{X}) \|A^{-1}E\|/\text{relsep}(\Lambda,\hat{\Lambda}), \tag{3.2}$$

where

relsep
$$(\Lambda, \hat{\Lambda}) \equiv \min_{\|Z\|=1} \|\Lambda^{-1}(\Lambda Z - Z\hat{\Lambda})\|.$$

Unfortunately, bounds (3.1) and (3.2) contain a quantity in which we are not really interested,  $\kappa(Y)$ , the conditioning of a basis for  $\mathscr{S}^{\perp}$ . Usually, Y is not explicitly specified, and we have some freedom of choice here. There are two simple options. Either choose Y as a basis of Schur vectors (then Y has orthonormal columns and  $\kappa(Y) = 1$ ), or choose Y as a basis of Jordan vectors (then  $\Lambda$  is diagonal when  $\Lambda$  is diagonalizable). We make the later choice for diagonalizable matrices, so that absolute and relative separations can be expressed in terms of eigenvalues. For normal and Hermitian matrices, fortunately, the two choices coincide.

# 3.2. Diagonalizable matrices

Relative and absolute bounds for eigenspaces of diagonalizable matrices are expressed in terms of eigenvalues and conditioning of eigenvector bases.

Let  $\mathscr{S}$  and  $\widehat{\mathscr{S}}$  be respective eigenspaces for diagonalizable matrices A and A + E, and let the columns of Y and  $\widehat{X}$  be respective bases for  $\mathscr{S}^{\perp}$  and  $\widehat{\mathscr{S}}$ . That is

$$\mathscr{S}^{\perp} = \operatorname{range}(Y), \quad \hat{\mathscr{S}} = \operatorname{range}(\hat{X})$$

and

$$Y^*A = \Lambda Y^*, \quad (A+E)\hat{X} = \hat{X}\hat{A},$$

where  $\Lambda$  and  $\hat{\Lambda}$  are diagonal. We (ab)use the notation

$$\min_{\lambda \in \Lambda, \hat{\lambda} \in \hat{\Lambda}} |\lambda - \hat{\lambda}| \quad \text{and} \quad \min_{\lambda \in \Lambda, \hat{\lambda} \in \hat{\Lambda}} \frac{|\lambda - \lambda|}{|\lambda|}$$

to mean that the minima range over all diagonal elements  $\hat{\lambda}$  of  $\Lambda$  and all diagonal elements  $\hat{\hat{\lambda}}$  of  $\hat{\Lambda}$ .

**Theorem 3.4.** If A and A + E are diagonalizable then

$$\|\sin\Theta\|_{\mathrm{F}} \leq \kappa(Y)\kappa(\hat{X}) \|E\|_{\mathrm{F}} / \min_{\lambda \in A, \hat{\lambda} \in \hat{A}} |\lambda - \hat{\lambda}|.$$

If, in addition, A is nonsingular, then

$$\|\sin\Theta\|_{\mathrm{F}} \leq \kappa(Y)\kappa(\hat{X}) \|A^{-1}E\|_{\mathrm{F}} / \min_{\lambda \in \mathcal{A}, \hat{\lambda} \in \hat{\mathcal{A}}} \frac{|\lambda - \lambda|}{|\lambda|}.$$

**Proof.** The absolute bound follows from (3.1) and the fact that  $\operatorname{abssep}_{F}(\Lambda, \hat{\Lambda}) = \min_{\lambda \in \Lambda, \hat{\lambda} \in \hat{\Lambda}} |\lambda - \hat{\lambda}|$  [34, p. 245, Problem 3]. Regarding the relative bound, the norm in relsep $(\Lambda, \hat{\Lambda})$  can be bounded by

$$||Z - \Lambda^{-1} Z \hat{\Lambda}||_{\mathrm{F}}^{2} = \sum_{i,j} \left| 1 - \frac{\hat{\lambda}_{j}}{\lambda_{i}} \right|^{2} ||z_{ij}|^{2} \ge \min_{i,j} \left| 1 - \frac{\hat{\lambda}_{j}}{\lambda_{i}} \right|^{2} ||Z||_{\mathrm{F}}^{2}.$$

Now use  $\operatorname{relsep}_{F}(\Lambda, \hat{\Lambda}) \ge \min_{\lambda \in \Lambda, \hat{\lambda} \in \hat{\Lambda}} \frac{|\lambda - \hat{\lambda}|}{|\lambda|}$  in (3.2).  $\Box$ 

Thus, the eigenspace  $\mathscr{S}$  is insensitive to absolute (relative) perturbations if the eigenvector bases are well-conditioned and if the perturbed eigenvalues are well-separated in the absolute (relative) sense from the undesired exact eigenvalues.

In the particular case when  $\hat{\mathscr{G}}$  has dimension 1, the absolute bound in Theorem 3.4 reduces to [13, Theorem 3.1], see also Theorem 4.1.

Bounds similar to the Frobenius norm bounds in Theorem 3.4 can be derived for the two-norm. This is done either by bounding the Frobenius norm in terms of the two-norm and inheriting a factor of  $\sqrt{n}$  in the bound, where *n* is the order of *A* [18, Corollary 5.2], or by assuming that all eigenvalues of one matrix ( $\Lambda$  or  $\hat{\Lambda}$ ) are smaller in magnitude than all eigenvalues of the other matrix [18, Theorem 5.3].

When A and A + E are normal, the condition numbers for the eigenvector bases equal one, and the Frobenius norm bounds in Theorem 3.4 simplify.

**Corollary 3.5.** If A and A + E are normal then

$$\|\sin\Theta\|_{\rm F} \leq \|E\|_{\rm F} / \min_{\lambda \in \Lambda, \hat{\lambda} \in \hat{\Lambda}} |\lambda - \hat{\lambda}|.$$

If, in addition, A is non-singular, then

$$\|\sin \Theta\|_{\rm F} \leq \|A^{-1}E\|_{\rm F} / \min_{\lambda \in A, \, \hat{\lambda} \in \hat{A}} \frac{|\lambda - \hat{\lambda}|}{|\lambda|}.$$

Now the sensitivity of the subspace to absolute (relative) perturbations depends solely on the absolute (relative) eigenvalue separation. The absolute bound represents one of Davis and Kahan's  $\sin \Theta$  Theorems [4, Section 6]; [5, Section 2].

In particular, the above bounds hold for Hermitian matrices. However, the relative perturbation  $A^{-1}E$  is, in general, not Hermitian. By expressing the relative perturbation differently, one can obtain Hermitian perturbations. This is done in the following sections, where things become more complex

because we demand structure from relative perturbations. For instance, when relative perturbations are restricted to be Hermitian, subspaces of indefinite Hermitian matrices appear to be more sensitive than those of definite matrices.

# 3.3. Hermitian positive-definite matrices

Relative bounds with Hermitian perturbations are derived for eigenspaces of Hermitian positivedefinite matrices. We start by discussing positive-definite matrices because it is easy to construct relative perturbations that are Hermitian. Construction of Hermitian relative perturbations for indefinite matrices is more intricate, but the derivations are often guided by those for definite matrices.

In contrast to the preceding results, one would like to express relative perturbations for Hermitian matrices as  $A^{-1/2}EA^{-1/2}$ , where  $A^{1/2}$  is a square-root of A. The nice thing about Hermitian positive-definite matrices A is that one can choose  $A^{1/2}$  to be Hermitian. Hence  $A^{-1/2}EA^{-1/2}$  remains Hermitian whenever E is Hermitian.

Let  $\mathscr{S}$  and  $\hat{\mathscr{S}}$  be respective eigenspaces for Hermitian positive-definite matrices A and A + E, and let the columns of Y and  $\hat{X}$  be respective orthonormal bases for  $\mathscr{S}^{\perp}$  and  $\hat{\mathscr{S}}$ . That is

$$\mathscr{S}^{\perp} = \operatorname{range}(Y), \quad \hat{\mathscr{S}} = \operatorname{range}(\hat{X})$$

and

$$Y^*A = \Lambda Y^*, \quad (A+E)\hat{X} = \hat{X}\hat{A},$$

where  $\Lambda$  and  $\hat{\Lambda}$  are diagonal with positive diagonal elements. Since Y and  $\hat{X}$  have orthonormal columns,  $\|\sin \Theta\| = \|Y^* \hat{X}\|$ .

The derivation of the relative bound below was inspired by the proof of [26, Theorem 1].

**Theorem 3.6** (Londrè and Rhee [22, Theorem 1], Li [21, Theorem 3.3]). If A and A + E are Hermitian positive-definite, and if  $\eta_2 \equiv ||A^{-1/2}EA^{-1/2}||_2 < 1$  then

$$\|\sin\Theta\|_{\rm F} \leq \frac{\eta_{\rm F}}{\sqrt{1-\eta_2}} / \min_{\lambda \in A, \, \hat{\lambda} \in \hat{A}} \, \frac{|\lambda-\lambda|}{\sqrt{\lambda \hat{\lambda}}},$$

where  $\eta_{\rm F} \equiv \|A^{-1/2} E A^{-1/2}\|_{\rm F}$ .

**Proof.** Multiply  $(A + E)\hat{X} = \hat{X}\hat{A}$  on the left by  $Y^*$  and set  $S \equiv Y^*\hat{X}$ ,

$$\Lambda S - S\hat{A} = -Y^* E\hat{X} = -\Lambda^{1/2} W \hat{\Lambda}^{1/2},$$

where  $W \equiv -\Lambda^{-1/2} Y^* E \hat{X} \hat{\Lambda}^{-1/2}$ . Element (i, j) of the equation is

$$s_{ij} = -W_{ij} / rac{\lambda_i - \hat{\lambda}_j}{\sqrt{\lambda_i \hat{\lambda}_j}} ,$$

where  $\lambda_i$  and  $\hat{\lambda}_j$  are respective diagonal elements of  $\Lambda$  and  $\hat{\Lambda}$ . Summing up all elements gives

$$\|\sin\Theta\|_{\rm F} = \|S\|_{\rm F} \leq \|W\|_{\rm F} / \min_{\lambda \in A, \, \hat{\lambda} \in \hat{A}} \frac{|\lambda - \lambda|}{\sqrt{\lambda \hat{\lambda}}}.$$

From

$$W = Y^* A^{-1/2} E(A+E)^{-1/2} \hat{X} = Y^* A^{-1/2} E A^{-1/2} A^{1/2} (A+E)^{-1/2} \hat{X}$$

follows

 $||W||_{\mathrm{F}} \leq \eta_{\mathrm{F}} ||A^{1/2}(A+E)^{-1/2}||_{2}.$ 

Positive-definiteness is crucial for bounding  $||A^{1/2}(A + E)^{-1/2}||_2$ . Since  $A^{1/2}$  and  $(A + E)^{1/2}$  are Hermitian,

$$\begin{split} \|A^{1/2}(A+E)^{-1/2}\|_2^2 &= \|A^{1/2}(A+E)^{-1}A^{1/2}\|_2 = \|(I+A^{-1/2}EA^{1/2})^{-1}\|_2 \\ &\leqslant \frac{1}{1-\eta_2}. \quad \Box \end{split}$$

Thus, the eigenspace  $\mathscr{S}$  is insensitive to relative perturbations  $A^{-1/2}EA^{-1/2}$  if the relative separation between perturbed eigenvalues and the undesirable exact eigenvalues is large. Since the relative perturbation  $A^{-1/2}EA^{-1/2}$  in Theorem 3.6 is different from the preceding perturbation  $A^{-1}E$ , so is the relative eigenvalue separation. However, this is of little consequence: If one measure of relative eigenvalue separation is small, so are all others [20, Section 2]; [25, Section 1]. The above bound holds more generally for unitarily invariant norms [21, Theorem 3.4].

With regard to related developments, a bound on  $|A^{1/2}(A + E)^{-1/2} - I||_2$  is derived in [24]. Relative eigenvector bounds for the hyperbolic eigenvalue problem  $Ax = \lambda Jx$ , where A is Hermitian positive-definite and J is a diagonal matrix with diagonal entries of magnitude one are given in [31, Section 3.2], with auxiliary results in [33].

A relative perturbation of the form  $A^{-1/2}EA^{-1/2}$  not only has the advantage that it is Hermitian, it is also invariant under grading, when both A and E are graded in the same way. This is discussed in the next section.

# 3.4. Graded Hermitian positive-definite matrices

It is shown that the relative perturbations  $A^{-1/2}EA^{-1/2}$  from the previous section are invariant under grading. By 'grading' (or 'scaling') [1, Section 2]; [25, Section 1] we mean the following: There exists a nonsingular matrix D such that  $A = D^*MD$  where M is in some sense 'better-behaved' than A.

**Lemma 3.7** (Eisenstat and Ipsen [14, Corollary 3.4], Mathias [25, Lemma 2.2]). If  $A=D^*MD$  is positive definite and  $E = D^*FD$  then

$$||A^{-1/2}EA^{-1/2}|| = ||M^{-1/2}FM^{-1/2}||.$$

**Proof.** We reproduce here the proof of [19, Corollary 2.13]. Because A is Hermitian positive-definite, it has a Hermitian square-root  $A^{1/2}$ . Hence  $A^{-1/2}EA^{-1/2}$  is Hermitian, and the norm is an eigenvalue,

$$||A^{-1/2}EA^{-1/2}|| = \max_{1 \le j \le n} |\lambda_j(A^{-1/2}EA^{-1/2})|.$$
Now comes the trick. Since eigenvalues are preserved under similarity transformations, we can reorder the matrices in a circular fashion until all grading matrices have cancelled each other out,

$$\lambda_j(A^{-1/2}EA^{-1/2}) = \lambda_j(A^{-1}E) = \lambda_j(D^{-1}M^{-1}FD) = \lambda_j(M^{-1}F)$$
$$= \lambda_j(M^{-1/2}FM^{-1/2}).$$

At last recover the norm,

$$\max_{1 \le j \le n} |\lambda_j(M^{-1/2}FM^{-1/2})| = ||M^{-1/2}FM^{-1/2}||. \square$$

Application of Lemma 3.7 to Theorem 3.6 demonstrates that the relative perturbations do not depend on the grading matrix D.

**Corollary 3.8** (Li [21, Theorem 3.3]). If  $A = D^*MD$  and  $A + E = D^*(M + F)D$  are Hermitian positive-definite, where D is nonsingular, and if

$$\eta_2 \equiv \|M^{-1/2}FM^{-1/2}\|_2 < 1$$

then

$$\|\sin\Theta\|_{\rm F} \leq \frac{\eta_{\rm F}}{\sqrt{1-\eta_2}} / \min_{\lambda \in \Lambda, \hat{\lambda} \in \hat{\Lambda}} \frac{|\lambda - \hat{\lambda}|}{\sqrt{\lambda \hat{\lambda}}},$$

where  $\eta_{\rm F} \equiv \|M^{-1/2} F M^{-1/2}\|_{\rm F}$ .

Again, the above bound holds more generally for unitarily invariant norms [21, Theorem 3.4]. There are other relative bounds for Hermitian positive-definite matrices  $D^*MD$  that exploit grading in the error  $D^*FD$ .

- Component-wise first-order bounds on the difference between perturbed and exact eigenvectors, containing the perturbation  $||M^{-1/2}FM^{-1/2}||_2$ , a relative gap, as well as eigenvalues and diagonal elements of M [25, Section 3].
- Component-wise exact bounds with the same features as above [22, Theorem 4].
- Norm-wise and component-wise first-order bounds on the difference between exact and perturbed eigenvectors, containing eigenvalues of M and  $||F||_2$  [8, Section 2]. Here D is diagonal so that all diagonal elements of M are equal to one.

The next section shows how to deal with indefinite matrices, first without and then with grading.

#### 3.5. Hermitian indefinite matrices

The bound for positive-definite Hermitian matrices in Section 3.3 is extended to indefinite matrices, however with a penalty. The penalty comes about, it appears, because the relative perturbation is asked to be Hermitian.

To understand the penalty, it is necessary to introduce polar factors and J-unitary matrices. Let A be Hermitian matrix with eigendecomposition  $A = V\Omega V^*$  and denote by  $|\Omega|$  the diagonal matrix whose diagonal elements are the absolute values of the diagonal elements in  $\Omega$ . The generalization of this absolute value to non-diagonal matrices is the Hermitian positive-definite polar factor (or spectral

absolute value [37, Section 1]) of A,  $|A| \equiv V |\Omega| V^*$ . When A happens to be positive-definite then |A| = A. Note that the polar factor |A| has the same eigenvectors as A.

The J in the J-unitary matrices comes from the inertia of A. Write an eigendecomposition of A  $A = V O U^* = V |O|^{1/2} U |O|^{1/2} U^*$ 

$$A = V\Omega V^* = V |\Omega|^{1/2} J |\Omega|^{1/2} V^*,$$

where J is a diagonal matrix with  $\pm 1$  on the diagonal that reflects the inertia of A. A matrix Z with  $ZJZ^* = J$  is called J-unitary. When A is definite then  $J = \pm I$  is a multiple of the identity, hence J-unitary matrices are just plain unitary. One needs J-unitary matrices to transform one decomposition of an indefinite matrix into another. For instance, suppose one has two decompositions  $A = Z_1JZ_1^* = Z_2JZ_2^*$ . Then there exists a J-unitary matrix Z that transforms  $Z_1$  into  $Z_2$ . That is,

$$Z_1 = Z_2 Z, \quad ZJZ^* = J.$$

One such matrix is simply  $Z = Z_2^{-1}Z_1$ .

Now we are ready to extend Theorem 3.6.

**Theorem 3.9** (Simpler Version of Theorem 2 in Truhar and Slapničar, [36]). If A and A + E are Hermitian with the same inertia, and if  $\eta_2 \equiv ||A|^{-1/2} E|A|^{-1/2} ||_2 < 1$  then

$$\|\sin\Theta\|_{\rm F} \leq \|Z\|_2 \frac{\eta_{\rm F}}{\sqrt{1-\eta_2}} / \min_{\lambda \in A, \hat{\lambda} \in \hat{A}} \frac{|\lambda - \hat{\lambda}|}{\sqrt{\lambda \hat{\lambda}}},$$

where  $\eta_F \equiv ||A|^{-1/2} E|A|^{-1/2} ||_F$  and Z is J-unitary and defined in the proof below.

**Proof.** The proof is very similar to that of Theorem 3.6. Multiply  $(A + E)\hat{X} = \hat{X}\hat{A}$  on the left by  $Y^*$  and set  $S \equiv Y^*\hat{X}$ ,

$$\Lambda S - S\hat{\Lambda} = -Y^* E\hat{X} = -|\Lambda|^{1/2} W |\hat{\Lambda}|^{1/2},$$

where  $W \equiv -|\Lambda|^{-1/2} Y^* E \hat{X} |\hat{\Lambda}|^{-1/2}$ . Element (i, j) of the equation is

$$s_{ij} = -W_{ij} / \frac{\lambda_i - \hat{\lambda}_j}{\sqrt{|\lambda_i \hat{\lambda}_j|}},$$

where  $\lambda_i$  and  $\hat{\lambda}_j$  are respective diagonal elements of  $\Lambda$  and  $\hat{\Lambda}$ . Summing up all elements gives

$$\|\sin \Theta\|_{\mathrm{F}} = \|S\|_{\mathrm{F}} \leqslant \|W\|_{\mathrm{F}} / \min_{\lambda \in A, \, \hat{\lambda} \in \hat{A}} \frac{|\lambda - \hat{\lambda}|}{\sqrt{|\lambda \hat{\lambda}|}}.$$

From

$$W = Y^* |A|^{-1/2} E |A + E|^{-1/2} \hat{X} = Y^* |A|^{-1/2} E |A|^{-1/2} |A|^{1/2} |A + E|^{-1/2} \hat{X}$$

follows

 $||W||_{\mathbf{F}} \leq \eta_{\mathbf{F}} ||A|^{1/2} |A + E|^{-1/2} ||_2.$ 

Bounding  $||A|^{1/2}|A+E|^{-1/2}||_2$  requires more work than in the positive-definite case. The eigendecompositions  $A = V\Omega V^*$  and  $A + E = \hat{V}\hat{\Omega}\hat{V}^*$  lead to two decompositions for A + E,

$$A + E = \hat{V} |\hat{\Omega}|^{1/2} J |\hat{\Omega}|^{1/2} \hat{V}$$

and

$$\begin{split} A + E &= V |\Omega|^{1/2} J |\Omega|^{1/2} V^* + E \\ &= V |\Omega|^{1/2} (J + |\Omega|^{-1/2} V^* EV| |\Omega|^{-1/2}) |\Omega|^{1/2} V^* \\ &= (V |\Omega|^{1/2} Q |\Delta|^{1/2}) J (|\Delta|^{1/2} Q^* |\Omega|^{1/2} V^*), \end{split}$$

where

$$J + |\Omega|^{-1/2} V^* EV ||\Omega|^{-1/2} = Q \Delta Q^*$$

is an eigendecomposition with the same inertia as A + E since we got there via a congruence transformation. To summarize the two expressions

$$A + E = Z_1 J Z_1^* = Z_2 J Z_2^*,$$

where

$$Z_1 \equiv \hat{V} | \hat{\Omega} |^{1/2}, \quad Z_2 \equiv V | \Omega |^{1/2} Q | \varDelta |^{1/2}.$$

As explained above there exists a J-unitary matrix Z such that  $Z_2 = Z_1 Z$ . Use this in

$$\||A|^{1/2}|A + E|^{-1/2}\|_{2} = \||\Omega|^{1/2}V^{*}\hat{V}|\hat{\Omega}|^{-1/2}\|_{2} = \||\Omega|^{1/2}V^{*}\hat{V}Z_{1}^{-*}\|$$

to obtain

$$||A|^{1/2}|A + E|^{-1/2}||_2 = |||\Delta|^{-1/2}Z^*||_2 \leq ||\Delta^{-1}||^{1/2}||Z||_2$$

since  $\Delta$  is a diagonal matrix. It remains to bound  $\|\Delta^{-1}\|_2$ ,

$$\begin{split} \|\Delta^{-1}\|_2 &= \|(J + |\Omega|^{-1/2} V^* EV| |\Omega|^{-1/2})^{-1}\| \\ &= \|(I + J|\Omega|^{-1/2} V^* EV| |\Omega|^{-1/2})^{-1}\| \leq \frac{1}{1 - \eta_2}. \qquad \Box \end{split}$$

The bound in Theorem 3.9 looks similar to the bound in Theorem 3.6. But the square-roots in  $\eta_2$  and  $\eta_F$  now contain polar factors, and the relative eigenvalue separation has absolute values under the square-root. Moreover, there is an additional factor ||Z||, that's the penalty. In the lucky case when A happens to be positive-definite, Z is unitary and Theorem 3.9 reduces to Theorem 3.6. When A is indefinite, the eigenspace sensitivity can be magnified by the norm of the J-unitary matrix, which in some sense reflects the deviation of A from definiteness.

At this point it is not known how large ||Z|| can be, under which circumstances it will be large or small, and how much it really contributes to the sensitivity of a subspace. A quantity corresponding to ||Z|| in [36] is bounded in terms of  $||A^{-1}||$  and a graded polar factor of A. Preliminary experiments in [36, Sections 4 and 5] suggest that ||Z|| does not grow unduly. At present, we do not yet have a good understanding of why a subspace of an indefinite Hermitian matrix should be more sensitive to Hermitian relative perturbations than a subspace of a definite matrix.

Not all relative bounds for Hermitian matrices necessarily look like the one above. For instance, there are relative bounds specifically geared towards real symmetric tridiagonal matrices. The cosine between two Ritz vectors associated with an eigenvalue cluster of a real, symmetric tridiagonal matrix can be expressed in terms of a relative gap [27, Section 5]. Perturbations of the  $LDL^{T}$  decomposition

144

of a real, symmetric tridiagonal matrix lead to relative bounds on the tangent between eigenvectors, and an eigenvector condition number that depends on all eigenvalues, not just a single eigenvalue separation [28, Section 10].

Like a tridiagonal matrix, one can decompose any Hermitian matrix as  $A = G^*JG$ , where J is a diagonal matrix with diagonal entries  $\pm 1$ . The norm-wise perturbation of a spectral projector induced by perturbations of the factor G can be bounded in terms of a relative eigenvalue separation [35, (12)]; [32, Theorem 1].

#### 3.6. Graded indefinite Hermitian matrices

The bound for graded positive-definite matrices from Section 3.4 is extended to graded indefinite matrices,

Fortunately, this requires only a slight modification in the proof of Theorem 3.9.

**Theorem 3.10** (Simpler Version of Theorem 2 in Truhar and Slapničar [36]). If  $A=D^*MD$  and  $A+E=D^*(M+F)D$  are Hermitian, where D is nonsingular, and if

$$\eta_2 \equiv \| |M|^{-1/2} F |M|^{-1/2} \|_2 < 1$$

then

$$\|\sin\Theta\|_{\rm F} \leq \|Z\| \frac{\eta_{\rm F}}{\sqrt{1-\eta_2}} / \min_{\lambda \in A, \hat{\lambda} \in \hat{A}} \frac{|\lambda - \hat{\lambda}|}{\sqrt{\lambda \hat{\lambda}}},$$

where  $\eta_{\rm F} \equiv || |M|^{-1/2} F |M|^{-1/2} ||_{\rm F}$ , and Z is J-unitary.

Proof. As in the proof of Theorem 3.9 derive

$$\|\sin \Theta\|_{\mathrm{F}} = \|S\|_{\mathrm{F}} \leq \|W\|_{\mathrm{F}} / \min_{\lambda \in \Lambda, \hat{\lambda} \in \hat{\Lambda}} \frac{|\lambda - \hat{\lambda}|}{\sqrt{|\lambda \hat{\lambda}|}}$$

To bound  $||W||_F$ , represent A and A + E in terms of D and eigendecompositions of M and M + F, respectively. The scaling matrices D then cancel out with the scaling matrices in the error  $E = D^*FD$ .  $\Box$ 

A quantity corresponding to ||Z|| in [36] is bounded in terms of  $||A^{-1}||$  and  $||D^*|M|D||$ .

Other relative bounds for indefinite Hermitian matrices that exploit grading include the following.

- Norm-wise and component-wise first-order bounds on the difference between exact and perturbed eigenvectors of real, symmetric scaled diagonally dominant matrices [1, Section 7].
- Bounds on the norm-wise difference between corresponding eigenvectors of Hermitian matrices  $A = D^*MD$  and  $D^*(M + F)D$  in terms of a relative gap,  $||F||_2$  and an eigenvalue of a principal submatrix of M [17, Theorem 7]. This is an improvement over the bounds for symmetric scaled diagonally dominant matrices in [1, Section 7] and for positive-definite matrices in [8].

- Bounds on the cosines of angles between exact and perturbed eigenvectors of possibly singular Hermitian matrices [2, Section 4]. They can be applied to analyze the accuracy of subspaces in ULV down-dating [3].
- Bounds on the norm-wise perturbations in spectral projectors [29, Section 2], [35, (6), (7)], [37, Theorem 2.48]; [30].

#### 3.7. Remarks

The sensitivity of invariant subspaces to absolute perturbations E and to relative perturbations  $A^{-1}E$  is influenced by the same factors: conditioning of subspace bases, and separation of matrices associated with eigenvalues. When the matrices involved are Hermitian the sensitivity to absolute perturbations E is amplified by an absolute eigenvalue separation, and the sensitivity to relative perturbations  $A^{-1}E$  by a relative eigenvalue separation. None of these two perturbations seems to care about whether the Hermitian matrices are definite or indefinite.

This changes when one restricts relative perturbations to be Hermitian as well, i.e., of the form  $|A|^{-1/2}E|A|^{-1/2}$ . Then subspaces of indefinite matrices appear to be more sensitive to these perturbations than those of definite matrices. This phenomenon is not yet completely understood. In particular, it is not clear how much the sensitivity can worsen for an indefinite matrix, and in what way the sensitivity depends on the indefiniteness of the matrix. In general, one does not completely understand how exactly the fine-structure of a matrix and a perturbation affect the sensitivity of subspaces.

There is another observation that has not been fully exploited yet either. Invariant subspaces do not change under shifts, i.e., A and the shifted matrix  $A - \mu I$  have the same invariant subspaces. The condition numbers for the absolute perturbations are invariant under a shift, while those for relative perturbations are not. The question is, are there *optimal* shifts for computing subspaces, and what would 'optimal' mean in this context? In particular, one could shift a Hermitian matrix so it becomes positive-definite. Then the subspaces of the shifted matrix would look less sensitive to Hermitian relative perturbations. This approach is pursued to assess the sensivity of eigenvectors of factored real symmetric tridiagonal matrices to relative perturbations in the factors in [9, Section 5], [28, Section 10], and used to compute the eigenvectors in [10]. The approach based on shifting a matrix before evaluating sensitivity and computing subspaces deserves more investigation for general, Hermitian matrices.

Now we consider a different type of perturbation.

#### 4. Multiplicative perturbations

The perturbed subspace  $\mathscr{S}$  is interpreted as an exact subspace of a perturbed matrix  $D_1AD_2$ , where  $D_1$  and  $D_2$  are nonsingular. Relative and absolute bounds on  $\|\sin \Theta\|$  are presented for diagonalizable and Hermitian matrices.

When  $D_2 = D_1^{-1}$ , the perturbed matrix  $D_1AD_2$  is just a similarity transformation of A, which means that A and  $D_1AD_2$  have the same eigenvalues. When  $D_2 = D_1^*$  then  $D_1AD_2$  is a congruence transformation of A, which means that A and  $D_1AD_2$  have the same inertia when A is Hermitian.

Since the nonsingularity of  $D_1$  and  $D_2$  forces A and  $D_1AD_2$  to have the same rank, multiplicative perturbations are more restrictive than additive perturbations.

Multiplicative perturbations can be used, for instance, to represent component-wise perturbations of real bidiagonal matrices and of real symmetric tridiagonal matrices with zero diagonal [1, p. 770], [12, Section 4], [19, Example 5.1]. This is exploited in [28, Section 4], where the relative sensitivity of eigenvalues and eigenvectors of real symmetric tridiagonal matrices with regard to perturbations in the factors of a  $LDL^{T}$  factorization is analyzed. Since *L* is bidiagonal, a component-wise perturbation of *L* can be represented as  $D_{1}LD_{2}$ .

In a different application illustrated below, multiplicative perturbations represent deflation in block triangular matrices.

**Example 4.1** (Eisenstat and Ipsen [12, *Theorem* 5.2]). The off-diagonal block in the block triangular matrix

$$A = \begin{pmatrix} A_{11} & A_{12} \\ & A_{22} \end{pmatrix}$$

is to be eliminated, making the deflated matrix

$$\begin{pmatrix} A_{11} \\ & A_{22} \end{pmatrix}$$

block diagonal. When  $A_{11}$  is nonsingular one can factor

$$\begin{pmatrix} A_{11} \\ A_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{22} \end{pmatrix} \begin{pmatrix} I & -A_{11}^{-1}A_{12} \\ I \end{pmatrix}$$

Therefore, the deflated matrix represents a multiplicative perturbation  $D_1AD_2$ , where  $D_1 = I$  and

$$D_2 = \begin{pmatrix} I & -A_{11}^{-1}A_{12} \\ & I \end{pmatrix}.$$

Similarly, when  $A_{22}$  is nonsingular one can factor

$$\begin{pmatrix} A_{11} \\ A_{22} \end{pmatrix} = \begin{pmatrix} I & -A_{12}A_{22}^{-1} \\ I \end{pmatrix} \begin{pmatrix} A_{11} & A_{12} \\ A_{22} \end{pmatrix}.$$

In this case the deflated matrix represents a multiplicative perturbation  $D_1AD_2$ , where  $D_2 = I$  and

$$D_2 = \begin{pmatrix} I & -A_{12}A_{22}^{-1} \\ I \end{pmatrix}. \qquad \Box$$

#### 4.1. Diagonalizable matrices

A bound is presented between a perturbed one-dimensional eigenspace and an eigenspace of a diagonalizable matrix.

Suppose A is diagonalizable and  $\hat{\mathscr{S}} = \operatorname{range}(\hat{x})$ , where

$$D_1 A D_2 \hat{x} = \hat{\lambda} \hat{x}, \quad \|\hat{x}\|_2 = 1$$

for some nonsingular  $D_1$  and  $D_2$ , where  $D_1AD_2$  is not necessarily diagonalizable. In this section we explicitly choose  $\mathscr{S}$  to be the eigenspace associated with all eigenvalues of A closest to  $\hat{\lambda}$ . The remaining, further away eigenvalues form the diagonal elements of the diagonal matrix A, i.e.,

$$\min_{\lambda \in A} |\lambda - \hat{\lambda}| > \min_{i} |\lambda_{i}(A) - \hat{\lambda}|.$$

Then  $\mathscr{S}^{\perp}$  is the left invariant subspace of A associated with the eigenvalues in  $\Lambda$ . Let the columns of Y be a basis for  $\mathscr{S}^{\perp}$ , so  $\mathscr{S}^{\perp} = \operatorname{range}(Y)$  and

$$Y^*A = \Lambda Y^*.$$

In the theorem below the residual of  $\hat{x}$  and  $\hat{\lambda}$  is

$$r \equiv (A - \hat{\lambda}I)\hat{x}.$$

Theorem 4.1 (Eisenstat and Ipsen [13, Theorem 4.3]). If A is diagonalizable then

$$|\sin \Theta||_2 \leq \kappa(Y) ||r||_2 / \min_{\lambda \in \Lambda} |\lambda - \hat{\lambda}|.$$

If, in addition,  $D_1$  and  $D_2$  are nonsingular then

$$\|\sin \Theta\|_2 \leq \kappa(Y) \min\{\alpha_1, \alpha_2\} / \min_{\lambda \in A} \frac{|\lambda - \hat{\lambda}|}{|\hat{\lambda}|} + \|I - D_2\|_2,$$

where

$$lpha_1 \equiv \|D_1^{-1} - D_2\|_2, \quad lpha_2 \equiv \|I - D_1^{-1}D_2^{-1}\|_2.$$

**Proof.** To derive the absolute bound, multiply  $r = (A - \hat{\lambda}I)\hat{x}$  by  $Y^*$  and use  $Y^*A = Y^*\Lambda$ ,  $Y^*\hat{x} = (A - \hat{\lambda}I)^{-1}Y^*r$ .

With P being the orthogonal projector onto  $\mathscr{S}^{\perp} = \operatorname{range}(Y)$  one gets

$$P\hat{x} = (Y^{\dagger})^* Y^* \hat{x} = (Y^{\dagger})^* (\Lambda - \hat{\lambda}I)^{-1} Y^* r.$$

From  $\|\hat{x}\|_2 = 1$  follows

$$\|\sin \Theta\|_2 = \|P\hat{x}\|_2 \leq \kappa(Y) \|(\Lambda - \hat{\lambda}I)^{-1}\|_2 \|r\|_2$$

To derive the relative bound, we will use the absolute bound. Multiply  $(D_1AD_2)\hat{x} = \hat{\lambda}\hat{x}$  by  $D_1^{-1}$  and set  $z \equiv D_2\hat{x}/||D_2\hat{x}||$ ,

$$Az = \hat{\lambda} D_1^{-1} D_2^{-1} z.$$

The residual for  $\hat{\lambda}$  and z is

$$f \equiv Az - \hat{\lambda}z = \hat{\lambda}(D_1^{-1}D_2^{-1} - I)z = \hat{\lambda}(D_1^{-1} - D_2)\hat{x}/\|D_2\hat{x}\|_2.$$

Hence

$$||f||_2 \leq |\hat{\lambda}|\alpha_2, \quad ||f||_2 \leq |\hat{\lambda}|\alpha_1/||D_2\hat{x}||_2.$$

The idea is to first apply the absolute bound to the residual f and then make an adjustment from z to  $\hat{x}$ . Since f contains  $\hat{\lambda}$  as a factor we will end up with a relative bound.

$$\|\sin \Phi\|_2 \leq \kappa(Y) \|f\|_2 / \min_{\lambda \in \Lambda} |\lambda - \hat{\lambda}|,$$

where  $\Phi$  represents the angle between z and  $\mathscr{S}$ . To make the adjustment from z to  $\hat{x}$  use the fact that [13, Lemma 4.2]

$$\|\sin \Theta\|_2 \leq \|\sin \Phi\|_2 + \|D_2 - I\|_2$$

and

$$\|\sin \Theta\|_2 \leq \|D_2 \hat{x}\|_2 \|\sin \Phi\|_2 + \|D_2 - I\|_2.$$

Now put the first bound for  $||f||_2$  into the first bound for  $||\sin \Theta||_2$  and the second bound for  $||f||_2$  into the second bound for  $||\sin \Theta||_2$ .  $\Box$ 

The relative bound consists of two summands. The first summand represents the (absolute or relative) deviation of  $D_1$  and  $D_2$  from a similarity transformation, amplified by the eigenvector conditioning  $\kappa(Y)$  and by the relative eigenvalue separation; while the second summand represents the (absolute and relative) deviation of the similarity transformation from the identity. The factor  $\alpha_1$  is an absolute deviation from similarity, while  $\alpha_2$  constitutes a relative deviation as

$$I - D_1^{-1}D_2^{-1} = (D_2 - D_1^{-1})D_2^{-1}$$

is a difference relative to  $D_2$ . Thus, for an eigenspace to be insensitive to multiplicative perturbations, the multiplicative perturbations must constitute a similarity transformation close to the identity.

Here again, as in Theorem 3.3, the relative bound is implied by the absolute bound. Also, when  $\hat{\mathscr{S}}$  has dimension 1, the absolute bound in Theorem 4.1 implies the absolute bound in Theorem 3.4.

**Example 4.2.** Let us apply Theorem 4.1 to Example 4.1. Suppose  $\hat{x}$  is a unit-norm eigenvector associated with an eigenvalue  $\hat{\lambda}$  of the deflated, block-diagonal matrix.

First consider the case when  $A_{11}$  is nonsingular. Then  $D_1 = I$  and

$$\alpha_1 = \alpha_2 = \|I - D_2\|_2 = \|A_{11}^{-1}A_{12}\|_2.$$

Hence

$$\|\sin\Theta\|_2 \leq \|A_{11}^{-1}A_{12}\|_2 \left(1 + 1/\min_{\lambda \in A} \frac{|\lambda - \hat{\lambda}|}{|\hat{\lambda}|}\right)$$

This means  $\hat{x}$  is close to an eigenvector of A if  $||A_{11}^{-1}A_{12}||_2$  is small compared to 1 and the relative eigenvalue separation. Hence, the matrix can be safely deflated without harming the eigenvector, if the leading diagonal block is 'large enough compared to' the off-diagonal block, and the meaning of 'large enough' is determined by the eigenvalue separation.

In the second case when  $A_{22}$  is nonsingular, one has  $D_2 = I$ . Hence  $||D_2 - I||_2 = 0$ ,  $\alpha_1 = \alpha_2 = ||A_{12}A_{22}^{-1}||_2$ , and

$$\|\sin \Theta\|_2 \leq \|A_{11}^{-1}A_{12}\|_2 / \min_{\lambda \in A} \frac{|\lambda - \hat{\lambda}|}{|\hat{\lambda}|}.$$

Now the matrix can be safely deflated without harming the eigenvector, if the trailing diagonal block is 'large enough compared to' the off-diagonal block.  $\Box$ 

In some cases the first summand in the bound of Theorem 4.1 can be omitted.

**Corollary 4.2** (Eisenstat and Ipsen [13, Corollary 4.4]). If  $D_1 = D_2^{-1}$  or  $\hat{\lambda} = 0$ , then

 $\|\sin\Theta\|_2 \leq \|I - D_2\|_2.$ 

**Proof.** First suppose  $D_1 = D_2^{-1}$ . Then  $D_1AD_2\hat{x} = \hat{\lambda}\hat{x}$  implies  $AD_2\hat{x} = \hat{\lambda}D_2\hat{x}$ , i.e.,  $\hat{\lambda}$  and  $D_2\hat{x}$  are an exact eigenpair of A. Since  $\mathscr{S}$  is the eigenspace associated with all eigenvalues closest to  $\hat{\lambda}$ , we must have  $D_2\hat{x} \in \mathscr{S}$ . Hence  $PD_2\hat{x} = 0$ , where P is the orthogonal projector onto  $\mathscr{S}^{\perp}$ , and

$$\|\sin \Theta\|_2 = \|P\hat{x}\|_2 = \|P(D_2\hat{x} - \hat{x})\|_2 \leq \|I - D_2\|_2.$$

Now suppose  $\hat{\lambda} = 0$ . Then  $D_1 A D_2 \hat{x} = 0 \cdot \hat{x}$  implies  $D_2^{-1} A D_2 \hat{x} = 0 \cdot \hat{x}$ , since  $D_1$  and  $D_2$  are nonsingular. Hence  $\hat{\lambda}$  and  $\hat{x}$  are an exact eigenpair of a similarity transformation of A, and we are back to the first case.  $\Box$ 

In the case of similarity transformations  $D_1 = D_2^{-1}$ , the eigenspace angle is bounded by the relative deviation of  $D_2$  from identity, without any amplification by  $\kappa(Y)$  or by a relative gap. As a consequence, eigenvectors of diagonalizable matrices are well-conditioned when the perturbation is a similarity transformation. Similarly, in the case  $\hat{\lambda} = 0$  it follows that null vectors of diagonalizable matrices are well-conditioned under multiplicative perturbations.

A different approach is sketched in [21, Remark 3.3] for deriving eigenspace bounds of diagonalizable matrices when both eigenspaces have the same dimension  $\ge 1$ .

#### 4.2. Hermitian matrices

Two-norm and Frobenius norm bounds are presented for multiplicative perturbations that are congruence transformations.

When applied to Hermitian matrices, Theorem 4.1 simplifies. Remember that in this context the perturbed eigenspace has dimension one,  $\hat{\mathscr{G}} = \operatorname{range}(\hat{x})$ , and

 $D^*AD\hat{x} = \lambda \hat{x}, \quad \|\hat{x}\|_2 = 1;$ 

and  $\mathscr{S}$  is the eigenspace of A associated with the eigenvalues of A closest to  $\hat{\lambda}$ .

Corollary 4.3 (Eisenstat and Ipsen [11, Theorem 2.1]). If A is Hermitian then

 $\|\sin\Theta\|_2 \leq \|r\|_2 / \min_{\lambda \in \Lambda} |\lambda - \hat{\lambda}|.$ 

If, in addition, D is nonsingular then

$$\|\sin \Theta\|_2 \leq \min\{lpha_1, lpha_2\} / \min_{\lambda \in A} \frac{|\lambda - \hat{\lambda}|}{|\hat{\lambda}|} + \|I - D\|_2,$$

where

$$\alpha_1 \equiv \|D^{-*} - D\|_2, \quad \alpha_2 \equiv \|I - D^{-*}D^{-1}\|_2.$$

The relative bound consists of two summands. The first summand represents the (absolute or relative) deviation of the equivalence transformation from a similarity, amplified by the relative eigenvalue separation; while the second summand represents the (absolute and relative) deviation of the similarity transformation from the identity. Hence the eigenspace  $\mathscr{S}$  is insensitive to perturbations that are equivalence transformations if the equivalence transformation is close to a similarity transformation that does not differ much from the identity.

In the special case when  $\mathscr{S}$  has dimension one, Corollary 4.3 is slightly stronger than [12, Theorem 2.2] and [11, Corollary 2.1].

Corollary 4.3 can be extended to bound angles between two eigenspaces of equal dimension  $k \ge 1$ , however at the expense of an additional factor  $\sqrt{k}$  in the bound [11, Theorem 3.1]. The following bounds for equally dimensioned subspaces do without this factor.

Let  $\mathscr{S}$  be an invariant subspace of A, and  $\hat{\mathscr{S}}$  be an invariant subspace of  $D^*AD$ , where D is nonsingular and  $\hat{\mathscr{S}}$  has the same dimension as  $\mathscr{S}$ . Let the columns of  $\hat{X}$  be an orthonormal basis for  $\hat{\mathscr{S}}$  and the columns of Y be an orthonormal basis for  $\mathscr{S}^{\perp}$ . Then

$$Y^*A = \Lambda Y^*, \quad D^*AD\hat{X} = \hat{X}\hat{A},$$

for some diagonal matrices  $\Lambda$  and  $\hat{\Lambda}$ . Below are Frobenius norm bounds on the angle between two equally dimensioned eigenspaces of a Hermitian matrix.

Theorem 4.4 (Li [21, Theorem 3.1]).<sup>2</sup> If A is Hermitian and D is nonsingular then

$$\|\sin\Theta\|_{\mathrm{F}} \leq \|(D-D^{-*})\hat{X}\|_{\mathrm{F}} / \min_{\lambda \in \mathcal{A}, \hat{\lambda} \in \hat{\mathcal{A}}} \frac{|\lambda - \hat{\lambda}|}{|\lambda|} + \|(I-D^{-*})\hat{X}\|_{\mathrm{F}}$$

and

$$\|\sin\Theta\|_{\mathrm{F}} \leq \sqrt{\|(I-D)\hat{X}\|_{\mathrm{F}}^{2} + \|(I-D^{-*})\hat{X}\|_{\mathrm{F}}^{2}} / \min_{\lambda \in \mathcal{A}, \, \hat{\lambda} \in \hat{\mathcal{A}}} \frac{|\lambda - \hat{\lambda}|}{\sqrt{\lambda^{2} + \hat{\lambda}^{2}}}.$$

These bounds give the same qualitative information as Corollary 4.3: The eigenspace is insensitive to perturbations that are congruence transformations if the congruence transformation is close to a similarity transformation that does not differ much from the identity. The first bound has the same form as the relative bound in Corollary 4.3. In particular, when  $\mathscr{S}$  and  $\hat{\mathscr{S}}$  have dimension one, the first bound in Theorem 4.4 implies

$$\|\sin\Theta\|_2 \leq \|(D-D^{-*})\|_2 / \min_{\lambda \in A, \hat{\lambda} \in \hat{A}} \frac{|\lambda - \hat{\lambda}|}{|\lambda|} + \|I - D^{-*}\|_2,$$

which is almost identical to the relative bound in Corollary 4.3. Note that the relative eigenvalue separation in the second bound is different. More generally, Theorem 4.4 holds in any unitarily invariant norm [21, Theorem 3.2].

Multiplicative eigenvector bounds for the hyperbolic eigenvalue problem  $Ax = \lambda Jx$ , where A is Hermitian positive-definite and J is a diagonal matrix with unit diagonal entries are given in [31, Section 4].

<sup>&</sup>lt;sup>2</sup> Here we have exchanged the roles of A and  $D^*AD$  compared to Theorem 3.1 in [21].

#### References

- J. Barlow, J. Demmel, Computing accurate eigensystems of scaled diagonally dominant matrices, SIAM J. Numer. Anal. 27 (1990) 762–791.
- [2] J. Barlow, I. Slapničar, Optimal perturbation bounds for the Hermitian eigenvalue problem, Technical Report 99-001, Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA, February 1999.
- [3] J. Barlow, P. Yoon, H. Zha, An algorithm and a stability theory for downdating the ULV decomposition, BIT 36 (1996) 14-40.
- [4] C. Davis, W. Kahan, Some new bounds on perturbation of subspaces, Bull. Amer. Math. Soc. 75 (1969) 863-868.
- [5] C. Davis, W. Kahan, The rotation of eigenvectors by a perturbation, III, SIAM J. Numer. Anal. 7 (1970) 1-46.
- [6] P. Deift, J. Demmel, L. Li, C. Tomei, The bidiagonal singular value decomposition and Hamiltonian mechanics, SIAM J. Numer. Anal. 28 (1991) 1463–1516.
- [7] J. Demmel, M. Gu, S. Eisenstat, I. Slapničar, K. Veselić, Z. Drmač, Computing the singular value decomposition with high relative accuracy, Linear Algebra Appl. 299 (1999) 21–80.
- [8] J. Demmel, K. Veselić, Jacobi's method is more accurate than QR, SIAM J. Matrix Anal. Appl. 13 (1992) 1204– 1245.
- [9] I. Dhillon, A New  $O(n^2)$  Algorithm for the Symmetric Tridiagonal Eigenvalue/Eigenvector Problem, Ph.D. thesis, Computer Science, University of California, Berkeley, CA, May 1997.
- [10] I. Dhillon, G. Fann, B. Parlett, in: M. Heath et al. (Eds.), Application of a new algorithm for the symmetric eigenproblem to computational quantum chemistry, in: Proceedings of the Eighth SIAM Conference on Parallel Processing for Scientific Computing, SIAM, Philadelphia, 1997.
- [11] S. Eisenstat, I. Ipsen, Relative perturbation bounds for eigenspaces and singular vector subspaces, in Applied Linear Algebra, SIAM, Philadelphia, 1994, pp. 62–65.
- [12] S. Eisenstat, I. Ipsen, Relative perturbation techniques for singular value problems, SIAM J. Numer. Anal. 32 (1995) 1972–1988.
- [13] S. Eisenstat, I. Ipsen, Relative perturbation results for eigenvalues and eigenvectors of diagonalisable matrices, BIT 38 (1998) 502–509.
- [14] S. Eisenstat, I. Ipsen, Three absolute perturbation bounds for matrix eigenvalues imply relative bounds, SIAM J. Matrix Anal. Appl. 20 (1998) 149–158.
- [15] I. Gohberg, R. Lancaster, L. Rodman, Invariant Subspaces of Matrices with Applications, Wiley, New York, 1986.
- [16] G. Golub, C. van Loan, Matrix Computations, 3rd Edition, The Johns Hopkins University Press, Baltimore, 1996.
- [17] M. Gu, S. Eisenstat, Relative perturbation theory for eigenproblems, Research Report YALEU/DCS/RR-934, Department of Computer Science, Yale University, 1993.
- [18] I. Ipsen, Absolute and relative perturbation bounds for invariant subspaces of matrices, Linear Algebra Appl. 309 (2000) 45–56.
- [19] I. Ipsen, Relative perturbation results for matrix eigenvalues and singular values, in: Acta Numerica 1998, Vol. 7, Cambridge University Press, Cambridge, 1998, pp. 151–201.
- [20] R. Li, Relative perturbation theory: I. eigenvalue variations, SIAM J. Matrix Anal. Appl. 19 (1998) 956-982.
- [21] R. Li, Relative perturbation theory: II. eigenspace and singular subspace variations, SIAM J. Matrix Anal. Appl. 20 (1999) 471–492.
- [22] T. Londrè, N. Rhee, A note on relative perturbation bounds, SIAM J. Matrix Anal. Appl. 21 (2000) 357-361.
- [23] R. Mathias, Fast accurate eigenvalue methods for graded positive-definite matrices, Numer. Math. 74 (1996) 85–103.
   [24] R. Mathias, A bound for matrix square root with application to eigenvector perturbation, SIAM J. Matrix Anal.
- Appl. 18 (1997) 861–867.
   [25] M. Matrix Spart root will application to eigenvector perturbation, STAIN J. Matrix Anal. Appl. 18 (1997) 861–867.
- [25] R. Mathias, Spectral perturbation bounds for positive definite matrices, SIAM J. Matrix Anal. Appl. 18 (1997) 959–980.
- [26] R. Mathias, K. Veselić, A relative perturbation bound for positive definite matrices, Linear Algebra Appl. 270 (1998) 315–321.
- [27] B. Parlett, Invariant subspaces for tightly clustered eigenvalues of tridiagonals, BIT 36 (1996) 542-562.
- [28] B. Parlett, I. Dhillon, Relatively robust representations of symmetric tridiagonals, Department of Mathematics, University of California, Berkeley, February 1999.

- [29] I. Slapničar, Accurate computation of singular values and eigenvalues of symmetric matrices, Math. Commun. 1 (1996) 153–167.
- [30] I. Slapničar, N. Truhar, Relative perturbation bound for invariant subspaces of graded indefinite matrices, Technical Report, University of Split and University of Osijek, Croatia, September 1998.
- [31] I. Slapničar, N. Truhar, Relative Perturbation theory for hyperbolic eigenvalue problem, tech. rep., University of Split and University of Osijek, Croatia, September 1998.
- [32] I. Slapničar, K. Veselić, Perturbations of the eigenprojections of a factorised Hermitian matrix, Linear Algebra Appl. 218 (1995) 273–280.
- [33] I. Slapničar, K. Veselić, A bound for the condition of a hyperbolic eigenvector matrix, Linear Algebra Appl. 290 (1999) 247–255.
- [34] G. Stewart, J. Sun, Matrix Perturbation Theory, Academic Press, San Diego, 1990.
- [35] N. Truhar, I. Slapničar, Relative perturbation of invariant subspaces, Math. Commun. 1 (1996) 169-174.
- [36] N. Truhar, I. Slapničar, Relative perturbation bound for invariant subspaces of graded indefinite Hermitian matrices, Linear Algebra Appl. 301 (1999) 171–185.
- [37] K. Veselić, I. Slapničar, Floating-point perturbations of Hermitian matrices, Linear Algebra Appl. 195 (1993) 81–116.



Journal of Computational and Applied Mathematics 123 (2000) 155-175

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.nl/locate/cam

# The trace minimization method for the symmetric generalized eigenvalue problem $\stackrel{\text{trace}}{\overset{\text{trace}}}{\overset{\text{trace}}{\overset{\text{trace}}{\overset{\text{trace}}}{\overset{\text{trace}}{\overset{\text{trace}}}{\overset{\text{trace}}{\overset{\text{trace}}}{\overset{\text{trace}}}{\overset{\text{trace}}}{\overset{\text{trace}}}{\overset{\text{trace}}{\overset{\text{trace}}}{\overset{\text{trace}}}{\overset{\text{trace}}}{\overset{\text{trace}}}{\overset{\text{trace}}}{\overset{\text{trace}}}{\overset{\text{trace}}}{\overset{\text{trace}}}{\overset{\text{trace}}}{\overset{\text{trace}}}{\overset{\text{trace}}}{\overset{\text{trace}}}{\overset{\text{trace}}}{\overset{\text{trace}}}{\overset{\text{trace}}}{\overset{\text{trace}}}{\overset{\text{trace}}}{\overset{\text{trace}}}}{\overset{\text{trace}}}{\overset{\text{trace}}}{\overset{\text{trace}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}}$

Ahmed Sameh\*, Zhanye Tong

Department of Computer Sciences, Purdue University, West Lafayette, IN 47907, USA

Received 9 June 1999; received in revised form 3 February 2000

#### Abstract

In this paper, the trace minimization method for the generalized symmetric eigenvalue problems proposed by Sameh and Wisniewski [35] is reviewed. Convergence of an inexact trace minimization algorithm is established and a variant of the algorithm that uses expanding subspaces is introduced and compared with the block Jacobi–Davidson algorithm. © 2000 Elsevier Science B.V. All rights reserved.

#### MSC: 65F15

Keywords: Trace minimization; Jacobi-Davidson scheme; Eigenvalue; Eigenvector

#### 1. Introduction

The generalized eigenvalue problem

 $Ax = \lambda Bx$ ,

(1.1)

where A and B are  $n \times n$  real symmetric matrices with B being positive definite, arises in many applications, most notably in structural mechanics [1,2] and plasma physics [17,19]. Usually, A and B are large, sparse, and only a few of the eigenvalues and the associated eigenvectors are desired. Because of the size of the problem, methods that rely only on operations like matrix-vector multiplications, inner products, and vector updates, that utilize only high-speed memory are usually considered.

Many methods fall into this category (see, for example [42,43]). The basic idea in all of these methods is building a sequence of subspaces that, in the limit, contain the desired eigenvectors. Most

<sup>&</sup>lt;sup>th</sup> This work was partially supported by NSF grant CCR-9619763.

<sup>\*</sup> Corresponding author.

E-mail addresses: sameh@cs.purdue.edu (A. Sameh), tong@cs.purdue.edu (Z. Tong).

of the early methods iterate on a single vector, i.e., using one-dimensional subspaces, to compute one eigenpair at a time. If several eigenpairs are needed, a deflation technique is frequently used. Another alternative is to use block analogs of the single vector methods to obtain several eigenpairs simultaneously. The well-known *simultaneous iteration* [31], or *subspace iteration* [28], is a block analog of the power method. Simultaneous iteration is originally developed by Bauer [3] under the name *treppeniteration*. It was extensively studied in the late 1960s and early 1970s [5,24,31,41,42].

Let A be symmetric positive definite and assume that the smallest p eigenpairs are the ones we desire to obtain, where  $1 \le p \le n$ . In simultaneous iteration, the sequence of subspaces of dimension p is generated by the following recurrence:

$$X_{k+1} = A^{-1}BX_k, \quad k = 0, 1, \dots,$$
(1.2)

where  $X_0$  is an  $n \times p$  matrix of full rank. The eigenvectors of interest are magnified at each iteration step, and will eventually dominate  $X_k$ . The downside of simultaneous iteration is that linear systems of the form Ax = b have to be solved repeatedly which is a significant challenge for large problems. Solving these linear systems inexactly often compromises global convergence. A variant of simultaneous iteration, called *the trace minimization method*, was proposed in 1982 by Sameh and Wisniewski [35] in an attempt to avoid this difficulty. Let  $X_k$  be the current approximation to the eigenvectors corresponding to the p smallest eigenvalues where  $X_k^T B X_k = I_p$ . The idea of the trace minimization scheme is to find a correction term  $\Delta_k$  that is *B*-orthogonal to  $X_k$  such that

$$\operatorname{tr}(X_k - \Delta_k)^{\mathrm{T}} A(X_k - \Delta_k) < \operatorname{tr}(X_k^{\mathrm{T}} A X_k).$$

It follows that, for any *B*-orthonormal basis  $X_{k+1}$  of the new subspace span $\{X_k - \Delta_k\}$ , we have

$$\operatorname{tr}(X_{k+1}^{\mathrm{T}}AX_{k+1}) < \operatorname{tr}(X_{k}^{\mathrm{T}}AX_{k}),$$

i.e., span{ $X_k - \Delta_k$ } gives rise to a better approximation of the desired eigenspace than span{ $X_k$ }. This trace reduction property can be maintained without solving any linear systems exactly.

Just as simultaneous iteration is accelerated by the use of Chebyshev polynomials, the trace minimization method is accelerated via shifting strategies. The introduction of shifts, however, may compromise the robustness of the trace minimization scheme. Various techniques have been developed to prevent *unstable convergence* (see Section 3.2 for details). A simple way to get around this difficulty is to utilize expanding subspaces. This, in turn, places the trace minimization method into a class of methods that includes the Lanczos method [23], Davidson's method [10], and the Jacobi–Davidson method [12,37,38].

The Lanczos method has become increasingly popular since the ground-breaking analysis by Paige [27], and many practical algorithms are known today [7,14,30,36] (see [8,15] for an overview). The original Lanczos algorithm was developed for handling the standard eigenvalue problem only, i.e., B=I. Extensions to the generalized eigenvalue problem [11,21,16] require solving a linear system of the form Bx=b at each iteration step, or factorizing matrices of the form  $A - \sigma B$  during the iteration. Davidson's method can be regarded as a preconditioned Lanczos method. It was intended to be a practical method for standard eigenvalue problems in quantum chemistry where the matrices involved are diagonally dominant. In the past two decades, Davidson's method has gone through a series of significant improvements [6,25,26,40,44]. A recent development is the Jacobi–Davidson method [38], published in 1996, which is a variant of Davidson's original scheme and the well-known Newton's method. The Jacobi–Davidson algorithm for the symmetric eigenvalue problem may be regarded as a

generalization of the trace minimization scheme that uses expanding subspaces. Both utilize an idea that dates back to Jacobi [20]. As we will see in Section 5, the current Jacobi–Davidson scheme can be further improved by the techniques developed in the trace minimization method.

In this paper, we give a detailed account of the trace minimization method including the derivation of the scheme, its convergence theory, acceleration techniques, and some implementation details. Some of this material is new. The outline of the paper is as follows. In Section 2, we "derive" the trace minimization method and describe the basic algorithm. In Section 3, we prove convergence of the basic algorithm under the assumption that the inner systems are solved inexactly. Shifting techniques are introduced in Section 4, and a Davidson-type generalization is given in Section 5.

Throughout the paper, the eigenpairs of the eigenvalue problem (1.1) are denoted by  $(x_i, \lambda_i)$ ,  $1 \le i \le n$ . The eigenvalues are always arranged in ascending order. The following eigenvalue problem of order 100:

$$A = \text{diag}(1 \times 0.1, 2 \times 0.2, 3 \times 0.3, \dots, 100 \times 10.0),$$
(1.3)

$$B = diag(0.1, 0.2, 0.3, \dots, 10.0),$$

will be used in Sections 2–5 to illustrate the techniques discussed in the paper. All numerical experiments for this small eigenvalue problem are performed with MATLAB on a SUN SPARC 5. The initial guesses are generated by the MATLAB function RAND, and the eigenpairs are accepted when the 2-norm of the residual vectors are less than  $10^{-10}$ . Numerical experiments for large problems are performed on SGI/Origin 2000. The results are presented in Section 5.3.

#### 2. The trace minimization method

In this section, we derive the trace minimization method originally presented in [35]. We assume that A is positive definite, otherwise problem (1.1) can be replaced by

$$(A - \mu B)x = (\lambda - \mu)Bx$$

with  $\mu < \lambda_1 < 0$ , that ensures a positive definite  $(A - \mu B)$ .

The trace minimization method is motivated by the following theorem.

**Theorem 2.1** (Beckenbach and Bellman [4], Sameh and Wisniewski [35]). Let A and B be as given in problem (1.1), and let  $X^*$  be the set of all  $n \times p$  matrices X for which  $X^TBX = I_p$ ,  $1 \le p \le n$ . Then

$$\min_{X \in X^*} \operatorname{tr}(X^{\mathrm{T}}AX) = \sum_{i=1}^{p} \lambda_i.$$
(2.1)

where  $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$  are the eigenvalues of problem (1.1). The equality holds if and only if the columns of the matrix X, which achieves the minimum, span the eigenspace corresponding to the smallest p eigenvalues.

If we denote by E/F the matrix  $EF^{-1}$  and  $\mathscr{X}$  the set of all  $n \times p$  matrices of full rank, then (2.1) is equivalent to

$$\min_{X\in\mathscr{X}} \operatorname{tr}\left(\frac{X^{\mathrm{T}}AX}{X^{\mathrm{T}}BX}\right) = \sum_{i=1}^{p} \lambda_{i}.$$

 $X^{T}AX/X^{T}BX$  is called the generalized Rayleigh quotient. Most of the early methods that compute a few of the smallest eigenvalues are devised explicitly or implicitly by reducing the generalized Rayleigh quotient step by step. A simple example is the simultaneous iteration scheme for a positive definite matrix A where the current approximation  $X_k$  is updated by (1.2). It can be shown by the Courant–Fischer theorem [29, p. 206] and the Kantorovic inequality [22] that

$$\lambda_i \left( \frac{X_{k+1}^{\mathrm{T}} A X_{k+1}}{X_{k+1}^{\mathrm{T}} B X_{k+1}} \right) \leqslant \lambda_i \left( \frac{X_k^{\mathrm{T}} A X_k}{X_k^{\mathrm{T}} B X_k} \right), \quad 1 \leqslant i \leqslant p.$$

$$(2.2)$$

The equality holds only when  $X_k$  is already an eigenspace of problem (1.1). Originally, the columns of  $X_{k+1}$  were taken as approximations to the desired eigenvectors. It was later found out that a Rayleigh–Ritz process on the subspace span  $\{X_{k+1}\}$  yields more accurate approximations. A detailed treatment of simultaneous iteration can be found in [29, Chapter 14]. The following is an outline of the basic algorithm:

#### Algorithm 1. Simultaneous iteration.

Choose a block size  $s \ge p$  and an  $n \times s$  matrix  $V_1$  of full rank such that  $V_1^T B V_1 = I_s$ .

For k = 1, 2, ... until convergence, do

- 1. Compute  $W_k = AV_k$  and the interaction matrix  $H_k = V_k^{\mathrm{T}}W_k$ .
- 2. Compute the eigenpairs  $(Y_k, \Theta_k)$  of  $H_k$ . The eigenvalues are arranged in ascending order and the eigenvectors are chosen to be orthogonal.
- 3. Compute the corresponding Ritz vectors  $X_k = V_k Y_k$ .
- 4. Compute the residuals  $R_k = W_k Y_k B X_k \Theta_k$ .
- 5. Test for convergence.
- 6. Solve the linear system

$$AZ_{k+1} = BX_k,$$

by an iterative method.

7. *B*-orthonormalize  $Z_{k+1}$  into  $V_{k+1}$  by the Gram–Schmidt process with reorthogonalization [9]. End for

In [35], simultaneous iteration was derived in a way that the trace minimization property is explicitly explored. At each iteration step, the previous approximation  $X_k$ , which satisfies  $X_k^T B X_k = I_s$  and  $X_k^T A X_k = \Theta_k$ , is corrected with  $\Delta_k$  that is obtained by

minimizing 
$$\operatorname{tr}(X_k - \Delta_k)^T A(X_k - \Delta_k)$$
,  
subject to  $X_k^T B \Delta_k = 0$ . (2.4)

(2.3)

As a result, the matrix  $Z_{k+1} = X_k - \Delta_k$  always satisfies

$$\operatorname{tr}(Z_{k+1}^{\mathrm{T}}AZ_{k+1}) \leq \operatorname{tr}(X_{k}^{\mathrm{T}}AX_{k}), \tag{2.5}$$

and

$$Z_{k+1}^{\mathrm{T}}BZ_{k+1} = I_s + \varDelta_k^{\mathrm{T}}B\varDelta_k,$$
(2.6)

which guarantee that

$$\operatorname{tr}(X_{k+1}^{\mathrm{T}}AX_{k+1}) \leq \operatorname{tr}(X_{k}^{\mathrm{T}}AX_{k})$$
(2.7)

for any *B*-orthonormal basis  $X_{k+1}$  of the subspace span $\{Z_{k+1}\}$ . The equality in (2.7) holds only when  $\Delta_k = 0$ , i.e.,  $X_k$  spans an eigenspace of (1.1) (see Theorem 3.3 for details).

Using Lagrange multipliers, the solution of the minimization problem (2.4) can be obtained by solving the saddle-point problem

$$\begin{bmatrix} A & BX_k \\ X_k^{\mathsf{T}}B & 0 \end{bmatrix} \begin{bmatrix} \Delta_k \\ L_k \end{bmatrix} = \begin{bmatrix} AX_k \\ 0 \end{bmatrix},$$
(2.8)

where  $2L_k$  represents the Lagrange multipliers. In [35], (2.8) is further reduced to the following positive-semidefinite system

$$(PAP)\Delta_k = PAX_k, \quad X_k^{\mathrm{T}}B\Delta_k = 0, \tag{2.9}$$

where *P* is the projector  $P = I - BX_k(X_k^T B^2 X_k)^{-1} X_k^T B$ . This system is solved by the conjugate gradient method (CG) in which zero is chosen as the initial iterate so that the linear constraint  $X_k^T B \Delta_k^{(l)} = 0$  is automatically satisfied for any intermediate  $\Delta_k^{(l)}$ . This results in the following basic trace minimization algorithm:

Algorithm 2. The basic trace minimization algorithm.

Choose a block size  $s \ge p$  and an  $n \times s$  matrix  $V_1$  of full rank such that  $V_1^T B V_1 = I_s$ . For k = 1, 2, ... until convergence, do

- 1. Compute  $W_k = AV_k$  and the interaction matrix  $H_k = V_k^T W_k$ .
- 2. Compute the eigenpairs  $(Y_k, \Theta_k)$  of  $H_k$ . The eigenvalues are arranged in ascending order and the eigenvectors are chosen to be orthogonal.
- 3. Compute the corresponding Ritz vectors  $X_k = V_k Y_k$ .
- 4. Compute the residuals  $R_k = AX_k BX_k\Theta_k = W_kY_k BX_k\Theta_k$ .
- 5. Test for convergence.
- 6. Solve the positive-semidefinite linear system (2.9) approximately via the CG scheme.

7. *B*-orthonormalize  $X_k - \Delta_k$  into  $V_{k+1}$  by the Gram–Schmidt process with reorthogonalization [9]. End for

From now on, we will refer to the linear system (2.9) in step (6) as the *inner system*(s). It is easy to see that the exact solution of the inner system is

$$\Delta_k = X_k - A^{-1} B X_k (X_k^{\mathrm{T}} B A^{-1} B X_k)^{-1}, \qquad (2.10)$$

thus the subspace spanned by  $X_k - \Delta_k$  is the same subspace spanned by  $A^{-1}BX_k$ . In other words, if the inner system (2.9) is solved exactly at each iteration step, the trace minimization algorithm above is mathematically equivalent to simultaneous iteration. As a consequence, global convergence of the basic trace minimization algorithm follows exactly from that of simultaneous iteration.

**Theorem 2.2** (Rutishauser [32], Parlett [29], Sameh and Wisniewski [35]). Let A and B be positive definite and let  $s \ge p$  be the block size such that the eigenvalues of problem (1.1) satisfy  $0 < \lambda_1 \le \lambda_2 \le \cdots \le \lambda_s < \lambda_{s+1} \le \cdots \le \lambda_n$ . Let also the initial iterate  $X_0$  be chosen such that it has linearly independent columns and is not deficient in any eigen-component associated with the p smallest eigenvalues. Then the ith column of  $X_k$ , denoted by  $x_{k,i}$ , converges to the eigenvector  $x_i$  corresponding to  $\lambda_i$  for i = 1, 2, ..., p with an asymptotic rate of convergence bounded by  $\lambda_i/\lambda_{s+1}$ . More specifically, at each step, the error

$$\phi_i = (x_{k,i} - x_i)^{\mathrm{T}} A(x_{k,i} - x_i)$$
(2.11)

is reduced asymptotically be a factor of  $(\lambda_i/\lambda_{s+1})^2$ .

The only difference between the trace minimization algorithm and simultaneous iteration is in step (6). If both (2.3) and (2.9) are solved via the CG scheme exactly, the performance of either algorithm is comparable in terms of time consumed, as observed in practice. The additional cost in performing the projection P at each CG step (once rather than twice) is not high because the block size s is usually small, i.e.,  $1 \le s \le n$ . This additional cost is sometimes compensated for by the fact that PAP, when it is restricted to subspace  $\{v \in \mathbb{R}^n | Pv = v\}$ , is better conditioned than A as will be seen in the following theorem.

**Theorem 2.3.** Let A and B be as given in Theorem 2.2 and P be as given in (2.9), and let  $v_i$ ,  $\mu_i$ ,  $1 \le i \le n$  be the eigenvalues of A and PAP arranged in ascending order, respectively. Then, we have

$$0 = \mu_1 = \mu_2 = \cdots = \mu_s < v_1 \leqslant \mu_{s+1} \leqslant \mu_{s+2} \leqslant \cdots \leqslant \mu_n \leqslant v_n$$

**Proof.** The proof is a straightforward consequence of the Courant–Fischer theorem [29, p. 206], and hence omitted.  $\Box$ 

#### 3. Practical considerations

In computing practice, however, the inner systems (2.3) and (2.9) are always solved approximately, particularly for large problems. There are two reasons for this: (i) the error (2.11) in the *i*th column of  $X_k$  is reduced asymptotically by a factor of  $(\lambda_i/\lambda_{s+1})^2$  at each iteration step. Thus we should not expect high accuracy in the early Ritz vectors even if the inner systems are solved to machine precision, and (ii) it is often too expensive to solve the inner systems to high-order accuracy by an iterative method. Numerical experiments have shown that, for simultaneous iteration, the inner system (2.3) has to be solved in a progressive way, i.e., the absolute stopping tolerance for the inner systems must be decreasing such that it is smaller than the specified error tolerance at the end of the outer iteration. On the contrary, for the trace minimization algorithm, the convergence is guaranteed if a constant relative residual tolerance is used for the inner system (2.9). Table 1 shows the behavior of both algorithms for example (1.3), where  $\times$  indicates stagnation.

#### 3.1. A convergence result

In this section, we prove the convergence of the trace minimization algorithm under the assumption that the inner systems in (2.9) are solved inexactly. We assume that, for each *i*,  $1 \le i \le s$ , the *i*th inner system in (2.9) is solved approximately by the CG scheme with zero as the initial iterate such that the 2-norm of the residual is reduced by a factor  $\gamma < 1$ . The computed correction matrix will be

```
Table 1
```

The basic trace minimization algorithm (Algorithm 2) versus simultaneous iteration. The inner systems are solved by the CG scheme which is terminated such that the 2-norm of the residual is reduced by a specified factor. The number of outer iterations (#its) and the number of matrix vector multiplications with A (A mults) are listed for different residual reduction factors

Methods	$10^{-4}$		$10^{-2}$		0.5		Dynamic	
	#its	A mults	#its	A mults	#its	A mults	#its	A mults
Simult	×		×		×		×	
Tracmn	59	6638	59	4263	77	4030	66	4479

denoted by  $\Delta_k^c = \{d_{k,1}^c, d_{k,2}^c, \dots, d_{k,s}^c\}$  to distinguish it from the exact solution  $\Delta_k = \{d_{k,1}, d_{k,2}, \dots, d_{k,s}\}$  of (2.9).

We begin the convergence proof with two lemmas. We first show that, at each iteration step, the columns of  $X_k - \Delta_k^c$  are linearly independent, and the sequence  $\{X_k\}_0^\infty$  in the trace minimization algorithm is well-defined. In Lemma 3.2, we show that the computed correction matrix  $\Delta_k^c$  satisfies

$$\operatorname{tr}(X_k - \Delta_k^{\operatorname{c}})^{\operatorname{T}} A(X_k - \Delta_k^{\operatorname{c}}) \leq \operatorname{tr}(X_k^{\operatorname{T}} A X_k).$$

This assures that, no matter how prematurely the CG process is terminated, the trace tr( $X_k^T A X_k$ ) always forms a decreasing sequences bounded from below by  $\sum_{i=1}^s \lambda_i$ .

**Lemma 3.1.** For each  $k = 0, 1, 2, ..., Z_{k+1} = X_k - \Delta_k^c$  is of full rank.

**Proof.** Since  $d_{k,i}^{c}$  is an intermediate approximation obtained from the CG process, there exists a polynomial p(t) such that

$$d_{ki}^{c} = p(PAP)(PAx_{ki}),$$

where  $x_{k,i}$  is the *i*th column of  $X_k$  and P is the projector in (2.9). As a consequence, for each *i*,  $d_{k,i}^c$  is *B*-orthogonal to  $X_k$ , i.e.,  $X_k^T B d_{k,i}^c = 0$ . Thus the matrix

$$Z_{k+1}^{\mathrm{T}}BZ_{k+1} = I_s + \varDelta_k^{\mathrm{cT}}B\varDelta_k^{\mathrm{c}}$$

is nonsingular, and  $Z_{k+1}$  is of full rank.  $\Box$ 

**Lemma 3.2.** Suppose that the inner systems in (2.9) are solved by the CG scheme with zero as the initial iterate. Then, for each i,  $(x_{k,i} - d_{k,i}^{(l)})^T A(x_{k,i} - d_{k,i}^{(l)})$  decreases monotonically with respect to step l of the CG scheme.

**Proof.** The exact solution of the inner system (2.9),

$$\Delta_k = X_k - A^{-1}BX_k(X_k^{\mathrm{T}}BA^{-1}BX_k)^{-1}$$

satisfies  $P \Delta_k = \Delta_k$ . For each *i*,  $1 \le i \le s$ , the intermediate  $d_{k,i}^{(l)}$  in the CG process also satisfies  $P d_{k,i}^{(l)} = d_{k,i}^{(l)}$ . It follows that

$$(d_{k,i}^{(l)} - d_{k,i})^{\mathrm{T}} PAP(d_{k,i}^{(l)} - d_{k,i}) = (d_{k,i}^{(l)} - d_{k,i})^{\mathrm{T}} A(d_{k,i}^{(l)} - d_{k,i})$$
$$= (x_{k,i} - d_{k,i}^{(l)})^{\mathrm{T}} A(x_{k,i} - d_{k,i}^{(l)}) - [(X_k^{\mathrm{T}} BA^{-1} BX_k)^{-1}]_{ii}.$$

Since the CG process minimizes the *PAP*-norm of the error  $e_{k,i}^{(l)} = d_{k,i}^{(l)} - d_{k,i}$  on the expanding Krylov subspace [33, p. 130], we have  $(d_{k,i}^{(l)} - d_{k,i})^{\mathrm{T}} PAP(d_{k,i}^{(l)} - d_{k,i})$  decreases monotonically. So does  $(x_{k,i} - d_{k,i}^{(l)})^{\mathrm{T}} A(x_{k,i} - d_{k,i}^{(l)})$ .  $\Box$ 

**Theorem 3.3.** Let  $X_k$ ,  $\Delta_k^c$ , and  $Z_{k+1}$  be as given for Lemma 3.1. Then we have  $\lim_{k\to\infty} \Delta_k^c = 0$ .

**Proof.** First, by the definition of  $\Delta_k^c$ , we have

$$Z_{k+1}^{\mathrm{T}}BZ_{k+1} = I_s + \varDelta_k^{\mathrm{cT}}B\varDelta_k^{\mathrm{c}} \triangleq I_s + T_k$$

Consider the spectral decomposition of  $Z_{k+1}^{T}BZ_{k+1}$ 

$$Z_{k+1}^{\mathrm{T}}BZ_{k+1} = U_{k+1}D_{k+1}^{2}U_{k+1}^{\mathrm{T}},$$

where  $U_{k+1}$  is an  $s \times s$  orthogonal matrix and  $D_{k+1}^2 = \text{diag}(\delta_1^{(k+1)}, \delta_2^{(k+1)}, \dots, \delta_s^{(k+1)})$ . It is easy to see that  $\delta_i^{(k+1)} = 1 + \lambda_i(T_k) \ge 1$ .

Second, by the definition of  $X_{k+1}$ , there exists an orthogonal matrix  $V_{k+1}$  such that

$$X_{k+1} = Z_{k+1} \cdot U_{k+1} D_{k+1}^{-1} V_{k+1}$$

Denote by  $z_i^{(k+1)}$  the diagonal elements of the matrix  $U_{k+1}^T Z_{k+1}^T A Z_{k+1} U_{k+1}$ . It follows that

$$\operatorname{tr}(X_{k+1}^{\mathrm{T}}AX_{k+1}) = \operatorname{tr}(D_{k+1}^{-1}(U_{k+1}^{\mathrm{T}}Z_{k+1}^{\mathrm{T}}AZ_{k+1}U_{k+1})D_{k+1}^{-1})$$

$$= \frac{z_{1}^{(k+1)}}{\delta_{1}^{(k+1)}} + \frac{z_{2}^{(k+1)}}{\delta_{2}^{(k+1)}} + \dots + \frac{z_{s}^{(k+1)}}{\delta_{s}^{(k+1)}},$$

$$\leqslant z_{1}^{(k+1)} + z_{2}^{(k+1)} + \dots + z_{s}^{(k+1)},$$

$$= \operatorname{tr}(Z_{k+1}^{\mathrm{T}}AZ_{k+1}),$$

$$\leqslant \operatorname{tr}(X_{k}^{\mathrm{T}}AX_{k}),$$

which implies that

$$\cdots \geq \operatorname{tr}(X_k^{\mathrm{T}}AX_k) \geq \operatorname{tr}(Z_{k+1}^{\mathrm{T}}AZ_{k+1}) \geq \operatorname{tr}(X_{k+1}^{\mathrm{T}}AX_{k+1}) \geq \cdots.$$

Since the sequence is bounded from below by  $\sum_{i=1}^{s} \lambda_i$ , it converges to a positive number  $t \ge \sum_{i=1}^{s} \lambda_i$ . Moreover, the two sequences

$$\frac{z_1^{(k+1)}}{\delta_1^{(k+1)}} + \frac{z_2^{(k+1)}}{\delta_2^{(k+1)}} + \dots + \frac{z_s^{(k+1)}}{\delta_s^{(k+1)}}, \quad k = 1, 2, \dots$$

and

$$z_1^{(k+1)} + z_2^{(k+1)} + \cdots + z_s^{(k+1)}, \quad k = 1, 2, \dots$$

also converge to t. Therefore,

$$\left(\frac{z_1^{(k+1)}\lambda_1(T_k)}{1+\lambda_1(T_k)}\right) + \left(\frac{z_2^{(k+1)}\lambda_2(T_k)}{1+\lambda_2(T_k)}\right) + \dots + \left(\frac{z_s^{(k+1)}\lambda_s(T_k)}{1+\lambda_s(T_k)}\right) \to 0.$$

Observing that for any  $i, 1 \leq i \leq s$ ,

$$\begin{split} z_{i}^{(k+1)} &\geq \lambda_{1}(U_{k+1}^{\mathrm{T}}Z_{k+1}^{\mathrm{T}}AZ_{k+1}U_{k+1}), \\ &= \lambda_{1}(Z_{k+1}^{\mathrm{T}}AZ_{k+1}), \\ &= \min_{y \neq 0} \frac{y^{\mathrm{T}}Z_{k+1}^{\mathrm{T}}AZ_{k+1}y}{y^{\mathrm{T}}y}, \\ &= \min_{y \neq 0} \left(\frac{y^{\mathrm{T}}Z_{k+1}^{\mathrm{T}}AZ_{k+1}y}{y^{\mathrm{T}}Z_{k+1}^{\mathrm{T}}BZ_{k+1}y}\right) \cdot \left(\frac{y^{\mathrm{T}}Z_{k+1}^{\mathrm{T}}BZ_{k+1}y}{y^{\mathrm{T}}y}\right), \\ &\geq \min_{y \neq 0} \frac{y^{\mathrm{T}}Z_{k+1}^{\mathrm{T}}AZ_{k+1}y}{y^{\mathrm{T}}Z_{k+1}^{\mathrm{T}}BZ_{k+1}y}, \\ &\geq \lambda_{1}(A, B), \\ &> 0, \end{split}$$

we have

 $\lambda_1(T_k) \to 0, \quad i = 1, 2, \dots, s,$ i.e.,  $\lim_{k \to \infty} \Delta_k^c = 0.$ 

## **Theorem 3.4.** If, for each $i, 1 \le i \le s$ , the CG process for the *i*th inner system

 $(PAP)d_{k,i} = PAx_{k,i}, \quad d_{k,i}^{\mathrm{T}}BX_k = 0,$ 

in (2.9) is terminated such that the 2-norm of the residual is reduced by a factor  $\gamma < 1$ , i.e.,

$$||PAx_{k,i} - (PAP)d_{k,i}^{c}||_{2} \leq \gamma ||PAx_{k,i}||_{2},$$
(3.1)

then columns of  $X_k$  converge to s eigenvectors of problem (1.1).

**Proof.** Condition (3.1) implies that

 $||PAx_{k,i}||_2 - ||PAd_{k,i}^{c}||_2 \leq \gamma ||PAx_{k,i}||_2,$ 

and consequently

$$||PAx_{k,i}||_2 \leq \frac{1}{1-\gamma} ||PAd_{k,i}^{c}||_2.$$

It follows from Theorem 3.3 that  $\lim_{k\to\infty} PAX_k = 0$ , i.e.,

 $\lim_{k\to\infty} (AX_k - BX_k[(X_k^{\mathrm{T}}B^2X_k)^{-1}X_k^{\mathrm{T}}BAX_k]) = 0.$ 

This shows that  $span\{X_k\}$  converges to an eigenspace of problem (1.1).  $\Box$ 

#### 3.2. Randomization

Condition (3.1) in Theorem 3.4 is not essential because the constant  $\gamma$  can be arbitrarily close to 1. The only deficiency in Theorem 3.4 is that it does not establish ordered convergence in the sense that the *i*th column of  $X_k$  converges to the *i*th eigenvector of the problem. This is called *unstable* 

convergence by Rutishauser. In computing practice, roundoff errors usually turn unstable convergence into delayed stable convergence. In [32], Rutishauser introduced a randomization technique to prevent unstable convergence in simultaneous iteration; it can be incorporated into the trace minimization algorithm as well: After step (6) of Algorithm 2, we append a random vector to  $X_k$  and perform the Ritz processes (1)–(2) on the augmented subspace of dimension s + 1. The extra Ritz pair is discarded after step (2).

Randomization slightly improves the convergence of the first s Ritz pairs [31]. Since it comes with additional cost, it should be used only in the first few steps and when a Ritz pair is about to converge.

#### 3.3. Terminating the CG process

Theorem 3.4 gives a sufficient condition for the convergence of the trace minimization algorithm. However, the asymptotic rate of convergence of the trace minimization algorithm will be affected by the premature termination of the CG processes. Table 3.1 shows how differently the trace minimization algorithm behaves when the inner systems are solved inexactly. It is not clear how the parameter  $\gamma$  should be chosen to avoid performing excessive CG iterations while maintaining the asymptotic rate of convergence. In [35], the CG processes are terminated by a heuristic stopping strategy.

Denote by  $d_{k,i}^{(l)}$  the approximate solution at the *l*th step of the CG process for the *i*th column of  $X_k$ , and  $d_{k,i}$  the exact solution. The heuristic stopping strategy in [35] can be outlined as follows:

1. From Theorem 2.2, it is reasonable to terminate the CG process for the *i*th column of  $\Delta_k$  when the error

$$\varepsilon_{k,i}^{(l)} = [(d_{k,i}^{(l)} - d_{k,i})^{\mathrm{T}} A (d_{k,i}^{(l)} - d_{k,i})]^{1/2}$$

is reduced by a factor of  $\tau_i = \lambda_i / \lambda_{s+1}$ , called *error reduction factor*. 2. The quantity  $\varepsilon_{k,i}^{(l)}$  can be estimated by

$$[(d_{k,i}^{(l)} - d_{k,i}^{(l+1)})^{\mathrm{T}} A(d_{k,i}^{(l)} - d_{k,i}^{(l+1)})]^{1/2},$$

which is readily available from the CG process.

3. The error reduction factor  $\tau_i = \lambda_i / \lambda_{s+1}$ ,  $1 \leq i \leq s$ , can be estimated by  $\tau_{k,i} = \theta_{k,i} / \theta_{k,s+1}$ . Since  $\theta_{k,s+1}$ is not available,  $\theta_{k-1,s}$  is used instead and is fixed after a few steps because it will eventually converge to  $\lambda_s$  rather than  $\lambda_{s+1}$ .

This strategy has worked well in practice. The last column of Table 1 shows the result obtained with this stopping strategy.

#### 4. Acceleration techniques

The algorithm discussed in Section 3 effectively reduces the work at each iteration step. It requires, however, about the same number of outer iteration steps as the simultaneous iteration. For problems in which the desired eigenvalues are poorly separated from the remaining part of the spectrum, the algorithm converges too slowly. Like other inverse iteration schemes, the trace minimization

Safe shift		Single sh	ift	Multiple	Multiple shifts		
#its 46	A mults 4153	#its 22	A mults 3619	#its 18	<i>A</i> mults 3140		

 Table 2

 The trace minimization algorithm with various shifting strategies

algorithm can be accelerated by shifting. Actually, the formulation of the trace minimization algorithm makes it easier to incorporate shifts. For example, if eigenpairs  $(x_i, \theta_i)$ ,  $1 \le i \le i_0$ , have been accepted and  $\theta_{i_0} < \theta_{i_0+1}$ ,  $\theta_{i_0}$  can be used as a shift parameter for computing subsequent eigenpairs. Due to the deflation effect, the linear systems

$$[P(A - \theta_{i_0}B)P]d_{k,i} = PAx_{k,i}, \quad X_k^{\mathrm{T}}Bd_{k,i} = 0, \quad i_0 + 1 \leq i \leq s,$$

are consistent and can still be solved by the CG scheme. Moreover, the trace reduction property still holds. The first column of Table 2 shows the result of the trace minimization scheme with such a conservative shifting strategy, which we call *safe shift*. The performance is obviously improved over that of the basic trace minimization algorithm shown in Table 1. In the following, we introduce two more efficient shifting techniques which improve further the performance of the trace minimization algorithm.

#### 4.1. Single shift

We know from Section 2 that global convergence of the trace minimization algorithm follows from the monotonic reduction of the trace, which in turn depends on the positive definiteness of A. A simple and robust shifting strategy would be finding a scalar  $\sigma$  close to  $\lambda_1$  from below and replace A with  $A - \sigma B$  in step (6) of the algorithm. After the first eigenvector is converged, find another  $\sigma$  close to  $\lambda_2$  from below and continue until all the desired eigenvectors are obtained. If both A and B are explicitly available, it is not hard to find a  $\sigma$  satisfying  $\sigma \leq \lambda_1$ . Gerschgorin disks [13], for example, provide reliable bounds on the spectrum of (1.1). These bounds, however, are usually too loose to be useful.

In the trace minimization algorithm, the subspace spanned by  $X_k$  converges to the invariant subspace  $V_s$  corresponding to the *s* smallest eigenvalues. If the subspace spanned by  $X_k$  is close enough to  $V_s$ , a reasonable bound for the smallest eigenvalue can be obtained. More specifically, let Q be a *B*-orthonormal matrix obtained by appending n - s columns to  $X_k$ , i.e.,  $Q = (X_k, Y_k)$  and  $Q^T B Q = I_n$ . Then problem (1.1) is reduced to the standard eigenvalue problem

$$(Q^{\mathrm{T}}AQ)u = \lambda u. \tag{4.1}$$

Since

$$Q^{\mathrm{T}}AQ = \begin{bmatrix} \Theta_k & X_k^{\mathrm{T}}AY_k \\ Y_k^{\mathrm{T}}AX_k & Y_k^{\mathrm{T}}AY_k \end{bmatrix} = \begin{bmatrix} \Theta_k & C_k^{\mathrm{T}} \\ C_k & Y_k^{\mathrm{T}}AY_k \end{bmatrix},$$
(4.2)

by the Courant-Fischer theorem, we have

$$egin{aligned} \lambda_1 &\geqslant \lambda_{\min} \begin{bmatrix} egin{aligned} \Theta_k & 0 \ 0 & Y_k^{\mathrm{T}} A Y_k \end{bmatrix} + \lambda_{\min} \begin{bmatrix} 0 & C_k^{\mathrm{T}} \ C_k & 0 \end{bmatrix} \ &\geqslant \min\{ heta_1, \lambda_1(Y_k^{\mathrm{T}} A Y_k)\} - ||C_k||_2. \end{aligned}$$

Similar to [29, p. 241], it is easy to derive  $||C_k||_2 = ||R_k||_{B^{-1}}$ , in which  $R_k = AX_k - BX_k\Theta_k$  is the residual matrix. If

$$\theta_{k,1} \leqslant \lambda_1 (Y_k^{\mathrm{T}} A Y_k), \tag{4.3}$$

we get

$$\lambda_1 \ge \theta_{k,1} - ||R_k||_{B^{-1}}. \tag{4.4}$$

In particular, if (4.3) holds for the orthonormal complement of  $x_{k,1}$ , we have

$$\lambda_1 \ge \theta_{k,1} - ||r_{k,1}||_{B^{-1}}. \tag{4.5}$$

This heuristic bound for the smallest eigenvalue suggests the following shifting strategy (we denote  $-\infty$  by  $\lambda_0$ ):

If the first  $i_0$ ,  $i_0 \ge 0$ , eigenvalues have converged, use  $\sigma = \max\{\lambda_{i_0}, \theta_{k,i_0+1} - ||r_{k,i_0+1}||_{B^{-1}}\}$  as the shift parameter. If  $\theta_{k,i_0+1}$  lies in a cluster, replace  $r_{k,i_0+1}$  by the residual matrix corresponding to the cluster containing  $\theta_{k,i_0+1}$ .

#### 4.2. Multiple dynamic shifts

In [35], the trace minimization algorithm is accelerated with a more aggressive shifting strategy. At the beginning of the algorithm, a single shift is used for all the columns of  $X_k$ . As the algorithm proceeds, multiple shifts are introduced dynamically and the CG process is modified to handle possible breakdown. This shifting strategy is motivated by the following theorem.

**Theorem 4.1** (Parlett [29, p. 357]). For an arbitrary nonzero vector u and scalar  $\sigma$ , there is an eigenvalue  $\lambda$  of (1.1) such that

$$|\lambda - \sigma| \leq ||(A - \sigma B)u||_{B^{-1}}/||Bu||_{B^{-1}}.$$

We know from the Courant–Fischer theorem that the targeted eigenvalue  $\lambda_i$  is always below the Ritz value  $\theta_{k,i}$ . Further, from Theorem 4.1, if  $\theta_{k,i}$  is already very close to the targeted eigenvalue  $\lambda_i$ , then  $\lambda_i$  must lie in the interval  $[\theta_{k,i} - ||r_{k,i}||_{B^{-1}}, \theta_{k,i}]$ . This observation leads to the following shifting strategy for the trace minimization algorithm. At step k of the outer iteration, the shift parameters  $\sigma_{k,i}$ ,  $1 \le i \le s$ , are determined by the following rules (Here,  $\lambda_0 = -\infty$  and the subscript k is dropped for the sake of simplicity):

1. If the first  $i_0$ ,  $i_0 \ge 0$ , eigenvalues have converged, choose

$$\sigma_{k,i_0+1} = \begin{cases} \theta_{i_0+1} & \text{if } \theta_{i_0+1} + ||r_{i_0+1}||_{B^{-1}} \leqslant \theta_{i_0+2} - ||r_{i_0+2}||_{B^{-1}}, \\ \max\{\theta_{i_0+1} - ||r_{i_0+1}||_{B^{-1}}, \lambda_{i_0}\} & \text{otherwise.} \end{cases}$$

166

2. For any other column j,  $i_0 + 1 < j \le p$ , choose the largest  $\theta_l$  such that

$$\theta_l < \theta_j - ||r_j||_{B^{-1}}$$

as the shift parameter  $\sigma_j$ . If no such  $\theta_l$  exists, use  $\theta_{i_0+1}$  instead.

3. Choose  $\sigma_i = \theta_i$  if  $\theta_{i-1}$  has been used as the shift parameter for column i-1 and

$$\theta_i < \theta_{i+1} - ||r_{i+1}||_{B^{-1}}.$$

4. Use  $\sigma_{i_0+1}$  as the shift parameters for other columns if any.

This heuristic shifting strategy turns out to be quite efficient and robust in practice. Table 2 shows the results for the shifting strategies discussed in this section. Since A is positive definite, zero is a good shift parameter. In our experiments, however, we did not take advantage of this fact and selected the shift parameters according to the strategies described above with  $B^{-1}$ -norms replaced by 2-norms. We see that both the number of outer iteration steps and the number of matrix vector multiplications with A are reduced considerably by the multiple dynamic shifting strategy. The number of matrix vector multiplications with B is not shown in the table because it is almost identical to that with A.

#### 4.3. Solving the inner systems

With multiple shifts, the inner systems in (2.9) become

$$[P(A - \sigma_{k,i}B)P]d_{k,i} = PAx_{k,i}, \quad X_k^{\mathrm{T}}Bd_{k,i} = 0, \quad 1 \le i \le s$$

$$(4.6)$$

with  $P = I - BX_k(X_k^T B^2 X_k)^{-1} X_k^T B$ . Clearly, the linear systems can be indefinite, and the CG processes for such systems are numerically unstable and may break down. A simple way to get around this problem is terminating the CG process when a near breakdown is detected. In [35], the CG process is also terminated when the error  $(x_{k,i} - d_{k,i}^{(l)})^T A(x_{k,i}^{(l)} - d_{k,i}^{(l)})$ , increases by a small factor. This helps maintain global convergence which is not guaranteed in the presence of shifting.

Due to the deflation effect, the inner systems in (4.6) are usually not ill-conditioned when restricted to the subspace  $\{v \in \mathbb{R}^n | Pv=v\}$  unless some of the gap ratios  $(\lambda_{s+1}-\lambda_i)/(\lambda_n-\lambda_i)$ ,  $1 \le i \le p$ , are small. In this case, the inner systems have to be preconditioned. Suppose  $\hat{A} = CC^T$  is a symmetric positive definite preconditioner of  $A - \sigma_{k,i}B$  (for example, an approximate incomplete Cholesky factorization of  $A - \sigma_{k,i}B$ ). The *i*th indefinite system in (4.6) can be written as

$$[\tilde{P}(\tilde{P} - \sigma_{k,i}\tilde{B})\tilde{P}]\tilde{d}_{k,i} = \tilde{P}\tilde{P}\tilde{x}_{k,i}, \quad \tilde{X}_{k}^{\mathrm{T}}\tilde{B}\tilde{d}_{k,i} = 0,$$
(4.7)

with

$$\tilde{A} = C^{-1}AC^{-T}, \quad \tilde{B} = C^{-1}BC^{-T}, \quad \tilde{d}_{k,i} = C^{T}d_{k,i}, \quad \tilde{X}_{k} = C^{T}X_{k}, \quad \tilde{x}_{k,i} = C^{T}x_{k,i},$$

and

$$\tilde{P} = I - \tilde{B}\tilde{X}_k(\tilde{X}_k^{\mathrm{T}}\tilde{B}^2\tilde{X}_k)^{-1}\tilde{X}_k^{\mathrm{T}}\tilde{B}.$$

Since it is usually difficult to construct a symmetric positive-definite preconditioner for a symmetric indefinite matrix, we suggest that a fixed preconditioner be used for all the matrices  $A - \sigma_{k,i}B$ .

In the presence of shifting, the asymptotic error reduction factor for the *i*th Ritz vector becomes  $(\lambda_i - \sigma_{k,i})/(\lambda_{s+1} - \sigma_{k,i})$ . As a consequence, the CG process is now terminated when the error  $\varepsilon_{k,i}^{(l)} = [(d_{k,i}^{(l)} - d_{k,i})^T (A - \sigma_{k,i}B)(d_{k,i}^{(l)} - d_{k,i})]^{1/2}$ 

is reduced by a factor of

$$\tau_{i} = \begin{cases} (\theta_{k,i} - \sigma_{k,i})/(\theta_{k,s+1} - \sigma_{k,i}), & \theta_{k,i} \neq \theta_{k,i}, \\ (\theta_{k-1,i} - \sigma_{k,i})/(\theta_{k,s+1} - \sigma_{k,i}), & \theta_{k,i} = \theta_{k,i} \end{cases}$$
(4.8)

and  $\theta_{k,s+1}$  is estimated as in Section 3.3. In practice, we have terminated the CG process when the 2-norm of the residual is reduced by a factor of  $\tau_i$ .

#### 5. A Davidson-type generalization

The shifting strategies described in Section 4 improve the performance of the trace minimization algorithm considerably. Although the randomization technique, the shifting strategy, and the roundoff error actually make the algorithm surprisingly robust for a variety of problems, further measures to guard against unstable convergence are necessary for problems in which the desired eigenvalues are clustered. A natural way to maintain stable convergence is by using expanding subspaces, with which the trace reduction property is automatically maintained.

The best-known method that utilizes expanding subspaces is that of Lanczos. It uses the Krylov subspaces to compute an approximation of the desired eigenpairs, usually the largest. This idea was adopted by Davidson, in combination with the simultaneous coordinate relaxation method, to obtain what he called the "compromise method" [10], known as Davidson's method today. In this section, we generalize the trace minimization algorithm described in the previous sections by casting it into the framework of the Davidson method. We start by the Jacobi–Davidson method, explore its connection to the trace minimization method, and develop a Davidson-type trace minimization algorithm.

#### 5.1. The Jacobi–Davidson method

As was mentioned in Section 1, the Jacobi–Davidson scheme is a modification of the Davidson method. It uses the same ideas presented in the trace minimization method to compute a correction term to a previous computed Ritz pair, but with a different objective. In the Jacobi–Davidson method, for a given Ritz pair  $(x_i, \theta_i)$  with  $x_i^T B x_i = 1$ , a correction vector  $d_i$  is sought such that

$$A(x_{i} + d_{i}) = \lambda_{i}B(x_{i} + d_{i}), \quad x_{i}^{T}Bd_{i} = 0,$$
(5.1)

where  $\lambda_i$  is the eigenvalue targeted by  $\theta_i$ . Since the targeted eigenvalue  $\lambda_i$  is not available during the iteration, it is replaced by an approximation  $\sigma_i$ . Ignoring high-order terms in (5.1), we get

$$\begin{bmatrix} A - \sigma_i B \ B x_i \\ x_i^{\mathrm{T}} B \ 0 \end{bmatrix} \begin{bmatrix} d_i \\ l_i \end{bmatrix} = \begin{bmatrix} -r_i \\ 0 \end{bmatrix},$$
(5.2)

where  $r_i = Ax_i - \theta_i Bx_i$  is the residual vector associated with the Ritz pair  $(x_i, \theta_i)$ . Note that replacing  $r_i$  with  $Ax_i$  does not affect  $d_i$ . In [37,38], the Ritz value  $\theta_i$  is used in place of  $\sigma_i$  at each step. A block Jacobi–Davidson algorithm, described in [37], is outlined as follows:

#### Algorithm 3. The block Jacobi–Davidson algorithm.

Choose a block size  $s \ge p$  and an  $n \times s$  matrix  $V_1$  such that  $V_1^T B V_1 = I_s$ . For k = 1, 2, ... until convergence, do

- 1. Compute  $W_k = AV_k$  and the interaction matrix  $H_k = V_k^{\mathrm{T}} W_k$ .
- 2. Compute the s smallest eigenpairs  $(Y_k, \Theta_k)$  of  $H_k$ . The eigenvalues are arranged in ascending order and the eigenvectors are chosen to be orthogonal.
- 3. Compute the corresponding Ritz vectors  $X_k = V_k Y_k$ .
- 4. Compute the residuals  $R_k = W_k Y_k B X_k \Theta_k$ .
- 5. Test for convergence.
- 6. for  $1 \leq i \leq s$ , solve the indefinite system

$$\begin{bmatrix} A - \theta_i B \ B x_{k,i} \\ x_{k,i}^{\mathrm{T}} B \ 0 \end{bmatrix} \begin{bmatrix} d_{k,i} \\ l_{k,i} \end{bmatrix} = \begin{bmatrix} r_{k,i} \\ 0 \end{bmatrix},$$
(5.3)

or preferably its projected form

$$[P_i(A - \theta_{k,i}B)P_i]d_{k,i} = P_i r_{k,i}, \quad x_{k,i}^{\mathrm{T}}Bd_{k,i} = 0,$$
(5.4)

approximately, where  $P_i = I - Bx_{k,i}(x_{k,i}^T B^2 x_{k,i})^{-1} x_{k,i}^T B$  is an orthogonal projector, and  $r_{k,i} = Ax_{k,i} - \theta_{k,i}Bx_{k,i}$  is the residual corresponding to the Ritz pair  $(x_{k,i}, \theta_{k,i})$ .

7. If  $\dim(V_k) \leq m - s$ , then

$$V_{k+1} = \operatorname{Mod} GS_B(V_k, \Delta_k),$$

else

$$V_{k+1} = \operatorname{Mod} GS_B(X_k, \Delta_k).$$

Here, Mod  $GS_B$  stands for the Gram–Schmidt process with reorthogonalization [9] with respect to *B*-inner products, i.e.  $(x, y) = x^T B y$ .

End for

This algorithm can be regarded as a trace minimization algorithm with expanding subspaces. The performance of the block Jacobi–Davidson algorithm depends on how good the initial guess is and how efficiently and accurately the inner system (5.3) is solved.

If the right-hand side of (5.3) is taken as the approximate solution to the inner system (5.3), the algorithm is reduced to the Lanczos method. If the inner system (5.3) is solved to high-order accuracy, it is reduced to simultaneous Rayleigh quotient iteration (RQI, see [28]) with expanding subspaces, which converges cubically. If the inner system (5.3) is solved crudely, the performance of the algorithm is in-between. Cubic convergence has been observed for some test problems [38]. In practice, however, the stage of cubic convergence is often reached after many iterations. Fig. 1 shows the convergence history of the block Jacobi–Davidson algorithm for the sample problem (1.3), where four eigenpairs are computed with m=20 and only the errors in the first Ritz value are plotted. The algorithm always "stagnates" at the beginning and increasing the number of iteration steps for the inner systems makes little difference or, in some cases, even derails convergence to the desired eigenpairs. This can be explained by the following. On the one hand, the Ritz shifting strategy in the block Jacobi–Davidson algorithm to converge to eigenvalues closest to the Ritz values that are often far away from the desired eigenvalues at the beginning of the iteration. On



Fig. 1. The block Jacobi-Davidson algorithm.

the other hand, since the subspace is expanding, the Ritz values are decreasing and the algorithm is forced to converge to the smallest eigenpairs.

Another problem with the block Jacobi–Davidson algorithm is ill-conditioning. At the end of the Jacobi–Davidson iteration, when a Ritz value approaches a multiple eigenvalue or a cluster of eigenvalues, the inner system (5.4) becomes poorly conditioned. This makes it difficult for an iterative solver to compute even a crude approximation to the solution of the inner system.

All these problems can be partially solved by the techniques developed in the trace minimization method, i.e., the multiple dynamic shifting strategy, the implicit deflation technique ( $d_{k,i}$  is required to be *B*-orthogonal to all the Ritz vectors obtained in the previous iteration step), and the dynamic stopping strategy. We call the modified algorithm *the Davidson-type trace minimization algorithm* [34].

#### 5.2. The Davidson-type trace minimization algorithm

Let  $s \ge p$  be the block size,  $m \ge s$  be a given integer that limits the dimension of the subspaces. The Davidson-type trace minimization algorithm is as follows.

Algorithm 4. The Davidson-type trace minimization algorithm. Choose a block size  $s \ge p$  and an  $n \times s$  matrix  $V_1$  such that  $V_1^T B V_1 = I_s$ . For k = 1, 2, ... until convergence, do

- 1. Compute  $W_k = AV_k$  and the interaction matrix  $H_k = V_k^T W_k$ .
- 2. Compute the s smallest eigenpairs  $(Y_k, \Theta_k)$  of  $H_k$ . The eigenvalues are arranged in ascending order and the eigenvectors are chosen to be orthogonal.
- 3. Compute the corresponding Ritz vectors  $X_k = V_k Y_k$ .
- 4. Compute the residuals  $R_k = W_k Y_k B X_k \Theta_k$ .
- 5. Test for convergence.



Fig. 2. The Davidson-type trace minimization algorithm.

6. For  $1 \leq i \leq s$ , solve the indefinite system

$$[P(A - \sigma_{k,i}B)P]d_{k,i} = Pr_{k,i}, \quad X_k^{\mathrm{T}}Bd_{k,i} = 0,$$
(5.5)

to a certain accuracy determined by the stopping criterion described in Section 4.3. The shift parameters  $\sigma_{k,i}$ ,  $1 \le i \le s$ , are determined according to the dynamic shifting strategy described in Section 4.2.

7. If dim $(V_k) \leq m - s$ , then

$$V_{k+1} = \operatorname{Mod} GS_B(V_k, \Delta_k)$$

else

$$V_{k+1} = \operatorname{Mod} GS_B(X_k, \Delta_k).$$

End for

The orthogonality requirement  $d_i^{(k)} \perp_B X_k$  is essential in the original trace minimization algorithm for maintaining the trace reduction property (2.7). In the current algorithm, it appears primarily as an implicit deflation technique. A more efficient approach is to require  $d_i^{(k)}$  to be *B*-orthogonal only to "good" Ritz vectors. Fig. 2 displays the convergence history of the Davidson-type trace minimization algorithm for the sample problem (1.3) where  $d_i^{(k)}$  is only required to be *B*-orthogonal to  $x_{k,i}$ . The number of outer iterations is decreased compared to the trace minimization algorithm in Section 4, and compared to the block Jacobi–Davidson algorithm: 15 iterations vs. 18 and 22 iterations, respectively. Moreover, in the block Jacobi–Davidson algorithm, the number of outer iterations cannot be reduced further when the number of iterations for the inner systems reaches 30. On the contrary, in the Davidson-type trace minimization algorithm, the number of outer iterations decreases steadily even when the number of iterations for the inner systems reaches 50. Note that reducing the number of outer iterations is important in a parallel or distributed computing environment.

Inner iterations	Block Jacobi–Davidson			Davidson-type Tracemin			
	#its	A mults	Time	#its	A mults	Time	
10	208	9368	28.5	216	9728	29.8	
20	103	8760	19.2	76	6468	14.8	
40	69	11392	19.0	34	5616	9.3	
60	54	13236	21.0	27	6564	10.1	
80	48	15608	22.0	24	7808	11.3	
100	57	23065	31.3	20	8108	11.6	
DS(MAX=120)	33	9653	17.0	23	7364	11.2	

Table 3 Numerical results for the test problem in [6] with 4 processors

#### 5.3. Numerical results

172

The block Jacobi–Davidson algorithm and the Davidson-type trace minimization algorithm have been coded in C with MPI [18] and PETSc [39]. Numerical experiments have been done on a variety of problems. In this section, we present some of the numerical results obtained on the SGI/Origin 2000.

We first show the results for an example used in [6]. This is a standard eigenvalue problem. The matrix A is defined by

$$a_{ij} = \begin{cases} i & \text{if } i = j, \\ 0.5 & \text{if } j = i + 1 \text{ or } j = i - 1, \\ 0.5 & \text{if } (i, j) \in \{(1, n), (n, 1)\}, \\ 0 & \text{otherwise.} \end{cases}$$

The size of the problem is n=10,000. We compute the four smallest eigenpairs with block size s=4 and maximum subspace size m=20. For both algorithms, the inner systems are solved approximately by the CG scheme. The eigenpairs are accepted when the relative residuals are less than  $10^{-10}$ .

In Table 3, we list the number of outer iterations, the number of matrix vector multiplications with A, and the execution time (in seconds) as functions of the number of inner iteration steps. We see that the performance of both algorithms are very close if the inner systems are solved crudely. The difference becomes clear when we increase the number of inner iteration steps. The dynamic shifting strategy accelerates the algorithm significantly. When the number of inner iteration steps reaches 40, the number of outer iterations is almost half that of the Ritz shifting strategy. When the number of inner iteration algorithm, but continues to decrease for the Davidson-type trace minimization algorithm. There are plenty of examples for which the block Jacobi–Davidson algorithm actually converges to wrong eigenpairs when the inner systems are solved to high accuracy. The last row of the table shows the result with the dynamic stopping strategy, where the maximum number of inner iteration steps is set to 120. We see that the dynamic shifting strategy improves the performance of the block Jacobi–Davidson algorithm dramatically. In our experiments, the starting subspaces for both algorithms are identical and were chosen randomly. The results clearly show that the success of the block Jacobi–Davidson algorithm depends on good starting spaces.

Problem	Maximum	Block Jacobi–Davidson			David	Davidson-type Tracemin		
	inner iterations	#its	A mults	Time	#its	A mults	Time	
BCSST08	40	34	3954	4.7	10	759	0.8	
BCSST09	40	15	1951	2.2	15	1947	2.2	
BCSST11	100	90	30990	40.5	54	20166	22.4	
BCSST21	100	40	10712	35.1	39	11220	36.2	
BCSST26	100	60	21915	32.2	39	14102	19.6	

Table 4 Numerical results for problems from the Harwell–Boeing collection with four processors

In Table 4, we show the results obtained for a few generalized eigenvalue problems in the Harwell– Boeing collection. These problems are difficult because the gap ratios for the smallest eigenvalues are extremely small due to the huge span of the spectra. Without preconditioning, none of these problems can be solved with a reasonable cost. In our experiments, we use the incomplete Cholesky factorization (IC(0)) of A as the preconditioner for all the matrices of the form  $A-\sigma B$ . The Davidson-type trace minimization algorithm works better than the block Jacobi–Davidson algorithm for three of the five problems. For the other two, the performance for both algorithms is similar. Both the shifting strategy and the stopping strategy do not work very well for these two problems because the 2-norms of the residuals are too large to be useful in selecting the shifting parameters.

In all the experiments, for both algorithms, the inner systems are solved by the CG scheme that is terminated when either the specified condition is met or an abnormal case is detected. It is surprising that the CG scheme works well considering that the inner systems for both algorithms are indefinite. The performance with other solvers for the inner systems are similar to that with the CG scheme. For the first problem in Table 4, however, if the inner systems are solved by GMRES(20) with IC(0) pre-conditioning, the block Jacobi–Davidson algorithm returns

84.78615951, 84.78643355, 84.78643355, 85.53681115

while the smallest four eigenvalues are

6.90070261, 18.14202961, 18.14236644, 18.14236645,

which were correctly returned by the Davidson-type trace minimization algorithm using the same inner system solver. This indicates that the Davidson-type trace minimization algorithm is also more robust than the block Jacobi–Davidson algorithm for some problems.

#### 6. Conclusions

In this paper, we presented a comprehensive overview of the trace minimization scheme, its variants, and comparisons with the block Jacobi–Davidson scheme. We demonstrated that, compared to a variant of the trace minimization scheme, the block Jacobi–Davidson algorithm depends more on a good initial subspace due to its choice of the Ritz values as the shift parameters. We showed that the Davidson-type trace minimization scheme can alleviate this dependence by adopting the dynamic shifting strategy and the stopping criterion developed for the original trace minimization algorithm. This variant of the trace minimization algorithm is not only more efficient but also more

robust than the block Jacobi–Davidson algorithm for symmetric generalized eigenvalue problems. Further research is needed, however, on how one can optimally precondition the indefinite systems that arise in both the Davidson-type trace minimization algorithm and the block Jacobi–Davidson algorithm. Our experience indicates that obtaining a positive-definite pre-conditioner for  $A - \sigma B$ , via an approximate Cholesky factorization that involves boosting of the diagonal elements, is a viable approach.

#### References

- [1] K.J. Bathe, E.L. Wilson, Large eigenvalue problems in dynamic analysis, ASCE, J. Eng. Mech. Div. 98 (1972) 1471–1485.
- [2] K.J. Bathe, E.L. Wilson, Solution methods for eigenvalue problems in structural mechanics, Internat. J. Numer. Methods Engrg 6 (1973) 213–226.
- [3] F.L. Bauer, Das Verfahren der Treppeniteration und Verwandte Verfahren zur Lösung Algebraischer Eigenwertprobleme, Z. Angew. Math. Phys. 8 (1957) 214–235.
- [4] E.F. Beckenbach, R. Bellman, Inequalities, Springer, New York, 1965.
- [5] M. Clint, A. Jennings, The evaluation of eigenvalues and eigenvectors of real symmetric matrices by simultaneous iteration, Comput. J. 13 (1970) 76–80.
- [6] M. Crouzeix, B. Philippe, M. Sadkane, The Davidson method, SIAM J. Sci. Comput. 15 (1994) 62-76.
- [7] J. Cullum, R.A. Willoughby, Lanczos and the computation in specified intervals of the spectrum of large, sparse real symmetric matrices, in: I.S. Duff, G.W. Stewart (Eds.), Sparse Matrix Proceedings 1978, SIAM Publications, Philadelphia, PA, 1979.
- [8] J. Cullum, R.A. Willoughby, Computing eigenvalues of very large symmetric matrices an implementation of a Lanczos algorithm with no reorthogonalization, J. Comput. Phys. 44 (1984) 329–358.
- [9] J. Daniel, W.B. Gragg, L. Kaufman, G.W. Stewart, Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization, Math. Comp. 33 (1976) 772–795.
- [10] E.R. Davidson, The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices, J. Comput. Phys. 17 (1975) 817–825.
- [11] T. Ericsson, A. Ruhe, The spectral transformation Lanczos method for the solution of large sparse generalized symmetric eigenvalue problems, Math. Comp. 35 (1980) 1251–1268.
- [12] D.R. Fokkema, G.L.G. Sleijpen, H.A. van der Vorst, Jacobi–Davidson style QR and QZ algorithms for the reduction of matrix pencils, SIAM J. Sci. Comput. 20 (1999) 94–125.
- [13] A. Gerschgorin, Über die Abgrenzung der Eigenwerte einer Matrix, Izv. Akad. Nauk SSSR Ser. Fiz.-Mat. 6 (1931) 749–754.
- [14] G.H. Golub, R. Underwood, The block Lanczos method for computing eigenvalues, in: J.R. Rice (Ed.), Mathematical Software III, Academic Press, New York, 1977, pp. 361–377.
- [15] G.H. Golub, C.F. van Loan, Matrix Computation, 3rd Edition, Johns Hopkins University Press, Baltimore, MD, 1993.
- [16] R.G. Grimes, J.G. Lewis, H.D. Simon, A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems, SIAM J. Matrix Anal. Appl. 15 (1994) 228–272.
- [17] R.C. Grimm, J.M. Greene, J.L. Johnson, Computation of the magnetohydrodynamic spectrum in axisymmetric toroidal confinement systems, in: Methods of Computational Physics 16, Academic Press, New York, 1976.
- [18] W.D. Gropp, E. Lusk, A. Skjellum, Using MPI: Portable Parallel Programming with the Message Passing Interface, MIT Press, Boston, MA, 1994.
- [19] R. Gruber, Finite hybrid elements to compute the ideal magnetohydrodynamic spectrum of an axisymmetric plasma, J. Comput. Phys. 26 (1978) 379–389.
- [20] C.G.J. Jacobi, Über ein leichtes Verfahren die in der Theorie der Säculärstörungen vorkommenden Gleichungen numerisch aufzulösen, J. Reine Angew. Math 30 (1846) 51–94.
- [21] T.Z. Kalamboukis, A Lanczos-type algorithm for the generalized eigenvalue problem  $Ax = \lambda Bx$ , J. Comput. Phys. 53 (1984) 82–89.

- [22] L.V. Kantorovič, Funkcional'nyi analiz i prikladnaja mathematika, Uspekhi Mat. Nauk 3 (1948) 9–185.
- [23] C. Lanczos, An iteration method for the solution of the eigenvalue problem of linear differential and integral operators, J. Res. Nat. Bur. Stand. Sect. B 45 (1950) 225–280.
- [24] A.A. Levin, On a method for the solution of a partial eigenvalue problem, USSR J. Comput. Math. Math. Phys. 5 (1965) 206–212.
- [25] B. Liu, The simultaneous expansion for the solution of several of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices, in: C. Moler, I. Shavitt (Eds.), Numerical Algorithms in Chemistry: Algebraic Method, Lawrence Berkeley Laboratory, University of California, California, 1978, pp. 49–53.
- [26] R.B. Morgan, D.S. Scott, Generalizations of Davidson's method for computing eigenvalues of sparse symmetric matrices, SIAM J. Sci. Statist. Comput. 7 (1986) 817–825.
- [27] C.C. Paige, The computation of eigenvalues of very large sparse matrices, Ph. D. Thesis, University of London, 1971.
- [28] B.N. Parlett, The Symmetric Eigenvalue Problem, Prentice-Hall, Englewood Cliffs, NJ, 1980.
- [29] B.N. Parlett, The Symmetric Eigenvalue Problem, SIAM, Philadelphia, 1998.
- [30] B.N. Parlett, D.S. Scott, The Lanczos algorithm with selective orthogonalization, Math. Comp. 33 (1979) 217-238.
- [31] H. Rutishauser, Computational aspects of F.L. Bauer's simultaneous iteration method, Numer. Math. 13 (1969) 4–13.
- [32] H. Rutishauser, Simultaneous iteration method for symmetric matrices, Numer. Math. 13 (1970) 205–223.
- [33] Y. Saad, Numerical Methods for Large Eigenvalue Problems, Halsted Press, New York, 1992.
- [34] A. Sameh, Z. Tong, Trace minimization and Jacobi–Davidson-type algorithms for large symmetric eigenvalue problems, Tech. Rep. CS-98, Purdue University, West Lafayette, Indiana, 1998.
- [35] A. Sameh, J.A. Wisniewski, A trace minimization algorithm for the generalized eigenvalue problem, SIAM J. Numer. Anal. 19 (1982) 1243–1259.
- [36] H.D. Simon, The Lanczos algorithm with partial reorthogonalization, Math. Comp. 42 (1984) 115–142.
- [37] G.L.G. Sleijpen, A.G.L. Booten, D.R. Fokkema, H.A. van der Vorst, Jacobi–Davidson type methods for generalized eigenproblems and polynomial eigenproblems, BIT 36 (1996) 595–633.
- [38] G.L.G. Sleijpen, H.A. van der Vorst, A Jacobi–Davidson iteration method for linear eigenvalue problems, SIAM J. Matrix Anal. Appl. 17 (1996) 401–425.
- [39] B.F. Smith, W.D. Gropp, L.C. McInnes, S. Balay, Petsc 2.0 users manual, Tech. Rep. ANL-95/11, Argonne National Laboratory, Chicago, IL, 1995.
- [40] A. Stathopoulos, Y. Saad, C.F. Fischer, Robust preconditioning of large, sparse, symmetric eigenvalue problems, J. Comput. Appl. Math. (1995) 197–215.
- [41] G.W. Stewart, Accelerating the orthogonal iteration for the eigenvalues of a hermitian matrix, Numer. Math. 13 (1969) 362–376.
- [42] G.W. Stewart, A bibliographical tour of the large, sparse generalized eigenvalue problems, in: J.R. Banch, D.J. Rose (Eds.), Sparse Matrix Computations, Academic Press, New York, 1976, pp. 113–130.
- [43] H.A. van der Vorst, G.H. Golub, 150 years old and still alive: Eigenproblems, in: I.S. Duff, G.A. Watson (Eds.), The State of the Art in Numerical Analysis, Clarendon Press, Oxford, 1997, pp. 93–119.
- [44] K. Wu, Preconditioning techniques for large eigenvalue problems, Ph. D. Thesis, University of Minnesota, Minneapolis, MN, 1997.



Journal of Computational and Applied Mathematics 123 (2000) 177-199

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.nl/locate/cam

# Successive overrelaxation (SOR) and related methods $\stackrel{\text{tr}}{\sim}$

A. Hadjidimos\*

Department of Mathematics, University of Crete, GR-714 09 Heraklion, Greece

Received 18 May 1999; received in revised form 19 September 1999

#### Abstract

Covering the last half of the 20th century, we present some of the basic and well-known results for the SOR theory and related methods as well as some that are not as well known. Most of the earlier results can be found in the excellent books by Varga (Matrix Iterative Analysis, Prentice-Hall, Englewood Cliffs, NJ, 1962) Young (Iterative Solution of Large Linear systems, Academic Press, New York, 1971) and Berman and Plemmons (Nonnegative Matrices in the Mathematical Sciences, SIAM, Philadelphia, PA, 1994) while some of the most recent ones are given in the bibliography of this paper. In this survey, both the point and the block SOR methods are considered for the solution of a linear system of the form Ax = b, where  $A \in \mathbb{C}^{n,n}$  and  $b \in \mathbb{C}^n \setminus \{0\}$ . Some general results concerning the SOR and related methods are given and also some more specific ones in cases where A happens to possess some further property, e.g., positive definiteness, L-, M-, H-matrix property, p-cyclic consistently ordered property etc. © 2000 Elsevier Science B.V. All rights reserved.

MSC: 65F10; CR Category: 5.14

Keywords: Iterative methods; Jacobi method; Gauss-Seidel method; SOR methods

### 1. Introduction and Preliminaries

For the numerical solution of a large nonsingular linear system

$$Ax = b, \quad A \in \mathbb{C}^{n,n}, \quad b \in \mathbb{C}^n \setminus \{0\}, \tag{1.1}$$

we consider iterative methods based on a splitting of the matrix A (see, e.g. [83,93] or [3]). Namely, we write

$$A = M - N \tag{1.2}$$

0377-0427/00\\$ - see front matter © 2000 Elsevier Science B.V. All rights reserved. PII: S0377-0427(00)00403-9

 $<sup>^{\</sup>star}$  This work is dedicated to Professors David M. Young and Richard S. Varga on their 75th and 70th birthday, respectively.

<sup>\*</sup> Tel.: 30-81-393865; fax: 30-81-393881.

E-mail address: hadjidim@math.uch.gr (A. Hadjidimos).

where M, the preconditioner, or preconditioning matrix, is taken to be invertible and cheap to invert, meaning that a linear system with matrix coefficient M is much more economical to solve than (1.1). Based on (1.2), (1.1) can be written in the fixed-point form

$$x = Tx + c, \quad T := M^{-1}N, \quad c := M^{-1}b,$$
(1.3)

which yields the following iterative scheme for the solution of (1.1):

$$x^{(m+1)} = Tx^{(m)} + c, \quad m = 0, 1, 2, ..., \text{ and } x^{(0)} \in \mathbb{C}^n \text{ arbitrary.}$$
 (1.4)

A sufficient and necessary condition for (1.4) to converge, to the solution of (1.1), is  $\rho(T) < 1$ , where  $\rho(.)$  denotes spectral radius, while a sufficient condition for convergence is ||T|| < 1, where ||.|| denotes matrix norm induced by a vector norm (see, e.g. [83,93,3]).

To derive the classical iterative methods one writes A = D - L - U, with D = diag(A), assuming  $\det(D) \neq 0$ , and L strictly lower and U strictly upper triangular matrices, respectively. Thus, the Jacobi iterative method  $(M \equiv D)$  is defined by

$$x^{(m+1)} = D^{-1}(L+U)x^{(m)} + D^{-1}b,$$
(1.5)

the Gauss–Seidel iterative method  $(M \equiv D - L)$  by

$$x^{(m+1)} = (D-L)^{-1} U x^{(m)} + (D-L)^{-1} b$$
(1.6)

and the Successive Overrelaxation (SOR) iterative method  $(M \equiv (1/\omega)(D - \omega L))$  by

$$x^{(m+1)} = \mathscr{L}_{\omega} x^{(m)} + c_{\omega}, \quad \mathscr{L}_{\omega} := (D - \omega L)^{-1} [(1 - \omega)D + \omega U], \quad c_{\omega} := \omega (D - \omega L)^{-1} b.$$
(1.7)

In (1.7),  $\omega \in \mathbb{C} \setminus \{0\}$  is the relaxation factor (or overrelaxation parameter). For  $\omega = 1$  the SOR becomes the Gauss–Seidel method.

The above three methods are called point methods to distinguish them from the block methods. For the latter, consider a partitioning of A in the following block form:

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1p} \\ A_{21} & A_{22} & \cdots & A_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ A_{p1} & A_{p2} & \cdots & A_{pp} \end{bmatrix},$$
(1.8)

where  $A_{ii} \in \mathbb{C}^{n_i,n_i}$ , i = 1(1)p, and  $\sum_{i=1}^p n_i = n$ . If we define  $D = \text{diag}(A_{11}, A_{22}, \dots, A_{pp})$ , assuming  $\det(A_{ii}) \neq 0$ , i = 1(1)p, set A = D - L - U with L and U being strictly lower and strictly upper triangular matrices, respectively, then the block Jacobi, the block Gauss–Seidel and the block SOR methods associated with the partitioning (1.8) of A are the iterative methods defined by precisely the same iterative schemes as their point counterparts in (1.5)–(1.7), respectively.

#### 2. Successive overrelaxation (SOR) method

The SOR method seems to have appeared in the 1930s and is mentioned in [79]. However, formally its theory was established almost simultaneously by Frankel [16] and Young [90].

In the development of the SOR theory one seeks values of  $\omega \in \mathbb{C} \setminus \{0\}$  for which the SOR method converges, the set of which defines the region of convergence, and, if possible, the best value of

 $\omega$ ,  $\omega_b$ , for which the convergence is asymptotically optimal, namely  $\rho(\mathscr{L}_{\omega_b}) = \min_{\omega \in \mathbb{C} \setminus \{0\}} \rho(\mathscr{L}_{\omega})$ . To find regions of convergence is a problem generally much easier than to determine  $\omega_b$ . In either case, however, one assumes that some information regarding the spectrum of the associated Jacobi iteration matrix J,  $\sigma(J)$ , is available. This information comes mostly from the properties of the matrix A (and the partitioning considered).

The only property of the SOR method that does not depend on properties of A, except for those needed to define the method, is the one below due to Kahan [46].

**Theorem 2.1** (Kahan). A necessary condition for the SOR method to converge is  $|\omega - 1| < 1$ . (For  $\omega \in \mathbb{R}$  this condition becomes  $\omega \in (0, 2)$ .)

*Note*: From now on it will be assumed that  $\omega \in \mathbb{R}$  unless otherwise specified.

#### 2.1. Hermitian matrices

**Definition 2.2.** A matrix  $A \in \mathbb{C}^{n,n}$  is said to be Hermitian if and only if (*iff*)  $A^{H} = A$ , where the superscript H denotes complex conjugate transpose. (A real Hermitian matrix is a real symmetric matrix and there holds  $A^{T} = A$ , where T denotes transpose.)

**Definition 2.3.** An Hermitian matrix  $A \in \mathbb{C}^{n,n}$  is said to be positive definite *iff*  $x^{H}Ax > 0$ ,  $\forall x \in \mathbb{C}^{n} \setminus \{0\}$ . (For A real symmetric, the condition becomes  $x^{T}Ax > 0$ ,  $\forall x \in \mathbb{R}^{n} \setminus \{0\}$ .)

A well-known result due to Ostrowski, who extended a previous one for the Gauss–Seidel method due to Reich, is given in [83]. Varga [84] gave a different proof and found the best value of  $\omega$ ,  $\omega_b$ .

**Theorem 2.4** (Reich–Ostrowski–Varga). Let  $A = D - E - E^{H} \in \mathbb{C}^{n,n}$  be Hermitian, D be Hermitian and positive definite, and det $(D - \omega E) \neq 0$ ,  $\forall \omega \in (0, 2)$ . Then,  $\rho(\mathscr{L}_{\omega}) < 1$  iff A is positive definite and  $\omega \in (0, 2)$ . (Note: Notice that except for the restrictions in the statement the matrices  $D, E \in \mathbb{C}^{n,n}$  must satisfy, they can be any matrices!)

*Note*: It is worth mentioning that there is a form of the theorem due to Kuznetsov [53] that applies also in singular cases.

2.2. L-, M- and H-matrices

Notation. Let  $A, B \in \mathbb{R}^{n,n}$ . If  $a_{ij} \ge b_{ij}$   $(a_{ij} > b_{ij})$ , i, j = 1(1)n, we write  $A \ge B$  (A > B). The same notation applies to vectors  $x, y \in \mathbb{R}^n$ .

**Definition 2.5.** If  $A \in \mathbb{R}^{n,n}$  satisfies  $A \ge 0$  (> 0) then it is said to be *nonnegative* (*positive*). The same terminology applies to vectors  $x \in \mathbb{R}^n$ .

**Notation.** Let  $A \in \mathbb{C}^{n,n}$ . Then |A| denotes the matrix whose elements are the moduli of the elements of A. The same notation applies to vectors  $x \in \mathbb{C}^n$ .

From the Perron–Frobenius theory for nonnegative matrices (see [83,93] or [3]) the following statement holds.
**Theorem 2.6.** Let  $A \in \mathbb{C}^{n,n}$  and  $B \in \mathbb{R}^{n,n}$  satisfy  $0 \leq |A| \leq B$ , then  $0 \leq \rho(A) \leq \rho(|A|) \leq \rho(B)$ .

**Definition 2.7.** A matrix  $A \in \mathbb{R}^{n,n}$  is said to be an *L*-matrix *iff*  $a_{ii} > 0$ , i = 1(1)n, and  $a_{ij} \leq 0$ ,  $i \neq j = 1(1)n$ .

**Definition 2.8.** A matrix  $A \in \mathbb{R}^{n,n}$  is said to be an *M*-matrix iff  $a_{ij} \leq 0$ ,  $i \neq j=1(1)n$ , *A* is nonsingular and  $A^{-1} \geq 0$ .

**Remark.** It is pointed out that in [3] 50 equivalent conditions for a matrix  $A \in \mathbb{R}^{n,n}$ , with  $a_{ij} \leq 0$ ,  $i \neq j = 1(1)n$ , to be an *M*-matrix are given!

**Definition 2.9.** A matrix  $A \in \mathbb{C}^{n,n}$  is said to be an *H*-matrix *iff* its *companion matrix*, that is the matrix  $\mathcal{M}(A)$  with elements  $m_{ii} = |a_{ii}|, i = 1(1)n$ , and  $m_{ij} = -|a_{ij}|, i \neq j = 1(1)n$ , is an *M*-matrix.

**Definition 2.10.** A splitting (1.2) of a nonsingular matrix  $A \in \mathbb{R}^{n,n}$  is said to be *regular* if  $M^{-1} \ge 0$  and  $N \ge 0$ . (Varga proved among others that the iterative scheme (1.4) based on a regular splitting is convergent; he also made comparisons of the spectral radii corresponding to two different regular splittings of the same matrix A (see [83]).)

**Definition 2.11.** A splitting (1.2) of a nonsingular matrix  $A \in \mathbb{R}^{n,n}$  is said to be *weak regular* if  $M^{-1} \ge 0$  and  $M^{-1}N \ge 0$ . (As Neumann and Plemmons proved, see, e.g. [3], this definition leads to some results very similar to those of the regular splittings.)

A theorem connecting spectral radii of the Jacobi and the Gauss–Seidel iteration matrices associated with an L-matrix A was given originally by Stein and Rosenberg. In Young [93] a form of it that includes the spectral radius of the SOR iteration matrix is given below. Its proof is mainly based on the Perron–Frobenius theory.

**Theorem 2.12.** If  $A \in \mathbb{R}^{n,n}$  is an L-matrix and  $\omega \in (0,1]$ , then:

- (a)  $\rho(J) < 1$  iff  $\rho(\mathscr{L}_{\omega}) < 1$ .
- (b)  $\rho(J) < 1$  iff A is an M-matrix; if  $\rho(J) < 1$  then  $\rho(\mathscr{L}_{\omega}) \leq 1 \omega + \omega \rho(J)$ .
- (c) If  $\rho(J) \ge 1$  then  $\rho(\mathscr{L}_{\omega}) \ge 1 \omega + \omega \rho(J) \ge 1$ .

*Notes*: (i) The original form of Stein–Rosenberg theorem restricts to  $\omega = 1$  and gives four mutually exclusive relations:

(a) 
$$0 = \rho(\mathcal{J}) = \rho(\mathcal{L}_1)$$
, (b)  $0 < \rho(\mathcal{L}_1) < \rho(J) < 1$ ,  
(c)  $1 = \rho(\mathcal{J}) = \rho(\mathcal{L}_1)$ , (d)  $1 < \rho(\mathcal{J}) < \rho(\mathcal{L}_1)$ .
(2.1)

(ii) Buoni and Varga [5,6] and also Buoni et al. [4] generalized the original Stein-Rosenberg theorem in another direction than that of Theorem 2.12 by assuming that  $A \in \mathbb{C}^{n,n}$ , D, L and U are any matrices with  $D^{-1}L$  and  $D^{-1}U$  strictly lower and strictly upper triangular matrices, respectively, and  $\mathbb{R}^{n,n} \ni J = D^{-1}(L+U) \ge 0$ .

In [93] a theorem that gives an interval of  $\omega$  for which the SOR method converges for *M*-matrices *A* is based on the previous statement and on the theory of regular splittings is stated.

**Theorem 2.13.** If  $A \in \mathbb{R}^{n,n}$  is an *M*-matrix and if  $\omega \in (0, 2/(1 + \rho(J)))$  then  $\rho(\mathscr{L}_{\omega}) < 1$ .

The following is a statement extending the previous one to H-matrices.

**Theorem 2.14.** If  $A \in \mathbb{C}^{n,n}$  is an *H*-matrix and if  $\omega \in (0, 2/(1 + \rho(|J|)))$  then  $\rho(\mathscr{L}_{\omega}) < 1$ .

## 2.3. 2- and p-cyclic consistently ordered matrices

There is a class of matrices for which the investigation for the optimal value of  $\omega$  leads to the most beautiful theory that has been developed for the last 50 years and which is still going on. It is associated with the class of *p*-cyclic consistently ordered matrices. Such matrices naturally arise, e.g., for p = 2 in the discretization of second-order elliptic or parabolic PDEs by finite differences, finite element or collocation methods, for p = 3 in the case of large-scale least-squares problems, and for any  $p \ge 2$  in the case of Markov chain analysis.

**Definition 2.15.** A matrix  $A \in \mathbb{C}^{n,n}$  possesses Young's "property A" if there exists a permutation matrix P such that

$$PAP^{\mathrm{T}} = \begin{bmatrix} D_1 & B\\ C & D_2 \end{bmatrix}, \tag{2.2}$$

where  $D_1, D_2$  are nonsingular diagonal matrices not necessarily of the same order.

A special case of Young's "property A" is what Varga calls *two-cyclic consistently ordered* property [83].

**Definition 2.16.** A matrix  $A \in \mathbb{C}^{n,n}$  is said to be *two-cyclic consistently ordered* if  $\sigma(D^{-1}(\alpha L + (1/a)U))$  is independent of  $\alpha \in \mathbb{C} \setminus \{0\}$ .

Among others, matrices that possess both Young's "property A" and Varga's "two-cyclic consistently ordered property" are the tridiagonal matrices, with nonzero diagonal elements, and the matrices that have already form (2.2).

For two-cyclic consistently ordered matrices A, Young discovered [90,91] that the eigenvalues  $\mu$  and  $\lambda$  of the Jacobi and the SOR iteration matrices, respectively, associated with A satisfy the functional relationship

$$(\lambda + \omega - 1)^2 = \omega^2 \mu^2 \lambda. \tag{2.3}$$

He also found that if  $J = D^{-1}(L + U)$ , the eigenvalues of  $J^2$  are nonnegative and  $\rho(J) < 1$ , then there exists an optimal value of  $\omega$ ,  $\omega_b$ , such that

$$\omega_b = \frac{2}{1 + (1 - \rho^2(J))^{1/2}}, \quad \rho(\mathscr{L}_{\omega_b}) = \omega_b - 1 \ (< \rho(\mathscr{L}_{\omega}) \text{ for all } \omega \neq \omega_b).$$
(2.4)

(Note: For more details see [93].)

Varga generalized the concept of two-cyclic consistently ordered matrices to what he called (*block*) *p*-cyclic consistently ordered.

**Definition 2.17.** A matrix  $A \in \mathbb{C}^{n,n}$  in the block form (1.8) is said to be *(block) p-cyclic consistently* ordered if  $\sigma(D^{-1}(\alpha L + (1/\alpha^{p-1})U))$  is independent of  $\alpha \in \mathbb{C} \setminus \{0\}$ .

The best representative of such a block partitioned matrix will be the following:

	$\begin{bmatrix} A_{11} \end{bmatrix}$	0	0	• • •	$A_{1p}$
	$A_{21}$	$A_{22}$	0		0
A =	0	$A_{32}$	$A_{33}$		0
	:	÷	۰.	·	:
		0		$A_{p, p-1}$	$A_{pp}$

**Remark.** The spectrum  $\sigma(J)$ , of the eigenvalues of the (block) Jacobi iteration matrix associated with a *p*-cyclic consistently ordered matrix *A* (2.5), which Varga calls *weakly cyclic of index p* [83], presents a *p*-cyclic symmetry about the origin. That is, with each eigenvalue  $\mu \in \sigma(J) \setminus \{0\}$  there are another p - 1 eigenvalues of *J*, of the same multiplicity as that of  $\mu$ , given by the expressions  $\mu \exp(i(2\pi k)/p)$ , k = 1(1)p - 1.

**Notation.** From now on the Jacobi iteration matrix associated with a block *p*-cyclic consistently ordered matrix will be denoted by  $J_p$ .

For such matrices Varga [82] extended Young's results (2.3)-(2.4) to any  $p \ge 3$ , namely

$$(\lambda + \omega - 1)^p = \omega^p \mu^p \lambda^{p-1}.$$
(2.6)

He also proved that if the *p*th powers of the eigenvalues  $\mu \in \sigma(J_p)$  are real nonnegative and  $\rho(J_p) < 1$ , then there exists an optimal value of  $\omega$ ,  $\omega_b$ , which is the unique positive real root in (1, p/(p-1)) of the equation

$$\left(\rho(J_p)\omega_b\right)^p = \frac{p^p}{(p-1)^{p-1}}(\omega_b - 1),\tag{2.7}$$

which is such that

$$\rho(\mathscr{L}_{\omega_b}) = (p-1)(\omega_b - 1) \ (< \rho(\mathscr{L}_{\omega}) \text{ for all } \omega \neq \omega_b).$$
(2.8)

Similar optimal results for  $\sigma(J_p^p)$  nonpositive have been obtained for p = 2 by Kredell [52] and Niethammer [66], for p = 3 by Niethammer et al. [67] and for any  $p \ge 3$  by Wild and Niethammer [88] and also by Galanis et al. [18].

In the analyses given in [83,93,52,66,67,88], the regions of convergence, in all the previous cases where optimal  $\omega$ 's were obtained, are also determined. In the following statement [3] the optimal values and the regions of convergence are given.

**Theorem 2.18.** Let the matrix  $A \in \mathbb{C}^{n,n}$  be p-cyclic consistently ordered and suppose that all the eigenvalues of  $J_p^p$  are nonnegative (nonpositive). Let s = 1 (-1) if the signs of the eigenvalues of

 $J_p^p$  are nonnegative (nonpositive). If

$$\rho(J_p) < \frac{p-1-s}{p-2},\tag{2.9}$$

then the regions of convergence of the SOR method ( $\rho(\mathscr{L}_{\omega}) < 1$ ) are

For 
$$s = 1$$
,  $\omega \in \left(0, \frac{p}{p-1}\right)$  and for  $s = -1$ ,  $\omega \in \left(\frac{p-2}{p-1}, \frac{2}{1+\rho(J_p)}\right)$ . (2.10)

The optimal relaxation factor  $\omega_b$  is the unique real positive root  $\omega_b \in ((2p-3+s)/(2(p-1)), (2p-1+s)/(2(p-1)))$  of the equation

$$(\rho(J_p)\omega_b)^p = s p^p (p-1)^{1-p} (\omega_b - 1)$$
(2.11)

and the optimal SOR spectral radius is given by

$$\rho(\mathscr{L}_{\omega_b}) = s(p-1)(\omega_b - 1) \ (< \rho(\mathscr{L}_{\omega}) \text{ for all } \omega \neq \omega_b).$$
(2.12)

*Note:* For p = 2, (p-2)/(p-2) and p/(p-2) should be interpreted as 1 and  $\infty$ , respectively.

In passing we mention that the only case in which a complex optimal  $\omega_b$  has been determined [52] is the case of a two-cyclic consistently ordered matrix with  $\sigma(J_2)$  on a straight line segment, namely  $\sigma(J_2) \subset [-\rho(J_2)\exp(i\theta), \rho(J_2)\exp(i\theta)]$ , with any  $\rho(J_2)$  and any  $\theta \in (0, \pi)$ . The corresponding optimal values are given by

$$\omega_b = \frac{2}{1 + (1 - \rho^2(J_2)\exp(2i\theta))^{1/2}}, \quad \rho(\mathscr{L}_{\omega_b}) = |\omega_b - 1| \ (<\rho(\mathscr{L}_{\omega}) \text{ for all } \omega \neq \omega_b), \tag{2.13}$$

where of the two square roots the one with the nonnegative real part is taken. It is noted that for  $\theta = 0$  and  $\rho(J_2) < 1$ , and also for  $\theta = \pi/2$  and any  $\rho(J_2)$ , the optimal formulas by Young [90,91,93], and by Kredell [52] and Niethammer [66], respectively, are recovered.

As Varga first noticed [82], the transformation (2.6) that maps  $\mu$  to  $\lambda^{1/p}$  is a conformal mapping transformation. The study of this transformation, to find regions of convergence for  $\omega$  and its optimal value,  $\omega_b$ , involves ellipses for p = 2 and *p*-cyclic hypocycloids (cusped, shortened and stretched) for  $p \ge 3$ . The latter curves for p = 5 are depicted in Fig. 1. (In [88] not only  $\omega_b$  and  $\rho(\mathscr{L}_{\omega_b})$  are determined but also an excellent analysis with hypocycloids is done which allows the authors to obtain regions of convergence for  $\omega$ .)

So, for matrices  $A \in \mathbb{C}^{n,n}$  *p*-cyclic consistently ordered, because of  $\omega \in \mathbb{R}$ , the *p*-cyclic symmetry of the spectrum  $\sigma(J_p)$  and of the *p*-cyclic hypocycloids about the origin and the symmetry of the latter with respect to (*wrt*) the real axis, the optimal problems that have been considered so far can be called *one-point problems*. This is justified from the fact that the coordinates of only one *critical* point suffice to determine the optimal parameters. E.g., for any  $p \ge 2$  and  $0 \le \mu^p < 1$ , the point  $(\rho(J_p), 0)$  is the only information needed, for p = 2 and  $\mu^2 \le 0$  we only need the point  $(0, i\rho(J_2))$  while for  $p \ge 3$  and  $-(p/(p-2))^p < \mu^p \le 0$ , only the point  $(\rho(J_p)\cos(\pi/2p), i\rho(J_p)\sin(\pi/2p))$  suffices.

One may notice that in the case of the *one-point problem* because we are dealing with complex matrices A, in general, one has to consider not only the spectrum  $\sigma(J_p)$  but also its symmetric *wrt* the real axis. E.g., for p = 2, if there is a rectangle symmetric *wrt* both axes that contains  $\sigma(J_2)$  in the closure of its interior and which lies within the infinite unit strip  $S := \{z \in \mathbb{C} \mid |Rez| < 1\}$ , then



Fig. 1. Hypocycloids of all kinds and types for p = 5.

the only information needed to find  $\omega_b$  and  $\rho(\mathscr{L}_{\omega_b})$  is the pair of coordinates of its vertex in the first quadrant ((2p) - ant with p = 2). This problem was solved by Kjellberg [50] and Russell [73] and the optimal values are given by the elements of the unique *best* ellipse that passes through the vertex in question (see also [93]).

The most general *one-point problem* has been solved recently in [19] where, among others, use of most of the previous results and also of those in [69] was made. In [19] it is assumed that  $A \in \mathbb{C}^{n,n}$  is *p*-cyclic consistently ordered and there exists one element of  $\sigma(J_p)$  or of its mirror image  $\sigma'(J_p)$  wrt the real axis in the first (2p) - ant with polar coordinates  $(r, \theta)$  such that the cusped hypocycloid of type II that passes through  $(r, \theta)$  crosses the real axis at a point with abscissa strictly less that 1 and on the other hand, the hypocycloid just mentioned and the cusped hypocycloid of type I through  $(r, \theta)$  contain both  $\sigma(J_p)$  and  $\sigma'(J_p)$  in the closure of the intersection of their interiors. In such a case  $\omega_b$  and  $\rho(\mathscr{L}_{\omega_b})$  can be found through analytical expressions in terms of the semiaxes of the unique *best* shortened hypocycloid that passes through  $(r, \theta)$ . It is worth pointing out that all the previous cases mentioned so far are particular subcases of the one just described.

The case of the *two-point problem* in its simplest form is when p = 2 and its spectrum  $\sigma(J_2)$  is real and such that  $-a^2 \le \mu^2 \le b^2$  with a, b > 0 and b < 1. This problem was solved by Wrigley [89] (see also [93]) and the optimal parameters are given by

$$\omega_b = \frac{2}{1 + (1 - b^2 + a^2)^{1/2}}, \quad \rho(\mathscr{L}_{\omega_b}) = \left(\frac{b + a}{1 + (1 - b^2 + a^2)^{1/2}}\right)^2 \ (< \rho(\mathscr{L}_{\omega}) \text{ for all } \omega \neq \omega_b).$$
(2.14)

(*Note:* The solution just given solves a more general problem, namely the one where  $\sigma(J_2)$  lies in the closed interior of the ellipse with semiaxes b and a. We note that the cases of nonnegative and nonpositive  $\mu^2$  presented previously are particular subcases of the present one.)

The solution to the *two-point problem* for any  $p \ge 3$ , provided that  $-a^p \le \mu^p \le b^p$ , with a, b > 0and a < p/(p-2), b < 1, was given by Eiermann et al. [11] by means of the unique *p*-cyclic shortened hypocycloid through both points (b, 0) and  $(a \cos(\pi/2p), ia \sin(\pi/2p))$  iff (p-2)/p < a/b< p/(p-2) which becomes a cusped I through (b, 0) iff  $a/b \le (p-2)/p$  and a cusped II through  $(a \cos(\pi/2p), ia \sin(\pi/2p))$  iff  $p/(p-2) \le a/b$ . More specifically:

**Theorem 2.19.** Under the notation and the assumptions so far, for (p-2)/p < a/b < p/(p-2),  $\omega_b$  is given as the unique positive real root in ((p-2)/(p-1), p/(p-1)) of the equation

$$\left(\frac{b+a}{2}\omega_b\right)^p = \frac{b+a}{b-a}(\omega_b - 1),\tag{2.15}$$

which is such that

$$\rho(\mathscr{L}_{\omega_b}) = \frac{b+a}{b-a}(\omega_b - 1) \ (< \rho(\mathscr{L}_{\omega}) \ for \ all \ \omega \neq \omega_b).$$
(2.16)

(*Note*: When  $a/b \le (p-2)/p$  and  $a/b \ge p/(p-2)$  the above equations and expressions reduce to the ones of the *one-point problem* for the nonnegative and nonpositive case, respectively.)

For p = 2 and for a *two-point problem* where the vertices of two rectangles, both lying in the open infinite unit strip S and are symmetric *wrt* both axes, in the first quadrant are given and the closure of the intersection of their interiors contains  $\sigma(J_2)$ , the solution was given by Young and Eidson [94] (see also [93]) by a simple algorithm that uses the two *best* ellipses through each of the vertices and also the ellipse through the two points. An ingenious extension of this algorithm [94] gives the solution to the corresponding *many-point problem*.

The analogs to the *two-* and *many-point problems* for any  $p \ge 3$  has been solved very recently by Galanis et al. [20]. The solutions are given by means of algorithms analogous to the ones by Young and Eidson where instead of ellipses shortened hypocycloids are used.

## 2.3.1. Generalized (q, p - q)-cyclic consistently ordered matrices

The block form of these matrices is the following:

	$\int A_{11}$	0	•••	0	$A_{1, p-q+1}$	0	• • •	0 ]		
	0	$A_{22}$	•••	0	0	$A_{2, p-q+2}$	• • •	0		
	:	÷		÷	÷	÷		÷		
A =	0	0		0	0	0		$A_{qp}$	,	(2.17)
	$A_{q+1,1}$	0	•••	0	0	0	0	0		
	:	:		:	÷	÷		÷		
	0	0	•••	$A_{p,p-q}$	0	0		$A_{pp}$		

where the diagonal blocks satisfy the same restrictions as in (1.8) and p and q are relatively prime. Obviously, for q = 1 the generalized (1, p - 1)-cyclic consistently ordered matrices reduce to the block p-cyclic consistently ordered ones of the previous section. This time the functional relationship that connects the spectra of the block Jacobi iteration matrix  $J_{q,p-q}$  and of the block SOR matrix associated with A in (2.17) is

$$(\lambda + \omega - 1)^p = \omega^p \mu^p \lambda^{p-q}.$$
(2.18)

(2.18) is attributed to Verner and Bernal [87]. However, it seems that Varga implies the corresponding class of matrices and in some cases the optimal parameters of the associated SOR method (see [83, pp. 108–109, Exs 1,2]). For the basic theory concerning matrices of the present class as well as of their point counterparts the reader is referred to Young [93].

**Remarks.** (i) In Young [93], a more general than (2.17) form of matrices, called *generalized* (q, p - q)-consistently ordered (GCO(q, p - q)), are analyzed and studied extensively. (ii) Varga brought to the attention of the present author [85] that it appears that GCO(q, p - q) matrices are, from a graph-theoretic point of view, essentially reorderings of the (block) *p*-cyclic consistently ordered matrices. This *new* result seems to make the theory of GCO(q, p - q) matrices redundant! (iii) Two more points: (a) Optimal SOR results to cover all possible cases for the two classes of matrices (*p*-cyclic consistently ordered and GCO(q, p - q) ones) have not been found, and (b) As was shown in [36] there are cases where for certain values of  $\omega \in (0, 2)$ , the SOR method applied to a GCO(q, p - q) matrix *A* converges while when it is applied to the corresponding reordered *p*-cyclic consistently ordered matrix diverges. (iv) In view of the two points in (iii) in the following we shall keep on considering the GCO(q, p - q) matrices mostly in the form (2.17).

For the optimal parameters little has been done because it seems that the corresponding problems are not only difficult to attack but also there are no obvious practical applications associated with them. The only optimal results known to us are those by Nichols and Fox [64] who found that for  $\sigma(J_{q,p-q}^p)$  nonnegative and  $\rho(J_{q,p-q}) < 1$ , it is  $\omega_b = 1$  and  $\rho(\mathscr{L}_{\omega_b}) = \rho^{p/q}(J_{q,p-q})$  and also the one by Galanis et al. [21] who treated the nonpositive case for q = p - 1 and p = 3 and 4 and obtained analytical expressions for  $\omega_b$  and  $\rho(\mathscr{L}_{\omega_b})$ .

### 2.3.2. Regions of convergence

Besides optimal results in the case of *p*-cyclic and GCO(q, p - q) matrices researchers in the area are also interested in the regions of convergence of the SOR method in the  $(\rho(J_p), \omega)$ -plane especially in case the spectrum  $\sigma(J_p^p)$  is nonnegative or nonpositive. The 2-cyclic consistently ordered cases are trivial but the cases of *p*-cyclic consistently ordered matrices for any  $p \ge 3$  are not. For p=3, Niethammer et al. [67] determined the exact regions in the nonnegative and nonpositive cases. For any  $p \ge 3$  the solution was given in [28] where use of the famous Schur–Cohn algorithm was made (see [45]). The only other case where the Schur–Cohn algorithm was successfully applied was in the case of nonnegative and nonpositive spectra  $\sigma(J_p^p)$  for  $p \ge 3$  in the case of the GCO(p-1,1) matrices (see [36]). By using *asteroidal* hypocycloids, regions of convergence for the SOR are found in [34] for GCO(q, p - q) matrices when  $\sigma(J_p^p)$  is nonnegative or nonpositive. Finally, as in the previous case, but dropping the assumption on nonnegativeness and nonpositiveness, using the Rouché's Theorem [80], as in [29], one can find that sufficient conditions for the SOR method to converge for all  $p \ge 3$  are  $\rho(J_{q,p-q}) < 1$  and  $0 < \omega < 2/(1 + \rho(J_{q,p-q}))$ , that is the same basic conditions as in Theorem 2.13.

### 2.3.3. Singular linear systems and p-cyclic SOR

For singular linear systems the associated Jacobi iteration matrix has in its spectrum the eigenvalue 1. The very first theoretical results in this case for the SOR method were given by Buoni, Neumann and Varga [4]. If, however, the matrix coefficient A happens to be p-cyclic consistently ordered and in the Jordan form of the Jacobi iteration matrix the eigenvalue 1 is associated with  $1 \times 1$  blocks only then the theory regarding convergence and optimal results seems to be precisely that of the nonsingular case where simply the eigenvalue 1 is discarded (see, e.g., [26,51,38]). Since this case is of much practical importance in the Markov Chain Analysis the reader is specifically referred to [51] for details on this and also on the concept of what is called *Extended SOR*.

### 2.3.4. Block p-cyclic repartitioning

In a case arising in the solution of large linear systems for least-squares problems Markham, Neumann and Plemmons [60] observed that if a block 3-cyclic consistently ordered matrix as in (2.5), with  $\sigma(J_3^3)$  nonpositive and  $\rho(J_3) < 3$ , was repartitioned and considered as a block 2-cyclic consistently ordered matrix as

$$A = \begin{bmatrix} A_{11} & 0 & A_{13} \\ A_{21} & A_{22} & 0 \\ \hline 0 & A_{23} & A_{33} \end{bmatrix},$$
 (2.19)

then the SOR method associated with the latter had much better convergence properties than the SOR associated with the former and also it was convergent for any  $\rho(J_3)$ . This was mainly based on the observation that  $\sigma(J_2^2) \setminus \{0\} \equiv \sigma(J_3^3) \setminus \{0\}$ .

The previous work was the starting point for an investigation that followed. So, Pierce, Hadjidimos and Plemmons [72] proved that for block *p*-cyclic consistently ordered matrices when the spectrum  $\sigma(J_p^p)$  was either nonnegative, with  $\rho(J_p) < 1$ , or nonpositive, with any  $\rho(J_p)$ , the 2-cyclic repartitioning was not only always the *best* among all possible repartitionings but was also giving convergent SOR methods in the nonpositive case even when the corresponding to the original partitioning SOR method failed to converge!

Later Eiermann et al. [11], using theoretical and numerical examples, showed that the result obtained in [72] was *not* always true for *real* spectra  $\sigma(J_p^p)$ .

Finally, Galanis and Hadjidimos [17] considered the general case of the real spectra  $\sigma(J_p^p)$  and all q-cyclic repartitionings for  $2 \le q \le p$  of the original block p-cyclic matrix A and found the best q-cyclic repartitioning out of all possible repartitionings.

### 3. Modified SOR method

The idea of Modified SOR (or MSOR) method is to associate a different  $\omega$  with each (block) row of the original linear system. The idea goes back to Russell [73] but it was mainly McDowell [61] and Taylor [81], who analyzed its convergence properties (see also [49]). It is best applied when the matrix A is 2-cyclic consistently ordered of the form (2.2). In such a case the MSOR method will be defined by the following iterative scheme:

$$x^{(m+1)} = \mathscr{L}_{\omega_1,\omega_2} x^{(m)} + c_{\omega_1,\omega_2}, \tag{3.1}$$

where

$$\mathcal{L}_{\omega_{1},\omega_{2}} := (D - \omega_{2}L)^{-1} [\operatorname{diag}((1 - \omega_{1})D_{1}, (1 - \omega_{2})D_{2}) + \omega_{1}U],$$
  

$$c_{\omega_{1},\omega_{2}} := (D - \omega_{2}L)^{-1} \operatorname{diag}(\omega_{1}I_{n_{1}}, \omega_{2}I_{n_{2}})b$$
(3.2)

with  $I_{n_1}, I_{n_2}$  the unit matrices of the orders of  $D_1, D_2$ , respectively.

In such a case the basic relationship that connects the eigenvalues  $\mu$  and  $\lambda$  of the spectra  $\sigma(J_2)$  and  $\sigma(\mathscr{L}_{\omega_1,\omega_2})$  is

$$(\lambda + \omega_1 - 1)(\lambda + \omega_2 - 1) = \omega_1 \omega_2 \mu^2 \lambda, \tag{3.3}$$

which reduces to the classical one by Young for the SOR method for  $\omega_1 = \omega_2$ . Optimal results for spectra  $\sigma(J_2)$  of various configurations have been successfully obtained in some cases. For example: (i) For  $\sigma(J_2)$  lying on a cross-shaped region optimal results can be found in [43] from which several other ones previously obtained by Taylor and other researchers can be easily recovered. (ii) For spectra  $\sigma(J_2)$  lying on the unit circle and at the origin, except at the points  $(\pm 1, 0)$ , which is the case of the Jacobi iteration matrices arising in the discretization of second order elliptic boundary value problems by the finite-element collocation method with Hermite elements [40] and (iii) For  $\sigma(J_2)$  lying in a "bow-tie" region which is the case arising in the discretization of the convection-diffusion equation by finite differences [2]. It is proved that the optimal MSOR method converges much faster than the optimal SOR and it also converges even in cases where the optimal SOR diverges. (For more details see [7,13] and especially Section 6 of [2].)

For extensions of the theory to GCO(q, p - q)-matrices the reader is referred to [43].

A problem which seemed to have been dealt with by Young and his colleagues (see [93]) in the 1960s, (see also [47]), was recast rather recently by Golub and de Pillis [22] in a more general form. More specifically, because the spectral radius is only an asymptotic rate of convergence of a linear iterative method the question raised was to determine, for each  $k \ge 1$ , a relaxation parameter  $\omega \in (0, 2)$  and a pair of relaxation parameters  $\omega_1, \omega_2$  which minimize the Euclidean norm of the *k*th power of the SOR and MSOR iteration matrices associated with a real symmetric positive-definite matrix with *property A*. In [31] these problems were solved completely for k = 1. Here are the corresponding results:

**Theorem 3.1.** Let  $A \in \mathbb{R}^{n,n}$  be a symmetric positive-definite matrix having property A and the block form

$$A = \begin{bmatrix} I_{n_1} & -M \\ -M^T & I_{n_2} \end{bmatrix} =: I - J_2, \quad n_1 + n_2 = n.$$
(3.4)

Then for any fixed  $t := \rho^2(J_2) \in [0,1)$  the value of  $\omega$ , call it  $\hat{\omega}$ , which yields the minimum in  $\min_{\omega \in (0,2)} \|\mathscr{L}_{\omega}\|_2$  is the unique real positive root in (0,1) of the quartic equation

$$(t^{2} + t^{3})\omega^{4} + (1 - 4t^{2})\omega^{3} + (-5 + 4t + 4t^{2})\omega^{2} + (8 - 8t)\omega + (-4 + 4t) = 0.$$
(3.5)

In fact  $\hat{\omega} \in (0, \omega^*)$ , where  $\omega^*$  is the unique real positive root in (0, 1) of the cubic

$$(t+t^2)\omega^3 - 3t\omega^2 + (1+2t)\omega - 1 = 0.$$
(3.6)

**Theorem 3.2.** Under the assumptions and notation of the previous theorem and for any fixed  $t \in [0,1)$  the pair  $(\omega_1, \omega_2)$ , call it  $(\hat{\omega}_1, \hat{\omega}_2)$ , which yields the minimum in  $\hat{\delta} := \min_{\omega_1, \omega_2 \in (0,2)} \|\mathscr{L}_{\omega_1, \omega_2}\|_2$  is as follows: For  $t \in [0, \frac{1}{3}]$ 

$$(\hat{\omega}_1, \hat{\omega}_2) = \left(\frac{1}{1+t}, \frac{1}{1-t}\right) \text{ when } \hat{\delta} = \left(\frac{t}{1+t}\right)^{1/2}$$
 (3.7)

while for  $t \in [\frac{1}{3}, 1)$ 

$$(\hat{\omega}_1, \hat{\omega}_2) = \left(\frac{4}{5+t}, \frac{4}{3-t}\right) \text{ when } \hat{\delta} = \frac{1+t}{3-t}.$$
 (3.8)

**Remark.** (i) It is worth pointing out that in [93] the values of  $\hat{\omega}$  and the corresponding ones for  $\|\mathscr{L}_{\hat{\omega}}\|_2$  are given for  $t^{1/2} = \rho(J_2) = 0(0.1)1$ . (ii) Part of Theorem 3.2 is also given in [93] where its proof at some points is based on strong numerical evidence.

We conclude this section by giving the functional eigenvalue relationship connecting the spectra of the Jacobi iteration matrix of a GCO(q, p-q) matrix A of the class (2.17) and of the corresponding MSOR operator when each block is associated with a different relaxation factor  $\omega_j$ , j = 1(1)p. The formula below is an extension of the one given by Taylor [81]

$$\prod_{j=1}^{p} (\lambda + \omega_j - 1) = \prod_{j=1}^{p} \omega_j \mu^p \lambda^{p-q}.$$
(3.9)

### 4. Symmetric SOR method

Each iteration step of the Symmetric SOR (SSOR) method consists of two semi-iterations the first of which is a usual (forward) SOR iteration followed by a backward SOR iteration, namely an SOR where the roles of L and U have been interchanged. More specifically

$$x^{(m+(1/2))} = (D - \omega L)^{-1} [(1 - \omega)D + \omega U] x^{(m)} + \omega (D - \omega L)^{-1} b,$$
  

$$x^{(m+1)} = (D - \omega U)^{-1} [(1 - \omega)D + \omega L] x^{(m+(1/2))} + \omega (D - \omega U)^{-1} b.$$
(4.1)

An elimination of  $x^{(m+(1/2))}$  from the above equations yields

$$x^{(m+1)} = \mathscr{S}_{\omega} x^{(m)} + c_{\omega}, \quad k = 0, 1, 2, \dots, \ x^{(0)} \in \mathbb{C}^n \text{ arbitrary}$$
 (4.2)

with

$$\mathcal{S}_{\omega} := (D - \omega U)^{-1} [(1 - \omega)D + \omega L] (D - \omega L)^{-1} [(1 - \omega)D + \omega U],$$
  

$$c_{\omega} := \omega (2 - \omega) (D - \omega U)^{-1} D (D - \omega L)^{-1} b.$$
(4.3)

The SSOR method was introduced by Sheldon and constitutes a generalization of the method introduced previously by Aitken for  $\omega = 1$  (see [93]).

Statements analogous to Kahan's theorem and also to Reich-Ostrowski-Varga's theorem of the SOR method can be proved. Specifically we have:

**Theorem 4.1.** A necessary condition for the SSOR method defined in (4.2)–(4.3) to converge is  $|\omega - 1| < 1$ . For  $\omega \in \mathbb{R}$  the condition becomes  $\omega \in (0, 2)$ .

**Theorem 4.2.** Let  $A \in \mathbb{C}^{n,n}$  be Hermitian with positive diagonal elements. Then for any  $\omega \in (0,2)$  the SSOR iteration matrix  $\mathscr{S}_{\omega}$  has real nonnegative eigenvalues. In addition, if A is positive definite then the SSOR method converges. Conversely, if the SSOR method converges and  $\omega \in \mathbb{R}$  then  $\omega \in (0,2)$  and A is positive definite.

*Note*: Compared to SOR, SSOR requires more work per iteration and in general converges slower. Due to its symmetry, however, it can be combined with the semi-iterative method to produce other methods with nice convergence properties (see, e.g. [93]).

For 2-cyclic consistently ordered matrices the first functional relationship between the eigenvalues  $\mu$  and  $\lambda$  of the associated Jacobi and SSOR iteration matrices was given by D'Sylva and Miles [10] and Lynn [55] and is the following:

$$(\lambda - (\omega - 1)^2)^2 = \omega^2 (2 - \omega)^2 \mu^2 \lambda.$$
(4.4)

It can be found that for A as in (2.2) the optimal  $\omega$ ,  $\omega_b = 1$ . Then  $\rho(\mathscr{S}_1) = \rho(\mathscr{S}_1) = \rho^2(J_2)$ .

In case A is block two-cyclic consistently ordered and  $\sigma(J_2)$  lies in the open infinite unit strip S one can develop a Young-Eidson's-type algorithm for the determination of the optimal parameter  $\omega_b$  and subsequently of  $\rho(\mathscr{L}_{\omega_b})$  (see [32]).

The functional eigenvalue relationship in the case of block p-cyclic consistently ordered matrices was discovered by Varga, Niethammer and Cai [86], who obtained the relationship

$$(\lambda - (\omega - 1)^2)^p = \omega^p (2 - \omega)^2 \mu^p \lambda (\lambda - (\omega - 1))^{p-2}.$$
(4.5)

The relationship above was then extended by Chong and Cai [8] to cover the class of GCO(q, p-q) matrices in (2.17) to

$$(\lambda - (\omega - 1)^2)^p = \omega^p (2 - \omega)^{2q} \mu^p \lambda^q (\lambda - (\omega - 1))^{p - 2q}.$$
(4.6)

Optimal values of the SSOR method for spectra  $\sigma(J_p^p)$  nonnegative or nonpositive for any  $p \ge 3$  cannot be found anywhere in the literature except in a very recent article [37], where a number of cases are covered analytically and experimentally and a number of conjectures based on strong numerical evidence are made.

As for the SOR method also for the SSOR method researchers have tried to find regions of convergence for various classes of matrices. Thus Neumaier and Varga [63] determined for the class of *H*-matrices the region in the  $(\rho(|D^{-1}(L+U)|), \omega)$ -plane for which the SSOR method converges. Motivated by their work, Hadjidimos and Neumann [29], using Rouché's theorem, studied and determined the region of convergence of the SSOR method in the  $(\rho(J_p), \omega)$ -plane for each value of  $p \ge 3$  for the class of the *p*-cyclic consistently ordered matrices. It is noted that the intersection of all the domains obtained for all  $p \ge 3$  is the same domain as the one obtained by Neumaier and Varga for the whole class of the *H*-matrices with the only difference being that in the latter case the domain is obtained in the  $(\rho(|D^{-1}(L+U)|), \omega)$ -plane. An extension of the work in [29] is

given in [30], where GCO(q, p-q) matrices for each possible value of  $l=q/p < \frac{1}{2}$  and each  $p \ge 3$  are considered. Finally, in [35] the domains of convergence of the SSOR method for the class of *p*-cyclic consistently ordered matrices for each  $p \ge 3$  in the  $(\rho(J_p), \omega)$ -plane is determined in the two cases of the nonnegative and nonpositive spectra  $\sigma(J_p^p)$ .

### 4.1. Unsymmetric SOR method

The unsymmetric SOR (USSOR) method differs from the SSOR method in the second (backward) SOR part of each iteration where a different relaxation factor is used (see [10,55,93]). It consists of the following two half steps:

$$x^{(m+(1/2))} = (D - \omega_1 L)^{-1} [(1 - \omega_1)D + \omega_1 U] x^{(m)} + \omega_1 (D - \omega_1 L)^{-1} b,$$
  

$$x^{(m+1)} = (D - \omega_2 U)^{-1} [(1 - \omega_2)D + \omega_2 L] x^{(m+(1/2))} + \omega_2 (D - \omega_2 U)^{-1} b.$$
(4.7)

On elimination of  $x^{(m+(1/2))}$  it is produced

$$x^{(m+1)} = \mathscr{S}_{\omega_1,\omega_2} x^{(m)} + c_{\omega_1,\omega_2}, \quad k = 0, 1, 2, \dots, \ x^{(0)} \in \mathbb{C}^n \text{ arbitrary}$$
(4.8)

with

$$\mathscr{S}_{\omega_{1},\omega_{2}} := (D - \omega_{2}U)^{-1}[(1 - \omega_{2})D + \omega_{2}L](D - \omega_{1}L)^{-1}[(1 - \omega_{1})D + \omega_{1}U],$$
  

$$c_{\omega_{1},\omega_{2}} := (\omega_{1} + \omega_{2} - \omega_{1}\omega_{2})(D - \omega_{2}U)^{-1}D(D - \omega_{1}L)^{-1}b.$$
(4.9)

Theory analogous to that of the SSOR method can be developed and the interested reader is referred to [92,93].

The only point we would like to make is that for *p*-cyclic consistently ordered and for GCO(q, p-q) matrices *A* there are functional eigenvalue relationships connecting the eigenvalue spectra of the Jacobi and of the USSOR iteration matrices. They were discovered by Saridakis [75] and the most general one below by Li and Varga [54]

$$(\lambda - (1 - \omega_1)(1 - \omega_2))^p$$

$$= (\omega_1 + \omega_2 - \omega_1 \omega_2)^{2q} \mu^p \lambda^q (\lambda \omega_1 + \omega_2 - \omega_1 \omega_2)^{|\zeta_L| - q} (\lambda \omega_2 + \omega_1 - \omega_1 \omega_2)^{|\zeta_U| - q},$$
(4.10)

where  $|\zeta_L|$  and  $|\zeta_U|$  are the cardinalities of the sets  $\zeta_L$  and  $\zeta_U$ , which are the two disjoint subsets of  $P \equiv \{1, 2, ..., p\}$  associated with the cyclic permutation  $\sigma = (\sigma_1, \sigma_2, ..., \sigma_p)$  as these are defined in [54].

## 5. Accelerated overrelaxation (AOR) method

A technique that sometimes "accelerates" the convergence of a convergent iterative scheme or makes it converge if it diverges is the introduction of an "acceleration" or "relaxation" parameter  $\omega \in \mathbb{C} \setminus \{0\}$  as follows. Based on (1.2) we consider as a new preconditioner the matrix  $M_{\omega} = \frac{1}{\omega}M$ . It is then readily seen that the new iterative scheme is given by

$$x^{(m+1)} = T_{\omega} x^{(m)} + c_{\omega}, \quad T_{\omega} := (1-\omega)I + \omega T, \quad c_{\omega} := \omega c.$$

$$(5.1)$$

The parameter  $\omega$  is called the *extrapolation* parameter and the corresponding scheme is the *extrapolated* of the original one. The most general algorithm to determine the best extrapolation parameter

 $\omega \in \mathbb{C} \setminus \{0\}$  under some basic assumptions regarding some information on the spectrum  $\sigma(T)$  of T can be found in [25] (see also the references cited therein and also [71] which treats a similar case).

Exploiting the idea of extrapolation a two-parameter SOR-type iterative method was introduced in [24]. It was called Accelerated overrelaxation (AOR) method and can be defined as follows:

$$x^{(m+1)} = \mathscr{L}_{r,\omega} x^{(m)} + c_{r,\omega}, \quad m = 0, 1, 2, \dots, \ x^{(0)} \in \mathbb{C}^n \text{ arbitrary},$$
(5.2)

where

$$\mathscr{L}_{r,\omega} := (D - rL)^{-1} [(1 - \omega)D + (\omega - r)L + \omega U], \quad c_{r,\omega} := \omega (D - rL)^{-1} b.$$
(5.3)

It can be readily proved that the AOR method is the union of the Extrapolated Jacobi method (r=0) with extrapolation parameter  $\omega$  and of the Extrapolated SOR method  $(r \neq 0)$  with extrapolation parameter  $s = \omega/r$  of an SOR method with relaxation factor r. It is obvious that the Jacobi, the Gauss-Seidel, the SOR method and their extrapolated counterparts can be considered as special cases of the AOR method.

*Note*: Niethammer [65] refers to a similar to the AOR method that was introduced in a series of papers by Sisler [76–78].

For Hermitian matrices  $A \in \mathbb{C}^{n,n}$  a statement analogous to the Reich–Ostrowski–Varga theorem holds for the AOR method as well. Here is one version of it given in [42].

**Theorem 5.1.** Let  $A = D - E - E^H \in \mathbb{C}^{n,n}$  be Hermitian, D be Hermitian and positive definite, det $(D-rE) \neq 0$ ,  $\forall \omega \in (0,2)$  and  $r \in (\omega + (2-\omega)/\mu_m, \omega + (2-\omega)/\mu_M)$  with  $\mu_m < 0 < \mu_M$  being the smallest and the largest eigenvalues of  $D^{-1}(E + E^H)$ . Then,  $\rho(\mathscr{L}_{r,\omega}) < 1$  iff A is positive definite. (Note: Except for the restrictions in the statement the matrices  $D, E \in \mathbb{C}^{n,n}$  can be any matrices.)

Many more theoretical results can be proved in case A is p-cyclic consistently ordered. For example, if A is 2-cyclic consistently ordered and  $\sigma(J_2^2)$  is either nonnegative or nonpositive then optimal parameters for the AOR method can be derived. They are better than the optimal ones for the corresponding SOR method if some further assumptions are satisfied. These results can be found in [1,62,27].

**Theorem 5.2.** Under the notation and the assumptions so far, let  $\underline{\mu}$  and  $\overline{\mu}$  denote the absolutely smallest and the absolutely largest of the eigenvalues of the Jacobi iteration matrix  $J_2$  of a 2-cyclic consistently ordered matrix A. Then: For  $\sigma(J_2^2)$  nonnegative and  $0 < \underline{\mu} < \overline{\mu} < 1$  if  $1 - \underline{\mu}^2 < (1 - \overline{\mu}^2)^{1/2}$  the optimal parameters of the AOR method are given by the expressions

$$r_b = \frac{2}{1 + (1 - \bar{\mu}^2)^{1/2}}, \quad \omega_b = \frac{1 - \underline{\mu}^2 + (1 - \bar{\mu}^2)^{1/2}}{(1 - \bar{\mu}^2)(1 + (1 - \bar{\mu}^2)^{1/2})},$$
(5.4)

$$\rho(\mathscr{L}_{r_{b},\omega_{b}}) = \frac{\underline{\mu}(\bar{\mu}^{2} - \underline{\mu}^{2})^{1/2}}{(1 - \underline{\mu}^{2})^{1/2}(1 + (1 - \bar{\mu}^{2})^{1/2})}.$$

Furthermore, for  $0 < \mu = \overline{\mu} < 1$  there are two pairs of optimal parameters

$$(r_b, \omega_b) = \left(\frac{2}{1 + \varepsilon(1 - \bar{\mu}^2)^{1/2}}, \frac{\varepsilon}{(1 - \bar{\mu}^2)^{1/2}}\right), \quad \varepsilon = \pm 1,$$
(5.5)

both of which give  $\rho(\mathscr{L}_{r_b,\omega_b}) = 0$ . For  $\sigma(J_2^2)$  nonpositive and if  $(1 + \overline{\mu}^2)^{1/2} < 1 + \underline{\mu}^2$  the optimal parameters of the AOR method are given by the expressions

$$r_b = \frac{2}{1 + (1 + \bar{\mu}^2)^{1/2}}, \quad \omega_b = \frac{1 + \underline{\mu}^2 + (1 + \bar{\mu}^2)^{1/2}}{(1 + \bar{\mu}^2)(1 + (1 + \bar{\mu}^2)^{1/2})},$$
(5.6)

$$\rho(\mathscr{L}_{r_b,\omega_b}) = \frac{\underline{\mu}(\bar{\mu}^2 - \underline{\mu}^2)^{1/2}}{(1 + \underline{\mu}^2)^{1/2}(1 + (1 + \bar{\mu}^2)^{1/2})}$$

Again for  $0 < \mu = \overline{\mu}$  there are two pairs of optimal parameters

$$(r_b, \omega_b) = \left(\frac{2}{1 + \varepsilon(1 + \bar{\mu}^2)^{1/2}}, \frac{\varepsilon}{(1 + \bar{\mu}^2)^{1/2}}\right), \quad \varepsilon = \pm 1,$$
(5.7)

both of which give  $\rho(\mathscr{L}_{r_b,\omega_b}) = 0$ .

*Notes*: (i) The assumptions on  $\underline{\mu}$  and  $\overline{\mu}$  of Theorem 5.2 are very demanding. Practically, to have an optimal AOR better than the optimal SOR,  $\underline{\mu}$  must be "different" from 0 and "very close" to  $\overline{\mu}$ . It is not known whether these assumptions are true for any *real life* problem. (ii) The assumptions  $\underline{\mu} = \overline{\mu} \neq 0$  indicate that the Jacobi iteration matrix  $J_2$  has only two distinct, of opposite sign and of the same multiplicity eigenvalues. This leads directly to the fact that all eigenvalues of  $\mathscr{L}_{r_b,\omega_b}$  are zero.

Methods analogous to the MSOR, SSOR, etc, have been developed and thus MAOR [39], SAOR [41], etc., can be found in the literature. Here we only give the functional eigenvalue relationship for GCO(q, p-q) matrices that generalizes many other similar equations and especially the one by Saridakis [74] for the AOR method

$$\prod_{j=1}^{p} (\lambda + \omega_j - 1) = \prod_{j=1}^{q} \omega_j \mu^p \prod_{j=q+1}^{p} (\omega_j - r_j + r_j \lambda).$$
(5.8)

### 6. Linear (non-)stationary higher-order(degree), semi-iterative methods and SOR

All the iterative methods studied so far are *linear stationary first-order(degree)* ones. The term *stationary* means that any parameters involved in the iterative scheme are kept fixed during the iterations, *first order(degree)* means that the new iteration  $x^{(m+1)}$  depends only on the previous one  $x^{(m)}$  and *linear* that  $x^{(m+1)}$  is a linear function of  $x^{(m)}$ .

Among the linear non-stationary first-order methods the adaptive SOR method is one of the most important and most popular in practical problems and is now incorporated in all the well-known computer packages like, e.g., ITPACK [48]. For an introduction to the adaptive SOR which was first considered for real symmetric positive definite 2-cyclic consistently ordered matrices but now is of a more general application the reader is referred to [44].

A class of linear stationary second-order methods, where each new iteration depends linearly on the two previous ones, that can handle effectively linear systems Ax = b, rewritten equivalently as x = Tx + c, where  $\sigma(T)$  is assumed to be enclosed by an ellipse lying strictly to the left of the line Re z < 1 of the complex plane, are described in [93]. In [93] the reader can find interesting results, when A is 2-cyclic consistently ordered with  $\sigma(J_2^2)$  nonnegative and  $\rho(J_2) < 1$ , as well as some other interesting references.

A similar linear stationary second-order method is also given by Manteuffel in [59]. This method is derived directly from a linear non-stationary second-order one [56–58] which, in turn, is developed by using translated and scaled Chebyshev polynomials in the complex plane. It is worth pointing out that a 2-cyclic MSOR method is equivalent in the *Chebyshev sense* to a linear stationary second-order one and therefore "optimal" values of its parameters can be found by using either Manteuffel's algorithm [40] or a "continuous" analog of it [2].

There is also a class of iterative methods that are called Semi-Iterative and are described in a very nice way in [83] (see also [93]). In [83] it is shown that if one uses Chebyshev polynomials and bases one's analysis on them one can derive a linear non-stationary second-order scheme with very nice properties. The study of semi-iterative methods seems to have begun in [68] followed by a number of papers among which are [12–15]. Especially in the last two (see also [9]) when as the matrix T in x = Tx + c, the SOR iteration matrix, associated with a 2-cyclic consistently ordered matrix A with  $\sigma(J_2^2)$  nonnegative and  $\rho(J_2) < 1$ , is considered, it is proved that it converges for all  $\omega \in (-\infty, 2/(1 - (1 - \rho^2(J_2))^{1/2})) \setminus \{0\}$  which constitutes an amazingly wider range than that of the SOR method!

### 7. Operator relationships for generalized (q, p - q)-cyclic consistently ordered matrices

Before we conclude this article we would like to mention one more point. As we have seen so far in case A is a GCO(q, p - q) matrix there is always a functional relationship that connects the eigenvalues of the Jacobi iteration matrix and the eigenvalues of the iteration operator associated with any of the methods considered. E.g., SOR, MSOR, SSOR, USSOR, AOR, MAOR, SAOR, etc. However, it seems that exactly the same functional relationship holds for the iteration operators involved.

The first who observed that such a relationship held was Young and Kincaid [95] (see also [93]), who proved that for a 2-cyclic consistently ordered matrix there holds

$$\left(\mathscr{L}_{\omega} + (\omega - 1)I\right)^2 = \omega^2 J_2^2 \mathscr{L}_{\omega}.$$
(7.1)

Using this equation as a starting point a discussion started whether similar relationships held as well for other functional relationships associated with operators of a *p*-cyclic consistently ordered matrix. The theory behind the proof of such relationships is basically graph theory and combinatorics. The most general relationships that can be found in the literature are the following two which refer to the MSOR and to the USSOR methods associated with a GCO(q, p - q) matrix, respectively,

$$\prod_{j=1}^{p} (\mathscr{L}_{\Omega} + (\omega_j - 1)I) = (\Omega J_{q, p-q})^{p} \mathscr{L}_{\Omega}^{p-q},$$
(7.2)

where  $\Omega = \text{diag}(\omega_1 I_{n_1}, \omega_2 I_{n_2}, \dots, \omega_p I_{n_p}), \sum_{i=1}^p n_i = n$ , and

$$(\mathscr{G}_{\omega_{1},\omega_{2}} - (1 - \omega_{1})(1 - \omega_{2})I)^{p} = (\omega_{1} + \omega_{2} - \omega_{1}\omega_{2})^{2q}J_{q,p-q}^{p}\mathscr{G}_{\omega_{1},\omega_{2}}^{q}(\omega_{1}\mathscr{G}_{\omega_{1},\omega_{2}} + \omega_{2}(1 - \omega_{1})I)^{|\zeta_{L}|-q} \times (\omega_{2}\mathscr{G}_{\omega_{1},\omega_{2}} + \omega_{1}(1 - \omega_{2})I)^{|\zeta_{U}|-q},$$
(7.3)

where for the various notations see previous section and [33,54,70]. From these relationships simpler ones can be obtained, e.g., for the *p*-cyclic consistently ordered SOR, for the GCO(q, p-q) SSOR, and also the same relationships can be extended to cover the *p*-cyclic AOR, MAOR, SAOR, etc., cases.

Use of the functional relationships can be made in order to transform a one-step iterative scheme into another equivalent p-step one. For more details the reader is referred to the references of this section and also to [68].

## 8. Final remarks

In this article an effort was made to present the SOR method and some of its properties together with some other methods closely related to it. For the methods presented the most common classes of matrices *A* that led to some interesting results were considered. Of course, not all of the well-known classes of matrices *A* was possible to cover. For example, matrices like strictly diagonally dominant, irreducibly diagonally dominant, etc., were left out.

Finally, we mentioned only very briefly the role of the SOR, SSOR, etc, methods as preconditioners for the class of semi-iterative (see [93]) and we did not examine at all their roles for the class of conjugate gradient methods (see [23]). This was done purposefully for otherwise the basic theory of the other classes of methods involved should have been analyzed to some extent and this would be beyond the scope of the present article. On the other hand, it is the author's opinion that other expert researchers in the corresponding areas will cover these subjects in a much better and more efficient way in their articles in the present volume.

### Acknowledgements

The author cordially thanks Dr. Dimitrios Noutsos of the University of Ioannina for constructive criticism on a first draft of this article. He is most grateful to Professor Richard S. Varga for many helpful comments and suggestions as well as for material he kindly provided from the revised version of his "matrix iterative analysis" book and also to an anonymous referee, who made a number of points and provided some basic bibliography that led to the improvement of the previous version of this article.

### References

- G. Avdelas, A. Hadjidimos, Optimum accelerated overrelaxation method in a special case, Math. Comp. 36 (1981) 183–187.
- [2] G. Avdelas, A. Hadjidimos, Optimal 2-cyclic MSOR for "Bowtie" spectra and the "Continuous" Manteuffel algorithm, Linear Algebra Appl. 265 (1997) 29–54.

- [3] A. Berman, R.J. Plemmons, Nonnegative Matrices in the Mathematical Sciences, SIAM, Philadelphia, PA, 1994.
- [4] J.J. Buoni, M. Neumann, R.S. Varga, Theorems of Stein-Rosenberg type III, the singular case, Linear Algebra Appl. 42 (1982) 183–198.
- [5] J.J. Buoni, R.S. Varga, Theorems of Stein–Rosenberg type, in: R. Ansorge, K. Glashoff, B. Werner (Eds.), Numerical Mathematics, Birkhäuser, Basel, 1979, pp. 65–75.
- [6] J.J. Buoni, R.S. Varga, Theorems of Stein-Rosenberg type II, optimal paths of relaxation in the complex plane, in: M.H. Schultz (Ed.), Elliptic Problem Solvers, Academic Press, New York, 1981, pp. 231–240.
- [7] R.C.Y. Chin, T.A. Manteuffel, An analysis of block successive overrelaxation for a class of matrices with complex spectra, SIAM J. Numer. Anal. 25 (1988) 564–585.
- [8] L. Chong, D.-Y. Cai, Relationship between eigenvalues of Jacobi and SSOR iterative matrix with p-weak cyclic matrix J. Comput. Math. Coll. Univ. 1 (1985) 79–84 (in Chinese).
- [9] J. Dancis, The optimal  $\omega$  is not best for the SOR iteration method, Linear Algebra Appl. 154–156 (1991) 819–845.
- [10] E. D'Sylva, G.A. Miles, The S.S.O.R. iteration scheme for equations with  $\sigma_1$ -orderings, Comput. J. 6 (1963) 271–273.
- [11] M. Eiermann, W. Niethammer, A. Ruttan, Optimal successive overrelaxation iterative methods for *p*-cyclic matrices, Numer. Math. 57 (1990) 593–606.
- [12] M. Eiermann, W. Niethammer, R.S. Varga, A study of semi-iterative methods for non-symmetric systems of linear equations, Numer. Math. 47 (1985) 503–533.
- [13] M. Eiermann, W. Niethammer, R.S. Varga, Acceleration of relaxation methods for non-Hermitian linear systems, SIAM J. Matrix Anal. Appl. 13 (1991) 979–991.
- [14] M. Eiermann, R.S. Varga, Is the optimal  $\omega$  best for the SOR iterative method? Linear Algebra Appl. 182 (1993) 257–277.
- [15] M. Eiermann, R.S. Varga, Optimal semi-iterative methods applied to SOR in the mixed case, in: L. Reichel, A. Ruttan, R.S. Varga (Eds.), Numerical Linear Algebra, Walter de Gruyter, New York, 1993, pp. 47–73.
- [16] S.P. Frankel, Convergence rates of iterative treatments of partial differential equations, Math. Tables Aids Comput. 4 (1950) 65–75.
- [17] S. Galanis, A. Hadjidimos, Best cyclic repartitioning for optimal SOR convergence, SIAM J. Matrix Anal. Appl. 13 (1992) 102–120.
- [18] S. Galanis, A. Hadjidimos, D. Noutsos, On the equivalence of the k-step iterative Euler methods and successive overrelaxation (SOR) methods for k-cyclic matrices, Math. Comput. Simulation 30 (1988) 213–230.
- [19] S. Galanis, A. Hadjidimos, D. Noutsos, Optimal *p*-cyclic SOR for complex spectra, Linear Algebra Appl. 263 (1997) 233–260.
- [20] S. Galanis, A. Hadjidimos, D. Noutsos, A Young-Eidson's type algorithm for complex *p*-cyclic SOR spectra, Linear Algebra Appl. 286 (1999) 87–106.
- [21] S. Galanis, A. Hadjidimos, D. Noutsos, M. Tzoumas, On the optimum relaxation factor associated with p-cyclic matrices, Linear Algebra Appl. 162–164 (1992) 433–445.
- [22] G.H. Golub, J. de Pillis, Toward an effective two-parameter method, in: D.R. Kincaid, L. Hayes (Eds.), Iterative Methods for Large Linear Systems, Academic Press, New York, 1990, pp. 107–118.
- [23] A. Greenbaum, Iterative Methods for Solving Linear Systems, SIAM, Philadelphia, PA, 1997.
- [24] A. Hadjidimos, Accelerated overrelaxation method, Math. Comp. 32 (1978) 149–157.
- [25] A. Hadjidimos, The optimal solution to the problem of complex extrapolation of a first order scheme, Linear Algebra Appl. 62 (1984) 241–261.
- [26] A. Hadjidimos, On the optimization of the classical iterative schemes for the solution of complex singular linear systems, SIAM J. Algebra Discrete Methods 6 (1985) 555–566.
- [27] A. Hadjidimos, A survey of the iterative methods for the solution of linear systems by extrapolation, relaxation and other techniques, J. Comput. Appl. Math. 20 (1987) 37–51.
- [28] A. Hadjidimos, X.-Z. Li, R.S. Varga, Application of the Schur-Cohn theorem to precise convergence domains for the cyclic SOR iterative method, unpublished manuscript, 1985.
- [29] A. Hadjidimos, M. Neumann, Precise domains of convergence for the block SSOR method associated with *p*-cyclic matrices, BIT 29 (1989) 311–320.
- [30] A. Hadjidimos, M. Neumann, Convergence domains of the SSOR method for generalized consistently ordered matrices, J. Comput. Appl. Math. 33 (1990) 35–52.

- [31] A. Hadjidimos, M. Neumann, Euclidean norm minimization of the SOR operators, SIAM J. Matrix Anal. Appl. 19 (1998) 191–204.
- [32] A. Hadjidimos, D. Noutsos, The Young-Eidson algorithm: applications and extensions, SIAM J. Matrix Anal. Appl. 11 (1990) 620–631.
- [33] A. Hadjidimos, D. Noutsos, On a matrix identity connecting iteration operators associated with a p-cyclic matrix, Linear Algebra Appl. 182 (1993) 157–178.
- [34] A. Hadjidimos, D. Noutsos, M. Tzoumas, Exact SOR convergence regions for a general class of p-cyclic matrices, BIT 35 (1995) 469–487.
- [35] A. Hadjidimos, D. Noutsos, M. Tzoumas, On the exact *p*-cyclic SSOR convergence domains, Linear Algebra Appl. 232 (1996) 213–236.
- [36] A. Hadjidimos, D. Noutsos, M. Tzoumas, On the convergence domains of the *p*-cyclic SOR, J. Comput. Appl. Math. 72 (1996) 63–83.
- [37] A. Hadjidimos, D. Noutsos, M. Tzoumas, Towards the determination of the optimal *p*-cyclic SSOR, J. Comput. Appl. Math. 90 (1997) 1–14.
- [38] A. Hadjidimos, R.J. Plemmons, Optimal p-cyclic SOR, Numer. Math. 67 (1994) 475-490.
- [39] A. Hadjidimos, A. Psimarni, A.K. Yeyios, On the convergence of the modified accelerated overrelaxation (MAOR) method, Applied Numer. Math. 10 (1992) 115–127.
- [40] A. Hadjidimos, Y.G. Saridakis, Modified successive overrelaxation (MSOR) and equivalent 2-step iterative methods for collocation matrices, J. Comput. Appl. Math. 42 (1992) 375–393.
- [41] A. Hadjidimos, A. Yeyios, Symmetric accelerated overrelaxation (SAOR) method, Math. Comput. Simulation XXIV (1982) 72–76.
- [42] A. Hadjidimos, A. Yeyios, On some extensions of the accelerated overrelaxation (AOR) theory, Internat. J. Math. Math. Sci. 5 (1982) 49–60.
- [43] A. Hadjidimos, A.K. Yeyios, Some recent results on the modified SOR theory, Linear Algebra Appl. 154–156 (1991) 5–21.
- [44] L.A. Hageman, D.M. Young, Applied Iterative Methods, Academic Press, New York, 1981.
- [45] P. Henrici, Applied and Computational Complex Analysis, Wiley, New York, 1974.
- [46] W. Kahan, Gauss–Seidel methods of solving large systems of linear equations, Doctoral Thesis, University of Toronto, Toronto, Canada, 1958.
- [47] D.R. Kincaid, Norms of the successive overrelaxation method, Math. Comp. 118 (1972) 345–357.
- [48] D.R. Kincaid, J.R. Respess, D.M. Young, R.G. Grimes, ITPACK 2C: a fortran package for solving large sparse linear systems by adaptive accelerated iterative methods, ACM Trans. Math. Software 8 (1992) 302–322.
- [49] D.R. Kincaid, D.M. Young, The modified successive overrelaxation method with fixed parameters, Math. Comp. 119 (1972) 705–717.
- [50] G. Kjellberg, On the convergence of the successive over-relaxation applied to a class of linear systems of equations with complex eigenvalues, Ericsson Technics Stockholm 2 (1958) 245–258.
- [51] K. Kontovasilis, R.J. Plemmons, W.J. Stewart, Block cyclic SOR for Markov chains with p-cyclic infinitesimal generator, Linear Algebra Appl. 154–156 (1991) 145–223.
- [52] B. Kredell, On complex successive overrelaxation, BIT 2 (1962) 143-152.
- [53] Y.A. Kuznetsov, in: Matrix iterative methods in subspaces, Proceedings of the International Congress of Mathematicians, Warszawa, August 16–24, 1983, North Holland, Amsterdam, 1984.
- [54] X. Li, R.S. Varga, A note on the SSOR and USSOR iterative methods applied to *p*-cyclic matrices, Numer. Math. 56 (1989) 109–121.
- [55] M.S. Lynn, On the equivalence of SOR, SSOR and USSOR as applied to  $\sigma_1$ -ordered systems of linear equations, Comput. J. 7 (1964) 72–75.
- [56] T.A. Manteuffel, An iterative method for solving nonsymmetric linear systems with dynamic estimation of parameters, UTUCSD-R-75, Department of Computer Science, University of Illinois, Urbana, IL, 1975.
- [57] T.A. Manteuffel, The Tchebychev iteration for non-symmetric linear systems, Numer. Math. 28 (1977) 307–327.
- [58] T.A. Manteuffel, Adaptive procedure for estimating parameters for the non-symmetric Tchebychev iteration, Numer. Math. 31 (1978) 183–208.
- [59] T.A. Manteuffel, Optimal parameters for linear second-degree stationary iterative methods, SIAM J. Numer. Anal. 19 (1982) 833–839.

- [60] T.L. Markham, M. Neumann, R.J. Plemmons, Convergence of a direct-iterative method for large-scale least squares problems, Linear Algebra Appl. 69 (1985) 155–167.
- [61] L.K. McDowell, Variable successive overrelaxation, Report No. 244, Department of Computer Sciences, University of Illinois, Urbana, IL.
- [62] N.M. Missirlis, Convergence theory of extrapolated iterative methods for a certain class of non-symmetric linear systems, Numer. Math. 45 (1984) 447–458.
- [63] A. Neumaier, R.S. Varga, Exact convergence and divergence domains for the symmetric successive overrelaxation iterative (SSOR) method applied to *H*-matrices, Linear Algebra Appl. 58 (1984) 261–272.
- [64] N.K. Nichols, L. Fox, Generalized consistent ordering and the optimum successive over-relaxation factor, Numer. Math. 13 (1969) 425–433.
- [65] W. Niethammer, On different splittings and the associated iteration methods, SIAM J. Numer. Anal. 16 (1979) 186–200.
- [66] W. Niethammer, Relaxation bei Matrizen mit der Eigenschaft "A", Z. Angew. Math. Mech. 44 (1964) T49-T52.
- [67] W. Niethammer, J. de Pillis, R.S. Varga, Convergence of block iterative methods applied to sparse least squares problems, Linear Algebra Appl. 58 (1984) 327–341.
- [68] W. Niethammer, R.S. Varga, The analysis of k-step iterative methods for linear systems from summability theory, Numer. Math. 41 (1983) 177–206.
- [69] D. Noutsos, Optimal stretched parameters for the SOR iterative method, J. Comput. Appl. Math. 48 (1993) 293–308.
- [70] D. Noutsos, An operator relation of the USSOR and the Jacobi iteration matrices of a *p*-cyclic matrix, SIAM J. Matrix. Anal. Appl. 17 (1996) 515–529.
- [71] G. Opfer, G. Schober, Richardson's iterations for nonsymmetric matrices, Linear Algebra Appl. 58 (1984) 343-361.
- [72] D.J. Pierce, A. Hadjidimos, R.J. Plemmons, Optimality relationships for p-cyclic SOR, Numer. Math. 56 (1990) 635–643.
- [73] D.B. Russell, On obtaining solutions to Navier–Stokes equations with automatic digital computers, Aeronautical Research Council Report R & M 3331 Engineering Laboratory, Oxford, 1963.
- [74] Y.G. Saridakis, Generalized consistent orderings and the accelerated overrelaxation method, BIT 26 (1986) 369–376.
- [75] Y.G. Saridakis, On the analysis of the unsymmetric overrelaxation method when applied to *p*-cyclic matrices, Numer. Math. 49 (1986) 461–473.
- [76] M. Sisler, Uber ein Zweiparametrigen Iterationsverfahrens, Apl. Mat. 18 (1973) 325-332.
- [77] M. Sisler, Uber die Optimierung eines Zweiparametrigen Iterationsverfahrens, Apl. Mat. 20 (1975) 126-142.
- [78] M. Sisler, Bemerkungen zur Optimierung eines Zweiparametrigen Iterationsverfahrens, Apl. Mat. 21 (1976) 213–220.
   [79] R.V. Southwell, Relaxation Methods in Theoretical Physics, Clarendon Press, Oxford, 1946.
- [79] K.V. Southwen, Relaxation Methods in Theoretical Physics, Clarendon Press, Oxford, 1946
- [80] D.O. Tall, Functions of a Complex Variable, Library of Mathematics, Routledge & Kegan Paul, London, 1970.
- [81] P.J. Taylor, A generalisation of systematic relaxation methods for consistently ordered matrices, Numer. Math. 13 (1969) 377–395.
- [82] R.S. Varga, p-cyclic matrices: a generalization of the Young-Frankel successive overrelaxation scheme, Pacific J. Math. 9 (1959) 617–628.
- [83] R.S. Varga, Matrix Iterative Analysis, Prentice-Hall, Englewood Cliffs, NJ, 1962.
- [84] R.S. Varga, Extensions of the successive overrelaxation theory with applications to finite element approximations, in: J.H.H. Miller (Ed.), Topics in Numerical Analysis, Academic Press, New York, 1973, pp. 242–247.
- [85] R.S. Varga, Personal communication (also, revision of [83], 1999).
- [86] R.S. Varga, W. Niethammer, D.Y. Cai, p-cyclic matrices and the symmetric successive overrelaxation method, Linear Algebra Appl. 58 (1984) 425–439.
- [87] J.H. Verner, M.J.M. Bernal, On generalizations of the theory of consistent orderings for successive overrelaxation methods, Numer. Math. 12 (1968) 215–222.
- [88] P. Wild, W. Niethammer, Over and underrelaxation for linear systems with weakly cyclic Jacobi matrices of index *p*, Linear Algebra Appl. 91 (1987) 29–52.
- [89] E.E. Wrigley, On accelerating the Jacobi method for solving simultaneous equations by Chebyshev extrapolation when the eigenvalues of the iteration matrix are complex, Comput. J. 6 (1963) 169–176.
- [90] D.M. Young, Iterative methods for solving partial differential equations of elliptic type, Doctoral Thesis, Harvard University, Cambridge, MA, 1950.

- [91] D.M. Young, Iterative methods for solving partial differential equations of elliptic type, Trans. Amer. Math. Soc. 76 (1954) 92–111.
- [92] D.M. Young, Convergence properties of the symmetric and unsymmetric successive overrelaxation methods and related methods, Math. Comp. 112 (1970) 793–807.
- [93] D.M. Young, Iterative Solution of Large Linear Systems, Academic Press, New York, 1971.
- [94] D.M. Young, H.E. Eidson, On the determination of the optimum relaxation factor for the SOR method when the eigenvalues of the Jacobi matrix are complex, Report CNA-1, Center for Numerical Analysis, University of Texas, Austin, TX, 1970.
- [95] D.M. Young, D.R. Kincaid, Norms of the successive overrelaxation method and related methods, Report TNN-94, Computation Center, University of Texas, Austin, TX, 1969.



Journal of Computational and Applied Mathematics 123 (2000) 201-216

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.nl/locate/cam

# On asynchronous iterations

Andreas Frommer<sup>a</sup>, Daniel B. Szyld<sup>b, \*,1</sup>

<sup>a</sup> Fachbereich Mathematik, Bergische Universität GH Wuppertal, Gauss-Strasse 20, 42 097 Wuppertal, Germany <sup>b</sup>Department of Mathematics, Temple University (038-16), 1805 N. Broad Street, Philadelphia, PA 19122-6094,

USA

Received 3 June 1999

### Abstract

Asynchronous iterations arise naturally on parallel computers if one wants to minimize idle times. This paper reviews certain models of asynchronous iterations, using a common theoretical framework. The corresponding convergence theory and various domains of applications are presented. These include nonsingular linear systems, nonlinear systems, and initial value problems. © 2000 Elsevier Science B.V. All rights reserved.

## 1. Introduction

With the advent of parallel computers, many new algorithms were devised or rediscovered for the new architectures. An important concept in the design of parallel algorithms is that of load balancing, which simply means that the work has to be approximately equally distributed among processors. Otherwise, some processors finish their task much earlier than others, and the waiting time (also called idle time) degrades the performance of the algorithm. This concept has been widely accepted as a requirement for efficient algorithms, and has dictated for example that when the geometric domain of a physical problem is divided into subdomains (to be processed by the different processors), each should be of approximately the same size.

In contrast to load balancing, the idea of asynchronous methods is to avoid processor idle time by eliminating as much as possible synchronization points, i.e., points at which a processor must wait for information from other processors. In this way, problems which naturally would decompose into processes of very different size, e.g., those with unstructured meshes, can do so without difficulty. The price one pays for this freedom is that some processors will perform extra computations, and it

\* Corresponding author.

E-mail addresses: frommer@math.uni-wuppertal.de (A. Frommer), szyld@math.temple.edu (D.B. Szyld).

<sup>&</sup>lt;sup>1</sup> Supported by National Science Foundation grants DMS-9625865 and DMS-9973219.

is only when the load is not well balanced, or when communication between the processors is slow, that this approach is advantageous.

Since the publication of the pioneering paper in 1969 by Chazan and Miranker [21], the theory and application of asynchronous iterations has been studied and used by many authors. For early surveys of asynchronous iterative methods, see [3,13,14,33] (see also the recent papers [53,56]).

Asynchronous methods are not considered "mainstream" by many researchers, at least in numerical analysis. We believe that this is so in part because the load balancing requirement is so prevalent in the minds of many practitioners. Nevertheless, asynchronous methods are being increasingly used and studied, particularly so in connection with the use of heterogeneous workstation clusters where the available computational power of each processor becomes unpredictable. Experiments reported in the literature, e.g., in [18,35,42], show practical problems for which the asynchronous parallel times are about half to two-thirds of those reported for synchronous parallel times (which of course are much faster than sequential times); see further Section 4.3. In [18,35], asynchronous solutions of systems of several million variables are reported. In addition, asynchronous iterations are possibly the kind of methods which will allow the next generation of parallel machines to attain the expected potential. These machines are being designed today with thousands of processors.

Let us mention some recent papers where the application of asynchronous iterations to different areas is discussed: to the solution of partial differential equations [1,2,52]; to inverse problems in geophysics and oil exploration [48]; to continuous time Markov chains problems for queueing and stochastic automata networks [18]; to electrical power networks [9]; to network flow [60], to convex programming [58], and other optimization [25,26], and nonlinear problems [7,59,65]; and to singular systems of linear equations [5,50].

The purpose of this paper is to review some of the different models of asynchronous iterations which have been developed during the last three decades, using a common theoretical framework. We give some results on their convergence, and illustrate the use of these models and convergence results in various applications, including the iterative solution of linear systems, nonlinear systems, and initial value problems for systems of ordinary differential equations.

It is outside the scope of the paper to present a complete survey of the state-of-the-art in parallel asynchronous computations. Of the topics not covered we mention a few: analysis and implementation of stopping criteria [24,51] (and also [15]), enclosure methods for nonlinear systems of equations [32,34], the theory of multisplittings for the analysis of asynchronous iterations [17,20,55], and its application to domain decomposition methods using overlapping subdomains [7,35]. Our aim instead is to present a snapshot of some broad class of applications, together with a general theory which applies to them.

To that end, we present, in the next section, general computational and mathematical models representing asynchronous iterations. The computational models correspond to the way the methods are actually programmed in the parallel computers. The mathematical models are tools used to analyze the algorithms. In Section 3 we present very general convergence results which apply to these mathematical models, and in Section 4 we apply these results to specific problems.

## 2. Computational and mathematical models

To start, let us consider a structurally simple and quite general construct. Assume that we are given a product space  $E = E_1 \times \cdots \times E_m$  and an application  $H : E \to E$  whose components are

denoted  $H_i$ , i.e., we have

$$H: E \to E, \quad x = (x_1, \dots, x_m) \to ((Hx)_1, \dots, (Hx)_m), \tag{1}$$

where  $x_i, (Hx)_i = H_i(x) \in E_i$ , i = 1, ..., m. The problem at hand is to find a fixed point of *H*. A standard procedure is to approximate such fixed point by variants of the successive approximation procedure

$$x^{k+1} = H(x^k), \quad k = 0, 1, \dots$$
 (2)

Assume for now that we are working with a (shared memory) parallel computer with p processors  $P_1, \ldots, P_p$  ( $p \le m$ ) and associate a *block* of components  $J_j \subseteq \{1, \ldots, m\}$  with each processor  $P_j$ . Then a parallel variant of the successive approximation procedure (2) can be implemented as follows (pseudocode for processor  $P_j$ ):

### Computational Model 2.1.

until convergence do read x from common memory compute  $x_i^{\text{new}} = H_i(x)$  for  $i \in J_j$ overwrite  $x_i$  in common memory with  $x_i^{\text{new}}$ ,  $i \in J_j$ .

If processors would wait for each other to complete each run through the loop we would indeed get a (parallel synchronous) implementation of the successive approximation scheme (2). Since here processors do not wait, we actually get a much less structured iterative process where, due to different run times for each loop, processors get out of phase. At a given time point, different processors will have achieved different numbers of iterations (the iteration number k in (2) looses its meaning in this context). No idle times occur, since processors never wait for each other.

In order to mathematically analyze the Computational Model 2.1, we now step the iteration counter k by 1 each time x is read from the common memory by some processor  $P_{j(k)}$ . Then this x is made up of components each of which has been written back to memory as the result of the computation belonging to some earlier iteration. We therefore have  $x = (x_1^{s_1(k)}, \ldots, x_m^{s_m(k)})$  with iteration counts  $s_{\ell}(k) \in \mathbb{N}_0$ ,  $\ell = 1, \ldots, m$ , prior to k, indicating the iteration when the  $\ell$ th component just read was computed. A set  $I^k$  is defined indicating which components are computed at the kth iteration, i.e.,  $I^k = J_{j(k)}$ . Using these sets, and under the very weak assumptions (3) explained further below, the Computational Model 2.1 can be modeled mathematically according to the following definition; see, e.g., [33,57],

**Definition 2.2.** For  $k \in \mathbb{N}$ , let  $I^k \subseteq \{1, \ldots, m\}$  and  $(s_1(k), \ldots, s_p(k)) \in \mathbb{N}_0^m$  such that

$$s_{i}(k) \leq k - 1 \quad \text{for } i \in \{1, \dots, m\}, \quad k \in \mathbb{N},$$

$$\lim_{k \to \infty} s_{i}(k) = \infty \quad \text{for } i \in \{1, \dots, m\},$$

$$|\{k \in \mathbb{N} : i \in I^{k}\}| = \infty \quad \text{for } i \in \{1, \dots, m\}.$$
(3)

Given an initial guess  $x^0 \in E = E_1 \times \cdots \times E_m$ , the iteration

$$x_{i}^{k} = \begin{cases} x_{i}^{k-1} & \text{for } i \notin I^{k} \\ H_{i}(x_{1}^{s_{1}(k)}, \dots, x_{m}^{s_{m}(k)}) & \text{for } i \in I^{k}, \end{cases}$$
(4)

is termed an asynchronous iteration (with strategy  $I^k$ ,  $k \in \mathbb{N}$  and delays  $d_i(k) = k - s_i(k)$ ,  $i = 1, ..., n, k \in \mathbb{N}$ ).

The first hypothesis in (3) simply indicates that only components computed earlier (and not future ones) are used in the current approximation. The second one indicates that as the computation proceeds, eventually one reads newer information for each of the components. The third one indicates that no component fails to be updated as time goes on.

This mathematical model goes back at least to Baudet [10], although other authors had equivalent models; see the historical remarks in [57]. Note that Definition 2.2 includes as special cases the classical synchronous successive approximation method (2) ( $s_i(k) = k - 1$ ,  $I^k = \{1, ..., m\}$ ) as well as block Gauss–Seidel-type methods ( $s_i(k) = k - 1$ ,  $I^k = \{k \mod m + 1\}$ ) or symmetric block Gauss–Seidel methods.

Let us mention at this point that asynchronous iterations on *local memory* machines (using message passing to communicate data) are also modeled by Definition 2.2.

The fundamental model (4) has a wide range of applications. Nevertheless, other various extensions to account for more general or more specific situations are possible. For example, some authors impose additional conditions on the sequence of delays  $d_i(k) = k - s_i(k)$  such as being uniformly bounded; some others restrict them in such a way that overlap is not allowed; see some examples of these, e.g., in [57] and the bibliography therein. These additional restrictions appear to be necessary in the convergence theory for the solution of singular linear systems; see [5,41,50].

In several practical situations, the component  $H_i$  of H may be given only implicitly (or it may be expensive to compute) so that we will actually only compute an approximation (which may change at each step k) to  $H_i(x_1^{s_1(k)}, \ldots, x_m^{s_m(k)})$  in (4). We are then in a *non-stationary* setting, which includes in particular the case of *two-stage* iterations (with an "inner" and an "outer" iteration) which can be modeled by making H dependent of the iteration index k, i.e., we have the following process

$$x_{i}^{k} = \begin{cases} x_{i}^{k-1} & \text{for } i \notin I^{k}, \\ H_{i}^{k}(x_{1}^{s_{1}(k)}, \dots, x_{m}^{s_{m}(k)}) & \text{for } i \in I^{k} \end{cases}$$
(5)

with  $H^k: E \to E$ , for  $k \in \mathbb{N}$ , having the same fixed point as H.

One way to study the inner iterations is to consider a "splitting" of the application H of (1) into  $K: E \times E \to E$  such that K(x, x) = H(x), and the following model.

### Computational Model 2.3.

```
until convergence do

read (x) from common memory

set y = x

until convergence do

compute y_i^{\text{new}} = K_i(x, y) for i \in J_j

overwrite x_i in common memory with y_i^{\text{new}}, i \in J_j

set y_i = y_i^{\text{new}} (in local memory).
```

This computational model describes in particular asynchronous methods with *flexible communication* (see [25,44]), in which new information is sent to the other processors as soon as it is computed, even before the inner iterations have converged. A mathematical model for it can be obtained by introducing a second set of delays defined through iteration indices  $r_{\ell}(k) \in \mathbb{N}_0$ ,  $\ell = 1, ..., m$ ,  $k \in \mathbb{N}$ , satisfying the same first two hypotheses in (3), and by considering the following process, which is slightly more general than the one given in [38],

$$x_{i}^{k} = \begin{cases} x_{i}^{k-1} & \text{for } i \notin I^{k}, \\ K_{i}((x_{1}^{s_{1}(k)}, \dots, x_{m}^{s_{m}(k)}), (x_{1}^{r_{1}(k)}, \dots, x_{m}^{r_{m}(k)})) & \text{for } i \in I^{k}. \end{cases}$$
(6)

We note that further generalizations of these mathematical models are possible (and applicable to specific situations), where the domain of the application analogous to H (or K) consists of multiple copies of E, and each component of each copy of E may be subject to different delays; see [31,37,50].

It is crucial to realize that our Computational Models 2.1 and 2.3 do not preclude the blocks  $J_j$  to overlap, i.e., we may have  $J_j \cap J_l \neq \emptyset$  for  $j \neq l$ . This situation cannot be modeled by the expression (1), but our mathematical models (4)–(6) are still applicable. In some instances, it turns out that a certain degree of overlapping together with a scheme for combining different contributions within the overlap will usually accelerate the overall iteration (see, e.g., [7,35]).

### **3.** Convergence theory

A general convergence theorem for the asynchronous iteration (4) is the following result of Bertsekas [12] (see also [61]).

**Theorem 3.1.** Assume that there are sets  $E^k \subseteq E$  which satisfy (a)  $E^k = E_1^k \times \cdots \times E_m^k$ ,  $k \in \mathbb{N}_0$ , (box condition) (b)  $H(E^k) \subseteq E^{k+1} \subseteq E^k$ ,  $k \in \mathbb{N}_0$ , (nested sets condition) (c) there exists  $x^*$  such that

 $y^k \in E^k, \quad k \in \mathbb{N} \Rightarrow \lim_{k \to \infty} y^k = x^*$ 

(synchronous convergence condition).

Then the sequence of asynchronous iterates  $x^k$  from (4) converges to  $x^*$ , the unique fixed point of H, provided assumptions (3) hold.

The idea of the proof is to show that starting in a box  $E^k$ , after some time all components  $x_i$  belong to some  $E_i^{\ell}$ ,  $\ell > k$ , and by collecting them we are now in the box  $E^{k+1}$ . A careful inspection of the proof of this result, e.g., in [12], reveals that we can easily obtain the following corollary for non-stationary iterations.

## Corollary 3.2. Replace (b) in Theorem 3.1 by

(b')  $H^k(E^k) \subseteq E^{k+1} \subseteq E^k, k \in \mathbb{N}_0.$ 

Then the asynchronous nonstationary iterates  $x^k$  from (5) converge to  $x^*$ , the unique common fixed point of all  $H^k$ .

There are several special cases of Theorem 3.1 which merit further discussion. Let us first consider the case where each component space  $E_i$  is a normed linear space  $(E_i, \|\cdot\|_i)$ . Define  $\|\cdot\|_w$  the weighted max-norm on E given as

$$\|x\|_{w} = \max_{i=1}^{m} \frac{\|x_{i}\|_{i}}{w_{i}},$$
(7)

where  $w = (w_1, \ldots, w_m)$  is a positive vector, i.e.,  $w_i > 0$  for  $i = 1, \ldots, m$ .

**Theorem 3.3.** Assume that there exists  $x^* \in E$  such that  $H^k(x^*) = x^*$  for all k. Moreover, assume that there exists  $\gamma \in [0, 1)$  and  $w \in \mathbb{R}^m$  positive, such that for all k we have

$$\|H^{k}(x) - x^{*}\|_{w} \leq \gamma \cdot \|x - x^{*}\|_{w}.$$
(8)

Then the asynchronous (non-stationary) iterates  $x^k$  from (5) converge to  $x^*$ , the unique common fixed point of all  $H^k$ .

For a proof, set  $E^k = \{x \in E : ||x - x^*||_w \leq \gamma^k \cdot ||x^0 - x^*||_w\}$  and apply Corollary 3.2 (see [37]). Different proofs of similar theorems can be found in [28,29]. For the stationary case  $(H^k = H)$  the above theorem is known as El Tarazi's theorem [27].

An even more special case arises in the presence of *P*-contractions. The mapping *H* is called a *P*-contraction with respect to a fixed point  $x^*$ , if there exists a nonnegative matrix  $P \in \mathbb{R}^{m \times m}$  with  $\rho(P) < 1$  such that for all  $x \in E$  we have

$$\begin{pmatrix} \|(Hx)_1 - x_1^*\|_1\\ \vdots\\ \|(Hx)_m - x_m^*\|_m \end{pmatrix} \leqslant P \begin{pmatrix} \|x_1 - x_1^*\|_1\\ \vdots\\ \|x_m - x_m^*\|_m \end{pmatrix},$$

where the inequality in  $\mathbb{R}^m$  is componentwise [47]. It can be shown quite easily that a *P*-contraction with respect to  $x^*$  satisfies the assumption of Theorem 3.3 (*w* has to be taken as the Perron-vector of a positive matrix sufficiently close to *P*). We therefore have

**Corollary 3.4** (Baudet [10]). Assume that each  $H^k$  is a P-contraction with respect to  $x^*$  with P independent of k. Then the asynchronous (nonstationary) iterates  $x^k$  from (5) converge to  $x^*$ , the unique common fixed point of all  $H^k$ .

The contraction conditions considered so far can be somewhat relaxed to account for situations where, instead of (8) one just has

$$x \neq x^* \Rightarrow ||Hx - x^*||_w < ||x - x^*||_w.$$
(9)

This is particularly interesting for certain M-functions and diagonally dominant functions in the sense of Moré [46] (see [30]). We mention here that if the implication in (9) is in both directions, such maps are called *paracontracting* (with respect to the weighted max morm) [28,29,50].

The following is a further generalization of El Tarazi's theorem which is applicable to process (6), and in particular to asynchronous methods with flexible communication [38].

**Theorem 3.5.** Assume that there exists  $x^* \in E$  such that  $K(x^*, x^*) = x^*$ . Moreover, assume, that there exists  $\gamma \in [0, 1)$  and a weighted max norm such that

$$||K(x, y) - x^*||_w \leq \gamma \cdot \max\{||x - x^*||_w, ||y - x^*||_w\} \quad for \ all \ x, y \in E.$$

Then the asynchronous (with flexible communication) iterates  $x^k$  from (6) converge to  $x^*$ .

Another important special case arises for isotone mappings, i.e., mappings H where  $x \le y$  implies  $Hx \le Hy$ . The following result goes back to Miellou [43]; see also [31] for the slightly more general version given here, as well as for a related result for isotonically decomposable mappings.

**Theorem 3.6.** Assume that E is equipped with a partial ordering based on partial orderings for each component, and that the partial ordering is compatible with the topology on E so that we have

 $x^0 \leqslant x^1 \leqslant x^2 \leqslant \cdots \leqslant x^k \leqslant \cdots \leqslant y^0 \Rightarrow \lim_{k \to \infty} x^k = x^* \text{ exists and } x^* \leqslant y^0.$ 

Assume further that H is continuous and isotone and that there exist  $x^0 \leq y^0$  such that  $x^0 \leq Hx^0 \leq Hy^0 \leq y^0$ . Then the asynchronous iterates  $x^k$  from (4) converge to  $x^*$  with  $x^* = Hx^*$ .

For a proof, let  $z^k = H(z^{k-1})$  with  $z^0 = x^0$ , let  $x^* = \lim_{k \to \infty} z^k \leq y^0$  and take  $E^k = \{x: z^k \leq x \leq x^*\}$  in Theorem 3.1.

## 4. Applications of the theory

In the remainder of the paper we show how the convergence theory for the general models (4)-(6) can be applied to a wide range of scientific problems.

### 4.1. Nonsingular linear systems

Let us start by considering a linear system of the form

$$Ax = b, (10)$$

where  $A \in \mathbb{R}^{n \times n}$  is nonsingular. Let A = M - N be a splitting of A, i.e., M is nonsingular. Let us define the *iteration operator* 

$$H: \mathbb{R}^n \to \mathbb{R}^n, \quad x \to M^{-1}(Nx+b) \tag{11}$$

and analyze the convergence of its associated asynchronous iteration (4) in the case that  $E = \mathbb{R}^n = E_1 \times \cdots \times E_n$  with  $E_i = \mathbb{R}$ , i.e. we allow each component to be treated individually. One example of such splitting is the *Jacobi* operator, when *M* is the diagonal part of *A*. Let |H| denote the matrix of absolute values of entries of *H*.

**Theorem 4.1.** (a) *H* is a *P*-contraction if and only if  $\rho(|H|) < 1$ . (b) If  $\rho(|H|) < 1$ , then the asynchronous iteration (4) (with *H*) converges to  $x^*$ , the solution of Ax = b. (c) If  $\rho(|H|) \ge 1$ , then there exists an asynchronous iteration i.e., a set of delays and strategies satisfying (3), and an initial guess  $x^0$  such that the iterates  $x^k$  produced by (4) do not converge to  $x^* = A^{-1}b$ .

**Proof.** (a) is a simple calculation, (b) follows from Corollary 3.4. Part (c) can be found in [21], where a version of a proof of (b) was given for the first time. Bertsekas and Tsitsiklis [13], Strikwerda [54], and Su et al. [56] suggested different constructions of the non-convergent sequences.  $\Box$ 

We remark that the class of matrices with  $\rho(|H|) < 1$  is just the class of *H*-matrices (see, e.g., [11]). *H*-matrices include *M*-matrices and strictly diagonally dominant or irreducibly diagonally dominant matrices [62].

If we think of grouping components together into (disjoint) blocks  $B_i \subseteq \{1, ..., n\}$ , i = 1, ..., m, we can write (10) in block notation as

$$\sum_{j=1}^{m} A_{ij} x_j = b_i, \quad i = 1, \dots, m,$$
(12)

where  $x_j \in \mathbb{R}^{n_j}, n_j$  is the cardinality of  $B_j, A_{ij} \in \mathbb{R}^{n_i \times n_j}, \sum_{i=1}^m n_i = n$ . The corresponding *block Jacobi* operator *H* is given by (11), where now  $M = \text{diag}(A_{11}, \ldots, A_{mm})$  is the block diagonal of *A* which is assumed to be nonsingular, and A = M - N. In view of Theorem 3.3 we are now interested in cases where *H* is a contraction with respect to a weighted max-norm (7) where  $\|\cdot\|_i$  is a norm on block *i*. Interestingly, this is again so for *H*-matrices.

**Lemma 4.2.** Let A be an H-matrix and let  $x^* = A^{-1}b$ . Then there exist norms  $\|\cdot\|_i$  on each block  $i, i \in \{1, ..., m\}$  such that with the (unweighted) max-norm  $\|x\| = \max_{i=1}^n \|x_i\|_i$ , the Block Jacobi operator H satisfies

$$||Hx - x^*|| \leq \gamma \cdot ||x - x^*|| \quad with \ \gamma \in [0, 1).$$

**Proof.** One proceeds by showing  $\rho(|H|) < 1$ , which is true because the block Jacobi-splitting is an *H*-splitting [36]. This implies the existence of  $v \in \mathbb{R}^n$ , v > 0 with  $|H|v \leq \gamma \cdot v$ ,  $\gamma \in [0, 1)$ . One then defines  $\|\cdot\|_i$  to be the weighted max-norm on block *i* with weights from the respective components of *v*.  $\Box$ 

Alternatively, the following result can be helpful.

**Lemma 4.3.** Let  $\|\cdot\|_i$  be a norm on  $\mathbb{R}^{n_i}$ , i = 1, ..., n. For each block  $A_{ij}$  let  $\|\cdot\|_{ij}$  denote the corresponding matrix norm

 $||A_{ij}||_{ij} = \max_{||x_i||_j=1} ||A_{ij}x_j||_i.$ 

Let  $P = (m_{ij}) \in \mathbb{R}^{n \times n}$  with

$$m_{ij} = \begin{cases} 0 & \text{if } i = j, \\ \|A_{ij}\|_{ij} \cdot \|A_{ii}^{-1}\|_{ii} & \text{if } i \neq j. \end{cases}$$

Then, if  $\rho(P) < 1$  we have that H is a P-contraction with respect to  $x^*$ .

The proof is a straightforward computation.

In view of the above lemma one may thus generalize the concept of an H-matrix to block H-matrix being one for which

$$\begin{pmatrix} 1/\|A_{11}^{-1}\|_{11} & -\|A_{12}\|_{12} & \dots & -\|A_{1m}\|_{1m} \\ -\|A_{21}\|_{21} & 1/|A_{22}^{-1}\|_{22} & & -\|A_{2m}\|_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ -\|A_{m1}\|_{m1} & \dots & \dots & 1/\|A_{mm}^{-1}\|_{mm} \end{pmatrix}$$

is an *M*-matrix (see, e.g., [8,49]). This then implies that the block Jacobi operator is indeed a *P*-contraction, so that asynchronous iterations converge. Note also that in Lemma 4.3 we have freedom in the choice of the norms for each block.

Let us mention that the above results remain unchanged if we replace  $||A_{ii}^{-1}||_{ii}$  in the definition of P by  $1/m_{ii}$  if we assume each block  $A_{ii}$  to be *strongly accretive* with constant  $m_{ii}$  (see, e.g., [39]), i.e., we assume that for all  $x_i \in (\mathbb{R}^{n_i}, || \cdot ||_i)$  there exists a dual  $l_i(x_i)$  of  $x_i$  such that

$$\langle A_{ii}x_i, l_i(x_i)\rangle \geq m_{ii} ||x_i||_i^2.$$

Here,  $\langle \cdot, \cdot \rangle$  denotes the bilinear form between  $(\mathbb{R}^{n_i}, \|\cdot\|_i)$  as a Banach space and its dual, and  $l_i(x_i)$  is an element of  $(\mathbb{R}^{n_i}, \|\cdot\|_i)^*$  with

$$||l_i(x_i)||_i^* = ||x_i||_i$$
 and  $\langle l_i(x_i), x_i \rangle = ||x_i||_i^2$ .

In this asynchronous block Jacobi setting, each processor needs to solve a linear system with the coefficient matrix  $A_{ii}$  in (4) (see (11)). The solution of each of these systems by an iterative method based on a splitting  $A_{ii} = F_i - G_i$  in  $\mathbb{R}^{n_i} \times \mathbb{R}^{n_i}$  gives rise to a non-stationary process (5) with

$$H_i^k(x) = (F_i^{-1}G_i)^{\ell(i,k)} x_i + \sum_{j=0}^{\ell(i,k)-1} (F_i^{-1}G_i)^j (Nx+b),$$

where  $F = \text{diag}(F_1, \dots, F_m)$  and  $G = \text{diag}(G_1, \dots, G_m)$  are block diagonal, M = F - G, and  $\ell(i, k)$  is the number of inner iterations. In the context of the Computational Model 2.3, we have a process of the form (6) with  $K(x, y) = F^{-1}(Gy + Nx + b)$ . Under suitable hypotheses on the splittings A = M - N, and M = F - G (related to weak regular splittings and H-splittings), these methods can be shown to converge using Theorems 3.3 and 3.5, respectively (see [37,38] and also [66,67]).

In the case of overlapping variables, i.e., when the blocks  $B_i$  defining the partitions for (12) are not disjoint, one can still define a block Jacobi iteration with overlap by solving (or approximating) in different processors the linear systems

$$A_{ii}x_i = b_i - \sum_{j=1, j \neq i}^m A_{ij}x_j, \quad i = 1, ..., m_i$$

cf. the Computational Model 2.1. A consistent approximation to the solution of (10) can then be obtained by convex combinations of the elements in each component  $x_i$  belonging to nonempty intersections of the blocks  $B_i$ . The coefficients of these convex combinations, which can simply be ones and zeros, may change from one iteration to the next. A full mathematical description of this case will not be undertaken here, but we point out that for its analysis operators  $H^k : E^m \to E^m$  are defined representing each asynchronous step. Convergence of this asynchronous additive algebraic Schwarz iteration is then obtained using Theorem 3.3 (see [7,35]).

We conclude this subsection with some comments on the case where A in (10) is singular. Any splitting A = M - N (M nonsingular) of a singular matrix A produces an iteration matrix  $H = M^{-1}N$ with 1 as an eigenvalue. Assume for simplicity that  $H \ge 0$  so that |H| = H and  $\rho(H) = 1$  with 1 being an eigenvalue of H. The fixed points of H are the eigenvectors of H corresponding to the eigenvalue 1, and thus form a subspace of  $\mathbb{R}^n$ . Theorem 3.1 and its generalizations cannot be directly applied to this situation, since the nested set condition would normally be violated. In fact, part of the problem is that we no longer have a unique fixed point. In other words, the singular case lies outside the general theory of Section 3 and more restrictive hypotheses on the asynchronous iteration are needed in order to ensure convergence. For example, Lubachevsky and Mitra [41] consider the situation where (basically) H is irreducible, the starting vector is nonnegative and the asynchronous iteration is restricted in such a manner that for some fixed index i we always have  $s_i(k) = k - 1$  whenever  $i \in I^k$ . In this case, one can then actually again construct a nested set of boxes  $E^l$ ,  $l = 0, 1, \ldots$  which converge to some singleton consisting of a fixed point of H, and for all l the asynchronous iterates  $x^k$  satisfy  $x^k \in E^l$  for  $l \ge l(k)$ .

Another approach was taken in [5,50] using the concept of paracontractions. Again, additional restrictions have to be imposed on the asynchronous iteration in order to guarantee convergence. For example, Bahi [5] requires to do a "true" step of (synchronous) successive approximation every once in a while.

### 4.2. Nonlinear equations

Assume that we are given a nonlinear system of equations

$$F(x) = 0 \quad \text{where } F: D_F \subseteq \mathbb{R}^n \to \mathbb{R}^n.$$
(13)

Assume that this equation has exactly one solution  $x^*$  and let  $H : D_H \subseteq \mathbb{R}^n \to \mathbb{R}^n$  be an iteration function for this problem, i.e.,  $x^*$  is the unique fixed point of H. Not too surprisingly, the following local version of Corollary 3.4 can be shown to hold [27].

**Theorem 4.4.** Assume that  $x^*$  lies in the interior of  $D_H$  and that H is Fréchet differentiable at  $x^*$ . If  $\rho(|H'(x^*)|) < 1$ , then there exists a neighborhood  $\mathcal{N}$  of  $x^*$  such that the asynchronous iterates (4) converge to  $x^*$ , provided  $x^0 \in \mathcal{N}$ .

The standard iteration operator for (13) is the Newton operator  $H_N(x) = x - F'(x)^{-1}F(x)$ . Here,  $H'(x^*) = 0$  so that Theorem 4.4 can be applied. However, for practical reasons it is mandatory in asynchronous computations that the components of H can be evaluated individually at much lower cost than all components together. Due to the presence of the term  $F'(x)^{-1}$  in  $H_N$  this is usually not the case. A favorable situation arises, for example, in the Durand–Kerner method [22,23], for the simultaneous computation of all zeros of a polynomial, which is Newton's method on the equations expressing the coefficients of the polynomial via the elementary symmetric functions on its roots, but where one has a simple, explicit formula for each component of  $H_N$ .

If D(x) denotes the diagonal part of F'(x) = D(x) - B(x), the Newton-Jacobi operator is given as  $H_{NJ}(x) = x - D^{-1}(x)F(x)$  [47]. We can regard  $H_{NJ}$  as a two-stage approximation to  $H_N$  with one inner step. Here it is trivial that components of  $H_{NJ}$  can be evaluated individually. We have  $H'_{NJ}(x^*) = D(x^*)^{-1}B(x^*)$ . So, as in the remark following Theorem 4.1, we see that we get local convergence of the asynchronous iterates if  $H'(x^*)$  is an *H*-matrix. It is important to notice that functions *F* satisfying the above conditions arise in several applications, including discretizations of elliptic partial differential equations.

In a general way, we can define the components of the nonlinear Jacobi operator  $H_J$  for (13) through

$$y_i = (H_J)_i(x) \iff F_i(x_1, \dots, x_{i-1}, y_i, x_{i+1}, \dots, x_n) = 0.$$

The generalization to a block nonlinear Jacobi operator should be obvious (cf. (16)). Another asynchronous approach to Newton's method includes the work by Bojańczyk [16], where the Newton operator can be viewed as  $K(x, y) = x - F'(y)^{-1}F(x)$  (cf. Theorem 3.5). Yet another extension is to consider quasi-Newton methods [64].

Interestingly, there are several important situations where global convergence of asynchronous iterates for  $H_J$  can be proved. As a generalization of Lemma 4.3 it was shown in [39] that Theorem 3.3 holds for the (block) Jacobi operator  $H_J$  for certain mildly non-linear functions arising in discretizations of elliptic boundary value problems, the obstacle problem or the Hamilton–Jacobi–Bellman problem.

If the function F is an M-function (see [47]), one can also prove global convergence of the asynchronous iterates for  $H_J$ , now using Theorem 3.6 [68]. Generalizations for further classes of functions, including a nonlinear analog of H-matrices have been developed in [30].

### 4.3. Waveform relaxation

Waveform relaxation methods are parallel iterative methods for initial value problems based on a splitting principle. They were developed at the beginning of the 1980s for the simulation of electronic circuits (see [63]).

Consider the initial value problem

$$\dot{x}(t) = F(t, x(t)), \ t \in [0, T],$$
  
 $x(0) = x_0,$ 
(14)

where  $F : [0, T] \times \mathbb{R}^n \to \mathbb{R}^n$ ,  $x(t) \in \mathbb{R}^n$ . Instead of solving (14) directly, waveform relaxation methods split the function F into a function  $G : [0, T] \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$  with G(t, x, x) = F(t, x) for all t and x. Starting with a function  $x^0 : [0, T] \to \mathbb{R}^n$  satisfying  $x^0(0) = x_0$  (for example  $x^0 \equiv x_0$ ) one then successively solves the systems

$$\dot{x}^{k}(t) = G(t, x^{k}(t), x^{k-1}(t)), \ t \in [0, T],$$
  
$$x^{k}(0) = x_{0}$$
(15)

for k = 1, 2, .... Here, the function  $x^{k-1}$  is known and  $x^k$  is to be determined.

Note that the familiar Picard iteration

$$\dot{x}^{k}(t) = F(t, x^{k-1}(t)), \ t \in [0, T],$$
  
 $x^{k}(0) = x_{0}$ 

is a special waveform relaxation method where G(t, x, y) = F(t, y). Since the Picard iteration usually converges very slowly, one is interested in better splittings G which yield faster convergence. One

possibility is to take a block Jacobi type splitting which, given a block decomposition  $x = (x_1, ..., x_m)$ (and similarly  $F(t, x) = ((F(t, x))_1, ..., (F(t, x))_m))$  defines block *i* of *G* as

$$(G(t,x,y))_i := (F(t,(x_1,\ldots,x_{i-1},y_i,x_{i+1},\ldots,x_m)))_i.$$
(16)

From now on, let us assume that G satisfies a one-sided Lipschitz condition with respect to the first argument, and a strong Lipschitz-condition with respect to the second argument, i.e., one has

$$\langle G(t,x,y) - G(t,\bar{x},y), x - \bar{x} \rangle \leq l \cdot ||x - \bar{x}||^2, \qquad ||G(t,x,y) - G(t,x,\bar{y})|| \leq L \cdot ||y - \bar{y}||$$
(17)

for all  $t \in [0, T]$ , ];  $x, \bar{x}, y, \bar{y} \in \mathbb{R}^n$ . The Lipschitz condition with respect to the first argument implies that (15) has a unique solution  $x^k$ . Moreover, since F(t,x) = G(t,x,x), the function F also satisfies a (one-sided) Lipschitz condition which shows that (14) has a unique solution  $x^*$ .

Iteration (15) defines an operator *H* where y = H(x) if

$$\dot{y}(t) = G(t, y(t), x(t)), \ t \in [0, T],$$
  
 $y(0) = x_0.$ 
(18)

Here, *H* acts on a space of functions. We take this space to be  $\mathscr{C}([0,T], \mathbb{R}^n)$ , the Banach space of all continuous functions from [0,T] to  $\mathbb{R}^n$  with the norm

$$|||x|||_{\alpha} = \max_{t \in [0,T]} e^{-\alpha t} ||x(t)||_{\infty}, \quad \alpha > 0.$$
(19)

It is crucial for us to notice that  $\| \cdot \| _{\alpha}$  is in fact a maximum norm over the block components of x since

$$|||x|||_{\alpha} = \max_{i=1}^{m} ]|x_i||_{\alpha},$$
(20)

where for continuous  $f:[0,T] \to \mathbb{R}^{n_i}$   $(n_i \text{ is the dimension of block } i)$  the norm  $]| \cdot |[_{\alpha} \text{ is given as}]$ 

$$]|f|[_{\alpha} = \max_{t \in [0,T]} e^{-\alpha t} ||f(t)||_{\infty}.$$
(21)

The following theorem proved very recently by Martin [42] shows that for  $\alpha$  sufficiently large the operator *H* from (18) is contracting with respect to the max-norm  $|||x|||_{\alpha}$ . Therefore, Theorem 3.3 shows that asynchronous iterations for *H* converge.

**Theorem 4.5.** Assume that G satisfies (17) and let  $x^* \in \mathscr{C}([0, T], \mathbb{R}^n)$  be the solution of (14). There exists  $\alpha$  sufficiently large such that for all  $x \in \mathscr{C}([0, T], \mathbb{R}^n)$  we have

$$|||H(x) - x^*|||_{\alpha} \leq \frac{1}{2} \cdot |||x - x^*|||_{\alpha}$$

**Proof.** Denote y = H(x) and  $u(t) = y(t) - x^*(t)$ .

Then  $\dot{u}(t) = G(t, y(t), x(t)) - G(t, x^{*}(t), x^{*}(t))$ , so that from (17) we get

$$\dot{u}(t), u(t) \leq l \cdot ||u(t)||^2 + L \cdot ||u(t)|| \cdot ||x(t) - x^*(t)||, \quad t \in [0, T]$$

Since  $\langle \dot{u}(t), u(t) \rangle = (d/dt) ||u(t)||^2$  whenever  $u(t) \neq 0$ , a standard argument from the theory of differential inequalities (see, e.g., [19]) yields

$$||u(t)|| \leq L \cdot e^{|l|t} \int_0^t ||x(s) - x^*(s)|| e^{-|l|s} ds.$$

Turning from  $\|\cdot\|$  to  $\|\cdot\|_{\infty}$  we get

$$||u(t)||_{\infty} \leq cL \cdot e^{|l|t} \int_{0}^{t} ||x(s) - x^{*}(s)||_{\infty} e^{-|l|s} ds$$

with some constant c > 0. From the latter inequality we conclude

$$\begin{aligned} \|u(t)\|_{\infty} &\leqslant cL \cdot \mathrm{e}^{|l|t} \int_{0}^{t} \|x(s) - x^{*}(s)\|_{\infty} \mathrm{e}^{-\alpha s} \mathrm{e}^{\alpha s} \mathrm{e}^{-|l|s} \,\mathrm{d}s \\ &\leqslant cL \cdot \mathrm{e}^{|l|t} \cdot \||x - x^{*}\||_{\alpha} \int_{0}^{t} \mathrm{e}^{(\alpha - |l|)s} \,\mathrm{d}s. \end{aligned}$$

For  $\alpha > |l|$  the last integral is less than  $e^{(\alpha - |l|)t}/(\alpha - |l|)$ , so that we get

$$|u(t)||_{\infty}e^{-\alpha t} \leq \frac{cL}{\alpha - |l|} ||x - x^*|||_{\alpha}, \quad t \in [0, T]$$

and thus

$$|||u|||_{\alpha} \leqslant \frac{cL}{\alpha - |l|} \cdot |||x - x^*|||_{\alpha}.$$

So taking  $\alpha > 2cL + |l|$  proves the theorem.  $\Box$ 

For an infinite time interval [45] gives a convergence result for asynchronous iterations under much more restrictive assumptions. For differential-algebraic systems and asynchronous iterations, see [4,6].

In [42] several numerical results on asynchronous waveform relaxation methods have been reported. These computations were done on a cluster of 8 SUN Ultra Sparc 10 workstations, connected via fast Ethernet. The example considered arises from a model describing the penetration of radioactively marked antibodies into cancerous tissue (MedicalAkzo from [40]). The total system size was 400, and the splitting G was obtained by a block Jacobi decomposition of F assigning a block of 50 to each processor. The asynchronous variant then needed only 66% (120 s) of the time required for the synchronous variant (180 s).

### References

- D. Amitai, A. Averbuch, M. Israeli, S. Itzikowitz, On parallel asynchronous high-order solutions of parabolic PDEs, Numer. Algorithms 12 (1996) 159–192.
- [2] D. Amitai, A. Averbuch, M. Israeli, S. Itzikowitz, Implicit-explicit parallel asynchronous solver for PDEs, SIAM J. Sci. Comput. 19 (1998) 1366–1404.
- [3] D. Amitai, A. Averbuch, M. Israeli, S. Itzikowitz, E. Turkel, A survey of asynchronous finite-difference methods for parabolic PDEs on multiprocessors, Appl. Numer. Math. 12 (1993) 27–45.
- [4] J. Bahi, Asynchronous Runge–Kutta methods for differential-algebraic systems, Adv. Parallel Comput. 11 (1996) 205–212.
- [5] J. Bahi, Algorithmes parallèles asynchrones pour des systèmes singuliers, C.R. l'Acad. Sci. Sér. 326 (1998) 1421– 1425.
- [6] J. Bahi, E. Griepentrog, J.-C. Miellou, Parallel treatment of a class of differential-algebraic systems, SIAM J. Numer. Anal. 23 (1996) 1969–1980.

- [7] J. Bahi, J.-C. Miellou, K. Rhofir, Asynchronous multisplitting methods for nonlinear fixed point problems, Numer. Algorithms 15 (1997) 315–345.
- [8] Z.-Z. Bai, V. Migallón, J. Penadés, D.B. Szyld, Block and asynchronous two-stage methods for mildly nonlinear systems, Numer. Math. 82 (1999) 1–20.
- [9] B. Barán, E. Kaszkurewicz, A. Bhaya, Parallel asynchronous team algorithms: Convergence and performance analysis, IEEE Trans. Parallel Distributed Systems 7 (1996) 677–688.
- [10] G.M. Baudet, Asynchronous iterative methods for multiprocessors, J. Assoc. Comput. Mach. 25 (1978) 226-244.
- [11] A. Berman, R.J. Plemmons, Nonnegative Matrices in the Mathematical Sciences, Academic Press, New York, Third edition, 1979 (reprinted by SIAM, Philadelphia, 1994).
- [12] D.P. Bertsekas, Distributed asynchronous computation of fixed points, Math. Programming 27 (1983) 107–120.
- [13] D.P. Bertsekas, J.N. Tsitsiklis, Parallel and Distributed Computation, Prentice-Hall, Englewood Cliffs, NJ, 1989.
- [14] D.P. Bertsekas, J.N. Tsitsiklis, Some aspects of parallel and distributed iterative algorithms a survey, Automatica 27 (1991) 3–21.
- [15] K. Blathras, D.B. Szyld, Y. Shi, Timing models and local stopping criteria for asynchronous iterative algorithms. J. Parallel Distributed Comput. 58 (1999) 446–465.
- [16] A. Bojańczyk, Optimal asynchronous Newton method for the solution of nonlinear equations, J. Assoc. Comput. Mach. 31 (1984) 792–803.
- [17] R. Bru, V. Migallón, J. Penadés, D.B. Szyld, Parallel, synchronous and asynchronous two-stage multisplitting methods, Electron. Trans. Numer. Anal. 3 (1995) 24–38.
- [18] P. Buchholz, M. Fischer, P. Kemper, Distributed steady state analysis using Kronecker algebra, in: B. Plateau, W.J. Stewart (Eds.), Numerical Solution of Markov Chains (NSMC '99), Prensas Universitarias de Zaragoza, 1999, pp. 76–95.
- [19] K. Burrage, Sequential and Parallel Methods for Ordinary Differential Equations, Oxford University Press, Oxford, 1995.
- [20] M.J. Castel, V. Migallón, J. Penadés, Convergence of non-stationary multisplitting methods for Hermitian positive definite matrices, Math. Comput. 67 (1998) 209–220.
- [21] D. Chazan, W. Miranker, Chaotic relaxation, Linear Algebra Appl. 2 (1969) 199-222.
- [22] M. Cosnard, P. Fraigniaud, Asynchronous Durand-Kerner and Aberth polynomial root finding methods on a distributed memory multicomputer, in: D. Evans, G. Joubert, F. Peters (Eds.), Parallel Computing 89, North-Holland, Amsterdam, 1990.
- [23] E. Durand, Solutions numériques des équations algébriques, Vol. I, Masson, Paris, 1960.
- [24] D. El Baz, A method of terminating asynchronous iterative algorithms on message passing systems, Parallel Algorithms Appl. 9 (1996) 153–158.
- [25] D. El Baz, D. Gazen, J.-C. Miellou, P. Spiteri, Mise en œuvre de méthodes itératives asynchrones avec communication flexible, application à la résolution d'une classe de problèmes d'optimisation, Calculateurs Parallèles 8 (1996) 393– 410.
- [26] D. El Baz, P. Spiteri, J.-C. Miellou, Distributed asynchronous iterative methods with order intervals for a class of nonlinear optimization problems, J. Parallel Distributed Comput. 38 (1996) 1–15.
- [27] M.N. El Tarazi, Some convergence results for asynchronous algorithms, Numer. Math. 39 (1982) 325–340.
- [28] L. Elsner, I. Koltracht, M. Neumann, On the convergence of asynchronous paracontractions with application to tomographic reconstruction from incomplete data, Linear Algebra Appl. 130 (1990) 65–82.
- [29] L. Elsner, I. Koltracht, M. Neumann, Convergence of sequential and asynchronous nonlinear paracontractions, Numer. Math. 62 (1992) 305–319.
- [30] A. Frommer, Generalized nonlinear diagonal dominance and applications to asynchronous iterative methods, J. Comput. Appl. Math. 38 (1991) 105–124.
- [31] A. Frommer, On asynchronous iterations in partially ordered spaces, Numer. Functional Anal. Optim. 12 (1991) 315–325.
- [32] A. Frommer, Asynchronous iterations for enclosing solutions of fixed point problems, in: L. Atanassova, J. Herzberger (Eds.), Computer Arithmetic and Enclosure Methods, Elsevier, Amsterdam, 1992, pp. 243–252.
- [33] A. Frommer, Parallele asynchrone Iterationen. in: J. Herzberger (Ed.), Wissenschaftliches Rechnen, Akademie, Berlin, 1995, pp. 187–231 (Chapter 4).

- [34] A. Frommer, H. Schwandt, Asynchronous parallel methods for enclosing solutions of nonlinear equations, J. Comput. Appl. Math. 60 (1995) 47–62.
- [35] A. Frommer, H. Schwandt, D.B. Szyld, Asynchronous weighted additive Schwarz methods, Electron. Trans. Numer. Anal. 5 (1997) 48–61.
- [36] A. Frommer, D.B. Szyld, H-splittings and two-stage iterative methods, Numer. Math. 63 (1992) 345–356.
- [37] A. Frommer, D.B. Szyld, Asynchronous two-stage iterative methods, Numer. Math. 69 (1994) 141-153.
- [38] A. Frommer, D.B. Szyld, Asynchronous iterations with flexible communication for linear systems, Calculateurs Parallèles 10 (1998) 421–429.
- [39] L. Giraud, P. Spitéri, Résolution parallèle de problèmes aux limites non lineaires, Math. Modelling Numer. Anal. 25 (1991) 579–606.
- [40] W. Lioen, J. de Swart, W. van der Veen, Test set for IVP solvers. CWI Amsterdam, Online at http://www.cwi.nl/cwi/projects/IVPtestset.
- [41] B. Lubachevsky, D. Mitra, A chaotic asynchronous algorithm for computing the fixed point of a nonnegative matrix of unit spectral radius, J. Assoc. Comput. Mach. 33 (1986) 130–150.
- [42] S. Martin, Parallele asynchrone Waveform-Relaxation f
  ür Anfangswertaufgaben, Master's thesis, Department of Mathematics, University of Wuppertal, 1999.
- [43] J.C. Miellou, Itérations chaotiques à retards; études de la convergence dans le cas d'espaces partiellement ordonnés, C.R. l'Acad. Sci. Paris, Sér. A 280 (1975) 233–236.
- [44] J.C. Miellou, D. El Baz, P. Spitéri, A new class of asynchronous iterative algorithms with order intervals, Math. Comput. 67 (1998) 237–255.
- [45] D. Mitra, Asynchronous relaxations for the numerical solution of differential equations by parallel processors, SIAM J. Sci. Statist. Comput. 8 (1987) s43–s58.
- [46] J.J. Moré, Nonlinear generalizations of matrix diagonal dominance with application to Gauss–Seidel iterations, SIAM J. Numer. Anal. 9 (1972) 357–378.
- [47] J.M. Ortega, W.C. Rheinboldt, Iterative Solution of Nonlinear Equations in Several Variables, Academic Press, New York and London, 1970.
- [48] V. Pereyra, Asynchronous distributed solution of large scale nonlinear inversion problems, Appl. Numer. Math. 30 (1999) 31–40.
- [49] B. Polman, Incomplete blockwise factorization of (block) H-matrices, Linear Algebra Appl. 90 (1987) 119–132.
- [50] M. Pott, On the convergence of asynchronous iteration methods for nonlinear paracontractions and consistent linear systems, Linear Algebra Appl. 283 (1998) 1–33.
- [51] S.A. Savari, D.P. Bertserkas, Finite termination of asynchronous iterative algorithms, Parallel Comput. 22 (1996) 39–56.
- [52] W. Shangmeng, L. Xiaomei, A class of stable difference schemes for linear elliptic PDE's and their asynchronous parallel computation, Wuhan Univ. J. Nat. Sci. 1 (1996) 553–556.
- [53] J. Šilc, B. Robič, T. Ungerer, Asynchrony in parallel computing: From dataflow to multithreading, Parallel Distributed Comput. Practices 1 (1998) 3–30.
- [54] J.C. Strikwerda, A convergence theorem for chaotic asynchronous relaxation, Linear Algebra Appl. 253 (1997) 15–24.
- [55] Y. Su, A. Bhaya, Convergence of pseudocontractions and applications to two-stage and asynchronous multisplittings, Technical Report SB97/NACAD/05, Laboratory for High Performance Computing, Graduate School of Engineering, Federal University of Rio de Janeiro, 1997.
- [56] Y. Su, A. Bhaya, E. Kaszkurewicz, V.S. Kozyakin, Further results on convergence of asynchronous linear iterations, Linear Algebra Appl. 281 (1998) 11–24.
- [57] D.B. Szyld, Different models of parallel asynchronous iterations with overlapping blocks, Comput. Appl. Math. 17 (1998) 101–115.
- [58] X.-C. Tai, P. Tseng, An asynchronous space decomposition method, in: C.-H. Lai, P.E. Bjørstad, M. Cross, O.B. Widlund (Eds.), Eleventh International Conference on Domain Decomposition Methods (London), DDM.org Press, 1999, pp. 348–358.
- [59] P. Tseng, On the rate of convergence of a partially asynchronous gradient projection algorithm, SIAM J. Optim. 1 (1991) 603–619.

- [60] P. Tseng, D.P. Bertsekas, J.N. Tsitsiklis, Partially asynchronous, parallel algorithm for network flow and other problems, SIAM J. Control Optim. 28 (1990) 678–710.
- [61] A. Üresin, M. Dubois, Sufficient conditions for the convergence of asynchronous iterations, Parallel Comput. 10 (1989) 83–92.
- [62] R.S. Varga, On recurring theorems on diagonal dominance, Linear Algebra Appl. 13 (1976) 1-9.
- [63] J. White, A. Sangiovanni-Vincentelli, Relaxation Techniques for the Simulation on VLSI Circuits, Kluwer Academic Publishers, Boston, 1987.
- [64] J.-J. Xu, Convergence of partially asynchronous block quasi-Newton methods for nonlinear systems of equations, J. Comput. Appl. Math. 103 (1999) 307–321.
- [65] J. Aral, V. Migallón, J. Penadés, Non-stationary parallel multisplitting algorithms for almost linear systems, Numer. Linear Algebra Appl. 6 (1999) 79–92.
- [66] R. Bru, L. Elsner, M. Neumann, Models of parallel chaotic iteration methods, Linear Algebra Appl. 103 (1988) 175–192.
- [67] R. Bru, L. Elsner, M. Neumann, Convergence of infinite products of matrices and inner-outer iteration schemes, Electron. Trans. Numer. Anal. 2 (1994) 183–193.
- [68] D. El Baz, M-functions and parallel asynchronous algorithms, SIAM J. Numer. Anal. 27 (1990) 136-140.


Journal of Computational and Applied Mathematics 123 (2000) 217-240

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.nl/locate/cam

# Iterative methods for large continuation problems

D. Calvetti<sup>a, \*, 1</sup>, L. Reichel<sup>b, 2</sup>

<sup>a</sup>Department of Mathematics, Case Western Reserve University, Cleveland, OH 44106, USA <sup>b</sup>Department of Mathematics and Computer Science, Kent State University, Kent, OH 44242, USA

Received 1 September 1999; received in revised form 11 September 1999

#### Abstract

The computation of solution paths for continuation problems requires the solution of a sequence of nonlinear systems of equations. Each nonlinear system can be solved by computing the solution of a succession of linear systems of equations determined by Jacobian matrices associated with the nonlinear system of equations. Points on the solution path where the Jacobian matrix is singular are referred to as singular points and require special handling. They may be turning points or bifurcation points. In order to detect singular points, it is essential to monitor the eigenvalues of smallest magnitude of the Jacobian matrices generated as the solution path is traversed. We describe iterative methods for the computation of solution paths for continuation problems so large that factorization of the Jacobian matrices is infeasible or impractical. The iterative methods simultaneously solve linear systems of equations determined by the Jacobian matrices and compute a few eigenvalue–eigenvector pairs associated with the eigenvalues of smallest magnitude of each Jacobian. A bordering algorithm with a pseudo-arclength parametrization is applied in the vicinity of turning points to overcome the singularity of the Jacobian. A bifurcation perturbation strategy is used to compute solution paths at bifurcation points. Our iterative methods are based on the block-Lanczos algorithm and are applicable to problems with large symmetric Jacobian matrices. (© 2000 Elsevier Science B.V. All rights reserved.

#### MSC: 65F15

Keywords: Path following; Turning point; Bifurcation point; Eigenvalue computation; Bordering algorithm; Nonlinear system

# 1. Introduction

Many problems in science and engineering require the computation of a family of solutions  $u(\lambda) \in \mathbb{R}^n$ ,  $a \leq \lambda \leq b$ , of a nonlinear system of equations of the form

<sup>\*</sup> Corresponding author.

E-mail addresses: dxc57@po.cwru.edu (D. Calvetti), reichel@mcs.kent.edu (L. Reichel).

<sup>&</sup>lt;sup>1</sup>Research supported in part by NSF grant DMS-9806702.

<sup>&</sup>lt;sup>2</sup> Research supported in part by NSF grants DMS-9806413 and ASC-9720221.

$$G(u,\lambda) = 0, \quad u = u(\lambda), \tag{1.1}$$

where  $G : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$  is a continuously differentiable function of  $u \in \mathbb{R}^n$  and  $\lambda \in \mathbb{R}$ . The parameter  $\lambda$  is often a quantity of physical significance, such as temperature in liquid crystal modeling [4] or the Reynolds number in hydrodynamical flow [15], and is commonly referred to as the "natural parameter". We are interested in determining solution paths

$$\Gamma := \{ (u, \lambda) : G(u, \lambda) = 0, u = u(\lambda), a \leq \lambda \leq b \}$$

associated with (1.1). Here a and b are given bounds for  $\lambda$ .

The solutions  $u(\lambda)$ ,  $a \le \lambda \le b$ , of (1.1) are commonly computed by a continuation method. In these methods an initial value problem for u is derived by differentiating Eq. (1.1) with respect to  $\lambda$ . Thus, let  $u = u(\lambda)$  satisfy (1.1). Then differentiation of (1.1) yields

$$G_u(u(\lambda), \lambda)\dot{u}(\lambda) + G_\lambda(u(\lambda), \lambda) = 0,$$
(1.2)

where  $\dot{u} = du/d\lambda$ . Given u(a) and assuming that the Jacobian matrix  $G_u$  is nonsingular in a neighborhood of the solution path, we can compute  $u(\lambda)$  for  $a < \lambda \le b$  by solving the initial value problem (1.2) for  $u = u(\lambda)$ . Points where the Jacobian matrix  $G_u(u, \lambda)$  is nonsingular are referred to as regular points; points where  $G_u(u, \lambda)$  is singular are referred to as singular points. Singular points on the solution path are either turning points or bifurcation points of the solution path. The determination of the solution path in a neighborhood of a turning point or bifurcation point requires special care. It is therefore important to detect singular points on the solution path.

This paper describes new algorithms for path following. The algorithms are designed to be applicable to problems so large that factorization of the Jacobian matrices into triangular or orthogonal factors is unfeasible or undesirable. Our algorithms only evaluate matrix–vector products with the Jacobian matrices. Therefore, only a few of the nonvanishing entries of each Jacobian matrix generated have to be stored in fast computer memory simultaneously; the entries can be computed as they are required for the evaluation of matrix–vector products and discarded when they are not needed. This approach requires little computer memory and is therefore well suited for large problems, such as the liquid crystal modeling problem discussed in [4].

We assume that the Jacobian matrices are symmetric. Our algorithms are based on an iterative method for the simultaneous solution of linear systems of equations with the Jacobian matrix and computation of a few of the eigenvalues of smallest magnitude and associated eigenvectors of the Jacobian. Since the Jacobian is singular at turning and bifurcation points on the solution path and regular elsewhere on the solution path, knowledge of the eigenvalue closest to the origin makes it easy to identify these points. Moreover, the eigenvectors associated with the smallest eigenvalues are helpful for path following in the vicinity of a turning or bifurcation point.

Our iterative method for solving linear systems, while simultaneously computing a few eigenvalue–eigenvector pairs, is based on the implicitly restricted block-Lanczos (IRBL) method introduced in [4]. This method is a block-variant of the implicitly restarted Lanczos method discussed in [2,6,17].

Bifurcation points are traversed by two or more solution paths. Different methods for continuing paths across bifurcation points have been proposed in the literature. This paper only discusses the "perturbed bifurcation" method, where a small perturbation of Eq. (1.1) is introduced at a bifurcation point. This makes a bifurcation point split into close regular or turning points; see Georg [13] for a discussion and illustrations.

Continuation methods for path following have received considerable attention. A nice survey of the mathematical background is provided by Keller [16]. Only few algorithms are available for large-scale problems; see, e.g., [1,8,9,14,15,18]. Our algorithms differ from those available in the literature in that they are based on the IRBL method and are designed to be applicable for problems with very large symmetric Jacobian matrices.

This paper is organized as follows. Section 2 recalls a few useful results on the solution of continuation problems and discusses the calculations needed for path following. In Section 3, we outline an iterative method, previously introduced in [5], for the computation of a few eigenpairs of a large symmetric matrix, and the simultaneous solution of a linear system of equations with this matrix. Section 4 describes how to apply this iterative method to path following in the presence of turning and bifurcation points. We present path following algorithms based on the Euler–Newton predictor-corrector scheme for use at regular points on the solution path. A pseudo-arclength parametrization is used in the vicinity of turning points. Numerical examples are presented in Section 5 and concluding remarks can be found in Section 6.

#### 2. An overview of the path following problem

The first part of this section reviews the continuation problem and introduces notation to be used in the remainder of the paper. In the second part the computational problems are described.

# 2.1. Theory

In this subsection the Jacobian matrix is allowed to be nonsymmetric. We focus on the interplay between geometry and computations. An excellent introduction to the numerical analysis of continuation problems is provided by Keller [16] and much of our discussion follows his presentation.

The purpose of a continuation method is to determine solutions of problem (1.1) for all  $\lambda$  in a specified interval [a, b]. Let  $\lambda^1 = \lambda^0 + \Delta \lambda$  with  $\lambda^0, \lambda^1 \in [a, b]$  and assume that the solution  $u^0 = u(\lambda^0)$  of (1.1) for  $\lambda = \lambda^0$  is known. The Implicit Function Theorem provides the theoretical basis for computational methods for determining the solution  $u^1 = u(\lambda^1)$  of (1.1) for  $\lambda = \lambda^1$  when  $\Delta \lambda$  is of sufficiently small magnitude.

Throughout this paper  $\|\cdot\|$  denotes the Euclidean vector norm or the induced matrix norm. We note, however, that when G stems from the discretization of a differential or integral equation, it can be advantageous to select a norm that depends on the discretization; see Ferng and Kelley [12] for a discussion.

Introduce the sets

$$B_{\rho}(u^0) := \{ u \in \mathbb{R}^n \colon \|u - u^0\| < \rho \}, \qquad B_{\rho}(\lambda^0) := \{ \lambda \in \mathbb{R} : |\lambda - \lambda^0| < \rho \}$$

for  $\rho > 0$ .

**Theorem 2.1** (Implicit Function Theorem). Let  $G : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$  be a function, such that for some sufficiently small constants  $\rho_1 > 0$  and  $\rho_2 > 0$ ,

(i)  $G(u^0, \lambda^0) = 0$  for some  $u^0 \in \mathbb{R}^n$  and  $\lambda^0 \in \mathbb{R}$ ,

(ii) *G* and *G<sub>u</sub>* are continuous in  $B_{\rho_1}(u^0) \times B_{\rho_2}(\lambda^0)$ , (iii) *G<sub>u</sub>(u<sup>0</sup>, λ<sup>0</sup>)* is nonsingular with a bounded inverse. Then for every  $\lambda \in B_{\rho_2}(\lambda^0)$  there is a unique  $u := u(\lambda) \in B_{\rho_1}(u^0)$ , such that (a) *G(u, λ)* = 0 and  $u(\lambda^0) = u^0$ ,

(b)  $u = u(\lambda)$  is a continuous function of  $\lambda$  on  $B_{\rho_2}(\lambda^0)$ .

**Proof.** The theorem can be formulated for u and  $\lambda$  in more general sets than  $\mathbb{R}^n$  and  $\mathbb{R}$ , respectively. For instance, Keller [16, Section 2.7] presents a proof when u belongs to a Banach space and  $\lambda$  to a parameter space.  $\Box$ 

It is an immediate consequence of the Implicit Function Theorem that the continuation problem for (1.1) has a unique solution in a neighborhood of a regular point  $(u^0, \lambda^0)$  on the solution path.

Given a regular point  $(u^0, \lambda^0)$  for (1.1), the solution  $u^1 = u(\lambda^1)$  of (1.1) for  $\lambda = \lambda^1$  can be computed by a predictor-corrector scheme when  $\Delta \lambda = \lambda^1 - \lambda^0$  is of sufficiently small magnitude. The predictor determines an initial approximation  $u_0(\lambda^1)$  of the solution  $u(\lambda^1)$  of (1.1). It follows from Theorem 2.1 that for some  $\rho_2 > 0$  and every  $\lambda \in B_{\rho_2}(\lambda^0)$ , there is a unique  $u(\lambda) \in \mathbb{R}^n$ , such that

$$G(u(\lambda), \lambda) = 0. \tag{2.1}$$

Differentiating (2.1) with respect to  $\lambda$  yields (1.2). Substituting  $\lambda = \lambda^0$  into  $G_u$  and  $G_{\lambda}$  in (1.2) gives the linear system of equations

$$G_u(u^0,\lambda^0)\dot{u}^0 = -G_\lambda(u^0,\lambda^0) \tag{2.2}$$

for  $\dot{u}^0 = \dot{u}(\lambda^0)$ . Application of Euler's method as a predictor yields the approximation

$$u_0(\lambda) := u^0 + (\lambda - \lambda^0) \dot{u}^0 \tag{2.3}$$

of  $u(\lambda)$ . The error in this approximation is given by

$$u(\lambda) - u_0(\lambda) = \frac{1}{2}\ddot{u}(\lambda^0)(\lambda - \lambda^0)^2 + \mathcal{O}((\lambda - \lambda^0)^3)$$

which reflects that Euler's method is an integration method of order one. Here  $\ddot{u} = d^2 u/d\lambda^2$ .

In general,  $(u_0(\lambda), \lambda)$  does not satisfy (1.1). It is convenient to use Newton's method for (1.1) as a corrector. The iterates  $u_k$ , k = 1, 2, ..., determined by Newton's method are given by

$$u_{k+1} := u_k + \Delta u, \quad k = 0, 1, \dots, \tag{2.4}$$

where  $\Delta u$  solves the linear system of equations

$$G_u(u_k,\lambda)\Delta u = -G(u_k,\lambda). \tag{2.5}$$

Assume that the conditions of Theorem 2.1 are satisfied at  $(u^0, \lambda^0)$ . Then  $G_u$  is continuous and invertible in a neighborhood of  $(u^0, \lambda^0)$ . It can be shown that for a step size  $\Delta \lambda$  of sufficiently small magnitude, the iterates determined by the Euler–Newton predictor–corrector scheme converge to  $u(\lambda)$ . The following definitions are helpful for the analysis of the continuation problem for (1.1) at points where the Jacobian matrix is singular.

**Definition 2.2.** Let  $G : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$  be a continuously differentiable function. A point  $(u(\lambda), \lambda)$  on the solution path is said to be a simple singular point if  $G_u = G_u(u(\lambda), \lambda)$  is singular and

$$\dim \mathcal{N}(G_u) = 1, \tag{2.6}$$

where  $\mathcal{N}(G_u)$  denotes the null space of  $G_u$ .

**Definition 2.3.** A simple singular point  $(u(\lambda), \lambda)$  on a solution path  $\Gamma$  is said to be a turning point if

$$G_{\lambda} \notin \mathscr{R}(G_{u}) \tag{2.7}$$

and a bifurcation point if

$$G_{\lambda} \in \mathscr{R}(G_u), \tag{2.8}$$

where  $\mathscr{R}(G_u)$  denotes the range of  $G_u = G_u(u(\lambda), \lambda)$ , and  $G_{\lambda} = G_{\lambda}(u(\lambda), \lambda)$ .

In this paper we only consider turning points and bifurcation points that are simple singular points. We refer to Decker and Keller [10], Georg [13] and Keller [16] for discussions on more general singular points.

We first consider turning points. It is convenient to introduce the arclength parameter s of  $\Gamma$ . Henceforth, we write u = u(s),  $\lambda = \lambda(s)$  and  $G(s) = G(u(s), \lambda(s))$ , and the derivatives  $\dot{u}$  and  $\dot{\lambda}$  denote differentiation with respect to s. We have

$$\|\dot{u}\|^2 + \dot{\lambda}^2 = 1, \tag{2.9}$$

which shows that the tangent vector  $(\dot{u}(s), \dot{\lambda}(s))$  of  $\Gamma$  is of unit length.

Assume that  $(u(s^0), \lambda(s^0))$  is a turning point. Differentiating G(s) = 0 with respect to s yields, analogously to (1.2),

$$G_u(s)\dot{u}(s) + G_\lambda(s)\lambda(s) = 0.$$
(2.10)

**Proposition 2.4.** Let  $(u(s^0), \lambda(s^0))$  be a simple turning point on the solution path  $\Gamma$ . Then

$$\lambda(s^0) = 0, \quad \dot{u}(s^0) \in \mathcal{N}(G_u(s^0)). \tag{2.11}$$

**Proof.** Assume that  $\dot{\lambda}(s^0) \neq 0$ . Then

$$G_{\lambda}(s^0) = -rac{G_u(s^0)\dot{u}(s^0)}{\dot{\lambda}(s^0)},$$

which contradicts (2.7). Substituting  $\dot{\lambda}(s^0) = 0$  into (2.10) yields  $G_u(s^0)\dot{u}(s^0) = 0$  and the proposition follows.  $\Box$ 

The null spaces of  $G_u(s^0)$  and  $G_u^T(s^0)$  are of the same dimension. For future reference we introduce basis vectors  $\phi$  and  $\psi$  of these spaces, i.e.,

$$\mathcal{N}(G_{u}(s^{0})) = \operatorname{span}\{\phi\}, \quad \mathcal{N}(G_{u}^{\mathrm{T}}(s^{0})) = \operatorname{span}\{\psi\}, \quad \|\phi\| = \|\psi\| = 1.$$
(2.12)

Since  $G_u(s)$  is singular at a turning point, Newton's method cannot be applied with arclength parameterization to continue the solution path across a turning point. This difficulty can be overcome

by imposing an additional constraint. Recall that the unit tangent vector of  $\Gamma$  at  $(u(s^0), \lambda(s^0))$  is given by  $(\dot{u}(s^0), \dot{\lambda}(s^0))$ , cf. (2.9). The equation of a plane orthogonal to the unit tangent at a distance  $\Delta s$ from the point  $(u^0, \lambda^0)$  is given by  $N(u, \lambda, \Delta s) = 0$ , with

$$N(u,\lambda,\Delta s) := \dot{u}^{0\mathrm{T}}(u-u^0) + \dot{\lambda}^0(\lambda-\lambda^0) - \Delta s$$

This plane intersects the path  $\Gamma$  provided that the curvature of  $\Gamma$  at  $(u^0, \lambda^0)$  or  $\Delta s$  are sufficiently small. Thus, the point of intersection between the path and the plane satisfies the nonlinear system of equations

$$G(u,\lambda) = 0, \tag{2.13}$$

$$N(u,\lambda,\Delta s) = 0. \tag{2.14}$$

The solution of these equations by Newton's method yields iterates

$$(u_{k+1},\lambda_{k+1}):=(u_k+\Delta u,\lambda_k+\Delta\lambda),$$

where  $\Delta u$  and  $\Delta \lambda$  satisfy the linear system of equations

$$\begin{bmatrix} G_{u}^{k} & G_{\lambda}^{k} \\ \dot{u}^{\,0T} & \dot{\lambda}^{0} \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} -G^{k} \\ -N^{k} \end{bmatrix}.$$
(2.15)

Here and below, we use the notation

$$G^{k} = G(u_{k}, \lambda_{k}), \quad G^{k}_{u} = G_{u}(u_{k}, \lambda_{k}), \quad G^{k}_{\lambda} = G_{\lambda}(u_{k}, \lambda_{k}),$$
  

$$N^{k} = N(u_{k}, \lambda_{k}, \Delta s), \quad \dot{u}^{0} = \dot{u}(s^{0}), \quad \dot{\lambda}^{0} = \dot{\lambda}(s^{0}).$$
(2.16)

The solution of (2.15) exists and is unique even if  $G_u^k$  is singular, provided that the matrix

$$\tilde{G}^{k} := \begin{bmatrix} G_{u}^{k} & G_{\lambda}^{k} \\ \dot{u}^{0T} & \dot{\lambda}^{0} \end{bmatrix}$$
(2.17)

is nonsingular. Necessary and sufficient conditions for  $\tilde{G}^k$  to be nonsingular are discussed in Section 2.2 below.

Following Keller [16], we refer to the parameterization which uses the plane normal to the path as pseudo-arclength parameterization. Although the pseudo-arclength parameterization is usually applied to singular points, it can be used also at regular points.

Constraint (2.14) is advocated by Keller [16] and will be used in the numerical examples of Section 5. Other constraints, among them Eq. (2.38) below, have also been proposed in the literature; see e.g., [1,15] for discussions.

We turn to bifurcation points  $(u^0, \lambda^0)$  of the solution path and assume that the Jacobian  $G_u = G_u(u^0, \lambda^0)$  satisfies (2.6) and (2.8).

**Proposition 2.5.** Let  $(u^0, \lambda^0)$  be a bifurcation point on the solution path. Then

$$\mathcal{N}([G_u^0, G_\lambda^0]) = \operatorname{span}\left\{ \begin{bmatrix} \phi \\ 0 \end{bmatrix}, \begin{bmatrix} v \\ 1 \end{bmatrix} \right\},\$$

where  $\phi$  satisfies (2.12) and v is the unique vector determined by

$$G_{u}^{0}v = -G_{\lambda}^{0}, \quad \phi^{\mathrm{T}}v = 0.$$
(2.18)

Moreover, for some 
$$\alpha \in \mathbb{R}$$
,  
 $\dot{\mu}^0 = \alpha \phi + \dot{\lambda}^0 v.$  (2.19)

**Proof.** It follows from (2.8) that there is a vector w that satisfies

$$G_{\mu}^{0}w = -G_{\mu}^{0}. (2.20)$$

In view of (2.8) and (2.12), the general solution of (2.20) is given by  $w(\gamma) = w + \gamma \phi$  for  $\gamma \in \mathbb{R}$ . Assume first that  $G_{\lambda}^{0} \neq 0$ . Then the vectors w and  $\phi$  are linearly independent. Therefore, there is a unique constant  $\gamma$ , such that  $v = w(\gamma)$  and  $\phi$  are orthogonal. On the other hand, if  $G_{\lambda}^{0} = 0$ , then Eqs. (2.6) and (2.18) imply that v = 0.

We turn to the proof of (2.19). Substitute  $G^0_{\mu}v = -G^0_{\lambda}$  into (2.10) for  $s = s^0$ . We obtain

$$0 = G_u^0 \dot{u}^0 + G_\lambda^0 \dot{\lambda}^0 = G_u^0 \dot{u}^0 - (G_u^0 v) \dot{\lambda}^0,$$

which in view of (2.12) shows (2.19).  $\Box$ 

Assume that the conditions of Proposition 2.5 hold. Differentiate (2.10) with respect to s and evaluate the expression obtained at  $s = s^0$ . This yields

$$G_{u}^{0}\ddot{u}^{0} + G_{u}^{0}\ddot{\lambda}^{0} + G_{uu}^{0}\dot{u}^{0}\dot{u}^{0} + 2G_{u\lambda}^{0}\dot{u}^{0}\dot{\lambda}^{0} + G_{\lambda\lambda}^{0}\dot{\lambda}^{0}\dot{\lambda}^{0} = 0, \qquad (2.21)$$

where  $\ddot{u}^0$  and  $\ddot{\lambda}^0$  denote second-order derivatives of u(s) and  $\lambda(s)$  with respect to *s* evaluated  $s = s^0$ . Multiply (2.21) by  $\psi^T$  from the left and recall that  $\psi^T G_u^0 = 0$  to obtain

$$\psi^{\mathrm{T}} G^{0}_{uu} \dot{u}^{0} \dot{u}^{0} + 2 \psi^{\mathrm{T}} G^{0}_{u\lambda} \dot{u}^{0} \dot{\lambda}^{0} + \psi^{\mathrm{T}} G^{0}_{\lambda\lambda} \dot{\lambda}^{0} \dot{\lambda}^{0} = 0$$

Replacing  $\dot{u}^0$  by the right-hand side of (2.19),  $\dot{\lambda}^0$  by  $\beta$ , and letting

$$a_{11} := \psi^{\mathrm{T}} G^{0}_{uu} \phi \phi,$$
  

$$a_{12} := \psi^{\mathrm{T}} G^{0}_{uu} \phi v + \psi^{\mathrm{T}} G^{0}_{u\lambda} \phi,$$
  

$$a_{22} := \psi^{\mathrm{T}} G^{0}_{uu} vv + 2\psi^{\mathrm{T}} G^{0}_{u\lambda} v + \psi^{\mathrm{T}} G^{0}_{\lambda\lambda},$$

yields

$$a_{11}\alpha^2 + a_{12}\alpha\beta + a_{22}\beta^2 = 0. \tag{2.22}$$

Eq. (2.22) is usually referred to as the algebraic bifurcation equation. It can be shown, see, e.g., [16, Section 5.20] that its discriminant  $D := a_{12}^2 - 4a_{11}a_{22}$  is nonnegative. If D > 0, then (2.22) has two distinct roots,  $(\alpha_1, \beta_1)$  and  $(\alpha_2, \beta_2)$ . Each root  $(\alpha_*, \beta_*)$  corresponds to a smooth solution path for (1.1) in a neighborhood of the bifurcation point

$$u(s) = u^{0} + (s - s^{0})[\alpha_{*}(s)v + \beta_{*}(s)\phi] + (s - s^{0})^{2}w_{*}(s),$$
  
$$\lambda(s) = \lambda^{0} + (s - s^{0})\alpha_{*}(s),$$

where

$$\psi^{\mathrm{T}} w_*(s) = 0, \quad \alpha_*(s^0) = \alpha_*, \quad \beta_*(s^0) = \beta_*.$$

The quantity  $\psi^{T} G_{uu}^{0}$ , required for the evaluation of the coefficients  $a_{ij}$  in (2.22), is in general not available. Typically, approximations of  $\psi^{T} G_{uu}^{0}$  in terms of  $G_{u}$  are used in computations.

The perturbed bifurcation method avoids the computation of these approximants as well as the solution of the algebraic bifurcation Eq. (2.22). This method is based on the observation that the set of points in the domain of  $G(u, \lambda)$  where the Jacobian  $G_u(u, \lambda)$  is nonsingular are dense and the Jacobian is singular at bifurcation points. Therefore, if  $(u^0, \lambda^0)$  is not a regular point, then we can perturb the right-hand side in Eq. (1.1), i.e., we replace (1.1) by

$$G(u,\lambda) = \varepsilon q, \quad u = u(\lambda),$$
 (2.23)

where  $q \in \mathbb{R}^n$ , ||q|| = 1 and  $\varepsilon \neq 0$ , so that  $G_u$  is nonsingular at the solution of (2.23). Upon deletion of a small neighborhood containing in its interior the bifurcation point  $(u^0, \lambda^0)$ , the two smooth solutions branches intersecting at the bifurcation point generate four different solution branches. It can be shown that the two solution branches which belong to the same path lie adjacent to each other, and therefore paths do not cross in the neighborhood of a bifurcation point. Hence, path following across a bifurcation point can be viewed as a limit case of following two regular paths. Further details on the perturbed bifurcation method are discussed by Georg [13] and Keller [16].

# 2.2. Computation

In this subsection we consider the quantities that need to be computed for path following. The algorithms of Section 4 for computing these quantities are based on the discussion of the present subsection. At a regular point  $(u^0, \lambda^0)$  of a solution path, we compute an initial approximate solution of (1.1) by Euler's method (2.3), where  $\dot{u}_0$  is defined in (2.2). The vector  $\dot{u}_0$  is determined by solving the linear system of equations (2.2). This system has a unique solution at any regular point on the solution path, because the Jacobian matrix  $G_u^0$  is nonsingular there.

The iterates  $u_k$  of Newton's method are solutions of the linear systems (2.5). For each value of  $\lambda$  several systems (2.5) may have to be solved. It is important to monitor if any Jacobian matrix  $G_u^k$  determined during the iterations with Newton's method is singular, because a singular Jacobian may imply that (2.5) does not have a solution.

If  $G_u^k$  is singular and  $(u_k, \lambda)$  satisfies (1.1), then we are either at a bifurcation point or at a turning point. In either case we cannot use Newton's method to compute the next point on the path. However, if the eigenvector corresponding to the zero eigenvalue of the Jacobian matrix is available, then we can introduce a pseudo-arclength parameterization and use the bordering algorithm to find the next point on the path. The bordering algorithm requires that we solve linear systems of equations of the form (2.15). Conditions under which these linear systems have a unique solution are stated in the following proposition.

**Proposition 2.6.** Let  $\tilde{G}^k$  be defined by (2.17). The following situations can be distinguished: (i) If  $G_u^k$  is nonsingular, then  $\tilde{G}^k$  is nonsingular if and only if

$$\dot{\lambda}^0 - \dot{u}^{0\mathrm{T}} (G_u^k)^{-1} G_{\lambda}^k \neq 0.$$
(2.24)

- (ii) If  $G_u^k$  is singular and dim  $\mathcal{N}(G_u^k) = 1$ , then  $\tilde{G}^k$  is nonsingular if and only if  $G_{\lambda}^k \notin \mathscr{R}(G_u^k)$  and  $\dot{u}^0 \notin \mathscr{R}((G_u^k)^T)$ .
- (iii) If  $G_u^k$  is singular and dim  $\mathcal{N}(G_u^k) > 1$ , then  $\tilde{G}^k$  is singular.

**Proof.** The proposition follows by elementary matrix manipulations; see, e.g., [15] or [16, p. 76]. Note that the left-hand side of (2.24) is a Schur complement. Assume that both  $(u^0, \lambda^0)$  and  $(u^k, \lambda^k)$  lie on a solution path for (1.1). Then (2.24) expresses that the tangents  $(\dot{u}^0, \dot{\lambda}^0)$  and  $(\dot{u}^k, \dot{\lambda}^k)$  must not be orthogonal.  $\Box$ 

Several solution methods for linear systems of equations of form (2.15) will be discussed. Different methods are tailored to matrices (2.17) with different properties.

We use notation (2.16). It is convenient to write (2.15) in the form

$$G_{u}^{k}\Delta u + G_{\lambda}^{k}\Delta\lambda = -G^{k}, \qquad (2.25)$$

$$\dot{u}^{0\mathrm{T}}\Delta u + \dot{\lambda}^0 \Delta \lambda = -N^k. \tag{2.26}$$

Assume that  $G_u^k$  is nonsingular and  $\dot{\lambda}^0 \neq 0$ . We obtain from (2.26) that

$$\Delta \lambda = \frac{1}{\dot{\lambda}^0} (-N^k - \dot{u}^{0T} \Delta u), \qquad (2.27)$$

which, substituted into (2.25), yields

$$\left(G_u^k - \frac{1}{\dot{\lambda}^0} (G_\lambda^k \dot{u}^{0\mathrm{T}})\right) \Delta u = -G^k + \frac{1}{\dot{\lambda}^0} G_\lambda^k N^k.$$
(2.28)

Thus, when  $\dot{\lambda}^0 \neq 0$ , we can compute  $\Delta u$  by solving (2.28) and  $\Delta \lambda$  from (2.27). We will refer to this as the *bordering algorithm for regular points*. The matrix in (2.28) is a rank-one modification of  $G_u^k$ . It is nonsingular if and only if (2.24) is satisfied, i.e., if and only if the system of equations (2.25)–(2.26) has a unique solution.

The bordering algorithm for a regular point described above cannot be used at a turning point  $(u^0, \lambda^0)$  since  $\dot{\lambda}^0 = 0$  there. We now derive a simple solution method for system (2.25)–(2.26) for k=0 under the assumption that  $(u^0, \lambda^0)$  is a turning point on the solution path. Note that the right-hand side of (2.25) vanishes for k = 0 because the turning point is on the solution path. Multiply (2.25) by  $\psi^T$  from the left and recall that  $\psi^T G_u^0 = 0$ ; see (2.12). We obtain

$$\psi^{\mathrm{T}}G^{0}_{\lambda}\Delta\lambda = 0. \tag{2.29}$$

The factor  $\psi^{T}G_{\lambda}^{0}$  does not vanish because  $\psi \in \mathcal{N}(G_{u}^{0T})$  and it follows from (2.7) that  $G_{\lambda}^{0} \notin \mathcal{N}(G_{u}^{0T})^{\perp}$ . We conclude that  $\Delta \lambda = 0$ . Eq. (2.25) simplifies to

$$G^u_u \Delta u = 0, \tag{2.30}$$

which is satisfied by

$$\Delta u = \alpha \phi, \quad \forall \alpha \in \mathbb{R}.$$

We determine  $\alpha$  so that  $\Delta u$  satisfies (2.26), i.e.,

$$\alpha = -\frac{N^0}{\dot{\mu}^{0T}\phi}.$$
(2.32)

The denominator in (2.32) does not vanish due to (2.11).

Having determined the corrections  $\Delta u$  and  $\Delta \lambda = 0$  of the turning point  $(u^0, \lambda^0)$  as described above, yields the approximation  $u_1 = u^0 + \Delta u$  and  $\lambda_1 = \lambda^0$  of the solution of (2.13)–(2.14). The subsequent

Newton steps require the solution of linear systems (2.25)–(2.26) for k = 1, 2, ... The Jacobian  $G_u^k$  of these systems is in general nonsingular. We outline how the system (2.25)–(2.26) can be solved when  $G_u^k$  is nonsingular and  $\dot{\lambda}^0 = 0$ . Compute the solutions y and z of the linear systems

$$G_{u}^{k}z = G_{\lambda}^{k}, \tag{2.33}$$

$$G_u^k y = -G^k. (2.34)$$

Then

$$\Delta u = y - z \Delta \lambda \tag{2.35}$$

satisfies (2.25) for arbitrary  $\Delta \lambda \in \mathbb{R}$ . Substituting (2.35) into (2.26) gives

$$\Delta \lambda = \frac{N^k + \dot{u}^{0\mathrm{T}} y}{\dot{u}^{0\mathrm{T}} z}.$$
(2.36)

Thus, we solve (2.25)-(2.26) for k = 0 by using formulas (2.29)-(2.32), and for  $k \ge 1$  by first solving the linear systems (2.33) and (2.34) and then using (2.35)-(2.36). We refer to this method for determining  $\Delta u$  and  $\Delta \lambda$  at a turning point as the *bordering algorithm for simple turning points*.

The unit tangent vector  $(\dot{u}^0, \dot{\lambda}^0)$  to the solution path at  $(u^0, \lambda^0)$  plays a central role in the bordering algorithm. We described how it can be computed. Eqs. (2.9) and (2.10) can be written as

$$G_{u}^{0}\dot{u}^{0} + G_{\lambda}^{0}\dot{\lambda}^{0} = 0, (2.37)$$

$$\|\dot{u}^0\|^2 + |\dot{\lambda}^0|^2 = 1.$$
(2.38)

We first consider the case when  $G_u^0$  is nonsingular and  $\dot{\lambda}^0 \neq 0$ . Then we can express  $\dot{u}^0$  as

$$\dot{u}^0 = \dot{\lambda}^0 \chi, \tag{2.39}$$

where  $\chi$  solves the linear system

$$G_u^0 \chi = -G_\lambda^0. \tag{2.40}$$

Eqs. (2.38) and (2.39) yield

$$\dot{\lambda}^0 = \frac{\pm 1}{\sqrt{1 + \|\chi\|^2}}.$$
(2.41)

The sign of  $\dot{\lambda}^0$  is chosen so that the tangent vector points in the positive direction of the path. If  $(\dot{u}^{-1}, \dot{\lambda}^{-1})$  is the unit tangent vector at a previous point on the path, we choose the sign in the right-hand side of (2.41) so that the cosine of the angle between  $(\dot{u}^0, \dot{\lambda}^0)$  and  $(\dot{u}^{-1}, \dot{\lambda}^{-1})$  is positive, i.e.,

$$\dot{u}^{-1}\dot{u}^{0} + \dot{\lambda}^{-1}\dot{\lambda}^{0} > 0 \tag{2.42}$$

or, equivalently,

$$\dot{\lambda}^0(\dot{u}^{-1}\chi+\dot{\lambda}^{-1})>0.$$

If the wrong orientation of the tangent vector is chosen, then the bordering algorithm will backtrack the path already computed.

We turn to the solution on (2.37)-(2.38) at a turning point. Then  $G_u^0$  is singular,  $\dot{\lambda} = 0$  and  $\dot{u}^0 \in \mathcal{N}(G_u^0)$ . The computation of the unit tangent vector at a turning point amounts to computing the eigenvector of Euclidean norm one of the Jacobian associated with the zero eigenvalue and choosing its orientation according to (2.42).

We conclude this section with a discussion of the solution of linear systems with a nonsingular matrix (2.17), whose  $n \times n$  leading principal submatrix  $G_u^k$  is singular with dim  $\mathcal{N}(G_u^k) = 1$ . The right-hand side is a general vector. Thus, consider the linear system

$$G_{u}^{k}x + G_{\lambda}^{k}\xi = g, \qquad (2.43)$$

$$\dot{u}^{0\mathrm{T}}x + \dot{\lambda}^{0}\xi = \gamma \tag{2.44}$$

and let the vectors  $\phi$  and  $\psi$  satisfy (2.12). Then the solution of (2.43)–(2.44) can be expressed as

$$x = y - \xi z + \alpha \phi, \tag{2.45}$$

$$\xi = \frac{\psi^1 g}{\psi^{\mathrm{T}} G^k_{\lambda}},\tag{2.46}$$

where

$$G_u^k y = g - (\psi^{\mathrm{T}} g)\psi, \quad G_u^k z = G_\lambda^k - (\psi^{\mathrm{T}} G_\lambda^k)\psi, \qquad (2.47)$$

$$\phi^{\mathrm{T}} y = \phi^{\mathrm{T}} z = 0, \tag{2.48}$$

$$\alpha = \frac{\gamma - \dot{\lambda}^0 \xi - \dot{u}^{0T} (y - \xi z)}{\dot{u}^{0T} \phi}.$$
(2.49)

This can be seen as follows. Multiplying Eq. (2.43) by  $\psi^{T}$  yields (2.46). The denominator is nonvanishing because  $\psi \in \mathscr{R}(G_{u}^{k})^{\perp}$  and, by Proposition 2.6,  $G_{\lambda}^{k} \notin \mathscr{R}(G_{u}^{k})$ . The linear systems of equations (2.47) are consistent and the orthogonality conditions (2.48) determine the solutions *y* and *z* of these systems uniquely. Eq. (2.43) is satisfied by *x* and  $\xi$  given by (2.45)–(2.46) for any  $\alpha \in \mathbb{R}$ . Eq. (2.49) determines  $\alpha$  so that Eq. (2.44) is satisfied. It follows from Proposition 2.6 that the denominator in (2.49) does not vanish.

# 3. The IRBL method

We outline the implicitly restarted block-Lanczos (IRBL) method for the computation of a few of the smallest eigenvalues and associated eigenvectors of a symmetric matrix  $A \in \mathbb{R}^{n \times n}$ . This method is used in the algorithms for path following described in Section 4. In the applications discussed there, A is a Jacobian matrix  $G_u$  associated with the nonlinear system of equations (1.1). The IRBL method helps us determine whether the Jacobian matrix is singular. For singular matrices, the method yields a basis of the null space. It is important to detect singular points on the solution path during path following, and knowledge of the null space of singular Jacobian matrices helps us to follow the path across a singular point.

The IRBL method is an iterative scheme for the computation of a few eigenvalues and associated eigenvectors in a specified portion of the spectrum. It is based on the recurrence relations of the

block-Lanczos algorithm. The IRBL method was introduced in [4]; here we only discuss aspects pertinent for the application considered in Section 4. When the block-size r is chosen to be one, the IRBL method simplifies to the implicitly Restarted Lanczos (IRL) method described in [2,6]. In our experience the IRBL method is better suited for computing eigenvalue–eigenvector pairs associated with multiple or close eigenvalues than the IRL method.

In the present paper, we choose the block-size r to be the number of desired eigenpairs. Let  $\{v_j\}_{j=1}^r$  be a set of orthonormal vectors in  $\mathbb{R}^n$  and introduce the matrix  $V_r = [v_1, v_2, \dots, v_r]$ . Assume for simplicity that the block-Lanczos process does not break down. Then m steps the block-Lanczos process yield a symmetric block-tridiagonal matrix  $T_{mr} \in \mathbb{R}^{mr \times mr}$  with  $r \times r$  blocks and upper triangular subdiagonal blocks, such that

$$AV_{mr} = V_{mr}T_{mr} + F_r E_r^{\rm T},\tag{3.1}$$

where  $V_{mr} \in \mathbb{R}^{n \times mr}$ ,  $V_{mr}I_{mr \times r} = V_r$ ,  $V_{mr}^{\mathrm{T}}V_{mr} = I_{mr}$  and  $F_r \in \mathbb{R}^{n \times r}$  satisfies  $V_{mr}^{\mathrm{T}}F_r = 0$ . Here  $I_{mr}$  denotes the  $mr \times mr$  identity matrix,  $I_{mr \times r} \in \mathbb{R}^{mr \times r}$  consists of the first *r* columns of  $I_{mr}$  and  $E_r$  consists of the *r* last columns of  $I_{mr}$ . The columns of  $V_{mr}$  span the Krylov subspace  $\mathscr{K}_{mr}(A, V_r) := \operatorname{span}\{V_r, AV_r, \ldots, A^{m-1}V_r\}$  and

$$T_{mr} = V_{mr}^{\mathrm{T}} A V_{mr}.$$

Introduce the spectral factorization

$$T_{mr} = Y_{mr} \Theta_{mr} Y_{mr}^{\mathrm{T}},$$

where  $\Theta_{mr} = \text{diag}[\theta_1, \theta_2, \dots, \theta_{mr}]$ ,  $Y_{mr} \in \mathbb{R}^{mr \times mr}$ ,  $Y_{mr}^T Y_{mr} = I_{mr}$ . The eigenvalues  $\theta_1 \leq \theta_2 \leq \cdots \leq \theta_{mr}$  of  $T_{mr}$  approximate eigenvalues of A and are usually referred to as Ritz values. The vectors  $x_j = V_{mr} y_j$ ,  $1 \leq j \leq mr$ , approximate eigenvectors of A and are referred to as Ritz vectors. It follows from (3.1) that the residual error  $Ax_j - x_j\theta_j$  associated with the Ritz pair  $(\theta_j, x_j)$  satisfies

$$||Ax_j - x_j\theta_j|| = ||(AV_{mr} - V_{mr}T_{mr})y_j|| = ||F_r E_r^T y_j||.$$

We say that a Ritz pair  $(\theta_j, x_j)$  is an acceptable approximation of an eigenpair of the matrix A if

$$\|F_r E_r^{\mathrm{T}} y_j\| \leqslant \varepsilon, \tag{3.2}$$

where  $\varepsilon > 0$  is a small constant. Then  $(\theta_j, x_j)$  is an eigenpair of a matrix  $\hat{A} \in \mathbb{R}^{n \times n}$ , such that  $||A - \hat{A}|| \leq \varepsilon$ .

As the value of m increases the Ritz pairs become better approximations of eigenpairs of A. On the other hand each unit increase in m requires that r additional n-vectors be stored. Therefore, when the matrix A is very large and a large number of Lanczos steps are needed, use of secondary computer memory may become necessary. To avoid the slow-down which typically occurs when using secondary storage, a restarting scheme is employed. The recurrence relation of the IRBL method described in [4] for restarting allows us to compute

$$\hat{V}_r = \psi_m(A) V_r R \tag{3.3}$$

from the block-Lanczos decomposition (3.1) without additional evaluations of matrix-vector products with the matrix A. Here  $\psi_m(t)$  is a polynomial of degree m, R is an upper triangular matrix and  $\hat{V}_r$ is an  $n \times r$  matrix with orthonormal columns which will be used as the initial block for the next block-Lanczos recursion. We seek to choose the polynomials  $\psi_m$  so that the columns of  $\hat{V}_r$  are in, or close to, the invariant subspace of A associated with the desired eigenvalues. To achieve this, we allocate the zeros  $z_1, z_2, ..., z_m$  of  $\psi_m$  in one or two intervals away from the wanted portion of the spectrum. For example, if the smallest eigenvalues of A are desired, then we choose the  $z_j$  to be in the interval  $[\theta_{m(r-1)}, \theta_{mr}]$ . When we wish to determine a few nearby nonextreme eigenvalues, the zeros are allocated in two interval, one on each side of the set of desired eigenvalues. Details on how to select the zeros can be found in [2–4].

Once the matrix  $\hat{V}_r$  and its columns have been orthogonalized against each other as well as against any converged eigenvectors, we compute a new block-Lanczos factorization (3.1) with  $V_r := \hat{V}_r$ . We repeat this process until *r* eigenpairs have been determined. The reliability of this scheme is illustrated by computed examples reported in [2–4].

We conclude this section with a few implementation issues. In each application of the IRBL method we determine the initial matrix  $V_r$  by orthogonalizing r columns of random numbers from the standard normal distribution; see Ericsson and Ruhe [11] for an explanation of the advantage of using normally distributed instead of uniformly distributed random numbers. Further, we note that in order to reduce data movement, it may be advantageous to implement the block-Lanczos process so that the r vectors in a block are multiplied by the matrix A simultaneously.

Columns of the matrix  $V_{mr}$  that are associated with the same block of r vectors are generated by first applying a three-term recursion formula determined by (3.1) followed by orthogonalization of the columns in the same block. In exact arithmetic and in the absence of break-down, these computations give a matrix  $V_{mr}$  with orthogonal columns. In order to secure orthogonality in the presence of round-off errors, we explicitly reorthogonalize the generated columns to all already available columns of the matrix. If the new columns generated fail to be numerically orthogonal after one reorthogonalization, then this signals that they are linearly dependent and a break down of the block-Lanczos process has occurred. Details of how to handle break downs will be discussed elsewhere. Here it suffices to say that break downs are easy to handle, and a procedure for this has been implemented in the code used for the numerical experiments reported in Section 5.

# 4. Algorithms for path following

We present several iterative methods for computing a solution path for nonlinear systems of equations (1.1) with a symmetric Jacobian  $G_u$ . First we describe the iterative method proposed in [5] for the simultaneous computation of a few eigenpairs associated with the eigenvalues of smallest magnitude and the solution of a linear system.

Consider the nonlinear system of equations (1.1) and assume that  $(u^0, \lambda^0)$  is a regular point on a solution path. We would like to determine the point  $(u^1, \lambda^1)$  on the path, where  $u^1 = u(\lambda^1)$  and  $\lambda^1 := \lambda^0 + \Delta \lambda$ . The application of the Euler–Newton continuation method requires the solution of a sequence of linear systems of equations of the form (2.5) with  $\lambda = \lambda^1$ . Assume that the iterates  $u_k$ defined by (2.4) converge to  $u^1$  as k increases. If  $G_u(u^1, \lambda^1)$  is singular, then by a continuity argument the matrices  $G_u(u_k, \lambda^1)$  for k sufficiently large have an eigenvalue close to the origin. Therefore, by tracking the eigenvalues of smallest magnitude of the Jacobian matrices  $G_u(u_k, \lambda^1)$  while computing iterates (2.4), we can detect when we approach a singular point on the solution path. Algorithm 4.1 below determines an approximate solution of (2.5) while simultaneously computing r Ritz pairs  $\{(\theta_\ell^k, x_\ell^k)\}_{\ell=1}^r$  that approximate eigenpairs of  $G_u(u_k, \lambda^1)$  associated with the r smallest eigenvalues. Algorithm 4.1. Simultaneous solution of linear system and eigenproblem:

Input:  $\lambda^1$ ,  $u_0$ ,  $A := G_u(u_k, \lambda^1)$ ,  $b := -G(u_k, \lambda^1)$ ,  $m, r, V_r, \varepsilon$ .

*Output*: Approximations  $\{\theta_{\ell}^k\}_{\ell=1}^r$  of the *r* smallest eigenvalues of  $G_u(u_k, \lambda^1)$ , approximations  $\{x_{\ell}^k\}_{\ell=1}^r$  of associated orthonormal approximate eigenvectors and an approximate solution  $\Delta \tilde{u}$  of (2.5) with  $\lambda = \lambda^1$ .

- 1.  $\Delta \tilde{u} := 0$ .
- 2. for  $\mu := 1, 2, ...$  until r approximate eigenpairs found
- 3. Orthogonalize  $V_r$  against already computed approximate eigenvectors.
- 4. Compute block-Lanczos decomposition (3.1) with initial block  $V_r$ .
- 5. Update approximate solution of (2.5):

Solve  $T_{mr}\Delta y = V_{mr}^{\mathrm{T}}b$ ,  $\Delta \tilde{u} := \Delta \tilde{u} + V_{mr}\Delta y$ .

- 6. Determine Ritz pairs that satisfy (3.2) and store the Ritz vectors. We refer to the stored vectors as approximate eigenvectors and denote them by  $u_{\ell}^{(k)}$ .
- 7. Apply m shifts  $\{z_j\}_{j=1}^m$  to determine the matrix  $\hat{V}_r$  given by (3.3). The  $z_j$  are chosen to be fast Leja shifts described in [3]. Let  $V_r := \hat{V}_r$ .
- 8. endfor
- 9. Improve approximate solution  $\Delta \tilde{u}$  by a conjugate gradient method. Terminate the iterations when the residual error is of norm smaller than  $\varepsilon$ . Denote the computed solution by  $\Delta \tilde{u}$ .

Step 6 of Algorithm 4.1 yields approximations of the r eigenpairs associates with the eigenvalues of smallest magnitude. The solution of (2.5) can be expanded in terms of eigenvectors of the matrix. Step 5 essentially removes eigenvector components associated with the smallest eigenvalues from this expansion. The solution of the linear system that is solved in Step 9 can be expressed in terms of an expansion of eigenvectors associated with the remaining (larger) eigenvalues. Therefore, Step 5 of Algorithm 4.1 can be thought of as preconditioning the linear system (2.5).

In our numerical examples in Section 5, we use the conjugate gradient method designed for the solution of inconsistent linear systems with a symmetric possibly indefinite matrix described in [7].

The matrix  $V_r \in \mathbb{R}^{n \times r}$  required as input for Algorithm 4.1 is assumed to have orthonormal columns. In the very first application of Algorithm 4.1,  $V_r$  is initialized with normally distributed random entries in each column, which then are orthonormalized. In later applications of Algorithm 4.1 the columns of the input matrix  $V_r$  are chosen to be the approximate eigenvectors  $\{w_\ell^k\}_{\ell=1}^r$  determined during the most recent application of the algorithm.

Assume now that  $G_u^0$  is singular with dim  $\mathcal{N}(G_u^0) = 1$ , and let  $\phi$  satisfy (2.12). It follows from the symmetry of the Jacobian and a discussion analogous to the one following (2.29) that the point  $(u^0, \lambda^0)$  is a turning point on the solution path if and only if  $\phi^T G_{\lambda}^0 \neq 0$ . Let  $(u^0, \lambda^0)$  be a turning point. We describe how to determine the pseudo-arclength parameterization required by the bordering algorithm. This parameterization requires that the unit tangent vector

$$(\dot{u}^0\lambda^0) = (\pm\phi, 0) \tag{4.1}$$

be available. If the unit tangent vector  $(\dot{u}^{-1}, \dot{\lambda}^{-1})$  at the previous point on the path is available, then we choose the sign in the right-hand side of (4.1) so that (2.42) is satisfied. When the tangent vector  $(\dot{u}^{-1}, \dot{\lambda}^{-1})$  is not known, we approximate it by the secant  $(u^0, \lambda^0) - (u^{-1}, \lambda^{-1})$ . The solution of (2.25)–(2.26) for k = 0 is given by  $\Delta \lambda = 0$  and (2.31)–(2.32).

The performance of the Euler–Newton predictor–corrector method (2.3)–(2.5) with the natural parameter  $\lambda$  requires that the steps  $\Delta\lambda$  be small in the vicinity of a turning point. Therefore, we switch from the natural to the pseudo-arclength parameterization when we are close to a turning point and compute the next point on the path via the following variant of the bordering algorithm. We first solve the linear systems of equations (2.33) and (2.34) and then determine  $\Delta\lambda$  from

$$\Delta \lambda = \frac{\dot{u}^{0\mathrm{T}} y + N^{k}}{\dot{u}^{0\mathrm{T}} z - \dot{\lambda}^{0}}$$

$$\tag{4.2}$$

and  $\Delta u$  from (2.35). The vector ( $\Delta u, \Delta \lambda$ ) satisfies (2.25)–(2.26). Formula (4.2) simplifies to (2.36) when  $\dot{\lambda}^0 = 0$ . Since in finite precision arithmetic  $|\dot{\lambda}^0|$  is small but possibly nonvanishing at a computed approximate turning point, we use (4.2) instead of (2.36) in computations. We switch from natural to pseudo-arclength parameterization when either the Jacobian has an eigenvalue of magnitude smaller than a tolerance  $\varepsilon_s$  or the step size required with the natural parameterization is below a given threshold. The following algorithm summarizes how to organize the calculation to compute regular points on a solution path across a turning point.

Algorithm 4.2. Path following around a turning point:

Input:  $\lambda^0$ ,  $u^0$ ,  $\dot{\lambda}^0$ ,  $\dot{u}^0$ ,  $G_u(u^0, \lambda^0)$ ,  $\Delta s$ ,  $\varepsilon$ ,  $k_{\text{max}}$ . Output:  $u^1$ ,  $\lambda^1$ . 1. convergence := false,  $u_0 := u^0$ ,  $\lambda_0 := \lambda^0$ . 2. while not convergence, for  $k := 0, 1, ..., k_{\max}$ 3. Solve (2.33) for z and (2.34) for y. 4. *Compute*  $\Delta \lambda$  *from* (4.2). 5. Compute  $\Delta u$  from (2.35). 6.  $u_{k+1} := u_k + \Delta u, \ \lambda_{k+1} := \lambda_k + \Delta \lambda.$ 7. if  $||G(u_{k+1}, \lambda_{k+1})|| < \varepsilon$  then 8. convergence := true,  $u^1 := u_{k+1}$ ,  $\lambda^1 := \lambda_{k+1}$ , exit. 9. 10. endif endfor 11. 12.  $\Delta s := \Delta s/2.$ 

13. endwhile

We turn to the problem of branch switching at a bifurcation point. In finite precision arithmetic computation, we may not be able to determine exact bifurcation points. Instead we are in a position to compute approximations of bifurcation points. We refer to these points as perturbed bifurcation points. The presence of a perturbed bifurcation point on the solution path is signaled numerically by the Jacobian matrix becoming nearly singular and by the magnitude of  $\phi^T G^0_{\lambda}$  being very small.

Let  $(u^0, \lambda^0)$  be a perturbed bifurcation point on the solution path. We discuss how to switch path there. In view of (2.10) the tangent vectors at the bifurcation point are in the null space of the

matrix  $[G_u^0, G_\lambda^0]$  and can, by Proposition 2.5, be written as

$$\tau := \alpha \begin{bmatrix} \phi \\ 0 \end{bmatrix} + \beta \begin{bmatrix} v \\ 1 \end{bmatrix}.$$

A tangent to the active branch at  $(u^0, \lambda^0)$  is determined by  $\beta = \dot{\lambda}^0$  and  $\dot{u}^0 = \alpha \phi + \beta v$ , where the coefficient  $\alpha$  easily can be found by using the orthogonality (2.18). We search for the other branch by moving in a direction parallel to  $\tau$  starting from a point at a distance  $\varepsilon_b$  from  $(u^0, \lambda^0)$  on the normal to  $\tau$  in the plane  $\mathcal{N}([G_u^0, G_\lambda^0])$ . Note that the vector

$$\hat{\tau} := \hat{\alpha} \begin{bmatrix} \phi \\ 0 \end{bmatrix} + \hat{\beta} \begin{bmatrix} v \\ 1 \end{bmatrix}$$

with  $\hat{\alpha} = t\beta(1 + ||v||^2)$  and  $\hat{\beta} = -t\alpha ||\phi||^2$  is orthogonal to  $\tau$  and in  $\mathcal{N}([G_u^0, G_\lambda^0])$  for any  $t \ge 0$ . We choose the scaling factor t so that  $\hat{\tau}$  is of unit length. Thus, we would like to determine a point  $(u^2, \lambda^2) \in \mathbb{R}^{n+1}$  with coordinates of the form

$$u^{2} = u^{0} + \varepsilon_{b}(\hat{\beta}v + \hat{\alpha}\phi) + w,$$
  

$$\lambda^{2} = \lambda^{0} + \varepsilon_{b}\hat{\beta} + \zeta,$$
(4.3)

where  $w \in \mathbb{R}^n$  and  $\zeta \in \mathbb{R}$  are chosen so that

$$G(u^2,\lambda^2) = 0, (4.4)$$

$$N(u^2, \lambda^2) = 0 \tag{4.5}$$

and  $N(u^2, \lambda^2) := (\hat{\beta}v^T + \hat{\alpha}\phi^T)w + \hat{\beta}\zeta$ . Condition (4.5) imposes that the vector  $(w, \zeta)$  is orthogonal to  $\hat{\tau}$ . We compute w and  $\zeta$  by applying Newton's method to the system (4.4)–(4.5) with initial approximate solution w = 0 and  $\zeta = 0$ . The sequence of linear systems of equations obtained in this manner are solved by the methods described in Section 2.

The computations for branch switching at a bifurcation point  $(u^0, \lambda^0)$  are summarized by the following algorithm. The vector  $\phi$  is defined by (2.12).

Algorithm 4.3. Branch switching at a bifurcation point:

Input:  $\lambda^{0}$ ,  $u^{0}$ ,  $\dot{\lambda}^{0}$ ,  $\dot{u}^{0}$ ,  $\phi$ ,  $\varepsilon_{b}$ . Output:  $u^{2}$ ,  $\lambda^{2}$ . 1.  $\alpha := \phi^{T} \dot{u}^{0}$ ,  $\beta := \dot{\lambda}^{0}$ . 2.  $v := (\dot{u}^{0} - \alpha \phi)/\beta$ . 3.  $\hat{\alpha} := t\beta(1 + ||v||^{2})$ ,  $\hat{\beta} := -t\alpha ||\phi||^{2}$  with t > 0 chosen so that  $\hat{\alpha}^{2} + \hat{\beta}^{2} = 1$ . 4. Solve (4.4)–(4.5) for  $(u^{2}, \lambda^{2})$  given by (4.3) using Newton's method.

We conclude this section with an algorithm that outlines how to compute a new point on a solution path for (1.1) with special handling of turning and bifurcation points.

Algorithm 4.4. Iterative method for path following:

Input:  $\lambda^0$ ,  $\lambda^1$ ,  $u^0$ ,  $\varepsilon$ ,  $\varepsilon_n$ ,  $\varepsilon_s$ ,  $\theta_{\min} = eigenvector \ of \ smallest \ magnitude \ of \ G_u^0$ . Output:  $u^1$ .

- 1. *if*  $|\theta_{\min}| \ge \varepsilon_s$  *then*
- %  $(u^0, \lambda^0)$  is a regular point.
- 2. Compute the Euler predictor  $u_0$ .
- 3. Use Newton's method to find  $u^1$ , such that  $||G(u^1, \lambda^1)|| < \varepsilon$ . Compute Newton iterates by Algorithm 4.1.
- 4. *else*

5.

% Test whether we are at a turning point or at a bifurcation point.

- if  $|\phi^{\mathrm{T}}G^{0}_{\lambda}| > \varepsilon_{n}$  then
  - % Turning point
- 6. Compute  $(u^1, \lambda^1)$  by Algorithm 4.2.
- 7. *else*

% Bifurcation point: first continue on active branch then switch branch.

- 8. Compute  $(u^1, \lambda^1)$  by Algorithm 4.2.
- 9. Switch branch by Algorithm 4.3. Compute points on other branch.
- 10. endif
- 11. endif

# 5. Numerical examples

This section presents computations with the algorithms described in Section 4. The algorithms were implemented in MATLAB on a Silicon Graphics workstation. It is the purpose of the examples to illustrate the ease of use of the algorithms for path following in an interactive computing environment.

We report the number of times the Jacobian is used to multiply one or several *n*-vectors as a measure of the computational effort required. However, our algorithms have not (yet) been tuned for efficiency, and we expect that the computational effort required by carefully designed algorithms to be smaller. We return to this issue after the examples.

Example 5.1. Consider the nonlinear eigenvalue problem

$$-\mathscr{L}U - \lambda \sin(U) = 0 \quad \text{in } \Omega,$$
  

$$U = 0 \quad \text{on } \partial\Omega,$$
(5.1)

where  $\Omega := \{(s,t): 0 \le s \le 1, 0 \le t \le 1\}$  and  $\partial \Omega$  denotes the boundary of  $\Omega$ . We discretize  $\Omega$  by a uniform grid with grid points  $s_j := (j - 1)h$  and  $t_j := (j - 1)h$  for  $1 \le j \le \ell + 1$  and  $h = 1/\ell$ . The Laplacian  $\mathscr{L}$  is approximated by the standard five-point stencil. This yields a system of  $n := (\ell - 1)^2$  nonlinear algebraic equations of the form (1.1). The entries of the solution  $u \in \mathbb{R}^n$  of (1.1) approximate U(s,t) at the nodes  $s_j$  and  $t_j$ .

The discretization error is  $\mathcal{O}(h^2)$  and we let  $\varepsilon := h^2$  in the algorithms of Section 4. In particular, a Jacobian  $G_u$  is considered singular when it has an eigenvalue of magnitude smaller than  $h^2$ . Iterations

Table 1					
Legend	for	Examples	5.1	and	5.2

EP	Computation of eigenpairs of Jacobian by the IRBL algorithm.
NW	Newton's method for solution of (1.1), eigenpairs of Jacobian by Algorithm 4.1.
EN	Continuation by the Euler–Newton method, step length $\Delta \lambda$ .
SB	Switch branch using Algorithm 4.3, step length $\varepsilon_b$ .
BR	Continuation by bordering using Algorithm 4.2, Jacobian regular, step length $\Delta s$ .
BS	Continuation by bordering using Algorithm 4.2, Jacobian singular, step length $\Delta s$ .

Table 2

\_ . .

Example	5.1:	Solution	path	with	bifurcation	point
Example	5.1.	Solution	pum	** 1111	onurcation	point

Step	Comput.	λ	Step length	Smallest eig. val.	Solut. norm	Matrix acces.	Mat.–vec. prod.	CG iter.
a	EP	18.0000		1.6987	0	58	113	0
b	EN	19.6987	1.6987	$-4 \cdot 10^{-5}$	0	72	137	0
с	SB	19.6987*	0.1	$5 \cdot 10^{-4}$	$1 \cdot 10^{-1}$	139	214	53
d	BS	19.6987	0.1	$2 \cdot 10^{-3}$	$2 \cdot 10^{-1}$	200	285	100
e	BR	19.7003*	0.1	$5 \cdot 10^{-3}$	$3 \cdot 10^{-1}$	223	318	109
f	BR	19.7016	0.1	$6 \cdot 10^{-3}$	$4 \cdot 10^{-1}$	256	371	115
g	BR	19.7136*	0.1	$3 \cdot 10^{-2}$	$9 \cdot 10^{-1}$	288	423	120
h	EN	19.8136*	0.1	$3 \cdot 10^{-1}$	$2 \cdot 10^0$	336	511	128
i	EN	20.3136*	0.5	$1 \cdot 10^0$	$6 \cdot 10^0$	434	679	138

with Newton's method are terminated as soon as an iterate has been found that gives a value of G of norm less than  $h^2$ . In the present example  $\ell = 20$ ,  $h = 5 \cdot 10^{-2}$  and  $\varepsilon = 2.5 \cdot 10^{-3}$ .

We use the algorithms of Section 4 to determine a bifurcation point and switch path in an interactive computing environment. The computations carried out are shown in the column "comput." of Table 2. The abbreviations used are explained in Table 1.

The boundary value problem (5.1) has the trivial solution U(s,t) = 0 for any value of  $\lambda$ , and the discretized problem has the solution u = 0. The Jacobian matrices associated with the solution u = 0are singular when  $\lambda$  is an eigenvalue of the discretized Laplacian. We choose  $(u, \lambda) = (0, 18)$  as initial point on the solution path and use the IRBL algorithm with block-size r=2 and m=5 block-Lanczos steps between the restarts to compute the two smallest eigenvalues and associated eigenvectors of the Jacobian matrix. This is Step (a) in Table 2. The table shows the computed approximation of the eigenvalue of the Jacobian closest to the origin in the column labeled "Smallest eig. val." In all computations reported the eigenvalue closest to the origin is also the smallest one. Two measures of the computational effort required for the computation of the eigenpairs are reported in the columns "Matrix acces." and "Mat.-vec. prod." The former column reports the number of matrix accesses, i.e., the number of times the Jacobian is multiplied by a vector or a block of r vectors. This count is of interest when the entries of the Jacobian are evaluated every time they are used in order to reduce the computer storage required, and measures the number of times each matrix entry has to be computed. This approach is used in the code for liquid crystal modeling described in [4] in order to reduce the storage requirement for the Jacobian, as well as in finite element codes for large-scale problems. The column "Mat.-vec. prod." shows the number of times the Jacobian matrix is multiplied by an

*n*-vector. Multiplying a block of r *n*-vectors by the Jacobian counts as r matrix–vector products. Whether this count is relevant depends on the size of the problem and the on architecture of the computer used. For large-scale problems, this count, in general, is of less interest than the number of matrix accesses.

We add the computed approximation 1.6987 of the smallest eigenvalue of the Jacobian determined in Step (a) to  $\lambda$  in order to obtain a nearly singular Jacobian matrix and take a step with the Euler-Newton method with  $\Delta \lambda = 1.6987$ . This is Step (b) in Table 2. The table reports the cumulative number of matrix accesses and matrix-vector products required. Thus, the computations for Step (b) require 16 matrix accesses and the evaluation of 24 matrix-vector products. We do not use the fact that the eigenvectors for the Jacobian at  $\lambda = 18$  and 19.6987 are the same. The matrix-vector products reported are used to take a step with the Euler-Newton method and to verify that *r* eigenpairs and a solution *u* for  $\lambda = 19.6987$  have been determined to desired accuracy.

The smallest eigenvalue of the Jacobian determined in this manner is in magnitude smaller than  $\varepsilon = h^2$ . We therefore consider the Jacobian singular. Algorithm 4.3 with  $\varepsilon_b = 0.1$  is applied to determine a nontrivial solution u of (1.1) for  $\lambda$  close to 19.6987. Note that if u solves (1.1), then so does -u. The arithmetic work required is reported in step (c) in Table 2. Algorithm 4.1 is used with m = 5 block-Lanczos steps between restarts. The column "CG iter." displays the number of iterations, 53, carried out with the conjugate gradient method in Step 9 of Algorithm 4.1. These iterations are included in the count of matrix accesses and matrix–vector products reported in the table. Subsequent entries of the column "CG iter." report the cumulative number of iterations. The column "Solut. norm" of Table 2 displays the Euclidean norm of the computed solution u. Starred parameter values in the column " $\lambda$ " indicate that the solution is plotted in Fig. 1. The (s, t)-plane in Figs. 1(c)–(i) displays the computed approximations of  $U(s_j, t_k)$  as a function of j and k.

We turn to the computations of step (d). The Jacobian obtained in step (c) has an eigenvalue of magnitude smaller than  $h^2$ . We therefore consider the Jacobian singular and apply a bordering method with  $\Delta s = 0.1$  to determine a new point on the solution path. Steps (e)–(g) in Table 2 differ from step (d) only in that a bordering method, Algorithm 4.2, for nonsingular Jacobian matrices is used.

Finally, we determine two points on the solution path with the Euler–Newton method using step lengths  $\Delta \lambda = 0.1$  and 0.5, respectively. The computational work required is reported in Steps (h) and (i) in Table 2.

**Example 5.2.** The nonlinear boundary value problem

$$-\mathscr{L}U - \lambda \exp(U) = 0 \quad \text{in } \Omega,$$
  

$$U = 0 \quad \text{on } \partial\Omega,$$
(5.2)

where  $\Omega$  and  $\partial \Omega$  are the same as in (5.1) is known as the Bratu problem and is a common test problem for path following methods. We discretize (5.2) in the same manner as Eqs. (5.1) of Example 5.1.

This example illustrates the application of the algorithms of Section 4 to traverse a turning point. The computational steps are displayed in Table 3, which is analogous to Table 2. We let the initial approximate solution of the nonlinear system of equations (1.1) determined by (5.2) with  $\lambda = 5$  be



Fig. 1. Example 5.1: Solution path with bifurcation point. Figure (p) shows a solution path. Figures (c)–(i) display the solutions u associated with points on the graph in figure (p).

Table 3						
Example	5.2:	Solution	path	with	turning	point

Step	Comput.	λ	Step length	Smallest eig. val.	Solut. norm	Matrix acces.	Mat.–vec. prod.	CG iter.
<u> </u>	NW	5.0000*		$1 \cdot 10^{1}$	$6.14 \cdot 10^{0}$	307	507	98
b	EN	6.0000	1.00	$8 \cdot 10^0$	$8.63 \cdot 10^{0}$	399	639	144
c	EN	6.5000	0.50	$5 \cdot 10^0$	$1.08 \cdot 10^1$	481	761	180
d	EN	6.7500*	0.25	$1 \cdot 10^0$	$1.33 \cdot 10^1$	581	916	217
e	BR	6.7655	0.75	$2 \cdot 10^{-1}$	$1.41 \cdot 10^1$	689	1064	278
f	BR	6.7658	0.10	$4 \cdot 10^{-2}$	$1.42 \cdot 10^1$	786	1191	338
g	BR	6.7658*	0.05	$-4 \cdot 10^{-2}$	$1.42 \cdot 10^1$	889	1319	409
h	BR	6.7655	0.10	$-2 \cdot 10^{-1}$	$1.43 \cdot 10^1$	991	1451	474
i	BR	6.7509	0.75	$-1\cdot 10^0$	$1.51 \cdot 10^1$	1107	1602	548
j	EN	6.5009*	-0.25	$-7\cdot 10^0$	$1.81 \cdot 10^1$	1259	1839	605
k	EN	5.5009*	-1.00	$-2 \cdot 10^1$	$2.36 \cdot 10^1$	1398	2058	656
1	EN	4.5009	-1.00	$-3 \cdot 10^1$	$2.78\cdot 10^1$	1510	2225	707
m	EN	3.5009*	-1.00	$-5 \cdot 10^1$	$3.19\cdot 10^1$	1622	2397	753
n	EN	2.5009	-1.00	$-8 \cdot 10^1$	$3.61 \cdot 10^1$	1742	2577	807
0	EN	1.5009	-1.00	$-1 \cdot 10^{2}$	$4.07\cdot 10^1$	1927	2862	884



Fig. 2. Example 5.2: Solution path with turning point.

a random vector, and use Newton's method (2.4)-(2.5) to compute an approximate solution u of desired accuracy. The linear systems of Eqs. (2.5) in Newton's method are solved by Algorithm 4.1, which also gives eigenpairs of the Jacobian associated with the two smallest eigenvalues. The computational effort required is reported in Step (a) of Table 3. Steps (b)-(o) report path following by the Euler-Newton method and by bordering. All eigenvalues of all Jacobian matrices generated are of magnitude larger than  $\varepsilon = h^2$ . The Jacobian matrices are therefore considered nonsingular. Similarly as in Example 5.1, the computational effort reported is cumulative. For instance, the matrix accesses reported in Step (o) are for all computations in Steps (a)-(o). The matrix accesses and



Fig. 3. Example 5.2: Solutions u associated with points on the graph of Fig. 2.

matrix-vector products required for the CG iterations are included in the counts of matrix accesses and matrix-vector products, respectively.

Fig. 2 shows the solution path computed. Solutions u associated with points marked on the solution path, corresponding to starred  $\lambda$ -values in Table 3, are plotted in Fig. 3.

The examples illustrate that the algorithms of the present paper can be used for path following in the presence of bifurcation and turning points. We remark that the number of matrix accesses required often decreases when the block-size r is increased, but the number of matrix–vector products typically increases with the block-size. Numerical examples that illustrate this for Algorithm 4.1 are reported in [5]. The selection of block-size should depend both on the problem at hand and on the architecture of the computer being used.

The step sizes  $\Delta \lambda$ ,  $\Delta s$  and  $\varepsilon_b$  have been selected in a fairly arbitrary manner in the experiments. Implementation of a step-size control is likely to reduce the computational effort. We are presently investigating this issue.

# 6. Conclusion and future work

Experience from a large number of problems indicates that the algorithms presented in this paper are versatile tools for path following of large problems with symmetric Jacobians in an interactive computing environment. We are presently developing algorithms that are applicable to large-scale path following problems with nonsymmetric Jacobian matrices.

# References

- E.L. Allgower, C.-S. Chien, K. Georg, C.-F. Wang, Conjugate gradient methods for continuation problems, J. Comput. Appl. Math. 38 (1991) 1–16.
- [2] J. Baglama, D. Calvetti, L. Reichel, Iterative methods for computing a few eigenvalues of a large symmetric matrix, BIT 36 (1996) 400-421.
- [3] J. Baglama, D. Calvetti, L. Reichel, Fast Leja points, Elec. Trans. Numer. Anal. 7 (1998) 124-140.
- [4] J. Baglama, D. Calvetti, L. Reichel, A. Ruttan, Computation of a few small eigenvalues of a large matrix with application to liquid crystal modeling, J. Comput. Phys. 146 (1998) 203–226.
- [5] D. Calvetti, L. Reichel, A block Lanczos method for large continuation problems, Numer. Algorithms 21 (1999) 109–118.
- [6] D. Calvetti, L. Reichel, D.C. Sorensen, An implicitly restarted Lanczos method for large symmetric eigenvalue problems, Elec. Trans. Numer. Anal. 2 (1994) 1–21.
- [7] D. Calvetti, L. Reichel, Q. Zhang, Conjugate gradient algorithms for symmetric inconsistent linear systems, in: J.D. Brown, M.T. Chu, D.C. Ellison, R.J. Plemmons (Eds.), Proceedings of the Cornelius Lanczos International Centenary Conference, SIAM, Philadelphia, 1994, pp. 267–272.
- [8] C.-S. Chien, N.-H. Lu, Z.-L. Weng, Conjugate gradient methods for continuation problems II, J. Comput. Appl. Math. 62 (1995) 197–216.
- [9] K.A. Cliffe, T.J. Garratt, A. Spence, Eigenvalues of the discretized Navier–Stokes equation with application to the detection of Hopf bifurcations, Adv. Comput. Math. 1 (1993) 337–356.
- [10] D.W. Decker, H.B. Keller, Multiple limit point bifurcation, J. Math. Anal. Appl. 75 (1980) 417-430.
- [11] T. Ericsson, A. Ruhe, The spectral transformation Lanczos method for the solution of large sparse symmetric generalized eigenvalue problems, Math. Comput. 35 (1980) 1251–1268.
- [12] W.R. Ferng, C.T. Kelley, Mesh independence of matrix-free methods for path following, Report, Department of Mathematics, North Carolina State University, Raleigh, 1999.

- [13] K. Georg, On tracing an implicitly defined curve by quasi-Newton steps and calculating bifurcation by local perturbation, SIAM J. Sci. Statist. Comput. 2 (1981) 35–50.
- [14] J. Huitfeldt, Nonlinear eigenvalue problems prediction of bifurcation points and branch switching, Report, Department of Computer Science, Chalmers University of Technology, Göteborg, 1991.
- [15] J. Huitfeldt, A. Ruhe, A new algorithm for numerical path following applied to an example from hydrodynamics, SIAM J. Sci. Statist. Comput. 11 (1990) 1181–1192.
- [16] H.B. Keller, Lectures on Numerical Methods in Bifurcation Problems, Springer, Berlin, 1987.
- [17] D.C. Sorensen, Implicit application of polynomial filters in a *k*-step Arnoldi method, SIAM J. Matrix Anal. Appl. 13 (1992) 357–385.
- [18] Y.F. Zhou, A. Ruhe, Numerical path following and eigenvalue criteria for branch switching, in: J. Cullum, R.A. Willoughby (Eds.), Large Scale Eigenvalue Problems, Elsevier, Amsterdam, 1986, pp. 121–142.



Journal of Computational and Applied Mathematics 123 (2000) 241-260

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.nl/locate/cam

# The matrix and polynomial approaches to Lanczos-type algorithms

C. Brezinski<sup>a,\*</sup>, M. Redivo-Zaglia<sup>b</sup>, H. Sadok<sup>c</sup>

<sup>a</sup>Laboratoire d'Analyse Numérique et d'Optimisation, Université des Sciences et Technologies de Lille, France <sup>b</sup>Dipartimento di Matematica, Università degli Studi della Calabria, Arcavacata di Rende, 87036-Rende (CS), Italy <sup>c</sup>Laboratoire de Mathematiques Appliquées, Université du Littoral, Calais, France

Received 16 September 1999; received in revised form 7 December 1999

#### Abstract

Lanczos method for solving a system of linear equations can be derived by using formal orthogonal polynomials. It can be implemented by several recurrence relationships, thus leading to several algorithms. In this paper, the Lanczos/Orthodir algorithm will be derived in two different ways. The first one is based on a matrix approach and on the recursive computation of two successive regular matrices. We will show that it can be directly obtained from the orthogonality conditions and the fact that Lanczos method is a Krylov subspace method. The second approach is based on formal orthogonal polynomials. The case of breakdowns will be treated similarly. © 2000 Elsevier Science B.V. All rights reserved.

# 1. Introduction

In 1950, Lanczos [27] proposed a method for transforming a matrix into a similar tridiagonal one. Since, by the theorem of Cayley–Hamilton, the computation of the characteristic polynomial of a matrix and the solution of a system of linear equations are equivalent problems, Lanczos [28], in 1952, used his method for that purpose.

Owing to its numerous advantages, Lanczos method was the subject of very many investigations and several Lanczos-type algorithms for its implementation were obtained. Among them, the famous conjugate gradient algorithm of Hestenes and Stiefel [25] when the matrix is Hermitian and the bi-conjugate gradient algorithm of Fletcher [18] in the general case must be mentioned.

In these algorithms, the coefficients of the recurrence relationships are expressed as ratios of scalar products. When a scalar product in a denominator vanishes, then a breakdown occurs in the algorithm.

\* Corresponding author.

*E-mail addresses:* claude.brezinski@univ-lille1.fr (C. Brezinski), m.redivo-zaglia@unical.it (M. Redivo-Zaglia), sadok@lma.univ-littoral.fr (H. Sadok).

The treatment of breakdowns and near-breakdowns (division by a scalar product close to zero) in these algorithms has been an outstanding problem for many years. The first attempts for solving it make use of linear algebra techniques, a quite natural approach for a question related to that domain. However, these attempts were not completely satisfactory. But, as mentioned by Lanczos himself [27], his method is also related to formal orthogonal polynomials (FOP). In fact, all the recursive (old and new) algorithms for the implementation of Lanczos method can be derived from the theory of FOP [12]. Such polynomials also form the basis for the recursive computation of Padé approximants [32,3,15] since the denominator of the Padé approximant [k - 1/k] is a FOP after reversing the numbering of its coefficients. Thus, the problem of breakdown in the recursive computation of formal orthogonal polynomials was quite familiar to people working on Padé approximants. In the recursive computation of FOP, a breakdown can be due to the nonexistence of some orthogonal polynomial (*true* breakdown), or to the impossibility of using the recurrence under consideration (*ghost* breakdown). In such cases, it is possible to jump over the nonexisting polynomials, or over those which cannot be computed by the relation used, in order to obtain breakdown-free algorithms. The same idea can be applied to Lanczos-type algorithms; see, for example, [5,9].

In this paper, we will first use linear algebra techniques to derive the Lanczos/Orthodir algorithm [33,26]. Then, some algorithms for treating breakdowns will be explained in terms of linear algebra. However, the conclusion of this work is that, although such algorithms can be obtained by purely linear algebra techniques, the approach based on FOP is simpler and more powerfull. This is, in particular, true for transpose-free Lanczos-type methods such as the CGS and the BiCGSTAB. Let us mention that a linear algebra approach can also be used to derive the other algorithms for the implementation of Lanczos' method.

In the sequel, capital letters indicate matrices, bold lowercase letters indicate vectors, and Greek letters refer to scalars.

# 2. Preliminary results

We consider the system of linear equations

$$A\mathbf{x} = \mathbf{b},$$

where A is a real, nonsingular,  $n \times n$  matrix and  $\boldsymbol{b} \in \mathbb{R}^{n}$ .

Lanczos method [28] for solving this system consists of the following steps:

• choose two arbitrary nonzero vectors  $x_0$  and  $\tilde{r}_0$  in  $\mathbb{R}^n$ ,

• set 
$$\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$$
,

• determine  $x_k$  such that

$$\boldsymbol{x}_{k} - \boldsymbol{x}_{0} \in K_{k}(A, \boldsymbol{r}_{0}) = \operatorname{span}(\boldsymbol{r}_{0}, A\boldsymbol{r}_{0}, \dots, A^{k-1}\boldsymbol{r}_{0})$$
$$\boldsymbol{r}_{k} = \boldsymbol{b} - A\boldsymbol{x}_{k} \perp K_{k}(A^{\mathrm{T}}, \tilde{\boldsymbol{r}}_{0}) = \operatorname{span}(\tilde{\boldsymbol{r}}_{0}, A^{\mathrm{T}}\tilde{\boldsymbol{r}}_{0}, \dots, (A^{\mathrm{T}})^{k-1}\tilde{\boldsymbol{r}}_{0})$$

where  $A^{\mathrm{T}}$  is the transpose of A. Let us set

$$\boldsymbol{x}_k - \boldsymbol{x}_0 = -a_1^{(k)}\boldsymbol{r}_0 - \cdots - a_k^{(k)}A^{k-1}\boldsymbol{r}_0.$$

(1)

Multiplying both sides by A, adding and subtracting b, we obtain

$$\mathbf{r}_{k} = \mathbf{r}_{0} + a_{1}^{(k)} A \mathbf{r}_{0} + \dots + a_{k}^{(k)} A^{k} \mathbf{r}_{0}.$$
(2)

The orthogonality conditions can be written as

$$((A^{\mathrm{T}})^{i}\tilde{\boldsymbol{r}}_{0},\boldsymbol{r}_{k})=0$$
 for  $i=0,\ldots,k-1$ 

which is a system of k linear equations in the k unknowns  $a_1^{(k)}, \ldots, a_k^{(k)}$ . This system is nonsingular if and only if the following Hankel determinant, denoted  $H_k^{(1)}$ , is different from zero,

$$H_{k}^{(1)} = \begin{vmatrix} (\tilde{r}_{0}, Ar_{0}) & (\tilde{r}_{0}, A^{2}r_{0}) & \cdots & (\tilde{r}_{0}, A^{k}r_{0}) \\ (\tilde{r}_{0}, A^{2}r_{0}) & (\tilde{r}_{0}, A^{3}r_{0}) & \cdots & (\tilde{r}_{0}, A^{k+1}r_{0}) \\ \vdots & \vdots & \vdots \\ (\tilde{r}_{0}, A^{k}r_{0}) & (\tilde{r}_{0}, A^{k+1}r_{0}) & \cdots & (\tilde{r}_{0}, A^{2k-1}r_{0}) \end{vmatrix}$$

We set  $\mathbf{w}_i = A^i \mathbf{r}_0$  and  $\tilde{\mathbf{v}}_i = (A^T)^i \tilde{\mathbf{r}}_0$  for i = 0, ..., k. Let  $\tilde{V}_k = [\tilde{\mathbf{v}}_0, ..., \tilde{\mathbf{v}}_{k-1}]$  and  $W_k = [\mathbf{w}_1, ..., \mathbf{w}_k]$ . Then  $H_k^{(1)} = \det(\tilde{V}_k^T W_k)$ .

Under the assumption that  $\forall k, H_k^{(1)} \neq 0, x_k$  is well defined and we have

$$\boldsymbol{r}_{k} = \boldsymbol{r}_{0} - W_{k} (\tilde{\boldsymbol{V}}_{k}^{\mathrm{T}} W_{k})^{-1} \tilde{\boldsymbol{V}}_{k}^{\mathrm{T}} \boldsymbol{r}_{0} = \boldsymbol{b} - A \boldsymbol{x}_{k}.$$

If we denote by  $W_k^{\text{L}}$  the left inverse of the matrix  $W_k$  defined by  $W_k^{\text{L}} = (\tilde{V}_k^{\text{T}} W_k)^{-1} \tilde{V}_k^{\text{T}}$ , we also have  $\mathbf{r}_k = \mathbf{r}_0 - W_k W_k^{\text{L}} \mathbf{r}_0$ . (3)

We remind that the minimal polynomial of a matrix A for a vector  $\boldsymbol{u}$  is the polynomial p of smallest degree such that  $p(A)\boldsymbol{u} = 0$ .

We will now prove that a finite termination property holds and characterize the last iteration.

**Theorem 1.** Let *p* be the minimal polynomial of the matrix *A* for the vector  $\mathbf{r}_0$  and let *m* be its degree. If  $H_m^{(1)} \neq 0$ , then  $\mathbf{r}_m = p(A)\mathbf{r}_0 = 0$ .

**Proof.** Let us write the minimal polynomial p for  $r_0$  as  $p(\xi) = \sum_{i=0}^m d_i \xi^i$  with  $d_0 = p(0) = 1$  and

$$m = \min \left\{ k \left| \sum_{i=0}^{k} d_i A^i \mathbf{r}_0 = 0; d_0 = 1 \right\} \right\}$$

The equality  $\sum_{i=0}^{m} d_i A^i \mathbf{r}_0 = 0$  is, in matrix form,

$$W_m \boldsymbol{d} = -\boldsymbol{r}_0, \tag{4}$$

where  $d = (d_1, ..., d_m)^T$ . It is important to remark that system (4) has a unique solution and that the rank of  $W_m$  is *m*. Then

$$d = -W_m^{\rm L} r_0$$

and (4) becomes

$$\mathbf{r}_m = \mathbf{r}_0 - W_m W_m^{\mathsf{L}} \mathbf{r}_0 = p(A) \mathbf{r}_0 = 0. \qquad \Box$$

**Remark 2.** If  $H_m^{(1)} = 0$ , we have a so-called incurable breakdown. In this case, Lanczos method is not defined.

# 3. Matrix approach

In this section, we will first derive some Lanczos-type algorithms by a linear algebra approach in the regular case, that is when no breakdown occurs. Then, we will deal with a look-ahead strategy for treating breakdowns in the nonregular case and show how to obtain the MRZ algorithm [8] and the MRZ-Stab [10,11].

# 3.1. Regular case

First of all, we show how to compute the matrices  $W_{k+1}$  and  $W_{k+1}W_{k+1}^{L}$  from  $W_k$  and  $W_kW_k^{L}$ . We set

$$P_k = I - W_k W_k^{\mathsf{L}} \quad \text{with } P_0 = I. \tag{5}$$

We have the following result.

**Theorem 3.** If 
$$H_k^{(1)} \neq 0$$
 and  $H_{k+1}^{(1)} \neq 0$ , then, for  $k \ge 1$ ,  

$$P_{k+1} = P_k - u_k u_k^{\mathrm{L}} P_k,$$
(6)

where

r

$$\boldsymbol{u}_{k} = (I - W_{k}W_{k}^{L})\boldsymbol{w}_{k+1}$$

$$and \ \boldsymbol{u}_{k}^{L} = \tilde{\boldsymbol{v}}_{k}^{T}/(\tilde{\boldsymbol{v}}_{k}, \boldsymbol{u}_{k}).$$

$$(7)$$

**Proof.** The bordering method [17, pp. 105ff.] allows to compute the inverse of a matrix bordered by several new rows and columns from the initial matrix and, then, solving a bordered system of linear equations accordingly. It follows from the bordering method

$$W_{k+1}^{\mathrm{L}} = \left( egin{array}{c} W_k^{\mathrm{L}} - rac{1}{
ho_k} W_k^{\mathrm{L}} oldsymbol{w}_{k+1} oldsymbol{ ilde{v}}_k^{\mathrm{T}} (I - W_k W_k^{\mathrm{L}}) \ & rac{1}{
ho_k} oldsymbol{ ilde{v}}_k^{\mathrm{T}} (I - W_k W_k^{\mathrm{L}}) \end{array} 
ight),$$

where  $\rho_k = \tilde{\boldsymbol{v}}_k^{\mathrm{T}} (I - W_k W_k^{\mathrm{L}}) \boldsymbol{w}_{k+1}$ . Moreover

$$W_{k+1}W_{k+1}^{\mathrm{L}} = W_k W_k^{\mathrm{L}} + \boldsymbol{u}_k \boldsymbol{u}_k^{\mathrm{L}} (I - W_k W_k^{\mathrm{L}})$$

which proves the result.  $\Box$ 

**Remark 4.** If det $(\tilde{V}_k^T W_k) \neq 0$  then, using the Schur complement [14], we have  $\rho_k \neq 0$  if and only if det $(\tilde{V}_{k+1}^T W_{k+1}) \neq 0$  since

$$\rho_k = \frac{\det(\tilde{\boldsymbol{V}}_{k+1}^{\mathrm{T}} \boldsymbol{W}_{k+1})}{\det(\tilde{\boldsymbol{V}}_{k}^{\mathrm{T}} \boldsymbol{W}_{k})}$$

and  $\rho_0 = (\tilde{r}_0, w_1)$ .

**Remark 5.** By using the extensions of the Schur complement and the Schur determinantal formula given in [4], the vector  $\mathbf{r}_k$  can be expressed as a ratio of determinants

$$\mathbf{r}_{k} = \begin{vmatrix} \mathbf{w}_{0} & \mathbf{w}_{1} & \cdots & \mathbf{w}_{k} \\ (\tilde{\mathbf{v}}_{0}, \mathbf{w}_{0}) & (\tilde{\mathbf{v}}_{0}, \mathbf{w}_{1}) & \cdots & (\tilde{\mathbf{v}}_{0}, \mathbf{w}_{k}) \\ \vdots & \vdots & \vdots \\ (\tilde{\mathbf{v}}_{k-1}, \mathbf{w}_{0}) & (\tilde{\mathbf{v}}_{k-1}, \mathbf{w}_{1}) & \cdots & (\tilde{\mathbf{v}}_{k-1}, \mathbf{w}_{k}) \end{vmatrix} / \det(\tilde{\boldsymbol{V}}_{k}^{\mathrm{T}} \boldsymbol{W}_{k}),$$

where the determinant in the numerator denotes the vector obtained by developing it with respect to its first row by the classical rules (which means considering  $w_0, \ldots, w_k$  as numbers and, then, developing the determinant).

We see that  $\mathbf{r}_k = P_k \mathbf{r}_0$  and  $P_k^2 = P_k$ . Hence  $P_k$  is a projector. Now, from Theorem 3, we get

$$\boldsymbol{u}_{k} = P_{k}\boldsymbol{w}_{k+1} = \begin{vmatrix} \boldsymbol{w}_{k+1} & \boldsymbol{w}_{1} & \dots & \boldsymbol{w}_{k} \\ (\tilde{\boldsymbol{v}}_{0}, \boldsymbol{w}_{k+1}) & (\tilde{\boldsymbol{v}}_{0}, \boldsymbol{w}_{1}) & \dots & (\tilde{\boldsymbol{v}}_{0}, \boldsymbol{w}_{k}) \\ \vdots & \vdots & \vdots \\ (\tilde{\boldsymbol{v}}_{k-1}, \boldsymbol{w}_{k+1}) & (\tilde{\boldsymbol{v}}_{k-1}, \boldsymbol{w}_{1}) & \dots & (\tilde{\boldsymbol{v}}_{k-1}, \boldsymbol{w}_{k}) \end{vmatrix} / \det(\tilde{\boldsymbol{V}}_{k}^{\mathrm{T}} \boldsymbol{W}_{k}).$$

$$(8)$$

Multiplying  $r_0$  by the matrices on both sides of (6), we obtain

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \beta_{k+1} \mathbf{u}_k \tag{9}$$

with

$$\beta_{k+1} = \boldsymbol{u}_k^{\mathrm{L}} P_k \boldsymbol{r}_0 = \frac{(\tilde{\boldsymbol{v}}_k, \boldsymbol{r}_k)}{(\tilde{\boldsymbol{v}}_k, \boldsymbol{u}_k)}.$$
(10)

Let us now show how to compute the vector  $u_k$ . First, we need some properties of  $u_k$ .

Let  $U_k = [\mathbf{u}_0, \dots, \mathbf{u}_{k-1}]$ , with  $\mathbf{u}_0 = \mathbf{w}_1$ . As above  $U_k^{\mathrm{L}}$  will denote the left inverse matrix of the matrix  $U_k$  defined by  $U_k^{\mathrm{L}} = (\tilde{V}_k^{\mathrm{T}} U_k)^{-1} \tilde{V}_k^{\mathrm{T}}$ .

**Theorem 6.** If  $H_i^{(1)} \neq 0$  for j = 1, ..., k, then

- 1.  $(\tilde{v}_j, u_k) = 0$  for j = 0, ..., k 1,
- 2. span $(u_0, \ldots, u_{k-1}) = K_k(A, Ar_0),$
- 3. the vector  $\mathbf{u}_k$  can be written as

$$\boldsymbol{u}_k = A\boldsymbol{u}_{k-1} - U_k U_k^{\mathsf{L}} A\boldsymbol{u}_{k-1}, \quad \forall k \geq 1 \text{ and } \boldsymbol{u}_0 = A\boldsymbol{r}_0,$$

4.  $u_k = 0$  if and only if  $w_{k+1} \in K_k(A, Ar_0)$ .

**Proof.** 1. Multiplying (7) by the matrix  $\tilde{V}_k^{\mathrm{T}}$  and using the definition of  $W_k^{\mathrm{L}}$ , we obtain

$$\tilde{\boldsymbol{V}}_{k}^{\mathrm{T}}\boldsymbol{u}_{k} = \tilde{\boldsymbol{V}}_{k}^{\mathrm{T}}\boldsymbol{w}_{k+1} - (\tilde{\boldsymbol{V}}_{k}^{\mathrm{T}}\boldsymbol{W}_{k})(\tilde{\boldsymbol{V}}_{k}^{\mathrm{T}}\boldsymbol{W}_{k})^{-1}\tilde{\boldsymbol{V}}_{k}^{\mathrm{T}}\boldsymbol{w}_{k+1} = 0.$$

Consequently  $(\tilde{v}_j, u_k) = 0$  for  $j = 0, \dots, k - 1$ .

2. From the determinant in (8), we deduce that  $u_{k-1} \in \text{span}(w_1, \ldots, w_k)$ , hence  $\text{span}(u_0, \ldots, u_{k-1}) \subseteq \text{span}(w_1, \ldots, w_k)$ .

Now, assuming that  $\sum_{i=0}^{k-1} \alpha_i \boldsymbol{u}_i = 0$ , it follows that

$$\sum_{i=0}^{k-1} \alpha_i(\tilde{\boldsymbol{v}}_j, \boldsymbol{u}_i) = 0 \quad \text{for } j = 0, \dots, k-1.$$

We deduce, by using the first statement of this theorem, that the matrix of this system is lower triangular. Moreover, its determinant is equal to  $\prod_{i=0}^{k-1} (\tilde{v}_i, u_i)$ , which is nonzero since  $(\tilde{v}_i, u_i) = \rho_i$ , with

$$\rho_i = \frac{\det(\tilde{\boldsymbol{V}}_{i+1}^{\top} W_{i+1})}{\det(\tilde{\boldsymbol{V}}_i^{\top} W_i)} \neq 0 \quad \text{for } i = 1, \dots, k-1 \text{ and } \rho_0 = (\tilde{\boldsymbol{r}}_0, \boldsymbol{w}_1) \neq 0.$$

Thus span $(u_0, \ldots, u_{k-1})$  has dimension k and so it can only be identical to span $(w_1, \ldots, w_k)$ .

3. Multiplying the relation  $u_{k-1} = w_k - W_{k-1}W_{k-1}^L w_k$  by A, and since  $w_{k+1} = Aw_k$ , we obtain

$$A\boldsymbol{u}_{k-1} = \boldsymbol{w}_{k+1} - AW_{k-1}W_{k-1}^{\mathsf{L}}\boldsymbol{w}_{k}.$$

Then  $w_{k+1} = Au_{k-1} + AW_{k-1}d'_k$  with  $d'_k = W^{L}_{k-1}w_k$ . Since  $AW_{k-1} = [w_2, \ldots, w_k]$ , we have  $w_{k+1} \in \text{span}(w_2, \ldots, w_k, Au_{k-1})$ . Moreover, we can write

$$\boldsymbol{w}_{k+1} = A\boldsymbol{u}_{k-1} + W_k \boldsymbol{d}_k$$

with  $d_k = (0, d_k^{T})^{T}$ . Inserting the preceding relation in (7), we get

$$\boldsymbol{u}_{k} = A\boldsymbol{u}_{k-1} + W_{k}\boldsymbol{d}_{k} - W_{k}W_{k}^{L}(A\boldsymbol{u}_{k-1} + W_{k}\boldsymbol{d}_{k})$$
$$= A\boldsymbol{u}_{k-1} - W_{k}W_{k}^{L}A\boldsymbol{u}_{k-1}.$$

On the other hand, by induction, it is easy to show that we have the factorization

$$W_k = U_k R_k,$$

where  $R_k$  is a unit upper triangular  $k \times k$  matrix. Hence  $W_k W_k^{\rm L} = U_k U_k^{\rm L}$ .

4. If  $w_{k+1} \in \text{span}(w_1, \dots, w_k)$ , then we can write  $w_{k+1} = W_k d$ . Inserting this relation in (7), we obtain

$$\boldsymbol{u}_k = W_k \boldsymbol{d} - W_k W_k^{\mathrm{L}} W_k \boldsymbol{d} = 0.$$

Conversely if  $u_k = 0$ , then  $w_{k+1} - W_k W_k^L w_{k+1} = 0$ . Consequently  $w_{k+1} \in \text{span}(w_1, \dots, w_k)$ , which ends the proof.  $\Box$ 

We turn now to the computation of the vector  $\boldsymbol{u}_k$ . From the preceding Theorem, we deduce that  $\boldsymbol{u}_{k+1} = A\boldsymbol{u}_k - U_{k+1}U_{k+1}^{L}A\boldsymbol{u}_k = A\boldsymbol{u}_k - U_{k+1}\boldsymbol{d}_k$ , where  $\boldsymbol{d}_k$  is the solution of the linear system

$$(\tilde{\boldsymbol{V}}_{k+1}^{\mathrm{T}}\boldsymbol{U}_{k+1})\boldsymbol{d}_{k} = \tilde{\boldsymbol{V}}_{k+1}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{u}_{k}.$$
(11)

But the right-hand side of this system is  $(\tilde{\boldsymbol{v}}_0^T A \boldsymbol{u}_k, \dots, \tilde{\boldsymbol{v}}_k^T A \boldsymbol{u}_k)$ . Since  $\tilde{\boldsymbol{v}}_j^T A \boldsymbol{u}_k = \tilde{\boldsymbol{v}}_{j+1}^T \boldsymbol{u}_k$  and, by using the first statement of Theorem 6, we obtain

$$((\tilde{\boldsymbol{v}}_0, A\boldsymbol{u}_k), \dots, (\tilde{\boldsymbol{v}}_k, A\boldsymbol{u}_k)) = ((\tilde{\boldsymbol{v}}_1, \boldsymbol{u}_k), \dots, (\tilde{\boldsymbol{v}}_{k-1}, \boldsymbol{u}_k), (\tilde{\boldsymbol{v}}_k, \boldsymbol{u}_k), (\tilde{\boldsymbol{v}}_{k+1}, \boldsymbol{u}_k))$$
$$= (0, \dots, 0, (\tilde{\boldsymbol{v}}_k, \boldsymbol{u}_k), (\tilde{\boldsymbol{v}}_{k+1}, \boldsymbol{u}_k)).$$

246

Since the matrix  $\tilde{V}_{k+1}^{T}U_{k+1}$  is lower triangular, we deduce that  $\boldsymbol{d}_{k} = (0, \dots, 0, \gamma_{k+1}, \alpha_{k+1})^{T}$ , where

$$\begin{pmatrix} (\tilde{\boldsymbol{v}}_{k-1},\boldsymbol{u}_{k-1}) & 0\\ (\tilde{\boldsymbol{v}}_{k},\boldsymbol{u}_{k-1}) & (\tilde{\boldsymbol{v}}_{k},\boldsymbol{u}_{k}) \end{pmatrix} \begin{pmatrix} \gamma_{k+1}\\ \alpha_{k+1} \end{pmatrix} = \begin{pmatrix} (\tilde{\boldsymbol{v}}_{k-1},A\boldsymbol{u}_{k})\\ (\tilde{\boldsymbol{v}}_{k},A\boldsymbol{u}_{k}) \end{pmatrix}.$$

Using these relations, we can compute recursively the vector  $u_k$  by

$$u_{k+1} = Au_k - \alpha_{k+1}u_k - \gamma_{k+1}u_{k-1}, \tag{12}$$

where

$$\gamma_{k+1} = \frac{(\tilde{\boldsymbol{v}}_k, \boldsymbol{u}_k)}{(\tilde{\boldsymbol{v}}_{k-1}, \boldsymbol{u}_{k-1})} \quad \text{and} \quad \alpha_{k+1} = \frac{(\tilde{\boldsymbol{v}}_k, A\boldsymbol{u}_k - \gamma_{k+1}\boldsymbol{u}_{k-1})}{(\tilde{\boldsymbol{v}}_k, \boldsymbol{u}_k)}.$$
(13)

Now, since  $x_{k+1}$  cannot be computed directly from (9), we write  $u_k$  as  $u_k = Az_k$ , with  $z_0 = r_0$ . Since the matrix A is nonsingular, then from statement 1 of Theorem 6 we have  $\text{span}(z_0, \dots, z_{k-1}) = K_k(A, r_0)$  and from (8)

$$\boldsymbol{z}_{k} = \begin{vmatrix} \boldsymbol{w}_{k} & \boldsymbol{w}_{0} & \dots & \boldsymbol{w}_{k-1} \\ (\tilde{\boldsymbol{v}}_{0}, \boldsymbol{w}_{k+1}) & (\tilde{\boldsymbol{v}}_{0}, \boldsymbol{w}_{1}) & \dots & (\tilde{\boldsymbol{v}}_{0}, \boldsymbol{w}_{k}) \\ \vdots & \vdots & \vdots \\ (\tilde{\boldsymbol{v}}_{k-1}, \boldsymbol{w}_{k+1}) & (\tilde{\boldsymbol{v}}_{k-1}, \boldsymbol{w}_{1}) & \dots & (\tilde{\boldsymbol{v}}_{k-1}, \boldsymbol{w}_{k}) \end{vmatrix} / \det(\tilde{\boldsymbol{V}}_{k}^{\mathrm{T}} \boldsymbol{W}_{k}).$$

$$(14)$$

From (12) we immediately obtain that

 $\boldsymbol{z}_{k+1} = A\boldsymbol{z}_k - \boldsymbol{\alpha}_{k+1}\boldsymbol{z}_k - \boldsymbol{\gamma}_{k+1}\boldsymbol{z}_{k-1}$ 

and since  $\tilde{v}_{k+1} = A^T \tilde{v}_k$ , the expression for  $\alpha_{k+1}$  simplify. Gathering together all these formulas, we finally obtain the following algorithm [12].

Choose  $\boldsymbol{x}_0$  and  $\tilde{\boldsymbol{r}}_0$ Set  $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$ ,  $\tilde{\boldsymbol{v}}_0 = \tilde{\boldsymbol{r}}_0$ ,  $\boldsymbol{z}_0 = \boldsymbol{r}_0$ ,  $\boldsymbol{z}_{-1} = 0$ ,  $\gamma_1 = 0$ For  $k = 0, 1, 2, \dots$  until convergence do

$$\beta_{k+1} = \frac{(\boldsymbol{v}_{k}, \boldsymbol{r}_{k})}{(\tilde{\boldsymbol{v}}_{k}, A\boldsymbol{z}_{k})}$$

$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_{k} - \beta_{k+1}A\boldsymbol{z}_{k}$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_{k} + \beta_{k+1}\boldsymbol{z}_{k}$$

$$\tilde{\boldsymbol{v}}_{k+1} = A^{\mathrm{T}}\tilde{\boldsymbol{v}}_{k}$$
If  $k \neq 0$  then  $\gamma_{k+1} = \frac{(\tilde{\boldsymbol{v}}_{k}, A\boldsymbol{z}_{k})}{(\tilde{\boldsymbol{v}}_{k-1}, A\boldsymbol{z}_{k-1})}$ 

$$\alpha_{k+1} = \frac{(\tilde{\boldsymbol{v}}_{k+1}, A\boldsymbol{z}_{k} - \gamma_{k+1}\boldsymbol{z}_{k-1})}{(\tilde{\boldsymbol{v}}_{k}, A\boldsymbol{z}_{k})}$$

$$\boldsymbol{z}_{k+1} = A\boldsymbol{z}_{k} - \alpha_{k+1}\boldsymbol{z}_{k} - \gamma_{k+1}\boldsymbol{z}_{k-1}$$
end for

This algorithm is mathematically equivalent to the Lanczos/Orthodir algorithm given in [33,26] (also called BIODIR in [21]). Let us now show how to obtain the usual Lanczos/Orthodir algorithm.

From (10) we have

$$\beta_{k+1} = \frac{(\tilde{\boldsymbol{v}}_k, P_k \boldsymbol{r}_0)}{(\tilde{\boldsymbol{v}}_k, \boldsymbol{u}_k)} = \frac{(P_k^{\mathsf{T}} \tilde{\boldsymbol{v}}_k, P_k \boldsymbol{r}_0)}{(P_k^{\mathsf{T}} \tilde{\boldsymbol{v}}_k, \boldsymbol{u}_k)}.$$

Thus, if we set  $\tilde{\boldsymbol{u}}_k = P_k^{\mathrm{T}} \tilde{\boldsymbol{v}}_k$  and  $\tilde{\boldsymbol{u}}_0 = \tilde{\boldsymbol{r}}_0$ , we obtain

$$\beta_{k+1} = \frac{(\tilde{\boldsymbol{u}}_k, \boldsymbol{r}_k)}{(\tilde{\boldsymbol{u}}_k, \boldsymbol{u}_k)}.$$
(15)

We will now consider some fundamental properties of the projectors  $P_k$  and of the vectors  $\tilde{u}_k$ . Some of these properties are only needed in the proofs of statements 6 and 7 which are used in the sequel. We set  $\tilde{U}_k = [\tilde{u}_0, \dots, \tilde{u}_{k-1}]$ . We have the following results.

**Theorem 7.** If  $H_i^{(1)} \neq 0$  for j = 1, ..., k, then

- 1.  $P_i P_j = P_j P_i = P_j, \ \forall i \leq j \leq k.$
- $2. P_k W_k = 0.$
- 3.  $P_k = I \sum_{j=0}^{k-1} \boldsymbol{u}_j \boldsymbol{\tilde{u}}_j^{\mathrm{T}} / (\boldsymbol{\tilde{u}}_j, \boldsymbol{u}_j).$
- 4.  $(\tilde{u}_k, w_j) = (\tilde{u}_k, u_{j-1}) = 0$  for j = 1, ..., k.
- 5. span $(\tilde{\boldsymbol{u}}_0,\ldots,\tilde{\boldsymbol{u}}_{k-1}) = K_k(A^{\mathrm{T}},\tilde{\boldsymbol{r}}_0).$
- 6. The vector  $\tilde{\boldsymbol{u}}_k$  can be written as

$$\tilde{\boldsymbol{u}}_k = A^{\mathrm{T}} \tilde{\boldsymbol{u}}_{k-1} - \tilde{U}_k (U_k^{\mathrm{T}} \tilde{U}_k)^{-1} U_k^{\mathrm{T}} A^{\mathrm{T}} \tilde{\boldsymbol{u}}_{k-1}.$$

- 7. The matrix  $\tilde{U}_k^{\mathrm{T}} U_k$  is a diagonal matrix whose ith diagonal element is  $\rho_i$ .
- 8.  $(\tilde{u}_i, Au_k) = 0$  for i = 0, ..., k 2.
- 9.  $\tilde{\boldsymbol{u}}_{k} = 0$  if and only if  $\tilde{\boldsymbol{v}}_{k+1} \in K_{k}(A^{\mathrm{T}}, \tilde{\boldsymbol{r}}_{0})$ .

**Proof.** 1–3 are obvious.

4. By definition we have  $\tilde{\boldsymbol{u}}_k = P_k^T \tilde{\boldsymbol{v}}_k$  and, as  $P_{k-1}W_{k-1} = 0$ , it follows that  $P_{k-1}w_j = 0$  for  $j=1,\ldots,k-1$  and

$$(\tilde{\boldsymbol{u}}_k, \boldsymbol{w}_j) = (P_k^{\mathrm{T}} \tilde{\boldsymbol{v}}_k, \boldsymbol{w}_j) = (\tilde{\boldsymbol{v}}_k, P_k \boldsymbol{w}_j) = 0 \text{ for } j = 1, \dots, k.$$

Moreover, from statement 1, we have

$$(\tilde{\boldsymbol{v}}_k, P_k \boldsymbol{w}_i) = (\tilde{\boldsymbol{v}}_k, P_k P_{i-1} \boldsymbol{w}_i) = (P_k^{\mathrm{T}} \tilde{\boldsymbol{v}}_k, \boldsymbol{u}_{i-1}) = (\tilde{\boldsymbol{u}}_k, \boldsymbol{u}_{i-1}).$$

5. By transposing (5), we get  $P_k^{\rm T} = I - \tilde{V}_k (W_k^{\rm T} \tilde{V}_k)^{-1} W_k^{\rm T}$ . It follows that

$$\tilde{\boldsymbol{u}}_k = \tilde{\boldsymbol{v}}_k - \tilde{\boldsymbol{V}}_k (\boldsymbol{W}_k^{\mathrm{T}} \tilde{\boldsymbol{V}}_k)^{-1} \boldsymbol{W}_k^{\mathrm{T}} \tilde{\boldsymbol{v}}_k.$$

From the last relation, we deduce that  $\tilde{u}_{k-1} \in \operatorname{span}(\tilde{v}_0, \ldots, \tilde{v}_{k-1})$ . Hence  $\operatorname{span}(\tilde{u}_0, \ldots, \tilde{u}_{k-1}) \subseteq \operatorname{span}(\tilde{v}_0, \ldots, \tilde{v}_{k-1})$ .

Now, assume that  $\sum_{i=0}^{k-1} \alpha_i \tilde{\boldsymbol{u}}_i = 0$ , then

$$\sum_{i=0}^{k-1} \alpha_i(\tilde{\boldsymbol{u}}_i, \boldsymbol{w}_j) = 0 \quad \text{for } j = 1, \dots, k$$

But the matrix of this system is a lower triangular. Its determinant is the product, for i = 0 to k - 1, of  $(\tilde{u}_i, w_{i+1})$ . But  $(\tilde{u}_i, w_{i+1}) = (\tilde{v}_i, u_i) = \rho_i$ , which is different from zero as seen in the proof of Theorem

6, item 2. Thus, this matrix is nonsingular and span $(\tilde{u}_0, \ldots, \tilde{u}_{k-1})$  has dimension k and so it must be identical to span $(\tilde{v}_0, \ldots, \tilde{v}_{k-1})$ .

- 6. The proof is similar to that of item 3 of Theorem 6.
- 7. We have  $(\tilde{\boldsymbol{u}}_i, \boldsymbol{u}_j) = (P_i^{\mathrm{T}} \tilde{\boldsymbol{v}}_i, P_j \boldsymbol{w}_{j+1})$ . Consequently,

$$(\tilde{\boldsymbol{u}}_i, \boldsymbol{u}_j) = \begin{cases} (\tilde{\boldsymbol{v}}_i, P_j \boldsymbol{w}_{j+1}) = (\tilde{\boldsymbol{v}}_i, \boldsymbol{u}_j) & \text{if } i < j \leq k, \\ (\tilde{\boldsymbol{v}}_i, P_i \boldsymbol{w}_{j+1}) = (\tilde{\boldsymbol{u}}_i, \boldsymbol{w}_{j+1}) & \text{if } j < i \leq k. \end{cases}$$

Then, since  $(\tilde{u}_i, u_i) = \rho_i \neq 0$  and using the first part of Theorem 6 and part 4 of this theorem we get the result.

8. We already show that  $Au_k = u_{k+1} + \alpha_{k+1}u_k + \gamma_{k+1}u_{k-1}$ . Consequently, from the preceding statement, we get  $(\tilde{u}_i, Au_k) = (\tilde{u}_i, u_{k+1}) + \alpha_{k+1}(\tilde{u}_i, u_k) + \gamma_{k+1}(\tilde{u}_i, u_{k-1}) = 0$  for  $i = 0, \dots, k-2$ .

9. The proof is similar to that of the last part of Theorem 6.  $\Box$ 

The linear system (11) can be written as

$$(\tilde{\boldsymbol{U}}_{k+1}^{\mathrm{T}}\boldsymbol{U}_{k+1})\boldsymbol{d}_{k} = \tilde{\boldsymbol{U}}_{k+1}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{u}_{k}.$$
(16)

The matrix  $(\tilde{U}_{k+1}^{T}U_{k+1})$  is diagonal and from statement 8 of Theorem 7, we can see that only the last two components of the right-hand side of system (16) are nonzero. Consequently,  $\gamma_{k+1}$  and  $\alpha_{k+1}$  are given by

$$\gamma_{k+1} = \frac{(\tilde{\boldsymbol{u}}_k, \boldsymbol{u}_k)}{(\tilde{\boldsymbol{u}}_{k-1}, \boldsymbol{u}_{k-1})}, \quad \alpha_{k+1} = \frac{(A^{\mathrm{T}} \tilde{\boldsymbol{u}}_k, \boldsymbol{u}_k)}{(\tilde{\boldsymbol{u}}_k, \boldsymbol{u}_k)}.$$
(17)

The vectors  $\tilde{\boldsymbol{u}}_k$  can be recursively computed as the vectors  $\boldsymbol{u}_k$  but with  $A^T$  instead of A. These formulas define the algorithm known under the names Lanczos/Orthodir [33,26] and BIODIR [21]. It is as follows.

# Lanczos/Orthodir algorithm

Choose  $\boldsymbol{x}_0$  and  $\tilde{\boldsymbol{r}}_0$ Set  $\boldsymbol{r}_0 = \boldsymbol{b} - A\boldsymbol{x}_0$ ,  $\boldsymbol{z}_0 = \boldsymbol{r}_0$ ,  $\boldsymbol{z}_{-1} = 0$ ,  $\tilde{\boldsymbol{u}}_{-1} = 0$ ,  $\tilde{\boldsymbol{u}}_0 = \tilde{\boldsymbol{r}}_0$ ,  $\gamma_1 = 0$ For  $k = 0, 1, 2, \dots$  until convergence do

$$\beta_{k+1} = \frac{(\tilde{\boldsymbol{u}}_{k}, \boldsymbol{r}_{k})}{(\tilde{\boldsymbol{u}}_{k}, A\boldsymbol{z}_{k})}$$

$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_{k} - \beta_{k+1}A\boldsymbol{z}_{k}$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_{k} + \beta_{k+1}\boldsymbol{z}_{k}$$
If  $k \neq 0$  then  $\gamma_{k+1} = \frac{(\tilde{\boldsymbol{u}}_{k}, A\boldsymbol{z}_{k})}{(\tilde{\boldsymbol{u}}_{k-1}, A\boldsymbol{z}_{k-1})}$ 

$$\alpha_{k+1} = \frac{(A^{\mathrm{T}} \tilde{\boldsymbol{u}}_{k}, A\boldsymbol{z}_{k})}{(\tilde{\boldsymbol{u}}_{k}, A\boldsymbol{z}_{k})}$$

$$\boldsymbol{z}_{k+1} = A\boldsymbol{z}_{k} - \alpha_{k+1}\boldsymbol{z}_{k} - \gamma_{k+1}\boldsymbol{z}_{k-1}$$

$$\tilde{\boldsymbol{u}}_{k+1} = A^{\mathrm{T}} \tilde{\boldsymbol{u}}_{k} - \alpha_{k+1} \tilde{\boldsymbol{u}}_{k} - \gamma_{k+1} \tilde{\boldsymbol{u}}_{k-1}$$
for

end for

This algorithm is defined (that is  $x_k$  exists) only if  $H_k^{(1)} \neq 0$ ,  $\forall k$ . If this is not the case, it is possible to jump over the nonexisting iterates and to use only those which exist. We have now to show how to jump directly from  $x_k$  to  $x_{k+m}$  when  $H_k^{(1)} \neq 0$ ,  $H_{k+i}^{(1)} = 0$ , i = 1, ..., m-1 and  $H_{k+m}^{(1)} \neq 0$ .

# 3.2. Look-ahead strategy

Let us assume that some of the Hankel determinants  $H_k^{(1)}$  are equal to zero. Then, due to a division by zero, a breakdown occurs in the two preceding algorithms and the corresponding iterates  $x_k$  do not exist. It is possible to avoid such breakdowns by computing only the existing iterates. Two successive existing iterates will be denoted by  $x_{n_k}$  and  $x_{n_{k+1}}$ , where  $n_k$  is the dimension of the corresponding Krylov subspace,  $n_{k+1} = n_k + m_k$ , and where  $m_k$  denotes the length of the jump between the dimensions of the two Krylov subspaces.

# 3.2.1. Determination of the length of the jump

If the matrix  $\tilde{V}_{n_k}^{\mathrm{T}} W_{n_k}$  is nonsingular, that is  $H_{n_k}^{(1)} \neq 0$ , then  $u_{n_k}$  exists and we have

$$\boldsymbol{u}_{n_k} = \boldsymbol{w}_{n_k+1} - W_{n_k} W_{n_k}^{\mathrm{L}} \boldsymbol{w}_{n_k+1}, \quad \boldsymbol{u}_{n_0} = A \boldsymbol{r}_0 \text{ with } n_0 = 0.$$
(18)

Thus, from statement 1 of Theorem 6,  $\tilde{V}_{n_k}^{\mathrm{T}} \boldsymbol{u}_{n_k} = 0$ , which can be written, by setting  $\boldsymbol{u}_{n_k} = A\boldsymbol{z}_{n_k}$ , as

$$(\tilde{\mathbf{v}}_i, A \, \mathbf{z}_{n_k}) = 0 \quad \text{for } i = 0, \dots, n_k - 1.$$
<sup>(19)</sup>

We will now discuss how to determine the length  $m_k$  of the jump. We assume that  $H_{n_k+i}^{(1)} = 0$  for  $i=1,\ldots,m_k-1$  and that it is different from zero for i=0 and  $i=m_k$ . Applying Schur's determinantal formula to  $H_{n+1}^{(1)}$  (see Remark 4), we see that

$$H_{n_k+1}^{(1)}/H_{n_k}^{(1)} = (\tilde{v}_{n_k}, A z_{n_k}).$$

Hence  $H_{n_k+1}^{(1)} = 0$  if and only if  $(\tilde{v}_{n_k}, Az_{n_k}) = 0$ . More generally we have the following result.

# **Theorem 8.** If $H_{n_k+i}^{(1)} = 0$ for j = 1, ..., i - 1, then $H_{n_k+i}^{(1)} = 0$ if and only if $(\tilde{v}_{n_k}, A^i z_{n_k}) = 0$ .

Let us first remark that, if we assume that  $H_{n_j}^{(1)} \neq 0$  for  $j=0,\ldots,k$ , then only the vectors  $z_{n_0},\ldots,z_{n_k}$ are defined. To obtain a basis of  $K_{n_k+i}(A, \mathbf{r}_0)$ , some auxiliary vectors are needed. The natural choice is to take  $z_{n_{j,i}} = A^i z_{n_j}$ ,  $i \ge 0$ . These vectors are called the inner vectors.

Since 
$$H_{n_k+i}^{(1)} = \det(V_{n_k+i}W_{n_k+i})$$
, it is easy to verify that

- span(z<sub>n0</sub>, z<sub>n0,1</sub>,..., z<sub>n0,m0-1</sub>,..., z<sub>nk-1</sub>, z<sub>nk-1</sub>, z<sub>nk-1</sub>, 1, ..., z<sub>nk-1</sub>, m<sub>k-1</sub>-1, z<sub>nk</sub>, z<sub>nk,1</sub>, ..., z<sub>nk,i-1</sub>) = K<sub>nk+i</sub>(A, r<sub>0</sub>).
   det(Ṽ<sup>T</sup><sub>nk+i</sub>W<sub>nk+i</sub>) = 0 if and only if det(Ṽ<sup>T</sup><sub>nk+i</sub>AZ<sub>nk+i</sub>) = 0, where Z<sub>nk+i</sub> is the matrix whose columns are z<sub>n0</sub>, z<sub>n0,1</sub>, ..., z<sub>n0,m0-1</sub>, ..., z<sub>nk-1</sub>, z<sub>nk-1</sub>, 1, ..., z<sub>nk-1</sub>, m<sub>k-1-1</sub>, z<sub>nk</sub>, z<sub>nk,1</sub>, ..., z<sub>nk,i-1</sub>.
   |det(Ṽ<sup>T</sup><sub>nk+i</sub>AZ<sub>nk+i</sub>)| = |det(Ṽ<sup>T</sup><sub>nk</sub>AZ<sub>nk</sub>)| |(ṽ<sub>nk</sub>, A<sup>i</sup>z<sub>nk</sub>)|<sup>i</sup>.
- Hence  $m_k$  is such that

$$(\tilde{\boldsymbol{v}}_{n_k}, A^i \boldsymbol{z}_{n_k}) = 0 \quad \text{for } i = 1, \dots, m_k - 1 \text{ and } (\tilde{\boldsymbol{v}}_{n_k}, A^{m_k} \boldsymbol{z}_{n_k}) \neq 0.$$
(20)

# 3.2.2. Computation of the residual

We will now show how to deduce  $x_{n_{k+1}}$  and  $r_{n_{k+1}}$  from  $x_{n_k}$ ,  $r_{n_k}$  and  $z_{n_k}$ . We have  $r_{n_{k+1}} = P_{n_{k+1}} r_0$  and

$$P_{n_{k+1}} = I - W_{n_{k+1}} (\tilde{V}_{n_{k+1}}^{\mathrm{T}} W_{n_{k+1}})^{-1} \tilde{V}_{n_{k+1}}^{\mathrm{T}} = I - AZ_{n_{k+1}} (\tilde{V}_{n_{k+1}}^{\mathrm{T}} AZ_{n_{k+1}})^{-1} \tilde{V}_{n_{k+1}}^{\mathrm{T}}.$$

We have to consider a block version of (6). We write  $\tilde{V}_{n_{k+1}} = [\tilde{V}_{n_k}, \tilde{V}_{n_k,m_k}]$  and  $Z_{n_{k+1}} = [Z_{n_k}, Z_{n_k,m_k}]$ , which means that the matrix  $\tilde{V}_{n_k,m_k}$  is the matrix formed by the vectors to be added to  $\tilde{V}_{n_k}$  to obtain  $\tilde{V}_{n_{k+1}}$  and similarly for  $Z_{n_k,m_k}$ . We have

$$\tilde{\boldsymbol{\mathcal{V}}}_{n_{k+1}}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{Z}_{n_{k+1}} = \begin{pmatrix} \tilde{\boldsymbol{\mathcal{V}}}_{n_{k}}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{Z}_{n_{k}} & \tilde{\boldsymbol{\mathcal{V}}}_{n_{k}}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{Z}_{n_{k},m_{k}} \\ \tilde{\boldsymbol{\mathcal{V}}}_{n_{k},m_{k}}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{Z}_{n_{k}} & \tilde{\boldsymbol{\mathcal{V}}}_{n_{k},m_{k}}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{Z}_{n_{k},m_{k}} \end{pmatrix}$$

On the other hand,  $\tilde{V}_{n_k}^{\mathrm{T}} A Z_{n_k, m_k} = 0$ . Then

$$\tilde{V}_{n_{k+1}}^{\mathsf{T}} A Z_{n_{k+1}} = \begin{pmatrix} \tilde{V}_{n_{k}}^{\mathsf{T}} A Z_{n_{k}} & 0\\ \tilde{V}_{n_{k},m_{k}}^{\mathsf{T}} A Z_{n_{k}} & \tilde{V}_{n_{k},m_{k}}^{\mathsf{T}} A Z_{n_{k},m_{k}} \end{pmatrix}.$$
(21)

Inverting the matrix  $\tilde{V}_{n_{k+1}}^{T}AZ_{n_{k+1}}$ , we deduce

$$P_{n_{k+1}} = P_{n_k} - AZ_{n_k, m_k} (\tilde{V}_{n_k, m_k}^1 A Z_{n_k, m_k})^{-1} \tilde{V}_{n_k, m_k}^1 P_{n_k}$$

Consequently,

 $\mathbf{r}_{n_{k+1}} = \mathbf{r}_{n_k} - AZ_{n_k,m_k} \, \boldsymbol{\beta}_{n_{k+1}}$  and  $\mathbf{x}_{n_{k+1}} = \mathbf{x}_{n_k} + Z_{n_k,m_k} \, \boldsymbol{\beta}_{n_{k+1}}$ , where the vector  $\boldsymbol{\beta}_{n_{k+1}} \in \mathbb{R}^{m_k}$  is the solution of the linear system

$$(\tilde{\boldsymbol{V}}_{n_k,m_k}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{Z}_{n_k,m_k})\boldsymbol{\beta}_{n_{k+1}}=\tilde{\boldsymbol{V}}_{n_k,m_k}^{\mathrm{T}}\boldsymbol{r}_{n_k}$$

It is important to remark that the matrix of this system is an upper triangular Toeplitz matrix, since

$$\tilde{\boldsymbol{V}}_{n_k,m_k}^{\mathsf{T}} \boldsymbol{A} \boldsymbol{Z}_{n_k,m_k} = \begin{pmatrix} (\tilde{\boldsymbol{v}}_{n_k}, \boldsymbol{A}^{m_k} \boldsymbol{z}_{n_k}) \\ (\tilde{\boldsymbol{v}}_{n_k,m_k} \boldsymbol{A}^{m_k} \boldsymbol{z}_{n_k}) & (\tilde{\boldsymbol{v}}_{n_{k+1}}, \boldsymbol{A}^{m_k} \boldsymbol{z}_{n_k}) \\ \vdots & \vdots & \vdots & \vdots \\ (\tilde{\boldsymbol{v}}_{n_k}, \boldsymbol{A}^{m_k} \boldsymbol{z}_{n_k}) & (\tilde{\boldsymbol{v}}_{n_{k+1}}, \boldsymbol{A}^{m_k} \boldsymbol{z}_{n_k}) \cdots \cdots & (\tilde{\boldsymbol{v}}_{n_k+m_k-1}, \boldsymbol{A}^{m_k} \boldsymbol{z}_{n_k}) \end{pmatrix}$$

3.2.3. Computation of the vectors  $\mathbf{z}_{n_{k+1}}$ 

The two relations (18) and (19) determine  $z_{n_{k+1}}$ , which can also be defined as

$$m{z}_{n_{k+1}} - m{w}_{n_{k+1}} \in K_{n_{k+1}}(A, m{r}_0), \ ilde{m{\mathcal{V}}}_{n_{k+1}}^{\mathrm{T}} A m{z}_{n_{k+1}} = 0.$$

This leads to

$$m{z}_{n_{k+1}} - A^{m_k} m{z}_{n_k} \in K_{n_{k+1}}(A,m{r}_0), \ ilde{m{\mathcal{V}}}_{n_{k+1}}^{\mathrm{T}} A m{z}_{n_{k+1}} = 0.$$

It follows that

252

$$\boldsymbol{z}_{n_{k+1}} = A^{m_k} \boldsymbol{z}_{n_k} - Z_{n_{k+1}} (\tilde{\boldsymbol{V}}_{n_{k+1}}^{\mathrm{T}} A Z_{n_{k+1}})^{-1} \tilde{\boldsymbol{V}}_{n_{k+1}}^{\mathrm{T}} A^{m_k+1} \boldsymbol{z}_{n_k}$$

From (19) and (20) we deduce that  $(\tilde{v}_i, A^{m_k+1}z_{n_k}) = 0$  for  $i = 0, ..., n_k - 2$ . Consequently,

$$ilde{V}_{n_{k+1}}^{\mathrm{T}}A^{m_k+1}oldsymbol{z}_{n_k}=(0,\ldots,0,( ilde{oldsymbol{v}}_{n_k},A^{m_k}oldsymbol{z}_{n_k}),\ldots,( ilde{oldsymbol{v}}_{n_k+m_k},A^{m_k}oldsymbol{z}_{n_k}))^{\mathrm{T}}.$$

Let  $d_{n_{k+1}}$  be the solution of the linear system

$$(\tilde{\boldsymbol{V}}_{n_{k+1}}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{Z}_{n_{k+1}})\boldsymbol{d}_{n_{k+1}} = (0,\ldots,0,(\tilde{\boldsymbol{v}}_{n_k},\boldsymbol{A}^{m_k}\boldsymbol{z}_{n_k}),\ldots,(\tilde{\boldsymbol{v}}_{n_k+m_k},\boldsymbol{A}^{m_k}\boldsymbol{z}_{n_k}))^{\mathrm{T}}.$$

Then

$$d_{n_{k+1}} = (0, \ldots, 0, \gamma_{n_{k+1}}, 0, \ldots, 0, \alpha_{n_{k+1}})^{\mathrm{T}},$$

where  $\gamma_{n_{k+1}}$  is a scalar and  $\boldsymbol{\alpha}_{n_{k+1}} \in \mathbb{R}^{m_k}$  is a vector. Thus

$$\mathbf{z}_{n_{k+1}} = A^{m_k} \mathbf{z}_{n_k} - Z_{n_k, m_k} \mathbf{a}_{n_{k+1}} - \gamma_{n_{k+1}} \mathbf{z}_{n_{k-1}}$$

From (21) we have

$$\gamma_{n_{k+1}} = \frac{(\tilde{\boldsymbol{v}}_{n_k}, A^{m_k} \boldsymbol{z}_{n_k})}{(\tilde{\boldsymbol{v}}_{n_{k-1}}, A^{m_{k-1}} \boldsymbol{z}_{n_{k-1}})}$$

and

$$(\tilde{V}_{n_k,m_k}^{\mathrm{T}}AZ_{n_k,m_k})\boldsymbol{\alpha}_{n_{k+1}} = \tilde{V}_{n_k+1,m_k}^{\mathrm{T}}(A^{m_k}\boldsymbol{z}_{n_k} - \gamma_{n_{k+1}}\boldsymbol{z}_{n_{k-1}}).$$

Gathering together all these formulas, we finally obtain the following algorithm called method of recursive zoom (MRZ) [8]

## MRZ algorithm

Choose  $\mathbf{x}_0$  and  $\tilde{\mathbf{r}}_0$ Set  $n_0 = 0$ ,  $\mathbf{r}_{n_0} = \mathbf{b} - A\mathbf{x}_0$ ,  $\tilde{\mathbf{v}}_{n_0} = \tilde{\mathbf{r}}_0$ ,  $\mathbf{z}_{n_0} = \mathbf{r}_0$ ,  $\mathbf{z}_{n_{-1}} = 0$ For k = 0, 1, 2, ... until convergence do Find  $m_k$  such that  $(\tilde{\mathbf{v}}_{n_k}, A^i \mathbf{z}_{n_k}) = 0$  for  $i = 1, ..., m_k - 1$  and  $(\tilde{\mathbf{v}}_{n_k}, A^{m_k} \mathbf{z}_{n_k}) \neq 0$   $n_{k+1} = n_k + m_k$ Solve  $(\tilde{V}_{n_k, m_k}^T A Z_{n_k, m_k}) \mathbf{\beta}_{n_{k+1}} = \tilde{V}_{n_k, m_k}^T \mathbf{r}_{n_k}$   $\mathbf{r}_{n_{k+1}} = \mathbf{r}_{n_k} - A Z_{n_k, m_k} \mathbf{\beta}_{n_{k+1}}$ If  $k \neq 0$  then  $\gamma_{n_{k+1}} = \frac{(\tilde{\mathbf{v}}_{n_k}, A^{m_k} \mathbf{z}_{n_k})}{(\tilde{\mathbf{v}}_{n_{k-1}}, A^{m_{k-1}} \mathbf{z}_{n_{k-1}})}$  else  $\gamma_{n_{k+1}} = 0$ Solve  $(\tilde{V}_{n_k, m_k}^T A Z_{n_k, m_k}) \mathbf{\alpha}_{n_{k+1}} = \tilde{V}_{n_k+1, m_k}^T (A^{m_k} \mathbf{z}_{n_k} - \gamma_{n_{k+1}} \mathbf{z}_{n_{k-1}})$   $\mathbf{z}_{n_{k+1}} = A^{m_k} \mathbf{z}_{n_k} - Z_{n_k, m_k} \mathbf{\alpha}_{n_{k+1}} - \gamma_{n_{k+1}} \mathbf{z}_{n_{k-1}}$  and  $\tilde{\mathbf{v}}_{n_{k+1}} = (A^T)^{m_k} \tilde{\mathbf{v}}_{n_k}$ end for

Clearly, this algorithm is a generalization of the Lanczos/Orthodir algorithm. It cannot suffer from breakdown, except the incurable hard one which occurs when the matrix  $(\tilde{V}_m^T A Z_m)$  is singular, where
m is the degree of the minimal polynomial of A for  $r_0$ . The pseudo-code of this algorithm and the corresponding subroutine are given in [6,7].

This algorithm may be unstable due to the powers of  $A^{T}$ . This drawback will be avoided in the MRZ-Stab algorithm which generalizes the BIODIR algorithm and will now be presented. The derivation of the formulas will not be given in details.

Let us set  $\tilde{\boldsymbol{u}}_{n_i} = P_{n_i}^{\mathrm{T}} \tilde{\boldsymbol{v}}_{n_i}$  with  $\tilde{\boldsymbol{u}}_{n_0} = \tilde{\boldsymbol{r}}_0$ , and  $\tilde{\boldsymbol{u}}_{n_i,i} = A^{\mathrm{T}^i} \tilde{\boldsymbol{u}}_{n_i}$ ,  $i \ge 0$ . Then it is easy to show that

- 1. span $(\tilde{u}_{n_0}, \tilde{u}_{n_0,1}, \dots, \tilde{u}_{n_0,m_0-1}, \dots, \tilde{u}_{n_{k-1}}, \tilde{u}_{n_{k-1},1}, \dots, \tilde{u}_{n_{k-1}-1}, \tilde{u}_{n_k}, \tilde{u}_{n_k,1}, \dots, \tilde{u}_{n_k,i-1}) = K_{n_k+i}(A^{\mathrm{T}}, \tilde{r}_0).$
- 2. det $(\tilde{V}_{n_{k+i}}^{\mathrm{T}}W_{n_{k+i}}) = 0$  if and only if det $(\tilde{U}_{n_{k+i}}^{\mathrm{T}}AZ_{n_{k+i}}) = 0$ , where  $\tilde{U}_{n_{k+i}}$  is the matrix whose columns are  $\tilde{u}_{n_{0}}, \tilde{u}_{n_{0},1}, \dots, \tilde{u}_{n_{0},m_{0}-1}, \dots, \tilde{u}_{n_{k-1}}, \tilde{u}_{n_{k-1},1}, \dots, \tilde{u}_{n_{k-1}-1}, \tilde{u}_{n_{k}}, \tilde{u}_{n_{k},1}, \dots, \tilde{u}_{n_{k},i-1}$ . 3.  $P_{n_{k+1}} = P_{n_{k}} AZ_{n_{k},m_{k}}(\tilde{U}_{n_{k},m_{k}}^{\mathrm{T}}AZ_{n_{k},m_{k}})^{-1}\tilde{U}_{n_{k},m_{k}}^{\mathrm{T}}P_{n_{k}}$  where the matrices  $\tilde{U}_{n_{k},m_{k}}$  are defined similarly to
- the matrices  $\tilde{V}_{n_k,m_k}$  and  $Z_{n_k,m_k}$ .
- 4. The matrix  $(\tilde{U}_{n_k,m_k}^{\dagger}AZ_{n_k,m_k})$  is an upper triangular Toeplitz matrix

$$(\tilde{\boldsymbol{U}}_{n_k,m_k}^{\mathrm{T}}A\boldsymbol{Z}_{n_k,m_k}) = \begin{pmatrix} & (\tilde{\boldsymbol{u}}_{n_k},A^{m_k}\boldsymbol{z}_{n_k}) \\ & (\tilde{\boldsymbol{u}}_{n_k},A^{m_k}\boldsymbol{z}_{n_k}) & (\tilde{\boldsymbol{u}}_{n_{k-1}},A^{m_k}\boldsymbol{z}_{n_k}) \\ & \ddots & \ddots & \vdots \\ & \ddots & \ddots & \ddots & \vdots \\ & \ddots & \ddots & \ddots & \vdots \\ & (\tilde{\boldsymbol{u}}_{n_k},A^{m_k}\boldsymbol{z}_{n_k}) & (\tilde{\boldsymbol{u}}_{n_k,1},A^{m_k}\boldsymbol{z}_{n_k}) \cdots & \cdots & (\tilde{\boldsymbol{u}}_{n_k,m_k-1},A^{m_k}\boldsymbol{z}_{n_k}) \end{pmatrix}$$

We obtain the following algorithm called (MRZ)-Stab [10,11].

# **MRZ-Stab** algorithm

Choose  $x_0$  and  $\tilde{r}_0$ Set  $n_0 = 0$ ,  $\mathbf{r}_{n_0} = \mathbf{b} - A\mathbf{x}_0$ ,  $\mathbf{z}_{n_0} = \mathbf{r}_0$ ,  $\tilde{\mathbf{u}}_{n_{-1}} = 0$ ,  $\tilde{\mathbf{u}}_0 = \tilde{\mathbf{r}}_0$ ,  $\mathbf{z}_{n_{-1}} = 0$ For k = 0, 1, 2, ... until convergence do Find  $m_k$  such that  $(\tilde{\boldsymbol{u}}_{n_k}, A^i \boldsymbol{z}_{n_k}) = 0$  for  $i = 1, \dots, m_k - 1$  and  $(\tilde{\boldsymbol{u}}_{n_k}, A^{m_k} \boldsymbol{z}_{n_k}) \neq 0$  $n_{k+1} = n_k + m_k$ Solve  $(\tilde{U}_{n_k,m_k}^{\mathrm{T}}AZ_{n_k,m_k})\boldsymbol{\beta}_{n_{k+1}} = \tilde{U}_{n_k,m_k}^{\mathrm{T}}\boldsymbol{r}_{n_k}$  $\mathbf{r}_{n_{k+1}} = \mathbf{r}_{n_k} - A Z_{n_k, m_k} \boldsymbol{\beta}_{n_{k+1}}$  $\begin{aligned} \boldsymbol{x}_{n_{k+1}} &= \boldsymbol{x}_{n_k} + Z_{n_k, m_k} \boldsymbol{\beta}_{n_{k+1}} \\ \text{If } k \neq 0 \text{ then } \gamma_{n_{k+1}} &= \frac{(\boldsymbol{\tilde{u}}_{n_k}, A^{m_k} \boldsymbol{z}_{n_k})}{(\boldsymbol{\tilde{u}}_{n_{k-1}}, A^{m_{k-1}} \boldsymbol{z}_{n_{k-1}})} \text{ else } \gamma_{n_{k+1}} = 0 \end{aligned}$ Solve  $(\tilde{U}_{n_k,m_k}^{\mathrm{T}}AZ_{n_k,m_k})\boldsymbol{\alpha}_{n_{k+1}} = \tilde{U}_{n_k,m_k}^{\mathrm{T}}(A^{m_k+1}\boldsymbol{z}_{n_k})$  $z_{n_{k+1}} = A^{m_k} z_{n_k} - Z_{n_k, m_k} \alpha_{n_{k+1}} - \gamma_{n_{k+1}} z_{n_{k-1}}$  $\tilde{\boldsymbol{u}}_{n_{k+1}} = A^{m_k} \tilde{\boldsymbol{u}}_{n_k} - \tilde{U}_{n_k, m_k} \boldsymbol{\alpha}_{n_{k+1}} - \gamma_{n_{k+1}} \tilde{\boldsymbol{u}}_{n_{k-1}}$ end for

# 4. Polynomial approach

In this section, we will consider the same algorithms as in the preceding section, but we will now derive them from the formal orthogonal polynomial approach. Again, the regular and the nonregular cases will be treated. We will not develop this approach in much details since our aim is only to show that this polynomial approach is much simpler than the matrix one. The interested reader will find more details in the literature given at the end of the paper.

4.1. Regular case

If we set

$$p_k(\xi) = 1 + a_1^{(k)}\xi + \dots + a_k^{(k)}\xi^k = 1 + \xi q_{k-1}(\xi)\mathbf{r}_0$$

then, from (2), we have

$$\mathbf{r}_k = p_k(A)\mathbf{r}_0$$

and

$$\mathbf{x}_k = \mathbf{x}_0 - q_{k-1}(A)\mathbf{r}_0$$

Moreover, if we define the linear functional c on the space of polynomials by

$$c(\xi^{i}) = (\tilde{\mathbf{r}}_{0}, A^{i}\mathbf{r}_{0}) = (\tilde{\mathbf{r}}_{0}, \mathbf{w}_{i}) = c_{i}, \quad i = 0, 1, \dots$$
(22)

then

$$c(v_i(\xi)p_k(\xi)) = 0, \quad i = 0, \dots, k-1,$$

where  $\forall i, v_i$  is an arbitrary polynomial of exact degree *i*.

These relations show that  $p_k$  is the polynomial of degree at most k belonging to the family of formal orthogonal polynomials (FOP) with respect to c [3]. This polynomial is defined apart from a multiplying factor which is chosen, in our case, such that  $p_k(0) = 1$ . With this normalization, the polynomial  $p_k$  can be written as a ratio of determinants

$$p_{k}(\xi) = \begin{vmatrix} 1 & \xi & \cdots & \xi^{k} \\ c_{0} & c_{1} & \cdots & c_{k} \\ \vdots & \vdots & \vdots \\ c_{k-1} & c_{k} & \cdots & c_{2k-1} \end{vmatrix} \middle| \middle| \begin{vmatrix} c_{1} & c_{2} & \cdots & c_{k} \\ c_{2} & c_{3} & \cdots & c_{k+1} \\ \vdots & \vdots & \vdots \\ c_{k} & c_{k+1} & \cdots & c_{2k-1} \end{vmatrix}$$

Then we have

$$\mathbf{r}_{k} = \begin{vmatrix} \mathbf{r}_{0} & A\mathbf{r}_{0} & \cdots & A^{k}\mathbf{r}_{0} \\ c_{0} & c_{1} & \cdots & c_{k} \\ \vdots & \vdots & & \vdots \\ c_{k-1} & c_{k} & \cdots & c_{2k-1} \end{vmatrix} / \begin{vmatrix} c_{1} & c_{2} & \cdots & c_{k} \\ c_{2} & c_{3} & \cdots & c_{k+1} \\ \vdots & \vdots & & \vdots \\ c_{k} & c_{k+1} & \cdots & c_{2k-1} \end{vmatrix}$$

which correspond exactly to the ratio of determinants given in Remark 5, and

$$m{x}_k - m{x}_0 = egin{pmatrix} 0 & m{r}_0 & \cdots & A^{k-1} m{r}_0 \ c_0 & c_1 & \cdots & c_k \ dots & dots &$$

Since the determinants in the denominators of  $p_k$ ,  $r_k$  and  $x_k - x_0$  are in fact the preceding Hankel determinant  $H_k^{(1)}$ ,  $p_k$  exists and is unique if and only if this determinant is different from zero and

the existence of the polynomial  $p_k$  guarantees the existence and the uniqueness of  $r_k$  and  $x_k$ . Let us now consider the monic polynomial  $p_k^{(1)}$  of degree k belonging to the family of FOP with respect to the functional  $c^{(1)}$  defined by  $c^{(1)}(\xi^i) = c(\xi^{i+1})$ . It satisfies the orthogonality conditions

$$c^{(1)}(v_i(\xi)p_k^{(1)}(\xi)) = 0, \quad i = 0, \dots, k-1$$

and it can be written as a ratio of determinants

$$p_{k}^{(1)}(\xi) = \begin{vmatrix} c_{1} & c_{2} & \cdots & c_{k+1} \\ \vdots & \vdots & & \vdots \\ c_{k} & c_{k+1} & \cdots & c_{2k} \\ 1 & \xi & \cdots & \xi^{k} \end{vmatrix} \middle| \middle| \begin{vmatrix} c_{1} & c_{2} & \cdots & c_{k} \\ c_{2} & c_{3} & \cdots & c_{k+1} \\ \vdots & \vdots & & \vdots \\ c_{k} & c_{k+1} & \cdots & c_{2k-1} \end{vmatrix}$$

Since it has the same denominator as  $p_k$ ,  $p_k^{(1)}$  exists under the same condition and conversely. If we consider the vector  $\mathbf{z}_k = p_k^{(1)}(A)\mathbf{r}_0$ , we obtain

$$\mathbf{z}_{k} = \begin{vmatrix} A^{k} \mathbf{r}_{0} \ \mathbf{r}_{0} \ \cdots \ A^{k-1} \mathbf{r}_{0} \\ c_{k+1} \ c_{1} \ \cdots \ c_{k} \\ \vdots \ \vdots \ \vdots \ \vdots \\ c_{2k} \ c_{k} \ \cdots \ c_{2k-1} \end{vmatrix} \middle| \middle| \left| \begin{vmatrix} c_{1} \ c_{2} \ \cdots \ c_{k} \\ c_{2} \ c_{3} \ \cdots \ c_{k+1} \\ \vdots \ \vdots \ \vdots \\ c_{k} \ c_{k+1} \ \cdots \ c_{2k-1} \end{vmatrix} \right|$$

and since  $c_i = ((A^T)^{i-j} \tilde{r}_0, A^j r_0) = (\tilde{v}_{i-j}, w_j)$ , for j = 0, ..., i, we recover the ratio of determinants given in (14).

In the sequel, the linear functionals c and  $c^{(1)}$  will always act on the variable  $\xi$  which will be suppressed when unnecessary.

The recursive computation of the polynomials  $p_k$ , needed in Lanczos method, can be achieved in several ways. For instance, we can use the usual three-term recurrence relation, or relations involving also the polynomials of the family  $\{p_k^{(1)}\}$  or polynomials proportional to them. Using such recurrence relationships leads to all the known algorithms for implementing the method of Lanczos and also to new ones. See [12] for a unified presentation of all these methods based on the theory of FOP and [2] for more details.

Let us now see how to compute the polynomial  $p_{k+1}$  from  $p_k$  and  $p_k^{(1)}$ . The following relation holds:

$$p_{k+1}(\xi) = p_k(\xi) - \beta_{k+1}\xi p_k^{(1)}(\xi)$$

$$p_0(\xi) = p_0^{(1)}(\xi) = 1.$$
(23)

with  $p_0(\xi) = p_0^{(1)}(\xi) = 1$ . Indeed, let  $\tilde{p}_i$  be an auxiliary family of polynomials so that, for all *i*,  $\tilde{p}_i$  has degree *i* exactly. Multiplying (23) by  $\tilde{p}_i$  and applying *c* gives

$$c(\tilde{p}_{i}p_{k+1}) = c(\tilde{p}_{i}p_{k}) - \beta_{k+1}c^{(1)}(\tilde{p}_{i}p_{k}^{(1)}).$$

Owing to the orthogonality conditions of the two families of formal orthogonal polynomials, the quantities on the right-hand side are equal to zero for i = 0, ..., k - 1. So,  $c(\tilde{p}_i p_{k+1}) = 0$  for i = 0, ..., k - 1. Moreover, taking

$$\beta_{k+1} = c(\tilde{p}_k p_k) / c^{(1)}(\tilde{p}_k p_k^{(1)})$$
(24)

leads to  $c(\tilde{p}_k p_{k+1})=0$  which shows (by a uniqueness argument) that the polynomial  $p_{k+1}$  obtained by the preceding recurrence relationship is the (k+1)th polynomial of the family of formal orthogonal polynomials with respect to the linear functional c. Replacing  $\xi$  by A in (23) gives a matrix. Multiplying  $\mathbf{r}_0$  by this matrix leads to

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \beta_{k+1} A \mathbf{z}_k.$$

As it is easily seen, this recurrence relationship is much simpler to obtain by using the theory of formal orthogonal polynomials than by the matrix approach of Section 3.

It must be noticed that replacing  $p_k^{(1)}$  by a proportional polynomial  $q_k(\xi) = a_k p_k^{(1)}(\xi)$  in (23) and (24) does not change (23) since

$$\beta_{k+1}q_k = \frac{c(\tilde{p}_k p_k)}{c^{(1)}(\tilde{p}_k q_k)}q_k = \frac{c(\tilde{p}_k p_k)}{a_k c^{(1)}(\tilde{p}_k p_k^{(1)})}a_k p_k^{(1)} = \beta_{k+1} p_k^{(1)}.$$

A breakdown occurs in this relation if and only if

$$c^{(1)}(\tilde{p}_k p_k^{(1)}) = c^{(1)}(\xi^k p_k^{(1)}) = 0$$

Thus, we see that a breakdown occurs if and only if  $H_{k+1}^{(1)} = 0$  or, in other terms, if and only if  $p_{k+1}^{(1)}$  and  $p_{k+1}$  do not exist.

Since the polynomials  $\{p_k^{(1)}\}$  form a family of FOP, they also satisfy the usual three-term recurrence relationship which becomes, since they are monic

$$p_{k+1}^{(1)}(\xi) = (\xi - \alpha_{k+1})p_k^{(1)}(\xi) - \gamma_{k+1}p_{k-1}^{(1)}(\xi)$$
(25)

with  $p_0^{(1)}(\xi) = 1$  and  $p_{-1}^{(1)}(\xi) = 0$ . Again, by the same type of procedure as above, we obtain the coefficients  $\alpha_{k+1}$  and  $\gamma_{k+1}$  as

$$\begin{split} \gamma_{k+1} &= c^{(1)}(\xi \, \tilde{p}_{k-1} \, p_k^{(1)}) / c^{(1)}(\, \tilde{p}_{k-1} \, p_{k-1}^{(1)}), \\ \alpha_{k+1} &= [c^{(1)}(\xi \, \tilde{p}_k \, p_k^{(1)}) - \gamma_{k+1} c^{(1)}(\, \tilde{p}_k \, p_{k-1}^{(1)})] / c^{(1)}(\, \tilde{p}_k \, p_k^{(1)}). \end{split}$$

So, a breakdown occurs in this relation if and only if  $c^{(1)}(\tilde{p}_k p_k^{(1)}) = 0$  (since  $p_k^{(1)}$  exists it means that  $c^{(1)}(\tilde{p}_{k-1} p_{k-1}^{(1)}) \neq 0$  and, thus, no division by zero can occur in the expression of  $\gamma_{k+1}$ ). But

$$c^{(1)}(\tilde{p}_k p_k^{(1)}) = c^{(1)}(\xi^k p_k^{(1)}) = H_{k+1}^{(1)}/H_k^{(1)}$$

and we recover the condition for the existence of  $p_{k+1}$ . Thus, a breakdown occurs in (25) if and only if the polynomials  $p_{k+1}$  and  $p_{k+1}^{(1)}$  do not exist. Such a breakdown is called a *true breakdown*.

Using alternately relations (23) and (25) allows to compute simultaneously the two families  $\{p_k\}$  and  $\{p_k^{(1)}\}$ . Only true breakdowns can occur in these two relations.

These recurrence relationships can be used for implementing Lanczos method and they give

$$\mathbf{r}_{k+1} = \mathbf{r}_k - \beta_{k+1} A \mathbf{z}_k,$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \beta_{k+1} \boldsymbol{z}_k,$$

$$\boldsymbol{z}_{k+1} = A \boldsymbol{z}_k - \boldsymbol{\alpha}_{k+1} \boldsymbol{z}_k - \boldsymbol{\gamma}_{k+1} \boldsymbol{z}_{k-1}.$$

The coefficients of these recurrence relationships can be computed as above. Using definition (22) of the linear functional *c*, we obtain formulas (10) and (13) for the choice  $\tilde{p}_k(\xi) = \xi^k$ , and formulas (15) and (17) for the choice  $\tilde{p}_k(\xi) \equiv p_k^{(1)}(\xi)$ . Hence the algorithms of Section 3.1 have been recovered.

### 4.2. Avoiding breakdowns

Let us now see how to avoid the true breakdowns which can occur in the recurrence relationships given in the preceding section.

The treatment of a true breakdown consists of the following steps:

- 1. recognize the occurrence of such a breakdown, that is the nonexistence of the next orthogonal polynomial(s),
- 2. determine the degree of the next existing (called regular) orthogonal polynomial,
- 3. jump over the nonexisting orthogonal polynomials and built a recurrence relationship which only uses the regular ones.

This problem was completely solved by Draux [15] in the case of monic orthogonal polynomials. Since the polynomials  $p_k^{(1)}$  are monic and the conditions for the existence of the polynomials  $p_k$  and  $p_k^{(1)}$  are the same, we will apply his results to avoid true breakdowns.

The subsequence of regular orthogonal polynomials will be denoted by  $\{p_{n_k}^{(1)}\}\)$ . The polynomials of degrees  $n_k + 1, \ldots, n_k + m_k - 1$  do not exist. So, the *k*th regular polynomial of the family,  $p_{n_k}^{(1)}$ , has degree  $n_k$ , with  $n_k \ge k$ , and the next regular polynomial is  $p_{n_{k+1}}^{(1)}$  with  $n_{k+1} = n_k + m_k$ . Then, as above,  $m_k$  is the length of the *jump* between the degree of  $p_{n_k}^{(1)}$  and the degree of  $p_{n_{k+1}}^{(1)}$ . Similar considerations hold for the polynomials of the family  $\{p_k\}$ .

It was proved by Draux [15] that the length  $m_k$  of the jump is given by the conditions

$$c^{(1)}(\xi^i p_{n_k}^{(1)}) \begin{cases} = 0 \text{ for } i = 0, \dots, n_k + m_k - 2, \\ \neq 0 \text{ for } i = n_k + m_k - 1. \end{cases}$$

Moreover, these polynomials can be recursively computed by the relationship

$$p_{n_{k+1}}^{(1)}(\xi) = (\alpha_0 + \dots + \alpha_{m_k-1}\xi^{m_k-1} + \xi^{m_k})p_{n_k}^{(1)}(\xi) - \gamma_{k+1}p_{n_{k-1}}^{(1)}(\xi)$$
(26)  
for  $k = 0, 1, \dots$ , with  $p_{-1}^{(1)}(\xi) = 0, p_0^{(1)}(\xi) = 1, \gamma_1 = 0$  and  
 $\gamma_{k+1} = c^{(1)}(\xi^{n_k+m_k-1}p_{n_k}^{(1)})/c^{(1)}(\xi^{n_k-1}p_{n_{k-1}}^{(1)}),$   
 $\alpha_{m_k-1}c^{(1)}(\xi^{n_k+m_k-1}p_{n_k}^{(1)}) + c^{(1)}(\xi^{n_k+m_k}p_{n_k}^{(1)}) = \gamma_{k+1}c^{(1)}(\xi^{n_k}p_{n_{k-1}}^{(1)}),$   
:  
 $\alpha_0c^{(1)}(\xi^{n_k+m_k-1}p_{n_k}^{(1)}) + \dots + \alpha_{m_k-1}c^{(1)}(\xi^{n_k+2m_k-2}p_{n_k}^{(1)}) + c^{(1)}(\xi^{n_k+2m_k-1}p_{n_k}^{(1)})$   
 $= \gamma_{k+1}c^{(1)}(\xi^{n_k+m_k-1}p_{n_{k-1}}^{(1)}).$ 

Since, by definition of  $m_k$ ,  $c^{(1)}(\zeta^{n_k+m_k-1}p_{n_k}^{(1)}) \neq 0$ , this system is never singular and no breakdown can occur in (26).

For implementing Lanczos method by the algorithm Lanczos/Orthodir, we also need to compute  $p_{n_{k+1}}$  from  $p_{n_k}$  and  $p_{n_k}^{(1)}$ . As proved in [8], we have the following relation which generalizes (23):

$$p_{n_{k+1}}(\xi) = p_{n_k}(\xi) - \xi(\beta_0 + \dots + \beta_{m_k - 1}\xi^{m_k - 1})p_{n_k}^{(1)}(\xi),$$
(27)

where the  $\beta_i$ 's are the solution of the system

258

$$\beta_{m_k-1}c^{(1)}(\xi^{n_k+m_k-1}p_{n_k}^{(1)}) = c(\xi^{n_k}p_{n_k}),$$
  

$$\vdots$$
  

$$\beta_0c^{(1)}(\xi^{n_k+m_k-1}p_{n_k}^{(1)}) + \dots + \beta_{m_k-1}c^{(1)}(\xi^{n_k+2m_k-2}p_{n_k}^{(1)}) = c(\xi^{n_k+m_k-1}p_{n_k}).$$

Again, since, by definition of  $m_k$ ,  $c^{(1)}(\xi^{n_k+m_k-1}p_{n_k}^{(1)}) \neq 0$ , this system is never singular and no breakdown can occur in (27).

The systems giving the coefficients  $\alpha_i$ ,  $\gamma_{k+1}$  and  $\beta_i$  are the same as those given in Section 3.2 and we have  $\alpha_{n_{k+1}} = (\alpha_0, \dots, \alpha_{m_{k-1}})^T$  and  $\beta_{n_{k+1}} = (\beta_0, \dots, \beta_{m_{k-1}})^T$ . Then, using alternately (26) and (27) gives the MRZ algorithm.

The MRZ-Stab algorithm given above can be also obtained from the polynomial approach by writing the orthogonality conditions on a different basis than the canonical one. Another version of the MRZ, which is more stable and only needs the storage of a fixed number of vectors independently of the length of the jumps, was recently proposed in [11]; see also [10]. It is based on Horner's rule for computing a polynomial and, for that reason, it was called the HMRZ algorithm. The HMRZ-Stab was obtained as a variant of the HMRZ, writing again the orthogonality conditions on a basis different from the canonical one. A quite similar technique is also described in [1].

Quite similar algorithms for treating this kind of breakdowns were also given by Gutknecht [22,23]. They are based on the fact that the occurrence of a breakdown corresponds to a square block of adjacent identical approximants in the Padé table and a look-ahead strategy is also used. This approach and that of this subsection were compared in [16]. Another technique, due to Graves–Morris [19], is based on the connection between FOP and Padé approximants. Indeed, a breakdown also corresponds to a block of identical adjacent FOP and his technique consists of turning around such a block instead of going through it. This idea was extended to the BICGSTAB in [20]. Zero divisor-free Hestenes–Stiefel-type conjugate direction algorithms can be found in [24]. Another scheme, based on a modified Krylov subspace approach, is presented in [31]. The problem of breakdown can also be treated by introducing new vectors into Krylov subspaces [29] or by an adaptative block Lanczos algorithm [30]. Necessary and sufficient conditions for look-ahead versions of the block conjugate gradient algorithm to be free from serious and incurable breakdowns are given in [13]. Thus, unstable versions of the algorithms can be identified and stable ones proposed.

# Acknowledgements

We would like to thank the referees for constructive comments.

# References

- [1] H. Ayachour, Avoiding look-ahead in the Lanczos method and Padé approximation, Appl. Math. 26 (1999) 33-62.
- [2] C. Baheux, New implementations of Lanczos method, J. Comput. Appl. Math. 57 (1995) 3-15.
- [3] C. Brezinski, Padé-Type Approximation and General Orthogonal Polynomials, ISNM Vol. 50, Birkhäuser-Verlag, Basel, 1980.
- [4] C. Brezinski, Other manifestations of the Schur complement, Linear Algebra Appl. 111 (1988) 231-247.
- [5] C. Brezinski, M. Redivo-Zaglia, Breakdowns in the computation of orthogonal polynomials, in: A. Cuyt (Ed.), Nonlinear Numerical Methods and Rational Approximation, Kluwer, Dordrecht, 1994, pp. 49–59.
- [6] C. Brezinski, M. Redivo-Zaglia, H. Sadok, Avoiding breakdown and near-breakdown in Lanczos type algorithms, Numer. Algorithms 1 (1991) 261–284.
- [7] C. Brezinski, M. Redivo-Zaglia, H. Sadok, Addendum to Avoiding breakdown and near-breakdown in Lanczos type algorithms, Numer. Algorithms 2 (1992) 133–136.
- [8] C. Brezinski, M. Redivo-Zaglia, H. Sadok, A breakdown-free Lanczos type algorithm for solving linear systems, Numer. Math. 63 (1992) 29–38.
- [9] C. Brezinski, M. Redivo-Zaglia, H. Sadok, Breakdowns in the implementation of the Lánczos method for solving linear systems, Comput. Math. Appl. 33 (1997) 31–44.
- [10] C. Brezinski, M. Redivo-Zaglia, H. Sadok, New look-ahead implementations of Lanczos method for unsymmetric systems, in: J. Wang et al. (Eds.), Iterative Methods in Scientific Computation, IMACS Ser. Comput. Appl. Math. Vol. 4, 1998, pp. 9–14.
- [11] C. Brezinski, M. Redivo-Zaglia, H. Sadok, New look-ahead Lanczos-type algorithms for linear systems, Numer. Math. 83 (1999) 53–85.
- [12] C. Brezinski, H. Sadok, Lanczos type methods for solving systems of linear equations, Appl. Numer. Math. 11 (1993) 443–473.
- [13] C.G. Broyden, Look-ahead block-CG algorithms, Optim. Methods Software, to appear.
- [14] R.W. Cottle, Manifestations of the Schur complement, Linear Algebra Appl. 8 (1974) 189-211.
- [15] A. Draux, in: Polynômes Orthogonaux Formels. Applications, Lecture Notes in Mathematics, Vol. 974, Springer, Berlin, 1983.
- [16] A. El Guennouni, A unified approach to some strategies for the treatment of breakdown in Lanczos-type algorithms, Applicationes Mathematical 26 (1999) 477–488.
- [17] V.N. Faddeeva, Computational Methods of Linear Algebra, Dover, New York, 1959.
- [18] R. Fletcher, Conjugate gradient methods for indefinite systems, in: G.A. Watson (Ed.), Numerical Analysis, Dundee 1975, Lecture Notes in Mathematics, Vol. 506, Springer, Berlin, 1976.
- [19] P.R. Graves-Morris, A "Look-around Lanczos" algorithm for solving a system of linear equations, Numer. Algorithms 15 (1998) 246–274.
- [20] P.R. Graves-Morris, A. Salam, Avoiding breakdown in Van der Vorst's method, Numer. Algorithms 21 (1999) 205–223.
- [21] M.H. Gutknecht, The unsymmetric Lanczos algorithms and their relations to Padé approximation, continued fractions and the qd algorithm, Proceedings of the Copper Mountain Conference on Iterative Methods, 1990, unpublished.
- [22] M.H. Gutknecht, A completed theory of the unsymmetric Lanczos process and related algorithms, Part I, SIAM J. Matrix Anal. Appl. 13 (1992) 594–639.
- [23] M.H. Gutknecht, A completed theory of the unsymmetric Lanczos process and related algorithms, Part II, SIAM J. Matrix Anal. Appl. 15 (1994) 15–58.
- [24] Cs.J. Hegedüs, Generating conjugate directions for arbitrary matrices by matrix equations, Comput. Math. Appl. 21 (1991) 71–85; 87–94.
- [25] M.R. Hestenes, E. Stiefel, Methods of conjugate gradients for solving linear systems, J. Res. Natl. Bur. Stand. 49 (1952) 409–436.
- [26] K.C. Jea, D.M. Young, On the simplification of generalized conjugate gradient methods for nonsymmetrizable linear systems, Linear Algebra Appl. 52/53 (1983) 399–417.
- [27] C. Lanczos, An iteration method for the solution of the eigenvalue problem of linear differential and integral operators, J. Res. Natl. Bur. Stand. 45 (1950) 255–282.

- [28] C. Lanczos, Solution of systems of linear equations by minimized iterations, J. Res. Natl. Bur. Stand. 49 (1952) 33–53.
- [29] Qiang Ye, A breakdown-free variation of the nonsymmetric Lanczos algorithm, Math. Comput. 62 (1994) 179–207.
- [30] Qiang Ye, An adaptative block Lanczos algorithm, Numer. Algorithms 12 (1996) 97–110.
- [31] C.H. Tong, Qiang Ye, A linear system solver based on a modified Krylov subspace method for breakdown recovery, Numer. Algorithms 12 (1996) 233–251.
- [32] P. Wynn, Upon systems of recursions which obtain among the quotients of the Padé table, Numer. Math. 8 (1966) 264–269.
- [33] D.M. Young, K.C. Jea, Generalized conjugate-gradient acceleration of nonsymmetrizable iterative methods, Linear Algebra Appl. 34 (1980) 159–194.



Journal of Computational and Applied Mathematics 123 (2000) 261-292

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.nl/locate/cam

# Analysis of acceleration strategies for restarted minimal residual methods

Michael Eiermann<sup>a,\*</sup>, Oliver G. Ernst<sup>a</sup>, Olaf Schneider<sup>b</sup>

<sup>a</sup>Institut für Angewandte Mathematik II, TU Bergakademie Freiberg, Germany <sup>b</sup>Graduiertenkolleg Räumliche Statistik, TU Bergakademie Freiberg, Germany

Received 31 August 1999; received in revised form 30 November 1999

#### Abstract

We provide an overview of existing strategies which compensate for the deterioration of convergence of minimum residual (MR) Krylov subspace methods due to restarting. We evaluate the popular practice of using nearly invariant subspaces to either augment Krylov subspaces or to construct preconditioners which invert on these subspaces. In the case where these spaces are exactly invariant, the augmentation approach is shown to be superior. We further show how a strategy recently introduced by de Sturler for truncating the approximation space of an MR method can be interpreted as a controlled loosening of the condition for global MR approximation based on the canonical angles between subspaces. For the special case of Krylov subspace methods, we give a concise derivation of the role of Ritz and harmonic Ritz values and vectors in the polynomial description of Krylov spaces as well as of the use of the implicitly updated Arnoldi method for manipulating Krylov spaces. (c) 2000 Elsevier Science B.V. All rights reserved.

#### MSC: 65F10; 65F15

*Keywords:* Krylov subspace methods; Restarted Krylov subspace methods; Augmented Krylov subspace methods; Deflated Krylov subspace methods; Optimal truncation; GMRES; GMRES(m); Ritz values; Harmonic Ritz values; Implicitly restarted Arnoldi method

# 1. Introduction

When Krylov subspace methods are employed for approximating the solution of large sparse or structured linear systems of equations

 $A\mathbf{x} = \mathbf{b}, \quad A \text{ nonsingular},$ 

(1)

their stable implementation requires the construction of orthonormal bases of spaces which increase in dimension with each iteration step.

<sup>\*</sup> Corresponding author.

E-mail address: eiermann@math.tu-freiberg.de (M. Eiermann).

If the operator A is Hermitian, or if the notion of orthogonality is suitably modified (see [3]), then these bases can be generated by short recurrence formulas, and this is the key to the efficiency of such widely used methods as CG, MINRES, BCG and QMR (see the monographs of Saad [18] and Greenbaum [9] for an exposition of these methods). For non-Hermitian A, however, a result of Faber and Manteuffel [6] implies that the construction of such bases which are orthonormal with respect to a *given* inner product generally involves orthogonalization against all previously generated vectors, as in algorithms such as FOM, GCR and GMRES. When the resulting storage and computation requirements make these methods impractical, they are often modified to compute an approximation with respect to a space of affordable size, after which the algorithm is *restarted* using the current approximation as the initial guess. Since restarting usually results in slower convergence (or the loss thereof altogether), much recent work has been devoted to compensating for the loss of information that occurs upon restarting by retaining a judiciously chosen part of the previously generated space.

We distinguish two fundamental strategies in existing work: One lies in identifying a subspace  $\mathcal{U}$  which slows convergence, approximating this space and eliminating its influence from the iteration process. We shall refer to such a procedure as *deflation*. Such "problematic" subspaces are often identified as eigenspaces of A associated with eigenvalues of small magnitude, but other spaces may sometimes be better suited. Examples of this approach are the augmentation method introduced by Morgan [13,14] and analyzed by Saad [19,20] and Chapman and Saad [2]. Another device for eliminating  $\mathcal{U}$  from the iteration is to introduce a preconditioner which inverts the orthogonal section of A onto  $\mathcal{U}$ , as proposed by Erhel et al. [5], Baglama et al. [1] and, with certain modifications, by Kharchenko and Yeremin [10]. The second fundamental strategy consists of identifying the essential orthogonality constraints by comparing angles between subspaces and maintaining orthogonality only against the most important subspace of a given dimension. Such a strategy is proposed by de Sturler [26].

The main intent of this paper is to provide an abstract framework which permits a uniform presentation as well as a comparison of these methods. Although proposed originally in association with Krylov subspace methods, these approaches can all be applied in the case of completely general correction spaces, as we show in Sections 2.3 and 2.4. In the Krylov subspace case, much emphasis has been placed on the approximation properties of invariant or nearly invariant correction spaces, particularly so in connection with augmentation strategies. We present several results which attempt to shed light on exactly when nearly invariant subspaces are useful. We also show that Krylov spaces can never contain invariant subspaces without being themselves invariant; similarly, an invariant space cannot contain a Krylov space without also containing the associated smallest invariant Krylov space. However, we show that augmenting by invariant subspaces does eliminate the components of the resulting residual in this space.

Since none of the results we shall derive are restricted to the finite-dimensional case, the setting of a separable Hilbert space  $\mathscr{H}$  with inner product  $(\cdot, \cdot)$  and associated norm  $\|\cdot\|$  is the most natural, and we assume that A in (1) is a bounded linear operator.

Section 2 reviews the basic theory of iterative subspace correction methods for solving (1), which are based on the minimal residual (MR) and orthogonal residual (OR) approaches. We highlight the fundamental role of the angles between correction and approximation spaces as introduced in [3]. In addition, the necessary orthogonality relations are described, which must hold for the MR approximation with respect to two arbitrary subspaces to yield the MR approximation with respect

to the sum of these spaces, and it is shown how these orthogonality relations may be relaxed in an optimal way.

Section 3 reviews the implications of using Krylov spaces with regard to the simplification of the algorithms and the advantages of the polynomial representation of the residuals. We include new, much simplified derivations of the role of Ritz and harmonic Ritz values and vectors of *A* as well as how the recently developed implicitly restarted Arnoldi method can be used to restart the Arnoldi process without additional matrix–vector multiplications.

Section 4 discusses possible strategies for augmenting Krylov spaces and derives some results showing the limitations for augmenting Krylov spaces to obtain *A*-invariant subspaces. The remainder of Section 4 gives an overview of the most popular restart algorithms, beginning with restarted GMRES itself, for which we give a surprising example for which GMRES with longer restart lengths actually displays slower convergence than for shorter restart lengths. Next, the augmentation algorithm of Morgan is presented, and a new, much simplified proof is given that the augmented Krylov spaces are themselves Krylov spaces. In addition, we show that, at least in the case of exactly invariant subspaces, the augmentation approach is superior to the preconditioning algorithms of Erhel et al. [5] and Baglama et al. [1]. Finally, the optimal truncation algorithm of de Sturler is presented as an implementation of the selective orthogonalization strategy of Section 2.4.

#### 2. Minimal and orthogonal residual methods

#### 2.1. Definitions and basic theory

Given an initial guess  $x_0$  for the solution of (1) together with the associated residual vector  $r_0 = b - Ax_0$  and a sequence of nested *correction spaces* in  $\mathcal{H}$ ,

$$\{\mathbf{0}\} = \mathscr{C}_0 \subset \mathscr{C}_1 \subset \mathscr{C}_2 \subset \cdots \subset \mathscr{C}_m \subset \mathscr{C}_{m+1} \subset \cdots$$

(for notational convenience, we assume that dim  $\mathscr{C}_m = m$ ), all methods we shall consider lead to iterates of the form  $\mathbf{x}_m = \mathbf{x}_0 + \mathbf{c}_m$  with  $\mathbf{c}_m \in \mathscr{C}_m$ . They differ in the way the corrections  $\mathbf{c}_m$  are selected from  $\mathscr{C}_m$ .

For the *m*th MR iterate  $\mathbf{x}_m^{\text{MR}} = \mathbf{x}_0 + \mathbf{c}_m^{\text{MR}}$ , the correction  $\mathbf{c}_m^{\text{MR}}$  is chosen from  $\mathscr{C}_m$  to satisfy

$$\|\boldsymbol{b} - A\boldsymbol{x}_{m}^{\mathrm{MR}}\| = \|\boldsymbol{r}_{0} - A\boldsymbol{c}_{m}^{\mathrm{MR}}\| = \min_{\boldsymbol{c} \in \mathscr{C}_{m}} \|\boldsymbol{r}_{0} - A\boldsymbol{c}\|,$$
(2)

or equivalently, such that  $Ac_m^{MR}$  is the best approximation to  $r_0$  from the *m*th approximation space  $\mathcal{W}_m := A\mathcal{C}_m$ . As A is invertible,  $c_m^{MR}$  and  $x_m^{MR}$  are uniquely defined; specifically, they are characterized by

$$\boldsymbol{r}_{m}^{\mathrm{MR}} = \boldsymbol{b} - A\boldsymbol{x}_{m}^{\mathrm{MR}} = \boldsymbol{r}_{0} - A\boldsymbol{c}_{m}^{\mathrm{MR}} \perp \mathcal{W}_{m}.$$
(3)

To define the OR iterates we introduce the residual spaces

$$\mathscr{V}_{m+1} := \operatorname{span}\{\mathbf{r}_0\} + \mathscr{W}_m, \quad m \ge 0, \tag{4}$$

the name of which derives from the fact that the residual b - Ax lies in the space span  $\{r_0\} + A\mathscr{C}_m = \mathscr{V}_{m+1}$ whenever  $x = x_0 + c$ ,  $c \in \mathscr{C}_m$ . We now set  $x_m^{OR} = x_0 + c_m^{OR}$  with  $c_m^{OR} \in \mathscr{C}_m$  such that

$$\boldsymbol{r}_{m}^{\mathrm{OR}} = \boldsymbol{b} - A\boldsymbol{x}_{m}^{\mathrm{OR}} = \boldsymbol{r}_{0} - A\boldsymbol{c}_{m}^{\mathrm{OR}} \perp \mathscr{V}_{m}.$$
(5)

In contrast to the MR approximation, the OR iterate may not exist for every *m*; when it does exist, which is the case if and only if  $\mathscr{H} = \mathscr{W}_m \oplus \mathscr{V}_m^{\perp}$  (see [3, Proposition 2.2]), then it is uniquely determined.

Clearly, the *m*th MR approximant  $\mathbf{x}_m^{\text{MR}}$  as well as the *m*th OR approximation  $\mathbf{x}_m^{\text{OR}}$  coincide with the exact solution of (1) if and only if  $A^{-1}\mathbf{r}_0 \in \mathscr{C}_m$  or equivalently, if and only if  $\mathbf{r}_0 \in \mathscr{W}_m$ . If such an index *m* exists we define

$$L := \min\{m : \boldsymbol{x}_{m}^{\mathrm{MR}} = A^{-1}\boldsymbol{b}\} = \min\{m : \boldsymbol{x}_{m}^{\mathrm{OR}} = A^{-1}\boldsymbol{b}\}$$
(6)

and otherwise set  $L := \infty$ . Alternative characterizations of the termination index L are

$$L = \min\{m : \mathbf{r}_0 \in \mathcal{W}_m\} = \min\{m : \mathcal{W}_m = \mathcal{V}_m\} = \min\{m : \mathcal{V}_m = \mathcal{V}_{m+1}\}.$$
(7)

The most popular implementations of both MR and OR methods rely on orthonormal bases  $\{v_1, \ldots, v_{m+1}\}$  of the residual spaces  $\mathscr{V}_{m+1}$  generated inductively by orthonormalizing  $Ac_m$  against a (previously constructed) orthonormal basis  $\{v_1, \ldots, v_m\}$  of  $\mathscr{V}_m$  using the (modified) Gram–Schmidt algorithm. Here  $c_m$  is an arbitrary vector from  $\mathscr{C}_m \setminus \mathscr{C}_{m-1}$  and  $v_1 = r_0/\beta$  with  $\beta := ||r_0||$ . As long as  $Ac_m \notin \mathscr{V}_m$ , a new orthonormal vector  $v_{m+1}$  is generated and we may proceed to the next step. If, however,  $Ac_m \in \mathscr{V}_m$ , which is equivalent to  $Ac_m \in \text{span}\{r_0, Ac_1, \ldots, Ac_{m-1}\}$ , then the algorithm terminates in step m. Since  $Ac_1, \ldots, Ac_m$  are linearly independent (because A is invertible), we see from (6) that  $Ac_m \in \mathscr{V}_m$  is equivalent to m = L. In summary: The Gram–Schmidt process is well defined up to the last step, in which  $\mathbf{x}_{L}^{\text{IR}} = \mathbf{x}_{L}^{\text{OR}} = A^{-1}\mathbf{b}$ .

With  $C_m := [c_1 c_2 \cdots c_m]$  and  $V_{m+1} := [v_1 v_2 \cdots v_{m+1}]$ , the first *m* orthonormalization steps establish the following Arnoldi-type decomposition of *A*:

$$AC_{m} = V_{m+1}H_{m} = V_{m}H_{m} + \eta_{m+1,m}\boldsymbol{v}_{m+1}\boldsymbol{u}_{m}^{1},$$
(8)

(for m = L, we have  $AC_L = V_L H_L$ ), where  $\tilde{H}_m = [\eta_{j,k}] \in \mathbb{C}^{(m+1) \times m}$  is an upper Hessenberg matrix and  $H_m := [I_m \quad \mathbf{0}] \tilde{H}_m \in \mathbb{C}^{m \times m}$  is the square matrix obtained by deleting the last row of  $\tilde{H}_m$ . The entries of  $\tilde{H}_m$  are given by  $\eta_{j,k} = (Ac_k, \mathbf{v}_j)$ ,  $1 \le k \le j \le m$ , and  $\eta_{k+1,k} = ||Ac_k - \sum_{j=1}^k \eta_{j,k} \mathbf{v}_j|| \ge 0$ , with equality holding if and only if k = L. In other words,  $\tilde{H}_m$  is an unreduced upper Hessenberg matrix (and hence of full rank m) as long as m < L. For m = L,  $AC_L = V_L H_L$  implies that  $H_L$  is nonsingular because A is invertible and both  $C_L$  and  $V_L$  have rank L.

With respect to the orthonormal basis  $V_{m+1}$  of  $\mathscr{V}_{m+1}$ , the vector  $\mathbf{r}_0 = \beta \mathbf{v}_1 = V_{m+1}\beta \mathbf{u}_1^{(m+1)}$  has the coordinates  $\beta \mathbf{u}_1^{(m+1)}$  ( $\mathbf{u}_1^{(m+1)} \in \mathbb{C}^{m+1}$  denotes the first unit vector), while the approximation space  $\mathscr{W}_m = A\mathscr{C}_m$  is represented by the column space of  $\tilde{H}_m$ . Consequently,

$$\min_{\boldsymbol{c}=C_m\boldsymbol{y}\in\mathscr{C}_m} \|\boldsymbol{r}_0-A\boldsymbol{c}\| \text{ and } \min_{\boldsymbol{y}\in\mathbb{C}^m} \|\beta\boldsymbol{u}_1^{(m+1)}-\tilde{H}_m\boldsymbol{y}\|_2$$

are equivalent problems ( $\|\cdot\|_2$  denotes the Euclidean norm in  $\mathbb{C}^{m+1}$ ).

For  $\mathbf{x}_m^{\text{MR}} = \mathbf{x}_0 + C_m \mathbf{y}_m^{\text{MR}}$ , condition (2) therefore leads to the least-squares problem

$$\|\beta \boldsymbol{u}_{1}^{(m+1)} - \tilde{H}_{m} \boldsymbol{y}_{m}^{\mathrm{MR}}\|_{2} = \min_{\boldsymbol{y} \in \mathbb{C}^{m}} \|\beta \boldsymbol{u}_{1}^{(m+1)} - \tilde{H}_{m} \boldsymbol{y}\|_{2}.$$
(9)

Representing the OR iterate as  $\mathbf{x}_m^{\text{OR}} = \mathbf{x}_0 + C_m \mathbf{y}_m^{\text{OR}}$ , the Galerkin condition  $\mathbf{r}_m^{\text{OR}} \perp \mathcal{V}_m$  (cf. (5)) similarly leads to the linear system

$$\mathbf{0} = [I_m \quad \mathbf{0}](\beta \boldsymbol{u}_1^{(m+1)} - \tilde{H}_m \boldsymbol{y}_m^{\mathrm{OR}}) = \beta \boldsymbol{u}_1^{(m)} - H_m \boldsymbol{y}_m^{\mathrm{OR}}$$

<sup>&</sup>lt;sup>1</sup> For convenience we shall identify a basis  $\{v_1, \ldots, v_m\}$  with its representation as the row vector  $V_m = [v_1 \cdots v_m]$ .

It can be shown (see [3, Remark 4.2]) that nonsingularity of  $H_m$  is equivalent to the existence of the OR approximation  $\mathbf{x}_m^{OR}$ .

The orthonormal basis  $\{v_1, \ldots, v_m, v_{m+1}\}$  of the residual space  $\mathscr{V}_{m+1}$  is the key to a simple representation of the quantities related to the OR approximation. For instance,  $\mathbf{r}_m^{\text{OR}}$  is a scalar multiple of  $\mathbf{v}_{m+1}$ , as follows immediately from  $\mathscr{V}_{m+1} \ni \mathbf{r}_m^{\text{OR}} \perp \mathscr{V}_m = \text{span}\{\mathbf{v}_1, \ldots, \mathbf{v}_m\}$ . Since  $\mathbf{r}_m^{\text{MR}} \perp \mathscr{W}_m$  (see (3)), an orthonormal basis  $\{\hat{\mathbf{v}}_1, \ldots, \hat{\mathbf{v}}_m, \tilde{\mathbf{v}}_{m+1}\}$  of  $\mathscr{V}_{m+1}$  with the analogous property with regard to the MR approximation should fulfill the condition  $\text{span}\{\hat{\mathbf{v}}_1, \ldots, \hat{\mathbf{v}}_m\} = \mathscr{W}_m$ .

It was already noted by Paige and Saunders [16] that the construction of such a basis derives from the computation of a QR decomposition of  $\tilde{H}_m$ . Indeed, if

$$Q_m \tilde{H}_m = \begin{bmatrix} R_m \\ \mathbf{0} \end{bmatrix}$$
(10)

with  $Q_m \in \mathbb{C}^{(m+1)\times(m+1)}$  unitary and  $R_m \in \mathbb{C}^{m\times m}$  upper triangular (and nonsingular since  $\tilde{H}_m$  has full rank), then

$$[\hat{V}_m \ \tilde{\boldsymbol{v}}_{m+1}] = [\hat{\boldsymbol{v}}_1 \ \cdots \ \hat{\boldsymbol{v}}_m \ \tilde{\boldsymbol{v}}_{m+1}] := V_{m+1} Q_m^{\mathrm{H}}$$
(11)

forms an orthonormal basis of  $\mathscr{V}_{m+1}$ . Moreover,

$$AC_{m} = V_{m+1}\tilde{H}_{m} = V_{m+1}Q_{m}^{\mathrm{H}}\begin{bmatrix}R_{m}\\\mathbf{0}\end{bmatrix} = [\hat{V}_{m}\ \tilde{\boldsymbol{v}}_{m+1}]\begin{bmatrix}R_{m}\\\mathbf{0}\end{bmatrix} = \hat{V}_{m}R_{m}$$
(12)

shows that  $\hat{V}_m$  constitutes a basis of  $\mathscr{W}_m = A\mathscr{C}_m$ .

On the other hand, using the QR factorization (10) the least-squares problem (9) can be rewritten as

$$\begin{split} \min_{\boldsymbol{y}\in\mathbb{C}^{m}} \|\beta\boldsymbol{u}_{1}^{(m+1)} - \tilde{H}_{m}\boldsymbol{y}\|_{2} &= \min_{\boldsymbol{y}\in\mathbb{C}^{m}} \left\| \mathcal{Q}_{m}^{H} \left( \beta \mathcal{Q}_{m}\boldsymbol{u}_{1}^{(m+1)} - \begin{bmatrix} R_{m} \\ \boldsymbol{0} \end{bmatrix} \boldsymbol{y} \right) \right\|_{2} \\ &= \min_{\boldsymbol{y}\in\mathbb{C}^{m}} \left\| \beta \mathcal{Q}_{m}\boldsymbol{u}_{1}^{(m+1)} - \begin{bmatrix} R_{m} \\ \boldsymbol{0} \end{bmatrix} \boldsymbol{y} \right\|_{2} = \min_{\boldsymbol{y}\in\mathbb{C}^{m}} \left\| \begin{bmatrix} \beta \boldsymbol{q}_{m} - R_{m}\boldsymbol{y} \\ \beta \boldsymbol{q}_{m+1,1}^{(m)} \end{bmatrix} \right\|_{2}, \end{split}$$

where  $[\boldsymbol{q}_m^{\mathrm{T}} \ \boldsymbol{q}_{m+1,1}^{(m)}]^{\mathrm{T}} = Q_m \boldsymbol{u}_1^{(m+1)} \ (\boldsymbol{q}_m \in \mathbb{C}^m)$  denotes the first column of  $Q_m$ . The unique solution of the above least-squares problem is  $\boldsymbol{y}_m^{\mathrm{MR}} = \beta R_m^{-1} \boldsymbol{q}_m$  and the associated least-squares error is given by  $\|\boldsymbol{r}_m^{\mathrm{MR}}\| = \beta |\boldsymbol{q}_{m+1,1}^{(m)}|$ .

A QR factorization of  $\tilde{H}_m$  (and simultaneously the basis  $[\hat{V}_m \ \tilde{v}_{m+1}]$ ) can be computed inductively. The matrices  $Q_m$ , m = 1, 2, ..., L - 1, are usually constructed as products of Givens rotations

$$Q_m = G_m \begin{bmatrix} Q_{m-1} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} = G_m \begin{bmatrix} G_{m-1} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} G_{m-2} & O \\ O & I_2 \end{bmatrix} \cdots \begin{bmatrix} G_1 & O \\ O & I_{m-1} \end{bmatrix}$$
(13)

where, for k = 1, 2, ..., m,

$$G_k := \begin{bmatrix} I_{k-1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & c_k & s_k e^{-i\phi_k} \\ \mathbf{0} & -s_k e^{i\phi_k} & c_k \end{bmatrix} \quad (c_k, s_k \ge 0, c_k^2 + s_k^2 = 1, \phi_k \in \mathbb{R})$$

(for the choice of  $c_k$ ,  $s_k$  and  $\phi_k$  see, e.g., [3]).

In view of (11) we have

$$\begin{bmatrix} \hat{V}_m \ \tilde{\boldsymbol{v}}_{m+1} \end{bmatrix} = V_{m+1} Q_m^{\mathrm{H}} = \begin{bmatrix} V_m \ \boldsymbol{v}_{m+1} \end{bmatrix} \begin{bmatrix} Q_{m-1}^{\mathrm{H}} \ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix} G_m^{\mathrm{H}} = \begin{bmatrix} \hat{V}_{m-1} \ \tilde{\boldsymbol{v}}_m \ \boldsymbol{v}_{m+1} \end{bmatrix} G_m^{\mathrm{H}}$$

i.e., with  $\tilde{\boldsymbol{v}}_1 = \boldsymbol{v}_1$ ,

$$\hat{\boldsymbol{v}}_m = c_m \tilde{\boldsymbol{v}}_m + s_m \mathrm{e}^{-\mathrm{i}\phi_m} \boldsymbol{v}_{m+1},$$
  
 $\tilde{\boldsymbol{v}}_{m+1} = -s_m \mathrm{e}^{\mathrm{i}\phi_m} \tilde{\boldsymbol{v}}_m + c_m \boldsymbol{v}_{m+1}$ 

for m = 1, 2, ..., L - 1. For ease of notation, we further set  $\hat{v}_L := \tilde{v}_L$ .

Expressions for the entries of the unitary matrices  $Q_m = [q_{k,j}^{(m)}]_{1 \le k,j \le m+1}$  can be obtained by forming the products of the Givens matrices in (13). For the first column, this yields

$$q_{k,1}^{(m)} = c_k \prod_{j=1}^{k-1} [-s_j e^{i\phi_j}] \quad (1 \le k \le m), \quad q_{m+1,1}^{(m)} = \prod_{j=1}^m [-s_j e^{i\phi_j}],$$

which immediately leads to the following result (cf. [3, Proposition 4.7]):

**Proposition 2.1.** For the MR and OR residual vectors of index m = 1, 2, ..., L - 1 there holds:

$$\mathbf{r}_{m}^{\text{MR}} = \beta q_{m+1,1}^{(m)} \tilde{\mathbf{v}}_{m+1} = \beta \prod_{j=1}^{m} [-s_{j} e^{i\phi_{j}}] \tilde{\mathbf{v}}_{m+1},$$
$$\mathbf{r}_{m}^{\text{OR}} = -\beta \frac{s_{m}}{c_{m}} e^{i\phi_{m}} q_{m,1}^{(m-1)} \mathbf{v}_{m+1} = \frac{\beta}{c_{m}} \prod_{j=1}^{m} [-s_{j} e^{i\phi_{j}}] \mathbf{v}_{m+1}$$
$$\mathbf{r}_{m-1}^{\text{MR}} - \mathbf{r}_{m}^{\text{OR}} = \frac{\beta}{c_{m}} q_{m,1}^{(m-1)} \hat{\mathbf{v}}_{m} = \frac{\beta}{c_{m}} \prod_{j=1}^{m-1} [-s_{j} e^{i\phi_{j}}] \hat{\mathbf{v}}_{m}.$$

Proposition 2.1 shows that the *convergence history* of an MR method (and, in essence, also of an OR method) is completely determined by the entries in the first column of the matrices  $Q_m$ . To emphasize this point we assume a finite termination index L and note that the matrix  $H_L$  possesses the QR factorization  $Q_{L-1}H_L = R_L$ . Now  $\mathbf{r}_0 \in \mathcal{W}_L$  (cf. (6)) can be represented as a linear combination of the orthonormal basis  $\{\hat{\mathbf{v}}_1, \hat{\mathbf{v}}_2, \dots, \hat{\mathbf{v}}_L\}$  of  $\mathcal{W}_L$ ,  $\mathbf{r}_0 = \beta V_L \mathbf{u}_1 = \beta V_L Q_{L-1}^H \mathbf{u}_1 = \beta \hat{V}_L Q_{L-1} \mathbf{u}_1$ , or equivalently,  $\mathbf{r}_0 = \beta \sum_{j=1}^L q_{j,1}^{(L-1)} \hat{\mathbf{v}}_j = \beta \sum_{j=1}^L q_{j,1}^{(j)} \hat{\mathbf{v}}_j$  (where we set  $q_{L,1}^{(L)} := q_{L,1}^{(L-1)}$ ). This equation states that, up to the factor  $\beta$ , the first column of the matrix  $Q_{L-1}$  contains the Fourier coefficients of the expansion of  $\mathbf{r}_0$ with respect to the basis  $\hat{V}_L$  of  $\mathcal{W}_L$ . The MR correction  $\mathbf{c}_m^{MR}$  is selected such that  $A\mathbf{c}_m^{MR}$  is the best approximation to  $\mathbf{r}_0$  from  $\mathcal{W}_m$ , i.e.,

$$Ac_{m}^{\text{MR}} = \beta \sum_{j=1}^{m} q_{j,1}^{(j)} \hat{v}_{j}$$
 and  $r_{m}^{\text{MR}} = r_{0} - Ac_{m}^{\text{MR}} = \beta \sum_{j=m+1}^{L} q_{j,1}^{(j)} \hat{v}_{j}.$ 

#### 2.2. The angle connection

We saw in Proposition 2.1 that the sines and cosines of the Givens rotations used to construct the QR decomposition of  $\tilde{H}_m$  completely determine the residuals of both the MR and the OR approach. In this section, we recall that these sines and cosines are not merely artifacts of the computational scheme but are the sines and cosines of the angles between  $\mathcal{W}_m$  and  $\mathcal{V}_m$ , i.e., between the *m*th approximation and the *m*th residual space.

By

$$\varphi_m := \angle (\mathbf{r}_{m-1}^{\mathrm{MR}}, \mathscr{A}\mathscr{C}_m) = \angle (\mathbf{r}_{m-1}^{\mathrm{MR}}, \mathscr{W}_m) \quad (m = 1, 2, \dots, L),$$

we denote the angle between  $\mathbf{r}_{m-1}^{\text{MR}}$  and  $A\mathscr{C}_m = \mathscr{W}_m$ .<sup>2</sup> Note that  $0 < \varphi_m \leq \pi/2$  for m = 1, 2, ..., L-1, but  $\varphi_L = 0$  because  $\mathbf{r}_{L-1}^{\text{MR}} \in \mathscr{V}_L = \mathscr{W}_L$ .

The following relations are fundamental for our later investigations (for a proof, see [3, Section 2]).

**Theorem 2.2.** For m = 2, 3, ..., L, there holds

$$\sin \varphi_m = \frac{\sin \angle (\mathbf{r}_0, \mathscr{W}_m)}{\sin \angle (\mathbf{r}_0, \mathscr{W}_{m-1})} = \sin \angle (\mathscr{V}_m, \mathscr{W}_m)$$

where  $\angle(\mathscr{V}_m, \mathscr{W}_m)$  denotes the largest canonical angle between the spaces  $\mathscr{V}_m$  and  $\mathscr{W}_m$ .<sup>3</sup> For the case of m = 1, we have  $\mathscr{V}_1 = \operatorname{span}\{\mathbf{r}_0\}$  and thus  $\sin \varphi_1 = \sin \angle(\mathscr{V}_1, \mathscr{W}_1)$ . In addition, there holds

 $\sin \angle (\mathbf{r}_0, \mathscr{W}_m) = \sin \varphi_1 \sin \varphi_2 \cdots \sin \varphi_m \quad (m = 1, 2, \dots, L).$ 

Moreover, the quantities  $c_m$  and  $s_m$  which define the Givens rotations  $G_m$  of (13) are given by

 $c_m = \cos \varphi_m$  and  $s_m = \sin \varphi_m$  (m = 1, 2, ..., L - 1).

As a consequence of these assertions, we cite from [3, Section 3] how the vectors involved in the MR and OR approximations are related.

**Theorem 2.3.** With  $s_m = \sin \angle (\mathbf{r}_{m-1}^{MR}, A\mathscr{C}_m)$  and  $c_m = \cos \angle (\mathbf{r}_{m-1}^{MR}, A\mathscr{C}_m)$  the MR and OR approximations with respect to the correction spaces  $\mathscr{C}_m$  satisfy

$$\|\boldsymbol{r}_{m}^{\mathrm{MR}}\| = s_{m} \|\boldsymbol{r}_{m-1}^{\mathrm{MR}}\| = s_{1} s_{2} \dots s_{m} \|\boldsymbol{r}_{0}\|,$$
(14)

$$\|\boldsymbol{r}_{m}^{\mathrm{MR}}\| = c_{m} \|\boldsymbol{r}_{m}^{\mathrm{OR}}\|,\tag{15}$$

$$\boldsymbol{x}_{m}^{\text{MR}} = s_{m}^{2} \boldsymbol{x}_{m-1}^{\text{MR}} + c_{m}^{2} \boldsymbol{x}_{m}^{\text{OR}} \quad and \quad \boldsymbol{r}_{m}^{\text{MR}} = s_{m}^{2} \boldsymbol{r}_{m-1}^{\text{MR}} + c_{m}^{2} \boldsymbol{r}_{m}^{\text{OR}}.$$
(16)

<sup>2</sup> For the reader's convenience, we recall that the angle between a nonzero vector  $\mathbf{x} \in \mathcal{H}$  and a subspace  $\mathcal{U} \subset \mathcal{H}$ ,  $\mathcal{U} \neq \{\mathbf{0}\}$ , is defined by

$$\cos \angle (\mathbf{x}, \mathscr{U}) = \sup_{\mathbf{0} \neq \mathbf{u} \in \mathscr{U}} \cos \angle (\mathbf{x}, \mathbf{u}) = \sup_{\mathbf{0} \neq \mathbf{u} \in \mathscr{U}} \frac{|(\mathbf{x}, \mathbf{u})|}{\|\mathbf{x}\| \|\mathbf{u}\|}.$$

If  $\mathscr{U}$  is finite dimensional this angle is also given by  $\cos \angle (\mathbf{x}, \mathscr{U}) = ||P_{\mathscr{U}}\mathbf{x}||/||\mathbf{x}||$ , where  $P_{\mathscr{U}}$  denotes the orthogonal projection onto  $\mathscr{U}$ , and consequently,  $\angle (\mathbf{x}, \mathscr{U}) = 0$  if and only if  $\mathbf{x} \in \mathscr{U}$ . Moreover,  $\sin \angle (\mathbf{x}, \mathscr{U}) = ||(I - P_{\mathscr{U}})\mathbf{x}||/||\mathbf{x}||$ , and consequently,  $\angle (\mathbf{x}, \mathscr{U}) = \pi/2$  if and only if  $\mathbf{x} \perp \mathscr{U}$ .

<sup>3</sup>Given orthonormal bases  $\{\mathbf{v}_j\}_{j=1}^m$  and  $\{\mathbf{w}_j\}_{j=1}^m$  of two *m*-dimensional subspaces  $\mathscr{V}$  and  $\mathscr{W}$ , then the cosines of the canonical angles between  $\mathscr{V}$  and  $\mathscr{W}$  are the singular values of the matrix of inner products  $[(\mathbf{v}_j, \mathbf{w}_k)] \in \mathbb{C}^{m \times m}$ . For later use, we remark that the sine of the largest canonical angle between the spaces  $\mathscr{V}$  and  $\mathscr{W}$  of equal dimension is given by  $\|(I - P_{\mathscr{V}})P_{\mathscr{W}}\|$  (cf. [23, Theorem 4.37]).

For later use, we mention another important relation

$$\|\boldsymbol{r}_{m}^{\mathrm{MR}}\| = \sin \angle (\boldsymbol{r}_{0}, A\mathscr{C}_{m}) \|\boldsymbol{r}_{0}\|, \tag{17}$$

which follows from (14) and Theorem 2.2.

In view of  $s_m = \|\mathbf{r}_m^{\text{MR}}\| / \|\mathbf{r}_{m-1}^{\text{MR}}\|$ , i.e.,  $c_m = \sqrt{1 - \|\mathbf{r}_m^{\text{MR}}\|^2 / \|\mathbf{r}_{m-1}^{\text{MR}}\|^2}$ , (15) and (16) are easily rewritten as

$$\|\boldsymbol{r}_{m}^{\text{MR}}\| = \sqrt{1 - \frac{\|\boldsymbol{r}_{m}^{\text{MR}}\|^{2}}{\|\boldsymbol{r}_{m-1}^{\text{MR}}\|^{2}} \|\boldsymbol{r}_{m}^{\text{OR}}\|,}$$
$$\boldsymbol{x}_{m}^{\text{OR}} = \frac{\|\boldsymbol{r}_{m-1}^{\text{MR}}\|^{2}}{\|\boldsymbol{r}_{m-1}^{\text{MR}}\|^{2} - \|\boldsymbol{r}_{m}^{\text{MR}}\|^{2}} \boldsymbol{x}_{m}^{\text{MR}} - \frac{\|\boldsymbol{r}_{m}^{\text{MR}}\|^{2}}{\|\boldsymbol{r}_{m-1}^{\text{MR}}\|^{2} - \|\boldsymbol{r}_{m}^{\text{MR}}\|^{2}} \boldsymbol{x}_{m-1}^{\text{MR}}.$$

As can be seen from the last two equations, the OR approximation and residual can easily be computed from the corresponding MR quantities. Moreover, since the latter can always be computed in a stable fashion, this is the preferable way of computing these quantities. (An exception is, of course, the Hermitian positive-definite case, in which the OR quantities may be computed stably and at lower expense than their MR counterparts by the classical method of conjugate gradients.)

# 2.3. Multiple subspace correction

Various recently developed enhancements of the basic MR and OR schemes presented above are based on introducing additional subspace corrections aside from those associated with the stepwise increasing correction spaces. Existing approaches include generating such auxiliary projections from spectral information on the operator A gained during the iteration process or from additional inner iteration or restart cycles. In addition, time and storage constraints often make it necessary to form these projections only approximately, while at the same time keeping this approximation as effective as possible. To better describe and compare these new developments, we first formulate the basic projection steps required to combine two subspace corrections and then, in Section 2.4, discuss how subspace information may be quantified in order to construct effective approximate projections.

Consider an initial approximation  $x_0$  to the solution of (1) for which we seek the MR approximation  $x_0 + c$  with c selected from the correction space  $\mathscr{C}$ . We assume  $\mathscr{C}$  to be the direct sum  $\mathscr{C} = \mathscr{C}_1 \oplus \mathscr{C}_2$  of two spaces  $\mathscr{C}_1$  and  $\mathscr{C}_2$ , and our goal is to obtain the MR approximation as the result of two separate projection steps involving  $\mathscr{C}_1$  and  $\mathscr{C}_2$ , respectively. This task is equivalent to finding the best approximation  $w = Ac \in \mathscr{W} = A\mathscr{C} = \mathscr{W}_1 \oplus \mathscr{W}_2$  to  $r_0$ , where  $\mathscr{W}_j := A\mathscr{C}_j$ , j = 1, 2.

If, in a first step, we obtain the best approximation  $w_1 = P_{\mathcal{W}_1} r_0$  in  $\mathcal{W}_1$ , then the best approximation in  $\mathcal{W}$  is obtained by introducing the orthogonal complement  $\mathcal{Z} := \mathcal{W} \cap \mathcal{W}_1^{\perp}$  of  $\mathcal{W}_1$  in  $\mathcal{W}$ , in terms of which  $\mathcal{W}$  has the direct *and orthogonal* decomposition  $\mathcal{W} = \mathcal{W}_1 \oplus \mathcal{Z}$ . The global best approximation is now given by

$$w := P_{\mathcal{H}} r_0 = (P_{\mathcal{H}_1} + P_{\mathcal{X}}) r_0 = P_{\mathcal{H}_1} r_0 + P_{\mathcal{X}} (I - P_{\mathcal{H}_1}) r_0.$$
<sup>(18)</sup>

The last expression shows that the contribution from the second projection consists of the orthogonal projection onto  $\mathscr{Z}$  of the error  $(I - P_{\mathscr{W}_1})\mathbf{r}_0$  of the first approximation.

268

Expressing all spaces in terms of  $\mathscr{C}_1$  and  $\mathscr{C}_2$  and noting that  $\mathscr{Z} = (I - P_{A\mathscr{C}_1})A\mathscr{C}_2$ , we conclude that the correction *c* associated with the residual approximation *w* satisfies

$$A\boldsymbol{c} = \boldsymbol{w} = P_{A\mathscr{C}_1}\boldsymbol{r}_0 + P_{(I-P_{A\mathscr{C}_1})A\mathscr{C}_2}(I-P_{A\mathscr{C}_1})\boldsymbol{r}_0.$$

(1)

The global correction is thus of the form  $c = c_1 + d$ , where

$$A\boldsymbol{c}_1 = P_{\mathcal{A}\mathcal{C}_1}\boldsymbol{r}_0,\tag{19}$$

$$Ad = P_{(I - P_{A\mathscr{C}_1})A\mathscr{C}_2}(I - P_{A\mathscr{C}_1})\mathbf{r}_0.$$
 (20)

The solution  $c_1$  of (19) is simply the MR solution of the equation  $Ac = r_0$  with respect to the correction space  $\mathscr{C}_1$ . To obtain a useful representation of d, we note that the right-hand side of (20) may be viewed as the MR approximation with respect to  $\mathscr{C}_2$  of the equation

$$(I - P_{A\mathscr{C}_1})A\boldsymbol{c} = (I - P_{A\mathscr{C}_1})\boldsymbol{r}_0.$$
(21)

**Lemma 2.4.** The operator  $(I - P_{A\mathscr{C}_1})A$  restricted to  $\mathscr{C}_2$  is a bijection from  $\mathscr{C}_2$  to  $\mathscr{Z}$ .

**Proof.** The assertion follows by showing that the operator in question is one-to-one:  $(I - P_{A\mathscr{C}_1})A\tilde{c} = \mathbf{0}$  for  $\tilde{c} \in \mathscr{C}_2$  implies  $A\tilde{c} \in A\mathscr{C}_1 \cap A\mathscr{C}_2 = \{\mathbf{0}\}$ .  $\Box$ 

The solution d of (20) yielding the second component of the combined correction c may thus be obtained by first determining the MR solution  $c_2$  of (21) and then evaluating

$$d = A^{-1}(I - P_{A\mathscr{C}_1})Ac_2 = c_2 - A^{-1}P_{A\mathscr{C}_1}Ac_2.$$
(22)

**Lemma 2.5.** The operator  $P := A^{-1}(I - P_{A\mathscr{C}_1})A$  restricted to  $\mathscr{C}_2$  is the oblique projection onto  $A^{-1}\mathscr{Z}$  along  $\mathscr{C}_1$ .

**Proof.** The projection property follows immediately upon squaring *P*. Since *A* is nonsingular, null(*P*)= $A^{-1}\mathcal{W}_1 = \mathcal{C}_1$  and range(*P*) =  $A^{-1}(A\mathcal{C}_1)^{\perp}$ . Restricted to  $\mathcal{C}_2$ , the range reduces to the preimage under *A* of the orthogonal complement of  $A\mathcal{C}_1$  with respect to  $A\mathcal{C}_2$ , i.e.,  $A^{-1}\mathcal{Z}$ .  $\Box$ 

At first glance, the evaluation of d as given in (22) appears to require a multiplication by A as well as the solution of another equation involving A with a right-hand side from  $A\mathcal{C}_1$ , in addition to the computation of the two projections. In fact, we show how d can be calculated inexpensively using quantities generated in the course of the two MR approximation steps.

Assume  $\mathscr{C}_1$  has dimension m and that  $C_m^{(1)} = [c_1^{(1)} \cdots c_m^{(1)}]$  denotes a basis of  $\mathscr{C}_1$ . The MR approximation  $c_1$  has the coordinate representation  $c_1 = C_m^{(1)} y_1$  with  $y_1 \in \mathbb{C}^m$ . We write the associated Arnoldi-type decomposition (8) as  $AC_m^{(1)} = V_{m+1}^{(1)} \tilde{H}_m^{(1)}$ . The QR decomposition  $Q_m^{(1)} \tilde{H}_m^{(1)} = R_m^{(1)}$  (cf. (10)) makes available the Paige–Saunders basis  $\hat{V}_m^{(1)}$  (cf. (11)), which forms an orthonormal basis of  $A\mathscr{C}_1$ . Note also that, in view of relation (12), there holds

$$A^{-1}\hat{V}_m^{(1)} = C_m^{(1)}R_m^{-1}.$$
(23)

The orthogonal projection  $P_{\mathcal{A}\mathscr{C}_1}$  may be expressed in terms of  $\hat{V}_m^{(1)}$  as  $\hat{V}_m^{(1)}[\hat{V}_m^{(1)}]^*$ , (for  $V = [\boldsymbol{v}_1 \cdots \boldsymbol{v}_m]$ ,  $W = [\boldsymbol{w}_1 \cdots \boldsymbol{w}_m]$ , we denote by  $VW^*$  the linear operator  $\boldsymbol{x} \mapsto \sum_{j=1}^m (\boldsymbol{x}, \boldsymbol{w}_j) \boldsymbol{v}_j$ ) and, denoting the residual

of the first MR approximation by  $r_1 := r_0 - Ac_1$ , Eq. (21) may be written

$$(I - \hat{V}_m^{(1)} [\hat{V}_m^{(1)}]^*) A c = r_1.$$

The Arnoldi-type decomposition associated with Eq. (21) in terms of the basis  $C_k^{(2)} = [c_1^{(2)} \cdots c_k^{(2)}]$  of the correction space  $\mathscr{C}_2$  is given by

$$(I - \hat{V}_m^{(1)} [\hat{V}_m^{(1)}]^*) A C_k^{(2)} = V_{k+1}^{(2)} \tilde{H}_k^{(2)}$$
(24)

with the associated MR approximation  $c_2 = C_k^{(2)} y_2$ , for some  $y_2 \in \mathbb{C}^k$ . The solution d of (20) as given in (22) can now be expressed as

$$d = c_2 - A^{-1} P_{A \mathscr{C}_1} A c_2 = C_k^{(2)} y_2 - A^{-1} \hat{V}_m^{(1)} [\hat{V}_m^{(1)}]^* A C_k^{(2)} y_2$$
  
=  $C_k^{(2)} y_2 - C_m^{(1)} [R_m^{(1)}]^{-1} ([\hat{V}_m^{(1)}]^* A C_k^{(2)}) y_2,$ 

which shows that the action of  $A^{-1}$  in (22) is effected by the inverse of the (small) triangular matrix  $R_m^{(1)}$ . We further observe that the evaluation of  $Ac_2$  in (22) is accomplished through the  $m \times k$  matrix  $[\hat{V}_m^{(1)}]^* AC_k^{(2)}$ , which is available at no extra cost as a by-product of the orthogonalization process carried out in the second MR step to obtain (24). In fact, (23) and (24) can be combined to yield the global decomposition

$$A[C_m^{(1)} C_k^{(2)}] = [\hat{V}_m^{(1)} V_{k+1}^{(2)}] \begin{bmatrix} R_m^{(1)} & [\hat{V}_m^{(1)}]^* A C_k^{(2)} \\ O & \tilde{H}_k^{(2)} \end{bmatrix}$$
(25)

with respect to  $\mathscr{C}$ . We summarize the coordinate representation of these two successive projections as

**Theorem 2.6.** The MR approximation of the solution of  $A\mathbf{c} = \mathbf{r}_0$  with respect to the correction space  $\mathscr{C} = \mathscr{C}_1 \oplus \mathscr{C}_2$  is given by

$$\boldsymbol{c} = C_m^{(1)} \boldsymbol{y}_1 + C_k^{(2)} \boldsymbol{y}_2 + C_m^{(1)} [R_m^{(1)}]^{-1} ([\hat{V}_m^{(1)}]^* A C_k^{(2)}) \boldsymbol{y}_2$$

where the coefficient vectors  $y_1 \in \mathbb{C}^m$  and  $y_2 \in \mathbb{C}^k$  solve the least-squares problems

$$\| \| \boldsymbol{r}_0 \| \boldsymbol{u}_1^{(m+1)} - \tilde{H}_m^{(1)} \boldsymbol{y}_1 \|_2 \to \min_{\boldsymbol{y}_1 \in \mathbb{C}^m}, \quad \| \| \boldsymbol{r}_1 \| \boldsymbol{u}_1^{(k+1)} - \tilde{H}_k^{(2)} \boldsymbol{y}_2 \|_2 \to \min_{\boldsymbol{y}_2 \in \mathbb{C}^k}$$

and the matrices  $C_m^{(1)}$ ,  $C_k^{(2)}$ ,  $\hat{V}_m^{(1)}$ ,  $R_m^{(1)}$ ,  $\tilde{H}_m^{(1)}$ , and  $\tilde{H}_k^{(2)}$  as well as the vector  $\mathbf{r}_1$  are defined above.

# 2.4. Incomplete orthogonalization

The MR approximation applied to Eq. (21) in effect maintains orthogonality of the basis vectors of the residual space  $\mathscr{V}_2$  against  $\mathscr{W}_1 = A\mathscr{C}_1$ . Computationally, this is manifested in the generation of the  $m \times k$  matrix  $(\hat{V}_m^{(1)})^* A C_k^{(2)}$  during the orthonormalization process (cf. (25)). In order to reduce the cost of both the storage of  $\hat{V}_m^{(1)}$  and the work involved in the orthogonalization, we now consider performing the MR approximation to the solution of (21) only approximately in the sense that orthogonality is maintained only against a subspace of  $\mathscr{W}_1$  of fixed dimension. When faced with the choice of such a subspace against which one can afford to maintain orthogonality, one possible

270

271

criterion is to select that space which results in the greatest reduction of the residual norm after the second MR approximation. Such an approach was proposed by de Sturler [25], and will be further described in Section 4.5.

As in Section 2.3, consider the MR approximation with respect to the correction space  $\mathscr{C} = \mathscr{C}_1 \oplus \mathscr{C}_2$ . The global MR approximation (18) consists of an MR approximation with respect to  $\mathscr{C}_1$  followed by a second projection involving the orthogonal complement  $\mathscr{Z} := (I - P_{\mathscr{W}_1})\mathscr{W}_2$  of  $\mathscr{W}_1 = A\mathscr{C}_1$  with respect to  $\mathscr{W}_2 = A\mathscr{C}_2$ . The simplest approach of completely omitting the orthogonalization involved in constructing  $P_{\mathscr{Z}}$  results in the combined approximation

$$\tilde{\boldsymbol{w}} := P_{\mathscr{W}_1} \boldsymbol{r}_0 + P_{\mathscr{W}_2} (I - P_{\mathscr{W}_1}) \boldsymbol{r}_0$$

in place of (18). This is the standard way of restarting an MR algorithm. Besides the two extremes of complete orthogonalization against  $\mathscr{W}_1$  or none at all, it is also possible to orthogonalize against only a subspace  $\widetilde{\mathscr{W}}_1 \subset \mathscr{W}_1$  of dimension  $\ell < m$ , which brings up the problem of determining  $\widetilde{\mathscr{W}}_1$  such that, if orthogonality of the residual space  $\mathscr{V}_2$  of the second MR approximation is maintained against  $\widetilde{\mathscr{W}}_1$ , this results in the smallest residual norm over all  $\ell$ -dimensional subspaces of  $\mathscr{W}_1$ .

The solution of this problem is greatly facilitated by a judicious choice of bases: Let  $W_m^{(1)} = [w_1^{(1)} \cdots w_m^{(1)}]$  and  $W_k^{(2)} = [w_1^{(2)} \cdots w_k^{(2)}]$  denote biorthogonal orthonormal bases of  $\mathcal{W}_1$  and  $\mathcal{W}_2$  ordered such that the (diagonal)  $m \times k$  matrix  $\Gamma := [W_m^{(1)}]^* W_k^{(2)}$  has nonincreasing nonnegative entries  $\gamma_1, \ldots, \gamma_{\min\{m,k\}}$ . The numbers  $\gamma_j$  are the cosines of the canonical angles between the spaces  $\mathcal{W}_1$  and  $\mathcal{W}_2$  (cf. [23, Chapter 4.5]) and therefore lie between zero and one. In addition, the assumption  $\mathcal{C}_1 \cap \mathcal{C}_2 = \{\mathbf{0}\}$  along with the nonsingularity of A implies  $\mathcal{W}_1 \cap \mathcal{W}_2 = \{\mathbf{0}\}$  and therefore each  $\gamma_j$  is strictly less than one.

An orthogonal basis of  $\mathscr{Z}$  is given by  $\hat{Z}_k := (I - W_m^{(1)} [W_m^{(1)}]^*) W_k^{(2)}$ , and we set  $\hat{Z}_k^* \hat{Z}_k = I - \Gamma^H \Gamma =:$  $\Sigma^2 \in \mathbb{C}^{k \times k}$ , where  $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_k)$  with

$$\sigma_j = \begin{cases} \sqrt{1 - \gamma_j^2}, & 1 \leq j \leq \min\{k, m\}, \\ 1, & \text{otherwise,} \end{cases}$$

in view of which  $Z_k := \hat{Z}_k \Sigma^{-1}$  is an orthonormal basis of  $\mathscr{Z}$ . Denoting  $Z_k = [z_1 \cdots z_k]$ , the following theorem expresses the effect of complete orthogonalization versus none at all:

**Theorem 2.7.** In the notation introduced above and with  $\mathbf{r}_1 := (I - P_{\mathscr{W}_1})\mathbf{r}_0$ , there holds

$$(P_{\mathscr{W}_2} - P_{\mathscr{Z}})\mathbf{r}_1 = \sum_{j=1}^{\min\{k,m\}} (\mathbf{r}_1, \mathbf{z}_j) \gamma_j (\sigma_j \mathbf{w}_j^{(1)} - \gamma_j \mathbf{z}_j),$$
(26)

$$\|(P_{\mathscr{W}_{2}}-P_{\mathscr{Z}})\mathbf{r}_{1}\|^{2} = \sum_{j=1}^{\min\{k,m\}} \gamma_{j}^{2} |(\mathbf{r}_{1},\mathbf{z}_{j})|^{2}.$$
(27)

**Proof.** Taking note of  $\mathbf{r}_1 \perp \mathcal{W}_1$  and  $W_k^{(2)} = Z_k \Sigma + W_m^{(1)} \Gamma$ , we obtain

$$(P_{\mathscr{W}_{2}} - P_{\mathscr{X}})\mathbf{r}_{1} = (W_{k}^{(2)}[W_{k}^{(2)}]^{*} - Z_{k}Z_{k}^{*})\mathbf{r}_{1}$$
  
=  $((Z_{k}\Sigma + W_{m}^{(1)}\Gamma)(Z_{k}\Sigma + W_{m}^{(1)}\Gamma)^{*} - Z_{k}Z_{k}^{*})\mathbf{r}_{1}$   
=  $(W_{m}^{(1)}\Gamma\Sigma - Z_{k}\Gamma^{\mathrm{H}}\Gamma)Z_{k}^{*}\mathbf{r}_{1},$ 

which is a reformulation of (26). Taking norms and noting  $\mathcal{W}_1 \perp \mathcal{Z}$  as well as  $\gamma_j^2 + \sigma_j^2 = 1$  yields (27):

$$\|(P_{\mathscr{W}_{2}}-P_{\mathscr{Z}})\mathbf{r}_{1}\|^{2}=(Z_{k}^{*}\mathbf{r}_{1})^{*}(\Sigma\Gamma^{\mathrm{H}}\Gamma\Sigma+(\Gamma^{\mathrm{H}}\Gamma)^{2})(Z_{k}^{*}\mathbf{r}_{1})=\|\Gamma Z_{k}^{*}\mathbf{r}_{1}\|_{2}^{2}.$$

We see that the difference between the two projections depends on the  $\mathscr{Z}$ -components of the approximation error  $\mathbf{r}_1$  remaining after the first projection weighted by the corresponding cosines  $\gamma_j$  of the canonical angles between  $\mathscr{W}_1$  and  $\mathscr{W}_2$ . Whenever  $\gamma_j = 0$ , the projection onto  $\mathscr{W}_2$  would already have produced the correct component in the direction  $\mathbf{z}_j$ , whereas in case  $\gamma_j = 1$  the associated basis vectors  $\mathbf{w}_i^{(1)}$  and  $\mathbf{w}_i^{(2)}$  are collinear and  $P_{\mathscr{W}_2}$  would have yielded no component in direction  $\mathbf{z}_j$ .

To consider the case of incomplete orthogonalization, let  $\tilde{\mathcal{W}}_1 \subset \mathcal{W}_1$  with dim  $\tilde{\mathcal{W}}_1 = \ell < m$ . By orthogonalizing the basis of  $\mathcal{W}_2$  against  $\tilde{\mathcal{W}}_1$ , we construct the orthogonal projection onto  $\tilde{\mathcal{Z}} := (\tilde{\mathcal{W}}_1 \oplus \mathcal{W}_2) \cap \tilde{\mathcal{W}}_1^{\perp}$ , which, applied to  $\mathbf{r}_1$ , yields the difference between  $P_{\mathcal{W}_1}\mathbf{r}_0$  and the best approximation of  $\mathbf{r}_0$  in  $\tilde{\mathcal{W}}_1 \oplus \mathcal{W}_2$ .

**Theorem 2.8.** Of all  $\ell$ -dimensional subspaces  $\tilde{\mathcal{W}}_1 \subset \mathcal{W}_1$ , that which minimizes  $||(P_{\tilde{\mathscr{Z}}} - P_{\mathscr{Z}})(I - P_{\mathscr{W}_1})\mathbf{r}_0||$ over all  $\mathbf{r}_0 \in \mathscr{H}$  is given by  $\tilde{\mathscr{W}}_1 = \operatorname{span}\{\mathbf{w}_1^{(1)}, \dots, \mathbf{w}_\ell^{(1)}\}$ , and results in

$$\|(P_{\tilde{\mathscr{Z}}}-P_{\mathscr{Z}})(I-P_{\mathscr{W}_{1}})\mathbf{r}_{0}\|=\|(P_{\tilde{\mathscr{Z}}}-P_{\mathscr{Z}})\mathbf{r}_{1}\|=\sum_{j=\ell+1}^{\min\{k,m\}}\gamma_{j}^{2}|(\mathbf{r}_{1},\mathbf{z}_{j})|^{2}.$$

**Proof.** Any orthonormal basis  $\tilde{W}_{\ell}^{(1)} = [\tilde{w}_1 \cdots \tilde{w}_{\ell}]$  of  $\tilde{\mathscr{W}}_1$  has the form  $\tilde{W}_{\ell}^{(1)} = W_m^{(1)}Q_1$  with a matrix  $Q_1 \in \mathbb{C}^{m \times \ell}$  consisting of the first  $\ell$  columns of a unitary  $m \times m$  matrix  $Q = [Q_1 Q_2]$ . We obtain a basis of  $\tilde{\mathscr{Z}}$  by orthogonalizing  $W_k^{(2)}$  against  $\tilde{W}_{\ell}^{(1)}$ :

$$\begin{split} \hat{Z}_{\ell} &:= (I - \tilde{W}_{\ell}^{(1)} [\tilde{W}_{\ell}^{(1)}]^*) W_k^{(2)} = W_k^{(2)} - W_m^{(1)} Q_1 Q_1^{\mathrm{H}} \Gamma \\ &= (Z_k \Sigma + W_k^{(2)} \Gamma) - W_m^{(1)} Q_1 Q_1^{\mathrm{H}} \Gamma = Z_k \Sigma + W_m^{(1)} (I - Q_1 Q_1^{\mathrm{H}}) \Gamma \\ &= Z_k \Sigma + W_m^{(1)} Q_2 Q_2^{\mathrm{H}} \Gamma. \end{split}$$

Because of  $0 \leq \gamma_i < 1$  the Hermitian matrix

$$\hat{Z}_{\ell}^{*}\hat{Z}_{\ell} = \Sigma^{2} + \Gamma^{\mathrm{H}}Q_{2}Q_{2}^{\mathrm{H}}\Gamma = I - \Gamma^{\mathrm{H}}Q_{1}Q_{1}^{\mathrm{H}}\Gamma =: S^{2}$$

is positive definite and therefore possesses a square root S, by means of which we obtain an orthonormal basis of  $\tilde{\mathscr{Z}}$  as  $Z_{\ell} := \hat{Z}_{\ell} S^{-1}$ . Again recalling  $r_1 \perp \mathscr{W}_1$ , we obtain for the difference of the two projections

$$(P_{\tilde{\mathscr{Z}}} - P_{\mathscr{Z}})\mathbf{r}_{1} = (Z_{\ell}Z_{\ell}^{*} - Z_{k}Z_{k}^{*})\mathbf{r}_{1} = (Z_{k}(\Sigma S^{-2}\Sigma - I) + W_{m}^{(1)}(Q_{2}Q_{2}^{H}\Gamma S^{-2}\Sigma))Z_{k}^{*}\mathbf{r}_{1}.$$
(28)

From the definition of  $S^2$ , we have

$$\Sigma S^{-2} \Sigma = (\Sigma^{-1} S^2 \Sigma^{-1})^{-1} = (I + \Sigma^{-1} \Gamma^{\mathrm{H}} Q_2 Q_2^{\mathrm{H}} \Gamma \Sigma^{-1})^{-1} = : (I + M M^{\mathrm{H}})^{-1}$$

with  $M = \Sigma^{-1} \Gamma^{\mathrm{H}} Q_2$ . We thus obtain

$$\Sigma S^{-2}\Sigma - I = (I + MM^{H})^{-1} - I = -MM^{H}(I + MM^{H})^{-1}$$

as well as  $Q_2 Q_2^{\rm H} \Gamma S^{-2} \Sigma = Q_2 M^{\rm H} (I + M M^{\rm H})^{-1}$ , which we insert in (18) to obtain

$$\|(P_{\tilde{\mathscr{Z}}} - P_{\mathscr{Z}})\mathbf{r}_{1}\|^{2} = (Z_{k}^{*}\mathbf{r}_{1})[(I + MM^{\mathrm{H}})^{-1}MM^{\mathrm{H}}](Z_{k}^{*}\mathbf{r}_{1}).$$
<sup>(29)</sup>

This expression is minimized for all  $\mathbf{r}_1$  – hence also for all  $\mathbf{r}_0$  – by choosing  $Q_1$  to minimize the largest eigenvalue of the Hermitian matrix  $(I + MM^{\rm H})^{-1}MM^{\rm H}$  or, equivalently, that of  $MM^{\rm H} = \Sigma^{-1}\Gamma^{\rm H}(I - Q_1Q_1^{\rm H})\Gamma\Sigma^{-1}$ . The entries  $\gamma_j/\sigma_j$  of the  $m \times k$  diagonal matrix  $\Gamma\Sigma^{-1}$  are nonincreasing, hence the minimum occurs for

$$Q_1 = \begin{bmatrix} I_\ell \\ O \end{bmatrix}$$

and the assertion follows by inserting the resulting choice of M in (29).  $\Box$ 

#### 3. Corrections selected from Krylov spaces

The overwhelming majority of subspace correction methods for solving linear systems of equations employ correction spaces of a particularly simple structure known as *Krylov spaces* (or *Krylov subspaces*), which are defined by

$$\mathscr{K}_m := \mathscr{K}_m(A, \mathbf{r}_0) := \operatorname{span}\{\mathbf{r}_0, A\mathbf{r}_0, \dots, A^{m-1}\mathbf{r}_0\}.$$
(30)

In this section we survey some of the ramifications of this choice. Section 3.1 discusses the advantages of using Krylov spaces, recalls their description in terms of polynomial spaces and states some technical lemmata. In Sections 3.2 and 3.3 we derive the polynomial counterparts of the OR and MR residual vectors and express their zeros as Ritz and harmonic Ritz values of A, respectively. Finally, we describe the implicitly restarted Arnoldi process of Sorensen [22] for later use as a technique for manipulating Krylov spaces.

### 3.1. Why Krylov subspaces?

One regard in which (30) is a reasonable choice for a correction space is that it enables the successive generation of the sequence  $\{\mathscr{C}_m\}$  using only matrix-vector multiplication by A, an operation which is inexpensive for sparse or structured matrices. Moreover, note that  $\mathscr{C}_m = \mathscr{K}_m(A, \mathbf{r}_0)$  results in the residual space (cf. (4))

$$\mathscr{V}_{m+1} = \operatorname{span} \{ \mathbf{r}_0 \} + A\mathscr{C}_m = \operatorname{span} \{ \mathbf{r}_0 \} + A\mathscr{K}_m = \mathscr{K}_{m+1},$$

i.e., the residual space  $\mathscr{V}_{m+1}$  of index m+1 coincides with the correction space  $\mathscr{C}_{m+1}$  of the next iteration, obviating the need to store two separate bases. This effectively halves the storage requirements of algorithms which are based on orthonormal bases of the residual spaces. As another consequence, the Arnoldi-type decomposition (8) now becomes a proper Arnoldi decomposition

$$AV_m = V_{m+1}\tilde{H}_m = V_mH_m + \eta_{m+1,m}\boldsymbol{v}_{m+1}\boldsymbol{u}_m^{\mathrm{T}}$$

which identifies  $H_m$  as the orthogonal section of A onto  $\mathscr{K}_m$ , i.e., it represents the linear map  $A_{\mathscr{K}_m} := P_{\mathscr{K}_m} A|_{\mathscr{K}_m} : \mathscr{K}_m \to \mathscr{K}_m$  with respect to the basis  $V_m$ .

Whether or not Krylov spaces are well suited as correction spaces will, as shown before, depend on the behavior of the angles  $\angle (\mathscr{K}_m, A\mathscr{K}_m)$  as *m* approaches  $\infty$ . There are classes of problems for which this behavior is very favorable. An example where the angles actually tend to zero, which, in view of (14), implies superlinear convergence of the MR and OR approximants, is given by second-kind Fredholm equations (cf. [3, Theorem 6.12]). On the other hand, there are matrix problems of dimension *n* for which  $\angle (\mathscr{K}_m, A\mathscr{K}_m) = \pi/2$  (m = 1, 2, ..., n-1), i.e., no Krylov subspace method is able to improve the initial residual until the very last step.

Finally, the theoretical investigation of Krylov subspace methods is greatly facilitated by the intimate connection between a Krylov space and an associated space of polynomials, as can be seen from the representation

$$\mathscr{K}_m(A,\mathbf{r}_0) = \{q(A)\mathbf{r}_0 : q \in \mathscr{P}_{m-1}\} \quad (m = 1, 2, \ldots),$$

where  $\mathscr{P}_m$  denotes the space of all complex polynomials of degree at most *m*. The linear map

$$\mathscr{P}_{m-1} \ni q \mapsto q(A)\mathbf{r}_0 \in \mathscr{K}_m(A,\mathbf{r}_0)$$

is thus always surjective, but fails to be an isomorphism if and only if there exists a nonzero polynomial  $q \in \mathscr{P}_{m-1}$  with  $q(A)\mathbf{r}_0 = \mathbf{0}$ . If such a polynomial exists (e.g., if A has finite rank) then there also exists a (unique) monic polynomial  $c = c_{A,r_0}$  of minimal degree for which  $c(A)\mathbf{r}_0 = \mathbf{0}$ , which is usually called the *minimal polynomial of*  $\mathbf{r}_0$  with respect to A. It is easy to see that the degree of c equals the smallest integer m for which  $\mathscr{K}_m = \mathscr{K}_{m+1}$  and thus coincides with the index L introduced in (6) (cf. also (7)),

$$L = \min\{m \in \mathbb{N}_0 : \mathscr{H}_m = \mathscr{H}_{m+1}\} = \min\{m \in \mathbb{N}_0 : A^{-1}\mathbf{r}_0 \in \mathscr{H}_m\}$$
  
= min{deg q : q monic and q(A)\mathbf{r}\_0 = 0}. (31)

In other words,  $\mathscr{P}_{m-1}$  and  $\mathscr{K}_m$  are isomorphic linear spaces if and only if  $m \leq L$ .

The positive-semidefinite sesquilinear form

$$(p,q) := (p(A)\mathbf{r}_0, q(A)\mathbf{r}_0) \quad \left(p,q \in \mathscr{P}_\infty := \bigcup_{m \ge 0} \mathscr{P}_m\right)$$
(32)

is therefore positive definite when restricted to  $\mathscr{P}_{L-1}$  and hence defines an inner product on this space. We will use the same notation  $(\cdot, \cdot)$  for this inner product as for its counterpart on  $\mathscr{H}$ , as well as for derived quantities such as its induced norm  $\|\cdot\| := (\cdot, \cdot)^{1/2}$  and the orthogonality relation  $\perp$ .

Since every vector  $\mathbf{x} \in \mathbf{x}_0 + \mathscr{K}_m$  is of the form  $\mathbf{x} = \mathbf{x}_0 + q_{m-1}(A)\mathbf{r}_0$  for some  $q_{m-1} \in \mathscr{P}_{m-1}$ , the corresponding residual  $\mathbf{r} = \mathbf{b} - A\mathbf{x}$  can be written

$$\mathbf{r} = \mathbf{r}_0 - Aq_{m-1}(A)\mathbf{r}_0 = p_m(A)\mathbf{r}_0$$
, where  $p_m(\zeta) := 1 - \zeta q_{m-1}(\zeta) \in \mathscr{P}_m$ 

Note that the *residual polynomial*  $p_m$  satisfies the normalization condition  $p_m(0) = 1$ . Later in this section we will characterize the residual polynomials which belong to the OR and MR iterates as well as their zeros.

First, however, we provide three lemmata for later use. The first recalls a well-known (see, e.g., [15]) consequence of the Arnoldi decomposition  $AV_m = V_m H_m + \eta_{m+1,m} \boldsymbol{v}_{m+1} \boldsymbol{u}_m^{\mathrm{T}}$  of A (see (8)), the second states the conditions under which a Krylov space can have A-invariant subspaces. The third lemma shows that the orthogonal complement of a Krylov space with respect to an A-invariant subspace is itself a Krylov space.

**Lemma 3.1.** For every polynomial  $q(\zeta) = \alpha_m \zeta^m + \cdots + \alpha_1 \zeta + \alpha_0 \in \mathscr{P}_m$ , there holds

$$q(A)\mathbf{r}_0 = \beta V_m q(H_m) \mathbf{u}_1 + \alpha_m \beta \prod_{j=1}^m \eta_{j+1,j} \mathbf{v}_{m+1},$$

where  $u_1 \in \mathbb{C}^m$  denotes the first unit vector. In particular,  $q(A)\mathbf{r}_0 = \beta V_m q(H_m)u_1$  for every  $q \in \mathscr{P}_{m-1}$ .

**Lemma 3.2.** A Krylov space  $\mathscr{K}_m(A, \mathbf{r}_0)$  contains an A-invariant subspace if and only if it is itself A-invariant.

**Proof.** If  $\mathscr{U} \subset \mathscr{K}_m(A, \mathbf{r}_0)$  is *A*-invariant, it must contain an eigenvector  $\mathbf{z}$  of *A*. As an element of  $\mathscr{K}_m$ ,  $\mathbf{z}$  has a representation  $\mathbf{z} = q_{m-1}(A)\mathbf{r}_0$  in terms of a nonzero polynomial  $q_{m-1}$  of degree at most m-1. Moreover, if  $\lambda$  denotes the eigenvalue of *A* associated with  $\mathbf{z}$  and  $p(\zeta) := (\zeta - \lambda)q_{m-1}(\zeta)$ , then  $p(A)\mathbf{r}_0 = \mathbf{0}$  and hence the degree of the minimal polynomial  $c_{A,\mathbf{r}_0}$  of  $\mathbf{r}_0$  with respect to *A* is at most *m*. Consequently  $L = \deg c_{A,\mathbf{r}_0} \leq m$  and  $\mathscr{K}_m$  is *A*-invariant (cf. (31)).  $\Box$ 

**Lemma 3.3.** Let  $\mathscr{U}$  be an A-invariant subspace,  $\mathscr{T} = \mathscr{U}^{\perp}$  its orthogonal complement and set  $A_{\mathscr{T}} := P_{\mathscr{T}}AP_{\mathscr{T}}$ . Then there holds for m = 1, 2, ...

$$P_{\mathscr{T}}\mathscr{K}_m(A,\mathbf{r}_0) = \mathscr{K}_m(P_{\mathscr{T}}A,P_{\mathscr{T}}\mathbf{r}_0) = \mathscr{K}_m(A_{\mathscr{T}},P_{\mathscr{T}}\mathbf{r}_0)$$

and

$$P_{\mathscr{T}}A\mathscr{K}_m(A,\mathbf{r}_0) = P_{\mathscr{T}}A\mathscr{K}_m(P_{\mathscr{T}}A,P_{\mathscr{T}}\mathbf{r}_0) = A_{\mathscr{T}}\mathscr{K}_m(A_{\mathscr{T}},P_{\mathscr{T}}\mathbf{r}_0).$$

**Proof.** We have  $P_{\mathcal{T}}AP_{\mathcal{U}} = O$ , because  $\mathcal{U}$  is A-invariant, and therefore

 $P_{\mathscr{T}}A = P_{\mathscr{T}}AP_{\mathscr{U}} + P_{\mathscr{T}}AP_{\mathscr{T}} = P_{\mathscr{T}}AP_{\mathscr{T}}.$ 

An obvious induction now shows that for k = 1, 2, ...

$$P_{\mathscr{T}}A^{k}\boldsymbol{r}_{0}=\left[P_{\mathscr{T}}A\right]^{k}\boldsymbol{r}_{0}=\left[P_{\mathscr{T}}AP_{\mathscr{T}}\right]^{k}\boldsymbol{r}_{0},$$

which proves the assertions.  $\Box$ 

With regard to the notation used in Lemma 3.3, we remark that so far in this paper  $A_{\mathcal{F}}$  has denoted the orthogonal section  $P_{\mathcal{F}}A|_{\mathcal{F}}$  of A onto  $\mathcal{F}$ . We henceforth identify  $P_{\mathcal{F}}AP_{\mathcal{F}}$  with  $A_{\mathcal{F}}$  since  $P_{\mathcal{F}}AP_{\mathcal{F}} = P_{\mathcal{F}}A|_{\mathcal{F}}$  on  $\mathcal{F}$  and  $P_{\mathcal{F}}AP_{\mathcal{F}} = O$  on  $\mathcal{F}^{\perp}$ .

#### 3.2. OR residual polynomials

We first investigate the residual polynomials associated with the OR approach:  $\mathbf{r}_m^{\text{OR}} = p_m^{\text{OR}}(A)\mathbf{r}_0$ . The condition  $\mathbf{r}_m^{\text{OR}} \perp \mathscr{H}_m$  translates to  $p_m^{\text{OR}} \perp \mathscr{P}_{m-1}$ , i.e.,  $p_m^{\text{OR}}$  is an orthogonal polynomial of degree m (normalized to satisfy  $p_m^{\text{OR}}(0) = 1$ ). This also follows from the fact that  $\mathbf{r}_m^{\text{OR}}$  is a scalar multiple of  $\mathbf{v}_{m+1}$ , the last element of the orthonormal basis  $\{\mathbf{v}_1, \dots, \mathbf{v}_m, \mathbf{v}_{m+1}\}$  of  $\mathscr{H}_{m+1}$  (cf. Section 2.1): The basis vector  $\mathbf{v}_{m+1} \in \mathscr{H}_{m+1} \setminus \mathscr{H}_m$  has the form  $\mathbf{v}_{m+1} = v_m(A)\mathbf{r}_0$  for some polynomial  $v_m$  of exact degree m, and  $p_m^{\text{OR}}$  must be a scalar multiple of  $v_m$ . Next,  $\mathbf{v}_{m+1} \perp \mathscr{H}_m$ , i.e.,  $v_m \perp \mathscr{P}_{m-1}$ , and  $\|\mathbf{v}_{m+1}\| = \|v_m\| = 1$  show that  $v_m$  is an orthonormal polynomial of degree m. We arrive at  $p_m^{\text{OR}} = v_m/v_m(0)$ , a normalization which is, of course, only possible if  $v_m$  does not vanish at the origin. The close relation of  $v_m$  to the characteristic polynomial of the Hessenberg matrix  $H_m$  will show that  $v_m(0) = 0$  is equivalent to  $H_m$  being singular: We know that  $v_{m+1} = v_m(A)r_0$  spans the one-dimensional space  $\mathscr{K}_{m+1} \cap \mathscr{K}_m^{\perp}$ . If, on the other hand,  $h_m(\zeta) := \det(\zeta I - H_m) \in \mathscr{P}_m$  denotes the characteristic polynomial of  $H_m$ , then by Lemma 3.1 and the Cayley–Hamilton theorem

$$(h_m(A)\mathbf{r}_0, \mathbf{v}_k) = \beta(V_m h_m(H_m)\mathbf{u}_1, \mathbf{v}_k) + \beta \prod_{j=1}^m \eta_{j+1,j}(\mathbf{v}_{m+1}, \mathbf{v}_k) = 0$$
(33)

 $(k=1,2,\ldots,m)$ . In other words,  $v=h_m(A)r_0$  belongs to  $\mathscr{K}_{m+1}\cap \mathscr{K}_m^{\perp}$  and is therefore a scalar multiple of  $v_{m+1}$ . We have thus shown that the polynomials  $v_m$  and  $h_m$  can differ only by a scalar factor. We summarize these observations in

**Proposition 3.4.** The characteristic polynomial  $h_m$  of the Hessenberg matrix  $H_m$  is the (unique) monic orthogonal polynomial of degree m with respect to the inner product (32). The mth OR iterate exists if and only if  $h_m(0) \neq 0$  and, in this case, the corresponding residual polynomial is given by  $p_m^{OR} = h_m/h_m(0)$ .

We next consider the zeros of  $p_m^{OR}$  or, equivalently, the eigenvalues of  $H_m = V_m^* A V_m$ , the orthogonal section  $A_{\mathcal{H}_m}$  of A onto  $\mathcal{H}_m$ . Its eigenvalues  $\theta_i$ , where

$$H_m \mathbf{y}_j = \theta_j \mathbf{y}_j \qquad \text{with } \mathbf{y}_j \in \mathbb{C}^m, \|\mathbf{y}_j\|_2 = 1$$
(34)

are called the *Ritz values* of A (with respect to  $\mathscr{K}_m$ ), while  $z_i := V_m y_i$  are the associated *Ritz vectors*.

As the eigenvalues of the nonderogatory matrix  $H_m$ , Ritz values have geometric multiplicity one. In case  $\theta_j$  has algebraic multiplicity  $k_j > 1$ , we denote by  $\mathbf{y}_j^{(0)} = \mathbf{y}_j, \mathbf{y}_j^{(1)}, \dots, \mathbf{y}_j^{(k_j-1)}$  the principal vectors of  $H_m$  which belong to the eigenvalue  $\theta_j$ , so that

$$H_m \boldsymbol{y}_j^{(\ell)} = \theta_j \boldsymbol{y}_j^{(\ell)} + \boldsymbol{y}_j^{(\ell-1)} \quad (\ell = 1, \dots, k_j - 1)$$

and define  $z_j^{(0)} := V_m y_j^{(0)}$  and  $z_j^{(\ell)} := V_m y_j^{(\ell)}$  as the associated Ritz vectors.

Although all our conclusions remain valid in this more general case, we will assume in the remaining sections that  $H_m$  has *m* distinct eigenvalues to avoid the (notational) complication of requiring principal vectors.

The Ritz vectors constitute a basis of  $\mathscr{K}_m$ , and their residual vectors with regard to the eigenvalue problem (34) are given by

$$Az_{j} - \theta_{j}z_{j} = AV_{m}y_{j} - \theta_{j}V_{m}y_{j} = V_{m}H_{m}y_{j} + \eta_{m+1,m}\boldsymbol{v}_{m+1}\boldsymbol{u}_{m}^{\mathrm{T}}y_{j} - \theta_{j}V_{m}y_{j}$$
$$= \eta_{m+1}(\boldsymbol{u}_{m}^{\mathrm{T}}y_{j})\boldsymbol{v}_{m+1}.$$
(35)

This implies  $Az_j - \theta_j z_j \perp \mathscr{K}_m$ , which is the commonly used definition of Ritz values and Ritz vectors. We also observe that  $(A - \theta_j I)z_j \in \text{span}\{v_{m+1}\} = \text{span}\{h_m(A)r_0\}$  for every eigenvalue  $\theta_j$  of  $H_m$ . As an element of  $\mathscr{K}_m$ , each Ritz vector  $z_j$  can be represented as  $z_j = z_j(A)r_0$  with a polynomial  $z_j \in \mathscr{P}_{m-1}$ . Eq. (35) now implies  $(\zeta - \theta_j)z_j(\zeta) = \tau_j h_m(\zeta)$  with  $\tau_j \in \mathbb{C} \setminus \{0\}$ , which we express as

$$z_j(\zeta) = \tau_j \frac{h(\zeta)}{\zeta - \theta_j}.$$

#### **Proposition 3.5.** Let

$$h_m(\zeta) = \prod_{j=1}^J (\zeta - \theta_j)^{k_j} \quad (\theta_i \neq \theta_j \text{ for } i \neq j)$$

denote the characteristic polynomial of  $H_m$ . The Ritz vectors  $\mathbf{z}_j^{(\ell)}$  ( $\ell = 0, ..., k_j - 1$ ) of A with respect to  $\mathscr{K}_m(A, \mathbf{r}_0)$  have the form

$$z_{j}^{(\ell)} = z_{j}^{(\ell)}(A)\mathbf{r}_{0}, \quad where \ z_{j}^{(\ell)}(\zeta) = h_{m}(\zeta) \sum_{i=0}^{\ell} \frac{\tau_{j,i}}{(\zeta - \theta_{j})^{i+1}}$$

is a polynomial of exact degree m - 1. Moreover, there holds

$$(A - \theta_j I)^{\ell+1} \boldsymbol{z}_j^{(\ell)} \in \operatorname{span} \{ \boldsymbol{v}_{m+1} \} = \operatorname{span} \{ h_m(A) \boldsymbol{r}_0 \} = \operatorname{span} \{ \boldsymbol{r}_m^{\operatorname{OR}} \},$$

where the last equality assumes that the mth OR iterate is defined.

## 3.3. MR residual polynomials

We now turn to the investigation of the residual polynomials  $p_m^{MR}$  associated with the MR residuals  $\mathbf{r}_m^{MR} = p_m^{MR}(A)\mathbf{r}_0$ . Obviously, these polynomials possess the following minimization property:

$$||p_m^{\mathrm{MR}}|| = \min\{||p||: \deg p \leq m, p(0) = 1\}.$$

The condition  $\mathbf{r}_m^{\text{MR}} \perp A \mathscr{K}_m$  translates to  $p_m^{\text{MR}} \perp \zeta \mathscr{P}_{m-1}$ , from which we deduce the *reproducing property* (36) of the MR residual polynomials: For any  $q(\zeta) = q(0) + \sum_{j=1}^m \alpha_j \zeta^j \in \mathscr{P}_m$ , there holds

$$(q, p_m^{\mathrm{MR}}) = (q(0), p_m^{\mathrm{MR}}) + \left(\sum_{j=1}^m \alpha_j \zeta^j, p_m^{\mathrm{MR}}\right) = q(0)(1, p_m^{\mathrm{MR}})$$

and because this identity is valid in particular for  $q = p_m^{MR}$  yielding  $||p_m^{MR}||^2 = p_m^{MR}(0)(1, p_m^{MR}) = (1, p_m^{MR})$ , we obtain

$$(q, p_m^{\mathrm{MR}}) = q(0) \|p_m^{\mathrm{MR}}\|^2 \quad \text{for all } q \in \mathscr{P}_m.$$
(36)

The coefficients of  $p_m^{MR}$  with respect to the orthonormal basis  $\{v_0, v_1, \ldots, v_m\}$  of  $\mathscr{P}_m$  are thus given by

$$(p_m^{\mathrm{MR}}, v_j) = \overline{v_j(0)} \|p_m^{\mathrm{MR}}\|^2$$

which, in view of  $\|p_m^{MR}\|^2 = \sum_{j=0}^m |v_j(0)|^2 \|p_m^{MR}\|^4$ , results in the expansion

$$p_m^{\rm MR}(\zeta) = \frac{\sum_{j=0}^m \overline{v_j(0)} v_j(\zeta)}{\sum_{j=0}^m |v_j(0)|^2}$$
(37)

(note that the denominator  $\sum_{j=0}^{m} |v_j(0)|^2 \ge |v_0(0)|^2$  is always positive since  $v_0$  is a nonzero constant). Furthermore, this representation shows that, since the polynomials  $v_j$  are of exact degree j,  $p_m^{MR}$  will have degree m if and only if  $v_m(0) \ne 0$ , i.e., if the OR polynomial of degree m exists. Otherwise  $p_m^{MR} = p_{m-1}^{MR} = \cdots = p_k^{MR}$  and deg  $p_m^{MR} = k$ , if k is the largest index less than m for which  $v_k(0) \ne 0$ . To characterize the zeros of the MR residual polynomials in an analogous manner as for the OR residual polynomials, we begin by identifying them as the eigenvalues of an orthogonal section of  $A^{-1}$  onto the Krylov space

$$\mathscr{K}_m(A^{-1}, A^m \mathbf{r}_0) = \operatorname{span}\{A^m \mathbf{r}_0, A^{m-1} \mathbf{r}_0, \dots, A \mathbf{r}_0\} = A \mathscr{K}_m(A, \mathbf{r}_0).$$

We denote the associated Arnoldi decomposition by

$$A^{-1}W_m = W_{m+1}\tilde{G}_m = W_mG_m + \gamma_{m+1,m}\boldsymbol{w}_{m+1}\boldsymbol{u}_{m+1}^{\mathrm{T}}$$

in which  $\tilde{G}_m = [\gamma_{j,k}] \in \mathbb{C}^{(m+1)\times m}$  is an upper Hessenberg matrix,  $G_m$  the associated square Hessenberg matrix obtained by deleting the last row of  $\tilde{G}_m$  and  $W_{m+1} = [w_1 \dots w_{m+1}]$  is an orthonormal basis of  $\mathscr{H}_{m+1}(A^{-1}, A^m \mathbf{r}_0) = \mathscr{H}_{m+1}(A, \mathbf{r}_0)$ . If we invoke Lemma 3.1 applied to this Arnoldi decomposition, we obtain

$$q(A^{-1})A^m \mathbf{r}_0 = W_m q(G_m)\beta_m \mathbf{u}_1^{\mathrm{T}} + \alpha_m \beta_m \prod_{j=1}^m \gamma_{j+1,j} \mathbf{w}_{m+1}$$

for any polynomial  $q(\zeta) = \alpha_m \zeta^m + \cdots + \alpha_1 \zeta + \alpha_0 \in \mathscr{P}_m$ , where  $\beta_m = ||A^m \mathbf{r}_0||$ . Denoting by  $g_m$  the characteristic polynomial of  $G_m$ , we conclude just as in (33) that

$$(g_m(A^{-1})A^m \mathbf{r}_0, \mathbf{w}_k) = 0, \quad k = 1, \dots, m$$

and that  $w := g_m(A^{-1})A^m r_0$  belongs to

$$\mathscr{K}_{m+1}(A^{-1}, A^m \mathbf{r}_0) \cap \mathscr{K}_m(A^{-1}, A^m \mathbf{r}_0)^{\perp} = \mathscr{K}_{m+1}(A, \mathbf{r}_0) \cap (A \mathscr{K}_m(A, \mathbf{r}_0))^{\perp}$$

By virtue of its inclusion in the latter space, we conclude that the vector w is a scalar multiple of the MR residual vector  $\mathbf{r}_m^{\text{MR}}$ . Moreover, we observe that  $\hat{g}_m(\zeta) := g_m(\zeta^{-1})\zeta^m$  is a polynomial in  $\zeta$  of degree at most m, which is sometimes denoted as the *reversed polynomial* of  $g_m$  since it is obtained from  $g_m$  by reversing the order of the coefficients. Since  $\mathbf{w} = \hat{g}_m(A)\mathbf{r}_0$  and  $\mathbf{r}_m^{\text{MR}} = p_m^{\text{MR}}(A)\mathbf{r}_0$ are collinear, the same is true for the associated polynomials. Furthermore, since the characteristic polynomial  $g_m$  is monic, it follows that  $\hat{g}_m$  has value one at zero, and therefore that  $\hat{g}_m$  coincides with  $p_m^{\text{MR}}$ . The desired zeros of  $p_m^{\text{MR}}$  thus coincide with those of  $\hat{g}_m$ , which are easily seen to be the reciprocals of the zeros of  $g_m$ , which in turn are the eigenvalues of  $G_m$ . Since this matrix is not readily available, we instead derive a matrix which is similar to  $G_m$  and therefore has the same characteristic polynomial.

Departing from  $AV_m = \hat{V}_m R_m$  (cf. (12)), where  $\hat{V}_m$  denotes the Paige–Saunders basis of  $A\mathscr{K}_m(A, \mathbf{r}_0)$ and  $R_m$  is the triangular factor in the QR-factorization of  $\tilde{H}_m$ , we obtain

$$A^{-1}\hat{V}_{m} = V_{m}R_{m}^{-1} = V_{m+1}\begin{bmatrix} R_{m}^{-1} \\ \mathbf{0} \end{bmatrix} = V_{m+1}Q_{m}^{H}Q_{m}\begin{bmatrix} R_{m}^{-1} \\ \mathbf{0} \end{bmatrix}$$
$$= [\hat{V}_{m} \quad \tilde{\mathbf{v}}_{m+1}]Q_{m}\begin{bmatrix} R_{m}^{-1} \\ \mathbf{0} \end{bmatrix} =: [\hat{V}_{m} \quad \tilde{\mathbf{v}}_{m+1}]\tilde{F}_{m}$$
$$= \hat{V}_{m}F_{m} + \tilde{\mathbf{v}}_{m+1}f_{m}^{T} \quad \text{with } \tilde{F}_{m} \text{ partitioned as } \tilde{F}_{m} = \begin{bmatrix} F_{m} \\ f_{m}^{T} \end{bmatrix}.$$
(38)

We note that both  $\hat{V}_m$  and  $W_m$  are orthonormal bases of the same space  $A\mathscr{K}_m$ , which implies a relation of the form  $\hat{V}_m = W_m T$  with a unitary matrix  $T \in \mathbb{C}^{m \times m}$ . Therefore,

$$F_m = \hat{V}_m^* A^{-1} \hat{V}_m = T^{\mathrm{H}} G_m T$$

and  $F_m$  is similar to  $G_m$ . The zeros  $\tilde{\theta}_j$  of  $p_m^{MR}$  are therefore the reciprocals of the eigenvalues of  $F_m$ , determined by

$$\frac{1}{\tilde{\theta}_j}\hat{\boldsymbol{y}}_j = F_m\hat{\boldsymbol{y}}_j = [I_m \quad \boldsymbol{0}]\mathcal{Q}_m \begin{bmatrix} R_m^{-1} \\ \boldsymbol{0} \end{bmatrix} \hat{\boldsymbol{y}}_j = [I_m \quad \boldsymbol{0}]\mathcal{Q}_m \begin{bmatrix} I_m \\ \boldsymbol{0} \end{bmatrix} R_m^{-1}\hat{\boldsymbol{y}}_j = :\hat{\mathcal{Q}}_m R_m^{-1}\hat{\boldsymbol{y}}_j,$$

or equivalently, as solution of the generalized eigenvalue problem

$$R_m \tilde{\mathbf{y}}_j = \tilde{\theta}_j \hat{Q}_m \tilde{\mathbf{y}}_j, \quad \tilde{\mathbf{y}}_j := R_m^{-1} \hat{\mathbf{y}}_j.$$

The matrix  $\hat{Q}_m$  is obtained by deleting the last row and column of  $Q_m$ , which, by (13), yields

$$\hat{Q}_m = \begin{bmatrix} I_{m-1} & \mathbf{0} \\ \mathbf{0} & c_m \end{bmatrix} G_{m-1} \begin{bmatrix} G_{m-2} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \cdots \begin{bmatrix} G_1 & O \\ O & I_{m-2} \end{bmatrix}$$

Eq. (38) shows that  $F_m$  represents the orthogonal section of  $A^{-1}$  onto  $A\mathscr{K}_m$  with respect to  $\hat{V}_m$ . Its eigenvalues  $1/\tilde{\theta}_j$  are therefore the Ritz values of  $A^{-1}$  with respect to this space, and thus satisfy

$$\mathbf{0} = \hat{V}_m^* \left( A^{-1} \hat{V}_m \hat{\mathbf{y}}_j - \frac{1}{\tilde{\theta}_j} \hat{V}_m \hat{\mathbf{y}}_j \right) = \hat{V}_m^* \left( A^{-1} \hat{\mathbf{z}}_j - \frac{1}{\tilde{\theta}_j} \hat{\mathbf{z}}_j \right)$$

with Ritz vectors  $\hat{z}_j := \hat{V}_m \hat{y}_j$ , which, upon multiplication by  $\tilde{\theta}_j$ , substituting  $A^{-1}\hat{V}_m = V_m R_m^{-1}$  and multiplication by  $R_m^{\rm H}$ , becomes

$$\mathbf{0} = R_m^{\mathrm{H}} \hat{V}_m^* (A V_m R_m^{-1} \hat{\mathbf{y}}_j - \tilde{\theta}_j V_m R_m^{-1} \hat{\mathbf{y}}_j) = (A V_m)^* (A \tilde{\mathbf{z}}_j - \tilde{\theta}_j \tilde{\mathbf{z}}_j),$$
(39)

where  $\tilde{z}_j := V_m \tilde{y}_j = V_m R_m^{-1} \hat{y}_j = A^{-1} \hat{z}_j$ . Vectors  $\tilde{z}_j$  and numbers  $\tilde{\theta}_j$  which satisfy (39) are called *harmonic Ritz vectors and values* with respect to A and  $\mathscr{H}_m$  (cf. [15]). A better known characterization of these quantities is

$$(AV_m)^*V_m\tilde{\mathbf{y}}_j = \frac{1}{\tilde{\theta}_j}(AV_m)^*AV_m\tilde{\mathbf{y}}_j, \quad \text{i.e., } H_m^{\mathrm{H}}\tilde{\mathbf{y}}_j = \frac{1}{\tilde{\theta}_j}\tilde{H}_m^{\mathrm{H}}\tilde{H}_m\tilde{\mathbf{y}}_j.$$

That this formulation gives rise to the same set of eigenvalues can be seen from the similarity transformation

$$(\tilde{H}_{m}^{\mathrm{H}}\tilde{H}_{m})^{-1}H_{m}^{\mathrm{H}} = \begin{bmatrix} R_{m}^{-1} & \mathbf{0} \end{bmatrix} Q_{m} \begin{bmatrix} I_{m} \\ \mathbf{0} \end{bmatrix} = R_{m}^{-1}\begin{bmatrix} I_{m} & \mathbf{0} \end{bmatrix} Q_{m} \begin{bmatrix} R_{m}^{-1} \\ \mathbf{0} \end{bmatrix} R_{m} = R_{m}^{-1}F_{m}R_{m}$$

The harmonic Ritz vectors lie in  $\mathscr{K}_m$  and, in view of (39), satisfy

$$(A - \theta_j I) \tilde{z}_j \perp A \mathscr{K}_m.$$

In other words,  $(A - \tilde{\theta}_j I)\tilde{z}_j \in \mathscr{K}_{m+1} \cap (A\mathscr{K}_m)^{\perp} = \operatorname{span} \{\mathbf{r}_m^{\mathrm{MR}}\}\)$  and therefore, if the polynomials  $\tilde{z}_j \in \mathscr{P}_{m-1}$  are defined by  $\tilde{z}_j = \tilde{z}_j(A)\mathbf{r}_0$ , there holds

$$\tilde{z}_{j}(\zeta) = \tau_{j} \frac{p_{m}^{\text{MR}}(\zeta)}{\zeta - \tilde{\theta}_{j}} = \tau_{j} \frac{\hat{g}_{m}(\zeta)}{\zeta - \tilde{\theta}_{j}}$$

$$\tag{40}$$

for some normalization factor  $\tau_i \neq 0$ .

**Remark.** Polynomials which possess the reproducing property (36) are called *kernel polynomials*. Their role in Krylov subspace methods was first explored by Stiefel [24] in the Hermitian case and later extended to the non-Hermitian case by Freund [8,7] (see also [11]).

## 3.4. The implicitly restarted Arnoldi process

When manipulating Krylov subspaces, the following fundamental task often arises: given a Krylov space  $\mathscr{K}_m(A, \boldsymbol{v}_1)$  which is not A-invariant, along with the associated Arnoldi factorization

$$AV_{m} = V_{m}H_{m} + \eta_{m+1,m}\boldsymbol{v}_{m+1}\boldsymbol{u}_{m}^{1} \quad (\eta_{m+1,m} \neq 0)$$
(41)

and given an arbitrary vector  $v \in \mathscr{K}_{m-1}(A, v_1)$ , generate the Arnoldi factorization associated with  $\mathscr{K}_p(A, v)$ , i.e., using v as the initial vector, with p as large as possible *without* performing additional multiplications with A. The technique which accomplishes this task is known as the *implicitly* restarted Arnoldi (IRA) process and is due to Sorensen [22].

As a member of  $\mathscr{K}_{m-1}$ , v has the representation  $v = q_{k-1}(A)v_1$  with  $q_{k-1}$  of exact degree k-1,  $1 \leq k < m$ . In other words,  $v \in \mathscr{K}_k \setminus \mathscr{K}_{k-1}$ . We will show that p = m-k is maximal and the resulting Arnoldi factorization has the form

$$A\dot{V}_{p} = \dot{V}_{p}\dot{H}_{p} + \check{\eta}_{p+1,p}\check{\boldsymbol{v}}_{p+1}\boldsymbol{u}_{p}^{\mathrm{T}}$$

$$\tag{42}$$

with  $\check{v}_1 = v/||v||$ . That p = m - k holds should not come as a surprise because the construction of factorization (41) requires *m* multiplications by *A*, whereas *v* can be computed by only k - 1 matrix-vector products. Exactly p+1=m-k+1, i.e., the number of the 'remaining' multiplications by *A* are needed to construct (42) in the conventional way.

We assume the polynomial  $q_{k-1}$  is given in factored form  $q_{k-1}(\zeta) = \prod_{j=1}^{k-1} (\zeta - \theta_j)$ , as this is how it is used in the IRA method. The arguments that follow remain valid upon multiplying by a nonzero factor, so we may, without loss of generality, assume  $q_{k-1}$  to be monic. It is obviously sufficient to show how decomposition (42) can be established in the case k=2, i.e., if  $\mathbf{v} = (A - \theta I)\mathbf{v}_1$ . Polynomials of higher degree can then be handled by repeated application of the procedure below.

Each step of the IRA method is based on one step of the shifted QR algorithm. Following Sorensen [22, p. 363], we begin by subtracting  $\theta V_m$  on both sides of the Arnoldi decomposition (41)

$$(A - \theta I)V_m = V_m(H_m - \theta I) + \eta_{m+1,m} \boldsymbol{v}_{m+1} \boldsymbol{u}_m^{\mathrm{T}}$$

then form the QR factorization of  $H_m - \theta I$ ,

$$(A - \theta I)V_m = V_m Q R + \eta_{m+1,m} \boldsymbol{v}_{m+1} \boldsymbol{u}_m^1, \tag{43}$$

multiply by Q from the right,

$$(A - \theta I)V_m Q = (V_m Q)(RQ) + \eta_{m+1,m} \boldsymbol{v}_{m+1} \boldsymbol{u}_m^{\mathrm{T}} Q,$$

and add  $\theta V_m Q$  on both sides to obtain

$$A(V_mQ) = (V_mQ)(RQ + \theta I) + \eta_{m+1,m} \boldsymbol{v}_{m+1} \boldsymbol{u}_m^{\mathrm{T}} Q.$$
(44)

We rewrite (44) to introduce some extra notation:

$$A[\check{\boldsymbol{v}}_1 \quad \dots \quad \check{\boldsymbol{v}}_{m-1} \quad \boldsymbol{v}_m^+] = [\check{\boldsymbol{v}}_1 \quad \dots \quad \check{\boldsymbol{v}}_{m-1} \quad \boldsymbol{v}_m^+] \begin{bmatrix} \check{H}_{m-1} & * \\ \eta^+ \boldsymbol{u}_{m-1}^T & * \end{bmatrix} \\ + \eta_{m+1,m} \boldsymbol{v}_{m+1}[0 \quad \dots \quad 0 \quad q_{m,m-1} \quad q_{m,m}],$$

where we have made use of the fact that

$$RQ + \theta I = \begin{bmatrix} \check{H}_{m-1} & * \\ \eta^+ \boldsymbol{u}_{m-1}^{\mathrm{T}} & * \end{bmatrix} \in \mathbb{C}^{m \times m}$$

281

is again an upper Hessenberg matrix due to the upper Hessenberg form of Q. We note in passing that, in case  $\theta$  happens to be an eigenvalue of  $H_m$  (and only then), the last row of R is zero (and only the last row since  $H_m$  is nonderogatory) and therefore  $\eta^+ = 0$ .

We now omit the last column in (44), giving

$$A[\check{v}_1 \quad \dots \quad \check{v}_{m-1}] = [\check{v}_1 \quad \dots \quad \check{v}_{m-1}]H_{m-1} + (\eta^+ v_m^+ + \eta_{m+1,m} q_{m,m-1} v_{m+1})u_{m-1}^1$$

which, setting  $\check{\eta}_{m,m-1} := \|\eta^+ \boldsymbol{v}_m^+ + \eta_{m+1,m} q_{m,m-1} \boldsymbol{v}_{m+1}\|$ , becomes

$$A\check{V}_{m-1} = \check{V}_{m-1}\check{H}_{m-1} + \check{\eta}_{m,m-1}\check{v}_{m}\boldsymbol{u}_{m-1}^{\mathrm{T}}.$$
(45)

**Theorem 3.6.** With the notation introduced above, the decomposition (45) is an Arnoldi factorization of A with respect to the Krylov space  $\mathscr{K}_{m-1}(A, (A - \theta I)\mathbf{v}_1)$ .

**Proof.** Since Q is unitary, it follows that the elements of  $\check{V}_{m-1} = [\check{v}_1 \quad \dots \quad \check{v}_{m-1}]$  are orthonormal as the first m-1 elements of  $V_m Q$ . Next, the vector

$$\check{\boldsymbol{v}}_m = (\eta^+ \boldsymbol{v}_m^+ + \eta_{m+1,m} q_{m,m-1} \boldsymbol{v}_{m+1}) / \check{\eta}_{m,m-1}$$

has unit norm and is orthogonal to  $\check{v}_1, \ldots, \check{v}_{m-1}$  since  $v_m^+$ , as the last element  $V_mQ$ , is orthogonal to the previous elements  $\check{v}_1, \ldots, \check{v}_{m-1}$  and since  $v_{m+1}$  is orthogonal to  $V_m$  and hence also to  $V_mQ$ . That the new first basis vector  $\check{v}_1$  is a multiple of  $(A - \theta I)v_1$  follows by equating the first vector on both sides of (43). It remains to show that the Hessenberg matrix  $\check{H}_{m-1}$  is unreduced. If  $\check{\eta}_{k+1,k} = 0$  for some k < m, then this would imply that  $\mathscr{K}_k(A, \check{v}_1)$  is a proper A-invariant subspace of  $\mathscr{K}_m(A, v_1)$ , which, in view of Lemma 3.2, contradicts the assumption  $\eta_{m+1,m} \neq 0$ .  $\Box$ 

As mentioned previously, decomposition (42) involving a new starting vector  $\check{v}_1 = q_{k-1}(A)v_1$  is effected by k-1 steps of the procedure outlined above. For later use, we note that the associated Krylov space is given by

$$\mathscr{K}_p(A,\check{\mathbf{v}}_1) = \{r(A)q_{k-1}(A)\mathbf{v}_1 : r \in \mathscr{P}_{p-1}\} \subset \mathscr{K}_{p+k-1}(A,\mathbf{v}_1).$$

## 4. Augmentation strategies and some algorithmic realizations

Up to this point we have not yet considered the question of how to construct suitable correction spaces  $\mathscr{C}_m$  for a given initial approximation  $\mathbf{x}_0$  to the solution of a linear system (1). In practice, this task usually arises in the following form. Given a correction space  $\mathscr{C}$ , select vectors  $\mathbf{a}_1, \ldots, \mathbf{a}_k$  such that the *augmented* correction space  $\widetilde{\mathscr{C}} := \mathscr{C} + \text{span}\{\mathbf{a}_1, \ldots, \mathbf{a}_k\}$  has better correction properties.

We first specify the (obvious) meaning of phrases such as 'well-suited correction space', 'better correction properties', etc. Let  $\mathscr{C} \subset \mathscr{H}$  be a finite-dimensional subspace and denote by  $\mathbf{r}^{MR}$  the residual vector of the MR approximation with respect to the correction space  $\mathscr{C}$ . Whether or not  $\mathscr{C}$ is well suited as a correction space, i.e., whether or not  $A\mathscr{C}$  contains an acceptable approximation to  $\mathbf{r}_0$ , depends, in view of  $\|\mathbf{r}^{MR}\| = \sin \angle (\mathbf{r}_0, A\mathscr{C}) \|\mathbf{r}_0\|$  (cf. (17)), only on the size of the angle  $\varphi := \angle (\mathbf{r}_0, A\mathscr{C})$ .  $\mathscr{C}$  is optimal, i.e.,  $\mathbf{r}^{MR} = \mathbf{0}$  if and only if  $\varphi = 0$ . The worst case is that in which the optimal correction from  $\mathscr{C}$  is the null vector (i.e.,  $\mathbf{r}^{MR} = \mathbf{r}_0$ ), and this occurs precisely for  $\varphi = \pi/2$ , or equivalently, for  $\mathbf{r}_0 \perp A\mathscr{C}$ . In Section 4.1 we comment on two general strategies for augmenting correction spaces, the first of which adds nearly *A*-invariant subspaces to the correction space, whereas the second adds approximate solution of residual equations. Subsequently we survey and compare existing algorithms in which the ideas and strategies developed in the previous sections have been realized.

# 4.1. General augmentation strategies

It has often been suggested, primarily in the context of Krylov subspace methods, that it is a desirable goal that the correction space  $\mathscr{C}$  be either nearly A-invariant or contain a nearly A-invariant subspace, usually spanned by a few approximate eigenvectors of A. Clearly, if a given correction space  $\mathscr{C}$  which contains the initial residual  $\mathbf{r}_0$  – as do e.g. all Krylov spaces – is exactly A-invariant, then  $\varphi = 0$  and the MR approximation with respect to  $\mathscr{C}$  yields the exact solution. If only a subspace  $\mathscr{U}$  of  $\mathscr{C}$  is A-invariant, or nearly so in the sense that it lies at a small angle to its image under A, Proposition 4.1 shows that the MR residual with respect to  $\mathscr{C}$  then has a small component in the direction of  $\mathscr{U}$ .

**Proposition 4.1.** Given a correction space  $\mathscr{C}$ , let  $\mathscr{U} \subset \mathscr{C}$  denote a subspace such that  $\sin \angle (A\mathscr{U}, \mathscr{U}) \leq \varepsilon$ . Then the MR residual  $\mathbf{r}^{MR}$  with respect to  $\mathscr{C}$  satisfies  $\|P_{\mathscr{U}}\mathbf{r}^{MR}\| \leq \varepsilon \|\mathbf{r}_0\|$ .

**Proof.** The assertion follows from  $P_{\mathscr{U}} \mathbf{r}^{\mathrm{MR}} = P_{\mathscr{U}}(I - P_{A\mathscr{C}})\mathbf{r}_0$  and  $||P_{\mathscr{U}}(I - P_{A\mathscr{C}})|| \leq ||P_{\mathscr{U}}(I - P_{A\mathscr{U}})|| = \sin \angle (A\mathscr{U}, \mathscr{U}) \leq \varepsilon$ .  $\Box$ 

In particular, if  $\mathscr{C}$  contains an exactly invariant subspace  $\mathscr{U}$ , then the MR approximation removes the components of the initial residual in the direction of  $\mathscr{U}$  completely. Of course, this may only be of limited use if  $||(I - P_{\mathscr{U}})\mathbf{r}_0||/||\mathbf{r}_0||$  is large, i.e., if  $\mathscr{U}$  does not contain a good approximation of  $\mathbf{r}_0$ . In short, the existence of A-invariant subspaces of  $\mathscr{C}$  per se need not be beneficial.

In Lemma 3.2 we already proved that if  $\mathscr{C} = \mathscr{K}_m(A, \mathbf{r}_0)$  is a Krylov space, then it cannot contain an *A*-invariant subspace  $\mathscr{U}$  unless  $\mathscr{K}_m(A, \mathbf{r}_0)$  is itself *A*-invariant, i.e.,  $\mathscr{K}_m(A, \mathbf{r}_0) = \mathscr{K}_L(A, \mathbf{r}_0)$ . Obviously, augmenting  $\mathscr{K}_m(A, \mathbf{r}_0)$  by span $\{A^m \mathbf{r}_0, \dots, A^{L-1} \mathbf{r}_0\}$  leads to the new correction space  $\mathscr{K}_L(A, \mathbf{r}_0)$  which is *A*-invariant. We now show that there is no 'faster' way to augment  $\mathscr{K}_m(A, \mathbf{r}_0)$  to an *A*-invariant space.

**Proposition 4.2.** Let  $\tilde{\mathscr{C}}$  be an A-invariant subspace containing  $\mathscr{K}_m(A, \mathbf{r}_0)$ . Then  $\tilde{\mathscr{C}}$  contains  $\mathscr{K}_L(A, \mathbf{r}_0)$ .

**Proof.** By  $\mathscr{U}_0 := \cap \{\mathscr{U} : \mathscr{U} \text{ is an } A \text{-invariant subspace with } \mathscr{K}_m \subseteq \mathscr{U} \}$  we denote the smallest A -invariant subspace containing  $\mathscr{K}_m$ . By definition,  $\mathscr{U}_0 \subseteq \mathscr{K}_L$ . On the other hand, since  $\mathscr{U}_0$  contains  $\mathbf{r}_0$  and is invariant under A, it must contain also  $A^m \mathbf{r}_0$  for very  $m = 0, 1, \ldots, i.e., \mathscr{K}_L \subseteq \mathscr{U}_0$ .  $\Box$ 

Proposition 4.2 should not lead to the conclusion that it is useless to augment a Krylov subspace  $\mathscr{C} = \mathscr{K}_m$  by an *A*-invariant subspace  $\mathscr{U}$ . After all, by Proposition 4.1 the MR residual with respect to  $\tilde{\mathscr{C}} = \mathscr{C} + \mathscr{U}$  contains no component in the direction of  $\mathscr{U}$ . We show next that the MR approach with respect to the augmented space  $\tilde{\mathscr{C}}$  yields an MR approximation with respect to another Krylov subspace, associated with a 'smaller' linear system.

**Lemma 4.3.** Let  $\tilde{\mathbf{r}}^{MR}$  denote the MR residual with respect to  $\tilde{\mathscr{C}} = \mathscr{K}_m(A, \mathbf{r}_0) + \mathscr{U}$ , where  $\mathscr{U}$  is an *A*-invariant subspace. Set further  $\mathscr{T} := \mathscr{U}^{\perp}$ ,  $A_{\mathscr{T}} := P_{\mathscr{T}}AP_{\mathscr{T}}$  and, finally, let  $\mathbf{r}^{MR}$  be the residual of the MR approximation for  $A_{\mathscr{T}}\mathbf{x} = P_{\mathscr{T}}\mathbf{r}_0$  with respect to the correction space  $\mathscr{K}_m(A_{\mathscr{T}}, P_{\mathscr{T}}\mathbf{r}_0)$ . Then there holds

 $\tilde{\mathbf{r}}^{MR} = \mathbf{r}^{MR}$  or, equivalently  $P_{\mathcal{U}}\tilde{\mathbf{r}}^{MR} = \mathbf{0}$  and  $P_{\mathcal{T}}\tilde{\mathbf{r}}^{MR} = \mathbf{r}^{MR}$ .

**Proof.** As in Section 2.3 we split the computation of  $\tilde{r}^{MR}$  into two subtasks and write (using that  $\mathcal{U}$  is *A*-invariant)

$$\tilde{\boldsymbol{r}}^{\mathrm{MR}} = (I - P_{\mathscr{U}})\boldsymbol{r}_0 - P_{\mathscr{U}}(I - P_{\mathscr{U}})\boldsymbol{r}_0 = (I - P_{\mathscr{U}})P_{\mathscr{T}}\boldsymbol{r}_0,$$

where  $\mathscr{Z} = (I - P_{\mathscr{U}})A\mathscr{K}_m(A, \mathbf{r}_0) = A\mathscr{K}_m(A, \mathbf{r}_0) \cap \mathscr{T} \subseteq \mathscr{T}$ , whereby  $P_{\mathscr{U}}P_{\mathscr{Z}} = O$ . This implies  $P_{\mathscr{U}}\tilde{\mathbf{r}}^{MR} = \mathbf{0}$  (a fact we could also have deduced directly from Proposition 4.1).

Since  $P_{\mathcal{T}}A\mathscr{K}_m(A, \mathbf{r}_0) = A_{\mathcal{T}}\mathscr{K}_m(A_{\mathcal{T}}, P_{\mathcal{T}}\mathbf{r}_0)$  (cf. Lemma 3.3),

$$\tilde{\mathbf{r}}^{\mathrm{MR}} = (I - P_{P_{\mathscr{T}} A \mathscr{K}_{m}(A, \mathbf{r}_{0})}) P_{\mathscr{T}} \mathbf{r}_{0} = (I - P_{A_{\mathscr{T}} \mathscr{K}_{m}(A_{\mathscr{T}}, P_{\mathscr{T}} \mathbf{r}_{0})}) P_{\mathscr{T}} \mathbf{r}_{0},$$

identifying  $\tilde{r}^{MR}$  as the residual of the MR approximation for  $A_{\mathscr{T}} \mathbf{x} = P_{\mathscr{T}} \mathbf{r}_0$  with respect to the Krylov space  $\mathscr{H}_m(A_{\mathscr{T}}, P_{\mathscr{T}} \mathbf{r}_0)$ .  $\Box$ 

A different strategy for enriching correction spaces is common for many inner-outer iteration schemes and based on the following trivial observation: Suppose that, for a given correction space  $\mathscr{C}$  and associated residual space  $\mathscr{V} = \operatorname{span}\{r_0\} + A\mathscr{C}$ , we are able to solve Ac = r for some  $r \in \mathscr{V}$ . Such an r has a representation  $r = r_0 - A\tilde{c}$  with  $\tilde{c} \in \mathscr{C}$ , and therefore, by virtue of

$$A\mathbf{c} = \mathbf{r} = \mathbf{r}_0 - A\tilde{\mathbf{c}}, \quad \text{i.e.}, \quad \mathbf{r}_0 = A(\mathbf{c} + \tilde{\mathbf{c}}),$$

we see that the augmented correction space  $\tilde{\mathscr{C}} = \mathscr{C} + \text{span}\{c\}$  contains the exact correction. In practice, since solving Ac = r is generally as difficult as the original problem, one applies an inexpensive approximate solution method to this auxiliary problem, yielding a vector c satisfying Ac = r + h and consequently,  $\|\tilde{r}^{MR}\| \leq \|h\|$  for the MR residual with respect to  $\tilde{\mathscr{C}}$ .

The FGMRES algorithm of Saad [17], which is the natural generalization of GMRES to the case of an arbitrary correction space, was originally introduced as a technique that enlarges the correction space at each step by an approximate solution of such a residual equation. In [17], this is achieved by selecting the new correction direction  $c_{m+1}$  as the result of a preconditioning step applied to the most recent basis vector  $v_{m+1}$  of the residual space  $\mathscr{V}_{m+1}$ , which may be viewed as an approximate solution of the equation  $Ac = v_{m+1}$ .

A similar approach is taken in the GMRESR (which stands for GMRES Recursive) method of van der Vorst and Vuik [27]. In each step of GMRESR, the new correction vector  $c_{m+1}$  is chosen as the approximate solution of the equation  $Ac = r_m$  obtained by a given number of GMRES steps, where  $r_m$  is the residual of the MR approximation using the current correction space  $\mathscr{C}_m$ . This method was improved upon by de Sturler [25], who observed that, by enforcing orthogonality of the approximation space of the inner GMRES iteration, one can obtain as a result of the inner GMRES iteration the best approximation of  $r_0$  from the sum of the inner and outer approximation spaces as described in Section 2.3. In other words, the inner iteration consists of GMRES applied to Eq. (21). The resulting inner–outer iteration scheme is called GCRO.

## 4.2. Restarted GMRES

In general, the implementation of OR and MR methods require computing and storing at least one orthonormal basis of a space which grows in dimension with each step. A result of Faber and Manteuffel [6] shows that this considerable computational effort can be avoided essentially only for self-adjoint A. It is therefore not surprising that the necessity of truncating or restarting in practical implementations of MR and OR methods is as old as these methods themselves (cf. [21,4]). The most widely used algorithm is GMRES(m), the restarted version of GMRES, which uses a Krylov space of dimension m. One cycle of GMRES(m) for solving (1) with initial residual  $\mathbf{r}_0$  consists of generating the Krylov space  $\mathscr{K}_m(A, \mathbf{r}_0)$ , forming the MR approximation with respect to the correction space  $\mathscr{C} = \mathscr{K}_m(A, \mathbf{r}_0)$  and repeating this process using the resulting residual as the initial residual for the next cycle until a stopping criterion is satisfied.

In the terminology of Section 2, two consecutive cycles of GMRES(m) consist of two MR approximations with respect to the correction spaces

$$\mathscr{C}_1 = \mathscr{K}_m(A, \mathbf{r}_0)$$
 and  $\mathscr{C}_2 = \mathscr{K}_m(A, \mathbf{r}_m),$ 

where  $\mathbf{r}_m$  denotes the residual of the MR approximation computed in the first cycle. No orthogonalization of the residual space  $\mathscr{V}_2$  against the approximation space  $\mathscr{AC}_1$  is performed in the second cycle, and thus, in general, the approximation after the second cycle is no longer the MR approximation with respect to  $\mathscr{C}_1 + \mathscr{C}_2$ . Besides this inexact approximation, it may also happen that the sum is not direct. In the extreme case there holds  $\mathbf{r}_m = \mathbf{r}_0$  after the first cycle, so that the second cycle constructs the identical Krylov space (as do all subsequent cycles) and no progress is made, a phenomenon known as *stalling*.

**Proposition 4.4.** For two consecutive cycles of GMRES(m) with initial residual  $\mathbf{r}_0$ , there holds

$$\mathscr{K}_m(A,\mathbf{r}_0) \oplus \mathscr{K}_m(A,\mathbf{r}_m) = \mathscr{K}_{2m}(A,\mathbf{r}_0) \tag{46}$$

if and only if no stagnation occurs in the last step of the first cycle.

**Proof.** By definition,  $\mathscr{K}_m(A, \mathbf{r}_m) = \{q(A) p_m^{MR}(A) \mathbf{r}_0 : q \in \mathscr{P}_{m-1}\}$ , where  $p_m^{MR}$  denotes the MR polynomial of the last step of the first cycle, and this shows that (46) holds if and only if  $p_m^{MR}$  has degree *m*. Representation (37) of  $p_m^{MR}$  shows that this is equivalent with  $v_m(0) \neq 0$ , which is equivalent to stagnation at step *m*.  $\Box$ 

One of the more common misconceptions regarding GMRES(m) is that a method with larger restart length *m* applied to the same problem will converge at least as fast as the method with smaller *m*. A simple counterexample<sup>4</sup> is provided by the  $3 \times 3$  system

		[1 0 0]			[ -1 ]
$A\mathbf{x} = \mathbf{b},$	A =	1 1 0	,	<b>b</b> =	1
		011			1

<sup>&</sup>lt;sup>4</sup> The authors would like to thank E. de Sturler for pointing out this phenomenon reporting a similar observation in the context of a discrete convection–diffusion problem.

with initial guess  $\mathbf{x}_0 = \mathbf{0}$ . Two cycles of GMRES(2) applied to this example result in a residual norm of  $\|\mathbf{r}_4^{(2)}\| = 4/15 = 0.26...$ , whereas four cycles of GMRES(1), which involve the same number of matrix-vector multiplications, yields  $\|\mathbf{r}_4^{(1)}\| = 0.057...$ . The gap between GMRES(1) and GMRES(2) widens further in subsequent iteration steps, e.g.,  $\|\mathbf{r}_{18}^{(1)}\|_2 = 1.6...10^{-12}$ , whereas  $\|\mathbf{r}_{18}^{(2)}\|_2 = 3.9...10^{-5}$ . Even more surprising in this example is that  $\|\mathbf{r}_{10}^{(1)}\|_2 < \|\mathbf{r}_{20}^{(2)}\|_2$ , showing that ten cycles of GMRES(1) have reduced the residual further than ten cycles of GMRES(2). By expanding this example to the analogous matrix for higher dimensions *n* one can observe that GMRES(*m*) is ultimately slower for this system than GMRES(*m* - 1) for m = 2, ..., n - 1.

#### 4.3. Deflation by augmentation

The first algorithm which attempts to improve the restarted GMRES method by augmenting the Krylov space is due to Morgan [14]. This approach selects a fixed number of approximate eigenvectors of A to add to the Krylov space of the following cycle, as motivated, e.g., by Lemma 4.3. Since the emphasis of [13] is on cases in which the eigenvalues close to the origin limit the convergence rate the most – as is the case, e.g., for the so-called model problem of the discrete Laplacian on the unit cube – harmonic Ritz vectors are chosen as the eigenvalues close to zero more accurately than classical Ritz values.

Each step except the first consists of forming the MR approximation with respect to a correction space  $\mathscr{C} = \mathscr{C}_1 + \mathscr{C}_2$  with  $\mathscr{C}_1 = \mathscr{K}_m(A, \mathbf{r}_0)$  and  $\mathscr{C}_2 = \operatorname{span}\{\tilde{z}_1, \dots, \tilde{z}_k\}$ . The vectors  $\tilde{z}_1, \dots, \tilde{z}_k$  are the harmonic Ritz vectors associated with the *k* harmonic Ritz values  $\tilde{\theta}_1, \dots, \tilde{\theta}_k$  of *A* with respect to the previous correction space which are closest to the origin. Since no eigenvector information is available in the first cycle, the first correction space is chosen simply as  $\mathscr{C} = \mathscr{K}_{m+k}(A, \mathbf{r}_0)$ .

As subsequently shown by Morgan [14], there is a less expensive implementation of this approach. Consider the MR approximation with initial residual  $\mathbf{r}_0$  with respect to the (m+k)-dimensional Krylov space  $\mathscr{K}_{m+k}(A, \mathbf{r}_0)$ . As shown in Section 3.3, the associated residual vector has the representation

$$\mathbf{r}_{m+k}^{\mathrm{MR}} = p_{m+k}^{\mathrm{MR}}(A)\mathbf{r}_0, \text{ where } p_{m+k}^{\mathrm{MR}}(\zeta) = \prod_{j=1}^{m+k} \left(1 - \frac{\zeta}{\tilde{\theta}_j}\right).$$

We denote by  $q_m$  the polynomial whose zeros are the harmonic Ritz values  $\tilde{\theta}_{k+1}, \ldots, \tilde{\theta}_{k+m}$ , i.e., those largest in modulus.

**Theorem 4.5.** The correction space *C* of Morgan's method is itself a Krylov space, namely

$$\mathscr{C} = \mathscr{K}_m(A, \mathbf{r}_{m+k}) + \operatorname{span}\{\tilde{\mathbf{z}}_1, \dots, \tilde{\mathbf{z}}_k\} = \mathscr{K}_{m+k}(A, q_m(A)\mathbf{r}_0).$$
(47)

**Proof.** The rightmost member of (47) can be represented as

$$\mathscr{K}_{m+k}(A, q_m(A)\mathbf{r}_0) = \{r(A)q_m(A)\mathbf{r}_0 : r \in \mathscr{P}_{m+k-1}\}.$$

On the other hand, by (40), the harmonic Ritz vectors may be represented in terms of polynomials as  $\tilde{z}_i = \tilde{z}_i(A)r_0$  with

$$ilde{z}_{j}(\zeta) = rac{p_{m+k}^{\mathrm{MR}}(\zeta)}{\zeta - ilde{ heta}_{j}} = q_{m}(\zeta) \prod_{\substack{\ell=1 \ \ell 
eq j}}^{k} \left(1 - rac{\zeta}{ ilde{ heta}_{\ell}}\right),$$

whereas  $\mathbf{r}_{m+k}^{\text{MR}} = p_{m+k}^{\text{MR}}(A)\mathbf{r}_0$ , with

$$p_{m+k}^{\mathrm{MR}}(\zeta) = q_m(\zeta) \prod_{\ell=1}^k \left(1 - \frac{\zeta}{\tilde{ heta}_\ell}\right).$$

Therefore, the correction space of Morgan's method may be characterized as

$$\mathscr{C} = \{q_m(A)q(A)\mathbf{r}_0 : q \in \mathscr{Q}\},$$

where the polynomial space  $\mathcal{Q}$  is given by

$$\begin{split} \mathscr{Q} &:= \prod_{\ell=1}^k \left(1 - rac{\zeta}{ ilde{ heta}_\ell}
ight) \mathscr{P}_{m-1} + \operatorname{span} \left\{\prod_{\substack{\ell=1 \ \ell \neq j}}^k \left(1 - rac{\zeta}{ ilde{ heta}_\ell}
ight) : j = 1 \dots, k 
ight\} \\ &= \prod_{\ell=1}^k \left(1 - rac{\zeta}{ ilde{ heta}_\ell}
ight) \mathscr{P}_{m-1} + \mathscr{P}_{k-1} = \mathscr{P}_{m+k-1}, \end{split}$$

where the middle equality follows from the fact that  $\tilde{\theta}_1, \ldots, \tilde{\theta}_k$  are distinct.  $\Box$ 

Eq. (47) shows that  $\mathscr{C}$  can be generated by applying the IRA method to  $\mathscr{K}_{m+k}(A, \mathbf{r}_0)$ , using  $\tilde{\theta}_{k+1}, \ldots, \tilde{\theta}_{m+k}$  as shifts, to obtain  $\mathscr{K}_k(A, q_m(A)\mathbf{r}_0)$ . The space  $\mathscr{C}$  is then obtained after *m* further steps of the Arnoldi process. This approach is computationally less expensive in that *k* fewer matrix-vector multiplications with *A* are required.

As also noted by Morgan, an analogous method can be used to augment the Krylov space in conjunction with an OR iteration. In this case, however, Ritz values and vectors must be used in place of harmonic Ritz values/vectors, as the Ritz values are the zeros of the OR residual polynomial.

### 4.4. Deflation by preconditioning

The methods of the next class also attempt to utilize spectral information gained during the course of the iteration to accelerate convergence. Instead of augmenting the Krylov space, however, these methods use this information to construct preconditioners which can be improved as more accurate spectral information becomes available. Such an approach was proposed by Erhel et al. [5].

To motivate this approach, assume  $\mathcal{U}$  is an A-invariant subspace of dimension k with orthonormal basis U, i.e.,

$$AU =: UA_U, \qquad A_U \in \mathbb{C}^{k \times k}.$$

Note that  $A_U$  is the specific representation of the orthogonal section  $A_{\mathcal{U}}$  with respect to the basis U. Denoting by T an orthonormal basis of the orthogonal complement  $\mathcal{T} = \mathcal{U}^{\perp}$ , we can represent the

286

action of A as

$$A[U \quad T] = \begin{bmatrix} U & T \end{bmatrix} \begin{bmatrix} A_U & U^*AT \\ O & T^*AT \end{bmatrix}.$$

Under the assumption that k is small, it is feasible to solve systems involving  $A_U$  directly, and thus to precondition by M defined as

$$M[U \quad T] = \begin{bmatrix} U \quad T \end{bmatrix} \begin{bmatrix} A_U & O \\ O & I \end{bmatrix}$$
(48)

at each step of the iteration. The resulting right-preconditioned operator is then

$$AM^{-1}[U \quad T] = \begin{bmatrix} U \quad T \end{bmatrix} \begin{bmatrix} I \quad U^*AT \\ O \quad T^*AT \end{bmatrix}, \quad \text{i.e.,} \quad AM^{-1} = P_{\mathcal{U}} + AP_{\mathcal{T}}.$$
(49)

We want to compare this approach with Morgan's method of augmenting the Krylov space  $\mathscr{K}_m(A, \mathbf{r}_0)$  by the A-invariant subspace  $\mathscr{U}$ .

**Theorem 4.6.** Let  $\mathbf{r}_m^M$  denote the MR residual with respect to the correction space  $\mathcal{U} + \mathcal{K}_m(A, \mathbf{r}_0)$ , where  $\mathcal{U}$  is an A-invariant subspace, and let  $\mathbf{r}_m^E$  denote the MR residual with respect to the correction space  $\mathcal{K}_m(AM^{-1}, \mathbf{r}_0)$  resulting from preconditioning A from the right by M as defined in (48). Then there holds

$$0 = \|P_{\mathscr{U}}\boldsymbol{r}_{m}^{M}\| \leq \|P_{\mathscr{U}}\boldsymbol{r}_{m}^{E}\| \quad and \quad \|P_{\mathscr{T}}\boldsymbol{r}_{m}^{M}\| \leq \|P_{\mathscr{T}}\boldsymbol{r}_{m}^{E}\|,$$

$$(50)$$

in particular,  $\|\mathbf{r}_m^M\| \leq \|\mathbf{r}_m^E\|$ . If, in addition, also  $\mathcal{T} = \mathcal{U}^{\perp}$  is A-invariant, then,  $P_{\mathcal{U}}\mathbf{r}_0 = \mathbf{0}$  implies  $\mathbf{r}_m^E = \mathbf{r}_m^M$ .

**Proof.** The left set of inequalities in (50) follow from  $P_{\mathcal{U}} \mathbf{r}_m^M = \mathbf{0}$  which was proved in Lemma 4.3.

We next recall that  $A_{\mathcal{T}} = P_{\mathcal{T}}AP_{\mathcal{T}}$  is the orthogonal section of A onto  $\mathcal{T}$  (cf. the remark following Lemma 3.3). Since  $\mathbf{r}_m^E = \mathbf{r}_0 - AM^{-1}\mathbf{c}$ , for some  $\mathbf{c} \in \mathscr{K}_m(AM^{-1}, \mathbf{r}_0)$  we obtain using (49)

$$P_{\mathscr{T}}\boldsymbol{r}_{m}^{E}=P_{\mathscr{T}}\boldsymbol{r}_{0}-P_{\mathscr{T}}AM^{-1}\boldsymbol{c}=P_{\mathscr{T}}\boldsymbol{r}_{0}-P_{\mathscr{T}}AP_{\mathscr{T}}\boldsymbol{c}=P_{\mathscr{T}}\boldsymbol{r}_{0}-A_{\mathscr{T}}P_{\mathscr{T}}\boldsymbol{c}.$$

Moreover,  $AM^{-1}\mathcal{U} = \mathcal{U}$  together with Lemma 3.3 yield

$$P_{\mathscr{T}}\boldsymbol{c} \in P_{\mathscr{T}}\mathscr{K}_m(AM^{-1},\boldsymbol{r}_0) = \mathscr{K}_m(P_{\mathscr{T}}AM^{-1},P_{\mathscr{T}}\boldsymbol{r}_0) = \mathscr{K}_m(A_{\mathscr{T}},P_{\mathscr{T}}\boldsymbol{r}_0).$$

The last two statements show that  $P_{\mathscr{T}} \mathbf{r}_m^E$  is of the form  $P_{\mathscr{T}} \mathbf{r}_0 - A_{\mathscr{T}} \tilde{\mathbf{c}}$  with  $\tilde{\mathbf{c}} \in \mathscr{K}_m(A_{\mathscr{T}}, P_{\mathscr{T}} \mathbf{r}_0)$ . On the other hand, by Proposition 4.3 there holds

$$\|\boldsymbol{r}_m^M\| = \min_{\boldsymbol{c}\in\mathscr{K}_m(A_{\mathscr{T}},P_{\mathscr{T}}\boldsymbol{r}_0)} \|P_{\mathscr{T}}\boldsymbol{r}_0 - A_{\mathscr{T}}\boldsymbol{c}\|$$

i.e.,  $\|\mathbf{r}_m^M\|$  minimizes all expressions of this form, yielding the right inequality of (50).

Next, assuming  $A\mathcal{T} = \mathcal{T}$ , (49) implies  $AM^{-1}\mathbf{r}_0 = A_{\mathcal{T}}\mathbf{r}_0$  for  $\mathbf{r}_0 \in \mathcal{T}$ , and thus  $\mathscr{K}_m(AM^{-1}, \mathbf{r}_0) = \mathscr{K}_m(A_{\mathcal{T}}, P_{\mathcal{T}}\mathbf{r}_0)$ , which shows that in this case both methods minimize over the same space, hence  $\mathbf{r}_m^E = \mathbf{r}_m^M$ .  $\Box$ 

We note that the assumption  $P_{\mathscr{U}}\mathbf{r}_0 = \mathbf{0}$  is not restrictive, as the preconditioner is built upon the premise that  $A_U$  is easily invertible. Since  $P_{\mathscr{U}}\mathbf{r}_0 = \mathbf{0}$  by no means implies that  $P_{\mathscr{U}}\mathbf{r}_m^E = \mathbf{0}$ , it cannot be guaranteed that  $\|\mathbf{r}_m^E\| = \|\mathbf{r}_m^M\|$  even for such a special choice of initial residual unless  $A\mathcal{T} = \mathcal{T}$ .

The availability of an (exactly) A-invariant subspace  $\mathcal{U}$ , on the other hand, is an assumption that can rarely be satisfied in practice. In such a case, one can nonetheless still define the preconditioner as above, where now  $A_U := U^*AU$  represents the orthogonal section of A onto  $\mathcal{U}$ , resulting in

$$AM^{-1}[U T] = [U T] \begin{bmatrix} I & U^*AT \\ T^*AUA_U^{-1} & T^*AT \end{bmatrix},$$

based on the heuristic argument that  $T^*AUA_U^{-1}$  will be small whenever  $\mathscr{U}$  is nearly A-invariant. Such nearly A-invariant spaces are obtained as the span of selected Ritz or harmonic Ritz vectors determined from Krylov spaces generated during previous cycles. In practice it is common to suitably scale  $A_U$  in the preconditioner M (see [7]).

Baglama et al. [1] propose a similar algorithm, which preconditions by (48) from the left, leading again under the assumption that  $\mathscr{U}$  is exactly *A*-invariant—to the preconditioned operator

$$M^{-1}A[U\ T] = [U\ T] \begin{bmatrix} I\ A_U^{-1}U^*AT\\ O\ T^*AT \end{bmatrix},$$
$$M^{-1}A = P_{\mathcal{U}} + AP_{\mathcal{F}} + (A^{-1} - I)P_{\mathcal{U}}AP_{\mathcal{F}}.$$

The MR correction of the left-preconditioned system is the solution of the minimization problem

$$||M^{-1}\mathbf{r}_{m}^{B}|| = \min\{||M^{-1}(\mathbf{r}_{0} - AM^{-1}\mathbf{c})||: \mathbf{c} \in \mathscr{K}_{m}(AM^{-1}, \mathbf{r}_{0})\}$$

(cf. [18, p. 255]).

From (48), it is evident that

$$M^{-1} = A^{-1}P_{\mathscr{U}} + P_{\mathscr{T}}$$

and, consequently, if  $A\mathcal{U} = \mathcal{U}$ ,

 $P_{\mathscr{T}}M^{-1}\boldsymbol{v} = P_{\mathscr{T}}\boldsymbol{v}$  for all  $\boldsymbol{v}$ .

These are the essential ingredients for showing that Proposition 4.6 holds in exactly the same way with  $\mathbf{r}_m^E$  in place of  $\mathbf{r}_m^B$ .

The construction of an approximately invariant subspace  $\mathscr{U}$  is accomplished in [1] by employing the IRA process (cf. Section 3.4).

Kharchenko and Yeremin [10] suggest another adaptive right preconditioner  $\tilde{M}$  constructed as follows: After each GMRES cycle the Ritz values and the corresponding left<sup>5</sup> and right Ritz vectors of A with respect  $\mathscr{K}_m$  are extracted. The aim is to obtain a preconditioner such that the extremal eigenvalues of A, which are approximated by the Ritz values, are translated to one (or at least to a small cluster around one).

The extremal Ritz values are partitioned into, say, k subsets  $\Theta_j$  of nearby Ritz values. For each  $\Theta_j$ , a rank-one transformation of the form  $I + v_j \tilde{v}_j^*$  is constructed, where  $v_j$  and  $\tilde{v}_j$  are linear combinations of the associated right and left Ritz vectors. These linear combinations are chosen to translate simultaneously all Ritz values of  $\Theta_j$  into a small cluster around one, while satisfying certain stability criteria. One preconditioning step now consists of successive multiplication by these rank-one matrices, i.e.,

$$\tilde{M}^{-1} = (I + \boldsymbol{v}_1 \tilde{\boldsymbol{v}}_1^*) \dots (I + \boldsymbol{v}_k \tilde{\boldsymbol{v}}_k^*) = I + [\boldsymbol{v}_1 \dots \boldsymbol{v}_k] [\tilde{\boldsymbol{v}}_1 \dots \tilde{\boldsymbol{v}}_k]^*.$$

<sup>&</sup>lt;sup>5</sup> Left Ritz vectors are defined by  $A^* \tilde{z}_j - \bar{\theta}_j \tilde{z}_j \perp \mathscr{K}_m$  and can be obtained from the left eigenvectors of  $H_m$ .
For the last equality we have made use of the fact that  $\tilde{v}_j^* v_i = 0$  for  $i \neq j$ , since all eigenvalues of  $H_m$  have geometric multiplicity one. Note that, if  $\Theta_j$  has a small diameter and the Ritz values contained in  $\Theta_j$  are good approximations of eigenvalues of A, then  $v_j$  and  $\tilde{v}_j$  are approximate right and left eigenvectors of A. It can be shown that the statement made in Theorem 4.6 also holds for this preconditioning approach.

#### 4.5. Optimal truncation

The methods of the preceding sections were based on restarting an MR iteration once the correction space has reached a given dimension m, and attempted to compensate for the attendant loss of information by augmenting or preconditioning. The methods discussed in this section are related to the former in that they also attempt to retain information contained in the current correction space – in this case orthogonality constraints – which is deemed most useful for convergence.

In place of restarting, the basic scheme underlying this class of methods is a *truncated* MR iteration, in which, as soon as the correction space has reached a maximal dimension m, only a subset of the most recent m basis vectors of the correction space is retained, or equivalently, one or more of these basis vectors is periodically discarded during the iteration. In [26] de Sturler proposes a scheme for selectively discarding subspaces rather than individual basis vectors. This selection process, however, does not rely on spectral or invariant subspace information, but rather on angles between subspaces.

To discard a subspace of dimension  $\ell$ , the subspace selection scheme proposed by de Sturler compares two approximation spaces  $\mathscr{W}_1$  and  $\mathscr{W}_2$  associated with correction spaces  $\mathscr{C}_1$  and  $\mathscr{C}_2$ . It assumes the availability of an orthonormal basis  $W_m^{(1)} = [w_1^{(1)}, \ldots, w_m^{(1)}]$  of  $\mathscr{W}_1$ , an *arbitrary* basis  $\hat{W}_k^{(2)} = [\hat{w}_1^{(2)}, \ldots, \hat{w}_k^{(2)}]$  of  $\mathscr{W}_2$  as well as a factorization

$$(I_k - W_m^{(1)}[W_m^{(1)}]^*)\hat{W}_k^{(2)} = Z_k R$$

with  $Z_k = [z_1, ..., z_k]$ ,  $Z_k^* Z_k = I_k$  and  $R \in \mathbb{C}^{k \times k}$  nonsingular and upper triangular. After computing the singular value decomposition

$$([W_m^{(1)}]^* \hat{W}_k^{(2)}) (Z_k^* \hat{W}_k^{(2)})^{-1} = X \Xi \hat{Y}^{\mathrm{H}},$$
(51)

the subspace of  $\mathcal{W}_1$  to be retained is chosen as that spanned by the vectors  $W_m^{(1)}[x_1 \cdots x_\ell]$ , where the vectors  $x_j$  are the left singular vectors associated with the  $\ell$  largest singular values. The following proposition relates this choice to the results of Section 2.4.

**Proposition 4.7.** With the above notation under the assumption  $\mathcal{W}_1 \cap \mathcal{W}_2 = \{\mathbf{0}\}$ , the singular values appearing in (51) are the cotangents of the canonical angles between the spaces  $\mathcal{W}_1$  and  $\mathcal{W}_2$ .

**Proof.** Let  $W_k^{(2)}$  denote an *orthonormal* basis of  $\mathscr{W}_2$  such that  $\hat{W}_k^{(2)} = W_k^{(2)}S$  with a nonsingular matrix  $S \in \mathbb{C}^{k \times k}$ . Then the cosines of the canonical angles between  $\mathscr{W}_1$  and  $\mathscr{W}_2$  are the singular values of  $[W_m^{(1)}]^* W_k^{(2)}$ , and we write the associated singular value decomposition as  $[W_m^{(1)}]^* W_k^{(2)} = X\Gamma Y^{\mathrm{H}}$  with a diagonal matrix  $\Gamma \in \mathbb{R}^{m \times k}$  and the unitary matrices  $X \in \mathbb{C}^{m \times m}$  and  $Y \in \mathbb{C}^{k \times k}$ . From

$$Z_k R = (I_k - W_m^{(1)} [W_m^{(1)}]^*) \hat{W}_k^{(2)} = (I_k - (W_m^{(1)} X) (W_m^{(1)} X)^*) (W_k^{(2)} Y) Y^{\mathrm{H}} S$$
  
=  $[(W_k^{(2)} Y) - (W_m^{(1)} X) \Gamma] Y^{\mathrm{H}} S$ ,

we obtain  $Z_k = [(W_k^{(2)}Y) - (W_m^{(1)}X)\Gamma]Y^{\mathrm{H}}SR^{-1}$  and therefore, defining the diagonal matrix  $\Sigma \in \mathbb{R}^{k \times k}$  by  $I_k - \Gamma^{\mathrm{H}}\Gamma = \Sigma^2$ , there results

$$I_k = Z_k^* Z_k = (SR^{-1})^{\mathrm{H}} Y \Sigma^2 Y^{\mathrm{H}} (SR^{-1}) = (\Sigma Y^{\mathrm{H}} SR^{-1})^{\mathrm{H}} \Sigma Y^{\mathrm{H}} SR^{-1}$$

which reveals that the  $k \times k$  matrix  $\Sigma Y^{H}SR^{-1}$  is also unitary. Note that, in view of  $\mathscr{W}_{1} \cap \mathscr{W}_{2} = \{\mathbf{0}\}$ , none of the cosines in  $\Gamma$  are one, hence  $\Sigma$  is nonsingular. Now, inserting

$$[W_m^{(1)}]^* \hat{W}_k^{(2)} = [W_m^{(1)}]^* W_k^{(2)} S = X \Gamma Y^{\mathrm{H}} S,$$

$$Z_k^* \hat{W}_k^{(2)} = (SR^{-1})^{\mathrm{H}} Y [(W_k^{(2)}Y)^* - \Gamma^{\mathrm{H}} (W_m^{(1)}X)^*] W_k^{(2)} S = (SR^{-1})^{\mathrm{H}} Y \Sigma^2 Y^{\mathrm{H}} S$$

can express the singular value decomposition (51) as

$$([W_m^{(1)}]^* \hat{W}_k^{(2)})(Z_k^* \hat{W}_k^{(2)})^{-1} = X(\Gamma \Sigma^{-1})(\Sigma Y^{\mathrm{H}} S R^{-1}).$$

which reveals that its singular values are indeed the cotangents of the angles between  $\mathcal{W}_1$  and  $\mathcal{W}_2$ .

The proof also shows that the left singular vectors of (51) coincide with those of  $[W_m^{(1)}]^* W_k^{(2)}$ , hence the selection scheme discards that subspace of  $\mathcal{W}_1$  which lies at the largest canonical angles with  $\mathcal{W}_2$ . As shown in Section 2.4, this choice yields the greatest possible residual reduction when replacing the approximation space  $\mathcal{W}_1 + \mathcal{W}_2$  by  $\tilde{\mathcal{W}}_1 + \mathcal{W}_2$  with  $\tilde{\mathcal{W}}_1$  a subspace of  $\mathcal{W}_1$  of dimension  $\dim \mathcal{W}_1 - k$ .

In [26] de Sturler applies this scheme to a GMRES cycle of length *m* in order to determine which directions of the *s*-dimensional Krylov subspace  $\mathscr{K}_s(A, \mathbf{r}_0)$ , s < m, are most important for convergence in the sense that maintaining orthogonality against these directions upon restarting after the first *s* steps results in the greatest residual reduction. The subspaces to be compared are thus  $A\mathscr{K}_s(A, \mathbf{r}_0)$  and  $A\mathscr{K}_{m-s}(A, \mathbf{r}_s)$ . The subspace comparison in this case is particularly inexpensive, as both spaces lie in  $\mathscr{K}_m(A, \mathbf{r}_0)$ , for which the Arnoldi process has computed an orthonormal basis. Hence, the angle computations can be performed in the coordinate space with respect to this basis, and therefore involve only small matrices. For details, we refer to [26].

This subspace selection scheme is further used in [26] to improve the inner-outer iteration algorithm GCRO (see Section 4.1). The resulting method, named GCROT, uses the subspace selection scheme specialized to GMRES to transfer several vectors from the inner to the outer approximation space after each inner iteration cycle. In addition, once the outer approximation space exceeds a maximal dimension, it is truncated by comparing it against the inner approximation space in the manner outlined above.

#### 5. Concluding remark

Having described all these improvements of restarted GMRES of course raises the question of which method one should use in practice. Some of the theoretical statements we have made in this paper required simplifying assumptions which seldom hold in practice. Our results can be viewed

291

as a mathematical justification of why and how these methods work, but need to be supplemented by thorough numerical investigations for realistic applications to yield a complete comparison.

We can, however, make the following statement independently of any numerical evidence: None of the techniques presented here can replace an effective preconditioning strategy, but can sometimes dramatically improve the performance of restarted GMRES when applied to a properly preconditioned linear system.

#### References

- J. Baglama, D. Calvetti, G.H. Golub, L. Reichel, Adaptively preconditioned GMRES algorithms, SIAM J. Sci. Comput. 20 (1998) 243–269.
- [2] A. Chapman, Y. Saad, Deflated and augmented Krylov subspace techniques, Numer. Linear Algebra Appl. 4 (1997) 43–66.
- [3] M. Eiermann, O.G. Ernst, Geometric aspects in the theory of Krylov subspace methods, Acta Numerica, to appear.
- [4] S.C. Eisenstat, H.C. Elman, M.H. Schultz, Variational iterative methods for nonsymmetric systems of linear equations, SIAM J. Sci. Comput. 20 (1983) 345–357.
- [5] J. Erhel, K. Burrage, B. Pohl, Restarted GMRES preconditioned by deflation, J. Comput. Appl. Math. 69 (1996) 303-318.
- [6] V. Faber, T.A. Manteuffel, Necessary and sufficient conditions for the existence of a conjugate gradient method, SIAM J. Numer. Anal. 21 (1984) 352–362.
- [7] R.W. Freund, Quasi-kernel polynomials and convergence results for quasi-minimal residual iterations, in: D. Braess, L.L. Schumaker (Eds.), Numerical Methods of Approximation Theory, Birkhauser, Basel, 1992.
- [8] R.W. Freund, Quasi-kernel polynomials and their use in non-Hermitian matrix iterations, J. Comput. Appl. Math. 43 (1992) 135–158.
- [9] A. Greenbaum, Iterative Methods for Solving Linear Systems, Frontiers in Applied Mathematics, Vol. 17, SIAM, Philadelphia, PA, 1997.
- [10] S.A. Kharchenko, A.Yu. Yeremin, Eigenvalue translation based preconditioners for the GMRES(k) method, Numer. Linear Algebra Appl. 2 (1995) 51–77.
- [11] T.A Manteuffel, J.S. Otto, On the roots of the orthogonal polynomials and residual polynomials associated with a conjugate gradient method, Numer. Linear Algebra Appl. 1 (5) (1994) 449–475.
- [12] R.B. Morgan, Computing interior eigenvalues of large matrices, Linear Algebra Appl. 154–156 (1991) 289–309.
- [13] R.B. Morgan, A restarted GMRES method augmented with eigenvectors, SIAM J. Matrix Anal. Appl. 16 (1995) 1154–1171.
- [14] R.B. Morgan, Implicitly restarted GMRES and Arnoldi methods for nonsymmetric systems of equations, SIAM J. Matrix Anal. Appl. 21 (2000) 1112–1135.
- [15] C.C. Paige, B.N. Parlett, H.A. van der Vorst, Approximate solutions and eigenvalue bounds from Krylov subspaces, Numer. Linear Algebra Appl. 2 (1995) 115–133.
- [16] C.C. Paige, M.A. Saunders, Solution of sparse indefinite systems of linear equations, SIAM J. Numer. Anal. 12 (1975) 617–629.
- [17] Y. Saad, A flexible inner-outer preconditioned GMRES algorithm, SIAM J. Sci. Comput. 14 (1993) 461-469.
- [18] Y. Saad, Iterative Methods for Sparse Linear Systems, Prindle, Weber & Schmidt, Boston, MA, 1996.
- [19] Y. Saad, Analysis of augmented Krylov subspace methods, SIAM J. Matrix Anal. Appl. 18 (1997) 435-449.
- [20] Y. Saad, Further analysis of minimumn residual iterations, Technical Report UMSI-97-14, University of Minnesota Supercomputer Institute, 1997.
- [21] Y. Saad, M.H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Comput. 7 (1986) 856–869.
- [22] D.C. Sorensen, Implicit application of polynomial filters in a k-step Arnoldi method, SIAM J. Matrix Anal. Appl. 13 (1992) 357–385.
- [23] G.W. Stewart, Matrix Algorithms, Vol. I, Basic Decompositions, SIAM, Philadelphia, PA, 1998.

- [24] E.L. Stiefel, Kernel polynomials in linear algebra and their numerical applications, J. Res. Nat. Bur. Standards Appl. Math. Ser. 49 (1958) 1–22.
- [25] E. de Sturler, Nested Krylov methods based on GCR, J. Comput. Appl. Math. 67 (1996) 15-41.
- [26] E. de Sturler, Truncation strategies for optimal Krylov subspace methods, SIAM J. Numer. Anal. 36 (1999) 864-889.
- [27] H.A. van der Vorst, C. Vuik, GMRESR: a family of nested GMRES methods, Numer. Linear Algebra Appl. 1 (1994) 369–386.



JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

Journal of Computational and Applied Mathematics 123 (2000) 293-306

www.elsevier.nl/locate/cam

# Refining an approximate inverse $\stackrel{\text{tr}}{\sim}$

Robert Bridson<sup>a,\*</sup>, Wei-Pai Tang<sup>b</sup>

<sup>a</sup>SCCM Gates 2B, Stanford University, CA 94305, USA <sup>b</sup>Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1

Received 29 July 1999; received in revised form 10 September 1999

#### Abstract

Direct methods have made remarkable progress in the computational efficiency of factorization algorithms during the last three decades. The advances in graph theoretic algorithms have not received enough attention from the iterative methods community. For example, we demonstrate how symbolic factorization algorithms from direct methods can accelerate the computation of a factored approximate inverse preconditioner. For very sparse preconditioners, however, a reformulation of the algorithm with outer products can exploit even more zeros to good advantage. We also explore the possibilities of improving cache efficiency in the application of the preconditioner through reorderings. The article ends by proposing a block version of the algorithm for further gains in efficiency and robustness. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Approximate inverse; Cache; Ordering; Performance; Preconditioner; Symbolic factorization

### 1. Introduction

So far research into sparse approximate inverse preconditioners has focused on convergence, ignoring more subtle efficiency issues for the most part. This paper explores how to get the best performance out of an approximate inverse preconditioner, particularly on modern superscalar work-stations.

The algorithm we turn our attention to is Benzi and Tuma's AINV [2,3], or more specifically, a slight variation on the stabilized version SAINV [1] that is guaranteed to avoid breakdown for positive-defininte problems. We previously explored the issue of ordering in [7], noting that for good orderings the set-up time for the preconditioner can be reduced dramatically. Here we go into details on that and other techniques for boosting the performance of the method. We note that in [2,3],

\* Corresponding author.

 $<sup>^{\</sup>ddagger}$  This work was supported by the Natural Sciences and Engineering Council of Canada, and Communications and Information Technology Ontario (CITO), funded by the Province of Ontario.

E-mail addresses: rbridson@stanford.edu (R. Bridson), wptang@elora.uwaterloo.ca (W.-P. Tang).

Benzi and Tuma had already implemented the symbolic factorization enhancement and outer-product form below, though do not discuss it in depth.

Before proceeding, we introduce some notation. Column *i* of a matrix *A* is written  $A_i$  (and so row *i* is  $(A^T)_i^T$ ). The *j*th entry of a vector *v* is given by  $v_j$  (and so the (i, j)th entry of a matrix is indeed  $A_{ij}$ ). The column vector of all zeros except for the one at the *i*th position is  $e_i$ :  $e_i$  is thus column *i* of the identity matrix *I*. In algorithms,  $x \leftarrow y$  indicates that variable *x* is assigned the value of *y*.

# 2. Basic implementation

The simplest form of SAINV is a left-looking, inner-product-based algorithm, given in Algorithm 1. It can be viewed as the generalization of classical Gram–Schmidt<sup>1</sup> to constructing A-biconjugate sets of vectors from the standard basis, with small entries dropped to preserve sparsity. The results are two upper-triangular matrices W and Z containing the sets of vectors as columns and a diagonal matrix D with  $W^T AZ \approx D$ . (In fact, with the presented choice of dropping, the diagonal of  $W^T AZ$  is exactly D – it is just the off-diagonal terms that might not be zero.) When A is symmetric, the algorithm can be simplified by eliminating the W computations, using W = Z.

Algorithm 1. The left-looking, inner-product form of SAINV

- Take A, an  $n \times n$  matrix, and some drop tolerance  $\delta \ge 0$  as input.
- For i = 1, ..., n
  - $\triangleright$  Initialize columns i of W and Z to the i'th standard basis vector
  - Set  $W_i \leftarrow e_i$  and  $Z_i \leftarrow e_i$ .
  - ▷ Make column i of W biconjugate with previous columns
  - Get row *i* of A:  $r \leftarrow (A^{\mathrm{T}})_i^{\mathrm{T}} = e_i^{\mathrm{T}} A$ .
  - For j = 1, ..., i 1
    - Set  $W_i \leftarrow W_i (rZ_j/D_{jj})W_j$
  - ▷ Make column i of Z biconjugate with previous columns
  - Get column *i* of  $A: c \leftarrow A_i = Ae_i$ .
  - For j = 1, ..., i 1

• Set 
$$Z_i \leftarrow Z_i - (W_i^{\mathrm{T}} c/D_{ii})Z_i$$

- ▷ Drop small entries to keep W and Z sparse
- Zero any above-diagonal entry of  $W_i$  or  $Z_i$  with magnitude  $\leq \delta$ .
- $\triangleright$  Find the "pivot"  $D_{ii}$
- Set  $D_{ii} \leftarrow W_i^{\mathrm{T}} A Z_i$ .
- Return W, Z, and D.

Of course, all the matrices should be stored in sparse mode. For this article, compressed column storage format is assumed: each matrix is a collection of n sparse column vectors.

However, the inner products  $rZ_j$  and  $W_j^Tc$  are more efficiently computed if one of the vectors is stored in full mode; while a sparse–sparse operation could theoretically be faster, a typical

<sup>&</sup>lt;sup>1</sup>SAINV in [1] is actually a generalization of modified Gram–Schmidt; this variation is a slightly faster but typically equal quality algorithm.

implementations more complicated branching and memory accesses make it slower on today's hardware. Since each r and c is reused over many inner iterations, it is natural to keep these in full storage – though of course, there is the drawback that often W and Z will be denser than A, so the inner products would be even more efficient with  $W_i$  and  $Z_i$  in full storage.

One *n*-vector suffices to store both *r* and *c*. To avoid unnecessary  $O(n^2)$  work, it should not be completely zeroed out after use: only the nonzero locations should be reset. Further unnecessary work can be eliminated by only copying nonzeros up to position i - 1, since *W* and *Z* are upper triangular and thus locations from *i* onwards will not be involved in the inner products.

With compressed column storage, accessing each column c is simple, but finding each row r is more time-consuming. In the symmetric case, this is of course unnecessary. Even if A just has symmetric structure, r can be found faster since not every column of A need be checked for a nonzero at position i: only those columns corresponding to nonzeros in column i.<sup>2</sup>

The updates to  $W_i$  and  $Z_i$  require some thought, as they should be done in sparse mode; if constructed as dense vectors, there will be unnecessary O(n) work in every iteration to gather them up into sparse storage. If the sparse columns are not kept in sorted order, the simplest way of adding the scaled  $W_j$  to  $W_i$  (or  $Z_j$  to  $Z_i$ ) is to do a linear search in  $W_i$  for each nonzero in  $W_j$ ; if there is a nonzero already in that location, add it to it, and otherwise append it. If the columns are sorted, then a faster merge operation may be used instead.

However, both of these methods require time depending on the number of nonzeros already in  $W_i$  (some fraction of the elements in  $W_i$  will be scanned to determine where to add the update), which may grow with each inner iteration as updates are applied. A better implementation is described below, adding the scaled  $W_j$  to  $W_i$  in time just proportional to the number of nonzeros in  $W_j$ , independent of how many are already in  $W_i$  (avoiding any scan of existing elements).

Maintain two *n*-vectors, exists and location. The former is a Boolean vector with exists (k) true when  $W_i$  has a nonzero in position k; then location (k) points to where that nonzero is stored in the sparse data structure. Now adding an entry from the scaled  $W_j$  to  $W_i$ , say at position k, takes O(1) time: look-up exists (k); if true use location (k) to modify the existing entry in  $W_i$ , otherwise append the new entry to  $W_i$ . If the vectors must be stored in sorted order, after the inner loop  $W_i$  can be radix or bin-sorted very efficiently.

Of course, exists must be reset to all false before each inner loop. A cheap method to avoid this cost is to let exists (k) = i indicate true for  $W_i$ , and n + i true for  $Z_i$ , on the *i*'th iteration.

The calculation of the pivot  $W_i^T A Z_i$  is best done with  $W_i$  in full storage, viewing it as a sum of full-sparse inner products:

$$W_i^{\mathrm{T}}AZ_i = \sum_{Z_{ii} \neq 0} (W_i^{\mathrm{T}}A_j)Z_{ji}.$$

Thus, after small entries have been dropped,  $W_i$  should be scattered into a full *n*-vector, and after the pivot has been calculated, only those nonzeros reset.

<sup>&</sup>lt;sup>2</sup> It is also possible to store a row-oriented copy of A along side the column-oriented version, as is done in [3]; for the scalar case here we have chosen not to, trading higher complexity for more lower storage requirements. Experiments indicate that typically the row-oriented copy is only worthwhile when A is not structurally symmetric, but then essentially the same performance can be obtained by adding zeros to symmetrize the structure, as will be discussed later.

#### **3. Fruitless inner products**

Even with good handling of the sparse vs. dense issues, the algorithm as it stands must take at least  $O(n^2)$  time due to the nested loops. This can be improved significantly after realizing that often many of the inner *j* iterations are unnecessary: the inner products  $rZ_j$  and  $W_j^Tc$  are often zero simply because there are no nonzero locations in common.

Table 1 shows some sample statistics of how many inner products turn out to be exactly zero in the preconditioner construction for some typical test matrices, symmetrically ordered with the nested dissection routine from Metis [13].<sup>3</sup> This does not include the small fraction of inner products from the pivot calculation  $W_i^T AZ_i$ .

Fortunately, many of these inner products can be avoided. We begin by considering those inner products which are zero even without small entries dropped in the algorithm, i.e., when the true inverse factors are computed. Because the algorithm does not rely on cancellation anywhere, dropping can only result in more zero dot products – thus, we are always safe to avoid the ones that are zero without dropping.

First, consider the case when A has symmetric structure, so the true inverse factors have the same structure as each other. Then we have the following result:

**Theorem 3.1.** Assuming symmetric structure, at step *i* with *r* equal to the *i*th row of *A*, the inner product  $rZ_j \neq 0$  only if j < i and *j* is an ancestor in the elimination tree [16] of some *k* with  $A_{ik} \neq 0$ .

**Proof.** In [7] the structure of the true inverse factors, assuming no felicitous cancellation, was shown:  $Z_{kj} \neq 0$  if and only if k is a descendent of j in the elimination tree of A. The inner product  $rZ_j$  is nonzero if and only if there is some k with  $A_{ik} \neq 0$  and  $Z_{kj} \neq 0$ . Therefore, the inner product is

Table 1

When SAINV with drop tolerance 0.1 is applied to several standard test matrices, almost all the inner products are exactly zero. The ordering in all cases is nested dissection

Matrix	Total number of inner products	Number that are exactly zero	Percentage of total (%)	
ADD32	24 596 640	24 580 218	99.9	
BCSSTK25	238.347.282	237.781.980	99.8	
MEMPLUS	315,328,806	315,240,589	99.97	
NASA2146	4,603,170	4,464,828	97.0	
ORSREG1	4,859,820	4,838,964	99.6	
PORES2	1,496,952	1,484,637	99.2	
SHERMAN2	1,165,320	1,118,651	96.0	
SHERMAN3	25,045,020	25,013,829	99.9	
WATSON5	3,431,756	3,421,143	99.7	

<sup>&</sup>lt;sup>3</sup> In [4,7] other orderings were considered for AINV, but as nested dissection is generally close to best in convergence, often best in construction time, and most easily parallelized, this article sticks with just nested dissection. Results for other inverse factor fill reducing orderings are similar.

nonzero only when there is some k with  $A_{ik} \neq 0$  and k a descendent of j, i.e., j an ancestor of k. Only values j < i are considered in the original loop, and so the result follows.  $\Box$ 

Another proof of this result can be made from the factorization A = LDU (with L unit lower triangular, U unit upper triangular, and D diagonal), so  $Z = U^{-1}$  and thus AZ = LD. Then the inner product  $rZ_j$  at step i is simply  $L_{ij}D_{jj}$ , and the nonzero structure of each row of L has been characterized precisely as above in [15]. The only difficulty with this route is determining what role cancellation plays in the structure of AZ – with inexact arithmetic and especially with dropping, it is not immediately clear that the structure of the lower triangle of AZ will be a subset of the structure of L.

In [15] a very efficient algorithm is given for finding the elimination tree of A, leading to a fast symbolic factorization. We can use this to create a symbolic factorization enhanced AINV, replacing the inner j=1,...,i-1 loop with one just over the nonzeros in row i of L. Of course, taking note of the symmetric structure and column-oriented storage of A, the upwards-traversals of the elimination tree to find those indices should start with the nonzeros in column i of A with indices less than i.

When A does not have symmetric structure, things get a little more complicated. Often A is close to structurally symmetric and so ordering, symbolic factorization, and biconjugation can all be done efficiently with zeros inserted into the sparsity structure to make it symmetric. However, there may be cases when it is best to exploit the nonsymmetric zeros in any or all of these steps. (For example, it may be possible to exploit unsymmetric zeros in ordering to reduce the matrix to block triangular form, in which case only smaller submatrices need be preconditioned.) Here we will consider an unsymmetric symbolic factorization enhancement.

The key again is the structure of the true inverse factors. This is most easily discussed with the language of graph theory, where the nonzero structure of an  $n \times n$  matrix M corresponds to a graph  $G_M$  with vertices labelled 1,...,n and directed edge  $i \rightarrow j$  if and only if  $M_{ij} \neq 0$ . See [10], for example, for more discussion of graph theory and sparse matrix computations.

As proven in [12], the inverse of a matrix M has the structure of the transitive closure  $G_M^*$  of  $G_M$ , that is a graph  $G_M^*$  with a directed edge  $i \to j$  whenever there is a path from i to j in  $G_M$ . The simplest characterization of the structure of the true inverse factors  $W^T = L^{-1}$  and  $Z = U^{-1}$  is then as the transitive closures of the graphs of L and U, respectively. However, there are many unnecessary edges in  $G_L$  and  $G_U$  from this standpoint – if an edge  $i \to j$  exists alongside a disjoint path from i to j, the edge  $i \to j$  may be deleted without effecting the transitive closure. If all such redundant edges are deleted, the result is called the transitive reduction. If A was structurally symmetric, this turns out to be the elimination tree mentioned above [16]; otherwise,  $G_L$  and  $G_U$  reduce to a pair of directed acyclic graphs called elimination dags [11].

Unfortunately, the elimination dags can be fairly expensive to compute, and so somewhat denser but cheaper graphs, intermediate between the triangular factors and their transitive reductions, have been investigated in [9]. For this application, an alternative route is to use graphs whose transitive closures contain the structures of  $W^T$  and Z but may be a little denser still – for example, the elimination tree of the symmetrized A. With these cases in mind, the unsymmetric generalization of the previous theorem is:

**Theorem 3.2.** Let  $G_L^{\circ}$  and  $G_U^{\circ}$  be directed acyclic graphs whose transitive closures contain the structures of  $W^{\mathsf{T}}$  and Z, respectively. Then at step *i* of AINV, the inner-product  $rZ_j \neq 0$  only

if j < i and there is a path in  $G_U^o$  to j from some k with  $A_{ik} \neq 0$ ; similarly, the inner-product  $W_i^T c \neq 0$  only if j < i and j is reachable in  $G_L^o$  from some k with  $A_{ki} \neq 0$ .

**Proof.** We will only prove the  $rZ_j$  part, as the  $W_j^{T}c$  part is essentially the same. Since the transitive closure of  $G_U^{\circ}$  contains the structure of Z,  $Z_{kj} \neq 0$  only if there is a path in  $G_U^{\circ}$  from k to j. The inner-product  $rZ_j \neq 0$  if and only if there is some k with  $A_{ik} \neq 0$  and  $Z_{kj} \neq 0$ . Therefore, the inner-product is nonzero only if there is some k with  $A_{ik} \neq 0$  and with a path to j in  $G_U^{\circ}$ . Only values j < i are considered in the original loop, and so the result follows.

Just as before, this can be interpreted as symbolic factorization, if  $G_L^o$  and  $G_U^o$  are chosen to be the elimination dags or other intermediate structures between the elimination dags and the triangular factors. For example, the inner-product  $rZ_j$  at step *i* is just  $L_{ij}D_{jj}$ , and the above characterization is the same as that shown for the rows of *L* in [11,9].

Table 2 compares the regular form of AINV with the symbolic factorization enhanced version, with a drop tolerance of 0.1 for each test matrix as before. The timing counts are from a C implementation running on an Apple Macintosh workstation with a 233 MHz Power PC 750 processor. For the matrices with nonsymmetric structure an elimination dag version is tested first, followed by a symmetrized version. Even without the time required for finding the elimination dags taken into account, and even though more unnecessary zero inner products are performed, the symmetrized version is clearly much faster for these typical matrices. In all cases, the enhanced algorithm is significantly faster than the original algorithm, often by an order of magnitude or more.

For a successful ordering, the number of nonzeros in the *LDU* factors, hence the number of inner products in the symbolic factorization enhanced algorithm, is an order of magnitude less than  $O(n^2)$  (see, e.g., [14] for guarantees on two-dimensional finite element meshes). Assuming that the average

1 4010 2	Tal	ble	2
----------	-----	-----	---

A comparison of regular and symbolic factorization enhanced SAINV on some standard test matrices. The matrices marked as "symmetrized" had zeros inserted in their sparsity structure to make them structurally symmetric, albeit not numerically symmetric

Matrix	Millions of inner products		Percentage zero inner	e of products	Seconds spent on AINV	
	Regular	Enhanced	Regular	Enhanced	Regular	Enhanced
ADD32	25	0.02	99.9	15.5	15.2	0.04
BCSSTK25	238	1.57	99.8	82.0	730	6.8
MEMPLUS	315	0.11	99.97	18.9	310	44.5
(symmetrized)		0.11		21.4	260	0.42
NASA2146	4.6	0.14	97.0	50.2	7.7	0.31
ORSREG1	4.9	0.17	99.6	87.6	3.4	0.20
PORES2	1.5	0.06	99.2	78.4	1.2	0.28
(symmetrized)		0.09		86.0	1.1	0.07
SHERMAN2	1.2	0.12	96.0	60.3	1.8	0.71
(symmetrized)		0.16		70.1	1.4	0.50
SHERMAN3	25	0.20	99.9	84.3	18.1	0.34
WATSON5	3.4	0.02	99.7	50.8	3.7	0.46
(symmetrized)		0.09		88.8	3.7	0.08

cost of an inner-product in the regular and the enhanced algorithms is the same – which is probably not strictly true, but still is a good rough estimate – this explains why the enhanced version is so much faster.

### 4. Revisiting the outer-product form

The symbolic factorization enhancement may avoid all inner products that can be determined zero a priori. However, there are still more that result from the nonzeros that are dropped during the algorithm. Possibly, these could be avoided by pruning the elimination structures as the algorithm goes, but a simpler approach is to rewrite SAINV as a right-looking outer-product algorithm by switching the order of the loops. With the obvious sparsity enhancement, the result is given in Algorithm 2.

Algorithm 2. The outer-product form of SAINV

- Take as input A and  $\delta$ .
- Set  $W \leftarrow I$  and  $Z \leftarrow I$ .
- For j = 1, ..., n
  - Set  $l \leftarrow AZ_j$
  - Set  $u \leftarrow W_i^{\mathsf{T}} A$
  - Set  $D_{jj} \leftarrow uZ_j$  or equivalently  $W_j^{\mathrm{T}}l$ , whichever is cheapest
  - For i > j,  $l_i \neq 0$ 
    - Update  $W_i \leftarrow W_i \operatorname{drop}((l_i/D_{jj})W_j, \delta)$ , where entries of the update vector with magnitude  $\leq \delta$  are dropped.
  - For i > j,  $u_i \neq 0$ 
    - Update  $Z_i \leftarrow Z_i \operatorname{drop}((u_i/D_{ij})Z_i, \delta)$ .
- Return W, Z, and D.

In exact arithmetic without dropping, the vectors l and u at step j are the jth column and row of LD and DU, respectively. With dropping, they naturally become sparser, giving the improvement over the symbolic factorization enhanced inner-product algorithm.

Note that because small entries are dropped before being added in this formulation, the result will in general be different from the inner-product version. Usually, the same drop tolerance will produce a sparser but less accurate preconditioner than the inner-product form.

The primary drawback of the outer-product form is its right-looking nature: all of columns j + 1, ..., n of W and Z must be stored in dynamic data structures, since updates to them may insert entries in any row up to j. The natural implementation with each column of W and Z in a sorted linked list then can suffer from inefficient insertions, poor cache usage, and difficulties for vectorization.<sup>4</sup> However, the savings from exploiting the dropped zeros hopefully can make up for this.

<sup>&</sup>lt;sup>4</sup> More sophisticated data structures such as B-trees may do better in some cases – normally though, the number of nonzeros in each column of W and Z is so small that the increased overhead is not worth it.

Just as with the inner-product form, there is a difficulty when A does not have symmetric structure and a row-oriented copy is not available: the left-multiplication  $W_j^T A$  cannot be made in an efficient fully sparse mode. All entries must be computed, even though most will be zero. One possibility to speed this up is to use a similar symbolic factorization approach as before, making use of a characterization of the columns of L (rather than the rows) to a priori eliminate most of the zero computations. However, this would lose the motivation for the outer-product form – exploiting the zeros that cannot be determined a priori – while still incurring the dynamic data structure penalties. Therefore, we have chosen to symmetrize the sparse matrix data structure as before.

A timing comparison between the inner-product and outer-product algorithms is given in Table 3. Since the same drop tolerance of 0.1 produces slightly different factors for the outer-product form than for the inner-product form, I have chosen new drop tolerances for the outer-product tests to give it roughly the same number of nonzeros.

As the results show, while for some problems the extra overhead of outer-product SAINV is not worth the small gain made from exploiting the full sparsity, in several cases the benefit is considerable.

With these results in mind the choice of algorithm depends on several factors:

- How much storage is available? Enough for the overhead of the dynamic data structures in the outer-product form? Enough for an additional row-oriented copy of *A*?
- Approximately how full will the factors be? Full enough that there will be so few zero inner products that inner-product AINV is faster?
- Is *A* so far from structurally symmetric that it pays to exploit the unsymmetric zeros in some way (e.g., using unsymmetric elimination structures for inner-product SAINV rather than the elimination tree of the symmetrized matrix)?

It should be noted also that for some problems, A is known only as an operator or as a product of matrices, not in explicit matrix form. In this case, finding elimination structures for A may be impossible, prompting the choice of the outer-product form which does not require them (for example, see [6]).

Matrix	Time for inner-product form	Time for outer-product form
ADD32	0.04	0.06
BCSSTK25	6.75	0.97
MEMPLUS	0.42	0.55
NASA2146	0.31	0.17
ORSREG1	0.20	0.06
PORES2	0.07	0.04
SHERMAN2	0.50	0.44
SHERMAN3	0.34	0.10
WATSON5	0.08	0.11

Table 3

Timing comparison for inner-product SAINV versus outer-product SAINV. Matrices with unsymmetric structure are symmetrized with additional zeros

#### 5. Ordering for application

Forgetting the trivial diagonal matrix D for the time being, the basic operation in an iterative solver is applying the preconditioned operator to a dense vector:  $W^{T}AZx$ . Algorithm 3 shows the simplest algorithm for doing this with compressed column storage.

One major issue in the speed of this algorithm on modern superscalar processors comes from the memory hierarchy: efficient cache usage. For example, in the first main loop (multiplying u = Zx) each entry of u may be accessed several times according to the structure of Z. The more cache misses there are – the more times an entry of u has to be fetched from main memory – the slower the loop will run. Ideally, once an entry from u is fetched from cache it will stay there until done with, and will not be prematurely bumped out of the cache. The situation is complicated by how entire cache "lines" of consecutive memory locations are brought into cache at each miss – typically on the order of 64 bytes.

# Algorithm 3. Multiplying $W^{T}AZx$

- Take as input sparse matrices W, A, and Z (in compressed column format) and a dense vector x.
- Initialize dense vectors u = 0 and v = 0
- For i = 1, ..., n
  - For *j* with  $Z_{ii} \neq 0$ 
    - Update  $u_i \leftarrow u_i + Z_{ii}x_i$
- For i = 1, ..., n
  - For *j* with  $A_{ii} \neq 0$ • Update  $v_i \leftarrow v_i + A_{ii}u_i$
- For i = 1, ..., n
  - Set  $u_i \leftarrow 0$
  - For *j* with  $W_{ji} \neq 0$ 
    - Update  $u_i \leftarrow u_i + W_{ii}v_i$
- Return the result in *u*.

One of the advantages of approximate inverses is that any orderings may be used in the matrixvector multiplies – the rows and columns of the matrices and vectors may be permuted without effecting the result, modulo finite precision arithmetic errors, with the only restriction coming from the compatibility of the orderings in multiplication (e.g., the ordering of x must be the same as the columns of Z). With detailed knowledge of the hardware hopefully this can be exploited to promote efficient cache usage in the multiplies. Such tuning is beyond the scope of this article, but some simple tests can show the potential effect of ordering for application. We hope to raise questions here, rather than provide answers.

Table 4 compares the performance for random orderings, the nested dissection ordering used in construction of the preconditioner, and a reordering of the elimination tree for the nested dissection ordering starting at the leaves and progressing upwards level by level. This last ordering is an entirely equivalent elimination sequence to the nested dissection, but mimics the greedy choices made by minimum degree or MIP [7].

#### Table 4

The number of milliseconds taken to compute  $W^T AZx$  for various orderings of the matrices and x. W and Z are computed from inner-product SAINV with a drop tolerance of 0.1 and the nested dissection ordering. The leaf reordering is an equivalent elimination sequence to the nested dissection, but begins with all the leaves of the elimination tree and progresses upwards level by level, mimicking minimum degree and MIP to some extent

Matrix	Ordering						
	Random	Nested dissection	Leaf reordering				
ADD32	5.6	4.7	4.8				
BCSSTK25	102.3	70.6	91.6				
MEMPLUS	33.6	7.4	28.6				
NASA2146	13.0	12.4	12.7				
ORSREG1	2.3	2.6	2.2				
PORES2	1.3	1.3	1.3				
SHERMAN2	4.2	4.3	4.4				
SHERMAN3	7.9	6.6	6.9				
WATSON5	3.7	3.4	3.8				

The differences in performance, at least for ADD32, BCSSTK25, MEM-PLUS, and SHERMAN3, highlight how important ordering might be here. Random ordering is clearly bad – indicating for example that unstructured meshes created with no natural ordering should be appropriately reordered for iterations. The standard nested dissection is generally better than the elimination tree equivalent leaf reordering, perhaps indicating that if minimum degree or MIP is used for construction a further reordering is necessary. We believe the reason for these differences is that standard nested dissection tends to cluster most of the nonzeros in small blocks (excepting the large block separators), which intuitively will allow efficient cache usage. We note that other factors may be involved, such as how fully used are multiple instruction pipelines, but for the moment we do not see a reason they would have this effect; we refer the reader to [8] for a full discussion of all the factors involved in tuning a (dense) matrix-multiplication routine.

The question remains whether there are significantly superior orderings to standard nested dissection. The following theorem suggests that for fairly full approximate inverses, the nested dissection ordering could well be the best. In general, for symmetrically structured matrices, a post-ordering of the elimination tree [10] is a natural generalization of the nested dissection ordering even when the ordering was not constructed in that manner.

**Theorem 5.1.** For symmetrically structured A with a post-ordering of the elimination tree [10], the true upper triangular inverse factor has a dense skyline. In other words, its columns consist of a block of zeros followed by a single dense block of nonzeros ending at the diagonal.

**Proof.** The key characterization of a post-ordering is that any subtree is ordered in a contiguous block, with the root of the subtree coming last. The nonzeros in column *i* of the true upper triangular inverse factor correspond to all children of *i* in the etree, i.e., to the others nodes in the subtree rooted at *i*. Thus, the nonzeros form one contiguous block, ending at the diagonal (the *i*th row).  $\Box$ 

This simple block structure is near optimal for cache use within each column, though the question of the order in which the columns should be considered is still open.

#### 6. Block methods

As with the direct methods, the eventual goal of the algorithms should be to cut the symbolic operations to a minimum while doing the necessary numerical operations efficiently in cache. We have shown the ways to eliminate unnecessary numerical operations in the preconditioner construction, and the possibility of promoting cache usage in the application. For further improvements we now turn to block methods to cut down symbolic operations and further cache efficiency.

This approach, partitioning A into small dense block matrices, is used to great advantage in direct methods, where for example supernodes [17] are used to eliminate redundant symbolic operations. There are also many problems, e.g., from systems of PDEs, that naturally have a block structure and it sometimes makes sense to treat them as such: convergence may sometimes be improved, as shown below.

The generalization of SAINV to block matrices is straightforward. We redefine our notation somewhat for block structures. Throughout we assume that A and all other matrices have been partitioned with  $1 = b_1 < b_2 < \cdots < b_{m+1} = n + 1$ . Then  $A_i$  indicates block column i of A, consisting of the "point" columns  $b_i$  to  $b_{i+1} - 1$ , and  $A_{ij}$  indicates the jth block in this block vector, the submatrix of A extending from position  $(b_i, b_j)$  to  $(b_{i+1} - 1, b_{j+1} - 1)$ . The *i*th block column of the identity is given by  $E_i$ . Notice that diagonal blocks of a matrix are necessarily square, but off-diagonal blocks might not be if the block size is not constant.

Block SAINV produces matrices W, Z, and D that approximately satisfy  $W^T A Z = D$ , where W and Z are block upper triangular and D is block diagonal. The inner-product form is given in Algorithm 4; for this paper we do not explore the performance of the somewhat more complicated outer-product form. The generalization of the scalar outer-product algorithm is straightforward nonetheless.

Algorithm 4. The left-looking, inner-product form of block SAINV with symbolic factorization enhancement

- Take A, an  $m \times m$  block matrix, and some drop tolerance  $\delta \ge 0$  as input.
- For i = 1, ..., m
  - ▷ Initialize block columns i of W and Z to the i'th standard basis block vector
  - Set  $W_i \leftarrow E_i$  and  $Z_i \leftarrow E_i$ .
  - Get block row *i* of  $A : R \leftarrow (A^T)_i^T = E_i^T A$  (up to column i 1)
  - For j < i, U<sub>ji</sub> ≠ 0 (determined by symbolic factorization)
     W<sub>i</sub> ← W<sub>i</sub> − W<sub>j</sub>(RZ<sub>j</sub>D<sup>-1</sup><sub>ji</sub>)<sup>T</sup>
  - Get block column *i* of  $A : C \leftarrow A_i = AE_i$  (up to row i 1)
  - For j < i, L<sub>ij</sub> ≠ 0 (determined by symbolic factorization)
     Z<sub>i</sub> ← Z<sub>i</sub> − Z<sub>j</sub>(D<sup>-1</sup><sub>ji</sub>W<sup>T</sup><sub>i</sub>C)
  - Zero any above-diagonal block of  $W_i$  or  $Z_i$  with norm  $\leq \delta$ .
    - Set  $D_{ii} \leftarrow W_i^{\mathrm{T}} A Z_i$  (and store  $D_{ii}^{-1}$ ).
- Return W, Z, and D.

The symbolic factorization enhancement now must use the graph of the block form of A, where each vertex represents a diagonal block and each edge a nonzero off-diagonal block. Also notice that since the storage requirements of a sparse block matrix is strongly dominated by the numerical entries in the dense blocks, it is perfectly reasonable to store a row-oriented version of the sparsity structure (referencing the same numerical entries) along side the column-oriented version – so finding block rows of unsymmetric A can be done with ease.

Determining when to drop "small" blocks from W and Z is an interesting issue, especially as one drop tolerance is used for blocks of potentially different sizes. One possibility, used here, is to compare the Frobenius norm of the block divided by the number of entries in the block against the drop tolerance  $\delta$ .

In the scalar case, it is possible that a pivot will be zero (or small enough to cause problems when dividing by the pivot later). This problem is alleviated somewhat with the block algorithm, since the inversion of the block pivots can be carried out more robustly with a partial pivoting LU decomposition, a QR decomposition, or even an SVD operation to be completely confident of numerical stability. However, it still may happen that a block pivot is singular or so close to singular that problems emerge. Our implementation currently only checks for exact zeros in the partial pivoting LU decomposition; this never happenned in the testing however. Several possibilities exist for recovery if this does happen – adding a diagonal shift to the block, for example, or using a shifted SVD instead.

Some of the test problems have a natural block structure while it is not so clear for others. One possibility is to use the supernodes following a nested dissection ordering, hoping that since the nodes making up a supernode have essentially the same structure in the true inverse factors, they should have similar structure in the approximate inverse and thus be correctly handled with dense blocks. The problem is that usually many supernodes are singletons, so unless care is taken in coding the algorithm, the overhead of the block algorithm is wasted. It is also important to note that there are typically some supernodes of very large size, which must be broken up into more manageable sizes for storage and computational efficiency.

Perhaps, a better approach is to use the aggregation algorithms of algebraic multigrid. Here we tried applying the ideas from [5] (using  $|A| + |A^T|$  for the unsymmetric matrices).

Table 5 shows construction times and convergence rates for scalar inner-product AINV and block inner-product AINV, with the same drop tolerance of 0.1 as before. For these examples, the block form is always slower – the overhead simply is not worth any gains in dense operations. It should be noted that the BLAS and LAPACK libraries used for these timings were not highly tuned, however, so better results are definitely anticipated in better implementations. The convergence is generally worse for the block method, presumably because the block version may drop important nonzeros in otherwise near zero blocks while retaining unimportant nonzeros that happen to occur in the same blocks as important nonzeros. The exception is SHERMAN2, where as suggested in [7] the block form succeeds but the scalar form fails.

It seems then that the block version might only be appropriate in certain cases, unless a better determination of blocks and a more sophisticated dropping strategy are adopted. For example, the improvement for SHERMAN2 over the scalar version is probably because the scalar version's simple diagonal pivoting is inappropriate – with weak diagonals and condition numbers ranging from  $10^7$  to  $10^{11}$ , the diagonal blocks require partial pivoting to be inverted. (For the other matrices, the diagonal

#### Table 5

A comparison of preconditioner construction times and convergence rates. The drop tolerance for scalar inner-product AINV is 0.1, and the drop tolerance for the block version is chosen to give approximately the same number of nonzeros. CG is used for s.p.d. problems and BiCGstab for the rest; the right-hand side is the vector of all ones, the initial guess is all zeros, and convergence is flagged when the residual 2-norm is decreased by a factor of  $10^6$ 

Matrix	Scalar		Block					
	Time for AINV	Iterations	Average block size	Time for AINV	Iterations			
ADD32	0.04	5	2	0.10	32			
BCSSTK25	6.75	$\infty$	3.0	7.88	$\infty$			
MEMPLUS	0.42	17	2	0.58	$\infty$			
NASA2146	0.31	85	3.9	0.30	135			
ORSREG1	0.20	31	2.5	0.31	46			
PORES2	0.07	$\infty$	2	0.20	$\infty$			
SHERMAN2	0.50	$\infty$	6	0.57	21			
SHERMAN3	0.34	96	1.7	1.13	127			
WATSON5	0.08	127	2.7	0.13	$\infty$			

blocks are not nearly as badly conditioned.) Of course, this raises the question whether a simple block diagonal rescaling applied before scalar AINV would be enough to cure the problem.

#### 7. Conclusions

We have presented several refinements for improving the performance of the SAINV-factored approximate inverse. Ideas and algorithms from direct methods allowed significant performance enhancements for the inner-product form of the algorithm; for many problems, however, even faster construction was possible with an outer-product reformulation. Experimental results demonstrated how reordering the approximate inverse can greatly effect the cache efficiency during its application in an iterative solver. We finally proposed a block version of the algorithm for further gains in cache efficiency, which unfortunately are off-set by increased overhead in the current implementation – we expect further tuning of the code will make block processing worthwhile. The block version can give better convergence for some badly conditioned block-structured problems thanks to its better treatment of pivots, but for other matrices appears to be less robust since the block-by-block dropping is more likely to make bad choices. More sophisticated ideas from algebraic multigrid for finding better block structures may alleviate this difficulty, as might better dropping strategies than the current ad hoc choice.

The underlying theme to this research is that significant gains can be made for iterative solvers by considering the techniques designed originally for direct solvers. Progress towards high-performance iterative methods requires solving many of the algorithmic problems that have confronted the direct methods community; the solutions developed there, tempered with knowledge of iterative approaches, are bound to be valuable.

#### References

- M. Benzi, J. Cullum, M. Tůma, Robust approximate inverse preconditioning for the conjugate gradient method, SIAM J. Sci. Comput., to appear.
- [2] M. Benzi, C. Meyer, M. Tůma, A sparse approximate inverse preconditioner for the conjugate gradient method, SIAM J. Sci. Comput. 17 (5) (1996) 1135–1149.
- [3] M. Benzi, M. Tůma, A sparse approximate inverse preconditioner for nonsymmetric linear systems, SIAM J. Sci. Comput. 19 (3) (1998) 968–994.
- [4] M. Benzi, M. Tůma, Orderings for factorized sparse approximate inverse preconditioners, SIAM J. Sci. Comput. 21 (2000) 1851–1868.
- [5] M. Brezina, J. Mandel, P. Vaněk, Algebraic multigrid by smoothed aggregation for second and fourth-order elliptic problems, Computing 56 (1996) 179–196.
- [6] R. Bridson, Multi-resolution approximate inverses, Masters Thesis, Department of Computer Science, University of Waterloo, 1999.
- [7] R. Bridson, W.-P. Tang, Ordering, anisotropy and factored sparse approximate inverses, SIAM J. Sci. Comput. 21 (1999) 867–882.
- [8] J. Dongarra, R. Whaley, Automatically Tuned Linear Algebra Software (ATLAS), SC98: High-Performance Networking and Computing Conference, Electronic Proceedings.
- [9] S. Eisenstat, J. Liu, Exploiting structural symmetry in unsymmetric sparse symbolic factorization, SIAM J. Matrix Anal. Appl. 13 (1) (1992) 202–211.
- [10] A. George, J. Liu, Computer Solution of Large Sparse Positive Definite Systems, Prentice-Hall, Englewood Cliffs, NJ, 1981.
- [11] J. Gilbert, J. Liu, Elimination structures for unsymmetric sparse LU factors, SIAM J. Matrix Anal. Appl. 14 (2) (1993) 334–352.
- [12] J. Gilbert, Predicting structure in sparse matrix computations, SIAM J. Matrix Anal. Appl. 15 (1994) 62-79.
- [13] G. Karypis, V. Kumar, A fast and high-quality multilevel scheme for partitioning irregular graphs, SIAM J. Sci. Comput. (electronic) 20 (1) (1999) 359–392.
- [14] R. Lipton, D. Rose, R. Tarjan, Generalized nested dissection, SIAM J. Numer. Anal. 16 (1979) 346-358.
- [15] J. Liu, A compact row storage scheme for Cholesky factors, ACM Trans. Math. Software 12 (2) (1986) 127-148.
- [16] J. Liu, The role of elimination trees in sparse factorization, SIAM J. Matrix Anal. Appl. 11 (1990) 134-172.
- [17] J. Liu, E. Ng, B. Peyton, On finding supernodes for sparse matrix computations, SIAM J. Matrix Anal. Appl. 14 (1993) 242–252.



Journal of Computational and Applied Mathematics 123 (2000) 307-321

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.nl/locate/cam

# Scalable preconditioned conjugate gradient inversion of vector finite element mass matrices $\stackrel{\text{$\stackrel{\leftrightarrow}{$}$}}{\Rightarrow}$

Joe Koning<sup>a</sup>, Garry Rodrigue<sup>a, \*</sup>, Dan White<sup>b</sup>

<sup>a</sup>Department of Applied Science, University of California, Davis, and Institute for Scientific Computing Research, Lawrence Livermore National Laboratory, L-561, P.O. Box 808 L-794, Livermore, CA 94550, USA <sup>b</sup>Center for Advanced Scientific Computing, Lawrence Livermore National Laboratory, L-561, Livermore, CA, USA

Received 4 June 1999

#### Abstract

Mass matrices arise in the numerical solution of time-dependent partial differential equations by the Galerkin method. Since these systems must be inverted at each time step, rapid inversion algorithms for these systems are important. When nodal finite elements are used as basis functions, it is known that the mass matrices can be consistently approximated by a diagonal matrix or solved by a scalable conjugate gradient method. This may not be the case for other basis functions. In this paper, we show that the preconditioned conjugate gradient method is scalable when used to invert mass matrices that arise from vector finite element basis functions. These basis functions are particularly important for solving Maxwell's equations on unstructured grids by the Galerkin method. (c) 2000 Elsevier Science B.V. All rights reserved.

Keywords: Galerkin method; Mass matrices; Vector finite elements; Conjugate gradient method; Vector wave equation; Maxwell's equations

# 1. Mass matrices

The Gram matrix of the linear-independent elements  $\phi_1, \phi_2, \dots, \phi_n$  in an inner product space V is the symmetric positive-definite  $n \times n$  matrix

<sup>&</sup>lt;sup>☆</sup> This research was supported under the auspices of the United States Department of Energy by Lawrence Livermore National Laboratory contract W-7405-Eng-48.

<sup>\*</sup> Correspondence address: Department of Applied Science, University of California, Davis, and Institute for Scientific Computing Research, Lawrence Livermore National Laboratory, L-561, P.O.Box 808 L-794, Livermore, CA 94550, USA. *E-mail addresses:* koning@llnl.gov (J. Koning), rodrigue@llnl.gov (G. Rodrigue), white37@llnl.gov (D. White).

J. Koning et al. | Journal of Computational and Applied Mathematics 123 (2000) 307-321

$$G = \begin{bmatrix} (\phi_1, \phi_1) & (\phi_1, \phi_2) & \cdots & (\phi_1, \phi_n) \\ (\phi_2, \phi_1) & (\phi_2, \phi_2) & \cdots & (\phi_2, \phi_n) \\ \vdots & \vdots & \vdots & \vdots \\ (\phi_n, \phi_1) & (\phi_n, \phi_2) & \cdots & (\phi_n, \phi_n) \end{bmatrix},$$
(1)

[5]. A classic example of a Gram matrix is the Hilbert matrix given by  $\phi(x) = x^{i-1}$  and the inner product  $(u, v) = \int_0^1 u(x)v(x) dx$  in which case  $G = [g_{ij}] = [(i+j-1)^{-1}]$ . Gram matrices naturally arise in the numerical solution of time-dependent partial differential equations by the Galerkin method [14]. Specifically, given the weak differential equation

$$\left(\frac{\partial u}{\partial t}, v\right) = (L[u], v), \tag{2}$$

a function  $\tilde{u}(\mathbf{x},t) = \sum_{i=1}^{n} \alpha_i(t)\phi_1(\mathbf{x})$  is sought out in a finite-dimensional subspace spanned by a linearly independent set of basis functions  $\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), \dots, \phi_n(\mathbf{x})$  that approximates the weak solution of Eq. (2). The Galerkin method calculates this approximation by defining  $\tilde{u}$  to satisfy

$$\left(\frac{\partial}{\partial t}\tilde{u},\phi_j\right) = (L[\tilde{u}],\phi_j), \quad j = 1, 2, \dots, n.$$
(3)

Then, if we let  $\alpha(t) = [\alpha_1, \alpha_2, \dots, \alpha_n]^t$ , this results in a system of ordinary differential equations

$$G\frac{\mathrm{d}\alpha}{\mathrm{d}t} = F(\alpha),\tag{4}$$

where G is the Gram matrix of Eq. (1) and is called the mass matrix of the Galerkin procedure.

If one approximates Eq. (4) by any numerical time differencing scheme, we see that it is necessary to invert the mass matrix at each time step. Hence, the ease and rapidity of the mass matrix inversion process is an important part of any Galerkin method.

#### 2. Numerical inversion of the mass matrix

Since the mass matrix G is symmetric and positive definite, the natural choice for its inversion is the preconditioned conjugate gradient method. The efficiency of the preconditioned conjugate gradient method relies on the choice of the preconditioner Q [11]. Examples of preconditioners include the incomplete Cholesky factorization [9], the SSOR preconditioner [22], multigrid preconditioners [3] and domain decomposition preconditioners [2].

An efficient preconditioner must possess three properties:

- 1. The preconditioner must be relatively easy to solve.
- 2. The matrix  $Q^{-1}G$  must "approximate the identity".
- 3. The preconditioner must yield a "scalable" method in the sense that the number of iterations to convergence must approach a constant as the size of the matrix n approaches infinity.

For the preconditioned conjugate gradient method, the spectral condition number ratio

$$\kappa(Q^{-1}G) = \frac{\lambda_{\max}(Q^{-1}G)}{\lambda_{\min}(Q^{-1}G)}$$

308

of the largest to the smallest eigenvalue of  $Q^{-1}G$  enters into the upper bound for the error

$$\frac{\|e^k\|_G}{\|e^0\|_G} \leqslant 2 \left[\frac{\kappa^{1/2} - 1}{\kappa^{1/2} + 1}\right]^k,\tag{5}$$

where the *G*-norm of the error  $||e^k||_G$ , is defined as  $(e^k)^t Ge^k$ . The bound in Eq. (5) is not sharp for the conjugate gradient method. A sharp error bound for the conjugate gradient method is more complicated [10], involving the distribution of the eigenvalues of  $Q^{-1}G$ . However, a spectral condition number close to 1 and bounded from above as the size *n* approaches infinity is sufficient to ensure fast and scalable convergence of the conjugate gradient algorithm.

In this paper we concentrate on determining preconditioners that yield scalable conjugate gradient algorithms. That is we seek preconditioners such that

$$\lim_{n \to \infty} \kappa(Q^{-1}G) < C$$

for some constant C independent of n.

Condition number bounds can sometimes be achieved by obtaining a bound on the condition number of an associated matrix and then "comparing" it to the original system. Unfortunately, there are few theoretical comparison results for the condition number of preconditioned systems. An exception is the case of diagonal and block diagonal preconditioners. Van der Sluis [19] proved the following theorem about diagonal scaling of a symmetric positive matrix G.

**Theorem** (Van der Sluis [19]). Let D be the diagonal of the symmetric positive-definite matrix G, and let  $\hat{D}$  be any other positive-definite diagonal matrix. Then

$$\kappa(D^{-1}G) \leqslant m\kappa(\hat{D}^{-1}G),$$

where m is the maximum number of nonzeros in any row of G.

When the matrix G has property-A, that is when G can be permuted in the form

$$G = \begin{bmatrix} D_1 & B \\ B^t & D_2 \end{bmatrix},$$

where  $D_1$  and  $D_2$  are diagonal matrices, a stronger result holds [8].

**Theorem** (Forsythe and Strauss [8]). Using the above notation, if the symmetric positive-definite matrix G has property-A, then

$$\kappa(D^{-1}G) \leq \kappa(\hat{D}^{-1}G).$$

A generalization of the Van der Sluis theorem has also been proved for block diagonal preconditioners [6].

**Theorem** (Demmel [6]). Let D be the block diagonal of the symmetric positive-definite matrix G, and let  $\hat{D}$  be any other symmetric positive-definite block diagonal matrix with same size blocks. Then

$$\kappa(D^{-1}G) \leqslant b\kappa(\hat{D}^{-1}G),$$

where b is the number of blocks in D.



Fig. 1. Numbering configuration for reference element  $K_0$  and quadrilateral element K.

A result similar to that of Forsythe and Strauss has also been proved for block diagonal preconditioners [7], when the matrix G is block 2-cyclic and is permuted in the form

$$G = \begin{bmatrix} D_1 & C \\ C^t & D_2 \end{bmatrix},\tag{6}$$

where  $D_i$ , i = 1, 2, is a block diagonal matrix with diagonal blocks  $D_{i,j}$ ,  $j = 1, 2, ..., r_i$ .

**Theorem** (Eisenstat et al. [7]). Let G be the form in Eq. (6) and let D be the block diagonal matrix whose diagonal blocks are  $\{D_{1,1}, \ldots, D_{1,r_1}, D_{2,1}, \ldots, D_{2,r_2}\}$ . Let  $\hat{D}$  be any other block diagonal matrix with same size blocks. Then

$$\kappa(D^{-1}G) \leq \kappa(\hat{D}^{-1}G).$$

#### 3. The finite element Galerkin method

The finite element Galerkin method is a systematic technique for constructing the basis functions  $\phi_i$  for the Galerkin method based around a numerical grid. Irregular domains and mixed boundary conditions are easily accommodated and the resulting equations describing the discrete model are generally well-conditioned [1].

Formally, a finite element  $(K, P_K, A_K)$  is defined as follows [4]:

- 1. K, a quadrilateral domain.
- 2.  $P_K = (P_1)^N = P_1 \otimes \cdots \otimes P_N$ , a vector space consisting of the tensor product of a polynomial vector spaces  $P_i$  defined on K.  $P_K$  has a basis  $\{\Psi_1, \Psi_2, \Psi_3, \Psi_4\}$ .
- 3.  $A_K$ , a set of linear functionals defined on  $P_K$  having a basis  $\alpha_1, \alpha_2, \alpha_3, \alpha_4$  (called degrees of freedom).

Each finite element  $(K, P_K, A_K)$  will be isoparametrically equivalent to a single reference finite element  $(K_0, P_0, A_0)$  where  $K_0 = \{-1 \le x, y \le 1\}$ . If we assume the numbering configuration for the nodes and edges of a given quadrilateral in Fig. 1, then the isoparametric mapping is given by

$$F_{\mathcal{K}}(\xi,\eta) = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} N_1 + \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} N_2 + \begin{bmatrix} x_3 \\ y_3 \end{bmatrix} N_3 + \begin{bmatrix} x_4 \\ y_4 \end{bmatrix} N_4,$$

where K is the quadrilateral with vertices  $\{(x_i, y_i), i = 1, 2, 3, 4\}$  and

$$N_{1}(\xi,\eta) = \frac{1}{4}(1-\xi)(1-\eta),$$

$$N_{2}(\xi,\eta) = \frac{1}{4}(1+\xi)(1-\eta),$$

$$N_{3}(\xi,\eta) = \frac{1}{4}(1+\xi)(1+\eta),$$

$$N_{4}(\xi,\eta) = \frac{1}{4}(1-\xi)(1+\eta).$$
(7)

Then,  $P_K$  is defined by

$$P_{K} = \{ p = p_{0} \cdot F_{K}^{-1} : p_{0} \in P_{0} \}$$
(8)

and the basis of  $P_K$  is given by  $\Psi_i = \Psi_i^{(0)} \cdot F_K^{-1}$  where

$$P_0 = \operatorname{span}[\Psi_1^{(0)}, \Psi_2^{(0)}, \Psi_3^{(0)}, \Psi_4^{(0)}]$$

A finite element is said to be unisolvent if the set of degrees of freedom  $A_K$  determines a unique polynomial in  $P_K$ . If this is the case, then for any function f defined on K, there exists a unique interpolant  $\pi(f) \in P$  such that  $\alpha(f) = \alpha[\pi(f)]$  for all  $\alpha \in A_K$ .

The element mass matrix is defined to be the  $4 \times 4$  matrix

$$M_K = \left[\int_K \Psi_i \cdot \Psi_j \,\mathrm{d}K\right] = \left[\int_{-1}^1 \int_{-1}^1 \Psi_i^{(0)} \cdot \Psi_j^{(0)} \,\mathrm{det}\,(J_K) \,\mathrm{d}\xi \,\mathrm{d}\eta\right],$$

where

$$J_K = \begin{bmatrix} x_{\xi} & y_{\xi} \\ x_{\eta} & y_{\eta} \end{bmatrix}.$$

The mass matrix is then given by

$$M=\sum_{K}M_{K},$$

where the matrix behind the summation signs are expanded or augmented by zero filling.

### 4. Nodal finite elements

Here, the polynomial space  $P_K = P = \text{span}[1, x, y, xy]$ . The four degrees of freedom are

$$4_{K} = \operatorname{span}[\alpha_{i}(p) = p(x_{i}, y_{i}), \quad p \in P_{K}, \ i = 1, 2, 3, 4],$$
(9)

where  $(x_i, y_i)$  are the coordinates of the nodes of *K*. Clearly, the finite element is unisolvent under the degrees of freedom in Eq. (9). For the reference element,  $P_0 = \text{span}[N_1, N_2, N_3, N_4]$  (the  $N_i$  are defined in Eq. (7)). An important result regarding the scalability of the conjugate gradient method for solving mass matrix systems arising from nodal finite elements is the following.

**Theorem** (Ciarlet [4]). Assume

- $\Omega$  a polynomial domain in  $\mathbb{R}^2$ .
- $\Gamma$  the boundary of  $\Omega$ .

- $G_h$  a quadrilateral decomposition of  $\Omega$ , i.e., a decomposition of  $\Omega$  into a set  $G_h = K_1, K_2, \ldots, K_m$  of nonoverlapping quadrilaterals  $K_i$  such that  $\Omega = \bigcup_{K \in G_h} K$  and no vertex of one quadrilateral lies on the edge of another quadrilateral.
- $h = \max_{K \in G_h} \operatorname{diam}(K)$  where  $\operatorname{diam}(K)$  is the longest side of quadrilateral K.
- There exists positive constants  $\beta_1, \beta_2$  independent of h such that for all  $K \in G_h$ ,

$$\beta_1 h \leq h_K = \operatorname{diam}(K) \leq \beta_2 h_K$$

•  $\phi_1, \phi_2, \ldots, \phi_n$  are a nodal basis functions of  $V_h$ .

Then if  $M = \text{matrix}(\int_{\Omega} \phi_i \phi_j d\Omega)$  is the mass matrix, there exist constants  $C_1, C_2$  depending only on  $\beta_1, \beta_2$  such that

$$\kappa(M) \leqslant \frac{C_2}{C_1}.$$

Hence, we see that if a sequence of grids satisfies the previous theorem, then the preconditioned conjugate gradient will attain a constant number of iterations as the number of grid points increases whenever the preconditioner satisfies any of the theorems in Section 2.

Another important property of nodal mass matrices is that they can be consistently "lumped" [15]. That is, they can be consistently approximated by a diagonal matrix.

#### 5. Motivation – the vector wave equation

The two-dimensional Maxwell's equations consist of two equations that relate the vector electric field  $E = [E_1, E_2]$ , a scalar magnetic field H and a divergence condition [12].

$$\nabla \times \boldsymbol{E} = -\frac{\partial H}{\partial t},\tag{10}$$

$$\vec{\nabla} \times H = \frac{\partial \boldsymbol{D}}{\partial t},\tag{11}$$

$$\nabla \cdot \boldsymbol{D} = 0, \tag{12}$$

where

$$\overrightarrow{\nabla} \times H = \left[\frac{\partial H}{\partial y}, -\frac{\partial H}{\partial x}\right]^t, \quad \nabla \times \mathbf{E} = \frac{\partial E_2}{\partial x} - \frac{\partial E_1}{\partial y}.$$

Two constitutive relations are required to close Maxwell's equations,

$$\boldsymbol{D} = \varepsilon \boldsymbol{E}, \quad \boldsymbol{B} = \boldsymbol{\mu} \boldsymbol{H}, \tag{13}$$

where the dielectric permittivity  $\varepsilon$  and the magnetic permeability  $\mu$  are scalar functions of position.

The magnetic field is eliminated by applying the operation  $\overline{\nabla} \times$  to Eq. (10) and applying the identities Eqs. (11) and (13) to obtain the vector wave equation for the electric field

$$\varepsilon \frac{\partial^2 \boldsymbol{E}}{\partial t^2} = -\overrightarrow{\nabla} \times \frac{1}{\mu} \nabla \times \boldsymbol{E}$$
(14)

The Galerkin method for solving Eq. (14) computes an approximation

$$\tilde{\boldsymbol{E}} = [\tilde{\boldsymbol{E}}_1, \tilde{\boldsymbol{E}}_2]^t = \sum_{i=1}^n \alpha_i(t) \, \vec{\boldsymbol{\Phi}}(x, y),$$

such that

$$\int_{\Omega} \varepsilon \left[ \frac{\partial^2}{\partial t^2} \tilde{\boldsymbol{E}} \right]^t \vec{\boldsymbol{\phi}}_j \mathrm{d}\Omega = -\int_{\Omega} \left[ \vec{\nabla} \times \frac{1}{\mu} \nabla \times \tilde{\boldsymbol{E}} \right]^t \vec{\boldsymbol{\phi}}_j \mathrm{d}\Omega$$
$$= -\int_{\Omega} \frac{1}{\mu} [\nabla \times \boldsymbol{E}]^t [\nabla \times \vec{\boldsymbol{\phi}}_j] \mathrm{d}\Omega,$$

where the second equality follows from Green's second vector theorem [20]. Substituting the expansion for E we get a square system of equations

$$\sum_{i} \left( \int_{\Omega} \vec{\Phi}_{i}^{t} \vec{\Phi}_{j} \, \mathrm{d}\Omega \right) \frac{\partial^{2} \alpha_{i}}{\partial t^{2}} = -\sum_{i} \left( \int_{\Omega} \left[ \nabla \times \vec{\Phi}_{i} \right]^{t} \left[ \nabla \times \vec{\Phi}_{j} \right] \mathrm{d}\Omega \right) \alpha_{i}$$

yielding the system of ordinary differential equations in Eq. (4) where the mass matrix G is given by

$$G = \left[ \int_{\Omega} \vec{\Phi}_i^t \vec{\Phi}_j \, \mathrm{d}\Omega \right].$$

One could, of course, use the nodal finite elements to provide Galerkin vector approximations to the vector wave equation in Eq. (14). This has the advantage that the mass matrices can be consistently lumped or be solved by a scalable preconditioned conjugate gradient algorithm. Unfortunately, the continuity of the nodal finite element approximations turns out to be a liability when applied to the vector wave equation when the dielectric  $\varepsilon$  has a jump discontinuity. In this case, it is known that the tangential component of the electric field is continuous across the discontinuity while its normal component may be discontinuous. Consequently, an important property of electric fields that should be preserved in any numerical approximation is the following: The tangential component of E across an interface is continuous but the normal component f E across the same interface may be discontinuous. To ensure this, the tangential component of the numerical approximation  $\tilde{E}$  should be continuous along the edges of each quadrilateral of the grid but its normal component need not be. More specifically, if  $K_1$  and  $K_2$  are two elements with a common edge e then the tangential components of  $\pi_1(u)$  and  $\pi_2(u)$  are the same on e for all  $u \in C^{\infty}(K_1 \cup K_2)$  [16]. Finite elements with this property are said to be conformal. Since the tangential and normal components of the Galerkin approximation provided by the nodal finite elements are continuous, nonphysical spurious oscillations have been observed when they are used to solve Eq. (14) [13].

#### 6. Edge elements

Finite elements that enforce continuity of the electric field across edges have been recently discovered and analyzed [16,17]. Basically, these "vector finite element" assign degrees of freedom to the edges rather than to the nodes of the elements. For this reason, they are called edge elements. Although these types of elements were described in [21], as early as 35 years ago, their use and importance in electromagnetics was not realized until recently. Extensive investigations as well as some very successful applications have been carried over the past few years [18,20]. In this section, we introduce the edge elements  $(K, P_K, A_K)$  in two dimensions and analyze the mass matrices that arise from their use in the Galerkin procedure.

The degrees of freedom  $A_K$  for the edge elements are the line integrals

$$\alpha_i(\boldsymbol{p}) = \int_{e_{i_i}} \boldsymbol{p} \cdot \boldsymbol{t}_i \, \mathrm{d} e_i, \quad \boldsymbol{p} \in P$$

where  $t_i$  is the unit tangent along edge  $e_i$ , i = 1, 2, 3, 4 [13]. The fact that these elements are conforming is found in [17]. On the reference element,

$$P_0 = \{a + b\eta\} \otimes \{c + \mathsf{d}\xi\}$$

and the conditions

$$\alpha_i(\boldsymbol{p}) = \int_{e_{i_j}} \boldsymbol{p} \cdot \boldsymbol{t}_j \, \mathrm{d}\sigma = \delta_{ij}, \quad \boldsymbol{p} \in P_0$$

yields the basis functions

$$\begin{split} \Psi_1^{(0)}(\xi,\eta) &= \frac{1}{4}(1-\eta) \begin{bmatrix} 1\\0 \end{bmatrix}, \\ \Psi_2^{(0)}(\xi,\eta) &= \frac{1}{4}(1+\eta) \begin{bmatrix} 1\\0 \end{bmatrix}, \\ \Psi_3^{(0)}(\xi,\eta) &= \frac{1}{4}(1-\xi) \begin{bmatrix} 0\\1 \end{bmatrix}, \\ \Psi_4^{(0)}(\xi,\eta) &= \frac{1}{4}(1+\xi) \begin{bmatrix} 0\\1 \end{bmatrix}. \end{split}$$

Then,  $P_K = \text{span}[\Psi_1, \Psi_2, \Psi_3, \Psi_4]$  where  $\Psi_i(x, y) = \Psi_i^{(0)}[F_K^{-1}(x, y)]$ . Note that if  $t_j$  is the unit tangent vector along edge  $e_j$ , then

$$\int_{e_j} (\Psi_i(x, y) \cdot \mathbf{t}_j) \,\mathrm{d} e_j = \delta_{ij}.$$

The element mass matrix is given by

$$\int_{K} \Psi_{i}^{t} \Psi_{j} \, \mathrm{d}K = \int_{-1}^{1} \int_{-1}^{1} (\Psi_{i}^{0})^{t} \Psi_{j}^{(0)} \, \mathrm{det}(J_{K}) \, \mathrm{d}\xi \, \mathrm{d}\eta.$$

n	$\kappa(M)$
12	2.37
40	3.87
144	3.92
544	3.97
2112	4.0
5320	4.0
33024	4.07

Table 1 Mass matrix condition number for n edge elements

#### 7. Edge element mass matrices

#### 7.1. Uniform grid

We first consider the edge element mass matrices generated on a uniform grid of grid size *h*. Here,  $K = \{(x_i \le x \le x_i + h, y_i \le y \le y_i + h)\}$  and the element matrix is given by

 $M_K = rac{h^2}{6} egin{bmatrix} 2 & 1 & 0 & 0 \ 1 & 2 & 0 & 0 \ 0 & 0 & 2 & 1 \ 0 & 0 & 1 & 2 \end{bmatrix}.$ 

This yields a block diagonal mass matrix

$$M = \frac{h^2}{6} \begin{bmatrix} A_1 & & & \\ & A_2 & & \\ & & \ddots & \\ & & & A_{n-1} \\ & & & & & A_n \end{bmatrix}, \quad A_i = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 2 \end{bmatrix}.$$

If we estimate the eigenvalues of the mass matrix M using Gerschgorin discs, we get the following result.

**Theorem 1.** If M is the mass matrix generated from vector edge elements on a uniform rectangular grid, then

$$\kappa(M) \leqslant 6. \tag{15}$$

Table 1 tabulates the actual condition number for a variety of matrix sizes. We see that the bound on the condition number in Eq. (15) appears to be an over-estimate.

Of course, a natural question to ask would be whether the mass matrix can be consistently approximated by a diagonal matrix much in the same manner as is commonly done using mass lumping techniques for nodal elements. In this regard, if the trapezoid rule is used to evaluate the inner product integrals, we get the following result.

n	f = 0.49	f = 0.47	f = 0.44	f = 0.35
144	3.86	4.41	5.27	10.31
544	3.92	4.95	6.76	20.28
2112	3.97	5.49	8.35	45.35
5320	3.99	6.15	10.69	86.18
33024	4.0	7.2	15.72	230

Table 2 Condition number of element mass matrix for f = 0.44

 Table 3

 Condition number calculations for diagonally preconditioned systems

n	f = 0.49	f = 0.47	f = 0.44	f = 0.35
40	3.0	3.15	3.33	4.08
144	3.0	3.18	3.43	4.43
544	3.0	3.28	3.64	5.58
2112	3.0	3.43	3.95	10.87
5320	3.0	3.63	4.3	16.28

#### Theorem 2.

$$\frac{h}{64}I = M_K + \mathcal{O}(h^2). \tag{16}$$

A corollary to Theorem 2 is that the diagonal approximation in Eq. (16) yields the well-known Yee's method which is totally consistent with the vector wave equation [13].

#### 7.2. Non-uniform grid

We now examine the edge element mass matrices based upon a nonuniform grid. In this case, no consistent mass lumping procedure is known to exist and matrix inversion of the mass matrix is necessary to use the Galerkin procedure. In this section, we examine the condition numbers of the preconditioned mass matrices to determine if a scalable preconditioned conjugate gradient method exists.

The nonuniform grids were constructed by recursively forming four new quadrilaterals out of one initial quadrilateral. Along each edge of the quad, a random position is chosen using:  $x_{newnode} = sx_{node i} + (1-s)x_{node j}$ ,  $y_{newnode} = sy_{node i} + (1-s)y_{node j}$ , where *s* is defined by a user chosen variable *f* as  $s = f + (1 - 2f) \operatorname{rand}(\cdot)$  and  $\operatorname{rand}(\cdot)$  is a random number between 0 and 1. These four new nodes are used to define the center by finding random positions between the new left and right nodes, as well as the new top and bottom nodes, thus giving four new quadrilaterals. This operation is performed on each new quadrilateral until the desired number of elements is reached (see Fig. 2).

Tables 2 and 3 list the condition number of the unconditioned and diagonally preconditioned mass matrices. Inner product integrations were performed using a four-point Gaussian quadrature rule.



f =.35

Fig. 2.  $32 \times 32$  numerical grids for different f values.

Table 4	
Number of preconditioned c	onj. grad. iterations for $f = 0.44$

n	Jacobi	ILU
144	14	5
544	15	6
2112	16	6
33024	16	7

Table 4 list the number of iterations for convergence of the preconditioned conjugate gradient algorithm for the mass matrices generated on the unstructured grids generated when f = 0.44. The preconditioners used were Jacobi diagonal scaling and the Incomplete-LU.

As one can be seen from the condition number computations for f = 0.35 in Table 3, the condition number of the preconditioned does not seem to be approaching a constant as would be hoped. The



Fig. 3.  $16 \times 16$  numerical grid for f = 0.35.

Table	5											
Ratio	of max	ximum	zone	area	to	minimum	zone	area	for	different	grid	sizes

n	f = 0.47	f = 0.44	f = 0.35
40	0.86	0.72	0.44
144	0.77	0.59	0.24
544	0.69	0.47	0.12
2112	0.62	0.37	0.07
5320	0.54	0.29	0.03
33024	0.48	0.23	0.01

Table 6

Ratio of maximum edge length to minimum edge length for different grid sizes

n	f = 0.47	f = 0.44	f = 0.35
40	0.856	0.733	0.441
144	0.796	0.626	0.293
544	0.719	0.509	0.172
2112	0.657	0.424	0.107
5320	0.597	0.351	0.069
33024	0.547	0.294	0.42

reason for this is that, unlike uniform grids, the fundamental structure of the grid is not the same as the number of grid points is increased, compare the grids in Figs. 2 and 3. This becomes evident when one compares the zone sizes and edge lengths of the different grids. Tables 5 and 6 list ratios of maximum to minimum zone sizes and edge lengths of the different grid sizes. In this case, n refers to the number of edges in the grid.

In order to determine if a result holds that is similar to the Ciarlet Theorem for nodal finite elements, the previous computations were carried out on a sequence of grids whose diameters are



Fig. 4.  $2^{n-1} + 1 \times 2^{n-1} + 1$  grids.

related (see Fig. 4). The initial coarse grid was constructed using a seed of f = 0.35. Mesh metrics, condition numbers and preconditioned conjugate gradient iterations are given in Tables 7–9, respectively.

### 8. Conclusions

In this paper we have established computationally that the condition number of the diagonally preconditioned mass edge element matrix essentially remains constant as the size of a grid increases provided the ratio of the mesh lengths remains constant. This is useful when the preconditioned conjugate gradient algorithm is used to invert the edge element mass matrix in Galerkin procedures for solving Maxwell's equations.

max(diam) min(diam)
0.293104
0.293099
0.293091
0.293072
0.293035

Table	7
Mesh	metrics

Table 8

Condition numbers of mass matrix M and diagonally preconditioned matrix  $Q^{-1}M$ 

n	$\kappa(M)$	$\kappa(Q^{-1}M)$
4	20.29	4.44
5	25.32	4.56
6	30.05	4.71
7	34.1	4.91
8	36.75	5.17

#### Table 9

Number of iterations for diagonally scaled conj. grad. and ILU conj. grad.

n	Jacobi	ILU
4	17	7
5	17	8
6	18	9
7	18	9
8	18	9

#### Acknowledgements

The authors would like to thank the Institute for Scientific Computing Research at the Lawrence Livermore National Laboratory for partial support of this research.

# References

- [1] A. Axelsson, V. Barker, Finite Element Solution of Boundary Value Problems, Academic Press, New York, 1984.
- [2] P.E. Bjorstad, O.B. Widlund, Iterative methods for the solution of elliptic problems on regions partitioned into substructures, SIAM J. Numer. Anal. 23(6) (1986) 1097–1120.
- [3] A. Brandt, Multilevel adaptive solutions to boundary value problems, Math. Comput. 31 (1977) 333-390.
- [4] P. Ciarlet, The Finite Element Methods for Elliptic problems, North-Holland, Amsterdam, 1987.
- [5] P. Davis, Interpolation and Approximation, Blaisdell, Waltham, MA, 1963.
- [6] J. Demmel, The condition number of equivalence transformations that block diagonalize matrix pencils, SIAM J. Numer. Anal. 20 (1983) 599–610.
- [7] S. Eisenstat, J. Lewis, M. Schultz, Optimal block diagonal scaling of blick 2-cyclic matrices, Linear Algebra Appl. 44 (1982) 181–186.

- [8] G.E. Forsythe, E.G. Strauss, On best conditioned matrices, Proc. Amer. Math. Soc. 6 (1955) 340-345.
- [9] G. Golub, C. Van Loan, Matrix Computations, John Hopkins University Press, Baltimore, 1983.
- [10] A. Greenbaum, Comparison of splitting used with the conjugate gradient method, Numer. Math. d 33 (1979) 181–194.
- [11] A. Greenbaum, G. Rodrigue, Optimal preconditioners of a given sparsity pattern, BIT 29 (1989) 610-634.
- [12] J.D. Jackson, Classical Electrodynamics, Wiley, New York, 1975.
- [13] J. Jin, The Finite Element Method in Electromagnetics, Wiley, New York, 1993.
- [14] C. Johnson, Numerical Solution of Partial Differential Equations by the Finite Element Method, Cambridge University Press, Cambridge, 1987.
- [15] L. Lapidus, G. Pinder, Numerical Solution of Partial Differential Equations in Science and Engineering, Wiley, New York, 1982.
- [16] J.C. Nedelec, Mixed finite elements in R3, Numer. Math. 35 (1980) 315-341.
- [17] J.C. Nedelec, A new family of mixed finite elements in R3, Numer. Math. 50 (1986) 57-81.
- [18] G. Rodrigue, D. White, A vector finite element time-domain method for solving Maxwell's equations on unstructured hexahedral grids, submitted for publication.
- [19] A. Van der Sluis, Condition numbers and equilibration matrices, Numer. Math. 14 (1969) 14-23.
- [20] D. White, G. Rodrigue, Improved vector FEM solutions of Maxwell's equations using grid pre-conditioning, Internat. J. Numer. Methods Eng. 40 (1997) 3815–3837.
- [21] H. Whitney, Geometric Integration Theory, Princeton University Press, Princeton, NJ, 1957.
- [22] D.M. Young, Iteration Solution of Large Linear Systems, Academic Press, New York, 1971.



Journal of Computational and Applied Mathematics 123 (2000) 323-352

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.nl/locate/cam

# Robust multigrid methods for nonsmooth coefficient elliptic linear systems

Tony F. Chan<sup>a,\*,1</sup>, W.L. Wan<sup>b, 2</sup>

<sup>a</sup>Department of Mathematics, University California at Los Angeles, Los Angeles, CA 90095-1555, USA <sup>b</sup>SCCM Program, Gates Building 2B, Stanford University, Stanford, CA 94305-9025, USA

Received 15 September 1999

#### Abstract

We survey the literature on robust multigrid methods which have been developed in recent years for solving second-order elliptic PDEs with *nonsmooth* coefficients. We highlight the key ideas of designing robust multigrid methods which are able to recover the usual multigrid efficiency for nonsmooth coefficient PDEs on structured or unstructured grids. In particular, we shall describe various approaches for constructing the interpolation and the smoothing operators, and the coarse grid points selections. © 2000 Elsevier Science B.V. All rights reserved.

#### 1. Introduction

Multigrid methods are multilevel techniques for solving partial differential equations (PDEs) by eliminating errors in different parts of the spectrum on a sequence of coarse grids, or more generally, coarse subspaces. The basic principle is based on the interplay of smoothing and coarse grid correction which complement each other; the smooth errors not being reduced by smoothing are eliminated by coarse grid corrections. These techniques can generally be applied directly to PDEs but are of most interest when applied to the linear systems arising from their discretizations. Multigrid methods have been widely used in a broad variety of applications, from Poisson equations to full Navier–Stokes equations, from two-dimensional square domains to three-dimensional unstructured airfoil grids, etc. Multigrid has proved itself as a powerful and successful numerical technology for fast and efficient computations. In contrast with many other iterative methods such as classical relaxation methods, multigrid offers the capability of solving elliptic PDE problems with complexity

<sup>\*</sup> Corresponding author.

E-mail addresses: chan@math.ucla.edu (T.F. Chan), wan@sccm.stanford.edu (W.L. Wan).

<sup>&</sup>lt;sup>1</sup> Partially supported by the NSF under Contract ASC-9720257, NASA Ames under Contract NAG2-1238, and Sandia National Laboratory under Contract LG-4440.

<sup>&</sup>lt;sup>2</sup> Partially supported by the Department of Energy under the Accelerated Strategic Computing Initiative (ASCI).



Fig. 1. A spectrum of multigrid methods.

and storage proportional to the number of unknowns, and its convergence rate is often independent of the problem size.

In this article, we survey robust multigrid methods in the literature which have been developed in recent years for solving second-order elliptic PDEs with *nonsmooth* coefficients. While multigrid converges rapidly for model problems such as the Poisson equation on a square, its convergence rate can be severely affected by PDEs with highly nonsmooth coefficients, or problems defined on complicated geometries, and unstructured grids. Unfortunately, these types of problems often arise in industrial applications, and hence traditional multigrid methods must be redesigned for them. The primary focus of this paper is in the design of robust multigrid methods which are able to retain the usual multigrid efficiency for smooth coefficient PDEs on structured grids. In particular, we shall describe various approaches for constructing the interpolation and the smoothing operators, and the coarse grid points selections.

General surveys of multigrid methods for solving different kinds of applications can be found in Brandt [20,21,23]. A survey on multilevel methods on unstructured grids can be found in Chan et al. [28]. Also, see [61,70] for a survey of parallel implementation of multigrid, which is not within the scope of this paper. Surveys on other aspects of multigrid methods can be found in [24,54,60,93]. We also note that the introductory note by Wagner [90] contains a lot of the details of the interpolation approaches discussed in this paper. Finally, we refer the readers to MGNet http://www.mgnet.org for a database of an extensive collection of multigrid papers in the literature.

The idea of multigrid was introduced and analyzed by Brakhage [15], and Fedorenko [47,48] in the 1960s, followed by Bachvalov [5]. Multigrid methods have not been paid much attention in the 1970s until the works of Astrachancer [2], Bank and Dupont [6], Brandt [19], Hackbusch [53], Nicolaides [74], and others showed that multigrid is indeed a very useful technique practically and theoretically. An enormous amount of progress has been achieved since then. Various multigrid methods have been developed, ranging from geometry specific to purely algebraic black box methods, and a spectrum of methods exist between the two extremes; see Fig. 1. We refer to this spectrum of methods as *gray box* methods: they require more information about the problem (e.g. grids, matrix graph, etc.) than a complete black box approach, but on the other hand, they can produce better robustness and performance.

Close to the geometric-dependent end of the spectrum where Cartesian grid was used, Alcouffe et al. [1] was one of the earliest papers to address the issue of nonsmooth coefficient PDE problems, and proposed robust interpolation methods for multigrid; see also [62]. Along this line were also the black box multigrid method by Dendy [38,39], and matrix-dependent approaches by de Zeeuw [102] and Reusken [76,77]. Other related approaches include frequency decomposition by Hackbusch [56], and filtering decomposition by Wittum [96,97]. The purely algebraic methods, on the other end of

the spectrum, were first proposed by Brandt et al. [25], and then popularized by Ruge and Stüben [79]. There is a recent resurgence of interest in AMG and other multigrid algorithms with focuses on parallel implementation and memory hierarchy aspects [26,36,37,43,44,64,75,87]. An introduction to AMG is recently given by Stüben [84]. See also the algebraic multilevel methods by Axelsson and Vassilevski [3,4], and an additive version of AMG by Grauschopf et al. [49]. The geometric unstructured multigrid methods were studied by Bank and Xu [9], Chan et al. [32], Guillard [52], Lallemand et al. [65], Morano et al. [72], and Xu [100]. The recent interest in energy minimization was studied by Brezina et al. [26], Brezina et al. [67,68], Chan et al. [33] with a local minimization perspective, and Wan et al. [92] with a global minimization perspective. Another recent interest is in the direction of bridging the gap between Gaussian elimination and multigrid; see [8,78]. Other multilevel methods include the hierarchical basis multigrid methods proposed by Yserentant [101], and Bank et al. [7], and the BPX method proposed by Bramble, Pasciak and Xu [17]. In Griebel [50], multilevel methods including multigrid and BPX were viewed as iterative methods on semidefinite systems. General multigrid references can be found in the books of Bramble [16], Briggs [27], Hackbusch [55], Smith et al. [80], and Wesseling [94]. Finally, we note that we are not able to survey the many more references in the literature here.

This paper is organized as follows: Section 1 begins with the basic principles of multigrid, and its classical convergence analysis. The design of robust multigrid will be discussed component by component. In Section 2, the construction of various sophisticated interpolation operators is described. Section 3 concerns the robustness and efficiency of smoothers. Algebraic and geometric coarsening strategies are covered in Section 4. Finally, Section 5 summarizes the current and future research on robust multigrid methods for elliptic linear systems.

In the rest of this section, we introduce the model problem and notation used in this paper, followed by the standard multigrid algorithm and the classical convergence analysis for smooth coefficient problems.

#### 1.1. Elliptic PDEs

Elliptic PDE problems are among the most extensively investigated problems in applied mathematics. Their relation to many physical models is well known and the theoretical and numerical results obtained in this area are very useful in practice. The design of numerical methods for such model problems can often be adapted and applied to more complicated situations. Elliptic problems are also important in their own right, for instance, in the solution of the pressure equation arising from incompressible fluid problems, implicit time integration schemes, etc.

The model problem of primary interest is the following elliptic PDE which exhibits the fundamental properties and challenges that the elliptic problems above generally experience:

$$-\nabla \cdot a(x)\nabla u(x) = f(x) \quad x \in \Omega,$$
  
$$u = 0 \quad x \in \partial\Omega,$$

where  $\Omega \subset \mathbb{R}^d$ , d=2,3, is a polygonal or polyhedral domain, and a(x), in general, is a  $d \times d$  symmetric positive-definite matrix whose eigenvalues are bounded uniformly on  $\overline{\Omega}$ , and its coefficients can be oscillatory or discontinuous with large jumps across the interfaces. We note that Dirichlet boundary condition is used just for simplicity, and other boundary conditions are also permissible.
Many of the multigrid methods discussed in this paper apply to the discretization matrices given by finite element, finite difference or finite volume methods. For easy exposition, we set up notations based on finite element discretization. Let  $H^1(\Omega)$  be the standard Sobolov space consisting of square integrable functions with square integrable derivatives of first order, and  $H_0^1(\Omega)$  the subspace of  $H^1(\Omega)$  whose functions vanish on  $\partial \Omega$ . Solving the PDE problem is equivalent to finding  $u \in H_0^1(\Omega)$ such that

$$a(u,v) = (f,v) \quad \forall v \in H_0^1(\Omega), \tag{1}$$

where

$$a(u,v) = \int_{\Omega} a(x) \nabla u \cdot \nabla v \, \mathrm{d}x, \quad (f,v) = \int_{\Omega} f v \, \mathrm{d}x.$$

Suppose  $\Omega$  is triangulated by quasi-uniform nonoverlapping simplices  $\tau_i$  with size *h*, i.e.,  $\Omega = \bigcup_i \tau_i$ . Define the finite element subspace by

$$V^{h} = \{v^{h} \in H_{0}^{1}(\Omega): v^{h}|_{\tau_{i}} \in P_{1}(\tau_{i}), \forall i\},\$$

where  $P_1(\tau_i)$  is the set of linear functions on  $\tau_i$ . The finite element approximation is the function  $u^h \in V^h$  such that

$$a(u^h, v^h) = (f, v^h) \quad \forall v^h \in V^h.$$
<sup>(2)</sup>

Define a linear operator  $A^h: V^h \to V^h$  by

$$(A^h u^h, v^h) = a(u^h, v^h) \quad u^h, v^h \in V^h.$$

Then (2) is equivalent to

$$4^h u^h = f^h, (3)$$

where  $f^h$  is the projection of f on  $V^h$ . Let  $\{\phi_j^h\}_{j=1}^n$  be the set of nodal basis of  $V^h$ . Write  $u^h = \sum_{j=1}^n \mu_j^h \phi_j^h$ , and  $f^h = \sum_{j=1}^n b_j \phi_j^h$ . Then (3) is equivalent to the linear system

$$\mathscr{A}^{h}\mu^{h} = b^{h}, \tag{4}$$

where  $\mathscr{A}^h$  is the stiffness matrix,  $\mu^h = (\mu_1^h, \dots, \mu_n^h)^T$ ,  $b^h = \mathscr{M}^h(b_1^h, \dots, b_n^h)^T$ , and  $\mathscr{M}^h$  the mass matrix. It is well known that the condition number of  $\mathscr{A}^h$  grows in the order of  $O(h^{-2})$ , and hence classical iterative methods converge very slowly for large-scale problems. In the next sections, we describe a fast solution procedure – *multigrid* – for solving (4) whose convergence rate is often independent of the mesh size *h*.

**Remark.** We distinguish  $A^h$ , a linear operator of  $V^h$ , from  $\mathscr{A}^h$ , the corresponding stiffness matrix. The multigrid algorithms discussed in this paper are fast solution methods for solving the matrix equation (4).

#### 1.2. Basic principles of multigrid

The idea of multigrid consists of two main components: *smoothing and coarse grid correction*. The smoothing process, usually carried out by a few iterations of a relaxation method, damps away the high frequency error components. The coarse grid correction process, carried out by a restriction,

a coarse grid solve, and an interpolation, eliminates the low-frequency error components. Hence, the key of multigrid is that smoothing and coarse grid correction complement each other. As a result, the combination of the two yields a significant error reduction, resulting in a fast solution procedure. Moreover, we gain efficiency since the coarse grid solves are less expensive than the fine grid one. The two-grid solution process is made more precise in the following.

We begin with an initial guess  $\mu^h$  and we smooth the error by applying a relaxation iteration

$$\mu_1^h = \mu^h + \mathscr{R}^h(b^h - \mathscr{A}^h\mu^h),\tag{5}$$

where  $\mathscr{R}^h$  is the approximate inverse of  $\mathscr{A}^h$  given by the relaxation method. Then, we improve  $\mu_1^h$  by a coarse grid correction consisting of the following steps (Eqs. (6)–(8)): First, we restrict the residual to the coarse grid  $V^H$ :

$$r^{H} = \mathscr{I}_{h}^{H}(b^{h} - \mathscr{A}^{h}\mu_{1}^{h}), \tag{6}$$

where  $\mathscr{I}_h^H$  is the restriction operator. Second, we solve the coarse grid error equation:

$$\mathscr{A}^{H}e^{H} = r^{H},\tag{7}$$

where the coarse grid matrix  $\mathscr{A}^{H}$  is formed by the Galerkin process:  $\mathscr{A}^{H} = \mathscr{I}_{h}^{H} \mathscr{A}^{h} \mathscr{I}_{H}^{h}$ . The Galerkin approach can be shown to be optimal for symmetric positive-definite elliptic problems. Here  $\mathscr{I}_{H}^{h} = (\mathscr{I}_{h}^{H})^{T}$  is the interpolation operator. The coarse grid error is then interpolated back to the fine grid and the fine grid approximation is updated by

$$\mu_2^h = \mu_1^h + \mathscr{I}_h^H e^H. \tag{8}$$

Finally, we apply a post-smoothing at the end:

$$\mu_{\text{new}}^h = \mu_2^h + \mathscr{R}^h (b^h - \mathscr{A}^h \mu_2^h). \tag{9}$$

Combining (5)-(9), the entire process can be summarized by the following result.

**Lemma 1.1.** The iteration matrix  $M_{tg}$  of the two-grid method with  $v_1$  steps of pre-smoothing and  $v_2$  steps of post-smoothing is given by

$$M_{\rm tg} = (I - \mathscr{R}_2^h \mathscr{A}^h)^{\nu_2} (I - \mathscr{I}_h^H (\mathscr{A}^H)^{-1} \mathscr{I}_h^H \mathscr{A}^h) (I - \mathscr{R}_1^h \mathscr{A}^h)^{\nu_1}, \tag{10}$$

where  $\mathscr{R}_1^H$  and  $\mathscr{R}_2^h$  denote the approximate inverses of  $\mathscr{A}^h$  given by the pre- and post-smoother, respectively.

If we solve the coarse grid problem (7) recursively by the same two-grid method, we obtain a multigrid algorithm. We use the following notations for the multilevel methods in the next sections. Let  $V_J = V^h$  be the fine grid space, and  $V_1 \subset V_2 \subset \cdots \subset V_J$  be a sequence of nested coarse grid subspace of  $V^h$ . On each level  $k, A^k : V_k \to V_k$  is the restriction of  $A^J = A^h$  on  $V_k$ , and  $R^k : V_k \to V_k$  is an approximate inverse of  $A^k$  given by the smoother. Their corresponding stiffness matrix is denoted by  $\mathscr{A}^k$  and  $\mathscr{R}^k$ , respectively. Let  $I_{k-1}^k : V_{k-1} \to V_k$  be the interpolation operator and its adjoint  $I_k^{k-1} : V_k \to V_{k-1}$  the restriction operator. Their matrix representation is denoted by  $\mathscr{I}_{k-1}^k$  and  $\mathscr{I}_{k-1}^k = (\mathscr{I}_{k-1}^k)^T$ , respectively.

In addition, associated with each  $A^k$ , we define the *A*-inner product by  $(\cdot, \cdot)_{A^k} \equiv (A^k \cdot, \cdot)$ . Let  $Q_k : V_j \to V_k$  and  $P_k : V_J \to V_k$  be the projection operators with respect to the  $L^2$  and *A* inner products, respectively.

#### 1.3. Convergence theory

In view of (10), the classical convergence analysis involves the norm estimates of

$$\|I - \mathscr{R}^h \mathscr{A}^h\|$$
 and  $\|I - \mathscr{I}_h^H (\mathscr{A}^H)^{-1} \mathscr{I}_h^H \mathscr{A}^h\|.$ 

See [16,22,55,71] for details. Here, we summarize the results in the literature based on the subspace correction framework developed by Xu [99]. The convergence of multigrid is governed by two constants  $K_0$  and  $K_1$  defined as

 $K_0$ : For any  $v \in V$ , there exists a decomposition  $v = \sum_{i=1}^J v_i$  for  $v_i \in V_i$  such that

$$\sum_{i=1}^{J} (R_i^{-1} v_i, v_i) \leqslant K_0(Av, v),$$
(11)

where  $R_i$  is the approximate inverse operator given by the smoother.

 $K_1$ : For any  $S \subset \{1, \ldots, J\} \times \{1, \ldots, J\}$  and  $u_i, v_i \in V$  for  $i = 1, \ldots, J$ ,

$$\sum_{(i,j)\in S} (T_i u_i, T_j u_j)_A \leqslant K_1 \left( \sum_{i=1}^J (T_i u_i, u_i)_A \right)^{1/2} \left( \sum_{j=1}^J (T_j v_j, v_j)_A \right)^{1/2},$$
(12)

where  $T_i = R_i A_i P_i$ .

**Theorem 1.2.** Let  $M_{mg}$  be the iteration matrix given by the V-cycle multigrid. Then

$$||M_{\rm mg}||_A^2 \leq 1 - \frac{2 - \omega_1}{K_0 (1 + K_1)^2},$$

where  $\omega_1 = \max_{1 \leq i \leq J} \rho(R_i A_i)$ .

**Proof.** See [99]. □

By Theorem 1.2, the convergence rate can be improved by producing a smaller  $K_0$  or  $K_1$ . By definition, it can be easily proved the following result.

#### Lemma 1.3.

 $K_1 \leq \omega_1 J.$ 

**Proof.** See [99]. □

Thus, the estimate of  $K_0$  is crucial. We analyze  $K_0$  from the domain decomposition perspective. For second order scaler elliptic PDEs,  $K_0$  depends on two inequalities:

$$\|Q_1v\|_A^2 + \sum_{k=2}^{5} \|(Q_k - Q_{k-1})v\|_A^2 \leqslant C_0 \|v\|_A^2,$$
(13)

$$\|(Q_k - Q_{k-1})v\| \leq C_0 h_{k-1} \|Q_k v\|_A, \quad k > 1,$$
(14)

where  $Q_k: V \to V_k$  is the  $L^2$  projection. More precisely, we have the following estimate:

Lemma 1.4. Suppose (13) and (14) are satisfied. Then

$$K_0 \leqslant \frac{C_0}{\omega_0},$$

where  $\omega_0 = \min_{1 \leq k \leq j} \lambda_{\min}(\mathscr{R}_k^{-1} \mathscr{A}_k).$ 

**Proof.** See [99]. □

The *stability* inequality (13) is known as the partition lemma [66,69] which plays an essential role in the convergence analysis of domain decomposition methods. It requires that for any given  $v \in V$ , we must be able to decompose it into  $v_k \in V_k$  such that the total energy of all the pieces  $v_k$  is bounded by a small constant factor of the original energy of v. In the multigrid context, it can be translated into the following: the coarse grid basis functions must have small energy. The *approximation* inequality (14) requires that the functions on the coarse grids approximate the fine grid functions to at least first-order accuracy. A sufficient condition is that the coarse subspace contains constant functions.

In conclusion, one major approach of improving robustness is to devise multigrid methods which lead to a small  $K_0$ . For instance, the constructions of the robust interpolation operators described in Section 2.5 are based on the stability and approximation inequalities.

## 1.4. Multigrid for nonsmooth coefficient PDEs

The success of multigrid hinges on the choice of coarse grids, and the smoothing, interpolation and coarse grid operators. In standard multigrid, full coarsening, damped Jacobi or Gauss–Seidel smoothing, and linear interpolation are often used. Classical convergence theory and practice shows that these simple choices are enough to achieve mesh independent convergence.

For PDE problems with nonsmooth coefficients, however, mesh-independent convergence does not necessarily result in fast convergence. The nonsmoothness of the PDE coefficients typically lead to a large constant  $C_0$  in (13) and (14). Thus, multigrid converges slowly when the coefficients exhibit anisotropy [55], large jumps in discontinuity [1,19,38,39], or large oscillations [46,85]. Special techniques such as line Gauss–Seidel [19], semi-coarsening [40,41,81], algebraic multigrid [14,25,76,79,83], frequency decomposition [42,56,85], and homogenization [46], are used to handle some of these cases. In the next sections, we survey the state-of-the-art of each individual multigrid components and discuss how they bring insight into the design of robust multigrid methods.

## 2. Interpolation

Sophisticated designs of interpolation have been the key in developing robust multigrid methods. The many different methods can be generally divided into four categories ranging from geometric specific to purely algebraic. The structured grid approach takes advantages of the special PDE and algebraic structures associated with the Cartesian grids. The unstructured grid approach exploits the given grid information to derive interpolations. The algebraic multigrid approach, on the other



Fig. 2. Linear interpolation makes an O(h) error for a typical solution of the PDEs whose coefficient is piecewise constant.

hand, focuses on the algebraic aspect and derives interpolation from the residual equations. A recent approach constructs interpolation based on energy minimization which exploits the properties of the underlying PDEs while allowing general computational domains.

In the following, without loss of generality, we only discuss interpolation from coarse to fine, since the Galerkin process will automatically generate a multigrid method; see Section 1.2. Thus, superscripts *h* or *H* are used to denote quantities in the fine or coarse grid. Moreover, we sometimes describe the construction of coarse grid basis functions rather than the interpolation operators since they are essentially the same. In the finite element context, the coarse grid space  $V^H$  is often a subspace of  $V^h$ . Thus, if  $\{\phi_i^h\}_{i=1}^n$  and  $\{\phi_i^H\}_{i=1}^m$  are the nodal basis for  $V^h$  and  $V^H$ , respectively, then we have the following equality:

$$[\phi_1^H \cdots \phi_m^H] = [\phi_1^h \cdots \phi_n^h] \mathscr{I}_h^H,$$

where  $\mathscr{I}_{h}^{H}$  is the interpolation matrix. Hence, the set of coarse grid basis functions defines an interpolation, and vice versa. In particular, in the subsequent sections on the agglomeration unstructured grid approach and energy minimization approach, we shall describe the constructions of the coarse grid basis in place of interpolation. We note that the coarse grid subspaces need not be nested, for instance, in geometric unstructured grid multigrid methods. However, multigrid methods resulting from nested coarse subspaces are generally more robust, and hence we shall focus on this case in the next sections.

Before going on, we first discuss a well-known interpolation technique in one dimension, which is one of the earliest attempts to construct robust interpolation operator for nonsmooth coefficients. It turns out the basic design strategies in higher dimensions can be viewed as trying to extend this one-dimensional approach.

#### 2.1. One dimension

For nonsmooth coefficient PDEs, linear interpolation is not able to accurately approximate the irregular shape of the numerical solutions during the multigrid process. For example, Fig. 2 shows a typical solution of the PDEs whose coefficient is piecewise constant. In the worst case, linear interpolation can make an O(h) error, which is much poorer than the usual  $O(h^2)$  error.

A robust interpolation can be constructed by solving local PDEs [55]. Given the values  $v_{2i}$  and  $v_{2i+2}$  at the coarse grid points  $x_{2i}$  and  $x_{2i+2}$ , respectively, the value  $v_{2i+1}$  is computed by solving a

homogeneous two-point boundary value problem:

$$-\frac{d}{dx}a(x)\frac{d}{dx}v(x) = 0, \quad x \in (x_{2i}, x_{2i+2}),$$
  

$$v(x_{2i}) = v_{2i}, \quad v(x_{2i+2}) = v_{2i+2}.$$
(15)

Suppose a(x) is piecewise constant, for instance,  $a(x) \equiv a^-$ ,  $x_{2i} < x < x_{2i+1}$ , and  $a(x) \equiv a^+$ ,  $x_{2i+1} < x < x_{2i+2}$ . Then the finite element solution of (15) yields

$$v_{2i+1} = \frac{a^-}{a^- + a^+} v_{2i} + \frac{a^+}{a^- + a^+} v_{2i+2}.$$
(16)

The new interpolated solution is more accurate at the discontinuities. It is well known that the resulting multigrid is very robust and converges rapidly for nonsmooth coefficient a(x).

The local PDE approach has the property of preserving flux continuity. It can be proved [55] that the interpolated v given by (16) satisfies the jump condition

$$\lim_{x \to x_{2i+1}^{-}} a(x)v'(x) = \lim_{x \to x_{2i+1}^{+}} a(x)v'(x)$$

at  $x_{2i+1}$  which the exact solution does. In fact, the converse is also true; that is, if v satisfies the jump condition, then it solves the local PDE (15).

The interpolation can be interpreted by pure linear algebra. Ordering the noncoarse grid points  $\mu_F$  and then the coarse grid points  $\mu_C$ , we can write the permuted matrix, still denoted by  $\mathscr{A}^h$ , in a  $2 \times 2$  block form

$$\begin{bmatrix} \mathscr{A}_{11} & \mathscr{A}_{12} \\ \mathscr{A}_{21} & \mathscr{A}_{22} \end{bmatrix} \begin{bmatrix} \mu_{\rm F} \\ \mu_{\rm C} \end{bmatrix} = \begin{bmatrix} b_{\rm F} \\ b_{\rm C} \end{bmatrix},$$

where  $\mathscr{A}_{11}$  is a diagonal matrix. After eliminating  $\mu_F$ , we obtain the Schur complement equation for  $\mu_C$ :

$$\mathscr{S}\mu_{\rm C} = b_{\rm C} - \mathscr{A}_{21}\mathscr{A}_{11}^{-1}b_{\rm F},\tag{17}$$

where  $\mathscr{S} = \mathscr{A}_{22} - \mathscr{A}_{21}\mathscr{A}_{11}^{-1}\mathscr{A}_{12}$ . Define the interpolation and restriction matrices, respectively, by

$$\mathscr{I}_{H}^{h} = \begin{bmatrix} -\mathscr{A}_{11}^{-1} \mathscr{A}_{12} \\ I \end{bmatrix}, \quad \text{and} \quad \mathscr{I}_{h}^{H} = (\mathscr{I}_{H}^{h})^{\mathsf{T}} = [-\mathscr{A}_{21} \mathscr{A}_{11}^{-1}, \quad I].$$
(18)

Then  $\mathscr{S} = \mathscr{I}_h^H \mathscr{A}^h \mathscr{I}_H^h$  is precisely the coarse grid matrix  $\mathscr{A}^H$  in the multigrid context, and (17) is the usual coarse grid equation for  $\mu_C$  with the right-hand side given by restriction:  $\mathscr{I}_h^H[b_F, b_C]^T$ . The noncoarse grid values  $\mu_F$  are obtained by backward substitution:

$$\mu_{\rm F} = -\mathscr{A}_{11}^{-1} \mathscr{A}_{12} \mu_{\rm C} + \mathscr{A}_{11}^{-1} b_{\rm C}. \tag{19}$$

Using (19), we can write  $[\mu_{\rm F}, \mu_{\rm C}]^{\rm T}$  as

$$\begin{bmatrix} \mu_{\rm F} \\ \mu_{\rm C} \end{bmatrix} = \mathscr{I}_{H}^{h} \mu_{\rm C} + \begin{pmatrix} \mathscr{A}_{11}^{-1} & 0 \\ 0 & 0 \end{pmatrix} (b^{h} - \mathscr{A}^{h} \mathscr{I}_{H}^{h} \mu_{\rm C}).$$
<sup>(20)</sup>

Thus, the backward substitution can be interpreted as applying a Jacobi relaxation smoothing on the noncoarse grid points to the interpolated solution  $\mathscr{I}_{H}^{h}\mu_{C}$ .

We note that the block Gaussian elimination can be also written as block LU form:

$$\mathscr{A}^{h} = \begin{bmatrix} \mathscr{A}_{11} & \mathscr{A}_{12} \\ \mathscr{A}_{21} & \mathscr{A}_{22} \end{bmatrix} = \begin{bmatrix} I & 0 \\ \mathscr{A}_{21}\mathscr{A}_{11}^{-1} & I \end{bmatrix} \begin{bmatrix} \mathscr{A}_{11} & 0 \\ 0 & \mathscr{A}^{H} \end{bmatrix} \begin{bmatrix} I & \mathscr{A}_{11}^{-1}\mathscr{A}_{12} \\ 0 & I \end{bmatrix},$$

where the interpolation  $\mathscr{I}_{H}^{h}$  and the restriction  $\mathscr{I}_{h}^{H}$  operators are related to the upper and lower triangular factors, respectively.

To summarize, the block Gaussian elimination process is a two-level multigrid with  $\mathscr{I}_{H}^{h}$  and  $\mathscr{I}_{h}^{H}$  defined as in (18), and post-smoothing step (20). Furthermore, the inversion of the Schur complement in (17) can be done by recursively applying the previous procedure to  $\mathscr{S}$ . The resulting algorithm is known as cyclic reduction, and the corresponding multigrid method is a backslash cycle [99]. Moreover, it can be easily verified that the particular matrix-dependent interpolation matrix defined in (18) is precisely the one obtained by solving local PDEs.

Lastly, one may interpret the local PDE solve as an energy minimization to be described in Section 2.5. These four interpretations of the same interpolation: local PDE solve, flux continuity, the Schur complement, and energy minimization constitute the basic design principles of constructing the robust interpolations discussed in the following sections.

Finally, we remark that convergence analysis of robust multigrid methods in general is very limited in the literature since the interpolation operator is usually complicatedly defined. However, in one dimension, we have the following result [91].

**Theorem 2.1.** If the interpolation operator discussed above is used together with damped Jacobi or Gauss–Seidel smoothers, the resulting multigrid convergence is independent of the mesh size and the PDE coefficient a(x).

The proof uses the fact that the coarse grid basis functions from all the levels form an *A*-orthogonal hierarchical basis, and hence the damped Jacobi and Gauss–Seidel smoothers give an optimal constant bound for  $K_0$  and  $K_1$  which are the essential elements for estimating multigrid convergence by Theorem 1.2. Details can be found in [91].

#### 2.2. Structured grid approaches

Structured grids, in particular, Cartesian grids, have been very popular in applications for their regularity in geometry and in the algebraic structure of the resulting discretization matrix. Thus efficient numerical methods can be easily derived and employed. This approach also includes nested finite element grids obtained by recursive refinement where the discontinuities of the PDE coefficients are aligned with all the coarse grids. As a result, according to the domain decomposition theory [18,45,98], it can be proved that the convergence rate is independent of the size of the jumps in the coefficient using even the linear interpolation. However, in general, the discontinuities do not align with some of the coarse grids. Then other approaches are needed; see Sections 2.3–2.5.

Assuming Cartesian grids, the structured grid multigrid methods can generally be categorized into two: the stencil and Schur complement approaches.



Fig. 3. A portion of a fine grid with coarse grid points denoted by circles, and noncoarse grid points by crosses.

#### 2.2.1. Stencil

The one-dimensional local PDE technique cannot be applied directly to higher dimensions. Consider a portion of the fine grid with coarse grid points denoted by circles as shown in Fig. 3. In contrast to one dimension, the noncoarse grid points are not enclosed by only coarse grid points, and hence a local boundary value problem similar to (15) cannot be set up. The challenge of extending the one-dimensional technique to higher dimensions is to set up local PDEs appropriately.

Alcouffe et al. [1] used special harmonic averaging techniques to construct operator-induced interpolations, and in black box multigrid, Dendy [38,39] simplified the interpolation procedure by considering the stencil of the discrete operators. The key of the stencil approach is to first define interpolation on edges, i.e. noncoarse grid points lying on coarse grid lines (nodes 1–4 in Fig. 3), and then the interpolation at the interior (node 5) can be defined by a local PDE solve. In particular, black box multigrid defines the interpolation on the edges by preserving the continuity of flux across the interfaces. Thus, this multigrid method is efficient for PDE problems with discontinuous coefficients; the convergence rate is often independent of the mesh size and the size of the jumps.

The construction is as follows. Consider the nine-point stencil at the noncoarse grid points, for instance, node 1 (Fig. 3) lying on a horizontal edges:

$\alpha_{NW}^{(1)}$	$\alpha_N^{(1)}$	$\alpha_{NE}^{(1)}$	
$\alpha_W^{(1)}$	$\alpha_C^{(1)}$	$\alpha_E^{(1)}$	
$\alpha_{SW}^{(1)}$	$\alpha_S^{(1)}$	$\alpha_{SE}^{(1)}$	

The first and third rows are lumped to the second row, thus producing a one-dimensional three-point stencil. The flux preserving interpolation (16) yields

$$\tilde{v}_{2i+1,2j+1}^{-h} = \frac{\alpha_{NW}^{(1)} + \alpha_{W}^{(1)} + \alpha_{SW}^{(1)}}{\alpha_{N}^{(1)} + \alpha_{C}^{(1)} + \alpha_{S}^{(1)}} v_{i,j+1}^{H} + \frac{\alpha_{NE}^{(1)} + \alpha_{E}^{(1)} + \alpha_{SE}^{(1)}}{\alpha_{N}^{(1)} + \alpha_{C}^{(1)} + \alpha_{S}^{(1)}} v_{i+1,j+1}^{H}.$$
(21)

The lumping is used to preserve the continuity of the flux on the average along the vertical segment through nodes 1 and 2. The interpolation on vertical coarse grid lines (nodes 3 and 4) are defined analogously. Finally, since all the values on the edges are known, the interpolated value at node 5

can be obtained by solving a local PDE problem as in (15) where the solution is given by

$$\tilde{v}_{2i+1,2j+1} = \frac{\alpha_{NW}^{(5)}}{\alpha_C^{(5)}} v_{i,j+1}^H + \frac{\alpha_{N}^{(5)}}{\alpha_C^{(5)}} \tilde{v}_{2i+1,2j+2}^h + \frac{\alpha_{NE}^{(5)}}{\alpha_C^{(5)}} v_{i+1,j+1}^H + \frac{\alpha_{W}^{(5)}}{\alpha_C^{(5)}} \tilde{v}_{2i,2j+1}^h \\ + \frac{\alpha_E^{(5)}}{\alpha_C^{(5)}} \tilde{v}_{2i+2,2j+1}^h + \frac{\alpha_{SW}^{(5)}}{\alpha_C^{(5)}} v_{i,j}^H + \frac{\alpha_S^{(5)}}{\alpha_C^{(5)}} \tilde{v}_{2i+1,2j}^h + \frac{\alpha_{SE}^{(5)}}{\alpha_C^{(5)}} v_{i+1,j}^H.$$
(22)

Another stencil-based method is the matrix-dependent prolongation proposed by de Zeeuw [102]. It differs from the previous method in that the interpolated values on the edges are defined based on a decomposition of the nine-point stencil. Viewing the stencil as a  $3 \times 3$  matrix, it can be written as a linear combination of nine basis matrices, or equivalently, stencils. De Zeeuw considered a particular set of basis stencils corresponding to the discretization of the first and second derivatives. The interpolation formula which depends on the coefficients of the linear combination is very technical, and we refer the interested readers to [90,102] for details. This approach coincides with the black box multigrid of Dendy [38] for solving the model equation (1), and can be directly applied to certain nonsymmetric problems such as convection diffusion equations.

### 2.2.2. Schur complement and lumping

In this approach, we exploit the special algebraic structure associated with the discretization matrix arising from Cartesian grids. As in one dimension, the five-point stencil matrix in two dimensions can be written in a  $2 \times 2$  block form:

$$\mathscr{A}^{h} = \begin{bmatrix} \mathscr{A}_{11} & \mathscr{A}_{12} \\ \mathscr{A}_{21} & \mathscr{A}_{22} \end{bmatrix},$$

where  $\mathscr{A}_{11}$  is diagonal if the red-black ordering is used. However, the coarse grid matrix  $\mathscr{A}^H = \mathscr{G} = \mathscr{A}_{22} - \mathscr{A}_{21}\mathscr{A}_{11}^{-1}\mathscr{A}_{12}$  now corresponds to a nine-point stencil instead. Thus the algorithm cannot be repeated recursively.

In order to recover a five-point stencil structure, Reusken [76,77] applies a lumping strategy to the nine-point stencil coarse grid operator as follows. He replaced the nine-point stencil at a noncoarse grid point by a five-point stencil:

$$egin{bmatrix} lpha_{NW} & lpha_N & lpha_{NE} \ lpha_W & lpha_C & lpha_E \ lpha_{SW} & lpha_S & lpha_{SE} \end{bmatrix} 
ightarrow egin{bmatrix} 0 & eta_N & 0 \ eta_W & eta_C & eta_E \ 0 & eta_S & 0 \end{bmatrix},$$

where

$$\begin{split} \beta_N &= \alpha_N + \alpha_{NW} + \alpha_{NE}, \quad \beta_W &= \alpha_W + \alpha_{NW} + \alpha_{SW}, \\ \beta_C &= \alpha_C - (\alpha_{NW} + \alpha_{NE} + \alpha_{SW} + \alpha_{SE}), \quad \beta_E &= \alpha_E + \alpha_{NE} + \alpha_{SE}, \\ \beta_E &= \alpha_S + \alpha_{SW} + \alpha_{SE}. \end{split}$$

The lumping procedure essentially substitutes the unknowns  $v_{i-1,j+1}^H$ ,  $v_{i+1,j+1}^H$ ,  $v_{i-1,j-1}^H$ , and  $v_{i+1,j-1}^H$  by the unknowns  $v_{i,j-1}^H$ ,  $v_{i,j+1}^H$ ,  $v_{i-1,j}^H$ ,  $v_{i+1,j}^H$ , and  $v_{i,j}^H$  in the finite difference equation corresponding to the coarse grid point  $(x_i^H, y_j^H)$  based on a linear approximation; for instance,  $v_{i-1,j+1}^H \approx -v_{i,j}^H + v_{i,j+1}^H + v_{i-1,j}^H$ . In matrix form, the resulting discretization matrix becomes

$$\tilde{\mathscr{A}}^{H} = \begin{bmatrix} \tilde{\mathscr{A}}_{11} & \tilde{\mathscr{A}}_{12} \\ \tilde{\mathscr{A}}_{21} & \tilde{\mathscr{A}}_{22} \end{bmatrix},$$

where  $\tilde{A}_{11}$  is now a diagonal matrix. Moreover, the interpolation and restriction operators given by (18) are local, and the entire procedure can be repeated recursively.

## 2.3. Unstructured grid approaches

Unstructured gridding, which has a high flexibility of capturing complex geometrical shapes and providing adaptive local refinements, are useful for solving problems involving rapidly changing solutions, irregular boundaries, and multiscale geometries. However, as a result, the computational grids do not have any particular nested grid hierarchical structure to be exploited. Thus, the structured grid multigrid methods must be redesigned to handle the irregularity without losing too much in terms of complexity and performance.

The two main difficulties of designing multigrid methods on unstructured grids are the extraction of a hierarchy of coarser grids from a given fine grid, and the definition of the interpolation operators between grids. In the following sections, we describe several approaches of solving the two problems with increasing mathematical structures and decreasing intuition.

#### 2.3.1. Independent grids

The first approach is based on independently generated coarse grids and piecewise linear interpolation between the grids. Thus, one can use the grid generator which generates the unstructured fine grid to generate a sequence of coarser grids. Moreover, since the coarser grids consist of the usual finite elements, for instance, linear elements on triangles, linear interpolation and the coarse grid operator can be easily defined.

The advantage of this approach is convenience; the coarse grids can be generated by using the same grid generator which produced the original fine grid. The disadvantage is that the construction of the interpolation operator is very expensive since one has to identify which coarse triangles the noncoarse grid points are in. Another disadvantage is nonblack box nature of the coarse grid construction; the user is required to manually generate the grids.

## 2.3.2. Node nested grids

An alternative approach [29,32,52] is based on generating node-nested coarse grids, which are created by selecting subsets of a vertex set, retriangulating the subset, and using piecewise linear interpolation between the grids. This provides an automatic way of generating coarse grids and a simpler implementation (O(n)) of the interpolation. The main disadvantage is that critical geometrical details may be lost through the coarsening and retriangulation process, and hence special treatments are needed to preserve the important geometric features of the fine grid. Moreover, the coarse grid boundaries may not match that of the fine grid, and hence the boundary conditions must be incorporated properly, especially for Neumann boundary condition [30]. Another drawback is that in three dimensions, reternahedralization can be problematic.

**Remark.** Both the independent grid and node nested grid approaches are not designed to be robust for PDE problems with nonsmooth coefficients since linear interpolation is used.



Fig. 4. Typical macroelements in a computational domain.

#### 2.3.3. Agglomeration

To avoid the problem of losing geometrical details, a promising agglomeration technique [65,89] motivated by finite volume based methods is introduced. Instead of regenerating the coarse grids, neighboring fine grid elements are agglomerated together to form macroelements; see Fig. 4. For first order PDE problems, Venkatakrishnan and Mavriplis [89] used piecewise constant interpolation. More precisely, let  $\tau_i$  be a macroelement and  $\tau_i^H = \bigcup_{j \in N_i} \tau_j^h$ , where  $N_i$  is the set of neighboring nodes. Then

 $v_i^h = c_i \equiv \text{ constant}, \quad j \in N_i.$ 

However, their constant interpolation approach leads to slow convergence for second-order PDE problems since the basis is not stable. Within each macroelement, we need more robust weightings which mimic linear interpolation on structured grids.

In general, the coarse space  $V^H$  can be defined as the subspace spanned by a set of coarse grid basis functions  $\{\phi_i^H\}$  constructed as follows. For each coarse grid point *i*, define

$$\phi_i^H = \sum_{j \in \hat{N}_i} w_{ij}^h \phi_j^h + \phi_i^h, \tag{23}$$

where  $w_{ij}^h$  are appropriately chosen constants for robust interpolation, and

 $\tilde{N}_i = \{ j: \mathscr{A}_{i,j}^h \neq 0, \text{ and line segment } [x_i, x_j] \text{ is an edge of a macroelement} \}.$ 

Thus, the coarse grid basis functions are linear combinations of the fine grid basis, and  $V^H$  is a subspace of  $V^h$ ; that is, we obtain a nested sequence of subspaces by recursive construction. Moreover, the interpolation weights are given by the coefficients  $w_{ij}^h$ . To summarize, the construction of  $V^H$  consists of two parts. We agglomerate the fine grid elements to form macroelements, and then we define robust coarse grid basis functions on the macroelements.

Smoothed aggregation: For second-order PDEs, the piecewise constant basis functions are not effective since they possess large energy norm due to the discontinuities. Vaněk et al. [88] proposed the smoothed aggregation approach which applies a relaxation method to *smooth* the piecewise constant basis, and hence reducing the energy norm. More precisely, similar to agglomeration, the computational nodes are aggregated into disjoint aggregates based on the AMG coarsening

technique (cf. Section 4.2). Trying to achieve the approximation property (14), one defines a tentative interpolation operator as the piecewise constant prolongator:

$$(\tilde{\mathscr{I}}_{H}^{h})_{ij} = \begin{cases} 1 & \text{if } i \in \tau_{j}, \\ 0 & \text{otherwise,} \end{cases}$$

where  $\tau_j$  is the *j*th aggregate. Since the induced piecewise constant basis functions exhibit large energy, the stability property (13) is violated. One may *smooth* the basis by applying a damped Jacobi smoother to  $\tilde{\mathscr{J}}_{H}^{h}$  and obtain

$$\mathscr{I}_{H}^{h} = (I - \omega(\mathscr{D}^{h})^{-1}\mathscr{A}^{\mathrm{F}})\tilde{\mathscr{I}}_{H}^{h},$$

where  $\mathscr{A}^{\mathrm{F}}$  is the *filtered matrix* of  $\mathscr{A}^{h}$  defined as

$$\mathscr{A}_{ij}^{\mathrm{F}} = \begin{cases} \mathscr{A}_{ij}^{h} & \text{if } j \in N_{i}(\varepsilon), \ i \neq j \\ \mathscr{A}_{ii}^{h} - \sum_{j=1, j \neq i}^{n} (\mathscr{A}_{ij}^{h} - \mathscr{A}_{ij}^{\mathrm{F}}) & i = j, \\ 0 & \text{otherwise} \end{cases}$$

and  $N_i(\varepsilon) = \{j: |\mathscr{A}_{ij}^h| \ge \varepsilon \sqrt{\mathscr{A}_{ii}^h \mathscr{A}_{jj}^h}\}$ . Basically,  $\mathscr{A}^F$  is obtained by lumping the small entries in  $\mathscr{A}^h$  to the diagonal, thus controlling the number of nonzeros in the interpolation and coarse grid operators. Due to the smoothing effect of damped Jacobi, it smears the sharp edges of the coarse grid basis functions obtained from the piecewise constant prolongator, and hence the energies are reduced. Moreover, it can be proved that  $\mathscr{I}_H^h$  preserves constant if  $\widetilde{\mathscr{I}}_H^h$  does.

## 2.3.4. Others

Other unstructured grid multigrid approaches have also been proposed. Bank and Xu [9] developed an effective coarsening and interpolation strategy using the geometrical coordinates of the fine grid. The basic idea is to treat the fine grid as if it came from a refinement procedure, and then recover the refinement structure through a symbolic Gaussian elimination. Another multigrid method based on incomplete Gaussian elimination was proposed by Reusken [78]. Hackbusch and Sauter [57] constructed a triangulation for the computational domain by adaptively refining a coarse triangulation of a rectangular domain covering the computational domain. Thus, a hierarchy of coarse grids is naturally embedded in the fine triangulation.

## 2.4. Algebraic multigrid approaches

The structured and unstructured grid approaches make use of the grid information either explicitly or implicitly and hence are geometry dependent. The algebraic multigrid (AMG) approach [79], on the other hand, exploits the algebraic information of the discretization matrix. This approach was first introduced by Brandt et al. [25] and later popularized by Ruge and Stüben [79]. Other related work have been studied by Huang [59], and Chang et al. [34] to extend AMG to matrices which are not symmetric M-matrices.

The success of AMG is that for symmetric positive-definite *M*-matrices, for instance, matrices arising from discretization of the Laplacian operator, AMG is able to identify algebraically the *smooth* errors obtained from standard relaxation methods such as Gauss–Seidel, and then construct

interpolation operators accordingly to eliminate such errors. In the following sections, we describe a definition of algebraic smooth errors and discuss how they motivate the construction of an interpolation operator.

#### 2.4.1. Algebraic smoothness and strong connection

Let  $G^h$  be the iteration matrix of the relaxation smoother. In AMG, an error  $e^h$  is algebraically smooth if it is slow to converge with respect to  $G^h$ , i.e.,

$$\|G^h e^h\|_A \approx \|e^h\|_A.$$

For common relaxation smoothers, it can be argued [79] that an algebraically smooth error  $e^h$  is characterized by a small residual

$$r^h = \mathscr{A}^h e^h \approx 0,$$

in the sense that the residual norm is small compared to the error. Thus, we obtain a good approximation for  $e_i^h$  as a function of its neighboring values  $e_i^h$  by making  $r_i^h = 0$ :

$$r_i^h = \mathscr{A}_{ii}^h e_i^h + \sum_{j \in N_i} \mathscr{A}_{ij}^h e_j^h = 0,$$
(24)

where  $N_i = \{j \neq i: \mathscr{A}_{ij}^h \neq 0\}$ , the set of neighboring nodes of *i*. For symmetric *M*-matrices, the smooth error  $e^h$  often satisfies

$$\|e^h\|_A \ll \|e^h\|_D$$

where  $||e^h||_D^2 = (e^h, \mathcal{D}^h e^h)$ , and  $\mathcal{D}^h$  is the diagonal of  $\mathcal{A}^h$ . Note that  $||e^h||_A$  essentially measures the norm of the residual. We have the following inequality:

$$\frac{1}{2}\sum_{i,j} -\mathscr{A}^h_{ij}(e^h_i - e^h_j)^2 + \sum_i \left(\sum_j \mathscr{A}^h_{ij}\right) (e^h_i)^2 \ll \sum_i \mathscr{A}^h_{ii}(e^h_i)^2.$$
(25)

If  $\sum_{j \neq i} |\mathscr{A}_{ij}^h| \approx \mathscr{A}_{ii}^h$ , for instance,  $\mathscr{A}^h = \text{Laplacian}$ , then (25) can be written as

$$\sum_{j \neq i} \frac{\mathscr{A}_{ij}^{h}}{\mathscr{A}_{ii}^{h}} \frac{(e_{i}^{h} - e_{j}^{h})^{2}}{(e_{i}^{h})^{2}} \ll 1,$$
(26)

on the average for each *i*. Thus, if  $|\mathscr{A}_{ij}^h/\mathscr{A}_{ii}^h|$  is relatively large, then  $e_i^h$  and  $e_j^h$  must be close, and hence  $e_j^h$  is not negligible compared to  $e_i^h$ . The nodes *i* and *j* are called strongly connected if  $|\mathscr{A}_{ij}^h/\mathscr{A}_{ii}^h|$  is relatively large. This will be made more precise in (31). The strongly connectedness forms the basic notion for algebraic smoothing and interpolation.

#### 2.4.2. Algebraic interpolation

Suppose  $\mathscr{A}^h$  is a symmetric, weakly diagonally *M*-matrix. The derivation of the algebraic interpolation of Ruge and Stüben [79], again, stems from the idea of the one-dimensional interpolation, and has a strong connection with the stencil approach for Cartesian grids (Section 2.2.1). We start with the residual equation (24) corresponding to algebraic smooth errors where *i* is an index corresponding to a noncoarse grid point. Let *C* be the set of coarse grid points, and  $C_i \subseteq N_i \cap C$  the set of coarse grid points in a neighborhood of *i*. Given the coarse grid values  $e_k^h$ ,  $k \in C_i$ , we want to



Fig. 5. A portion of a fine grid with coarse grid points denoted by circles, and noncoarse grid points by crosses. The coarse and noncoarse grid point connections with respect to  $e_i^h$  are denoted by the superscripts.

define a value  $e_i^h$  such that  $r_i^h$  is as small as possible. If C is selected such that  $C_i = N_i$ , then the choice of  $e_i^h$ 

$$e_i^h = \sum_{k \in C_i} w_{ik}^h e_k^h, \quad w_{ik}^h = -\frac{\mathscr{A}_{ik}^h}{\mathscr{A}_{ii}}$$
(27)

leads to an ideal interpolation since  $r_i^h = 0$ . This is indeed equivalent to solving a local PDE with the *i*th node as interior noncoarse grid point and its neighbors as coarse grid points. However, such selection of *C* yields a dense coarse grid operator. Hence, in general, one has  $C_i \subset N_i$  and  $D_i \equiv N_i \setminus C_i \neq \emptyset$ . For example, on a Cartesian grid with standard full coarsening (Fig. 5), the two set of variables in  $C_i$  and  $D_i$  are indicated by their superscripts.

Consider (24) again where  $r_i^h$  is to be made 0:

$$\mathscr{A}^{h}_{ii}e^{h}_{i} + \sum_{k \in C_{i}} \mathscr{A}^{h}_{ik}e^{h}_{k} + \sum_{j \in D_{i}} \mathscr{A}^{h}_{ij}e^{h}_{j} = 0.$$

$$(28)$$

The value  $e_i^h$  to be interpolated can be obtained by (28) provided  $e_k^h$ 's and  $e_j^h$ 's are known. Given only the coarse grid values  $e_k^h$ , the idea is to first interpolate the noncoarse grid values  $e_j^h$ ,  $j \in D_i$ , by the  $e_k^h$ 's  $k \in C_i$ . For  $j \in D_i$ ,  $e_j^h$  is approximated by a weighted average:

$$e_{j}^{h} \approx \left(\sum_{k \in C_{i}} \mathscr{A}_{jk}^{h} e_{k}^{h}\right) \middle/ \left(\sum_{k \in C_{i}} \mathscr{A}_{jk}^{h}\right).$$
<sup>(29)</sup>

This local interpolation formula (29) is nothing but the one-dimensional local solve technique. Considering  $e_{j_1}^D$  in Fig. 5, by formula (29), we have

$$e_{j_1}^D \approx \frac{\mathscr{A}_{j_1,k_1}^h e_{k_1}^C + \mathscr{A}_{j_1,k_2}^h e_{k_2}^C}{\mathscr{A}_{j_1,k_1}^h + \mathscr{A}_{j_1,k_2}^h}.$$
(30)

Comparing (30) with (21), and using the stencil terminology, we note that the one-dimensional interpolation (29) used by AMG is obtained by the second row of the stencil at the node  $e_{j_1}^D$  whereas the one used by the black box multigrid in Section 2.2.1 is obtained by the average of the three rows of the stencil. Once  $e_j^h$ 's are known, the interpolated value  $e_i^h$  is then given by (28), which is the same as the local PDE solve formula (22) used by Dendy's black box multigrid.

In general, the computation of (29) may still be too large. We want to interpolate  $e_i^h$  by only those  $e_j^h$ 's which are significant. In view of the discussion after formula (26), the complexity can be reduced by the notion of strong connectedness. A point *i* is strongly connected to *j* if

$$-\mathscr{A}^{h}_{ij} \ge \theta \max_{l \ne i} \left\{ -\mathscr{A}^{h}_{il} \right\}$$
(31)

with  $0 < \theta \le 1$  as an input parameter. One only considers strong connections in the construction. Specifically, denote by  $S_i$  the set of all strong connections of point *i*. Define  $C_i = C \cap S_i$ , and let  $D_i^S = D_i \cap S_i$  and  $D_i^W = D_i \setminus S_i$ . For the weak connections  $(j \in D_i^W)$ ,  $e_j^h$  is simply replaced by  $e_i^h$ ; i.e., lumping the weak entries to the diagonal. For the strong connections  $(j \in D_i^S)$ ,  $e_j^h$  is defined as in (29).

Other variants of the algebraic interpolation are discussed in [79]. See also the recent survey by Stüben [84].

## 2.5. Energy minimization approaches

The AMG approach is purely algebraic and potentially applies to more general problems than the other methods, but the underlying PDE and geometry information, if exist, may not be fully utilized. In this section, we discuss another approach based on energy minimization which bridges the gap between the two extremes: geometry dependent and purely algebraic. It exploits the properties of the underlying PDEs using primarily the algebraic information, thus allowing general computational domains. The essential idea is motivated by the classical multigrid convergence theory, in particular, the stability and approximation inequalities (13) and (14) described in Section 1.3. The key is to construct coarse grid basis which has minimal energy while preserving the zero energy modes. Vaněk et al. [88] identified altogether seven objectives the coarse grid basis should satisfy:

- 1. Compact support.
- 2. Coarse supports should follow strong couplings.
- 3. Reasonable geometry of supports.
- 4. Bounded intersections among the coarse supports.
- 5. Polynomial (null space) preserving property.
- 6. Small energy of coarse basis functions.
- 7. Uniform  $l^2$  equivalence with  $L^2$ .

Based on these objectives, we try to construct coarse grid basis to achieve them, in particular, small energy, and null space preserving. For our model equation (1), the null space consists of constant functions.

## 2.5.1. Smoothed aggregation

In the smoothed aggregation method described in Section 2.3.3, one begins with the piecewise constant basis, which has high energy, and then smooths (or reduce the energy of) the basis by applying a relaxation method such as damped Jacobi to the interpolation operator. Thus, the energy of the basis is minimized locally. Moreover, it can be shown that the resulting basis preserves constants.



Fig. 6. Basis defined on macroelements. (Left)  $H^{1/2}$  minimization on edges and harmonic extension inside. (Center) Graph distance weightings on edges and harmonic extension inside. (Right) Graph distance weightings on edges and inside.

We remark that the energy norm of the smoothed basis may not necessarily be minimal. One may further reduce the energy by applying more damped Jacobi steps which, however, increases the supports of the basis functions. Consequently, the coarse grid operator becomes denser for more smoothed interpolation operator.

#### 2.5.2. Harmonic extension

A graph and agglomeration-based technique was proposed by Chan et al. [28,33]. The supports of the basis functions compose of macroelements formed by agglomeration. The basis functions on the edges are first constructed, and then they are extended by harmonic extension to the interiors. This procedure can be viewed as a local energy minimization. Moreover, if the basis preserves constant on the edges, it will also preserve constants in the interiors. There are several strategies to define the basis functions on the edges as well as in the interior with small energy. They are made more precise in the following.

 $H^{1/2}$  norm minimization+harmonic extension: One defines the coarse grid basis functions on the edges as linear functions which are minimized in the  $H^{1/2}$  norm-the interface analogue of the energy norm. More precisely, consider the coarse grid basis  $\phi_i^H$  on a macroelement with coarse grid points denoted by the black dots; see Fig. 6. Suppose  $\phi_i^H = \phi_0$ , a linear function, on the edge formed by  $x_0, x_1, x_2$  and  $x_3$ , i.e.,

$$\phi_0 = bx + c.$$

With two boundary conditions:  $\phi_0(x_0) = 1$ ,  $\phi_0(x_3) = 0$ , and hence one degree of freedom, one requires that  $\phi_0$  minimizes the functional (discrete  $H^{1/2}$  norm):

$$F(\phi_0) = \sum_{i=1}^{3} \sum_{j=i+1}^{3} \frac{h_i h_j}{h_{ij}^2} (\phi_0(x_i) - \phi_0(x_j))^2,$$

where  $h_i$  is the length of the edge  $(x_i, x_{i+1})$  and  $h_{ij} = |x_i - x_j|$ . After incorporating the two boundary conditions, the one-dimensional minimization of  $F(\phi_0)$  can be solved analytically. The same procedure is applied to the other edges, and the values at the interior points are obtained by harmonic extension.

*Graph distance+harmonic extension*: The  $H^{1/2}$  norm minimization combined with the harmonic extension approach is robust but the entire procedure may be too complex. A simplified variant is to use a simpler boundary interpolation based on graph distance. Note that  $x_1$  is distance 1 from  $x_0, x_2$  distance 2 from  $x_0$ , etc. Define

$$\phi_i^H(x_j) = \frac{3-j}{3}$$

on the edge formed by  $x_0, x_1, x_2$  and  $x_3$ , and piecewise linear on the edges. As in the previous approach, the values in the interior are given by the solution of a local PDE.

*Pure graph distance*: One may simplify the construction further by substituting the local PDE solve by a technique similar to graph distance. Suppose the macroelement has *m* number of coarse grid points on the boundary. Then the value of coarse grid basis function at each of the interior point is 1/m. Thus, in our example,  $\phi_i^H(x) = 1/3$ , x = interior points. Note that constants are preserved.

#### 2.5.3. Energy-minimizing interpolation

The previous approaches construct basis which first satisfies the approximation property, followed by minimizing the energy locally. Another approach proposed by Wan et al. [92] is to prescribe the energy minimization and constant preserving explicitly into the formulation of the interpolation operator while fixing the size of the supports. As opposed to all the previous approaches, we determine the interpolation values on the edges and in the interior at the same time by a minimization formulation, and hence we do not actually identify edges nor interiors. Meanwhile, the constant preserving property is enforced by a constraint setup which globally couples all the individual basis functions.

The idea is based on another interpretation of the one-dimensional interpolation now described. Consider the two-point boundary value problem (15) again. The equivalent variational formulation is given by

Thus, the local PDE formulation in one dimension is precisely minimizing the energy of the coarse grid basis functions. Moreover, if constant functions are in the kernel of the differential operator, the minimal energy basis will automatically preserve constants [91].

The extension to higher dimensions, however, is not obvious. First, the basis  $\{\phi_i^H\}$ , each of which has minimum energy, does not preserve constant functions. Second, the boundary of the support of each  $\phi_i^H$ , in general, consists of both coarse and noncoarse grid points and hence the boundary conditions of (15) need to be modified. A clue is provided in the two-level setting. Let  $\Pi_H^h$  be the usual nodal value interpolant. By the Cauchy–Schwarz and Poincaré inequalities, we obtain a rough estimate

$$\|\Pi_{H}^{h}v^{H}\|_{A} = \left\|\sum_{i} v^{H}(x_{i}^{H})\phi_{i}^{H}\right\|_{A} \leq \frac{C}{H} \left(\sum_{i} \|\phi_{i}^{H}\|_{A}^{2}\right)^{1/2} \|v^{H}\|_{A},$$
(33)

where *C* is a constant independent of *h*. Comparing (33) with the stability inequality (13), we see that the constant  $C_0$  in (13) depends on the total energy of  $\{\phi_i^H\}$ . Thus, the formulation is to minimize the sum of energies of  $\{\phi_i^H\}$  so that the constant  $C_0$  and hence the multigrid convergence will be improved.

Write the coarse grid basis function  $\phi_i^H$  as in (23). We determine the coefficients  $w_{ij}^h$  by solving a constrained minimization problem:

$$\min \frac{1}{2} \sum_{i=1}^{m} \|\phi_i^H\|_A^2 \quad \text{s.t.} \quad \sum_{i=1}^{m} \phi_i^H(x) = 1 \quad \text{in } \bar{\Omega}.$$
(34)

Lemma 2.2. An equivalent formulation of (15) and (32) is the global minimization

$$\min \frac{1}{2} \sum_{i=1}^{m} \|\phi_i^H\|_A^2 \quad s.t. \quad \sum_{i=1}^{m} \phi_i^H(x) = 1 \quad on \ [0,1].$$

**Proof.** See [92]. □

Thus, we see a way to naturally generalize the approach for generating a robust interpolation from one dimension to multiple dimensions.

**Remark.** (1) The values of the basis functions are defined implicitly by the solution of (34) and are not known explicitly in general. However, for the Laplacian, we recover exactly the bilinear interpolation on Cartesian grids [91], which is known to lead to optimal multigrid convergence for Poisson equation. (2) Like algebraic multigrid, the construction is purely algebraic. In other words, geometry and in particular the grid information are not needed. However, if the additional knowledge of the geometry is useful, for instance, semi-coarsening on Cartesian grids for anisotropic problems, we can still apply the same formulation. In fact, the coarse grid points can be provided geometrically by semi-coarsening or interface preserving techniques (cf. Section 4.3), or algebraically by AMG coarsening. Moreover, the formulation of the interpolation remains valid even if the coarse grid points do not form an independent set. (3) Mandel et al. [67] generalized this approach to solve systems of elliptic PDEs arising from linear elasticity problems.

Solution of the minimization problem: A detailed solution procedure is described in [92], and we only discuss the main steps here. It can be shown that the minimization problem (34) can be written as a constrained quadratic minimization. Thus, Newton's method only takes one iteration to convergence, which, however, need to invert the Jacobian matrix. Since the solution of the minimization problem is used as an interpolation operator for multigrid only, we do not need an exact solution. An efficient approximation can be obtained by a preconditioned conjugate gradient method. Empirical evidence shows that in most cases, only a few iterations suffice, except for oscillatory PDE coefficients. In [67], Mandel et al. showed that the interpolation obtained from the first step of the steepest descent procedure in solving (34) yields the same result as the smoothed aggregation with a single smoothing step.

## 2.5.4. AMGe

The use of energy minimization in the formulation of the interpolation operator has shown to be powerful in the energy-minimizing interpolation approach. AMGe, algebraic multigrid based on element stiffness matrices, proposed by Brezina et al. [26], uses the local measures of algebraic smoothness derived from multigrid theory to construct the interpolation operator. The key observation is that the interpolation must be able to approximate an eigenvector with an error bound proportional to the size of the associated eigenvalue. More precisely, the interpolation must be defined such that either of the following measures are bounded by a constant independent of h:

$$egin{aligned} M_1(Q,e^h) &= rac{((I-Q)e^h,(I-Q)e^h)}{(Ae^h,e^h)}, \ M_2(Q,e^h) &= rac{(A(I-Q)e^h,(I-Q)e^h)}{(Ae^h,Ae^h)}, \end{aligned}$$

where Q is a projection onto the range of the interpolation matrix  $\mathscr{I}_{H}^{h}$ . Note that Q is related to  $\mathscr{I}_{H}^{h}$  by

 $Q = \begin{bmatrix} 0 & \mathscr{I}_H^h \end{bmatrix},$ 

if the unknowns corresponding to the noncoarse grid points are ordered before the coarse grid points. The boundedness of  $M_1$  or  $M_2$  requires Q to accurately interpolate the eigenvectors corresponding to small eigenvalues, but not necessarily as accurate as for the eigenvectors corresponding to large eigenvalues. In addition, the quantities  $M_1$  or  $M_2$ , as opposed to the matrix entries used in standard AMG, give a potentially better measure of strong connectedness, especially for non-M-matrices.

In the previous approaches, the interpolation matrix is constructed by defining the coarse grid basis whose coefficients  $w_{ij}$  (cf. (23)) are the entries of the *j*th column of  $\mathscr{I}_H^h$ . Hence, the matrix  $\mathscr{I}_H^h$  is constructed column by column whereas in AMGe,  $\mathscr{I}_H^h$  is constructed row by row. Let  $q_i$  be the *i*th row of Q. Then,  $q_i$  is defined as the solution of the following min–max problem:

$$\min_{q_i} \max_{e^h \notin \text{Null}(\mathscr{A}_i^h)} M_{i,p}(q_i, e^h)$$
(35)

for p = 1 or 2. Here,  $M_{i,p}(q_i, e^h)$  is a local measure derived from the corresponding global measure  $M_p(Q, e^h)$  which is practically inaccessible, and  $\mathscr{A}_i^h$  is the sum of local element stiffness matrices connected with *i*. It can be shown [26] that the solution of (35) is to fit the eigenvectors of  $\mathscr{A}_i^h$  subject to the constraint that constants are preserved. Hence, it can be considered as another local energy minimization strategy.

Finally, we note that AMGe requires the knowledge of the element stiffness matrices which sometimes may not be conveniently available. Thus, this approach is less algebraic than the other energy-minimizing approaches.

## 3. Smoothing

Interpolation alone is not enough for fast convergence as the success of multigrid requires different components complement each other. The interpolation is effective only when the smoothers produce smooth errors either in the geometric sense, or in the algebraic sense (cf. Section 2.4). A classical example in the literature where smoothing plays an important role in improving the multigrid convergence is when solving PDEs with anisotropic coefficients, for instance,

$$\begin{aligned} &-\varepsilon u_{xx} - u_{yy} = f \quad \text{in } \Omega, \\ &u = 0 \quad \text{on } \partial\Omega. \end{aligned}$$
(36)

Assuming standard coarsening, it can be shown by Fourier analysis [19,94] that point relaxation methods as smoothers are not effective for small  $\varepsilon$  since the errors are only smoothed in the y-direction, and the errors in the x-direction can be highly oscillatory, leading to slow convergence of multigrid. In the next sections, we discuss the use of block\line relaxation methods, incomplete LU (ILU) factorization preconditioners, and sparse approximate inverses as smoothers for anisotropic as well as other types of elliptic PDE problems.

## 3.1. Block\line relaxation

A well-suited smoother can be derived from considering the limiting case when  $\varepsilon \to 0$ . The resulting PDE is a decoupled system of elliptic equations along the vertical lines, suggesting the use of block Gauss–Seidel as smoother where the blocks are associated with the lines in the y-direction, or equivalently, the direction of the anisotropy. It can be proved [19,55] that the two-grid method with this smoother converges independently of  $\varepsilon$ . In the case of variable coefficients where the anisotropy direction may change from y to x at different locations, one may alternate the direction of the line relaxation. Another option is to use the alternating direction implicit method [68].

The disadvantage of this approach is that the smoothing is most effective when the anisotropy is either in the x- or y-direction. Another problem is that it is essentially a Cartesian grid technique. Although similar idea can be adapted in unstructured grid computations [72], the determination of the lines or planes of anisotropy in three dimensions is complicated. Besides, inverting a line or a plane requires more work than point relaxations.

#### 3.2. ILU

One needs a *direction free* and robust iterative method as smoother for solving anisotropic problems, and we shall discuss two possibilities in this and the next section. An incomplete LU factorization based on the stencil pattern was studied by Wesseling [94] and Wittum [95]. Given a five-point stencil matrix, for instance, one may use the stencil pattern for the incomplete L, U factors. Specifically, an incomplete LU factorization can be written as

$$\mathscr{A}^h = \mathscr{L}^h \mathscr{U}^h + \mathscr{E}^h,$$

where the incomplete lower triangular factor  $\mathscr{L}^h$  has a three-point stencil structure corresponding to the lower triangular part of a five-point stencil matrix, and similarly for the upper triangular factor  $\mathscr{U}^h$ . Similar ideas can be applied to other stencil patterns. The resulting ILU smoother is effective for a wide range of directions of anisotropy. More precisely, suppose the model equation (36) is rotated by an angle  $\theta$ . Hemker [58] showed that ILU is an effective smoother for  $\pi/4 \leq \theta \leq \pi$ . However, it may lead to divergence for  $0 < \theta < \pi/4$ . In the latter, the line relaxation smoother is still effective since the angle is small. One may combine both ideas and derive an incomplete line LU (ILLU) smoother [82] which uses block triangular factors for  $\mathscr{L}^h$  and  $\mathscr{U}^h$ .

#### 3.3. Sparse approximate inverse

The drawback of ILU smoothers is the lack of parallelism, since the (incomplete) LU factorization process is sequential in nature. Another class of direction free smoothers, which are inherently parallel, are sparse approximate inverses (SAI).

Most sparse approximate inverse (SAI) approaches seek a sparse matrix  $\mathcal{M}^h$  so that the error of the residual is minimized in some measure. One of the earliest approaches was the Frobenius norm approach proposed by Benson [10] and Benson and Frederickson [13]:

$$\min_{\mathcal{M}^h} \| \mathscr{A}^h \mathscr{M}^h - I \|_F^2,$$

subject to some constraints on the number and position of the nonzero entries of  $\mathcal{M}^h$ . The minimization problem is equivalent to *n* independent least-squares problems:

$$\min_{m_j} \|\mathscr{A}^h m_j - e_j\|_2, \quad j = 1, \dots, n,$$
(37)

where  $m_j$  and  $e_j$  are the *j*th column of  $\mathcal{M}^h$  and *I*, respectively, and they can be solved in parallel. For efficient construction, the sparsity pattern may be selected as banded diagonal [63], for example, or determined adaptively by heuristic searching algorithms [35,51] which, however, may decrease parallelism.

Several SAI smoothers have been studied. Benson [11], and Benson and Banerjee [12] used a sparsity pattern based on graph neighbors. Recently, Huckle and Grote independently experimented a sparse approximate smoother based on SPAI [51] which adaptively search the nonzero pattern.

In the following, we describe the approach proposed by Tang and Wan [86]. Since the major cost of multigrid algorithms is smoothing, it is important to derive simple and yet effective sparsity patterns. In addition, the least-squares problems (37) must be solved efficiently. It turns out that a pre-defined pattern based on neighbors of the matrix graph is sufficient for effective smoothing [86]. Given a node j, define  $L_k(j)$  as its k-level neighbor set in graph distance. For instance,  $L_0(j)$  contains simply the set of stencil points in case of PDE problems. Furthermore, one modify the Frobenius norm approach (37) and introduce the (k, l)-level least-squares approximation:

$$\min_{m_j} \|\mathscr{A}^{k,l}m_j - e_j\|_2,$$

where  $\mathscr{A}^{k,l} \equiv \mathscr{A}^h(L_k(j), L_l(j))$  is the (k, l)-level local submatrix of  $\mathscr{A}^h$ . The sparsity pattern is determined by the *l*-level neighbors, and the size of the least squares matrix is controlled by the selections of *k* and *l*. Hence, the two main issues of SAI smoothers are handled. Moreover, it can be proved that high frequency errors will be damped away efficiently for k = 1 and l = 0 [86].

More importantly, SAI smoothers have the flexibility of using larger values of k and l to improve the smoothing quality for difficult PDE problems. The potential higher computational cost can be reduced by dropping strategies. For anisotropic coefficient PDEs, the matrix  $\mathcal{A}^h$  and its inverse typically have many small entries. Thus, one may drop the neighbors with weak connections in  $\mathcal{A}^{k,l}$ before computing the approximate inverse. This is essentially the same idea as line relaxation which only applies to structured grids. One may further reduce the cost by discarding small entries in  $\mathcal{M}^h$ . It has been shown empirically [86] that the resulting complexity is only twice as expensive as point Gauss–Seidel for the anisotropic problem (36). In addition, since the determination of the lines or planes of anisotropy is done algebraically and automatically, SAI smoothing is applicable to both structured and unstructured grid computations in higher dimensions.

## 4. Coarsening

The design of interpolation and smoothing operators has been the main focus in improving multigrid performance. Coarsening, the selection of coarse grid points, can be as important as interpolation and smoothing, for instance, AMG coarsening [79] and semi-coarsening [40,41,81] are both critical components in their respective algorithms. The former selects coarse grid points algebraically according to strong connections and is robust for discontinuous and anisotropic coefficient PDEs. The latter selects coarse grid points geometrically according to the direction of strong coupling for anisotropic coefficient PDEs. Another approach, interface preserving coarsening [91], selects coarse grid points geometrically according to the shape of interfaces.

#### 4.1. Semi-coarsening

For anisotropic coefficient problems, special smoothing techniques are discussed in Section 3 to improve multigrid efficiency. Coarsening has also shown to be another approach to recover fast multigrid convergence. The failure of standard multigrid is that the errors in the direction of weak anisotropy are not smoothed. Thus, they cannot be solved on the coarse grid. In the case of structured grids, one can apply standard coarsening to the direction of strong coupling only, i.e., the y-direction for the model problem (36), and select all the grid points as coarse grid points in the other directions, resulting in alternating y = constant lines of coarse grid points. The drawback, however, is that the overall cost of multigrid will increase.

A related coarsening technique is the use of multiple semi-coarsened grids proposed by Mulder [73]. For nonlinear PDE problems where the direction of anisotropy changes from time to time, Mulder performed the coarse grid correction on two semi-coarsened grids in both x- and y-direction on each level of grid. The complexity of the resulting algorithm turns out still to be proportional to the number of unknowns. Frequency decomposition multigrid, proposed by Hackbusch [56], is another method using several grid corrections. Three additional fully coarsened grids are formed by shifting the standard coarse grid by one grid point in the x- and/or y-direction. Moreover, special prolongation and restriction operators are used to include also the high frequencies on the additional coarse grids so that the union of the ranges of the prolongation operators is the same as the fine grid function space. The filtering decomposition by Wittum [96,97] is another coarse grid correction method. Instead of including all the high frequencies, the coarse grid operator is required to have the same effect as the fine grid operator on a selected set of vectors, for instance, discrete sine functions with different frequencies. This principle is similar to the probing method proposed by Chan and Mathew [31] in domain decomposition.

## 4.2. AMG coarsening

In algebraic multigrid [79], the selection of coarse grid points ties strongly with the algebraic multigrid interpolation. Divide the fine grid points into the set of coarse (C) and noncoarse (F) grid points. In the ideal case where C is chosen such that for each noncoarse grid point  $i \in F$ , its neighbors are all coarse gird points, i.e.,  $N_i = C_i$  (Section 2.4), the algebraic interpolation defined in (27) is exact; it is just Gaussian elimination as described in one dimension. Otherwise, the interpolation needs to approximate the values at the noncoarse grid point connections as given in

(29). On the other hand, the notion of strong connectedness is introduced to maintain sparsity by ignoring weak connections. Moreover, the approximation in (29) is more accurate if many strong noncoarse grid connection neighbors of j are actually in  $C_i$ . Summing up, there are two criteria for algebraic coarsening:

- 1. For each noncoarse grid point *i*, each strong connection  $j \in S_i$  should either be a coarse grid point  $(j \in C_i)$ , or should be strongly connected to at least one point in  $C_i$ .
- 2. The set of coarse grid point should form a maximal independent set with respect to strong connections.

The two criteria, in general, are conflicting with each other. Usually, the second criterion is used to select a potential small set of coarse grid points. Then noncoarse grid points may be added to satisfy the first criterion.

## 4.3. Other approach

Algebraic coarsening selects coarse grid points based on matrix entries. A recent approach, the interface preserving coarsening, can be considered as its geometric counterpart which is specifically designed for discontinuous coefficient PDEs. For this class of problems, multigrid is typically improved by a sophisticated interpolation such as those described in Section 2 which captures the discontinuous behavior of the derivatives of the solution along the interfaces. This is particularly important since the interface may not necessarily align with the coarse grids as usually demanded by theory [18,45,98]. However, linear interpolation can be just fine if the interface aligns with *all* coarse grids. The main idea of interface preserving coarsening [91] is thus to select coarse grid points which resolve the shape of the interface on all coarse grids.

## 5. Conclusion

Significant advances have been made in robust multigrid methods for elliptic linear systems in recent years. The variety of techniques developed have been steadily maturing, but will still have not quite reached the holy grail for multigrid methods that is algebraic and easily parallelizable, with complexity proportional to the number of unknowns, and with rate of convergence independent of the mesh size, the nature of the PDE coefficients, and the computational grids. The gray box philosophy may ultimately lead to a more flexible approach to developing multigrid algorithms which can make optimal use of any available information. Our discussion has been confined to the algorithmic developments. Parallelization and theoretical issues are nonetheless essential for the practical and intelligent use of multigrid in large scale numerical simulations.

There is still plenty of room for improvements in every aspects of robust multigrid to come in the years ahead. For instance, it is likely that the energy minimization principle will continue to be useful in constructing robust interpolation operators as new techniques are developed. On the other hand, the interplay between coarse grid basis and interpolation provides another perspective for constructing robust interpolation, for instance, through the use of special finite element basis recently developed for discontinuous and oscillatory coefficient PDEs.

## References

- [1] R.E. Alcouffe, A. Brandt, J.E. Dendy Jr., J.W. Painter, The multi-grid method for the diffusion equation with strongly discontinuous coefficients, SIAM J. Sci. Statist. Comput. 2 (4) (1981) 430–454.
- [2] G.P. Astrachancer, An iterative method of solving elliptic net problems, USSR Comput. Math. Math. Phys. 11 (1971) 171–182.
- [3] O. Axelsson, P. Vassilevski, Algebraic multilevel preconditioning methods, Part I, Numer. Math. 56 (1989) 157–177.
- [4] O. Axelsson, P. Vassilevski, Algebraic multilevel preconditioning methods, Part II, SIAM J. Numer. Anal. 27 (1990) 1569–1590.
- [5] N.S. Bachvalov, On the convergence of a relaxation method with natural constraints on the elliptic operator, USSR Comput. Math. Phys. 6 (1966) 101–135.
- [6] R.E. Bank, T. Dupont, An optimal order process for solving elliptic finite element equations, Math. Comp. 36 (1981) 35–51.
- [7] R.E. Bank, T.F. Dupont, H. Yserentant, The hierarchical basis multigrid method, Numer. Math. 52 (1988) 427-458.
- [8] R.E. Bank, R.K. Smith, The incomplete factorization multigraph algorithm, SIAM J. Sci. Comput. 20 (1999) 1349–1364.
- [9] R.E. Bank, J. Xu, An algorithm for coarsening unstructured meshes, Numer. Math. 73 (1996) 1-36.
- [10] M.W. Benson, Iterative solution of large scale linear systems, Master's Thesis, Lakehead University, Thunder Bay, Ontario, 1973.
- [11] M.W. Benson, Frequency domain behavior of a set of parallel multigrid smoothing operators, Internat. J. Comput. Math. 36 (1990) 77–88.
- [12] M.W. Benson, R.N. Banerjee, An approximate inverse based multigrid approach to the biharmonic problem, Internat. J. Comput. Math. 40 (1991) 201–210.
- [13] M.W. Benson, P.O. Frederickson, Iterative solution of large sparse linear systems arising in certain multidimensional approximation problems, Utilitas Math. 22 (1982) 127–140.
- [14] D. Braess, Towards algebraic multigrid for elliptic problems of second order, Computing 55 (1995) 379–393.
- [15] H. Brakhage, Über die numerishe behandlung von integralgleichungen nach der quadraturformelmethode, Numer. Math. 2 (1960) 183–196.
- [16] J. Bramble, Multigrid Methods, Longman Scientific & Technical, Essex, UK, 1993.
- [17] J. Bramble, J. Pasciak, J. Xu, Parallel multilevel preconditioners, Math. Comp. 55 (1990) 1–22.
- [18] J. Bramble, J. Xu, Some estimates for a weighted  $l^2$  projection, Math. Comp. 56 (1991) 463–476.
- [19] A. Brandt, Multi-level adaptive solutions to boundary-value problems, Math. Comp. 31 (1977) 333-390.
- [20] A. Brandt, Guide to multigrid developments, in: W. Hackbusch, U. Trottenberg (Eds.), Multigrid Methods, Lectures Notes in Mathematics, Vol. 960, Springer, Berlin, 1982, pp. 220–312.
- [21] A. Brandt, Multigrid techniques: 1984 guide with applications to fluid dynamics, GMD-Studien No. 85, Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin, 1984.
- [22] A. Brandt, Algebraic multigrid theory: the symmetric case, Appl. Math. Comput. 19 (1986) 23-56.
- [23] A. Brandt, The gauss center research in multiscale scientific computation, Electron. Trans. Numer. Anal. 6 (1997) 1–34.
- [24] A. Brandt, General highly accurate algebraic coarsening schemes, Proceedings of the Nineth Copper Mountain Conference on Multigrid Methods, Copper Mountain, CO, 1999.
- [25] A. Brandt, S. McCormick, J.W. Ruge, Algebraic multigrid (AMG) for automatic multigrid solution with application to geodetic computations, Technical Report, Inst. for Computational Studies, Fort Collins, CO, 1982.
- [26] M. Brezina, A.J. Cleary, R.D. Falgout, V.E. Henson, J.E. Jones, T.A. Manteuffel, S.F. McCormick, J.W. Ruge, Algebraic multigrid based on element interpolation (AMGe), Technical Report UCRL-JC-131752, Lawrence Livermore National Laboratory, 1998.
- [27] W. Briggs, A Multigrid Tutorial, SIAM, Philadelphia, PA, 1987.
- [28] T.F. Chan, S. Go, L. Zikatanov, Lecture Notes on Multilevel Methods for Elliptic Problems on Unstructured Grids, Lecture Course 28th Computational Fluid Dynamics, March 3–7, 1997, von Karman Institute for Fluid Dynamics, Belgium, 1997.

- [29] T.F. Chan, S. Go, J. Zou, Multilevel domain decomposition and multigrid methods for unstructured meshes: algorithms and theory, Technical Report CAM 95-24, Department of Mathematics, UCLA, May 1995.
- [30] T.F. Chan, S. Go, J. Zou, Boundary treatments for multilevel methods on unstructured meshes, Technical Report CAM 96-30, Department of Mathematics, UCLA, September 1996.
- [31] T.F. Chan, T.P. Mathew, The interface probing technique in domain decomposition, SIAM J. Matrix Anal. Appl. 13 (1992) 212–238.
- [32] T.F. Chan, B. Smith, J. Zou, Multigrid and domain decomposition methods for unstructured meshes, in: I.T. Dimov, Bl. Sendov, P. Vassilevski (Eds.), Proceedings of the Third International Conference on Advances in Numerical Methods and Applications, Sofia, Bulgaria, World Scientific, Singapore, 1994, pp. 53–62.
- [33] T.F. Chan, J. Xu, L. Zikatanov, An agglomeration multigrid method for unstructured grids, Technical Report CAM 98-8, Department of Mathematics, UCLA, February 1998.
- [34] Q. Chang, Y.S. Wong, Z. Li, New interpolation formulas of using geometric assumptions in the algebraic multigrid method, Appl. Math. Comput. 50 (1992) 223–254.
- [35] E. Chow, Y. Saad, Approximate inverse preconditioners via sparse-sparse iterations, SIAM J. Sci. Comput. 19 (1998) 995–1023.
- [36] A.J. Cleary, R.D. Falgout, V.E. Henson, J.E. Jones, Coarse-grid selection for parallel algebraic multigrid, Technical Report UCRL-JC-130893, Lawrence Livermore National Laboratory, 1998.
- [37] A.J. Cleary, R.D. Falgout, V.E. Henson, J.E. Jones, T.A. Manteuffel, S.F. McCormick, G.N. Miranda, J.W. Ruge, Robustness and scalability of algebraic multigrid, Technical Report UCRL-JC-130718, Lawrence Livermore National Laboratory, 1998.
- [38] J.E. Dendy Jr., Black box multigrid, J. Comput. Phys. 48 (1982) 366-386.
- [39] J.E. Dendy Jr., Black box multigrid for nonsymmetric problems, Appl. Math. Comput. 13 (1983) 261–283.
- [40] J.E. Dendy Jr., M.P. Ida, J.M. Rutledge, A semi-coarsening multigrid algorithm for SIMD machines, SIAM J. Sci. Statist. Comput. 13 (1992) 1460–1469.
- [41] J.E. Dendy Jr., S.F. McCormick, J.W. Ruge, T.F. Russell, S. Schaffer, Multigrid methods for three-dimensional petroleum reservoir simulation, 10th Symposium on Reservoir Simulation, Houston, Society of Petroleum Engineers (SPE), February 6–8 1989, pp. 19–25.
- [42] J.E. Dendy Jr., C.C. Tazartes, Grandchild of the frequency decomposition multigrid method, SIAM J. Sci. Comput. 16 (2) (1994) 307–319.
- [43] C.C. Douglas, Caching in with multigrid algorithms: problems in two dimensions, Parallel Algorithms Appl. 9 (1996) 195–204.
- [44] C.C. Douglas, U. Rüde, J. Hu, M. Bittencourt, A guide to designing cache aware multigrid algorithms, submitted for publication.
- [45] M. Dryja, M. Sarkis, O. Widlund, Multilevel Schwarz methods for elliptic problems with discontinuous coefficients in three dimensions, Numer. Math. 72 (1996) 313–348.
- [46] B. Engquist, E. Luo, Convergence of a multigrid method for elliptic equations with highly oscillatory coefficients, SIAM J. Numer. Anal. 34 (1997) 2254–2273.
- [47] R.P. Fedorenko, A relaxation method for solving elliptic difference equations, USSR Comput. Math. Phys. 1 (1961) 1092–1096.
- [48] R.P. Fedorenko, The speed of convergence of one iterative process, USSR Comput. Math. Phys. 4 (1964) 227–235.
- [49] T. Grauschopf, M. Griebel, H. Regler, Additive multilevel-preconditioners based on bilinear interpolation, matrix-dependent geometric coarsening and algebraic multigrid coarsening for second order elliptic pdes, Appl. Numer. Math. 23 (1997) 63–96.
- [50] M. Griebel, Multilevel algorithms considered as iterative methods on semidefinite systems, SIAM J. Sci. Comput. 15 (1994) 547–565.
- [51] M. Grote, T. Huckle, Parallel preconditioning with sparse approximate inverses, SIAM J. Sci. Comput. 18 (1997) 838–853.
- [52] H. Guillard, Node-nested multigrid method with Delaunay coarsening, Technical Report 1898, INRIA, Sophia Antipolis, France, 1993.
- [53] W. Hackbusch, A fast iterative method solving Poisson's equation in a general region, in: R. Bulirsch, R.D. Grigorieff, J. Schröder (Eds.), Proceedings, Oberwolfach, July 1976, Lecture Notes in Mathematics, Vol. 631, Springer, Berlin, 1978, pp. 51–62.

- [54] W. Hackbusch, Survey of convergence proofs for multigrid iterations, in: J. Frehse, D. Pallaschke, U. Trottenberg (Eds.), Special Topics of Applied Mathematics, North-Holland, Amsterdam, 1980, pp. 151–164.
- [55] W. Hackbusch, Multi-grid Methods and Applications, Springer, Berlin, 1985.
- [56] W. Hackbusch, The frequency decomposition multigrid method, Part I: application to anisotropic equations, Numer. Math. 56 (1989) 229–245.
- [57] W. Hackbusch, S. Sauter, Adaptive composite finite elements for the solutions of pdes containing non-uniformly distributed micro-scales, Mat. Model. 8 (1996) 31–43.
- [58] P.W. Hemker, Multigrid methods for problems with a small parameter, in: D.F. Griffiths (Ed.), Proceedings, Dundee, June–July 1983, Lecture Notes in Mathematics, Vol. 1066, Springer, Berlin, 1983, pp. 106–121.
- [59] W.Z. Huang, Convergence of algebraic multigrid methods for symmetric positive definite matrices with weak diagonal dominance, Appl. Math. Comput. 46 (1991) 145–164.
- [60] D.C. Jespersen, Recent developments in multigrid methods for the steady Euler equations, Lecture Notes for Lecture Series on Computational Fluid Dynamics, von Karman Institute for Fluid Dynamics, Rhode-St.-Genese, Belgium, 1984.
- [61] J. Jones, S. McCormick, Parallel multigrid methods, in: D. Keyes, A. Sameh, V. Venkatakrishnan (Eds.), Parallel Numerical Algorithms, NASA/LaRC Interdisciplinary Series in Science and Engineering, Kluwer, Dordrecht, 1997.
- [62] R. Kettler, J.A. Meijerink, A multigrid method and a combined multigrid-conjugate gradient method for elliptic problems with strongly discontinuous coefficients in general domains, Shell Publ. 604, KSEPL, Rijswijk, 1981.
- [63] L.Yu. Kolotilina, A.Yu. Yeremin, Factorized sparse approximate inverse preconditionings I, theory, SIAM J. Matrix Anal. Appl. 14 (1993) 45–58.
- [64] A. Krechel, K. Stüben, Operator dependent interpolation in algebraic multigrid, in: Lecture Notes in Computational Science and Engineering, Vol. 3. Proceedings of the Fifth European Multigrid Conference, Stuttgart, October 1–4, 1996, Springer, Berlin, 1998.
- [65] M.H. Lallemand, H. Steve, A. Dervieux, Unstructured multigridding by volume agglomeration: current status, Comput. Fluids 21 (1992) 397–433.
- [66] P.L. Lions, On the Schwarz alternating method, I, in: R. Glowinski, G.H. Golub, G.A. Meurant, J. Periaux (Eds.), First International Conference on Domain Decomposition Methods for Partial Differential Equations, SIAM, Philadelphia, PA, 1988.
- [67] J. Mandel, M. Brezina, P. Vaně k, Energy optimization of algebraic multigrid bases, Technical Report UCD/CCM 125, Center for Computational Mathematics, University of Colorado, Denver, 1998.
- [68] G.I. Marchuk, Splitting and alternating direction methods, in: P.G. Ciarlet, J.L. Lions (Eds.), Handbook of Numerical Analysis, Vol. 1, North-Holland, Amsterdam, 1990.
- [69] A.M. Matsokin, S.V. Nepomnyaschikh, A Schwarz alternating method in a subspace, Soviet Math. 29 (10) (1985) 78–84.
- [70] O.A. McBryan, P.O. Frederickson, J. Linden, A. Schuller, K. Solchenbach, K. Stüben, C.A. Thole, U. Trottenberg, Multigrid methods on parallel computers a survey of recent developments, Impact Comput. Sci. Eng. 3 (1991) 1–75.
- [71] S. McCormick, Multigrid methods for variational problems: general theory for the V-cycle, SIAM J. Numer. Anal. 22 (1985) 634–643.
- [72] E. Morano, D.J. Mavriplis, V. Venkatakrishnan, Coarsening strategies for unstructured multigrid techniques with application to anisotropic problems, SIAM J. Sci. Comput. 20 (1999) 393–415.
- [73] W. Mulder, A high-resolution Euler solver, Technical Report CAM 89-08, Department of Mathematics, UCLA, April 1989.
- [74] R.A. Nicolaides, On multiple grid and related techniques for solving discrete elliptic systems, J. Comput. Phys. 19 (1975) 418–431.
- [75] C.W. Oosterlee, The covergence of parallel multiblock multigrid, Appl. Numer. Math. 19 (1995) 115–128.
- [76] A. Reusken, Multigrid with matrix-dependent transfer operators for a singular perturbation problem, Computing 50 (1993) 199–211.
- [77] A. Reusken, Multigrid with matrix-dependent transfer operators for convection-diffusion problems, in: Multigrid Methods, Vol. IV, Lectures in Mathematics, Springer, Berlin, 1993.
- [78] A. Reusken, A multigrid method based on incomplete Gaussian elimination, J. Numer. Linear Algebra Appl. 3 (1996) 369–390.

- [79] J.W. Ruge, K. Stüben, Algebraic multigrid, in: S. McCormick (Ed.), Multigrid Methods, SIAM, Philadelphia, PA, 1987, pp. 73–130.
- [80] B. Smith, P. Bjørstad, W. Gropp, Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations, Cambridge University Press, Cambridge, 1996.
- [81] R.A. Smith, A. Weiser, Semicoarsening multigrid on a hypercube, SIAM J. Sci. Statist. Comput. 13 (1992) 1314–1329.
- [82] P. Sonneveld, P. Wesseling, P.M. de Zeeuw, Multigrid and conjugate gradient methods as convergence acceleration techniques, in: H. Holstein, D.J. Paddon (Eds.), Short Course on Multigrid Methods, Proceedings, Bristol, September 1983, Oxford Press, Oxford, 1985.
- [83] K. Stüben, Algebraic multigrid: experience and comparisons, Appl. Math. Comput. 13 (1983) 419-451.
- [84] K. Stüben, Algebraic multigrid (AMG): an introduction with applications, Technical Report 53, GMD, March 1999, in: U. Trottenbery, C.W. Oosterlee, A. Schüller (Eds.), Multigrid (appendix), Academic Press, New York, 1999, to appear.
- [85] S. Ta'asan, Multigrid methods for highly oscillatory problems, Ph.D. Thesis, Weizmann Institute of Science, Rehovat, Israel, 1984.
- [86] W.P. Tang, W.L. Wan, Sparse approximate inverse smoother for multi-grid, SIAM J. Matrix Anal. Appl. (1999), to appear.
- [87] U. Trottenberg, C.W. Oosterlee, A. Schüller, Basic, Parallel and Adaptive Multigrid Methods, Academic Press, New York, 1998, to appear.
- [88] P. Vaně k, J. Mandel, M. Brezina, Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems, Computing 56 (1996) 179–196.
- [89] V. Venkatakrishnan, D.J. Mavriplis, Agglomeration multigrid for the three-dimensional Euler equations, Technical Report 94-5, ICASE, NASA Langley, USA, 1994.
- [90] C. Wagner, Introduction to algebraic multigrid, Course Notes of an Algebraic Multigrid Course, University of Heidelberg, 1999.
- [91] W.L. Wan, Scalable and multilevel iterative methods, Ph.D. Thesis, Department of Mathematics, UCLA, Los Angeles, June 1998.
- [92] W.L. Wan, T.F. Chan, B. Smith, An energy-minimizing interpolation for robust multigrid, SIAM J. Sci. Comput. (1998), to appear.
- [93] P. Wesseling, A survey of fourier smoothing analysis results, in: W. Hackbusch, U. Trottenberg (Eds.), Multigrid Methods III, International Series of Numerical Mathematics, Vol. 98, Birkhäuser, Basel, 1991, pp. 105–127.
- [94] P. Wesseling, An Introduction to Multigrid Methods, Wiley, Chichester, 1992.
- [95] G. Wittum, On the robustness of ilu smoothing, SIAM J. Sci. Statist. Comput. 10 (1989) 699-717.
- [96] G. Wittum, An ILU-based smoothing correction scheme, in: W. Hackbusch (Ed.), Parallel Algorithms for PDEs, Proceedings, Kiel, January, 1990, Notes on Numerical Fluid Mechanics, Vol. 31, Vieweg, Braunschweig, 1991.
- [97] G. Wittum, Filternde zerlegungen: schnelle löser für große gleichungssysteme (Filtering decomposition: fast solvers for large systems of equations), Teubner Skripten zur Numerik, Teubner, Stuttgart, 1992.
- [98] J. Xu, Theory of multilevel methods, Ph.D. Thesis, Cornell University, Ithaca, New York, 1989.
- [99] J. Xu, Iterative methods by space decomposition and subspace correction, SIAM Rev. 34 (1992) 581-613.
- [100] J. Xu, The auxiliary space method and optimal multigrid preconditioning techniques for unstructured grid, Computing 56 (1996) 215–235.
- [101] H. Yserentant, On the multilevel splitting of finite element spaces, Numer. Math. 49 (1986) 379-412.
- [102] P.M. De Zeeuw, Matrix-dependent prolongations and restrictions in a blackbox multigrid solver, J. Comput. Appl. Math. 33 (1990) 1–27.



JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

Journal of Computational and Applied Mathematics 123 (2000) 353-369

www.elsevier.nl/locate/cam

# The Rook's pivoting strategy

George Poole<sup>\*</sup>, Larry Neal<sup>1</sup>

Mathematics Department, East Tennessee State University, Johnson City, TN 37614-0663, USA

Received 24 February 1999

#### Abstract

Based on the geometric analysis of Gaussian elimination (GE) found in Neal and Poole (Linear Algebra Appl. 173 (1992) 239–264) and Poole and Neal (Linear Algebra Appl. 149 (1991) 249–272; 162–164 (1992) 309–324), a new pivoting strategy, *Rook's pivoting* (RP), was introduced in Neal and Poole (Linear Algebra Appl. 173 (1992) 239–264) which encourages stability in the back-substitution phase of GE while controlling the growth of round-off error during the sweep-out. In fact, Foster (J. Comput. Appl. Math. 86 (1997) 177–194) has previously shown that RP, as with complete pivoting, cannot have exponential growth error. Empirical evidence presented in Neal and Poole (Linear Algebra Appl. 173 (1992) 239–264) showed that RP produces computed solutions with consistently greater accuracy than partial pivoting. That is, Rook's pivoting is, on average, more accurate than partial pivoting, with comparable costs. Moreover, the overhead to implement Rook's pivoting in a scalar or serial environment is only about three times the overhead to implement partial pivoting. The theoretical proof establishing this fact is presented here, and is empirically confirmed in this paper and supported in Foster (J. Comput. Appl. Math. 86 (1997) 177–194). © 2000 Elsevier Science B.V. All rights reserved.

MSC: 65F05; 65F35

Keywords: Gaussian elimination; Pivoting

## 1. Introduction

The geometric analysis of Gaussian elimination (GE) presented in [3,5,6] suggests that if the pivoting strategy used during the sweep-out phase (SWOP) of GE makes no attempt to control instability during the back-substitution phase (BSP), then the computed solution after back-substitution has been performed may bear little resemblance to the exact solution of the triangular system produced

<sup>\*</sup> Corresponding author.

E-mail address: pooleg@etsu.edu (G. Poole).

<sup>&</sup>lt;sup>1</sup> On October 16, 1995, Larry Neal died after a nine-year, fiercely fought battle with cancer. Without his technical expertise, creative mind, and incomparable friendship, this work could not have been completed. This work was sponsored in part by a grant from the East Tennessee State University Research Development Committee, #2-25417.

by the sweep-out. Rook's pivoting (RP) was introduced in [3] and is designed to simultaneously reduce round-off error during the SWOP and control instability that might arise during the BSP. Very simply, RP searches for pivots that are maximal in absolute value in both the row and column they reside. Partial pivoting (PP) abdicates any power to control BSP instability while complete pivoting (CP) does, in fact, exercise some control (during the SWOP) over the instability that might arise during the BSP. In fact, PP can admit exponential growth error while CP and RP do not [2]. In a minimal sense, RP resembles the pivoting strategy of Bunch and Kaufmann [1] applied to symmetric matrices. Sometimes referred to as "diagonal pivoting", the BK strategy, at each pivot selection step, searches two rows and columns in order to find a "pivoting element" (either of order one or order two) to maintain stability and symmetry throughout the SWOP. However, the similarity of the two strategies abruptly ends here.

Section 2 outlines the philosophy of the RP strategy. Section 3 contains an outline of a formal mathematical proof of the fact that the cost or overhead to implement RP in a scalar or serial environment is the same order of magnitude as the cost to implement PP. The empirical evidence supporting the theory in Section 3 is the subject of Section 4 (and corroborated in [2]). Section 5 contains empirical data to show that computed solutions based on RP are, on average, more nearly accurate than those based on PP. In Section 6, an example promised in an earlier paper [3], is presented to show a worst-case instability during the BSP of GE, in either CP or RP. This example underscores and clarifies the misunderstandings regarding the so-called "no problem phase" (i.e., BSP) of GE. Section 7 contains some conclusions and remarks. Finally, an amplified version with complete details is contained in [4].

## 2. The Rook's pivoting strategy

Suppose  $A = [a_{ii}]$  is a square matrix of order n and Ax = b is a linear system to be solved by Gaussian elimination (GE). As with PP and CP, RP is a strategy based on the magnitude of elements of A. In RP, the kth pivot is selected to have "dominion" (maximal absolute value) over both the row and column in which it lies. As does CP, RP confines its search for the kth pivot to elements in rows k through n and columns k through n of  $A^{(k)}$ , the modified coefficient matrix before the kth step of the SWOP of GE is performed. Also, as with CP, RP selects for a pivot in the (k,k)position of the upper-triangular matrix U an entry whose magnitude is greater than or equal to the magnitudes of all other entries in both its column (to minimize round-off error in the SWOP) and in its row (to foster stability during the BSP). However, unlike CP, it is established theoretically in Section 3 (and confirmed empirically in Section 4 and [2]) that RP rarely requires a complete search of every entry in the unswept sub-matrix of order (n - k + 1) in the lower right corner of  $A^{(k)}$  in order to find a suitable kth pivot (k = 1, 2, ..., n - 1). To locate the kth pivot using RP, one performs a sequential search (column, row, column, etc.) of the remaining unsearched vectors until an element has been located whose magnitude in absolute value is not exceeded by any other element in either the row or column it resides. If r > k, rows r and k are interchanged. If c > k, columns c and k are interchanged. Then the kth step of the sweep-out phase of GE is performed on  $A^{(k)}$  to eliminate nonzero entries below the diagonal in column  $\hat{k}$ , thus producing  $A^{(k+1)}$ ,  $(k=1,2,\ldots,n-1)$ . Consequently, the cost of implementing RP in a serial environment varies between twice the cost of PP and the full cost of CP. In fact, we shall prove that the expected cost of implementing RP in a serial environment is about three times the cost of PP. This is precisely the goal of the next section.

## 3. The cost of implementing Rook's pivoting

No matter what pivoting-for-size strategy is employed, the amount of arithmetic performed to complete the SWOP and BSP of GE is exactly the same. The difference in the costs between strategies lies in the number of compares required to determine the pivots, as well as any other costs that might arise from data access and data management (to be discussed shortly).

When unnecessary duplicate compares are avoided, the total cost in number of compares required to locate all n - 1 pivots using RP varies between the cost of PP and the cost of CP, namely between  $O(n^2/2)$  and  $O(n^3/3)$ . We shall prove that the *expected total number of compares* for RP is  $O(3n^2/2)$  for all  $n \ge 2$ , even if duplicate compares are allowed. This compares favorably with the fixed cost of PP, namely  $O(n^2/2)$  compares. As will be demonstrated in this section and the next, within the confines of a serial computing environment, *rarely* does the cost of RP approach the fixed cost of CP. Foster [2] empirically confirms, under independent testing, that RP is only slightly more costly than PP. Moreover, Foster has shown that RP does not have the exponential growth error that PP sometimes experiences.

Before outlining the proof regarding the expected costs of RP, there is one other issue we must discuss regarding the implementation of RP, namely the costs of data access. Unlike PP and CP in which data can be organized to take advantage of storage and retrieval in some architectures relative to some languages (i.e., by rows in a "C" environment or by columns in a FORTRAN environment), the RP strategy necessitates that one access data by both rows and columns. On the surface, one might question the use of RP for large linear systems since severe page faulting may arise throughout the process of searching for its pivots. The investigation into this issue depends on the size of the linear system and the size of machine memory. However, a careful but simple analysis of the entire process of GE (including data management, the pivot selection strategy, the SWOP and the BSP), one can easily show for a given fixed memory size (pages of memory available), that as the size n of the linear system increases, the percentage of time spent in page faults for the pivot-selection phase against the total time of the GE algorithm decreases towards zero. In other words, if page faulting becomes a problem for pivot selection, it becomes a problem for the entire algorithm, especially in view of the theory presented in this section and the empirical evidence presented in the next section. Besides, cache and register management is more of a concern today for large data sets than page faults. For example, LAPACK involves a block algorithm design to address good cache behavior. Finally, all the numerical experiments found in [2], and in this paper, confirm the fact that RP is only slightly more expensive than PP in a serial environment.

We begin to outline our proof with a theoretical demonstration. Suppose we are attempting to locate the first Rook's pivot, labeled  $p^*$ , in a linear system of order *n*. Here, *V* may denote either a row vector or column vector. Eventually, we wish to (1) compute  $P_i$ , the probabilities that  $p^*$  is in  $V_i$ , the *i*th vector searched (and in no previous vector  $V_j$ , j < i), and (2) compute C(i), the total number of compares required to confirm that  $p^*$  is in the *i*th vector searched (and in no previous vectors) using the RP strategy. To perform these computations, we shall define a *random variable* C whose possible values depend on the size of the matrix and the (variable) number of vectors that must be searched in order to locate a suitable Rook's pivot. To this end, we offer a few preliminary observations and establish some basic notation.

We shall say that a component t of the vector V is maximal in V if it is the first element encountered (top-to-bottom of a column vector or left-to-right of a row vector) whose magnitude is

greater than or equal to the magnitude of every component of V. Also, we shall use L(V) to denote the number of components in V.

First, consider a coefficient matrix A of order n in which the first Rook's pivot,  $p^*$ , happens to be in column number one. This means that  $p^*$  is the first element in a top-to-bottom search of column one whose magnitude is greater than or equal to the magnitude of all other elements of column one, and whose magnitude (by chance) happens to be greater than or equal to the magnitudes of all other elements to its right in the same row. For convenience, it is initially assumed that  $p^*$  is in row rof A. That is,  $a_{r1}$  is maximal in the first column vector of the matrix A.

Let  $V_1 = [a_{11} \ a_{21} \ \cdots \ a_{n1}]^t$  denote the *n*-component vector in *column* one of *A* and let  $V_2 = [a_{r2} \ a_{r3} \ a_{r4} \ \cdots \ a_{rn}]$  denote the (n-1)-component vector in *row r* of *A* which omits the column one component. So  $|p^*|$  is at least as big as the magnitudes of every entry in  $V_1$  and  $V_2$ . We shall describe this situation by saying that the maximal entry  $p^*$  of  $V_1$  is *maximal through*  $V_2$ .

Let  $P_k$  be the probability that a suitable Rook's pivot is located in the *k*th vector searched (and in no previous vectors). Then  $P_1$  is the probability that  $p^*$  is in the first column of the matrix A. That is,  $P_1$  is the *conditional probability* that  $p^*$  is maximal through  $V_2$  given that  $p^*$  is maximal in  $V_1$ . The probability that  $p^*$  is maximal in  $V_1$  is just  $1/L(V_1) = 1/n$ . The probability that  $p^*$  is maximal through  $V_2$  is  $1/[L(V_1) + L(V_2)] = 1/(n + n - 1)$ . So,

$$P_1 = \frac{1/(L(V_1) + L(V_2))}{1/L(V_1)} = \frac{L(V_1)}{L(V_1) + L(V_2)} = \frac{n}{2n - 1}.$$
(3.3)

Moreover, the number of compares required to confirm that  $p^* = a_{r1}$  is the first pivot is given by

$$C(1) = (L(V_1) - 1) + L(V_2) = (n - 1) + (n - 1) = 2(n - 1).$$
(3.4)

Before determining the remaining probabilities,  $P_2$  through  $P_{2n-1}$ , it is important to clarify the assumptions used in computing  $P_1$ , which are also used in computing the remaining probabilities  $P_i$ . In the absence of knowing the initial matrix pattern (for example, symmetric, diagonally dominant, banded, etc.), we shall assume the entries of the coefficient matrix are randomly distributed among the positions of the matrix. Moreover, in the absence of knowing the initial distribution of the entries themselves (uniform, normal, symmetric, skewed, etc.), we shall assume the entries of the coefficient matrix form a uniformly distributed set.

The first assumption is not unreasonable. The more random the distribution of the matrix entries, the more time must be consumed to locate a suitable Rook's pivot. Likewise, the second assumption is not unreasonable because Trefethen and Schreiber [8] have observed that whatever the distribution of the entries in the initial coefficient matrix, the entries in the sequence  $A^{(k)}$  of submatrices determined by PP *tend toward a normal distribution*. We believe the same property holds for RP and will assume this to be the case. With these assumptions we shall proceed with the computation of probabilities.

Now, suppose  $a_{r1}$  is maximal in  $V_1$ , but  $a_{rs}$  (s > 1) is maximal in  $V_2$  so that  $|a_{r1}| < |a_{rs}|$ . Then  $a_{r1}$  is not the first Rook's pivot, and the magnitudes of the n-1 components of the column vector  $V_3 = [a_{1s} \ a_{2s} \ \cdots \ a_{r-1,s} \ a_{r+1,s} \ \cdots \ a_{n,s}]^t$  must be compared to  $|a_{rs}|$  to determine if  $p^* = a_{rs}$ . In what follows, it is important to understand how the sequence of vectors  $\{V_1, V_2, \ldots, V_k\}$  is determined and defined using the RP strategy to locate  $p^*$ .

In searching for  $p^*$  in any matrix A of order n,  $V_1$  is the first vector searched (column one of A),  $V_2$  is the second vector searched (the partial row of A corresponding to the entry maximal in  $V_1$ ,

omitting the component common to  $V_1$ ), and  $V_i$  is the *i*th vector searched, omitting any components in  $V_j$ , for j = 1, ..., i - 1. If i > 1 is odd,  $V_i$  is a partial column vector of A; otherwise,  $V_i$  is a partial row vector. Note that  $L(V_1) = n$ ,  $L(V_2) = L(V_3) = (n - 1)$ , and  $L(V_4) = L(V_5) = n - 2$ , and so forth. Furthermore, it is not difficult to show

$$L(V_i) = n - [i/2]$$
 for  $i = 1, 2, ..., (2n - 1),$  (3.5)

where [i/2] denotes the greatest integer function. Note that  $L(V_i) \ge L(V_{i+1})$  for each  $1 \le i \le 2n - 1$ , and  $L(V_{2n-2}) = L(V_{2n-1}) = n - (n-1) = 1$ . So  $n \ge L(V_i) \ge 1$  for each  $1 \le i \le 2n - 1$ . Also note that  $L(V_{2n}) = 0$ .

Now suppose  $\{V_1, V_2, V_3, ..., V_k, V_{k+1}\}$  is the *exact* sequence of vectors searched to locate a suitable Rook's pivot  $p^*$  where  $k \ge 1$ . If k < (2n - 2), then k represents the smallest integer for which (1)  $p^*$  is maximal in  $V_k$ , (2)  $p^*$  has magnitude greater than or equal to the magnitude of each component in  $V_{k+1}$ , and (3)  $p^*$  has magnitude strictly greater than the magnitude of each component of the vectors in  $\{V_1, V_2, ..., V_{k-1}\}$ . In this case,  $p^*$  is in  $V_k$ , the second to last vector searched and we shall say that  $p^*$  is *maximal up through*  $V_{k+1}$ . In case k = (2n - 2),  $p^*$  could be in either  $V_{2n-2}$ , the second-to-last vector searched or in  $V_{2n-1}$ , the last vector searched (a column vector of length one).

Now  $P_2$  is the probability that  $p^*$  is not in  $V_1$  multiplied by the conditional probability that  $p^*$  is maximal up through  $V_3$  given that  $p^*$  is maximal in  $V_2$ . That is,

$$P_2 = (1 - P_1) \frac{1/(L(V_1) + L(V_2) + L(V_3))}{1/(L(V_1) + L(V_2))} = \frac{n - 1}{3n - 1}.$$
(3.6)

Also, the number of compares required to confirm that  $p^*$  is in  $V_2$  is given by

$$C(2) = (L(V_1) - 1) + L(V_2) + L(V_3) = C(1) + L(V_3) = 3(n - 1).$$
(3.7)

To determine  $P_k$  for k = 3, 4, ..., 2n - 1, we list the following lemmas, corollaries, and theorems, but for the sake of space, proofs are omitted. However, some proofs are rather technical, but easy [4].

**Lemma 3.1.** For all  $n \ge 2$  and for each k = 2, 3, ..., 2n - 1,

$$P_k = \frac{(1 - \sum_{i=1}^{k-1} P_i)(\sum_{i=1}^{k} L(V_i))}{\sum_{i=1}^{k+1} L(V_i)}.$$

**Lemma 3.2.** *For all*  $n \ge 2$  *and for any* k = 2, 3, ..., 2n*,* 

$$1 - \sum_{i=1}^{k-1} P_i = \prod_{j=2}^{k} \left( \frac{L(V_j)}{\sum_{i=1}^{j} L(V_i)} \right).$$

**Corollary 3.3.** For all  $n \ge 2$ ,  $\sum_{i=1}^{2n-1} P_i = 1$ .

**Lemma 3.4.** For any  $n \ge 2$  and for all k = 2, 3, ..., 2n - 1,

$$P_k = \prod_{j=2}^{k-1} \left( \frac{L(V_j)}{\sum_{i=1}^j L(V_i)} \right) \left( \frac{L(V_k)}{\sum_{i=1}^{k+1} L(V_i)} \right).$$

**Lemma 3.5.** For any integer  $j \ge 2$ ,  $j[j/2] - \sum_{i=1}^{j} [i/2] > 0$ .

**Lemma 3.6.** For any  $n \ge 2$  and for all k = 2, 3, ..., 2n - 1,

$$0 < \left(\frac{L(V_j)}{\sum_{i=1}^j L(V_i)}\right) < \frac{1}{j}.$$

**Lemma 3.7.** For any  $n \ge 2$  and for all k = 2, 3, ..., 2n - 1,

$$0 < \left(\frac{L(V_k)}{\sum_{i=1}^{k+1} L(V_i)}\right) < \frac{1}{k+1}.$$

**Theorem 3.8.** *For any*  $n \ge 2$  *and for all* k = 2, 3, ..., 2n - 1*,* 

$$0 < P_k < \frac{k}{(k+1)!}$$

**Proof.** Follows immediately from Lemmas 3.4, 3.6 and 3.7.  $\Box$ 

Theorem 3.8 provides practical and useful upper bounds for  $P_k$  (k = 2, 3, ..., 2n - 1) which are *independent of n*, the size of the square matrix being searched using the Rook's pivot strategy. These bounds show just how unlikely it is that the RP search strategy requires anything near an entire search of the matrix A before an entry is found whose magnitude is maximal in both its row and its column.

Eq. (3.3) provides a lower bound of  $\frac{1}{2}$  for  $P_1$  if  $n \ge 2$ . So a rather *important observation* about PP is that for any position, PP selects a pivot that is maximal in its row as well as in its column with a probability at least  $\frac{1}{2}$ . This is precisely the reason PP is so successful in practice. At least half the time PP makes the correct choice for a pivot.

Corollary 3.3 and the first inequality of Theorem 3.8 show that the  $P_i$ 's, for i = 1, 2, ..., 2n - 1 satisfy the criteria to be a valid probability distribution. So the number of compares required to locate the first Rook's pivot  $p^*$  in a matrix of order n is indeed a random variable C with 2n - 1 integer values (one for each  $P_k$ ). Now C(k) denotes the number of compares required to confirm that the first pivot is found in  $V_k$ , the kth vector searched, and not in any previous vector  $V_i$  for i = 1, 2, ..., k - 1 using the RP strategy. We have seen that C(1) = 2(n - 1) is the minimum value of this random variable (Eq. (3.4)).

**Theorem 3.9.** For any  $n \ge 2$  and for each k = 2, 3, ..., 2n - 1,

$$C(k) = (k+1)(n-1) - \left[\frac{(k-1)^2}{4}\right].$$

**Proof.** Note that  $C(k) = C(k-1) + L(V_{k+1}) = C(k-1) + n - [(k+1)/2]$ . Now use induction on k and Eqs. (3.4) and (3.7).

From Theorem 3.9,

$$C(2n-1) = C(2n-2) = n^2 - 1. \qquad \Box$$
(3.8)

Just before proving the main theorem, by considering the previous lemmas and theorems, notice what is involved to determine the expected number of compares  $E_2$  to locate the first pivot for a system of order n = 2. That is, since 2n - 1 = 3, C(2) = C(3) = 3 and we have

$$E_2 = \sum_{k=1}^{3} P_k C(k) = \left(\frac{2}{3}\right)(2) + \left(\frac{1}{5}\right)(3) + \left(\frac{2}{15}\right)(3) = \frac{7}{3}.$$
(3.9)

We are now prepared to state and prove the main theorem in this paper which establishes that, on average and under the stated assumptions, RP requires approximately 3(n-1) compares to locate the first pivot of a matrix of order *n*. That is, RP is about three times more expensive than PP.

**Theorem 3.10.** The expected number of compares  $E_n$  required to locate the first Rook's pivot  $p^*$  in a linear system of order  $n \ge 2$  is O(3(n-1)).

**Proof.** By Eq. (3.9),  $E_2 = \frac{7}{3} < 3(n-1)$  when n = 2. Now assume that n > 2 and note that  $E_n = \sum_{k=1}^{2n-1} C(k)P_k$ . Consequently,  $E_n$  satisfies the following equalities and inequalities:

$$E_n = \sum_{k=1}^{2n-1} \{ (k+1)(n-1) - [(k-1)^2/4] \} P_k$$
  

$$\leq \sum_{k=1}^{2n-1} (k+1)(n-1)P_k = (n-1)\sum_{k=1}^{2n-1} (k+1)P_k$$
  

$$< (n-1) \left( 2\left(\frac{n}{2n-1}\right) + \sum_{k=2}^{2n-1} \left(\frac{(k+1)k}{(k+1)!}\right) \right)$$
  

$$= (n-1) \left( \left(\frac{2n}{2n-1}\right) + \sum_{k=2}^{2n-1} \left(\frac{1}{(k-1)!}\right) \right)$$
  

$$< (n-1) \left( 1 + \frac{1}{2n-1} + e - 1 \right)$$
  

$$= \left(\frac{n-1}{2n-1}\right) + e(n-1)$$
  

$$< \frac{1}{2} + e(n-1)$$
  

$$< 3(n-1) \text{ for all } n > 2.$$

Therefore,  $O(E_n) \leq O[e(n-1)] \leq O[3(n-1)] = O(n-1)$ .  $\Box$ 

Note that the first inequality above abandons the assumption that unnecessary duplicate compares are avoided. By way of contrast, recall the number of compares required to locate the *first pivot* using PP and CP are, respectively, (n-1) and  $(n^2-1)$ . So, with  $n \ge 2$ , the expected cost of locating the *first pivot* using RP is approximately 3 times the cost of PP, even when unnecessary duplicate

360

Table 1

	Number of compares required to locate all $n-1$ pivots using				
n	$\frac{PP}{\sum_{i=2}^{n} (i-1)}$	$\frac{\text{CP}}{\sum_{i=2}^{n}(i^2-1)}$	$\frac{\text{RP (expected)}}{\sum_{i=1}^{2n-1} P(i)C(i)}$	RP (expected) approximated $\frac{e(n-1)n}{2}$	
2	1	3	2.333	2.7	
5	10	50	25.125	27.2	
10	45	375	117.041	122.3	
50	1225	42 875	3296.626	3329.9	
100	4950	338 250	13 386.490	13 455.5	
$10^{3}$	499 500	333 832 500	1357063.656	1357781.8	
$10^{4}$	4 999 500	$3.33383325  imes 10^{11}$	$1.35893 \times 10^{8}$	$1.3590 \times 10^{8}$	
10 <sup>5</sup>	$4.99995 \times 10^{9}$	$3.3333833325  imes 10^{14}$	$1.35912 \times 10^{10}$	$1.35913  imes 10^{10}$	

Cost in number of compares to locate all n-1 pivots of a system of order n using the PP, CP, and RP strategies

compares are *not* avoided. As we shall observe, the expected cost to locate all (n-1) RP pivots is about 3 times the cost to locate all PP pivots.

The nature of the random variable *C*, that is, the formulas that give the number of compares required to locate a suitable Rook's pivot (Theorem 3.9) and the probabilities associated with these number of compares (Lemma 3.4 and Theorem 3.8) are assumed to be the same for any square sub-matrix of the original square matrix of order *n*. Thus, one need only replace *n* with n - q in Theorems 3.8, 3.9 and Lemma 3.4 to obtain the random variable *C* and its associated probability distribution for the number of compares to locate the (q + 1)th Rook's pivot (q = 1, 2, ..., n - 2).

**Corollary 3.11.** The expected total number of compares  $E_T$  required to locate all (n-1) Rook's pivots is  $O(3n^2/2) = O(n^2)$ .

**Proof.**  $E_T \sim \sum_{i=2}^n 3(i-1) = 3 \sum_{i=2}^n (i-1) = 3 \sum_{k=1}^{n-1} k = 3(n-1)n/2.$ 

By way of comparison, the total number of compares required to locate all (n-1) pivots by PP (Eq. (3.1)) and CP (Eq. (3.2)) are, respectively, (n-1)n/2 and  $(2n^3 + 3n^2 - 5n)/6$ .  $\Box$ 

Table 1 shows the cost in number of compares required to implement partial, Rook's, and complete pivoting on linear systems of order n. To insure accuracy, the third column of Table 1 was obtained using Lemma 3.4 and Theorem 3.9 and extended precision (19–20 digit floating point arithmetic).

Under the stated assumptions about the initial coefficient matrix and the subsequent, sweep-out induced submatrices  $A^{(k)}$ , it is clear that RP and PP are similar in costs and both are orders of magnitude less expensive than CP. Moreover, Section 5 will contain empirical evidence establishing that the accuracy expected from RP is superior to that expected from PP. This is due, in part, to two factors: First, RP cannot have exponential error growth as can PP [2]. Second, unlike PP, the favorably oriented hyperplanes (i.e., the measure of orthogonality between the rows of U in the LU decomposition) produced by RP fosters a high degree of stability during the back-substitution phase (see [3,5,6,9]).

Table 2

Ratio of number of compares using RP to number of compares using PP for randomly generated matrices with entries which represent either a uniform distribution or normal distribution

n	Number of	Uniform	Normal
matrix	systems	dist	dist
order	(k)	$A_n/W_n$	$A_n/W_n$
10	1000	2.719	2.658
15	1000	2.770	2.709
20	1000	2.819	2.753
25	1000	2.872	2.782
50	1000	2.989	2.894
75	1000	3.053	2.966
100	1000	3.107	3.021
125	1000	3.136	3.055
150	1000	3.163	3.096
200	500	3.2184	3.149
300	200	3.2787	3.215
400	150	3.3146	3.273
500	100	3.3432	3.305

## 4. Empirical evidence on cost of Rook's pivoting

This section contains empirical data supporting the conclusions of Theorem 3.10 regarding the cost of locating the first Rook's pivot. Moreover, we compare the cost of locating all of the Rook's pivots to the cost of locating all the pivots by PP and CP.

The power of MATLAB [11] and specially designed M-files were exploited to generate, massage, and provide appropriate graphs to empirically compare the cost of RP to PP. For certain values k and n, k coefficient matrices of order n were generated whose entries were *uniform* over the interval  $[-10^4, 10^4]$ , and *randomly distributed* by position. The actual number of compares required to implement the RP strategy was calculated for each system and the average  $A_n$  was determined over the k generated systems of order n. Then, to compare RP to PP, the quotient  $A_n/[n(n-1)/2] = A_n/W_n$  was computed for each size n. We repeated this experiment using a normal distribution of the entries in the coefficient matrix which were *randomly distributed* across the positions of the matrix. The results are provided in Table 2.

In addition, for each matrix, the number of vectors searched required to locate each of the n-1 Rook's pivots was saved. From this information, the number of compares required to locate each of the pivots could be computed. Then, over the k generated matrices, the average number of compares was computed. The pivot which generated the largest average over the k matrices was also determined. From the data generated for each triple (n = size, type = uniform or normal, k = How many matrices), a sequence of graphs was produced. The purpose of studying these sequences of graphs was to observe the behavior of RP as the size of the coefficient matrix increased, for either the uniform case or the normal case. Our analysis included systems of orders up to 1000 and was much more extensive than the results presented here. However, we will *limit our presentation* to matrices of order n = 200 and provide a sample of the results. Figs. 1a and b represent data from


Fig. 1. (a) Uniform. (b) Normal.

500 uniform and 500 normal systems, each of order 500. Additional data and graphs are contained in [4].

There are three important observations regarding Figs. 1a and b. *First*, since the theory presented in Section 3 is based on uniform distributions and all 500 coefficient matrices were designed to have such an *initial* distribution, it is not surprising that the ratio of averages for the *first pivot* is approximately Napier's e, as theorized and expected. In fact, by running a simulation over 600 matrices of orders 10–300, in which each submatrix throughout the sweep-out phase was designed to have a normal distribution, the average number of compares for RP was 2.7157 times the cost of



Fig. 2.

PP. By running an identical simulation for matrices with normal distributions, the factor was 2.7105. The graphs of this simulation are presented in Fig. 2. Second, since an initial uniform distribution challenges the RP strategy more than an initial normal one, the pivot requiring the largest number of compares, on average, is realized earlier in the uniform case than the normal one. *Third*, as noted in [8], the distribution of the elements in  $A^{(k)}$  tends toward normal throughout the SWOP. This fact explains why the two graphs in Figs. 1a and b resemble each other as the SWOP progresses. Namely, the graphs tend toward the same number of vectors searched, or the same ratio of compares (Rook's to partial). Fourth, even though  $A^{(k)}$  tends towards a normal distribution as k increases, "normal" does not have much meaning when n < 6 and the number of elements in the coefficient matrix is less than 30. This explains why the last portion of these two graphs drops off accordingly at the end. That is, when the SWOP of Gaussian elimination has reduced the search for the Rook's pivot from a given matrix A of order n to a matrix  $A^{(k)}$  of order less than 6, the number of compares required to locate the Rook's pivots is significantly less than e.

### 5. Empirical evidence on accuracy of Rook's pivoting

The geometric analysis of both phases of GE presented in [3,5] clearly indicates that a pivotingfor-size strategy which selects as pivots those entries whose magnitudes are maximal in *both* their column and their row will, on average, produce more nearly accurate results than PP. CP is such a strategy, but it is cost prohibitive. As demonstrated in the previous section, RP is such a strategy whose *magnitude of cost* is the same as PP, far less than the cost for CP. In this section we compare the average total error between solutions of linear systems computed by PP, MATLAB's  $A \setminus b$  Routine (ML), RP, and CP. These linear systems were designed to provide a range of examples with the following properties: (1) All entries and solutions are exactly machine representable, (2) all systems are solved in IEEE double-precision, (3) the entries of each matrix are either normally or uniformly distributed among themselves and the magnitudes of the entries in the coefficient matrices lie in the interval  $[10^{-4}, 10^4]$ , (4) the collection of systems and their condition numbers cover the spectrum from  $10^6$  to  $10^{18}$ , (5) systems of order 25–500 were considered in the analysis. However, we only discuss and present the graphs for the case n = 100. The results are comparable for systems of other orders [4].

The method by which these linear systems were crafted is explained in Section 3 of Neal and Poole [3]. Briefly, thousands of integer linear systems with integer solutions were randomly generated and then modified by the technique of Rice [7] to assume condition numbers within some designated interval, usually  $[10^6, 10^{18}]$ . *Note*: The reason for *not* including linear systems with condition numbers inside the interval  $[10^0, 10^6]$  is that in double-precision arithmetic, the errors resulting from any one of the four methods (PP, ML, RP, CP) is insignificant. That is, double-precision is sufficient to minimize the effects of round-off error during the SWOP and to *mask* any instability during the BSP, even for moderately ill-conditioned systems.

To compare the accuracies of the double-precision-computed solutions between the four pivoting strategies, over 6000 linear systems of order 100 were generated whose coefficients represented a Normal distribution, and whose condition numbers spanned the interval [10<sup>6</sup>, 10<sup>18</sup>]. Each of these linear systems was solved by each of the four methods: PP, RP, CP and ML. Fourteen pieces of data were stored for each of the linear systems in a 6000 by 14 data file called *datanorm.ill*:  $L_2$ -norm condition number, the seed producing the system, the  $L_2$ -norm error from each of the four methods, L-infinity norm error from each of the four methods, and the back-substitution phase error multipliers (a single vector) for each of the four strategies. To summarize the process of graphically comparing the accuracies between PP, RP, CP and ML, we constructed an M-file, called plotill.m, designed to massage the datanorm.ill file in the following manner: For size = 100, plotill (size, increments, minlog, maxlog) (1) loads the data file *datanorm.ill* and accepts as input k =increments, and  $6 \le \min\log < \max\log \le 18$ , then sets  $w = \max\log - \min\log$ ; (2) partitions the interval [minlog, maxlog + 1/k] into w \* k + 1 subintervals or "bins" of equal width 1/k; (3) scans the array datanorm.ill and locates all systems with condition numbers COND such that log<sub>10</sub>(COND) is contained in the interval [minlog, maxlog + 1/k]; (4) for each bin, the average  $L_2$ -norm error is computed for each of the four pivoting strategies over the systems in the bin. The resulting four plot points are assigned with first component being the average of the condition numbers over the systems in the bin. Any bin not containing a minimum of 30 systems is omitted; (5) plots four polygonal graphs, one for each of PP, RP, ML and CP, with horizontal axis ranging over [10minlog, 10maxlog] and vertical axis ranging over [0, maxerror + ], automatically scaled by MATLAB [11].

The generation of data was repeated for linear systems of order 100 whose coefficients represented a *uniform distribution*. As long as  $(\min\log - \max\log) \ge 2$ , the resulting graphs all had an appearance similar to that represented by the four graphs in Figs. 3–6.

Fig. 3 contains four graphs generated from the data bank *datanorm.ill* (containing 6000 normal systems) when the parameters entered for M-file *plotill* were minlog = 6, maxlog = 9, and increments = 2. The  $L_2$ -norm (rather than the *L*-infinity norm) was selected to measure the error. The code at the top of each figure matches the pivoting strategy to its graph type. The vertical scale is based on the maximum  $L_2$ -norm error. In Fig. 3, the error ranged from *very small* to approximately  $4.5 \times 10^{-8}$ . The text inside the figure indicates that 1668 of the 6000 systems had coefficient matrices with condition numbers in the interval  $[10^6, 10^9]$ .





Fig. 4. Normal.

The parameters and data sets give in Table 3 were used to generate the next three sets of graphs in Figs. 4-6.

The magnitudes of the average  $L_2$ -norm errors in each bin were generally ordered (largest to smallest) as PP $\ge$ ML $\ge$ RP $\ge$ CP. Moreover, RP tracked closely with CP, while MATLAB's method tracked closely with PP. It might be noted here that MATLAB (ML) uses the code that was historically attributed to Fortran's management of data, namely by columns and not rows. On the other hand, our code for PP uses the traditional, pedagogical management of data, namely by rows. It is



Fig. 5. Uniform.



Fig. 6. Uniform.

Table 3				
Figure	Data set	Condition	Increments	No. of sys.
5.2	Datanorm	$[10^8, 10^{10.5}]$	3	1549
5.3	Dataunif	$[10^{12}, 10^{16.5}]$	2	1869
5.4	Dataunif	$[10^{14}, 10^{16}]$	2	834

the difference in the order of arithmetic that explains the difference in computed solutions. However, the explanation of why ML does a better job than PP, on average, is not yet fully understood.

#### 6. Worst-case instability in back substitution

As noted in [3], computing and reviewing the back-substitution phase error multipliers (BSP EMs) is an inexpensive way to check for instability in the back-substitution phase of GE. It is our experience that the magnitudes of all BSP EMs for either CP or RP rarely exceed 10 [3]. However, it can be shown by induction that in the worst case, CP and RP may produce upper-triangular systems for which  $||m||_{\infty} = 2^{n-2}$  where *n* is the dimension of the system [3]. So even CP and RP might produce upper-triangular systems for which the BSP is unstable, a rarity under either strategy. We illustrate this fact with the following example which was promised in [3, p. 254].

**Example 6.1.** Recall from [3, p. 247] that the BSP error multipliers can be obtained by solving the triangular system  $Uy = \varepsilon$  where U is the triangular matrix derived from the sweep-out phase of GE and  $\varepsilon = [0, 0, ..., 0, u_{nn}]^t$ . Consider the upper triangular matrix U of order n for which  $u_{ii} = 1$ , for i = 1 to n and  $u_{ij} = -1$  for  $j \ge i$ . U could be obtained through the sweep-out phase using either CP or RP (or any other strategy which leaves  $|u_{ii}| \ge |u_{ij}|$  for each i). Solving  $Uy = \varepsilon$ , one finds that  $m = [2^{n-2}, 2^{n-3}, ..., 2, 1, 1]^t$  and hence,  $||m||_{\infty} = 2^{n-2}$ . Thus, apart from any round-off error that may occur during the back-substitution phase, if there is any error in the first computed component, no matter how small or seemingly insignificant, the error will be magnified (instability) as it is propagated through the back-substitution process (see example in [5, p. 257]). However, as noted earlier, such examples are very pathological and our experiences indicate that usually  $||m||_{\infty} \le 10$  with RP.

# 7. Conclusions

In addition to the many remarks found in the conclusions of Neal and Poole [3] and Poole and Neal [5,6] three additional practical conclusions are provided here, as well as one philosophical remark.

First, as with CP, RP usually produces an upper-triangular system whose hyperplanes are very well-oriented with respect to their corresponding coordinate axes [3,5,9]. Also, as with CP, RP usually produces BSP EMs whose magnitudes are much smaller than those produced by PP. That is, empirical evidence suggests that except for highly contrived pathological systems, RP and CP usually produce upper-triangular systems whose corresponding hyperplanes are well-oriented with respect to their coordinate axes, but more importantly, they are also well-oriented with respect to each other [3,9]. So, if the sweep-out phase of GE is performed in double-precision using the RP strategy, round-off error during the sweep-out phase is usually well controlled and the back-substitution phase is almost always numerically stable [3,9].

Second, if one insist on using PP with Gaussian elimination to solve a large linear system, one should calculate the back-substitution phase error multipliers [3]. If any are large compared to the precision used for the calculations, then iterative refinement should be used after the back-substitution

phase has been completed, or RP should be used. See Foster [2] for a "partial Rook pivoting" strategy, one threshold in nature.

Three, the important point made in Section 3 is worth repeating. That is, PP is very successful in practice because at least half the time it selects a pivot which controls round-off error during the SWOP while fostering stability during the BSP.

*Philosophical point*: The future of computing technology and management of binary information is unknown. Today's literature contains new creative ways to manage GE that differ significantly from former or even contemporary techniques [10]. Many current philosophies and practices are reflections of old computer architecture, one-dimensional data structures, and deified programming languages. History is very important. But in the area of electronic manipulation of binary information, sharing knowledge and perspectives among those who aspire to compute is more important than controlling research based on yesteryear's beliefs. David Wise [10] says it better than we: "Such a trend (separating ourselves from the knowledge and perspectives of others) is not only scientifically, but politically foolish; we are all colleagues learning how to compute. We must better share problems, solutions, styles, techniques, and philosophy".

# Acknowledgements

We appreciate the care and patience required of the referees who "walked" through this paper without many details available to them, but contained in [4]. Cleve Ashcraft read a version of this paper and suggested several good improvements. Moreover, we are grateful and indebted to the editorial staff of CAM for their support and encouragement, particularly Apostolos Hadjidimos. Finally, we are forever grateful to John R. Rice for his "underwing" encouragement throughout the last decade of the 20th century.

#### References

- [1] J. Bunch, L. Kaufmann, Math. Comput. 31 (137) (1977) 163–179.
- [2] L. Foster, The growth factor and efficiency of Gaussian elimination with rook pivoting, J. Comput. Appl. Math. 86 (1997) 177–194.
- [3] L. Neal, G. Poole, A Geometric analysis of Gaussian elimination, II, Linear Algebra Appl. 173 (1992) 239-264.
- [4] L. Neal, G. Poole, The Rook's pivoting strategy, Technical Report GP112595001, ETSU, 1995.
- [5] G. Poole, L. Neal, A geometric analysis of Gaussian elimination, I, Linear Algebra Appl. 149 (1991) 249-272.
- [6] G. Poole, L. Neal, Gaussian elimination: when is scaling beneficial?, Linear Algebra Appl. 162–164 (1992) 309–324.
- [7] J.R. Rice, Numerical Methods, Software, and Analysis, McGraw-Hill, New York, 1983.
- [8] L.N. Trefethen, R.S. Schreiber, Average-case stability of Gaussian elimination, SIAM J. Matrix Anal. Appl. 11 (1990) 335–360.
- [9] E. van der Biest, A study of different pivoting strategies and error reducing algorithms for Gaussian elimination, M.S. Thesis, East Tennessee State University, 1991.
- [10] D. Wise, Undulant-block elimination and integer-preserving matrix inversion, Sci. Comput. Programming 33 (1999) 29–85.
- [11] MATLAB, A software product of Math Works Inc.

# Further reading

- 1. L. Eldén, L. Wittmeyer-Koch, Numerical Analysis, An Introduction, Academic Press, San Diego, CA, 1990.
- 2. G. Forsythe, M. Malcolm, C.B. Moler, Computer Methods for Mathematical Computations, Prentice-Hall, Englewood Cliffs, NJ, 1977.
- 3. G.W. Stewart, Introduction to Matrix Computations, Academic Press, New York, 1973.



JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

Journal of Computational and Applied Mathematics 123 (2000) 371-394

www.elsevier.nl/locate/cam

# Numerical methods in control

Volker Mehrmann<sup>a, \*, 1</sup>, Hongguo Xu<sup>b, 2</sup>

<sup>a</sup> Fakultät für Mathematik, TU Chemnitz, D-09107 Chemnitz, Germany <sup>b</sup>Department of Mathematics, Case Western Reserve University, 10900 Euclid Ave., Cleveland, OH 44106, USA

Received 21 June 1999; received in revised form 30 October 1999

#### Abstract

We study classical control problems like pole assignment, stabilization, linear quadratic control and  $H_{\infty}$  control from a numerical analysis point of view. We present several examples that show the difficulties with classical approaches and suggest reformulations of the problems in a more general framework. We also discuss some new algorithmic approaches. © 2000 Elsevier Science B.V. All rights reserved.

MSC: 65F15; 93B40; 93B36; 93C60

*Keywords:* Pole placement; Linear quadratic control; Stabilization  $H_{\infty}$  control; Algebraic Riccati equation; Hamiltonian matrix; Skew Hamiltonian matrix; Two-point boundary-value problem

# 1. Introduction

In the last 40 years systems and control theory has evolved into a mature field that has found a stable position on the borderline between applied mathematics, engineering and computer science. The major success is not only due to the fact that beautiful mathematical theories (like linear algebra, ring theory, representation theory and others) find direct application but also since the results have immediately found their ways into production code software packages like MATLAB toolboxes [54,55] or the SLICOT subroutine library [13], which can be and are directly used by engineers working in practice. In this paper we will discuss several problems of linear control theory, as there are *pole assignment, stabilization, linear quadratic control and*  $H_{\infty}$  *control.* In

<sup>\*</sup> Corresponding author.

E-mail address: volker.mehrmann@mathematik.tu-chemnitz.de (V. Mehrmann).

<sup>&</sup>lt;sup>1</sup>Supported by *Deutsche Forschungsgemeinschaft*, within Sonderforschungsbereich SFB393, 'Numerische Simulation auf massiv parallelen Rechnern'.

<sup>&</sup>lt;sup>2</sup> Supported by *Deutsche Forschungsgemeinschaft*, within Sonderforschungsbereich SFB393, 'Numerische Simulation auf massiv parallelen Rechnern'.

the solution techniques for these problems important developments have taken place in recent years, which have lead to changes in viewpoints in particular what the numerical solution of these problems is concerned. In our opinion there are three central questions that need to be studied in more detail in the context of numerical methods for the solution of control problems and it is the aim of this paper to initiate more research and software developments in this direction.

First of all, as is well known, different mathematically equivalent formulations of the same problem may lead to drastically different sensitivity of the problem to perturbations (such as round-off errors) and thus it is important to find the best formulation for numerical solution.

The second issue is that the numerical methods should reflect the physical properties of the problem in the maximal way, to get higher efficiency but also to guarantee even in finite arithmetic that the computed results are physically meaningful.

The third important topic is that with the growing complexity of problems, in particular in the context of large-scale control problems, solution approaches and numerical methods have to be reviewed and completely new methods have to be developed.

We will only discuss the first two issues but large-scale control problems are currently a very important research topic.

Consider linear constant coefficient dynamical systems of the form

$$\dot{x} = Ax + Bu, \qquad x(t_0) = x^0, \tag{1}$$

where  $x(t) \in \mathbb{R}^n$  is the state,  $x^0$  is an initial vector,  $u(t) \in \mathbb{R}^m$  is the control input of the system and the matrices  $A \in \mathbb{R}^{n,n}$ ,  $B \in \mathbb{R}^{n,m}$  are constant. The topics that we discuss here also apply in a similar fashion to problems with output and also to complex problems, but for the sake of brevity we only discuss real problems.

The classical pole placement problem is to find a state feedback control law

$$u = Kx, \tag{2}$$

such that the closed-loop system

$$\dot{x} = (A + BK)x\tag{3}$$

has desired poles, or in linear algebra terminology, that the spectrum of the closed-loop system matrix A + BF is a given set of complex numbers. Here, the case of *stabilization*, where the closed-loop poles are desired to be in the open left-half plane represents an important special case.

For a discussion of the classical theory of the pole placement problem and related problems, we refer the reader to monographs in linear control theory, e.g. [7,27,41,44,50,65,85]. In Section 2 we discuss some new perturbation results and the resulting consequences for numerical methods. These results indicate that the numerical solution of the classical formulation of the pole placement problem is often and in particular for large n and small m a highly ill-conditioned problem that should be modified.

This analysis and the resulting conclusions hold also for the stabilization problem which alternatively may be solved also via the solution of a *linear quadratic control problem*. For this the objective is to find a control law u(t) such that the closed-loop system is asymptotically stable and such that the performance criterion

$$\mathscr{S}(x,u) = \int_{t_0}^{\infty} \begin{bmatrix} x(t) \\ u(t) \end{bmatrix}^1 \begin{bmatrix} Q & L \\ L^T & R \end{bmatrix} \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} dt,$$
(4)

is minimized, where  $Q = Q^{T} \in \mathbb{R}^{n,n}$ ,  $R = R^{T} \in \mathbb{R}^{m,m}$  is positive definite and  $\begin{bmatrix} Q & L \\ L^{T} & R \end{bmatrix}$  is positive semidefinite.

The basics for this problem can be found in classical monographs on linear control [4,7,16,27,51,41, 44,50,58,65,73,85].

Application of the maximum principle [58,69] leads to the problem of finding a stable solution to the two-point boundary value problem of Euler–Lagrange equations

$$\mathscr{E}_{c}\begin{bmatrix}\dot{x}\\\dot{\mu}\\\dot{u}\end{bmatrix} = \mathscr{A}_{c}\begin{bmatrix}x\\\mu\\u\end{bmatrix}, \qquad x(t_{0}) = x^{0}, \qquad \lim_{t \to \infty} \mu(t) = 0$$
(5)

with the matrix pencil

$$\alpha \mathscr{E}_{c} - \beta \mathscr{A}_{c} := \alpha \begin{bmatrix} I & 0 & 0 \\ 0 & -I & 0 \\ 0 & 0 & 0 \end{bmatrix} - \beta \begin{bmatrix} A & 0 & B \\ Q & A^{T} & L \\ L^{T} & B^{T} & R \end{bmatrix}.$$
(6)

If R is well conditioned with respect to inversion, then (5) may be reduced to the two-point boundary-value problem

$$\begin{bmatrix} \dot{x} \\ -\dot{\mu} \end{bmatrix} = \mathscr{H} \begin{bmatrix} x \\ -\mu \end{bmatrix}, \qquad x(t_0) = x^0, \qquad \lim_{t \to \infty} \mu(t) = 0$$
<sup>(7)</sup>

with the Hamiltonian matrix

$$\mathscr{H} = \begin{bmatrix} F & G \\ H & -F^{\mathsf{T}} \end{bmatrix} := \begin{bmatrix} A - BR^{-1}L^{\mathsf{T}} & BR^{-1}B^{\mathsf{T}} \\ Q - LR^{-1}L^{\mathsf{T}} & -(A - BR^{-1}L^{\mathsf{T}})^{\mathsf{T}} \end{bmatrix}.$$
(8)

The solution of the boundary value problems (5) and(7) can be obtained in many different ways. The classical way, that is implemented in most design packages is to determine first X, the positive semidefinite (stabilizing) solution of the associated algebraic Riccati equation

$$0 = H + XF + F^{\dagger}X - XGX \tag{9}$$

and then obtaining the optimal stabilizing feedback as

$$u(t) = -R^{-1}B^{\mathrm{T}}Xx(t).$$
(10)

The solution of the algebraic Riccati equation is also often used for the decoupling of the forward and backward integration. But one may also directly solve the two-point boundary value problem (5) or alternatively (7) without going via the Riccati equation and we will show in Section 3 that this is actually numerically a much better approach and that the Riccati equation presents an unnecessary and sometimes dangerous detour.

As we have already mentioned, we may use both linear quadratic control and pole placement for the objective of stabilization. In Section 4, we compare pole assignment and the solution of linear quadratic control problems for stabilization.

The third problem that we include into our discussion is the standard  $H_{\infty}$  control problem which arises in the context of robust control in frequency domain, see, e.g., the recent monographs [33,87].

In this problem one studies the linear system

$$\dot{x} = Ax + B_1 u + B_2 w, \quad x(t_0) = x^0,$$
  

$$z = C_1 x + D_{11} u + D_{12} w,$$
  

$$y = C_2 x + D_{21} u + D_{22} w,$$
  
(11)

where  $A \in \mathbb{R}^{n,n}$ ,  $B_k \in \mathbb{R}^{n,m_k}$ ,  $C_k \in \mathbb{R}^{p_k,n}$  for k = 1, 2, and  $D_{ij} \in \mathbb{R}^{p_i,m_j}$  for i, j = 1, 2. Here  $w(t) \in \mathbb{R}^{m_2}$  describes noise, modeling errors or an unknown part of the system,  $y(t) \in \mathbb{R}^{p_2}$  describes measured outputs while  $z \in \mathbb{R}^{p_1}$  describes the regulated outputs. The objective of optimal  $H_\infty$  control is to find a control law

$$\dot{q} = \hat{A}q + \hat{B}y,$$

$$u = \hat{C}q + \hat{D}y,$$
(12)

to minimize the closed-loop transfer function  $T_{zw}$  from w to z in  $H_{\infty}$  norm.

Under some technical assumptions, see [87] or [30] for the general case, for a given parameter  $\gamma > 0$ , a necessary and sufficient condition for the existence of an admissible controller such that  $||T_{zw}||_{\infty} < \gamma$ , is that the following conditions hold (e.g. [87, Theorem 16.4, p. 419]):

(A1) For the matrix

$$\mathscr{H}_{\infty} := \begin{bmatrix} A & \gamma^{-2}B_1B_1^{\mathrm{T}} - B_2B_2^{\mathrm{T}} \\ -C_1^{\mathrm{T}}C_1 & -A^{\mathrm{T}} \end{bmatrix},\tag{13}$$

there exists matrices  $Q_1, Q_2 \in \mathbb{R}^{n,n}$  such that

$$\mathscr{H}_{\infty} \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} = \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} T_x, \tag{14}$$

where  $T_x$  has only eigenvalues with nonpositive real parts,  $Q_1$  is nonsingular, and  $X_{\infty} := Q_2 Q_1^{-1}$  is symmetric positive semidefinite.

(A2) For the matrix

$$\mathscr{J}_{\infty} := \begin{bmatrix} A & -B_1 B_1^{\mathsf{T}} \\ \gamma^{-2} C_1^{\mathsf{T}} C_1 - C_2^{\mathsf{T}} C_2 & -A^{\mathsf{T}} \end{bmatrix},\tag{15}$$

there exist matrices  $U_1, U_2 \in \mathbb{R}^{n,n}$  such that

$$\begin{bmatrix} U_1 \\ U_2 \end{bmatrix}^{\mathrm{T}} \mathscr{J}_{\infty} = T_y \begin{bmatrix} U_1 \\ U_2 \end{bmatrix}^{\mathrm{T}}, \tag{16}$$

where  $T_y$  has only eigenvalues with nonpositive real parts,  $U_1$  is nonsingular, and  $Y_{\infty} := U_2 U_1^{-1}$  is symmetric positive semidefinite.

(A3) For the matrices  $X_{\infty}, Y_{\infty}$  we have that  $\gamma^2 > \rho(X_{\infty}Y_{\infty})$ , where  $\rho(A)$  denotes the spectral radius of the matrix A.

The optimal  $_{\infty}$  control is then obtained by finding the smallest admissable  $\gamma$  so that conditions (A1)–(A3) still hold. The optimal controller yields system (12) with

$$\hat{A} := A + \gamma^{-2} B_1 B_1^{\mathrm{T}} X_{\infty} + B_2 \hat{C} - \hat{B} C_2, 
\hat{B} := (I - \gamma^{-2} Y_{\infty} X_{\infty})^{-1} Y_{\infty} C_2^{\mathrm{T}}, \quad \hat{C} := - B_2^{\mathrm{T}} X_{\infty}, \quad \hat{D} := 0.$$
(17)

We see that for conditions (A1) and (A2), we have Hamiltonian matrices which (except for the indefiniteness of blocks) are similar to the Hamiltonians arising in the linear quadratic control problem, and hence the analysis and improvements for the linear quadratic control problem also hold for the  $H_{\infty}$  problem. We discuss this topic in Section 6.

Before going into details, let us recall that we have the following objectives in mind. We want to determine the best formulation of the problem for the use in numerical solution methods and furthermore we wish to obtain methods that are best adapted to all the underlying physical and mathematical structures in order to obtain efficient and accurate solution methods.

# 2. Pole placement

As we have discussed in the introduction, in linear algebra terminology the pole placement problem is as follows:

**Problem 1.** For given matrices  $A \in \mathbb{R}^{n,n}$ ,  $B \in \mathbb{R}^{n,m}$  and a given set of n complex numbers  $\mathscr{P} = \{\lambda_1, \ldots, \lambda_n\} \subset \mathbb{C}$ , that is closed under conjugation, find a matrix  $K \in \mathbb{R}^{m,n}$ , such that the set of eigenvalues of A + BK is equal to  $\mathscr{P}$ .

It is well known, see e.g. [41,84], that a *feedback gain* matrix K exists for all possible sets  $\mathscr{P} \subset \mathbb{C}$ , that are closed under conjugation if and only if (A, B) is *controllable*, i.e.,

$$\operatorname{rank}[A - \lambda I_n, B] = n, \quad \forall \lambda \in \mathbb{C}.$$
(18)

There is a large literature on this problem. Extensions of Ackermann's explicit formula [1] for the single-input case were given in [60,78] and also many numerical algorithms were developed for this problem, see [42,63,66,72,82]. For some of these methods, numerical backward stability has been established, see e.g. [6,25,26,42,63,66]. However, it is nevertheless often observed that the numerical results are very inaccurate. If a numerically stable method yields highly inaccurate results then this is due to ill-conditioning of the problem. Therefore the conditioning of the pole placement problem was analyzed but the conclusions from the analysis are quite different, see [5,35,45,47], and there are several reasons for these differences.

First of all pole assignment is usually approached via a two-step procedure, which first brings the pair (A, B) to a simpler form and then assigns the poles in this simpler form. But in such a two-step procedure it may sometimes happen that although the original problem was well conditioned (i.e., small perturbations in the data only lead to small changes in the solution) one of the intermediate steps is very ill-conditioned. To avoid this problem a good method for the initial reduction has to be used. For the pole assignment problem the best reduction is given by the staircase form of Van Dooren [79] or variations of it, see [46], which essentially does not affect the perturbations except

for situations where the problem is very near to an uncontrollable problem, i.e., a problem (A,B) for which the *distance to uncontrollability* defined as

$$d_u(A,B) := \min_{\lambda \in \mathscr{C}} \sigma_n[A - \lambda I, B], \tag{19}$$

see [29], is small. Here  $\sigma_n(A)$  is the smallest singular value of the matrix A. Since controllability is the necessary and sufficient condition for solvability of the pole placement problem, it is clear that a problem that is near to an uncontrollable problem will be very sensitive to perturbations. Hence the distance to uncontrollability (if small) is an important factor in the perturbation analysis of the pole placement problem but, as we will see below, other factors are equally or even more important.

The second reason for confusion in the evaluation of the pole placement problem is that one has to define clearly what the *solution* of the problem is. This could be the feedback K, the closed-loop matrix A + BK or its spectrum, respectively. All of these are solutions of the pole placement problem but they exhibit largely different perturbation results. A striking example of a stabilization problem is the case m = 1 in Example 1 below, see also [59], which shows that even though the feedback K is computed analytically, and the distance to uncontrollability is large, the (presumingly) stabilized closed-loop system has eigenvalues with positive real part, something which could be a disaster in a practical application.

In our opinion the most important goal of pole placement is that the poles of the closed-loop system obtained with the computed feedback are close to the desired ones and in the case of stabilization the resulting closed-loop system is robustly stable. If the desired poles of the exact closed-loop system are very sensitive to perturbations then this ultimate goal usually cannot be guaranteed. And this may happen even if the computation of K is reliable or even exact.

With this goal in mind, a new analysis and new explicit solution formulas that cover all the aspects of the problem have recently been given in [59,60] and we will interpret some of these results here. The major conclusions can be obtained from the following result which generalizes a perturbation result of [76]. For this result we need the scaled spectral condition number of a matrix A given by  $||TD|| ||(TD)^{-1}||$ , where T is the matrix that transforms A to Jordan canonical form and D is a diagonal matrix that scales the columns of T to have all unit norm, see [28].

**Theorem 1** (Mehrmann and Xu [61]). Consider a controllable matrix pair (A, B), and a set of poles  $\mathscr{P} = \{\lambda_1, \ldots, \lambda_n\}$ . Consider a perturbed system  $(\hat{A}, \hat{B})$  which is also controllable and a perturbed set of poles  $\mathscr{P} = \{\hat{\lambda}_1, \ldots, \hat{\lambda}_n\}$ . Set  $\hat{A} - A =: \delta A$ ,  $\hat{B} - B =: \delta B$  and  $\hat{\lambda}_k - \lambda_k =: \delta \lambda_k, =1, \ldots, n$ . Suppose that both the pole placement problems with A, B,  $\mathscr{P}$  and  $\hat{A}$ ,  $\hat{B}$ ,  $\hat{\mathscr{P}}$  have solutions with a diagonalizable closed-loop matrix. Set

$$\varepsilon := || \left[ \delta A, \delta B \right] || \tag{20}$$

and suppose that

$$\max_{i} \frac{\varepsilon + |\delta\lambda_i|}{\sigma_n([A - \lambda_i I, B])} < \frac{3}{4}.$$
(21)

Then there exists a feedback gain  $\hat{K} := K + \delta K$  of  $(\hat{A}, \hat{B})$  such that

$$||\delta K|| < \frac{5\sqrt{n}}{4} \kappa \sqrt{1+||\hat{K}||^2} \max_i \left\{ \frac{\sqrt{1+(||B^{\dagger}(A-\lambda_i I)||)^2}(\varepsilon+|\delta\lambda_i|)}{\sigma_n([A-\lambda_i I, B])} \right\},\tag{22}$$

the spectrum of  $(\hat{A} + \hat{B}\hat{K})$  is  $\hat{\mathscr{P}}$  and  $\hat{A} + \hat{B}\hat{K}$  is diagonalizable.

Moreover, for each eigenvalue  $\mu_i$  of the closed-loop matrix  $A + B\hat{K}$ , (i.e., the perturbed feedback is used for the unperturbed system), there is a corresponding  $\lambda_i \in \mathcal{P}$  such that

$$|\mu_i - \lambda_i| < |\delta\lambda_i| + \varepsilon \hat{\kappa} \sqrt{1 + ||\hat{K}||^2}.$$
(23)

Here  $\kappa$ ,  $\hat{\kappa}$  are the scaled spectral condition numbers of A + BK and  $\hat{A} + \hat{B}\hat{K}$ , respectively, and  $B^{\dagger}$  is the Moore–Penrose pseudoinverse of B.

Note that under additional mild assumptions in the bounds (22) and (23) the terms  $\hat{\kappa}, \hat{K}$  can be replaced by  $\kappa$  and K, respectively. If this is not possible, then the problem is extremely ill-conditioned and hence not suitable for numerical computation anyway.

Theorem 1 only gives upper bounds for the perturbations. This is the usual situation in most perturbation results. But these bounds are usually quite tight and very well describe the major difficulties of the pole placement problem. Consider the following numerical example from [59]. For this and all the other numerical examples the results were obtained on an HP-700 workstation with machine precision eps =  $2.22 \times 10^{-16}$ , under MATLAB Version 5.2.

**Example 1.** Let  $A = \text{diag}(1, \dots, 20)$ ,  $\mathscr{P} = \{-1, \dots, -20\}$  and let *B* be formed from the first *m* columns of a random  $20 \times 20$  orthogonal matrix.

The MATLAB pole placement code *place* of the *control system toolbox* Version 4.1, which is an implementation of the method given in [42], was used to compute the feedback gain K. We ran m from 1 to 20 and in each case we computed 20 times with 20 random orthogonal matrices B. In Table 1 we list the geometric means (over the 20 experiments) of  $\hat{\kappa}$ ,  $\hat{K}$ , bound= $eps||[A,B]||\hat{\kappa}\sqrt{1+||\hat{K}||^2}$ , and err =  $\max_{1 \le i \le 20} |\mu_i - \lambda_i|$ , with  $\lambda_i$  and the real parts of  $\mu_i$  arranged in increasing order.

It should be noted that for all 400 tests the values of  $\min_i \sigma_n([A - \lambda_i I, B])$  varied from 2.0 to 2.24, so the factor in the denominator of (22) is negligible. Furthermore, we computed in all cases the distance to uncontrollability and found that the pair (A, B) was controllable with a large distance to uncontrollability. Nevertheless for m = 1 the method produced an error message "Can't place eigenvalues there" and for m = 2, 3 a warning "Pole locations are more than 10% in error" was displayed. The reason for this failure of the method is probably due to the large norm of K and the large closed-loop condition number which is computed in the course of the algorithm. Other pole placement algorithms have similar difficulties for small m, see [59,60].

The results of Example 1 and most other examples with n-m large lead to the interpretation that the sensitivity (conditioning) of all possible results of the pole placement problem, i.e., the feedback gain K as well as the poles of the closed-loop system  $A + B\hat{K}$  obtained with the perturbed feedback  $\hat{K}$ , depends heavily on the size of n-m as well as on the factor

$$\mathscr{S} := \kappa \sqrt{1 + ||K||^2} \tag{24}$$

т	ĥ	Ŕ	Bound	Err
1				
2	$1.1 \times 10^{9}$	$2.5  imes 10^6$	$1.2  imes 10^1$	$2.0  imes 10^1$
3	$4.6  imes 10^8$	$1.3  imes 10^6$	2.6	$1.2  imes 10^1$
4	$9.6  imes 10^6$	$2.3  imes 10^5$	$9.6  imes 10^{-3}$	$1.2 \times 10^{-3}$
5	$3.0  imes 10^5$	$3.4  imes 10^4$	$4.6 \times 10^{-5}$	$1.6  imes 10^{-6}$
6	$3.0  imes 10^4$	$1.0  imes 10^4$	$1.3  imes 10^{-6}$	$3.1  imes 10^{-8}$
7	$5.6 \times 10^{3}$	$4.2 \times 10^{3}$	$1.0 \times 10^{-7}$	$1.3  imes 10^{-9}$
8	$1.6 \times 10^{3}$	$2.1 \times 10^{3}$	$1.5  imes 10^{-8}$	$1.3  imes 10^{-10}$
9	$5.3  imes 10^2$	$1.1 \times 10^{3}$	$2.6 \times 10^{-9}$	$1.9  imes 10^{-11}$
10	$2.7  imes 10^2$	$8.9  imes 10^2$	$1.1 \times 10^{-9}$	$6.3 \times 10^{-12}$
11	$1.2  imes 10^2$	$5.2 \times 10^{2}$	$2.7 imes10^{-10}$	$1.8  imes 10^{-12}$
12	$7.6  imes 10^1$	$4.0  imes 10^2$	$1.4  imes 10^{-10}$	$8.3  imes 10^{-13}$
13	$4.4  imes 10^1$	$2.7  imes 10^2$	$5.3  imes 10^{-11}$	$3.6  imes 10^{-13}$
14	$3.0  imes 10^1$	$1.9  imes 10^2$	$2.6  imes 10^{-11}$	$2.0  imes 10^{-13}$
15	$2.4  imes 10^1$	$1.6  imes 10^2$	$1.7  imes 10^{-11}$	$1.5  imes 10^{-13}$
16	$1.9  imes 10^1$	$1.3 \times 10^2$	$1.1  imes 10^{-11}$	$9.5  imes 10^{-14}$
17	$1.5  imes 10^1$	$1.2 \times 10^2$	$7.8 imes10^{-12}$	$6.9  imes 10^{-14}$
18	$1.3  imes 10^1$	$1.1 \times 10^{2}$	$6.8  imes 10^{-12}$	$6.6  imes 10^{-14}$
19	9.0	$8.8  imes 10^1$	$3.5  imes 10^{-12}$	$4.5  imes 10^{-14}$
20	1.0	$4.0  imes 10^1$	$1.8 \times 10^{-13}$	$3.2  imes 10^{-14}$

Table 1 Results for Example 1

378

even if the distance to uncontrollability is large. The additional factor  $d:=1/\min_i \sigma_n[A - \lambda_i I, B]$  in the perturbation bound only plays a role if the distance to uncontrollability is small. It is obvious that if  $d_u(A, B)$  is small then d may be very large and the problem to compute K is definitely ill-conditioned. If, however,  $d_u(A, B)$  is large, then clearly d is small and may be neglected.

The factor  $\mathscr{S}$  has been analyzed in detail in [59,60], where it was observed that in the single-input case  $\mathscr{S}$  is essentially given by the condition number of the Cauchy matrix  $C = [1/(v_i - \lambda_j)]$ , where the  $v_i$  are the eigenvalues of A and the  $\lambda_i$  are the desired poles. This condition number is very large if n is large. In the multi-input case  $\mathscr{S}$  is essentially given by the condition number of a Vandermonde-like matrix which is usually also very ill-conditioned (see [38, Chapter 21] and the references therein), in particular if n - m is large.

This analysis indicates that serious numerical difficulties may arise in the pole placement problem if n - m is large. Furthermore the analysis demonstrates that the currently used strategies to resolve the freedom in K in the numerical method, which is to minimize ||K||, see [15,43,63,66,72,82] or  $\kappa$  as in [42], may both not be sufficient to get good results. A better choice would be to minimize  $\mathscr{S}:=\kappa\sqrt{1+||K||^2}$ , since this factor describes the perturbation very well. A similar strategy has been proposed and implemented by Varga [83]. We can actually formulate this strategy as a refined *robust pole placement problem*.

**Problem 2.** For given matrices  $A \in \mathbb{R}^{n,n}$ ,  $B \in \mathbb{R}^{n,m}$  and a given set of n complex numbers  $\mathscr{P} = \{\lambda_1, \ldots, \lambda_n\} \subset \mathbb{C}$ , (closed under conjugation), find a matrix  $K \in \mathbb{R}^{m,n}$ , such that the set of eigenvalues of A + BK is equal to  $\mathscr{P}$ , and that minimizes  $\mathscr{S} := \kappa \sqrt{1 + ||K||^2}$ .

A solution to this problem for small systems can actually be obtained via standard optimization software by using the explicit formula for K given in [60]. In practice one probably does not even need the global minimum, but just one, where  $\mathscr{S}$  is small enough to guarantee small bounds (22) and (23), which then can be actually computed and used as condition estimator.

But we propose to go even further in the reformulation of the pole placement problem, see also [35]. One should first ask the following question.

Does one really have a fixed set of poles or does one rather have a specific region in the complex plane where one wants the closed-loop poles to be?

If the latter is the case then not only the minimization over the freedom in K but also a minimization over the position of the poles in the given set should be used. This would lead to the *optimized robust pole placement problem*:

**Problem 3.** For given matrices  $A \in \mathbb{R}^{n,n}$ ,  $B \in \mathbb{R}^{n,m}$  and a given set  $\mathscr{P} \subset \mathbb{C}$ , find a matrix  $K \in \mathbb{R}^{m,n}$ , such that the set of eigenvalues of A + BK is contained in  $\mathscr{P}$  and at the same time a robustness measure is optimized.

There are many papers that cover the placing of poles in specified regions like disks, strips or sectors, or the optimized placement of poles, see e.g. [14,24,39,40,49,68,71,74,77,86] and the references therein. A clear and practical formulation of such a general robust measure as well as suitable algorithms to determine this optimized pole assignment will depend on the application and on the set  $\mathcal{P}$ . In the stabilization problem this is the left-half plane or in the case of damped stabilization a particular part of the left-half plane, see [37]. If the set  $\mathcal{P}$  is a very small region of the complex plane, as when it has exactly *n* points, then, as we have demonstrated above, even an optimization of some robustness measures may still yield a very sensitive system, but if the set  $\mathcal{P}$  covers a large area in the complex plane, then quite good results may be obtained, see for example [22].

In the case of stabilization the robustness measure would certainly include the distance to instability, i.e., the smallest perturbation that makes the closed-loop system have an unstable eigenvalue. To make sure that the closed-loop system is securely stable, a constraint should be added in the optimization that guarantees that the perturbation bounds are smaller than the distance to instability. To verify and guarantee this constraint the distance to instability as well as the perturbation bound have to be computed, which alone is a difficult numerical problem, see [21]. In the context of stabilization this would be a part of the optimization loop and from this it may already be seen that the development of good numerical methods for this optimized stabilization is an important but extremely difficult problem that needs a lot of further attention, see also [61].

For large control problems with only few unstable poles the situation can be reduced to a small problem provided one can design a method for the separation of eigenvalues inside  $\mathcal{P}$  and outside of  $\mathcal{P}$ . If this can be done, then the complexity of the optimization problem can be drastically reduced, see [70,36,82] and the references therein.

As we have mentioned already before, for the stabilization problem there are also other approaches to design a stabilizing feedback, such as the solution of Lyapunov or Riccati equations or just the solution of the linear quadratic control problem which we discuss in the next section. A comparison of stabilization via pole placement and linear quadratic control is given in Section 4.

#### 3. Linear quadratic control

For the solution of the linear quadratic control problem, i.e., to minimize (4) subject to (1), a large number of approaches have been discussed in the literature, see the monographs [58,65,51,73]. Let us compare the Riccati equation approach with the solution of the two-point boundary value problem via a matrix pencil approach. An observation of Van Dooren [80] is that it suffices to study the deflating subspaces of the pencil  $(\mathscr{E}_{c},\mathscr{A}_{c})$  in (6). Suppose  $(\mathscr{E}_{c},\mathscr{A}_{c})$  has an *n*-dimensional deflating subspace associated with eigenvalues in the left-half plane. Let this subspace be spanned by the columns of a matrix  $\mathcal{U}$ , partitioned analogous to the pencil as

$$\mathscr{U} = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \end{bmatrix}.$$
(25)

Then, if  $U_1$  is invertible, the optimal control is a linear feedback of the form  $u(t) = U_3 U_1^{-1} x(t)$ . The solution of the associated Riccati equation (9) is  $X = U_2 U_1^{-1}$ , see [58] for details. We see that an explicit solution of the Riccati equation is not needed to determine the optimal control and it is also clear that the sensitivity of the computation of  $U_3U_1^{-1}x(t)$  may be different than that of the procedure to first compute  $X = U_2 U_1^{-1}$  and then the feedback  $u(t) = -R^{-1}B^T X x(t)$  from this. In particular if the matrix R is close to singular, then the coefficients in the Riccati equation (9) may be highly corrupted so that a solution approach via the Riccati equation may be completely useless. We demonstrate these observations in the following example.

**Example 2.** Let U be a randomly generated real orthogonal matrix, L = 0,  $A = U \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} U^{T}$ , B =

 $U, R = \begin{bmatrix} 0.5 & 0 \\ 0 & \gamma \end{bmatrix} \text{ and } Q = U \begin{bmatrix} 6 & 0 \\ 0 & 3\gamma \end{bmatrix} U^{\mathrm{T}} \text{ where } \gamma > 0.$ The positive-semidefinite (stabilizing) solution of the corresponding algebraic Riccati equation (9) is  $X = U \begin{bmatrix} 3 & 0 \\ 0 & 3\gamma \end{bmatrix} U^{\mathrm{T}}$ , the associated feedback gain matrix  $K = -\begin{bmatrix} 6 & 0 \\ 0 & 3 \end{bmatrix} U^{\mathrm{T}}$  and the closed-loop spectrum is  $\{-4, -2\}$ , both independent of the value of  $\gamma$ . Since U is orthogonal, we see that ||K||is small and hence we do not expect large perturbations in the solution. The solution via the Riccati equation, however, depends on  $\gamma$  and hence we may expect that the feedback K when computed via the Riccati equation will depend heavily on  $\gamma$ .

We applied the MATLAB m-files are, care from different versions of the MATLAB control tool box [54] which are solvers for algebraic Riccati equations and compare the results with those obtained by just computing the deflating subspace by the MATLAB implementation qz of the QZ-algorithm. The Riccati solution is used to compute  $K = -R^{-1}B^{T}X$  while via the deflating subspace (25) of  $\alpha \mathscr{E}_{c} - \beta \mathscr{A}_{c}$ , the feedback K is directly obtained as  $U_{3}U_{1}^{-1}$ . The method *are* uses the Hamiltonian matrix  $\mathscr{H}$  as in (8) to determine the Riccati solution X while the method *care* works on a balanced version of  $\mathscr{H}$  if  $(\lambda_{\min}(R)/\lambda_{\max}(R)) \ge \sqrt{eps}$  and on the extended pencil  $\alpha \mathscr{E}_c - \beta \mathscr{A}_c$  as in (6) otherwise.

The relative error in X and K for all three methods and different values of  $\gamma$  are listed in Table 2. We see that the direct computation of the optimal control via the subspace yields much smaller relative errors than the solution via the Riccati equation. Note that the subspace method always computed the Riccati solution to high relative accuracy.

γ	Method	$\frac{  \hat{X}-X  _2}{ X _2}$	$\frac{  \hat{K}-K  _2}{ K _2}$
	are	$7.6  imes 10^{-16}$	$2.1 \times 10^{-14}$
$10^{-2}$	care	$7.0 imes10^{-16}$	$1.3  imes 10^{-15}$
	qz	$2.4  imes 10^{-16}$	$4.9 \times 10^{-15}$
	are	$3.5  imes 10^{-11}$	$5.7 \times 10^{-7}$
$10^{-6}$	care	$3.1  imes 10^{-12}$	$3.2 \times 10^{-9}$
	qz	$2.6  imes 10^{-15}$	$4.7 \times 10^{-11}$
	are	$1.8  imes 10^{-8}$	$9.1 \times 10^{-1}$
$10^{-9}$	care	$2.1  imes 10^{-8}$	$1.3  imes 10^{-4}$
	qz	$1.6  imes 10^{-15}$	$5.9  imes 10^{-9}$
	are	$7.7  imes 10^{-5}$	$1.2  imes 10^4$
$10^{-13}$	care	$9.2 \times 10^{-5}$	$3.9  imes 10^1$
	qz	$1.7  imes 10^{-15}$	$5.0 imes10^{-4}$

Table 2				
Relative	errors	in	Example	2

This example demonstrates that the solution of the linear quadratic control problem via the solution of the algebraic Riccati equation presents a dangerous detour that may lead to very bad results and is really not necessary, since the feedback and the closed-loop matrix can be computed from the deflating subspace of the extended pencil directly. This is even more critical in the situation that R is indefinite or singular as in the  $H_{\infty}$  problem discussed below. The situation is even worse in the case of descriptor systems, see [8,9,58], where it is known that the Riccati equation may not have anything to do with the solution of the optimal control problem [48].

But also for the linear quadratic control problem the question of robustness has to be asked in terms of the performance criterion, i.e., the choice of Q, L, R which, as we have seen in Example 2, is critical in the Riccati approach. But since this is a freedom in the problem, we should make use of it to optimize the robustness. In the context of stabilization or other regions  $\mathscr{P}$  of the complex plane we may, therefore, formulate the *optimized linear quadratic control problem*.

**Problem 4.** Consider matrices  $A \in \mathbb{R}^{n,n}$ ,  $B \in \mathbb{R}^{n,m}$  and a set  $\mathscr{P} \subset \mathbb{C}$ . Determine cost matrices Q, L, R such that the closed-loop system obtained via the solution of the associated linear quadratic control problem has eigenvalues that are contained in  $\mathscr{P}$  and at the same time a robustness measure is optimized.

If the robustness measure in Problem 4 is the same as in Problem 3, then these two problems are actually equivalent.

**Proposition 2.** Consider matrices  $A \in \mathbb{R}^{n,n}$ ,  $B \in \mathbb{R}^{n,m}$  and a set  $\mathscr{P} \subset \mathbb{C}$ . Consider furthermore the optimized linear quadratic control Problem 4 and the optimized robust pole assignment Problem 3. If the same robustness measure is used in both problems, then the problems are equivalent, i.e., they have the same solution sets.

**Proof.** Since the feedbacks in Problem 3 are not restricted, it is clear that the solution set of Problem 3 contains the solution set of Problem 4. Suppose now that a feedback gain K optimizes Problem 3. Choosing an arbitrary positive-definite matrix R and setting  $L = -K^{T}R$ ,  $Q = LR^{-1}L^{T}$ , it follows that the linear quadratic control generates the same feedback gain matrix K as well as the same closed-loop system A + BK. Hence the solution set of Problem 3 is contained in the solution set of Problem 4.

It should be noted, however, that in many applications cost functionals with L=0 are used. In this situation the optimal solution via Problem 4 may be worse than that of Problem 3 as the following example demonstrates, see also Example 4.

**Example 3.** Consider the scalar system with A=1 and B=1 and the set  $\mathscr{P}=\{x | \text{Re} x \le -\alpha, 0 < \alpha < 1\}$ . Obviously in this case the distance to uncontrollability satisfies  $d_u(A, B) = 1$ , and the scaled spectral condition is  $\kappa(A + BK) = 1$  for arbitrary K. Thus we only need to minimize  $||K||_2$ . For Problem 3 the optimal feedback is  $K = -(1 + \alpha)$  and the closed-loop system is  $A + BK = -\alpha$ . However, for Problem 4 with L=0, the optimal solution, i.e., the minimum norm K, is K = -2 which is obtained with arbitrary R > 0 and Q=0. The associated closed-loop system is A + BK = -1. In fact for R > 0 and  $Q \ge 0$  the pole of A + BK is  $-\sqrt{1 + Q/R}$  which cannot be greater than -1.

It follows from this example that in order to obtain results which are as good as those from optimized robust pole placement the block L in the cost functional has to be included in the optimization.

As we have discussed already in the context of pole assignment, there are many different possibilities of general robust measures. These depend on the specific application and lead to different numerical methods. An analysis of different criteria should deserve more attention. Some numerical examples in the context of stabilization are discussed in the next section.

# 4. Stabilization

In this section we compare the results obtained from optimized robust pole assignment and optimized linear quadratic control for the specific problem of stabilization, i.e., the set  $\mathscr{P}$  is the open left-half plane.

Our first example discusses the optimization of the condition number  $\mathscr{S}$  in (24) in the particular situation that in the cost functional we use L = 0.

**Example 4.** Consider the stabilization problem with A = diag(1,2,3,4) and  $B = [1,1,1,1]^{T}$  and a stability margin of 0.5, i.e.,  $\mathcal{P} = \{\lambda \in \mathbb{C} \mid \text{Re}(\lambda) \leq -0.5\}.$ 

We used a heuristic "random search" algorithm for the optimal poles as in [61], to minimize the condition number  $\mathscr{S}$  in (24). For the solution of the pole-placement problem a MATLAB code based on the method of Miminis and Paige [63] was used. It should be noted that the MATLAB code *place* often generated incorrect results, which is probably due to a small distance to instability in some of the cases. The computed optimal poles, the norm of the feedback gain and the condition

Method	Closed-loop poles	K	S	Dis
Pole placement	$-0.5 \pm 3.69i, -0.5 \pm 1.02i$	222	$1.1 \times 10^5$	0.005
LQ	-12.6, -4.26, -3.04, -1.66	$2.0  imes 10^3$	$3.9  imes 10^7$	0.013

Table 3 A comparison between stabilization by LQ and pole placement

number  $\mathscr{S}$  are listed in Table 3, as well as the distance to instability displayed in column *dis* of the closed-loop matrix A + BK. The distance to instability was computed by the method of Byers [21].

For comparison, we used the solution of the optimized linear quadratic control problem with a shift, see e.g. [37], to compute the feedback gain using the MATLAB code *surv* based on the structure preserving Algorithm 1 below to determine the feedback gains. In the cost functional we chose L = 0 and  $R = 50 * ||B||^2/k$  with k = 1, ..., 100 as well as  $R = ||B||^2/2^{k+1}$  with k = 1, ..., 20. For each such R we chose 100 randomly chosen unit norm positive definite matrices Q. Note that, as desired, all eigenvalues of A + BK have real parts less than -0.5. Among all tests the minimum for  $\mathscr{S}$  was obtained for  $R = (\frac{1}{2})^6$  (note ||B|| = 2). The results are also shown in Table 3.

We see from this example, as we have already discussed before, that optimized robust pole assignment performs better than optimized linear quadratic control with L = 0. On the other hand even for this small-sized single-input problem the optimal condition number is very large.

Furthermore, we observe and this is typical, see also [61], that the optimal condition number is obtained with eigenvalues close to or on the boundary of the desired region. Thus if we choose the region  $\mathscr{P}$  to be the open left-half plane then we will typically get a small distance to instability. For this reason and to show that more theoretical investigation is necessary, in the next example we compare different optimality criteria.

**Example 5.** Let  $A = \begin{bmatrix} 1 & 1 \\ 0 & 2 \end{bmatrix}$ ,  $B = I_2$  and  $\mathscr{P} = \{\lambda \in \mathbb{C} | \operatorname{Re}(\lambda) \leq -1\}$ . As robustness measures we minimize  $\kappa_F$ ,  $||K||_F$  and  $\mathscr{S}_F = \kappa_F \sqrt{1 + ||K||_F^2}$ , respectively, where the index F indicates that the Frobenius norm is used. Clearly in this case  $K = T\Lambda T^{-1} - A$  for an arbitrary nonsingular real matrix T and arbitrary real  $\Lambda$  with eigenvalues in the required region.

If the scaled spectral condition number of the closed-loop system is to be minimized, then the optimal solution is obtained with an orthogonal matrix T and freely chosen  $\Lambda$ .

In the optimization of  $||K||_F$  and  $\mathscr{S}_F$  the optimal case is that  $\Lambda$  has a pair of complex conjugate eigenvalues. Let

$$\Lambda = \begin{bmatrix} \alpha & \beta \\ -\beta & \alpha \end{bmatrix}.$$

The general form of T is

$$T = \gamma T_{\rm s} \begin{bmatrix} a & b \\ 0 & 1 \end{bmatrix},$$

where  $\gamma, a \neq 0$  and  $T_s$  is a plane rotation. Since  $T_s$  commutes with  $\Lambda$  and since  $\gamma$  does not affect the norms, we can set  $T_s = I_2$  and  $\gamma = 1$ . To simplify the computation of the minimal  $\mathscr{S}_F$  we furthermore set b = 0, which only gives a suboptimal result. In Table 4 we give the resulting values of  $\mathscr{S}_F$ ,  $||K||_F$ 

Objective	Closed-loop poles	$  K  _{\mathrm{F}}$	$\kappa_{ m F}$	$\mathcal{S}_{\mathrm{F}}$	dis
$\kappa_{ m F}$	{-1}	3.74	2	7.75	1.0
$  K  _{\mathrm{F}}$	$-1 \pm 0.5 \times 10^{-8} i$	3.54	$2.4  imes 10^8$	$8.7  imes 10^8$	0.56
$\mathscr{S}_{\mathrm{F}}$	$-1 \pm 0.52i$	3.67	2.001	7.61	0.9994

Та	ible 4			
Α	comparison	of	optimality	criteria

as well as the distance to instability *dis* of the associated closed-loop matrix A + BK. Here in the optimization of  $\kappa_F$  we have chosen both eigenvalues to be at -1.

The associated feedback gain matrices in the three cases are

[2 1]	2.4534	0 ]		2	0.4656	
	0.2056	2.5466],	_	0.4988	3	,

respectively.

We see from this example that a pure optimization of  $||K||_F$  may lead to drastically different results than an optimization of  $\kappa_F$  and  $\mathscr{S}_F$ , but we also see that a detailed further investigation is necessary to obtain the best possible criteria.

## 5. Structure preservation

In the context of the linear quadratic control problem the second important topic that needs to be discussed, is the preservation of structure.

A feature of the pencils associated with the two-point boundary value problem (5) is that they have algebraic structures which lead to a certain symmetry in the spectrum. Roundoff errors can destroy this symmetry leading to physically meaningless results unless the numerical method also preserves the algebraic structure, see [79]. Moreover, preservation of the algebraic structure usually leads to more efficient as well as more accurate numerical methods. Let us briefly introduce the relevant structures.

# Definition 3. Let

$$J\!:=\!\begin{bmatrix} 0 & I_n \\ -I_n & 0 \end{bmatrix}\!,$$

where  $I_n$  is the  $n \times n$  identity matrix.

- (a) A matrix  $\mathscr{H} \in \mathbb{R}^{2n \times 2n}$  is Hamiltonian if  $(\mathscr{H}J)^{\mathrm{T}} = \mathscr{H}J$  and a matrix  $\mathscr{H} \in \mathbb{R}^{2n \times 2n}$  is skew-Hamiltonian if  $(\mathscr{H}J)^{\mathrm{T}} = -\mathscr{H}J$ .
- (b) A matrix  $\mathscr{U} \in \mathbb{R}^{n \times n}$  is symplectic if  $\mathscr{U}J\mathscr{U}^{\mathsf{T}} = J$  and a matrix  $\mathscr{U} \in \mathbb{R}^{2n \times 2n}$  is orthogonal symplectic if  $\mathscr{U}J\mathscr{U}^{\mathsf{T}} = J$  and  $\mathscr{U}\mathscr{U}^{\mathsf{T}} = I_{2n}$ . The group of orthogonal symplectic matrices in  $\mathbb{R}^{n \times n}$  is denoted by  $\mathscr{U}\mathscr{L}_{2n}$ .

(c) We call a real matrix Hamiltonian quasi-triangular if it is Hamiltonian and has the form

$$\begin{bmatrix} F & G \\ 0 & -F^{\mathrm{T}} \end{bmatrix},$$

where F is quasi-triangular in real Schur form, see [32]. If a Hamiltonian matrix  $\mathscr{H}$  can be transformed into Hamiltonian quasi-triangular form by a similarity transformation with a matrix  $\mathscr{U} \in \mathscr{US}_{2n}$ , then we say that  $\mathscr{U}^{\mathsf{T}}\mathscr{H}\mathscr{U}$  has Hamiltonian Schur form.

The reduced Euler-Lagrange equations (7) involve a Hamiltonian matrix, but the pencil (6) does not directly have this structure. Nonetheless many of the properties of Hamiltonian matrices carry over, see [58]. Furthermore, we may endow the pencil (6) with a similar structure by embedding the Euler-Lagrange equations (5) into a larger system. If m is even then this is easily done by splitting u(t), B, L, R into half-sized parts and a permutation of the pencil, see [8]. If m is odd then we may apply this splitting after introducing an artificial input. The resulting pencil (after some permutation) has the form

with one Hamiltonian and one skew-Hamiltonian matrix.

The solution of the eigenproblem for Hamiltonian matrices and skew-Hamiltonian/Hamiltonian pencils has been a topic of several publications, see [8,17,52,56–58] and the references therein. The goal is to obtain a numerically backward stable method, that has a complexity of  $O(n^3)$  and at the same time preserves the structure. There are two main reasons why this problem is difficult. First of all one needs a triangular-like form under orthogonal symplectic similarity transformations from which the desired invariant subspaces can be read off. Such a Hamiltonian Schur form was first suggested in [64] but not every Hamiltonian matrix or skew-Hamiltonian/Hamiltonian pencil has such a condensed form, see [53,56,57]. The second difficulty arises from the fact that even if a Hamiltonian Schur form exists, it is still difficult to construct a method with the desired features, see [2,3,9,10,19,20].

We dicuss here only the computation of the structured Schur form for Hamiltonian matrices. For skew-Hamiltonian/Hamiltonian pencils we refer the reader to [9,56,57]. Necessary and sufficient conditions for the Hamiltonian Schur form are given by the following theorem.

**Theorem 4** (Lin et al. [53]). Let  $\mathscr{H}$  be a real Hamiltonian matrix, let  $i\alpha_1, \ldots, i\alpha_v$  be its pairwise distinct nonzero purely imaginary eigenvalues and let  $U_k$ ,  $k = 1, \ldots, v$ , be the associated invariant subspaces. Then the following are equivalent:

- (i) There exists a real symplectic matrix  $\mathscr{Z}$  such that  $\mathscr{Z}^{-1}\mathscr{H}\mathscr{Z}$  is real Hamiltonian quasitriangular.
- (ii) There exists a real orthogonal symplectic matrix  $\mathcal{U}$  such that  $\mathcal{U}^{T}\mathcal{H}\mathcal{U}$  is real Hamiltonian quasi-triangular.
- (iii)  $U_k^H J U_k$  is congruent to J for all k = 1, ..., v, where J is always of the appropriate dimension.

A similar theorem for skew-Hamiltonian/Hamiltonian pencils has been given in [56,57].

This result shows that whenever a structured triangular form exists, then it also exists under orthogonal transformations and hence there is hope that these forms and therefore also the eigenvalues and invariant and deflating subspaces can be computed with structure preserving numerically stable methods.

Let us first discuss the computation of eigenvalues. It is well known that if  $\mathscr{H}$  is a Hamiltonian matrix, then  $\mathscr{H}^2$  is a skew-Hamiltonian matrix for which a structure preserving method was suggested in [81]. This suggests computing the eigenvalues of  $\mathscr{H}$  by taking square roots of the eigenvalues of  $\mathscr{H}^2$ . Unfortunately, in a worst case scenario via this approach one might obtain only half of the possible accuracy in the computed eigenvalues [19,81]. A way out of this dilemma was recently presented in [11]. This approach uses the following decomposition.

**Theorem 5** (Benner et al. [11]). Let  $\mathscr{H}$  be Hamiltonian. Then there exist  $Q_1, Q_2 \in \mathscr{U}\mathscr{S}_{2n}$ , such that

$$\mathcal{Q}_1^{\mathrm{T}} \mathscr{H} \mathcal{Q}_2 = \begin{bmatrix} H_{11} & H_{12} \\ 0 & H_{22} \end{bmatrix}, \tag{27}$$

with  $H_{11}$  upper triangular and  $H_{22}^{T}$  quasi-upper triangular. Furthermore the eigenvalues of  $\mathscr{H}$  are the square roots of the eigenvalues of  $-H_{11}H_{22}^{T}$ .

Note that the resulting matrix in (27) is neither Hamiltonian nor similar to  $\mathscr{H}$ , but a simple calculation shows that both  $Q_1^{\mathrm{T}}\mathscr{H}^2Q_1$  and  $Q_2^{\mathrm{T}}\mathscr{H}^2Q_2$  are real skew-Hamiltonian quasi-triangular. For skew-Hamiltonian/Hamiltonian pencils similar results have been given in [9]. After the form (27) has been computed, one can compute the eigenvalues of  $\mathscr{H}$  by solving  $1 \times 1$  or  $2 \times 2$  eigenvalue problems and taking square roots without loosing accuracy. For algorithmic details, a detailed error analysis as well as illustrative numerical examples, see [11], where it is demonstrated that these methods speed up the computation of eigenvalues while still achieving full possible accuracy.

This new approach has also been extended to the computation of the desired deflating and invariant subspaces. Let us first introduce the basic theory behind the method. Let for  $A \in \mathbb{R}^{n \times n}$  the sets  $\lambda_{-}(A), \lambda_{+}(A), \lambda_{0}(A)$  denote the part of the spectrum of A in the open left half-plane, in the open right half-plane and on the imaginary axis, respectively, and denote the associated invariant subspaces by  $\text{Inv}_{-}(A), \text{Inv}_{+}(A), \text{Inv}_{0}(A)$ . In [10] it has been observed that for  $A \in \mathbb{R}^{n \times n}$  and  $B = \begin{bmatrix} 0 & A \\ A & 0 \end{bmatrix}$ , if one determines an orthogonal matrix such that

$$B\begin{bmatrix} Q_1\\ Q_2\end{bmatrix} = \begin{bmatrix} Q_1\\ Q_2\end{bmatrix} R,$$
(28)

where

$$\lambda_{+}(B) \subseteq \lambda(R) \subseteq \lambda_{+}(B) \cup \lambda_{0}(B), \tag{29}$$

then

$$\operatorname{range}\{Q_1 + Q_2\} = \operatorname{Inv}_+(A) + \mathcal{N}_1 \quad \text{where } \mathcal{N}_1 \subseteq \operatorname{Inv}_0(A),$$
(30)

$$\operatorname{range}\{Q_1 - Q_2\} = \operatorname{Inv}_{-}(A) + \mathcal{N}_2 \quad \text{where } \mathcal{N}_2 \subseteq \operatorname{Inv}_0(A).$$
(31)

Moreover, if we partition  $R = \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix}$  with  $\lambda(R_{11}) = \lambda_+(B)$  and, accordingly,  $Q_1 = [Q_{11} & Q_{12}]$ ,  $Q_2 = [Q_{21} & Q_{22}]$ , then  $B \begin{bmatrix} Q_{11} \\ Q_{21} \end{bmatrix} = \begin{bmatrix} Q_{11} \\ Q_{21} \end{bmatrix} R_{11}$ (32)

and there exists an orthogonal matrix Z such that

$$\frac{\sqrt{2}}{2}(Q_{11} + Q_{21}) = [0 P_{+}]Z,$$

$$\frac{\sqrt{2}}{2}(Q_{11} - Q_{21}) = [P_{-} 0]Z,$$
(33)

where  $P_+$ ,  $P_-$  are orthogonal bases of  $Inv_+(A)$ ,  $Inv_-(A)$ , respectively.

In the case of a Hamiltonian matrix

$$\mathscr{H} = \begin{bmatrix} F & G \\ H & -F^{\mathrm{T}} \end{bmatrix}$$

one considers the block matrix

$$\mathscr{B} = \begin{bmatrix} 0 & \mathscr{H} \\ \mathscr{H} & 0 \end{bmatrix}$$

and, using the block permutation

$$\mathscr{P} = \begin{bmatrix} I_n & 0 & 0 & 0 \\ 0 & 0 & I_n & 0 \\ 0 & I_n & 0 & 0 \\ 0 & 0 & 0 & I_n \end{bmatrix},$$

one obtains that

$$\tilde{\mathscr{B}} := \mathscr{P}^{\mathrm{T}} \mathscr{B} \mathscr{P} = \begin{bmatrix} 0 \ F & 0 & G \\ F & 0 & G & 0 \\ 0 \ H & 0 & -F^{\mathrm{T}} \\ H & 0 & -F^{\mathrm{T}} & 0 \end{bmatrix}$$
(34)

is again Hamiltonian. Furthermore it follows from Theorem 4 that  $\tilde{\mathscr{B}}$  has a Hamiltonian Schur form.

**Theorem 6** (Benner et al. [10]). Let  $\mathscr{H}$  be Hamiltonian and let  $\mathscr{B} = \begin{bmatrix} 0 & \mathscr{H} \\ \mathscr{H} & 0 \end{bmatrix}$ . Then there exists an orthogonal matrix  $\mathscr{U}$  such that

$$\mathscr{U}^{\mathsf{T}}\mathscr{B}\mathscr{U} = \begin{bmatrix} R & D \\ 0 & -R^{\mathsf{T}} \end{bmatrix} =: \mathscr{R}$$
(35)

is in Hamiltonian quasi-triangular form and  $\lambda_{-}(R) = \emptyset$ . Moreover,  $\mathcal{U} = \mathscr{P}\mathcal{W}$  with  $\mathcal{W} \in US_{4n}$ , and  $\mathscr{R} = \mathscr{W}^{\mathrm{T}}\mathscr{B}\widetilde{\mathcal{W}}$ , (36)

*i.e.*,  $\mathcal{R}$  is the Hamiltonian quasi-triangular form of the Hamiltonian matrix  $\tilde{\mathcal{B}}$ . Furthermore, if  $\mathcal{H}$  has no purely imaginary eigenvalues, then R has only eigenvalues with positive real part.

The structure preserving, numerically stable algorithm to compute the invariant subspace of a Hamiltonian matrix associated with the eigenvalues in the left-half plane is then as follows.

## Algorithm 1.

Input: A Hamiltonian matrix  $\mathscr{H}$  having an n-dimensional Lagrangian invariant subspace associated with the eigenvalues in the left half-plane.

*Output*:  $Y \in \mathbb{R}^{2n \times n}$ , with  $Y^T Y = I_n$ , such that the columns of Y span this invariant subspace.

Step 1: Apply Algorithm 2 of [11] to  $\mathscr{H}$  and compute orthogonal symplectic matrices  $Q_1, Q_2 \in \mathscr{US}_{2n}$  such that

$$Q_1^{\mathrm{T}} \mathscr{H} Q_2 = \begin{bmatrix} H_{11} & H_{12} \\ 0 & H_{22} \end{bmatrix}$$

is the decomposition (27).

Step 2: Determine an orthogonal matrix  $Q_3$ , such that

$$Q_3^{\mathrm{T}} \begin{bmatrix} 0 & -H_{22}^{\mathrm{T}} \\ H_{11} & 0 \end{bmatrix} Q_3 = \begin{bmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{bmatrix}$$

is in real Schur form ordered such that the eigenvalues of  $T_{11}$  have positive real part and the eigenvalues of  $T_{22}$  have negative real part.

Step 3: Use the orthogonal symplectic reordering scheme of [20] to determine an orthogonal symplectic matrix  $V \in \mathscr{US}_{4n}$  such that with

$$U = \begin{bmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{bmatrix} := \begin{bmatrix} Q_1 Q_3 & 0 \\ 0 & Q_2 Q_3 \end{bmatrix} V$$

we have the Hamiltonian quasi-triangular form

$$U^{\mathrm{T}} \mathscr{B} U = \begin{bmatrix} F_{11} & F_{12} & G_{11} & G_{12} \\ 0 & F_{22} & G_{21} & G_{22} \\ 0 & 0 & -F_{11}^{\mathrm{T}} & 0 \\ 0 & 0 & -F_{12}^{\mathrm{T}} & -F_{22}^{\mathrm{T}} \end{bmatrix},$$

where  $F_{11}, F_{22}$  are quasi-upper triangular with eigenvalues only in the closed right-half plane.

Step 4: Set  $\hat{Y}:=(\sqrt{2}/2)(U_{11}-U_{21})$ . Compute Y, an orthogonal basis of range  $\{\hat{Y}\}$ , using any numerically stable orthogonalization scheme, for example a rank-revealing QR-decomposition; see, e.g. [23].

End

Generalizations of these results to the complex case and algorithms are presented in [12]. Corresponding results and methods for skew-Hamiltonian/Hamiltonian pencils have been constructed in [9].

It should be noted that these new methods are already very close to the desired structure preserving methods but they are still not optimal, since not all structures are fully exploited. But the methods are more efficient and at least as accurate as methods that do not address structure preservation. This approach works, in principle, also for Hamiltonian matrices with eigenvalues on the imaginary axis

provided the appropriate subspaces can be separated. When this is the case and how the numerical method can detect this, as well as the perturbation analysis is still under investigation, see [62]. A complete analysis of this case will be also very important for the treatment of  $H_{\infty}$  control problems, that we discuss in the next section.

#### 6. Standard $H_{\infty}$ control

The solution of the standard  $H_{\infty}$  control problem addresses another robustness measure in the computation of a feedback solution, which is different from the criteria that we have discussed so far. For the numerical solution of the  $H_{\infty}$  control problem the usual procedure is to use a optimization scheme to determine the smallest  $\gamma > 0$  so that all three conditions (A1)–(A3) in Section 1 hold by determining the first value of  $\gamma$  where one of these conditions fail, see for example [33,67,87]. In each step of the optimization procedure, two linear quadratic optimal control problems are solved plus a positivity check.

Typically in current design packages like the MATLAB robust control toolbox [55], the solution is obtained by a procedure which uses the solution of algebraic Riccati equations to determine  $X_{\infty}$  and  $Y_{\infty}$ .

In view of the discussion in Section 3 on the solution of linear quadratic control problems and Riccati equations we should construct new methods for the  $H_{\infty}$  control problem that avoid the detour via the Riccati equation. This conclusion is complemented by the observation that during the optimization procedure, typically one or both of the Riccati solutions becomes very large in norm. This leads to the question whether a numerical solution of the  $H_{\infty}$  via the solution of Riccati equations makes sense at all, since in order to obtain a robust control, a highly ill-conditioned numerical problem has to be solved.

The usual way out of this dilemma in practice is to compute suboptimal controls, see [34,67]. But in view of the previous discussions one might ask whether this potential ill-conditioning is inherent in the problem formulation or due to the approach for its solution. Let us consider an example.

**Example 6.** Let A = 1,  $B_1 = 2$ ,  $B_2 = 1$ ,  $C_1 = 1$  and  $C_2 = \sqrt{3}$ . Then for  $\gamma > \gamma_{x,1} = \sqrt{2}$  the matrix  $\mathscr{H}_{\infty}$  in (13) has no purely imaginary eigenvalues and hence a Lagrange subspace associated with the stable eigenvalues always exists. The stabilizing solution of the Riccati equation, however, is

$$X(\gamma) = \frac{\gamma^2 + \gamma \sqrt{2\gamma^2 - 4}}{\gamma^2 - 4}.$$

For  $\gamma > \gamma_{x,2} = 2$  we have that  $X(\gamma)$  is positive definite and for  $\gamma < \gamma_{x,2}$ ,  $X(\gamma)$  is negative definite. For  $\gamma = \gamma_{x,2}$  the Riccati solution is not defined.

Analogously for the Riccati equation associated with  $\mathscr{J}_{\infty}$  in (15) we have  $\gamma_{y,1} = (2\sqrt{13}/13)$  and  $\gamma_{y,2} = (\sqrt{3}/3)$ , and the associated stabilizing solution of the Riccati equation is

$$Y(\gamma) = \frac{\gamma^2 + \gamma\sqrt{13\gamma^2 - 4}}{3\gamma^2 - 1}$$

It follows that the optimal parameter  $\gamma_{opt}$  must be greater than  $\gamma_{x,2} = 2$ .

For the third condition (A3) we have  $\gamma_{x,1}^2 > \rho(X(\gamma_{x,1})Y(\gamma_{x,1}))$ , since  $X(\gamma_{x,1}) = -1$  and

$$Y(\gamma_{x,1}) = \frac{2(1+\sqrt{11})}{5}.$$

But  $\gamma_{x,1}$  is obviously not optimal. So in a typical optimization procedure to determine the optimal  $\gamma$  one needs first to determine  $\gamma_{x,2}$  and  $\gamma_{y,2}$ , but  $X(\gamma_{x,2}), Y(\gamma_{y,2})$  are not defined.

We see from this example that, as for the solution of the linear quadratic control problem, the Riccati solutions  $X_{\infty}$  and  $Y_{\infty}$  should be avoided. Fortunately, this can again be done quite easily. In [87, Theorem 16.16, p. 445] it is shown that conditions (A1)–(A3) may be replaced by the alternative conditions

(B1) There exist matrices  $Q_1, Q_2 \in \mathbb{R}^{n,n}$  such that

$$\mathscr{H}_{\infty}\begin{bmatrix} Q_1\\ Q_2\end{bmatrix} = \begin{bmatrix} Q_1\\ Q_2\end{bmatrix} T_x,$$

where  $T_x$  has only eigenvalues with nonpositive real parts and  $Q_1^T Q_2 = Q_2^T Q_1$ . (B2) There exist matrices  $U_1, U_2 \in \mathbb{R}^{n,n}$  such that

$$\begin{bmatrix} U_1 \\ U_2 \end{bmatrix}^{\mathrm{T}} \mathscr{J}_{\infty} = T_y \begin{bmatrix} U_1 \\ U_2 \end{bmatrix}^{\mathrm{T}},$$

where  $T_y$  has only eigenvalues with nonpositive real parts and  $U_1^T U_2 = U_2^T U_1$ . (B3)

$$\begin{bmatrix} Q_2^{\mathrm{T}} Q_1 & \gamma^{-1} Q_2^{\mathrm{T}} U_2 \\ \gamma^{-1} U_2^{\mathrm{T}} Q_2 & U_2^{\mathrm{T}} U_1 \end{bmatrix}$$

is symmetric positive semidefinite.

If these conditions hold then  $||T_{zw}||_{\infty} \leq \gamma$  and the admissable controller is in descriptor form

$$\hat{E}\dot{q} = \hat{A}q + \hat{B}y$$

$$u = \hat{C}q + \hat{D}y,$$
(37)

with  $\hat{E} = U_1^T Q_1 - \gamma^{-1} U_2^T Q_2$ ,  $\hat{B} = U_2^T C_2^T$ ,  $\hat{C} = -B_2^T Q_2$ ,  $\hat{D} = 0$  and  $\hat{A} = \hat{E} T_x - \hat{B} C_2 Q_1 = T_y \hat{E} + U_1^T B_2 \hat{C}$ . Using this result, only the invariant subspaces of  $\mathscr{H}_{\infty}$  and  $\mathscr{J}_{\infty}$  are involved and they can be

determined via the same methods that we have discussed in the previous section. Thus not only is it possible to avoid the ill-conditioned Riccati equation but also we can employ

structure preservation as described above and as in the case of the linear quadratic control problem. The computation of these subspaces is usually much better conditioned than the computation of the Riccati solutions.

Thus, the solution of the  $H_{\infty}$  control problem should be approached via the usual optimization procedures like in [18,31,34,75], using in each optimization step Algorithm 1 to determine the subspaces in (B1) and (B2) and a Cholesky factorization to check condition (B3). An implementation and analysis of such a procedure is currently under investigation.

## 7. Conclusion

We have discussed several standard problems of linear control theory, like pole assignment, stabilization, linear quadratic and  $H_{\infty}$  control and have demonstrated some of the difficulties that arise in the numerical solution of these problems due to inherent ill-conditioning in the problem. We have also suggested several reformulated versions of the problem, which are sometimes more complicated to solve, but which yield results that are much more robust to perturbations.

## References

- J. Ackermann, Der Entwurf linearer Regelungssysteme im Zustandsraum, Regelungstechnik und Prozessdatenverarbeitung 7 (1972) 297–300.
- [2] G.S. Ammar, P. Benner, V. Mehrmann, A multishift algorithm for the numerical solution of algebraic Riccati equations, Electr. Trans. Numer. Anal. 1 (1993) 33–48.
- [3] G.S. Ammar, V. Mehrmann, On Hamiltonian and symplectic Hessenberg forms, Linear Algebra Appl. 149 (1991) 55–72.
- [4] B.D.O. Anderson, J.B. Moore, Linear Optimal Control, Prentice-Hall, Englewood Cliffs, NJ, 1971.
- [5] M. Arnold, Algorithms and conditioning for eigenvalue assignment, Ph.D. Thesis, Department of Mathematics, Northern Illinois University, De Kalb, 1993.
- [6] M. Arnold, B.N. Datta, Single-input eigenvalue assignment algorithms: a close-look, SIAM J. Matrix Anal. Appl. 19 (1997) 444–467.
- [7] M. Athans, P.L. Falb, Optimal Control, McGraw-Hill, New York, 1966.
- [8] P. Benner, R. Byers, V. Mehrmann, H. Xu, Numerical methods for linear quadratic and H<sub>∞</sub> control problems, in: G. Picci, D.S. Gillian (Eds.), Dynamical Systems, Control, Coding, Computer Vision, Progress in Systems and Control Theory, Vol. 25, Birkhäuser, Basel, 1999, pp. 203–222.
- [9] P. Benner, R. Byers, V. Mehrmann, H. Xu, Numerical computation of deflating subspaces of embedded Hamiltonian pencils, preprint SFB393/99-22, Sonderforschungsbereich 393 Th Chemnitz, Germany (1999).
- [10] P. Benner, V. Mehrmann, H. Xu, A new method for computing the stable invariant subspace of a real Hamiltonian matrix, J. Comput. Appl. Math. 86 (1997) 17–43.
- [11] P. Benner, V. Mehrmann, H. Xu, A numerically stable, structure preserving method for computing the eigenvalues of real Hamiltonian or symplectic pencils, Numer. Math. 78 (1998) 329–358.
- [12] P. Benner, V. Mehrmann, H. Xu, A note on the numerical solution of complex Hamiltonian and skew-Hamiltonian eigenvalue problems, Electron. Trans. Numer. Anal. 8 (1999) 115–126.
- [13] P. Benner, V. Mehrmann, V. Sima, S. Van Huffel, A. Varga, SLICOT a subroutine library in systems and control theory, Appl. Comp. Control Signals Circuits 1 (1999) 499–532.
- [14] C.S. Berger, Robust pole-placement algorithm for adaptive control, Proc. IEE-D 135 (1988) 493-498.
- [15] S.P. Bhattacharyya, E. De Souza, Pole assignment via Sylvester's equations, Systems Control Lett. 1 (4) (1982) 261–263.
- [16] S. Bittanti, A. Laub, J.C. Willems (Eds.), The Riccati Equation, Springer, Berlin, 1991.
- [17] A. Bunse-Gerstner, R. Byers, V. Mehrmann, Numerical methods for algebraic Riccati equations, in: S. Bittanti (Ed.), Proceedings of the Workshop on the Riccati Equation in Control, Systems, and Signals, Como, Italy, 1989, pp. 107–116.
- [18] S. Boyd, V. Balakrishnan, A regularity result for the singular values of a transfer matrix and a quadratically convergent algorithm for computing its  $L_{\infty}$  norm, Systems Control Lett. 15 (1990) 1–7.
- [19] R. Byers, Hamiltonian, symplectic algorithms for the algebraic Riccati equation, Ph.D Thesis, Cornell University, Department of Computer Science, Ithaca, NY, 1983.
- [20] R. Byers, A Hamiltonian QR-algorithm, SIAM J. Sci. Statist. Comput. 7 (1986) 212-229.
- [21] R. Byers, A bisection method for measuring the distance of a stable to unstable matrices, SIAM J. Sci. Statist. Comput. 9 (1988) 875–881.

- [22] D. Calvetti, B. Lewis, L. Reichel, On the selection of poles in the single input pole placement problem, Linear Algebra Appl. 301–303 (1999) 331–345.
- [23] T. Chan, Rank revealing QR factorizations, Linear Algebra Appl. 88/89 (1987) 67-82.
- [24] J.-H. Chou, Control systems design for robust pole-assignment in a specified circular region, Control Theory Adv. Tech. 7 (2) (1991) 237–245.
- [25] C.L. Cox, W.F. Moss, Backward error analysis for a pole assignment algorithm, SIAM J. Matrix Anal. Appl. 10 (1989) 446–456.
- [26] C.L. Cox, W.F. Moss, Backward error analysis for a pole assignment algorithm II: The complex case, SIAM J. Matrix Anal. Appl. 13 (1992) 1159–1171.
- [27] L. Dai, Singular Control Systems, Lecture Notes in Control, Information Sciences, Vol. 118, Springer, Berlin, 1989.
- [28] J. Demmel, The condition number of equivalence transformations that block diagonalize matrix pencils, in: B. Kågström, A. Ruhe (Eds.), Matrix Pencils, Lecture Notes in Mathematies, Vol. 973, Springer, Berlin, 1982, pp. 2–16.
- [29] R. Eising, Between controllable and uncontrollable, Systems Control Lett. 4 (1984) 263-264.
- [30] P. Gahinet, A.J. Laub, Numerically reliable computation of optimal performance in singular  $H_{\infty}$  control, SIAM J. Control Optim. 35 (5) (1997) 1690–1710.
- [31] Y. Genin, P. Van Dooren, V. Vermaut, Convergence of the calculation of  $H_{\infty}$ -norms and related questions, in: G. Picci, D.S. Gillian (Eds.), Proceedings of the MTNS-98, Padua, Birkhäuser, Basel, 1999.
- [32] G.H. Golub, C.F. Van Loan, Matrix Computations, 3rd Edition, Johns Hopkins University Press, Baltimore, 1996.
- [33] M. Green, D.J.N. Limebeer, Linear Robust Control, Prentice-Hall, Englewood Cliffs, NJ, 1995.
- [34] D.W. Gu, P.Hr. Petkov, M.M. Konstantinov, Direct formulae for the  $H_{\infty}$  sub-optimal central controller, SLICOT Working Note SLWN1998-7, Department of Electrical Engineering, KU Leuven, see http://www.esat.kuleuven.ac.be/pub/WGS/REPORTS/.
- [35] C. He, A.J. Laub, V. Mehrmann, Placing plenty of poles is pretty preposterous, Preprint 95–17: DFG-Forschergruppe Scientific Parallel Computing, Fak. f. Mathematik, TU Chemnitz-Zwickau, D-09107 Chemnitz, FRG, May 1995.
- [36] C. He, V. Mehrmann, Stabilization of large linear systems, in: L. Kulhavá, M. Kárný, K. Warwick (Eds.), Proceedings of the European IEEE Workshop CMP'94, Prague, September 1994, pp. 91–100.
- [37] J.J. Hench, C. He, V. Kucera, V. Mehrmann, Dampening controllers via a Riccati equation approach, IEEE Trans. Automat. Control AC-43 (1998) 1280–1284.
- [38] N.J. Higham, Accuracy and Stability of Numerical Algorithms, SIAM, Philadelphia, PA, 1996.
- [39] Y.-T. Juang, Robust stability, robust pole assignment of linear systems with structured uncertainty, IEEE Trans. Automat. Control 36 (5) (1991) 635–637.
- [40] Y.-T. Juang, K.-H. Chen, Robust pole-assignment of linear dynamic systems, Control Theory Adv. Tech. 5 (1) (1989) 67–74.
- [41] T. Kailath, Systems Theory, Prentice-Hall, Englewood Cliffs, NJ, 1980.
- [42] J. Kautsky, N.K. Nichols, P. Van Dooren, Robust pole assignment in linear state feedback, Internat. J. Control 41 (1985) 1129–1155.
- [43] L.H. Keel, J.A. Fleming, S.P. Bhattacharya, Minimum norm pole assignment via Sylvester's equation, Contemp. Math. 47 (1985) 265–272.
- [44] H.W. Knobloch, H. Kwakernaak, Lineare Kontrolltheorie, Springer, Berlin, 1985.
- [45] M.M. Konstantinov, P.Hr. Petkov, Conditioning of linear state feedback, Technical Report 93-61, Department of Engineering, Leicester University, 1993.
- [46] M.M. Konstantinov, P.Hr. Petkov, N.D. Christov, Invariants and canonical forms for linear multivariable systems under the action of orthogonal transformation groups, Kybernetika, Prague 17 (1981) 413–424.
- [47] M.M. Konstantinov, P.Hr. Petkov, N.D. Christov, Conditioning of the output pole assignment, in Proceedings of Third European Control Conference, Rome, Italy, September 1995.
- [48] P. Kunkel, V. Mehrmann, The linear quadratic control problem for linear descriptor systems with variable coefficients, Math. Control Signals Systems 10 (1997) 247–264.
- [49] V. Kucera, F.J. Kraus, Regional pole placement, Kybernetika, Prague 31 (1995) 541–546.
- [50] H. Kwakernaak, R. Sivan, Linear Optimal Control Systems, Wiley-Interscience, New York, 1972.
- [51] P. Lancaster, L. Rodman, The Algebraic Riccati Equation, Oxford University Press, Oxford, 1995.

- [52] A.J. Laub, Invariant subspace methods for the numerical solution of Riccati equations, in: S. Bittanti, A.J. Laub, J.C. Willems (Eds.), The Riccati Equation, Springer, Berlin, 1991, pp. 163–196.
- [53] W.-W. Lin, V. Mehrmann, H. Xu, Canonical forms for Hamiltonian, symplectic matrices and pencils, Lin. Alg. Appl. 301–303 (1999) 469–553.
- [54] The MathWorks, Inc., Cochituate Place, 24 Prime Park Way, Natick, MA 01760, The MATLAB Control Toolbox, Version 3.0b, 1993.
- [55] The MathWorks, Inc., Cochituate Place, 24 Prime Park Way, Natick, MA 01760, The MATLAB Robust Control Toolbox, Version 2.0b, 1994.
- [56] C. Mehl, Condensed forms for skew-Hamiltonian/Hamiltonian pencils, SIAM J. Matrix Anal. Appl. 21 (2000) 454-476.
- [57] C. Mehl, Compatible Lie and Jordan Algebras and Applications to Structured Matrices and Pencils, Ph.D. Thesis, Fakultät für Mathematik, TU Chemnitz, 09107 Chemnitz (FRG), 1998.
- [58] V. Mehrmann, in: The Autonomous Linear Quadratic Control Problem, Theory and Numerical Solution, Lecture Notes in Control and Information Sciences, Vol. 163, Springer, Heidelberg, July 1991.
- [59] V. Mehrmann, H. Xu, An analysis of the pole placement problem. I. The single-input case, Electr. Trans. Numer. Anal. 4 (1996) 89–105.
- [60] V. Mehrmann, H. Xu, Analysis of the pole placement problem II. The multi-input case, Electr. Trans. Numer. Anal. 5 (1997) 77–97.
- [61] V. Mehrmann, H. Xu, Choosing poles so that the single-input pole placement problem is well-conditioned, SIAM J. Matrix Anal. Appl. 19 (1998) 664–681.
- [62] V. Mehrmann, H. Xu, Lagrangian invariant subspaces of Hamiltonian matrices, Preprint SFB393/98-25, Sonderforschungsbereich 393, 'Numerische Simulation auf massiv parallelen Rechnern', Fakultät f
  ür Mathematik, TU Chemnitz.
- [63] G.S. Miminis, C.C. Paige, A direct algorithm for pole assignment of time-invariant multi-input linear systems using state feedback, Automatica 24 (1988) 343–356.
- [64] C.C. Paige, C.F. Van Loan, A Schur decomposition for Hamiltonian matrices, Linear Algebra Appl. 14 (1981) 11–32.
- [65] P.Hr. Petkov, N.D. Christov, M.M. Konstantinov, Computational Methods for Linear Control Systems, Prentice-Hall, Hertfordshire, UK, 1991.
- [66] P.Hr. Petkov, N.D. Christov, M.M. Konstantinov, Reliability of the algorithms and software for synthesis of linear optimal systems, Technical Report, Department of Automatics, Higher Institute of Mechanical and Electrical Engineering B1.2; 1156 Sofia, Bulgaria, 1986.
- [67] P.Hr. Petkov, D.W. Gu, M.M. Konstantinov, Fortran 77 routines for H<sub>∞</sub>, H<sub>2</sub> design of continuous-time linear control systems. SLICOT Working Note SLWN1998-8, Dept. of Electrical Engineering, KU Leuven, see http://www.esat.kuleuven.ac.be/pub/WGS/REPORTS/.
- [68] M.M. Konstantinov, P.Hr. Petkov, D.W. Gu, I. Postlethwaite, Optimal pole assignment design of linear multi-input systems, Technical Report 96–11, Department of Engineering, Leicester University, 1996.
- [69] L.S. Pontryagin, V. Boltyanskii, R. Gamkrelidze, E. Mishenko, The Mathematical Theory of Optimal Processes, Interscience, New York, 1962.
- [70] Y. Saad, Projection and deflation methods for partial pole assignment in linear state feedback, IEEE Trans. Automat. Control AC-33 (1988) 290–297.
- [71] R. Santos-Mendes, J. Aguilar-Martin, Robust pole placement design, Internat. J. Control 50 (1) (1989) 113–128.
- [72] V. Sima, An efficient Schur method to solve the stabilization problem, IEEE Trans. Automat. Control AC-26 (1981) 724–725.
- [73] V. Sima, Algorithms for Linear-Quadratic Optimization, Pure and Applied Mathematics, Vol. 200, Marcel Dekker, New York, NY, 1996.
- [74] Y.C. Soh, R.J. Evans, I.R. Petersen, R.E. Betz, Robust pole assignment, Automatica 23 (1987) 601-610.
- [75] J. Sreedhar, P. Van Dooren, A.L. Tits, A fast algorithm to compute the real structured stability radius, in: R. Jeltsch, M. Mansour (Eds.), Stability Theory: Proceedings of Hurwitz Centenary Conference, Ticino, Switzerland, Birkhäuser, Basel, 1996, pp. 219–230.
- [76] J.G. Sun, Perturbation analysis of the pole assignment problem, SIAM J. Matrix Anal. Appl. 17 (1996) 313–331.

- [77] A.L. Tits, Y. Yang, Globally convergent algorithms for rubust pole assignment by state feedback, IEEE Trans. Automat. Control 41 (1996) 1432–1452.
- [78] M. Valaš, N. Olgac, Efficient eigenvalue assignment for general linear MIMO systems, Automatica 31 (1995) 1605– 1617.
- [79] P. Van Dooren, The generalized eigenstructure problem in linear system theory, IEEE Trans. Automat. Control AC-26 (1981) 111–129.
- [80] P. Van Dooren, A generalized eigenvalue approach for solving Riccati equations, SIAM J. Sci. Statist. Comput. 2 (1981) 121–135.
- [81] C.F. Van Loan, A symplectic method for approximating all the eigenvalues of a Hamiltonian matrix, Linear Algebra Appl. 61 (1984) 233–251.
- [82] A. Varga, A Schur method for pole assignment, IEEE Trans. Automat. Control AC-26 (1981) 517-519.
- [83] A. Varga, Robust and minimum norm pole assignement with periodic state feedback, IEEE Trans. Automat. Control (2000), to appear.
- [84] W.M. Wonham, On pole assignment in multi input controllable linear systems, IEEE Trans. Automat. Control AC-12 (1967) 660–665.
- [85] W.M. Wonham, Linear Multivariable Control: A Geometric Approach, 2nd Edition, Springer, New York, 1979.
- [86] Y. Yang, A.L. Tits, On robust pole assignment by state feedback, Proceedings of the 1993 American Control Conference San Francisco, Vol. 3, 1993, pp. 2765–2766.
- [87] K. Zhou, J.C. Doyle, K. Glover, Robust and Optimal Control, Prentice-Hall, Upper Saddle River, NJ, 1996.



Journal of Computational and Applied Mathematics 123 (2000) 395-421

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.nl/locate/cam

# Krylov-subspace methods for reduced-order modeling in circuit simulation

Roland W. Freund

Bell Laboratories, Lucent Technologies, Room 2C-525, 700 Mountain Avenue, Murray Hill, New Jersey 07974-0636, USA

Received 11 September 1999; received in revised form 9 December 1999

### Abstract

The simulation of electronic circuits involves the numerical solution of very large-scale, sparse, in general nonlinear, systems of differential-algebraic equations. Often, the size of these systems can be reduced considerably by replacing the equations corresponding to linear subcircuits by approximate models of much smaller state-space dimension. In this paper, we describe the use of Krylov-subspace methods for generating such reduced-order models of linear subcircuits. Particular emphasis is on reduced-order modeling techniques that preserve the passivity of linear RLC subcircuits. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Lanczos algorithm; Arnoldi process; Linear dynamical system; VLSI interconnect; Transfer function; Padé approximation; Stability; Passivity; Positive real function

# 1. Introduction

Today's integrated electronic circuits are extremely complex, with up to tens of millions of devices. Prototyping of such circuits is no longer possible, and instead, computational methods are used to simulate and analyze the behavior of the electronic circuit at the design stage. This allows to correct the design before the circuit is actually fabricated in silicon.

The simulation of electronic circuits involves the numerical solution of very large-scale, sparse, in general nonlinear, systems of time-dependent differential-algebraic equations (DAEs); see, e.g. [10,29,30] and the references given there. These systems can be so large that time integration becomes inefficient or even prohibitive. On the other hand, electronic circuits often contain large linear subcircuits of passive components that contribute only linear equations to the system of DAEs describing the whole circuit. In particular, such linear subcircuits may result from extracted RLC models of the circuit's wiring, the so-called interconnect, models of the circuit's package, or models of wireless propagation channels. By replacing the equations corresponding to linear subcircuits by approximate models of much smaller state-space dimension, the size of the system of DAEs

describing the whole circuit can be reduced significantly, so that time integration of the resulting system becomes feasible; see, e.g. [11,20,24,26] and the references given there. In recent years, there has been a lot of interest in generating suitable reduced-order models of linear subcircuits by means of Krylov-subspace methods, such as the Lanczos algorithm and the Arnoldi process. For a survey of these recent developments, we refer the reader to [13].

In this paper, we describe the use of Krylov-subspace methods for generating reduced-order models of systems of linear DAEs, such as the ones arising in circuit simulation. Particular emphasis is on projection techniques that, when applied to a passive circuit, preserve the passivity of the circuit. We stress that the methods discussed in this paper are not restricted to systems of DAEs arising in circuit simulation and that they can be applied to general time-invariant linear dynamical systems. However, the development of these methods was mostly motivated by the need for reduced-order modeling in circuit simulation.

The remainder of the paper is organized as follows. In Section 2, we briefly review the systems of DAEs that arise in circuit simulation, and we describe how reduced-order models of linear subcircuits can be employed to reduce the dimension of these systems. In Section 3, we introduce our notion of block Krylov subspaces and review the construction of basis vectors via Lanczos and Arnoldi algorithms. In Section 4, we define reduced-order models based on projection and describe their computation via Krylov-subspace methods. In Section 5, we discuss connections with Padé and Padé-type approximants. In Section 6, we establish results on the stability and passivity of reduced-order models obtained via projection. In Section 7, numerical results for two circuit examples are reported. Finally, in Section 8, we make some concluding remarks and mention a few open problems.

Throughout this article, we use boldface letters to denote vectors and matrices. Unless stated otherwise, vectors and matrices are allowed to have complex entries. As usual,  $\overline{M} = [\overline{m_{jk}}]$ ,  $M^{T} = [m_{kj}]$ , and  $M^{H} = \overline{M}^{T} = [\overline{m_{kj}}]$  denote the complex conjugate, transpose, and the conjugate transpose, respectively, of the matrix  $M = [m_{jk}]$ , and  $M \ge 0$  means that M is Hermitian positive semi-definite. The vector norm  $||\mathbf{x}|| := \sqrt{\mathbf{x}^{H} \mathbf{x}}$  is always the Euclidean norm, and  $||M|| := \max_{||\mathbf{x}||=1} ||M\mathbf{x}||$  is the corresponding induced matrix norm. We use  $I_n$  to denote the  $n \times n$  identity matrix and  $\mathbf{0}_{n \times m}$  to denote the  $n \times m$  zero matrix; we will omit these indices whenever the actual dimensions of I and  $\mathbf{0}$  are apparent from the context. The sets of real and complex numbers are denoted by  $\mathbb{R}$  and  $\mathbb{C}$ , respectively. For  $s \in \mathbb{C}$ ,  $\operatorname{Re}(s)$  is the real part of s. Finally,  $\mathbb{C}_+ := \{s \in \mathbb{C} \mid \operatorname{Re}(s) > 0\}$  is the open right-half of the complex plane.

# 2. Circuit equations

In this section, we briefly describe the systems of DAEs that arise in circuit simulation and review how reduced-order modeling of linear subcircuits is employed in the numerical solution of such systems. For introductions to circuit simulation and overviews of typical simulation tasks, we refer the reader to [10,29,30].

## 2.1. General circuit equations

Electronic circuits are usually modeled as networks whose branches correspond to the circuit elements and whose nodes correspond to the interconnections of the circuit elements; see, e.g.

[10,29,30]. Such networks are characterized by three types of equations: *Kirchhoff's current law* (KCL), *Kirchhoff's voltage law* (KVL), and *branch constitutive relations* (BCRs). The unknowns in these equations are the currents through the branches of the network, the voltage drops along the branches, and the voltages at the nodes of the network. The KCLs and KVLs are linear algebraic equations that only depend on the topology of the circuit. The KCLs state that, at each node  $\mathcal{N}$  of the network, the currents flowing in and out of  $\mathcal{N}$  sum up to zero. The KVLs state that, for each closed loop  $\mathcal{L}$  of the network, the voltage drops along  $\mathcal{L}$  sum up to zero. The BCRs are equations that characterize the actual circuit elements. For example, the BCR of a linear resistor is Ohm's law. The BCRs are linear equations for simple devices, such as linear resistors, capacitors, and inductors, and they are nonlinear equations for more complex devices, such as diodes and transistors. Furthermore, in general, the BCRs involve first time derivatives of the unknowns, and thus they are first-order DAEs.

All the KCLs, KVLs, and BCRs characterizing a given circuit can be summarized as a system of first-order, in general nonlinear, DAEs of the form

$$\boldsymbol{f}(\hat{\boldsymbol{x}},t) + \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{q}(\hat{\boldsymbol{x}},t) = \boldsymbol{0},\tag{1}$$

together with suitable initial conditions. Here,  $\hat{x} = \hat{x}(t)$  is the unknown vector of circuit variables at time t, the vector-valued function  $f(\hat{x}, t)$  represents the contributions of nonreactive elements such as resistors, sources, etc., and the vector-valued function  $(d/dt)q(\hat{x}, t)$  represents the contributions of reactive elements such as capacitors and inductors. There are a number of established methods, such as sparse tableau, nodal formulation, and modified nodal analysis, for writing down the system (1); see, e.g. [30]. The vector functions  $\hat{x}$ , f, q in (1), as well as their dimension,  $\hat{N}$ , depend on the chosen formulation method. The most general method is sparse tableau, which consists of just listing all the KCLs, KVLs, and BCRs. The other formulation methods can be interpreted as starting from sparse tableau and eliminating some of the unknowns by using some of the KCL or KVL equations. For all the standard formulation methods, the dimension  $\hat{N}$  is of the order of the number of devices in the circuit.

# 2.2. Linear subcircuits

Traditional circuit simulators are based on the numerical solution of the system of DAEs (1); see, e.g. [30]. However, the dimension of (1) can be so large that time integration of (1) is inefficient or even prohibitive. On the other hand, circuits often contain large linear subcircuits that can be well approximated by reduced-order models of much smaller dimension. By replacing the equations in (1) corresponding to such linear subcircuits by their respective reduced-order models, one obtains an approximate system of DAEs of much smaller dimension that can then be solved numerically by time integration. We now describe this process in more detail.

Let  $\mathscr{C}_1$  be a large linear subcircuit of a given circuit, and denote by  $\mathscr{C}_r$  the, in general nonlinear, remainder of the circuit. After a suitable reordering, the vector  $\hat{x}$  of circuit variables in (1) can be partitioned as follows:

$$\hat{\boldsymbol{x}} = \begin{bmatrix} \hat{\boldsymbol{x}}_r \\ \boldsymbol{y} \\ \hat{\boldsymbol{x}}_1 \end{bmatrix}.$$
<sup>(2)</sup>

Here,  $\hat{x}_r$  and  $\hat{x}_1$  denote the circuit variables exclusive to  $\mathscr{C}_r$  and  $\mathscr{C}_1$ , respectively, and y represents the variables shared by  $\mathscr{C}_r$  and  $\mathscr{C}_1$ . Using the partitioning (2) and setting

$$\boldsymbol{x}_0 := \begin{bmatrix} \hat{\boldsymbol{x}}_r \\ \boldsymbol{y} \end{bmatrix}$$
 and  $\boldsymbol{x} := \begin{bmatrix} \boldsymbol{y} \\ \hat{\boldsymbol{x}}_l \end{bmatrix}$ , (3)

the functions f and q in (1), after a suitable reordering of the equations in (1), can be expressed as follows:

$$\boldsymbol{f}(\hat{\boldsymbol{x}},t) = \begin{bmatrix} \boldsymbol{f}_0(\boldsymbol{x}_0,t) \\ \boldsymbol{0}_{k\times 1} \end{bmatrix} + \begin{bmatrix} \boldsymbol{0}_{\hat{N}-N\times 1} \\ \boldsymbol{G}\boldsymbol{x} \end{bmatrix}, \quad \boldsymbol{q}(\hat{\boldsymbol{x}},t) = \begin{bmatrix} \boldsymbol{q}_0(\boldsymbol{x}_0,t) \\ \boldsymbol{0}_{k\times 1} \end{bmatrix} + \begin{bmatrix} \boldsymbol{0}_{\hat{N}-N\times 1} \\ \boldsymbol{C}\boldsymbol{x} \end{bmatrix}.$$
(4)

Here,  $f_0$  and  $q_0$  represent the contributions of resistive and reactive elements from the subcircuit  $\mathscr{C}_r$ , and the matrices G and C represent the contributions of resistive and reactive elements in the linear subcircuit  $\mathscr{C}_1$ . In (4), without loss of generality, we have assumed that the vector-valued functions  $f_0$ and  $q_0$  have the same number of components, that the zero vectors below  $f_0$  and  $q_0$  have the same length, k, and that the matrices G and C are square and of the same size,  $N \times N$ ; this can always be achieved by padding  $f_0$ ,  $q_0$ , G, and C with additional zeros, if necessary. Unless the subcircuit  $\mathscr{C}_1$ is completely decoupled form the remainder circuit  $\mathscr{C}_r$ , we have m := N - k > 0. This means that, in (4), the last m components of the, in general nonlinear, functions  $f_0$  and  $q_0$  are connected with the first m components of the linear functions Gx and Cx. By introducing an additional m-dimensional vector, u = u(t), of circuit variables, these m connecting equations can be decoupled. Indeed, using (4), one readily verifies that the original system (1) is equivalent to the following system:

$$f_0(\boldsymbol{x}_0,t) + \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{q}_0(\boldsymbol{x}_0,t) + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{I}_m \end{bmatrix} \boldsymbol{u} = \boldsymbol{0}, \tag{5}$$

$$C\frac{\mathrm{d}x}{\mathrm{d}t} + Gx = \begin{bmatrix} I_m \\ 0 \end{bmatrix} u. \tag{6}$$

We remark that the additional variables u in (5) and (6) can be interpreted as interface signals between the subcircuits  $\mathscr{C}_r$  and  $\mathscr{C}_1$ .

Let p denote the length of the vector y in the partitioning (2) of  $\hat{x}$ , and set

$$\boldsymbol{B} := \begin{bmatrix} \boldsymbol{I}_m \\ \boldsymbol{0}_{N-m \times m} \end{bmatrix} \text{ and } \boldsymbol{L} := \begin{bmatrix} \boldsymbol{I}_p \\ \boldsymbol{0}_{N-p \times p} \end{bmatrix}.$$

Note that, by (3), the matrix  $L^{H} = L^{T}$  selects the subvector y from x, i.e.

$$y = L^{\mathrm{H}}x.$$

Eqs. (6) and (7) constitute a linear dynamical system of the form

$$C\frac{\mathrm{d}x}{\mathrm{d}t} = -Gx + Bu(t),$$
  
$$y(t) = L^{\mathrm{H}}x(t).$$
 (8)

In (8), in general,  $C, G \in \mathbb{C}^{N \times N}, B \in \mathbb{C}^{N \times m}$ , and  $L \in \mathbb{C}^{N \times p}$  are given matrices, *m* and *p* denote the number of inputs and outputs, respectively, the components of the given vector-valued function  $u : [0, \infty) \mapsto \mathbb{C}^m$  are the inputs, and  $y : [0, \infty) \mapsto \mathbb{C}^p$  is the unknown function of outputs. The components of the unknown vector-valued function  $x : [0, \infty) \mapsto \mathbb{C}^N$  are the state variables, and *N*
is the state-space dimension. In general, the matrices C and G in (8) are allowed to be singular. However, we assume that G+sC is a *regular matrix pencil*, i.e., G+sC is singular only for finitely many values of  $s \in \mathbb{C}$ . This condition is always satisfied for linear dynamical systems (8) arising in circuit simulation.

A reduced-order model of (8) is a linear dynamical system of the same form as (8), but of smaller state-space dimension n < N. More precisely, a reduced-order model of state-space dimension n is of the form

$$C_n \frac{\mathrm{d}z}{\mathrm{d}t} = -G_n z + B_n u(t),$$
  

$$v(t) = L_n^{\mathrm{H}} z(t).$$
(9)

Here,  $C_n$ ,  $G_n \in \mathbb{C}^{n \times n}$ ,  $B_n \in \mathbb{C}^{n \times m}$ , and  $L_n \in \mathbb{C}^{n \times p}$  are matrices that should be chosen such that the input-output mapping  $u(t) \mapsto y(t)$  of (9) somehow approximates the input-output mapping of the original system (8); see Section 2.3 below.

After a suitable reduced-order model (9) for systems (6) and (7) has been determined, the linear part (6) of the circuit equations is replaced by the first set of equations in (9). The result is a reduced-order system of DAEs that represents an approximation to the original system (1); see, e.g. [9,26]. Provided that the size of  $\mathscr{C}_1$  dominates that of  $\mathscr{C}_r$ , the approximate system has a much smaller state-space dimension than (1), and thus time integration by means of standard circuit simulators becomes feasible.

## 2.3. Transfer functions

Next, we introduce the so-called transfer function, which describes the input–output behavior of a linear dynamical system (9) in frequency domain.

For vector-valued functions g(t),  $t \in [0, \infty)$ , with g(0) = 0, we denote by

$$\hat{\boldsymbol{g}}(s) = \int_0^\infty \boldsymbol{g}(t) \mathrm{e}^{-st} \, \mathrm{d}t, \quad s \in \mathbb{C},$$
(10)

the (frequency-domain) Laplace transform of g. We remark that in (10), the purely imaginary values  $s = i\omega$ ,  $\omega \ge 0$ , correspond to the frequency  $\omega$ ; these are the physically meaningful values of the complex variable s.

We now assume, for simplicity, zero initial conditions x(0) = 0 and u(0) = 0 in (8). By applying (10) to the linear dynamical system (8), we obtain its frequency-domain formulation

$$sC\hat{x} = -G\hat{x} + B\hat{u}(s),$$
  
$$\hat{y}(s) = L^{H}\hat{x}(s).$$
 (11)

Eliminating  $\hat{x}$  in (11) results in the frequency-domain input-output relation  $\hat{y}(s) = H(s)\hat{u}(s)$ , where H, the *transfer function* of (8), is given by

$$\boldsymbol{H}(s) := \boldsymbol{L}^{\mathrm{H}}(\boldsymbol{G} + s\boldsymbol{C})^{-1}\boldsymbol{B}, \quad s \in \mathbb{C}.$$
(12)

Note that  $H : \mathbb{C} \mapsto (\mathbb{C} \cup \{\infty\})^{p \times m}$  is a matrix-valued rational function.

Similarly, the transfer function  $H_n : \mathbb{C} \mapsto (\mathbb{C} \cup \{\infty\})^{p \times m}$  of the reduced-order model (9) is given by

$$\boldsymbol{H}_{n}(s) := \boldsymbol{L}_{n}^{\mathrm{H}}(\boldsymbol{G}_{n} + s\boldsymbol{C}_{n})^{-1}\boldsymbol{B}_{n}, \quad s \in \mathbb{C}.$$

$$(13)$$

In terms of transfer functions, the problem of constructing a reduced-order model (9) of size n that approximates the input-output behavior of (8) can be stated as follows: Determine the matrices  $C_n$ ,  $G_n$ ,  $B_n$ , and  $L_n$  in (9) such that the reduced-order transfer function (13),  $H_n$ , in some sense approximates the transfer function (12), H, of the original linear dynamical system (8).

For systems (8) of small-to-moderate state-space dimension N, there is a variety of techniques to construct reduced-order models such that, in some appropriate norm,  $H_n$  approximates H over a whole range of values of s; see the references given in [13]. However, these techniques are usually not applicable to large-scale systems (8), such as the ones arising in circuit simulation. In the latter case, the matrices C and G in (8) are large and sparse. Note that, in view of (12), the evaluation of  $H(s_0)$  at even a single point  $s_0 \in \mathbb{C}$  requires the solution of systems of linear equations with the large sparse coefficient matrix  $G + s_0C$ . Fortunately, the circuit matrices C and G are usually such that sparse Gaussian elimination can be employed to compute an LU factorization

$$\boldsymbol{G} + \boldsymbol{s}_0 \boldsymbol{C} = \boldsymbol{P}_1 \boldsymbol{L}_0 \boldsymbol{U}_0 \boldsymbol{P}_2 \tag{14}$$

of the matrix  $G + s_0 C$ . In (14),  $P_1$  and  $P_2$  are permutation matrices that record pivoting for sparsity and numerical stability,  $L_0$  is a lower triangular matrix, and  $U_0$  is an upper triangular matrix. Pivoting for sparsity means that the original ordering of the rows and columns of  $G + s_0 C$  is changed so that potential fill-in in the factors  $L_0$  and  $U_0$  is reduced. For circuit matrices, typically very little fill-in occurs in  $L_0$  and  $U_0$ , although this cannot be guaranteed in general. Once the factorization (14) is computed, the solution of the linear systems needed to evaluate  $H(s_0)$  is obtained by sparse backsolves.

Note that evaluating  $H(s_0)$  at several points  $s_0$  would require the computation of a new factorization (14) for each new point  $s_0$ . Despite the limited fill-in for circuit matrices, the cost for factoring  $G + s_0 C$  is high enough that one tries to get away with computing a single factorization (14). This is the case for reduced-order models that are characterized by a matching of the leading terms in Taylor expansions of H and  $H_n$  about a given expansion point  $s_0$ . More precisely, such a reduced-order model of given size n is defined by

$$\boldsymbol{H}_{n}(s) = \boldsymbol{H}(s) + \mathcal{O}(s - s_{0})^{q(n)}.$$
(15)

If q(n) in (15) is as large as possible, then  $H_n$  is an *nth matrix-Padé approximant* of H; see, e.g. [5]. In Section 5, we will also discuss certain *matrix-Padé-type approximants* for which q(n) is not maximal.

# 2.4. Linear RLC subcircuits

In circuit simulation, an important special case is linear subcircuits that consist of only resistors, inductors, and capacitors. Such linear RLC subcircuits arise in the modeling of a circuit's interconnect and package; see, e.g. [16,17,20,24].

The equations describing linear RLC subcircuits are of the form (8). Furthermore, the equations can be formulated such that the matrices in (8) exhibit certain symmetries; see [15,17]. More precisely, the  $N \times N$  matrices G and C are real and symmetric, and have the following block structure:

$$\boldsymbol{G} = \boldsymbol{G}^{\mathrm{T}} = \begin{bmatrix} \boldsymbol{G}_{11} & \boldsymbol{G}_{12} \\ \boldsymbol{G}_{12}^{\mathrm{T}} & \boldsymbol{0} \end{bmatrix} \quad \text{and} \quad \boldsymbol{C} = \boldsymbol{C}^{\mathrm{T}} = \begin{bmatrix} \boldsymbol{C}_{11} & \boldsymbol{0} \\ \boldsymbol{0} & -\boldsymbol{C}_{22} \end{bmatrix}.$$
(16)

Here, the submatrices  $G_{11}, C_{11} \in \mathbb{R}^{N_1 \times N_1}$  and  $C_{22} \in \mathbb{R}^{N_2 \times N_2}$  are symmetric positive semi-definite, and  $N = N_1 + N_2$ . Note that, except for the special case  $N_2 = 0$ , the matrices G and C are indefinite. The special case  $N_2 = 0$  arises for RC subcircuits that contain only resistors and capacitors, but no inductors.

If the RLC subcircuit is viewed as an *m*-terminal component with m = p inputs and outputs, then the matrices **B** and **L** in (8) are identical and of the form

$$\boldsymbol{B} = \boldsymbol{L} = \begin{bmatrix} \boldsymbol{B}_1 \\ \boldsymbol{0}_{N_2 \times m} \end{bmatrix} \quad \text{with } \boldsymbol{B}_1 \in \mathbb{R}^{N_1 \times m}.$$
(17)

For such an *m*-terminal RLC subcircuit, in view of (16) and (17), the transfer function (12) reduces to

$$\boldsymbol{H}(s) = \boldsymbol{B}^{\mathrm{T}}(\boldsymbol{G} + s\boldsymbol{C})^{-1}\boldsymbol{B} \quad \text{where} \quad \boldsymbol{G} = \boldsymbol{G}^{\mathrm{T}}, \quad \boldsymbol{C} = \boldsymbol{C}^{\mathrm{T}}.$$
(18)

We call a transfer function *H* symmetric if it is of the form (18) with real matrices *G*, *C*, and *B*. For symmetric transfer functions, we will always assume that the expansion point  $s_0$  in (15) is chosen to be real:

$$s_0 \in \mathbb{R}$$
 if **H** is symmetric. (19)

The condition (19) is necessary in order to generate passive reduced-order models of symmetric transfer functions.

We will also use the following nonsymmetric formulation of (18). Let J be the block matrix

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{I}_{N_1} & \boldsymbol{0} \\ \boldsymbol{0} & -\boldsymbol{I}_{N_2} \end{bmatrix}.$$
(20)

Note that, by (17) and (20), we have B = JB. Using this relation, as well as (16), we can rewrite (18) as follows:

$$\boldsymbol{H}(s) = \boldsymbol{B}^{\mathrm{T}} (\boldsymbol{J}\boldsymbol{G} + s\boldsymbol{J}\boldsymbol{C})^{-1} \boldsymbol{B},$$
(21)

where

$$JG = \begin{bmatrix} G_{11} & G_{12} \\ -G_{12}^{\mathrm{T}} & \mathbf{0} \end{bmatrix} \text{ and } JC = \begin{bmatrix} C_{11} & \mathbf{0} \\ \mathbf{0} & C_{22} \end{bmatrix}$$

In this formulation, the matrix JG is no longer symmetric, but now

$$JG + (JG)^{\mathrm{T}} \ge 0$$
 and  $JC \ge 0$ . (22)

#### **3.** Basis vectors for block Krylov subspaces

In this section, we introduce our notion of block Krylov subspaces for multiple starting vectors. We also review variants of the Arnoldi and Lanczos algorithms for generating basis vectors for block Krylov subspaces.

# 3.1. Reduction to one matrix

Let  $s_0 \in \mathbb{C}$  be the expansion point that is to be used in the characterization (15) of the reduced-order transfer function  $H_n$ . The only assumption on  $s_0$  is that the matrix  $G + s_0 C$  be nonsingular; this guarantees that  $s_0$  is not a pole of the original transfer function (12), H.

An approximate transfer function  $H_n$  satisfying (15) could be obtained by first explicitly computing the leading q(n) Taylor coefficients of the expansion of H about  $s_0$  and then generating  $H_n$  from these coefficients; see, e.g. [25]. However, any approach based on explicitly computing the Taylor coefficients of H is inherently numerically unstable; see [8]. A much better alternative is to use block Krylov-subspace methods that obtain the same information as contained in the leading q(n)Taylor coefficients of H, but in a more stable manner.

Before block Krylov-subspace methods can be employed, the two matrices G and C in the definition (12) of H have to be reduced to a single matrix, denoted by A in the sequel. This can be done by rewriting (12) as follows:

$$H(s) = L^{H}(I + (s - s_{0})A)^{-1}R,$$
(23)

where

$$A := (\boldsymbol{G} + s_0 \boldsymbol{C})^{-1} \boldsymbol{C}$$
 and  $\boldsymbol{R} := (\boldsymbol{G} + s_0 \boldsymbol{C})^{-1} \boldsymbol{B}$ .

Although *G* and *C* are sparse matrices, in general, the matrix *A* in (23) is a dense matrix. However, block Krylov-subspace methods involve *A* only in the form of matrix-vector products *Av* and possibly  $A^{\rm H}w$ . To efficiently compute these products, one never needs to form *A* explicitly. Instead, one uses the sparse factorization (14) of  $G + s_0C$ . Each matrix-vector product *Av* then requires one multiplication with the sparse matrix *C* and two backsolves with the sparse triangular matrices  $L_0$  and  $U_0$  from (14). Similarly,  $A^{\rm H}w$  requires one multiplication with *C*<sup>H</sup> and two backsolves with the sparse triangular matrices  $L_0$ .

## 3.2. Block Krylov subspaces

Next, we introduce block Krylov subspaces. The proper definition of these subspaces is necessarily quite involved, and the reader may ask if block Krylov subspaces could not be avoided altogether by using standard Krylov subspaces induced by single vectors instead. For example, one can generate scalar approximations for all the  $p \cdot m$  coefficient functions of the  $p \times m$ -matrix-valued transfer function H via suitable basis vectors for m + p standard Krylov subspaces. However, the resulting approximation is not a matrix-Padé approximant of H, and in fact, one can show that, in order to obtain an approximation of the same quality as the matrix-Padé approximant, at least  $\lfloor (m + p)/2 \rfloor$  times more computational work is required compared to computing a matrix-Padé approximant. Therefore, the use of block Krylov subspaces results in much more efficient reduced-order modeling techniques than those based on standard Krylov subspaces.

Let  $A \in \mathbb{C}^{N \times N}$  be a given  $N \times N$  matrix and

$$\boldsymbol{R} = [\boldsymbol{r}_1 \quad \boldsymbol{r}_2 \quad \cdots \quad \boldsymbol{r}_m] \in \mathbb{C}^{N \times m} \tag{24}$$

be a given matrix of *m* right starting vectors,  $\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_m$ . Before we introduce block Krylov subspaces induced by A and R, we briefly review the standard case m=1 of a single starting vector  $\mathbf{r} = \mathbf{r}_1$ . In this case, the usual *nth Krylov subspace* (induced by A and  $\mathbf{r}$ ) is given by

$$\mathscr{K}_{n}(\boldsymbol{A},\boldsymbol{r}) := \operatorname{span}\{\boldsymbol{r},\boldsymbol{A}\boldsymbol{r},\boldsymbol{A}^{2}\boldsymbol{r},\ldots,\boldsymbol{A}^{n-1}\boldsymbol{r}\}.$$
(25)

Let  $n_0$  be defined as the largest possible integer *n* such that in (25), all the Krylov vectors,  $A^{j-1}r$ ,  $1 \le j \le n-1$ , are linearly independent. Note that  $n_0 \le N$ . By the definition of  $n_0$ , the *n*th Krylov

subspace (25) has dimension *n* if  $1 \le n \le n_0$  and dimension  $n_0$  if  $n > n_0$ . Moreover,  $\mathscr{K}_n(A, \mathbf{r}) = \mathscr{K}_{n_0}(A, \mathbf{r})$  for all  $n > n_0$ . Thus,  $\mathscr{K}_{n_0}(A, \mathbf{r})$  is the largest possible Krylov subspace (induced by *A* and  $\mathbf{r}$ ), and we call the Krylov sequence  $\mathbf{r}, A\mathbf{r}, A^2\mathbf{r}, \dots, A^{n-1}\mathbf{r}$  exhausted if  $n > n_0$ .

In the general case of  $m \ge 1$  starting vectors (24), the situation is more complicated; we refer the reader to the discussion in [1]. The main difficulty is that in contrast to the case m = 1, linear independence of the columns in the block Krylov sequence,

$$\boldsymbol{R}, \boldsymbol{A}\boldsymbol{R}, \boldsymbol{A}^{2}\boldsymbol{R}, \dots, \boldsymbol{A}^{j-1}\boldsymbol{R}, \dots,$$
<sup>(26)</sup>

is lost only gradually in general. More precisely, if the *j*th block,  $A^{j-1}R$ , contains a column that is linearly dependent on columns to its left in (26), then, in general, not all the columns of  $A^{j-1}R$ are linear dependent on columns to their left. Hence, the occurrence of a linear-dependent column does not necessarily imply that the block Krylov sequence  $R, AR, A^2R, \ldots, A^{j-1}R$  is exhausted. As a result, in general, the construction of suitable basis vectors for the subspaces spanned by the columns of (26) needs to be continued even after a linear-dependent column has been found in (26). However, a proper handling of such linear-dependent columns requires that the column itself and all its successive *A*-multiples need to be deleted. Formally, this can be done as follows. By scanning the columns of the matrices in (26) from left to right and deleting each column that is linearly dependent on earlier columns, we obtain the *deflated* block Krylov sequence

$$\boldsymbol{R}_{1}, \boldsymbol{A}\boldsymbol{R}_{2}, \boldsymbol{A}^{2}\boldsymbol{R}_{3}, \dots, \boldsymbol{A}^{j_{\max}-1}\boldsymbol{R}_{j_{\max}}.$$
(27)

This process of deleting linearly dependent vectors is referred to as *exact deflation* in the following. In (27), for each  $j = 1, 2, ..., j_{max}$ ,  $\mathbf{R}_j$  is a submatrix of  $\mathbf{R}_{j-1}$ , with  $\mathbf{R}_j \neq \mathbf{R}_{j-1}$  if, and only if, deflation occurred within the *j*th Krylov block  $A^{j-1}\mathbf{R}$  in (26). Here, for j = 1, we set  $\mathbf{R}_0 = \mathbf{R}$ . Denoting by  $m_j$  the number of columns of  $\mathbf{R}_j$ , we thus have

$$m \ge m_1 \ge m_2 \ge \dots \ge m_{j_{\text{max}}} \ge 1.$$
<sup>(28)</sup>

By construction, the columns of the matrices (27) are linearly independent, and for each *n*, the subspace spanned by the first *n* of these columns is called the *nth block Krylov subspace* (induced by *A* and *R*). In the following, we denote the *n*th block Krylov subspace by  $\mathscr{K}_n(A, R)$ . For later use, we remark that for

$$n = m_1 + m_2 + \dots + m_j \quad \text{where } 1 \leq j \leq j_{\text{max}}, \tag{29}$$

the *n*th block Krylov subspace is given by

$$\mathscr{K}_{n}(\boldsymbol{A},\boldsymbol{R}) = \operatorname{Colspan}\{\boldsymbol{R}_{1},\boldsymbol{A}\boldsymbol{R}_{2},\boldsymbol{A}^{2}\boldsymbol{R}_{3},\ldots,\boldsymbol{A}^{j-1}\boldsymbol{R}_{j}\}.$$
(30)

For Lanczos-based reduced-order modeling techniques, we will also need the block Krylov subspaces induced by  $A^{H}$  and a given matrix of *p* left starting vectors,

$$\boldsymbol{L} = [\boldsymbol{l}_1 \quad \boldsymbol{l}_2 \quad \cdots \quad \boldsymbol{l}_p] \in \mathbb{C}^{N \times p}. \tag{31}$$

Applying the above construction to  $A^{H}$  and L, the *nth block Krylov subspace* (induced by  $A^{H}$  and L),  $\mathscr{H}_{n}(A^{H}, L)$ , is defined as the subspace spanned by the first *n* columns of the *deflated* block Krylov sequence

$$\boldsymbol{L}_{1}, \boldsymbol{A}^{\mathrm{H}} \boldsymbol{L}_{2}, (\boldsymbol{A}^{\mathrm{H}})^{2} \boldsymbol{L}_{3}, \dots, (\boldsymbol{A}^{\mathrm{H}})^{k_{\mathrm{max}}-1} \boldsymbol{L}_{k_{\mathrm{max}}}.$$
(32)

Denoting by  $p_k$  the number of columns of  $L_k$ , we have

$$p \ge p_1 \ge p_2 \ge \cdots \ge p_{k_{\max}} \ge 1. \tag{33}$$

Note that for

$$n = p_1 + p_2 + \dots + p_k$$
 where  $1 \leq k \leq k_{\text{max}}$ ,

we have

$$\mathscr{K}_n(\mathbf{A}^{\mathrm{H}}, \mathbf{L}) = \mathrm{Colspan}\{\mathbf{L}_1, \mathbf{A}^{\mathrm{H}}\mathbf{L}_2, (\mathbf{A}^{\mathrm{H}})^2\mathbf{L}_3, \dots, (\mathbf{A}^{\mathrm{H}})^{k-1}\mathbf{L}_k\}.$$

We stress that in our construction of block Krylov subspaces, we only have used exact deflation of columns that are linearly dependent. In an actual algorithm for constructing basis vectors for  $\mathscr{K}_n(A, R)$  and  $\mathscr{K}_n(A^{\mathrm{H}}, L)$  in finite-precision arithmetic, one also needs to delete vectors that are in some sense "almost" linearly dependent on earlier vectors. We will refer to the deletion of such almost linearly dependent vectors as *inexact deflation*. In Sections 3.4 and 3.5 below, we describe how exact and inexact deflation can be built easily into Arnoldi- and Lanczos-type algorithms for multiple starting vectors. While inexact deflation is crucial in practice, concise statements of theoretical results are much simpler if only exact deflation is performed. Throughout this paper, theoretical results are thus formulated for exact deflation only.

For later use, we note the following invariance property of the block Krylov subspaces  $\mathscr{K}_n(A, \mathbf{R})$  induced by the matrices A and  $\mathbf{R}$  defined in (23).

**Lemma 1.** Let G, C, B be the matrix triplet used in the definition of the matrices A and R in (23), and let J be any nonsingular matrix of the same size as A. Then the matrix triplets G, C, B and JG, JC, JB lead to the same nth block Krylov subspace  $\mathcal{K}_n(A, R)$ .

**Proof.** By (23), we have

$$A = (G + s_0C)^{-1}C = (JG + s_0JC)^{-1}JC,$$
  
$$R = (G + s_0C)^{-1}B = (JG + s_0JC)^{-1}JB.$$

Thus both matrix triplets result in the same matrices A and R and the associated block Krylov subspaces are identical.  $\Box$ 

## 3.3. Basis vectors

The columns of the deflated block Krylov sequences (27) and (32), which are used in the above definitions of  $\mathscr{K}_n(A, \mathbf{R})$  and  $\mathscr{K}_n(A^{\mathrm{H}}, \mathbf{L})$ , respectively, tend to be almost linearly dependent even for moderate values of *n*. Therefore, they should not be used in numerical computations. Instead, we construct other suitable basis vectors.

In the following,

$$\boldsymbol{v}_1, \boldsymbol{v}_2, \dots, \boldsymbol{v}_n \in \mathbb{C}^N \tag{34}$$

denotes a set of basis vectors for  $\mathscr{K}_n(A, \mathbf{R})$ , i.e.,

$$\operatorname{span}\{\boldsymbol{v}_1,\boldsymbol{v}_2,\ldots,\boldsymbol{v}_n\}=\mathscr{K}_n(\boldsymbol{A},\boldsymbol{R})$$

The  $N \times n$  matrix

$$\boldsymbol{V}_n := \begin{bmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 & \cdots & \boldsymbol{v}_n \end{bmatrix}$$
(35)

whose columns are the basis vectors (34) is called a *basis matrix* for  $\mathcal{K}_n(A, \mathbf{R})$ . Similarly,

$$w_1, w_2, \dots, w_n \in \mathbb{C}^N \tag{36}$$

denotes a set of basis vectors for  $\mathscr{K}_n(A^{\mathrm{H}}, L)$ , i.e.,

$$\operatorname{span}\{w_1, w_2, \ldots, w_n\} = \mathscr{K}_n(A^{\mathrm{H}}, \boldsymbol{L}).$$

The  $N \times n$  matrix

$$W_n := \begin{bmatrix} w_1 & w_2 & \cdots & w_n \end{bmatrix}$$
(37)

is called a *basis matrix* for  $\mathscr{K}_n(A^{\mathrm{H}}, L)$ .

We stress that even though (34) and (36) are basis vectors of block Krylov subspaces, the algorithms discussed in this paper generate (34) and (36) in a vector-wise fashion, as opposed to the block-wise construction employed in more traditional block Krylov-subspace methods; see, e.g. the references given in [1]. There are two main reasons why the vector-wise construction is preferable to a block-wise construction. First, it greatly simplifies both the detection of necessary deflation and the actual deflation itself. In fact, all that is required is checking if a suitable candidate vector for the next basis vector is the zero vector (for exact deflation) or close to the zero vector (for inexact deflation). Second, for Lanczos-type methods, which simultaneously construct basis vectors (34) and (36) for  $\mathscr{H}_n(A, \mathbf{R})$  and  $\mathscr{H}_n(A^H, \mathbf{L})$ , respectively, only the vector-wise construction allows the treatment of the general case where the block sizes (28) and (33) of the deflated block Krylov sequences (27) and (32) are not necessarily the same; for a detailed discussion, we refer the reader to [1].

# 3.4. Arnoldi basis

The classical Arnoldi process [3] generates orthonormal basis vectors for the sequence of Krylov subspaces  $\mathscr{K}_n(A, \mathbf{r})$ ,  $n \ge 1$ , induced by A and a single starting vector  $\mathbf{r}$ . In this subsection, we state an Arnoldi-type algorithm that extends the classical algorithm to block-Krylov subspaces  $\mathscr{K}_n(A, \mathbf{R})$ ,  $n \ge 1$ .

Like the classical Arnoldi process, the algorithm constructs the basis vectors (34) to be orthonormal. In terms of the basis matrix (35), this orthonormality can be expressed as follows:

$$V_n^{\mathrm{H}} V_n = I$$

In addition to (34), the algorithm produces the so-called *candidate vectors*,

$$\hat{v}_{n+1}, \hat{v}_{n+2}, \dots, \hat{v}_{n+m_c},$$
(38)

for the next  $m_c$  basis vectors  $v_{n+1}, v_{n+2}, \ldots, v_{n+m_c}$ . Here,  $m_c = m_c(n)$  is the number *m* of columns in the starting block (24), **R**, reduced by the number of exact and inexact deflations that have occurred so far. The candidate vectors (38) satisfy the orthogonality relation

$$V_n^{\mathrm{H}}[\hat{v}_{n+1} \quad \hat{v}_{n+2} \quad \cdots \quad \hat{v}_{n+m_{\mathrm{c}}}] = \mathbf{0}.$$

Due to the vector-wise construction of (34) and (38), detection of necessary deflation and the actual deflation becomes trivial. In fact, essentially the same proof as given in [1] for the case of a Lanczos-type algorithm can be used to show that exact deflation at step *n* of the Arnoldi-type process occurs if, and only if,  $\hat{v}_n = 0$ . Similarly, inexact deflation occurs if, and only if,  $\|\hat{v}_n\| \approx 0$ , but  $\hat{v}_n \neq 0$ . Therefore, in the algorithm, one checks if

$$\|\hat{\boldsymbol{v}}_n\| \leqslant \text{dtol},\tag{39}$$

where  $dtol \ge 0$  is a suitably chosen deflation tolerance. If (39) is satisfied, then  $\hat{v}_n$  is deflated by deleting  $\hat{v}_n$ , shifting the indices of all remaining candidate vectors by -1, and setting  $m_c = m_c - 1$ . If this results in  $m_c = 0$ , then the block-Krylov subspace is exhausted and the algorithm is stopped. Otherwise, the deflation procedure is repeated until a vector  $\hat{v}_n$  with  $||\hat{v}_n|| > dtol$  is found. This vector is then turned into  $v_n$  by normalizing it to have Euclidean norm 1.

A complete statement of the resulting Arnoldi-type algorithm is as follows.

Algorithm 1 (Construction of Arnoldi basis for  $\mathscr{K}_n(A, \mathbf{R})$ .).

(0) Set  $\hat{\boldsymbol{v}}_i = \boldsymbol{r}_i$  for  $i = 1, 2, \dots, m$ . Set  $m_c = m$ .

For n = 1, 2, ..., do:

(1) Compute || v̂<sub>n</sub> || and check if the deflation criterion (39) is fulfilled. If yes, v̂<sub>n</sub> is deflated by doing the following: Set m<sub>c</sub> = m<sub>c</sub> − 1. If m<sub>c</sub> = 0, set n = n − 1 and stop. Set v̂<sub>i</sub> = v̂<sub>i+1</sub> for i = n, n + 1,..., n + m<sub>c</sub> − 1. Return to Step (1).

(2) Set 
$$t_{n,n-m_c} = \|\hat{\boldsymbol{v}}_n\|$$
 and  $\boldsymbol{v}_n = \hat{\boldsymbol{v}}_n/t_{n,n-m_c}$ .

- (3) Compute  $\hat{\boldsymbol{v}}_{n+m_c} = A\boldsymbol{v}_n$ .
- (4) For i = 1, 2, ..., n do: Set  $t_{i,n} = \mathbf{v}_i^{\mathrm{H}} \hat{\mathbf{v}}_{n+m_{\mathrm{c}}}$  and  $\hat{\mathbf{v}}_{n+m_{\mathrm{c}}} = \hat{\mathbf{v}}_{n+m_{\mathrm{c}}} - \mathbf{v}_i t_{i,n}$ .
- (5) For  $i = n m_{c} + 1, n m_{c} + 2, \dots, n 1$  do: Set  $t_{n,i} = \mathbf{v}_{n}^{H} \hat{\mathbf{v}}_{i+m_{c}}$  and  $\hat{\mathbf{v}}_{i+m_{c}} = \hat{\mathbf{v}}_{i+m_{c}} - \mathbf{v}_{n} t_{n,i}$ .

**Remark 2.** If dtol = 0 in (39), then Algorithm 1 performs only exact deflation.

**Remark 3.** Other block-Arnoldi algorithms (all without deflation though) can be found in [28, Section 6.12].

After *n* passes through the main loop, Algorithm 1 has constructed the first *n* basis vectors (34) and the candidate vectors (38) for the next  $m_c$  basis vectors. In terms of the basis matrix (35),  $V_n$ , the recurrences used to generate all these vectors can be written compactly in matrix format. To this end, we collect the recurrence coefficients computed during the first  $n = m_1$  and  $n \ge 1$  passes through the main loop of Algorithm 1 in the matrices

$$\rho := [t_{i,l-m}]_{\substack{i=1,2,\dots,m\\l=1,2,\dots,m}} \quad \text{and} \quad T_n := [t_{i,l}]_{\substack{i=1,2,\dots,n\\l=1,2,\dots,n}},$$
(40)

respectively. Here,  $m_1$  is the integer given by (27) and (28). Moreover, in (40), all elements  $t_{i,l-m}$  and  $t_{i,l}$  that are not explicitly defined in Algorithm 1 are set to be zero. The compact statement of the recurrences used in Algorithm 1 is now as follows. For  $n \ge 1$ , we have

$$AV_n = V_n T_n + \begin{bmatrix} \mathbf{0} & \cdots & \mathbf{0} & \hat{\mathbf{v}}_{n+1} & \hat{\mathbf{v}}_{n+2} & \cdots & \hat{\mathbf{v}}_{n+m_c} \end{bmatrix}.$$
(41)

Furthermore, for  $n = m_1$ , we have the relation

$$\boldsymbol{R} = \boldsymbol{V}_{m_1} \boldsymbol{\rho},\tag{42}$$

which reflects the fact that the first  $m_1$  basis vectors are obtained by orthonormalizing the columns of the matrix (24), **R**. In (41) and (42), we assumed that only exact deflations are performed. If both exact and inexact deflations are performed, then an additional matrix term, say  $\hat{V}_n^{\text{defl}}$ , appears on the right-hand side of (41), respectively (42). The only non-zero columns of  $\hat{V}_n^{\text{defl}}$  are those non-zero vectors that satisfied the deflation check (39). Since at any stage of Algorithm 1, at most  $m - m_c = m - m_c(n)$  vectors have been deflated, the additional matrix term is small in the sense that

$$\|\hat{V}_n^{ ext{defl}}\| \! \leqslant \! ext{dtol} \sqrt{m - m_{ ext{c}}(n)}$$

# 3.5. Lanczos basis

The classical Lanczos process [21] generates two sequences of basis vectors (34) and (36) that span the Krylov subspaces  $\mathscr{K}_n(A, \mathbf{r})$  and  $\mathscr{K}_n(A^H, \mathbf{l})$ , respectively, where  $\mathbf{r}$  and  $\mathbf{l}$  are single starting vectors. In [1], a Lanczos-type method was presented that extends the classical algorithm to block-Krylov subspaces  $\mathscr{K}_n(A, \mathbf{R})$  and  $\mathscr{K}_n(A^H, \mathbf{L})$  with blocks  $\mathbf{R}$  and  $\mathbf{L}$  of multiple right and left starting vectors (24) and (31). Such a Lanczos-type method is necessarily quite involved, and in order to keep this paper reasonably short, here we only state the governing equations that underlie the algorithm. For a complete listing of the actual algorithm, we refer the reader to [13, Algorithm 9.2].

Like the classical Lanczos process, the extended algorithm constructs the basis vectors (34) and (36) to be biorthogonal. In terms of the associated basis matrices (35) and (37), the biorthogonality can be expressed as follows:

$$\boldsymbol{W}_{n}^{\mathrm{H}}\boldsymbol{V}_{n} = \boldsymbol{\varDelta}_{n} := \mathrm{diag}(\delta_{1}, \delta_{2}, \dots \delta_{n}). \tag{43}$$

Here, for simplicity, we have assumed that no look-ahead steps are necessary. This implies that  $\Delta_n$  is a diagonal matrix, as stated in (43), and that all diagonal entries of  $\Delta_n$  are nonzero. If look-ahead steps occur, then  $\Delta_n$  is a block-diagonal matrix; see [1] for further details. In addition to (34) and (36), the algorithm constructs the candidate vectors

$$\hat{v}_{n+1}, \hat{v}_{n+2}, \dots, \hat{v}_{n+m_c}$$
 and  $\hat{w}_{n+1}, \hat{w}_{n+2}, \dots, \hat{w}_{n+p_c}$  (44)

for the next  $m_c$  basis vectors  $v_{n+1}, v_{n+2}, \ldots, v_{n+m_c}$  and the next  $p_c$  basis vectors  $w_{n+1}, w_{n+2}, \ldots, w_{n+p_c}$ , respectively. Here, as in Section 3.4,  $m_c = m_c(n)$  is the number m of columns in the right starting block (24), R, reduced by the number of exact and inexact deflations of candidate vectors  $\hat{v}_n$  that have occurred so far. Analogously,  $p_c = p_c(n)$  is the number p of columns in the left starting block (31), L, reduced by the number of exact and inexact deflations of candidate vectors  $\hat{w}_n$  that have

occurred so far. Similar to (39) a deflation of the candidate vector  $\hat{v}_n$ , respectively  $\hat{w}_n$ , is performed if, and only if,

$$\|\hat{v}_n\| \leq dtol, \text{ respectively } \|\hat{w}_n\| \leq dtol,$$
 (45)

where  $dtol \ge 0$  is a suitably chosen deflation tolerance. If dtol = 0, then (45) only checks for exact deflation.

The candidate vectors (44) are constructed to satisfy the following biorthogonality relations:

$$W_n^{\rm H}[\hat{v}_{n+1} \quad \hat{v}_{n+2} \quad \cdots \quad \hat{v}_{n+m_c}] = \mathbf{0},$$

$$V_n^{\rm H}[\hat{w}_{n+1} \quad \hat{w}_{n+2} \quad \cdots \quad \hat{w}_{n+p_c}] = \mathbf{0}.$$
(46)

The actual recurrences used to generate the basis vectors (34) and (36) and the candidate vectors (44) can be summarized compactly in matrix form. For simplicity, we again assume that only exact deflation is performed. Then, for  $n \ge 1$ , we have

$$AV_{n} = V_{n}T_{n} + [0 \cdots 0 \hat{v}_{n+1} \hat{v}_{n+2} \cdots \hat{v}_{n+m_{c}}],$$
  

$$A^{H}W_{n} = W_{n}\tilde{T}_{n} + [0 \cdots 0 \hat{w}_{n+1} \hat{w}_{n+2} \cdots \hat{w}_{n+p_{c}}].$$
(47)

Moreover, for  $n = m_1$ , respectively  $n = p_1$ , we have the relations

$$V_{m_1}\rho = R$$
, respectively  $W_{p_1}\eta = L$ , (48)

which summarize the recurrences for processing the starting blocks R and L. We note that the matrices  $T_n$  and  $\tilde{T}_n$  in (47) essentially encode the same information. In fact, by pre-multiplying the first and second relation in (47) by  $W_n$  and  $V_n$  and by using (43) and (46), it follows that

$$\boldsymbol{W}_{n}^{\mathrm{H}}\boldsymbol{A}\boldsymbol{V}_{n} = \boldsymbol{\varDelta}_{n}\boldsymbol{T}_{n} = \boldsymbol{\tilde{T}}_{n}^{\mathrm{H}}\boldsymbol{\varDelta}_{n}. \tag{49}$$

In particular, (49) implies that  $\tilde{T}_n^{\rm H} = \Delta_n T_n \Delta_n^{-1}$ .

Finally, we note that for symmetric transfer functions (18), such as the ones describing RLC subcircuits, the Lanczos-type method sketched in this section can exploit the symmetry inherent in (18). Indeed, in this case, the Lanczos basis vectors (34) and (36) are connected as follows:

$$\boldsymbol{w}_n = \gamma_n (\boldsymbol{G} + s_0 \boldsymbol{C}) \boldsymbol{v}_n \quad \text{for all } n.$$

Here,  $\gamma_n \neq 0$  are suitable normalization factors. In view of (50), only the vectors (34) need to be generated. This results in a *symmetric* variant of the Lanczos-type method that requires only half the computational work and storage of the general case; see [15–17] for further details. For later use, we note that for symmetric transfer functions (18), the coefficient matrices in (48) can be chosen to be identical:

$$\boldsymbol{\rho} = \boldsymbol{\eta} \in \mathbb{R}^{m_1 \times m}.\tag{51}$$

In fact, by (23), (18), and (19),  $(\boldsymbol{G} + s_0 \boldsymbol{C})\boldsymbol{R} = \boldsymbol{B} = \boldsymbol{L}$  and all these matrices are real. In view of (48) and (50), this implies (51).

## 4. Reduced-order models based on projection

In this section, we introduce two reduced-order models based on a one-sided projection onto  $\mathscr{K}_n(A, \mathbf{R})$ , respectively a two-sided projection onto  $\mathscr{K}_n(A, \mathbf{R})$  and  $\mathscr{K}_n(A^{\mathrm{H}}, \mathbf{L})$ .

## 4.1. Projection onto $\mathscr{K}_n(A, \mathbf{R})$

Let  $V_n \in \mathbb{C}^{N \times n}$  be a given matrix with full column rank *n*. By simply projecting the matrices *G*, *C*, *B*, and *L* in the original linear dynamical system (8) onto the subspace spanned by the columns of  $V_n$ , we obtain a reduced-order model (9) with matrices  $G_n$ ,  $C_n$ ,  $B_n$ , and  $L_n$  given by

$$\boldsymbol{G}_{n} := \boldsymbol{V}_{n}^{\mathrm{H}} \boldsymbol{G} \boldsymbol{V}_{n}, \quad \boldsymbol{C}_{n} := \boldsymbol{V}_{n}^{\mathrm{H}} \boldsymbol{C} \boldsymbol{V}_{n}, \quad \boldsymbol{B}_{n} := \boldsymbol{V}_{n}^{\mathrm{H}} \boldsymbol{B}, \quad \boldsymbol{L}_{n} := \boldsymbol{V}_{n}^{\mathrm{H}} \boldsymbol{L}.$$
(52)

In particular, we now assume that  $V_n$  is a basis matrix (35) for  $\mathcal{H}_n(A, \mathbf{R})$ . In this case, the reduced-order model defined by (9) and (35) represents a (one-sided) projection of the original system (8) onto the *n*th block-Krylov subspace  $\mathcal{H}_n(A, \mathbf{R})$ . In the sequel, we denote the associated reduced-order transfer function by

$$\boldsymbol{H}_{n}^{(1)}(s) := \boldsymbol{L}_{n}^{\mathrm{H}}(\boldsymbol{G}_{n} + s\boldsymbol{C}_{n})^{-1}\boldsymbol{B}_{n},$$
(53)

where the superscript <sup>(1)</sup> indicates the one-sided projection. In Section 5.1 below, we show that  $H_n^{(1)}$  is a certain Padé-type approximant of the original transfer function H.

The following proposition shows that  $H_n^{(1)}$  is independent of the actual choice of the basis matrix  $V_n$  for  $\mathcal{K}_n(A, \mathbf{R})$ .

**Proposition 4.** The reduced-order transfer function  $H_n^{(1)}$  given by (52) and (53) does not depend on the particular choice of the basis matrix (35),  $V_n$ , for the nth block Krylov subspace  $\mathcal{K}_n(A, \mathbf{R})$ .

**Proof.** Let  $V_n$  be the basis matrix for  $\mathscr{K}_n(A, \mathbf{R})$  that is used in (52). Let  $\tilde{V}_n$  be any other basis matrix for  $\mathscr{K}_n(A, \mathbf{R})$ . In analogy to (52) and (53),  $\tilde{V}_n$  induces the reduced-order transfer function

$$\tilde{\boldsymbol{H}}_{n}(s) = \tilde{\boldsymbol{L}}_{n}^{\mathrm{H}} (\tilde{\boldsymbol{G}}_{n} + s\tilde{\boldsymbol{C}}_{n})^{-1} \tilde{\boldsymbol{B}}_{n},$$
(54)

where

$$\tilde{\boldsymbol{G}}_n = \tilde{\boldsymbol{V}}_n^{\mathrm{H}} \boldsymbol{G} \tilde{\boldsymbol{V}}_n, \quad \tilde{\boldsymbol{C}}_n = \tilde{\boldsymbol{V}}_n^{\mathrm{H}} \boldsymbol{C} \tilde{\boldsymbol{V}}_n, \quad \tilde{\boldsymbol{B}}_n = \tilde{\boldsymbol{V}}_n^{\mathrm{H}} \boldsymbol{B}, \quad \tilde{\boldsymbol{L}}_n = \tilde{\boldsymbol{V}}_n^{\mathrm{H}} \boldsymbol{L}.$$

Since  $V_n$  and  $\tilde{V}_n$  are basis matrices for the same subspace, there exists a nonsingular  $n \times n$  matrix M such that  $\tilde{V}_n = V_n M$ . Using this relation, we obtain from (54) and (52) that

$$\tilde{\boldsymbol{G}}_{n} = \boldsymbol{M}^{\mathrm{H}}\boldsymbol{G}_{n}\boldsymbol{M}, \quad \tilde{\boldsymbol{C}}_{n} = \boldsymbol{M}^{\mathrm{H}}\boldsymbol{C}_{n}\boldsymbol{M}, \quad \tilde{\boldsymbol{B}}_{n} = \boldsymbol{M}^{\mathrm{H}}\boldsymbol{B}_{n}, \quad \tilde{\boldsymbol{L}}_{n} = \boldsymbol{M}^{\mathrm{H}}\boldsymbol{L}_{n}.$$
(55)

Inserting (55) into (54), we get

$$\tilde{\boldsymbol{H}}_{n}(s) = \boldsymbol{L}_{n}^{\mathrm{H}}\boldsymbol{M}(\boldsymbol{M}^{\mathrm{H}}(\boldsymbol{G}_{n}+s\boldsymbol{C}_{n})\boldsymbol{M})^{-1}\boldsymbol{M}^{\mathrm{H}}\boldsymbol{B}_{n}$$
$$= \boldsymbol{L}_{n}^{\mathrm{H}}(\boldsymbol{G}_{n}+s\boldsymbol{C}_{n})^{-1}\boldsymbol{B}_{n} = \boldsymbol{H}_{n}^{(1)}(s).$$

Thus, the reduced-order transfer functions  $H_n^{(1)}$  and  $\tilde{H}_n$  are identical.  $\Box$ 

# 4.2. Two-sided projection onto $\mathscr{K}_n(A, \mathbf{R})$ and $\mathscr{K}_n(A^{\mathrm{H}}, \mathbf{L})$

Let  $V_n$  and  $W_n$  be any two basis matrices of  $\mathscr{K}_n(A, \mathbf{R})$  and  $\mathscr{K}_n(A^H, \mathbf{L})$ , and consider the representation (23) of the transfer function H of (8). By projecting the matrices in (23) from the right and left onto the columns of  $V_n$  and  $W_n$ , respectively, we obtain the reduced-order transfer function

$$\boldsymbol{H}_{n}^{(2)}(s) := (\boldsymbol{V}_{n}^{\mathrm{H}}\boldsymbol{L})^{\mathrm{H}}(\boldsymbol{W}_{n}^{\mathrm{H}}\boldsymbol{V}_{n} + (s - s_{0})\boldsymbol{W}_{n}^{\mathrm{H}}\boldsymbol{A}\boldsymbol{V}_{n})^{-1}(\boldsymbol{W}_{n}^{\mathrm{H}}\boldsymbol{R}).$$
(56)

In analogy to Proposition 4, we have the following result.

**Proposition 5.** The reduced-order transfer function  $H_n^{(2)}$  given by (56) does not depend on the particular choice of the basis matrices  $V_n$  and  $W_n$  for the nth block Krylov subspaces  $\mathcal{K}_n(A, \mathbf{R})$  and  $\mathcal{K}_n(A^{\mathrm{H}}, \mathbf{L})$ .

**Proof.** Analogous to the proof of Proposition 4.  $\Box$ 

#### 4.3. Computation via Krylov-subspace algorithms

In view of Proposition 5, the reduced-order transfer function (56),  $H_n^{(2)}$ , can be generated from any two basis matrices  $V_n$  and  $W_n$ . However, there is one distinguished choice of  $V_n$  and  $W_n$  that eliminates the need to explicitly compute the projections in (56). This choice is the Lanczos basis described in Section 3.5.

Indeed, let  $n \ge \max\{m_1, p_1\}$ , and assume that the Lanczos-type algorithm is run for *n* steps. Let  $\Delta_n$  be the diagonal matrix defined in (43), and let  $T_n$ ,  $\rho$ , and  $\eta$  be the matrices of recurrence coefficients given by (47) and (48). Then, from (43) and (48), it readily follows that

$$W_n^{\rm H} \boldsymbol{R} = \boldsymbol{\Delta}_n \boldsymbol{\rho}_n \quad \text{and} \quad V_n^{\rm H} \boldsymbol{L} = \boldsymbol{\Delta}_n^{\rm H} \boldsymbol{\eta}_n,$$
(57)

where

$$\boldsymbol{\rho}_{n} := \begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{0}_{n-m_{1} \times m} \end{bmatrix} \quad \text{and} \quad \boldsymbol{\eta}_{n} := \begin{bmatrix} \boldsymbol{\eta} \\ \boldsymbol{0}_{n-p_{1} \times p} \end{bmatrix}.$$
(58)

Furthermore, by multiplying the first relation in (47) from the left by  $W_n^H$  and using the first relation in (46), as well as (43), we get

$$W_n^{\mathrm{H}}AV_n = \varDelta_n T_n \quad \text{and} \quad W_n^{\mathrm{H}}V_n = \varDelta_n.$$
 (59)

Inserting (57) and (59) into (56), it readily follows that

$$\boldsymbol{H}_{n}^{(2)}(s) = \boldsymbol{\eta}_{n}^{\mathrm{H}}(\boldsymbol{\varDelta}_{n}^{-1} + (s - s_{0})\boldsymbol{T}_{n}\boldsymbol{\varDelta}_{n}^{-1})^{-1}\boldsymbol{\rho}_{n}.$$
(60)

The MPVL (matrix-Padé via Lanczos) algorithm, which was first proposed in [9], is a numerical procedure for computing  $H_n^{(2)}$  via the formula (60).

For symmetric transfer functions (18), by (51) and (58), the reduced-order transfer function (60) is also symmetric:

$$\boldsymbol{H}_{n}^{(2)}(s) = \boldsymbol{\rho}_{n}^{\mathrm{T}}(\boldsymbol{\Delta}_{n}^{-1} + (s - s_{0})\boldsymbol{T}_{n}\boldsymbol{\Delta}_{n}^{-1})^{-1}\boldsymbol{\rho}_{n},$$
(61)

where  $\Delta_n^{-1}$  and  $T_n \Delta_n^{-1}$  are real symmetric matrices. The SyMPVL algorithm [16,17] is a special symmetric variant of the general MPVL algorithm that computes symmetric reduced-order transfer functions (61).

Furthermore, recall from Section 2.4 that RLC subcircuits are described by special symmetric transfer functions (18) with matrices G, C, and B of the form (16) and (17). In this case, as we will discuss in Section 6, the reduced-order transfer function (60) in general does not preserve the passivity of the RLC subcircuit. However, one can easily augment the SyMPVL algorithm to generate a second projected reduced-order model that, by Corollary 14 below, is always passive. To this end, let J be the matrix defined in (20), and consider the nonsymmetric formulation (21) of the symmetric transfer function (18). Note that by Lemma 1, both formulations (18) and (21) result in the same *n*th block Krylov subspace  $\mathcal{H}_n(A, R)$ . In particular, the Lanczos basis matrix  $V_n$ 

generated by SyMPVL is also a basis matrix for the *n*th block Krylov subspace associated with the nonsymmetric formulation (21). Hence we can also use  $V_n$  to apply the one-sided projection of Section 4.1 to (21). By (21), (52), and (53), the resulting projected reduced-order transfer function is given by

$$\boldsymbol{H}_{n}^{(1)}(s) := (\boldsymbol{V}_{n}^{\mathrm{T}}\boldsymbol{B})^{\mathrm{T}}(\boldsymbol{V}_{n}^{\mathrm{T}}(\boldsymbol{J}\boldsymbol{G})\boldsymbol{V}_{n} + s\boldsymbol{V}_{n}^{\mathrm{T}}(\boldsymbol{J}\boldsymbol{C})\boldsymbol{V}_{n})^{-1}(\boldsymbol{V}_{n}^{\mathrm{T}}\boldsymbol{B}).$$

$$(62)$$

#### 5. Connections with Padé-type and Padé approximants

In this section, we show that the one-sided projection  $H_n^{(1)}$  is actually a matrix-Padé-type approximant of H, and we review the matrix-Padé property of  $H_n^{(2)}$ .

# 5.1. $H_n^{(1)}$ is a matrix-Padé-type approximant

Although the reduced-order transfer function (53),  $H_n^{(1)}$ , is defined via the simple one-sided projection (52), it satisfies an approximation property of the form (15), where, however, q(n) is not maximal in general. This means that  $H_n^{(1)}$  is a matrix-Padé-type approximant of H. For the special case of expansion point  $s_0 = 0$  and a basis matrix  $V_n$  generated by a simple block Arnoldi procedure without deflation, it was first observed in [22,23] that  $H_n^{(1)}$  satisfies an approximation property (15). Here, we extend this result to the most general case of arbitrary expansion points  $s_0$  and arbitrary basis matrices  $V_n$  for the properly defined block Krylov subspaces  $\mathscr{K}_n(A, R)$  that allow for necessary deflation of linearly dependent vectors. The only further assumption we need is that the matrix

$$G_n + s_0 C_n$$
 is nonsingular. (63)

This guarantees that  $s_0$  is not a pole of  $H_n^{(1)}$ . Since, by (52),

$$\boldsymbol{G}_n + s_0 \boldsymbol{C}_n = \boldsymbol{V}_n^{\mathrm{H}} (\boldsymbol{G} + s_0 \boldsymbol{C}) \boldsymbol{V}_n$$

the condition (63) also ensures that the matrix  $G + s_0 C$  is nonsingular.

Expanding the transfer function H in (23) about  $s_0$ , we get

$$\boldsymbol{H}(s) = \sum_{i=0}^{\infty} (-1)^{i} \boldsymbol{M}_{i} (s - s_{0})^{i}, \text{ where } \boldsymbol{M}_{i} := \boldsymbol{L}^{\mathrm{H}} \boldsymbol{A}^{i} \boldsymbol{R}.$$
(64)

On the other hand, expanding the reduced-order transfer function  $H_n^{(1)}$  in (52) about  $s_0$  gives

$$\boldsymbol{H}_{n}^{(1)}(s) = \sum_{i=0}^{\infty} (-1)^{i} \boldsymbol{M}_{i}^{(1)}(s-s_{0})^{i}, \tag{65}$$

where

$$\boldsymbol{M}_i^{(1)} := \boldsymbol{L}_n^{\mathrm{H}} ((\boldsymbol{G}_n + s_0 \boldsymbol{C}_n)^{-1} \boldsymbol{C}_n)^i (\boldsymbol{G}_n + s_0 \boldsymbol{C}_n)^{-1} \boldsymbol{B}_n.$$

We now show that for any n of the form (29), the first j terms in the expansions (64) and (65) are identical. To this end, we first establish the following proposition.

**Proposition 6.** Let  $n = m_1 + m_2 + \cdots + m_j$ , where  $1 \le j \le j_{max}$ . Then, the matrix

$$\boldsymbol{F}_n := \boldsymbol{V}_n (\boldsymbol{G}_n + \boldsymbol{s}_0 \boldsymbol{C}_n)^{-1} \boldsymbol{V}_n^{\mathrm{H}} \boldsymbol{C}$$
(66)

satisfies the following relations:

$$\boldsymbol{M}_{i}^{(1)} = \boldsymbol{L}^{\mathrm{H}} \boldsymbol{F}_{n}^{i} \boldsymbol{R} \quad \text{for all } i = 0, 1, \dots,$$
(67)

$$F_n^i R = A^i R$$
 for all  $i = 0, 1, ..., j - 1.$  (68)

**Proof.** By (29) and (30), for each i = 1, 2, ..., j, the columns of the matrix  $A^i R$  are all contained in  $\mathcal{K}_n(A, R)$ . Since  $V_n$  is a basis matrix for  $\mathcal{K}_n(A, R)$ , for each i = 1, 2, ..., j, there exists an  $n \times m$  matrix  $E_i$  such that

$$A^{i-1}\boldsymbol{R} = \boldsymbol{V}_n \boldsymbol{E}_i. \tag{69}$$

We now prove (67). From (23) and (69) (for i = 1), we get

$$\boldsymbol{B} = (\boldsymbol{G} + s_0 \boldsymbol{C})\boldsymbol{R} = (\boldsymbol{G} \boldsymbol{V}_n + s_0 \boldsymbol{C} \boldsymbol{V}_n)\boldsymbol{E}_1.$$
<sup>(70)</sup>

Multiplying (70) from the left by  $V_n^{\rm H}$  and using (52), it follows that

$$(\boldsymbol{G}_n + \boldsymbol{s}_0 \boldsymbol{C}_n)^{-1} \boldsymbol{B}_n = \boldsymbol{E}_1.$$

Moreover, from (66), we obtain the relation  $F_n V_n = V_n (G_n + s_0 C_n)^{-1} C_n$ , which, by induction on *i*, implies that

$$F_n^i V_n = V_n ((G_n + s_0 C_n)^{-1} C_n)^i \quad \text{for all } i = 0, 1, \dots$$
(72)

Note that, by (52),  $L_n^H = L^H V_n$ . Using this relation, as well as (65), (71), (72), and (69) (for i = 1), it follows that, for all i = 0, 1, ...,

$$\boldsymbol{M}_{i}^{(1)} = \boldsymbol{L}^{\mathrm{H}}(\boldsymbol{V}_{n}((\boldsymbol{G}_{n} + s_{0}\boldsymbol{C}_{n})^{-1}\boldsymbol{C}_{n})^{i})((\boldsymbol{G}_{n} + s_{0}\boldsymbol{C}_{n})^{-1}\boldsymbol{B}_{n})$$
$$= \boldsymbol{L}^{\mathrm{H}}(\boldsymbol{F}_{n}^{i}\boldsymbol{V}_{n})\boldsymbol{E}_{1} = \boldsymbol{L}^{\mathrm{H}}\boldsymbol{F}_{n}^{i}\boldsymbol{R}.$$

This is just the desired relation (67).

Next, we prove (68) using induction on *i*. For i = 0, (68) is trivially satisfied. Now assume that (68) is true for some  $0 \le i < j - 1$ . We show that (68) then also holds true for i + 1, i.e.,

$$\boldsymbol{F}_{n}^{i+1}\boldsymbol{R} = \boldsymbol{A}^{i+1}\boldsymbol{R}.$$
(73)

Using (23), (68), and (69) (with i replaced by i + 1), we get

$$((G + s_0 C)^{-1} C)(F_n^i R) = A(A^i R) = A^{i+1} R = V_n E_{i+1}.$$
(74)

Multiplying (74) from the left by  $V_n^{\rm H}(G + s_0 C)$ , it follows that

$$(V_n^{\rm H}C)(F_n^{i}R) = (V_n^{\rm H}(G + s_0C)V_n)E_{i+1} = (G_n + s_0C_n)E_{i+1}.$$
(75)

Using (66) and (69) (with *i* replaced by i + 1), we obtain from (75) that

$$F_n^{i+1}R = F_n(F_n^iR) = V_n((G_n + s_0C_n)^{-1}V_n^{\mathrm{H}}C)(F_n^iR)$$
$$= V_nE_{i+1} = A^{i+1}R,$$

which is just the desired relation (73).  $\Box$ 

**Theorem 7.** Let  $n=m_1+m_2+\cdots+m_j$ , where  $1 \le j \le j_{\max}$ , and let  $H_n^{(1)}$  be the reduced-order transfer function given by (52) and (53). Let  $s_0 \in \mathbb{C}$  be an expansion point such that (63) is satisfied. Then,  $H_n^{(1)}$  satisfies

$$H_n^{(1)}(s) = H(s) + \mathcal{O}(s - s_0)^j, \tag{76}$$

*i.e.*,  $H_n^{(1)}$  is a matrix-Padé-type approximant of the transfer function (12), H.

**Proof.** By (64) and (65), the assertion (76) is equivalent to

$$M_i^{(1)} = M_i$$
 for all  $i = 0, 1, \dots, j - 1.$  (77)

By Proposition 6, the matrix  $F_n$  defined in (66) satisfies relations (67) and (68). Inserting (68) into (67) gives

$$\boldsymbol{M}_{i}^{(1)} = \boldsymbol{L}^{\mathrm{H}} \boldsymbol{F}_{n}^{i} \boldsymbol{R} = \boldsymbol{L}^{\mathrm{H}} \boldsymbol{A}^{i} \boldsymbol{R} = \boldsymbol{M}_{i}$$
 for all  $i = 0, 1, \dots, j - 1$ ,

which is just the desired property (77).  $\Box$ 

**Remark 8.** By (28) and (29), we have  $j \ge \lfloor n/m \rfloor$ . Therefore, by Theorem 7, the Taylor expansions of  $H_n^{(1)}$  and H about  $s_0$  match in at least the first  $\lfloor n/m \rfloor$  coefficient matrices.

**Remark 9.** If p < m, then a matrix-Padé-type approximant that matches at least the first  $\lfloor n/p \rfloor$ Taylor coefficient matrices of H about  $s_0$  can be obtained by performing the one-sided projection described in Section 4.1 onto  $\mathscr{K}_n(A^{\mathrm{H}}, L)$ , instead of  $\mathscr{K}_n(A, R)$ .

# 5.2. $H_n^{(2)}$ is a matrix-Padé approximant

It turns out that, in general, the reduced-order transfer function  $H_n^{(2)}$  defined in (56) is even a better approximation to H than  $H_n^{(1)}$ . To properly state this result, we first define the integers

$$n_{\min} := \max\{m_1, p_1\}$$
 and  $n_{\max} := \min\left\{\sum_{j=1}^{j_{\max}} m_j, \sum_{k=1}^{k_{\max}} p_k\right\},$ 

where the  $m_j$ 's and  $p_k$ 's are the integers given by (27), (28) and (32), (33), respectively. The main result of this section is then as follows.

**Theorem 10.** Let  $n_{\min} \leq n \leq n_{\max}$ , and let j = j(n) and k = k(n) be the maximal integers such that

$$m_1 + m_2 + \dots + m_j \leq n \quad and \quad p_1 + p_2 + \dots + p_k \leq n, \tag{78}$$

respectively. Let  $s_0 \in \mathbb{C}$  be an expansion point such that (63) is satisfied, and let  $H_n^{(2)}$  be the reduced-order transfer function given by the two-sided projection (56). Then,  $H_n^{(2)}$  satisfies

$$H_n^{(2)}(s) = H(s) + \mathcal{O}(s - s_0)^{q(n)}, \quad \text{where } q(n) = j(n) + k(n).$$
(79)

Moreover, in (79), the exponent q(n) is as large as possible, and hence  $H_n^{(2)}$  is a matrix-Padé approximant of the transfer function (12), H.

**Proof.** In [12], we studied the reduced-order transfer function  $H_n^{(2)}$  given by (60), where  $\rho_n$ ,  $\eta_n$ ,  $\Delta_n$ , and  $T_n$  are the matrices generated by *n* steps of the Lanczos-type method sketched in Section 3.5. In particular, in [12, Theorem 1], we showed that  $H_n^{(2)}$  satisfies the properties listed in Theorem 10 above. Recall from Section 4.2 that the reduced-order transfer functions defined in (56) via a two-sided projection and the one given by (60) in terms of the Lanczos-type method are identical. Therefore, the assertions in Theorem 10 follow from [12, Theorem 1].  $\Box$ 

**Remark 11.** In view of (28), (33), and (78), we have  $j(n) \ge \lfloor n/m \rfloor$  and  $k(n) \ge \lfloor n/p \rfloor$ . Therefore, by Theorem 10, the Taylor expansions of  $H_n^{(2)}$  and H about  $s_0$  match in at least the first  $\lfloor n/m \rfloor + \lfloor n/p \rfloor$  coefficient matrices.

#### 6. Passivity and stability

As we discussed in Section 2, in circuit simulation, reduced-order modeling is mostly applied to large passive linear subcircuits, such as RLC networks consisting of only resistors, inductors, and capacitors. When reduced-order models of such subcircuits are used within a simulation of the whole circuit, stability of the overall simulation can only be guaranteed if the reduced-order models preserve the passivity of the original subcircuits; see, e.g. [6,27]. Unfortunately, except for special cases such as RC subcircuits consisting of only resistors and capacitors, the Padé model given by  $H_n^{(2)}$  is not passive in general; see, e.g. [4,7,14,15,19]. In this section, we derive a sufficient criterion for passivity of general transfer functions, and then apply the criterion to establish passivity of the particular projected model given by (62).

Roughly speaking, a (linear or nonlinear) dynamical system is *passive* if it does not generate energy. The concept was first used in circuit theory; see, e.g. [2,18]. For example, networks consisting of only resistors, inductors, and capacitors are passive.

The following well-known theorem (see, e.g. [2,31]) relates the passivity of the linear dynamical system (8) to the positive realness of its transfer function. Here and in the sequel, we assume that m = p in (8).

**Theorem A.** The linear dynamical system (8) is passive if, and only if, the associated transfer function (12), H, is positive real.

The definition of a positive real matrix-valued function is as follows; see, e.g. [2].

**Definition 12.** A function  $H : \mathbb{C} \mapsto (\mathbb{C} \cup \{\infty\})^{m \times m}$  is called *positive real* if

(i) **H** has no poles in  $\mathbb{C}_+$ ;

(ii)  $H(\bar{s}) = \overline{H(s)}$  for all  $s \in \mathbb{C}$ ;

(iii)  $\operatorname{Re}(\boldsymbol{x}^{\mathrm{H}}\boldsymbol{H}(s)\boldsymbol{x}) \ge 0$  for all  $s \in \mathbb{C}_+$  and  $\boldsymbol{x} \in \mathbb{C}^m$ .

Recall that a function  $H : \mathbb{C} \mapsto (\mathbb{C} \cup \{\infty\})^{m \times m}$  is *stable* if H has no poles in  $\mathbb{C}_+$  and if all possible purely imaginary poles of H are simple. It is well known that any positive real function is necessary stable.

Next, we prove the following sufficient condition for positive realness.

**Theorem 13.** Let  $G, C \in \mathbb{R}^{N \times N}$ , and  $B \in \mathbb{R}^{N \times m}$ . Assume that  $G + G^T \ge 0$ ,  $C = C^T \ge 0$ , and that G + sC is a regular matrix pencil. Then, the transfer function

$$\boldsymbol{H}(s) := \boldsymbol{B}^{\mathrm{T}}(\boldsymbol{G} + s\boldsymbol{C})^{-1}\boldsymbol{B}, \quad s \in \mathbb{C},$$
(80)

is positive real.

**Proof.** We need to show that H satisfies the conditions (i)–(iii) given in Definition 12.

Condition (ii) follows directly from the fact that the matrices G, C, and B in (80) are real.

Next, we verify condition (iii). Let  $s=s_1+is_2 \in \mathbb{C}_+$  be arbitrary, but fixed. Here  $i := \sqrt{-1}$  denotes the purely imaginary unit. Note that

$$(\boldsymbol{G} + s\boldsymbol{C})^{\mathrm{H}} = \boldsymbol{S} - \boldsymbol{K},\tag{81}$$

where

$$S := \frac{1}{2}(G + G^{\mathrm{T}}) + s_1 C, \quad K := \frac{1}{2}(G - G^{\mathrm{T}}) + is_2 C.$$

Recall that  $G + G^T \ge 0$ ,  $C = C^T \ge 0$ , and  $s_1 = \operatorname{Re} s > 0$ . This guarantees that  $S \ge 0$  and  $K = -K^H$ . These properties imply that  $y^H S y \ge 0$  and  $\operatorname{Re}(y^H K y) = 0$  for all  $y \in \mathbb{C}^N$ . Therefore, by (81), we have

$$\operatorname{Re}(\boldsymbol{y}^{\mathrm{H}}(\boldsymbol{G}+s\boldsymbol{C})^{\mathrm{H}}\boldsymbol{y}) = \boldsymbol{y}^{\mathrm{H}}\boldsymbol{S}\boldsymbol{y} \ge 0 \quad \text{for all } \boldsymbol{y} \in \mathbb{C}^{N}.$$
(82)

We now assume that  $s \in \mathbb{C}_+$  is such that the matrix G + sC is nonsingular. Furthermore, let  $x \in \mathbb{C}^m$  be arbitrary, and set

$$\mathbf{y} := (\mathbf{G} + s\mathbf{C})^{-1}\mathbf{B}\mathbf{x}. \tag{83}$$

Then, by (80) and (83), we have

$$\boldsymbol{x}^{\mathrm{H}}\boldsymbol{H}(s)\boldsymbol{x} = \boldsymbol{x}^{\mathrm{H}}\boldsymbol{B}^{\mathrm{T}}(\boldsymbol{G} + s\boldsymbol{C})^{-1}\boldsymbol{B}\boldsymbol{x}$$
  
$$= \boldsymbol{x}^{\mathrm{H}}\boldsymbol{B}^{\mathrm{T}}(\boldsymbol{G} + s\boldsymbol{C})^{-\mathrm{H}}(\boldsymbol{G} + s\boldsymbol{C})^{\mathrm{H}}(\boldsymbol{G} + s\boldsymbol{C})^{-1}\boldsymbol{B}\boldsymbol{x}$$
  
$$= \boldsymbol{y}^{\mathrm{H}}(\boldsymbol{G} + s\boldsymbol{C})^{\mathrm{H}}\boldsymbol{y}.$$
 (84)

Combining (82) and (84), it follows that

$$\operatorname{Re}(\boldsymbol{x}^{\mathrm{H}}\boldsymbol{H}(s)\boldsymbol{x}) = \operatorname{Re}(\boldsymbol{y}^{\mathrm{H}}(\boldsymbol{G} + s\boldsymbol{C})^{\mathrm{H}}\boldsymbol{y}) \ge 0 \quad \text{for all } \boldsymbol{x} \in \mathbb{C}^{m}$$

$$(85)$$

and for all  $s \in \mathbb{C}_+$  for which G + sC is nonsingular. Now let  $\hat{s} \in \mathbb{C}_+$  be such that  $G + \hat{s}C$  is singular. Note that there are at most N such "singular" points  $\hat{s}$ , since G + sC is assumed to be a regular matrix pencil. Therefore, each  $\hat{s}$  is an isolated point in  $\mathbb{C}_+$ , i.e., there exists an  $\varepsilon = \varepsilon(\hat{s}) > 0$  and a (punctured) neighborhood

$$D_{\varepsilon} := \{ s \in \mathbb{C} \mid 0 < |s - \hat{s}| \leq \varepsilon \}$$

of  $\hat{s}$  such that  $D_{\varepsilon} \subset \mathbb{C}_+$  and the matrix G + sC is nonsingular for all  $s \in D_{\varepsilon}$ . Thus (85) holds true for all  $s \in D_{\varepsilon}$ . If  $\hat{s}$  is not a pole of the rational function H, then  $H(\hat{s}) = \lim_{s \to \hat{s}} H(s)$  is a finite  $m \times m$  matrix. In this case, by taking limits  $s \to \hat{s}$  in (85), it follows that (85) also holds true for  $s = \hat{s}$ . Now suppose that  $\hat{s}$  is a pole of H. Then at least one of the components  $h_{jk}(s)$  of  $H(s) = [h_{jk}(s)]_{1 \le j,k \le m}$  has a pole at  $\hat{s}$ . Such an  $h_{jk}(s)$  maps  $D_{\varepsilon}$  onto a suitable neighborhood of  $\infty$ in the complex plane and, in particular, attains large negative numbers in  $D_{\varepsilon}$ . By selecting a suitable component  $h_{jk}$  of H and an associated vector  $\mathbf{x} \in \mathbb{C}^N$ , it is thus possible to find a point  $s \in D_{\varepsilon}$  such that  $\operatorname{Re}(\mathbf{x}^{H}\mathbf{H}(s)\mathbf{x}) < 0$ . However, this is a contradiction to (85), and therefore,  $\hat{s}$  cannot be a pole of  $\mathbf{H}$ . This concludes the proof of (iii).

It remains to verify condition (i). By (80), if  $\hat{s}$  is a pole of H, then the matrix  $G + \hat{s}C$  is necessarily singular, i.e.,  $\hat{s}$  is a singular point. However, we have just shown that there are no such singular points  $\hat{s} \in \mathbb{C}_+$ . Consequently, H cannot have poles in  $\mathbb{C}_+$ .

The matrix function H satisfies all three conditions (i)–(iii), and hence H is positive real.  $\Box$ 

Finally, we apply Theorem 13 to the reduced-order transfer function (62).

**Corollary 14.** Let *H* be the transfer function given by (21) with matrices that satisfy (22). Let  $V_n \in \mathbb{R}^{N \times n}$  have rank *n* and assume that the matrix pencil

$$\boldsymbol{G}_{n} + s\boldsymbol{C}_{n} := \boldsymbol{V}_{n}^{\mathrm{T}}(\boldsymbol{J}\boldsymbol{G})\boldsymbol{V}_{n} + s\boldsymbol{V}_{n}^{\mathrm{T}}(\boldsymbol{J}\boldsymbol{C})\boldsymbol{V}_{n}$$

$$\tag{86}$$

is regular. Then, the reduced-order transfer function

$$\boldsymbol{H}_{n}^{(1)}(s) := (\boldsymbol{V}_{n}^{\mathrm{T}}\boldsymbol{B})^{\mathrm{T}}(\boldsymbol{G}_{n} + s\boldsymbol{C}_{n})^{-1}(\boldsymbol{V}_{n}^{\mathrm{T}}\boldsymbol{B})$$

$$\tag{87}$$

is positive real, and thus the reduced-order model given by (87) is passive.

**Proof.** By (22) and (86), it follows that  $G_n + G_n^T \ge 0$  and  $C_n = C_n^T \ge 0$ . The transfer function (87),  $H_n^{(1)}$ , is thus positive real by Theorem 13.  $\Box$ 

#### 7. Numerical examples

In this section, we present two circuit examples.

# 7.1. A package model

The first example arises in the analysis of a 64-pin package model used for an RF integrated circuit. Only eight of the package pins carry signals, the rest being either unused or carrying supply voltages. The package is characterized as a passive linear dynamical system with m = p = 16 inputs and outputs, representing eight exterior and eight interior terminals. The package model is described by approximately 4000 circuit elements, resistors, capacitors, inductors, and inductive couplings, resulting in a linear dynamical system with a state-space dimension of about 2000.

In [16], SyMPVL was used to compute a Padé-based reduced-order model (61) of the package, and it was found that a model  $H_n^{(2)}$  of order n = 80 is sufficient to match the transfer-function components of interest. However, the model  $H_n^{(2)}$  has a few poles in the right half of the complex plane, and therefore, it is not passive.

In order to obtain a passive reduced-order model, we ran SyMPVL again on the package example, and this time, also generated the projected reduced-order model  $H_n^{(1)}$  given by (62). The expansion point  $s_0 = 5\pi \times 10^9$  was used. Recall that  $H_n^{(1)}$  is only a Padé-type approximant and thus less accurate than the Padé approximant  $H_n^{(2)}$ . Therefore, one now has to go to order n = 112 to obtain a projected reduced-order model  $H_n^{(1)}$  that matches the transfer-function components of interest. Figs. 1 and 2 show the voltage-to-voltage transfer function between the external terminal of pin no. 1 and the



Fig. 1. Package: Pin no. 1 external to Pin no. 1 internal, exact, projected model, and Padé model.



Fig. 2. Package: Pin no. 1 external to Pin no. 2 internal, exact, projected model, and Padé model.



Fig. 3. Relative error of projected model and Padé model.

internal terminals of the same pin and the neighboring pin no. 2, respectively. The plots show results with the projected model  $H_n^{(1)}$  and the Padé model  $H_n^{(2)}$ , both of order n = 112, compared with an exact analysis.

In Fig. 3, we compare the relative error of the projected model  $H_{112}^{(1)}$  and the Padé model  $H_{112}^{(2)}$  of the same size. Clearly, the Padé model is more accurate. However, out of the 112 poles of  $H_{112}^{(2)}$ , 22 have positive real part, violating the passivity of the Padé model. On the other hand, the projected model is passive.

# 7.2. An extracted RC circuit

This is an extracted RC circuit with about 4000 elements and m = 20 ports. The expansion point  $s_0 = 0$  was used. Since the projected model and the Padé model are identical for RC circuits, we only computed the Padé model via SyMPVL.

The point of this example is to illustrate the usefulness of the deflation procedure built into SyM-PVL. It turned out that sweeps through the first two Krylov blocks, R and AR, of the block Krylov sequence (26) were sufficient to obtain a reduced-order model that matches the transfer function in the frequency range of interest. During the sweep through the second block, 6 almost linearly dependent vectors were discovered and deflated. As a result, the reduced-order model obtained with deflation is only of size n = 2m - 6 = 34. When SyMPVL was rerun on this example, with deflation turned off, a reduced-order model of size n = 40 was needed to match the transfer function. In Fig. 4, we show the  $H_{1,11}$  component of the reduced-order model obtained with deflation and without deflation, compared to the exact transfer function. Clearly, deflation leads to a significantly smaller reduced-order model that is as accurate as the bigger one generated without deflation.



Fig. 4. Impedance  $H_{1,11}$ .

#### 8. Concluding remarks

In the last few years, reduced-order modeling techniques based on Krylov subspaces have become indispensable tools for tackling the large linear subcircuits that arise in the simulation of electronic circuits. Much of this development was and continues to be driven by the emerging need to accurately simulate the interconnect of electronic circuits. Today, circuit interconnect is typically modeled as large linear passive subcircuits that are generated by automatic parasitics-extraction programs. Using reduced-order modeling techniques has become crucial in order to reduce these subcircuits to a size that is manageable for circuit simulators.

To guarantee stability of the overall simulation, it is crucial that passive subcircuits are approximated by passive reduced-order models. While reduced-order models based on projection are passive, they are - in terms of number of matched Taylor coefficients - only half as accurate as the corresponding, in general non-passive, Padé models of the same size. It remains an open problem to describe and construct reduced-order models of a given size that are both passive and of maximal possible accuracy.

Finally, today's circuit simulation is based on the paradigm of lumped circuit elements, which leads to systems of DAEs. As circuit feature sizes continue to decrease and circuit speeds continue to increase, feature sizes are becoming comparable in size with signal wavelengths. As a result, at least parts of a circuit must be modeled as distributed elements, such as transmission lines. Including distributed elements in the simulation paradigm requires a fusion of traditional lumped circuit simulation and electromagnetic simulation. Electromagnetic simulation, however, involves systems of partial differential equations (PDEs). Combining lumped circuit simulation with electromagnetic

simulation will thus require efficient techniques for the solution of very large systems of DAEs coupled with PDEs. One of the challenges then is to develop reduced-order modeling techniques that allow to replace parts of such coupled systems with much smaller models. Research into and development of such techniques have hardly begun.

# References

- J.I. Aliaga, D.L. Boley, R.W. Freund, V. Hernández, A Lanczos-type method for multiple starting vectors, Math. Comp. 69 (2000) 1577-1601; Also available online from http://cm.bell-labs.com/cs/doc/98.
- [2] B.D.O. Anderson, S. Vongpanitlerd, Network Analysis and Synthesis, Prentice-Hall, Englewood Cliffs, NJ, 1973.
- [3] W.E. Arnoldi, The principle of minimized iterations in the solution of the matrix eigenvalue problem, Quart. Appl. Math. 9 (1951) 17–29.
- [4] Z. Bai, P. Feldmann, R.W. Freund, How to make theoretically passive reduced-order models passive in practice, in: Proceedings of IEEE 1998 Custom Integrated Circuits Conference, IEEE, Piscataway, NJ, 1998, pp. 207–210.
- [5] G.A. Baker Jr., P. Graves-Morris, Padé Approximants, 2nd Edition, Cambridge University Press, New York, 1996.
- [6] P.M. Chirlian, Integrated and Active Network Analysis and Synthesis, Prentice-Hall, Englewood Cliffs, NJ, 1967.
- [7] I.M. Elfadel, D.D. Ling, Zeros and passivity of Arnoldi-reduced-order models for interconnect networks, in: Proceedings of the 34th ACM/IEEE Design Automation Conference, ACM, New York, 1997, pp. 28–33.
- [8] P. Feldmann, R.W. Freund, Efficient linear circuit analysis by Padé approximation via the Lanczos process, IEEE Trans. Computer-Aided Design 14 (1995) 639–649.
- [9] P. Feldmann, R.W. Freund, Reduced-order modeling of large linear subcircuits via a block Lanczos algorithm, in: Proceedings of the 32nd ACM/IEEE Design Automation Conference, ACM, New York, 1995, pp. 474–479.
- [10] P. Feldmann, R.W. Freund, Numerical Simulation of Electronic Circuits: State-of-the-Art Techniques and Challenges, Course Notes, 1995. Available online from http://cm.bell-labs.com/who/freund.
- [11] P. Feldmann, R.W. Freund, T. Young, Interconnect Extraction and Analysis in High-Frequency, Sub-Micron, Digital VLSI Design, Tutorial at the 35th Design Automation Conference, San Francisco, CA, 1998. Available online from http://cm.bell-labs.com/who/freund.
- [12] R.W. Freund, Computation of matrix Padé approximations of transfer functions via a Lanczos-type process, in: C.K. Chui, L.L. Schumaker (Eds.), Approximation Theory VIII, Vol. 1: Approximation and Interpolation, World Scientific, Singapore, 1995, pp. 215–222.
- [13] R.W. Freund, Reduced-order modeling techniques based on Krylov subspaces and their use in circuit simulation, in: B.N. Datta (Ed.), Applied and Computational Control, Signals, and Circuits, Vol. 1, Birkhäuser, Boston, 1999, pp. 435–498.
- [14] R.W. Freund, Passive reduced-order models for interconnect simulation and their computation via Krylov-subspace algorithms, in: Proceedings of the 36th ACM/IEEE Design Automation Conference, ACM, New York, 1999, pp. 195–200.
- [15] R.W. Freund, P. Feldmann, Reduced-order modeling of large passive linear circuits by means of the SyPVL algorithm, in: Tech. Dig. 1996 IEEE/ACM International Conference on Computer-Aided Design, IEEE Computer Society Press, Los Alamitos, CA, 1996, pp. 280–287.
- [16] R.W. Freund, P. Feldmann, The SyMPVL algorithm and its applications to interconnect simulation, in: Proceedings of the 1997 International Conference on Simulation of Semiconductor Processes and Devices, IEEE, Piscataway, NJ, 1997, pp. 113–116.
- [17] R.W. Freund, P. Feldmann, Reduced-order modeling of large linear passive multi-terminal circuits using matrix-Padé approximation, in: Proc. Design, Automation and Test in Europe Conference 1998, IEEE Computer Society Press, Los Alamitos, CA, 1998, pp. 530–537.
- [18] E.A. Guillemin, Synthesis of Passive Networks, Wiley, New York, 1957.
- [19] K.J. Kerns, A.T. Yang, Preservation of passivity during RLC network reduction via split congruence transformations, in: Proceedings of the 34th ACM/IEEE Design Automation Conference, ACM, New York, 1997, pp. 34–39.
- [20] S.-Y. Kim, N. Gopal, L.T. Pillage, Time-domain macromodels for VLSI interconnect analysis, IEEE Trans. Computer-Aided Design 13 (1994) 1257–1270.

- [21] C. Lanczos, An iteration method for the solution of the eigenvalue problem of linear differential and integral operators, J. Res. Nat. Bur. Standards 45 (1950) 255–282.
- [22] A. Odabasioglu, Provably passive RLC circuit reduction, M.S. Thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, 1996.
- [23] A. Odabasioglu, M. Celik, L.T. Pileggi, PRIMA: passive reduced-order interconnect macromodeling algorithm, in: Tech. Dig. 1997 IEEE/ACM International Conference on Computer-Aided Design, IEEE Computer Society Press, Los Alamitos, CA, 1997, pp. 58–65.
- [24] L.T. Pileggi, Coping with RC(L) interconnect design headaches, in: Tech. Dig. 1995 IEEE/ACM International Conference on Computer-Aided Design, IEEE Computer Society Press, Los Alamitos, CA, 1995, pp. 246–253.
- [25] L.T. Pillage, R.A. Rohrer, Asymptotic waveform evaluation for timing analysis, IEEE Trans. Computer-Aided Design 9 (1990) 352–366.
- [26] V. Raghavan, J.E. Bracken, R.A. Rohrer, AWESpice: A general tool for the accurate and efficient simulation of interconnect problems, in: Proceedings of the 29th ACM/IEEE Design Automation Conference, ACM, New York, 1992, pp. 87–92.
- [27] R.A. Rohrer, H. Nosrati, Passivity considerations in stability studies of numerical integration algorithms, IEEE Trans. Circuits Syst. 28 (1981) 857–866.
- [28] Y. Saad, Iterative Methods for Sparse Linear Systems, PWS Publishing Company, Boston, 1996.
- [29] A.L. Sangiovanni-Vincentelli, Circuit simulation, in: P. Antognetti, D.O. Pederson, D. de Man (Eds.), Computer Design Aids for VLSI Circuits, Sijthoff & Noordhoff, Alphen aan de Rijn, The Netherlands, 1981, pp. 19–112.
- [30] J. Vlach, K. Singhal, Computer Methods for Circuit Analysis and Design, 2nd Edition, Van Nostrand Reinhold, New York, 1993.
- [31] M.R. Wohlers, Lumped and Distributed Passive Networks, Academic Press, New York, 1969.



Journal of Computational and Applied Mathematics 123 (2000) 423-446

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.nl/locate/cam

# Tikhonov regularization and the L-curve for large discrete ill-posed problems

D. Calvetti<sup>a,1</sup>, S. Morigi<sup>b</sup>, L. Reichel<sup>c,\*,2</sup>, F. Sgallari<sup>d</sup>

<sup>a</sup>Department of Mathematics, Case Western Reserve University, Cleveland, OH 44106, USA <sup>b</sup>Dipartimento di Matematica, Università di Bologna, Bologna, Italy <sup>c</sup>Department of Mathematics and Computer Science, Kent State University, Kent, OH 44242, USA <sup>d</sup>Dipartimento di Matematica, Università di Bologna, Bologna, Italy

Received 10 December 1999; received in revised form 4 February 2000

#### Abstract

Discretization of linear inverse problems generally gives rise to very ill-conditioned linear systems of algebraic equations. Typically, the linear systems obtained have to be regularized to make the computation of a meaningful approximate solution possible. Tikhonov regularization is one of the most popular regularization methods. A regularization parameter specifies the amount of regularization and, in general, an appropriate value of this parameter is not known a priori. We review available iterative methods, and present new ones, for the determination of a suitable value of the regularization parameter by the L-curve criterion and the solution of regularized systems of algebraic equations. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Ill-posed problem; Regularization; L-curve criterion; Gauss quadrature

# 1. Introduction

Many problems in science and engineering can be formulated as linear inverse problems, i.e., problems that require the determination of the unknown input to a linear system from the known output. For instance, image reconstruction from projections is a linear inverse problems. The discretization of a linear inverse problem typically gives rise to a linear system of equations

$$A\mathbf{x} = \mathbf{b}, \quad A \in \mathbb{R}^{m \times n}, \quad \mathbf{x} \in \mathbb{R}^{n}, \quad \mathbf{b} \in \mathbb{R}^{m}$$
(1)

<sup>\*</sup> Corresponding author.

*E-mail addresses:* dxc57@po.cwru.edu (D. Calvetti), morigi@dm.unibo.it (S. Morigi), reiche1@mcs.kent.edu (L. Reichel), sgallari@dm.unibo.it (F. Sgallari).

<sup>&</sup>lt;sup>1</sup>Research supported in part by NSF grant DMS-9806702.

<sup>&</sup>lt;sup>2</sup> Research supported in part by NSF grants DMS-9721436 and DMS-9806413.

with a very ill-conditioned matrix A of ill-determined rank. The computation of a meaningful approximate solution of the linear system (1) in general requires that the system be replaced by a nearby system that is less sensitive to perturbations. This replacement is referred to as regularization. Tikhonov regularization is one of the oldest and most popular regularization methods. In its simplest form, Tikhonov regularization replaces the linear system (1) by the regularized system

$$(A^{\mathrm{T}}A + \mu I)\boldsymbol{x} = A^{\mathrm{T}}\boldsymbol{b},\tag{2}$$

where  $\mu \ge 0$  is a regularization parameter that determines the amount of regularization and *I* is the identity operator. For any fixed  $\mu > 0$ , system (2) has the unique solution

$$\boldsymbol{x}_{\mu} = (A^{\mathrm{T}}A + \mu I)^{-1}A^{\mathrm{T}}\boldsymbol{b}.$$
(3)

It is the purpose of the present paper to discuss several iterative methods for the determination of a suitable value of the regularization parameter  $\mu > 0$  and the computation of the associated solution  $x_{\mu}$  of large-scale problems of the form (2). We remark that Björck [3] described how iterative methods for the solution of (2) can be modified to be applicable to the solution of

$$(A^{\mathrm{T}}A + \mu B^{\mathrm{T}}B)\mathbf{x} = A^{\mathrm{T}}\mathbf{b}$$

for a large class of regularization operators B.

Note that solution (3) of (2) satisfies  $x_{\mu} \to x_0 = A^{\dagger} b$  as  $\mu \searrow 0$ , where  $A^{\dagger}$  denotes the Moore–Penrose pseudo-inverse of A. In problems of interest to us, the matrix A has many "tiny" singular values and the right-hand side vector b is contaminated by measurement errors (noise). Therefore, the Moore–Penrose solution  $x_0$  in general has "huge" components and is of little practical interest.

For future reference, we note that the solution  $x_{\mu}$  of (2) satisfies the minimization problem

$$\min_{\boldsymbol{x}\in\mathbb{R}^n}\{\|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|^2+\mu\|\boldsymbol{x}\|^2\}.$$
(4)

Here and below  $\|\cdot\|$  denotes the Euclidean norm.

The determination of a suitable value of the regularization parameter  $\mu$  is an important task. Several methods that assume knowledge of the norm of the noise in **b** are discussed in [7,12]. We are interested in problems for which the norm of the noise (error) in **b** is not explicitly known, and focus on methods that use the L-curve, defined below, to determine a suitable value of  $\mu$ . The L-curve was first applied by Lawson and Hanson, see [22, Chapter 26], and more recently by Hansen and O'Leary [14,18], to investigate properties of the regularized system (2) for different values of the regularization parameter  $\mu$ . Introduce the discrepancy

$$\boldsymbol{d}_{\mu} = A\boldsymbol{x}_{\mu} - \boldsymbol{b},\tag{5}$$

associated with the solution  $x_{\mu}$  of the regularized system (2). Let  $\phi$  be a monotonically increasing function and define the curve

$$\mathbb{L} = \{ (\phi(\|\boldsymbol{x}_{\mu}\|^{2}), \phi(\|\boldsymbol{d}_{\mu}\|^{2})) \colon \mu > 0 \}.$$
(6)

Often  $\phi$  is chosen to be one of the functions

$$\phi(t) = t$$
,  $\phi(t) = \sqrt{t}$ , or  $\phi(t) = \frac{1}{2} \log_{10} t$ ,  $t > 0$ .

The curve (6) is known as the L-curve, because under suitable conditions on A and b it is shaped roughly like the letter "L". The following result specifies the shape of the L-curve under quite general conditions on A and b.

**Proposition 1.1.** Let  $q(\mu) = \phi(||\mathbf{x}_{\mu}||^2)$  and  $s(\mu) = \phi(||\mathbf{d}_{\mu}||^2)$ , where  $\mathbf{x}_{\mu}$  and  $\mathbf{d}_{\mu}$  are defined by (3) and (5), respectively, and  $\phi$  is a differentiable monotonically increasing function. Then  $q(\mu)$  is a decreasing function of  $\mu$ ,  $s(\mu)$  is an increasing function of  $\mu$ , and the curve  $\{(q(\mu), s(\mu)): \mu > 0\}$  has negative slope. Moreover, the curve is convex for  $\phi(t) = t$ .

**Proof.** The proposition follows by substituting the singular value decomposition of A into (3); see, e.g., [24] for details. Related results are also discussed in [14,16,18].  $\Box$ 

An illuminating discussion on properties of the L-curve is presented by Hansen [16, Sections 7.5–7.7]. Hansen and O'Leary [14,18] proposed to use the value  $\mu$  of the regularization parameter that corresponds to the point  $(\phi(||\mathbf{x}_{\mu}||^2), \phi(||\mathbf{d}_{\mu}||^2))$  at the "vertex" of the "L". We denote this value by  $\mu_{\rm L}$ . A heuristic motivation for this choice of  $\mu$  is that when  $\mu > 0$  is "tiny", then the associated solution  $\mathbf{x}_{\mu}$  of (2) is of "huge" norm and is likely to be contaminated by the propagated error that stems from errors in the given right-hand side vector  $\mathbf{b}$ . Conversely, when  $\mu$  is large, the vector  $\mathbf{x}_{\mu}$  generally is a poor approximation of a solution of (1) and the associated discrepancy (5) is of large norm. The choice  $\mu = \mu_{\rm L}$  seeks to balance the discrepancy and the propagated error in the computed approximate solution  $\mathbf{x}_{\mu}$  due to errors in  $\mathbf{b}$ . The parameter value  $\mu_{\rm L}$  is said to satisfy the L-curve criterion.

The value  $\mu_L$  of the regularization parameter is not guaranteed to be appropriate for all linear systems of equations of the form (2) with very ill-conditioned matrices A of ill-determined rank. However, considerable computational experience indicates that the L-curve criterion is a powerful method for determining a suitable value of the regularization parameter for many problems of interest in science and engineering; see [13;16, Section 7.5.1] and [17,25] for insightful discussions on the properties and shortcomings of the L-curve criterion. This paper reviews available iterative methods, and describes new ones, for the simultaneous determination of approximations of the value  $\mu_L$  and the associated solution  $\mathbf{x}_{\mu_L}$  of (2).

The following result is helpful for determining the approximate location of the L-curve. It has been shown for the function  $\phi(t) = \sqrt{t}$  by, e.g., Lawson and Hanson [22, p. 193].

**Proposition 1.2.** Let  $\phi$  be a monotonically increasing function and let  $\mathbf{x} \in \mathbb{R}^n$ . Then the point  $(\phi(||\mathbf{x}||^2), \phi(||A\mathbf{x} - \mathbf{b}||^2))$  is on or above the L-curve (6).

**Proof.** The proposition follows if for any  $x_{\mu}$  given by (3),

$$\phi(\|A\boldsymbol{x}_{\mu} - \boldsymbol{b}\|^2) \leqslant \phi(\|A\boldsymbol{x} - \boldsymbol{b}\|^2), \quad \forall \boldsymbol{x} \in \mathbb{R}^n, \quad \text{such that } \phi(\|\boldsymbol{x}\|^2) \leqslant \phi(\|\boldsymbol{x}_{\mu}\|^2).$$
(7)

We show (7) by contradiction. Thus, assume that there is a vector  $\hat{x} \in \mathbb{R}^n$ , such that

$$\phi(\|A\hat{x} - b\|^2) < \phi(\|Ax_{\mu} - b\|^2), \quad \phi(\|\hat{x}\|^2) \leq \phi(\|x_{\mu}\|^2)$$

for some  $x_{\mu}$ . Since  $\phi$  is monotonically increasing, it follows that

$$||A\hat{x} - b||^2 < ||Ax_{\mu} - b||^2, ||\hat{x}||^2 \leq ||x_{\mu}||^2$$

and therefore

 $||A\hat{x} - b||^2 + \mu ||\hat{x}||^2 < ||Ax_{\mu} - b||^2 + \mu ||x_{\mu}||^2.$ 

However, this inequality violates the fact that  $x_{\mu}$  solves (4).  $\Box$ 

This paper is organized as follows. Section 2 discusses iterative methods for the computation of approximations of  $\mu_L$  and  $x_{\mu_L}$  based on the Lanczos bidiagonalization method. In particular, we describe a modification of the L-ribbon method proposed in [6] that is well suited for the iterative solution of large very ill-conditioned underdetermined systems. Section 3 discusses iterative methods based on the Arnoldi process. These methods do not require the evaluation of matrix-vector products with the transpose of the matrix A. Computed examples are presented in Section 4.

## 2. Iterative methods based on Lanczos bidiagonalization

The Lanczos bidiagonalization algorithm below is by Paige and Saunders [23] referred to as bidiag1. It has been applied in many algorithms for the computation of approximate solutions of large linear systems of equations (1) with very ill-conditioned matrices; see, e.g., [3–5,16] and references therein. The algorithm carries out  $\ell$  steps of the Lanczos bidiagonalization process applied to the matrix A.

Algorithm 1 (Lanczos Bidiagonalization Algorithm).

 $\begin{array}{ll} Input: \ \pmb{b} \in \mathbb{R}^{m}, \ A \in \mathbb{R}^{m \times n}, \ 0 < \ell < n; \\ Output: \ \{\pmb{u}_{j}\}_{j=1}^{\ell+1}, \{\pmb{v}_{j}\}_{j=1}^{\ell}, \{\rho_{j}\}_{j=1}^{\ell}, \{\sigma_{j}\}_{j=1}^{\ell+1}; \\ \sigma_{1} := \|\pmb{b}\|; \ \pmb{u}_{1} := \pmb{b}/\sigma_{1}; \ \tilde{\pmb{v}}_{1} := A^{\mathsf{T}}\pmb{u}_{1}; \ \rho_{1} := \|\tilde{\pmb{v}}_{1}\|; \ \pmb{v}_{1} := \tilde{\pmb{v}}_{1}/\rho_{1}; \\ for \ j = 2, 3, \dots, \ell \ \mathbf{do} \\ \quad \tilde{\pmb{u}}_{j} := A\pmb{v}_{j-1} - \rho_{j-1}\pmb{u}_{j-1}; \ \sigma_{j} := \|\tilde{\pmb{u}}_{j}\|; \ \pmb{u}_{j} := \tilde{\pmb{u}}_{j}/\sigma_{j}; \\ \quad \tilde{\pmb{v}}_{j} := A^{\mathsf{T}}\pmb{u}_{j} - \sigma_{j}\pmb{v}_{j-1}; \ \rho_{j} := \|\tilde{\pmb{v}}_{j}\|; \ \pmb{v}_{j} := \tilde{\pmb{v}}_{j}/\rho_{j}; \\ end \ j; \\ \quad \tilde{\pmb{u}}_{\ell+1} := A\pmb{v}_{\ell} - \rho_{\ell}\pmb{u}_{\ell}; \ \sigma_{\ell+1} := \|\tilde{\pmb{u}}_{\ell+1}\|; \ \pmb{u}_{\ell+1} := \tilde{\pmb{u}}_{\ell+1}/\sigma_{\ell+1}. \end{array}$ 

We assume that the parameter  $\ell$  in Algorithm 1 is chosen small enough so that all computed  $\rho_j$ and  $\sigma_j$  are positive. Then the algorithm determines the matrices  $U_{\ell} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \dots, \boldsymbol{u}_{\ell}] \in \mathbb{R}^{m \times \ell}$ ,  $U_{\ell+1} = [U_{\ell}, \boldsymbol{u}_{\ell+1}] \in \mathbb{R}^{m \times (\ell+1)}$  and  $V_{\ell} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \dots, \boldsymbol{v}_{\ell}] \in \mathbb{R}^{n \times \ell}$  with orthonormal columns, as well as the lower bidiagonal matrix

$$C_{\ell} = \begin{bmatrix} \rho_1 & & & \\ \sigma_2 & \rho_2 & & \\ & \ddots & \ddots & \\ & & \sigma_{\ell-1} & \rho_{\ell-1} \\ & & & \sigma_{\ell} & \rho_{\ell} \end{bmatrix} \in \mathbb{R}^{\ell \times \ell}.$$
(8)

For future reference, we also introduce the matrix

$$\bar{C}_{\ell-1} = \begin{bmatrix} C_{\ell-1} \\ \sigma_{\ell} \boldsymbol{e}_{\ell-1}^{\mathrm{T}} \end{bmatrix} \in \mathbb{R}^{\ell \times (\ell-1)},\tag{9}$$

which is made up of the  $\ell - 1$  first columns of  $C_{\ell}$ .

Define the Krylov spaces

$$\mathbb{K}_{\ell}(A^{\mathrm{T}}A, A^{\mathrm{T}}\boldsymbol{b}) = \operatorname{span}\{A^{\mathrm{T}}\boldsymbol{b}, A^{\mathrm{T}}AA^{\mathrm{T}}\boldsymbol{b}, \dots, (A^{\mathrm{T}}A)^{\ell-1}A^{\mathrm{T}}\boldsymbol{b}\},\tag{10}$$

$$\mathbb{K}_{\ell}(AA^{\mathrm{T}}, \boldsymbol{b}) = \operatorname{span}\{\boldsymbol{b}, AA^{\mathrm{T}}\boldsymbol{b}, \dots, (AA^{\mathrm{T}})^{\ell-1}\boldsymbol{b}\}.$$
(11)

It follows from the recursion formulas of Algorithm 1 that

$$\mathbb{K}_{\ell}(A^{\mathrm{T}}A, A^{\mathrm{T}}\boldsymbol{b}) = \operatorname{range}(V_{\ell}), \quad \mathbb{K}_{\ell}(AA^{\mathrm{T}}, \boldsymbol{b}) = \operatorname{range}(U_{\ell}), \tag{12}$$

i.e., the columns of  $V_{\ell}$  and  $U_{\ell}$  form orthonormal bases of the Krylov spaces (10) and (11), respectively.

Moreover, the recursion formulas of Algorithm 1 show that

$$AV_{\ell} = U_{\ell}C_{\ell} + \sigma_{\ell+1}\boldsymbol{u}_{\ell+1}\boldsymbol{e}_{\ell}^{\mathrm{T}}, \quad A^{\mathrm{T}}U_{\ell} = V_{\ell}C_{\ell}^{\mathrm{T}}, \quad \boldsymbol{b} = \|\boldsymbol{b}\|U_{\ell}\boldsymbol{e}_{1},$$
(13)

where  $\boldsymbol{e}_{\ell} = [0, \dots, 0, 1]^{\mathrm{T}} \in \mathbb{R}^{\ell}$  denotes the  $\ell$ th axis vector. Combining these equations yields

$$AA^{\mathrm{T}}U_{\ell} = U_{\ell}C_{\ell}C_{\ell}^{\mathrm{T}} + \sigma_{\ell+1}\rho_{\ell}\boldsymbol{u}_{\ell+1}\boldsymbol{e}_{\ell}^{\mathrm{T}}.$$
(14)

It follows that  $\{u_j\}_{j=1}^{\ell+1}$  are Lanczos vectors and

$$T_{\ell} = C_{\ell} C_{\ell}^{\mathrm{T}} \tag{15}$$

is the symmetric tridiagonal Lanczos matrix obtained when applying  $\ell$  steps of the Lanczos algorithm for symmetric matrices to the matrix  $AA^{T}$  with initial vector **b**; see, e.g., [11, Chapter 9.1] for a discussion of the Lanczos algorithm. We will use the matrix (15) in our review of Gauss quadrature rules in Section 2.1.1 below.

Our discussion so far has not taken the effect of round-off errors into account. These errors may cause the computed columns of the matrices  $U_{\ell+1}$  and  $V_{\ell}$  not to be orthogonal. Therefore Algorithm 1 is often implemented with reorthogonalization of the columns of  $U_{\ell+1}$  and  $V_{\ell}$ ; see, e.g., [5]. This approach requires that both matrices  $U_{\ell+1}$  and  $V_{\ell}$  be stored. On the other hand, when Algorithm 1 is implemented without reorthogonalization, the application described in Section 2.1 requires that only the matrix  $U_{\ell+1}$  and a few of the most recently generated columns of the matrix  $V_{\ell}$  be stored simultaneously in computer memory.<sup>1</sup> Analogously, the application described in Section 2.2 requires that only the matrix  $V_{\ell}$  and a few of the most recently generated columns of  $U_{\ell+1}$  be stored simultaneously. We propose to only reorthogonalize the columns of  $U_{\ell+1}$  or of  $V_{\ell}$ . This approach allows the same (low) storage requirement as when no reorthogonalization is carried out and gives higher accuracy.

In the following proposition and its proof we use the notation  $a \doteq b$  to denote that a is an accurate approximation of b.

<sup>&</sup>lt;sup>1</sup>We ignore the possibility of reducing storage demand by generating the columns of  $U_{\ell+1}$  and  $V_{\ell}$  twice because of the high computational effort of this approach.

**Proposition 2.1.** Assume that the columns of  $U_{\ell+1}$  are reorthogonalized in Algorithm 1 so that they are numerically orthogonal, and that the columns of  $V_{\ell}$  have been computed as described in the algorithm, i.e., without reorthogonalization. Then  $U_{\ell}^{T}AA^{T}U_{\ell} \doteq C_{\ell}C_{\ell}^{T}$ , i.e., the computed bidiagonal matrix  $C_{\ell}$  is such that the tridiagonal matrix  $C_{\ell}C_{\ell}^{T}$  is an accurate approximation of an orthogonal section of  $AA^{T}$ .

**Proof.** According to Paige and Saunders [23, p. 47], the relations (13) hold to machine precision also in the presence of round-off errors. This is independent of whether the columns of  $U_{\ell+1}$  or  $V_{\ell}$  are reorthogonalized to be numerically orthogonal. Thus, the computed matrices  $U_{\ell+1}$ ,  $V_{\ell}$  and  $C_{\ell}$  determined by Algorithm 1 in finite precision arithmetic with reorthogonalization of the columns of  $U_{\ell+1}$ , satisfy

$$AV_{\ell} \doteq U_{\ell}C_{\ell} + \sigma_{\ell+1}\boldsymbol{u}_{\ell+1}\boldsymbol{e}_{\ell}^{\mathrm{T}},\tag{16}$$

$$A^{\mathrm{T}}U_{\ell} \doteq V_{\ell}C_{\ell}^{\mathrm{T}}.$$
(17)

Multiplying (17) by A, and using the relation (16) yields

$$AA^{\mathrm{T}}U_{\ell} \doteq AV_{\ell}C_{\ell}^{\mathrm{T}} \doteq U_{\ell}C_{\ell}C_{\ell}^{\mathrm{T}} + \sigma_{\ell+1}\boldsymbol{u}_{\ell+1}\boldsymbol{e}_{\ell}^{\mathrm{T}}C_{\ell}^{\mathrm{T}}$$

It follows from  $\boldsymbol{e}_{\ell}^{\mathrm{T}} \boldsymbol{C}_{\ell}^{\mathrm{T}} = \rho_{\ell} \boldsymbol{e}_{\ell}^{\mathrm{T}}$  that

 $AA^{\mathrm{T}}U_{\ell} \doteq U_{\ell}C_{\ell}C_{\ell}^{\mathrm{T}} + \sigma_{\ell+1}\rho_{\ell}\boldsymbol{u}_{\ell+1}\boldsymbol{e}_{\ell}^{\mathrm{T}}.$ 

The proposition now follows from the orthonormality of the columns of  $U_{\ell+1}$ .  $\Box$ 

The above proposition justifies the use of the computed matrix  $C_{\ell}$  for the evaluation of Gauss and Gauss–Radau quadrature rules described in Section 2.1.1. These quadrature rules allow us to inexpensively compute an approximation of the L-curve, which we refer to as the L-ribbon. The latter makes it possible to fairly inexpensively determine a value of the regularization parameter that approximates the value  $\mu_{\rm L}$ .

Reorthogonalization of the columns of  $V_{\ell}$ , but not of the columns of  $U_{\ell+1}$ , is discussed in the context of overdetermined systems of equations in Section 2.2.

We advocate reorthogonalization of the columns of  $U_{\ell}$  because then the tridiagonal matrix (15) can be associated with a Gauss rule; see Section 2.1.1. Example 4.1 of Section 4 illustrates the effect of reorthogonalization. In addition, reorthogonalization may reduce the number of bidiagonalization steps necessary to determine an acceptable approximate solution of the linear system of equations (2) for a given value of the regularization parameter  $\mu$ .

# 2.1. Underdetermined systems

Assume that  $m \ll n$  in (1). Then the vectors  $u_j \in \mathbb{R}^m$  generated by Algorithm 1 have fewer components, and therefore require less computer storage, than the vectors  $v_j \in \mathbb{R}^n$  generated by the algorithm. Application of the decompositions (13) to the standard formulation of Tikhonov regularization (2) requires that all the vectors  $v_j$  generated be stored simultaneously, in addition to a few of the vectors  $u_j$ ; see Section 2.2 for a discussion. Here we present a variant of Tikhonov regularization that allows an interchange of the role of the vectors  $u_j$  and  $v_j$  with the purpose of reducing the storage requirement. Specifically, in this variant of Tikhonov regularization all of the vectors  $u_j$  generated

have to be stored simultaneously, but only a few of the vectors  $v_j$ . We achieve this by solving the following linear system of equations

$$(AA^{\mathrm{T}} + \mu I)\mathbf{y} = \mathbf{b},\tag{18}$$

instead of the standard Tikhonov system (2). The solution  $x_{\mu}$  of the latter system can be computed from the solution

$$\boldsymbol{y}_{\boldsymbol{\mu}} = (AA^{\mathrm{T}} + \boldsymbol{\mu}I)^{-1}\boldsymbol{b} \tag{19}$$

of (18). We note that for any fixed  $\mu > 0$ , the solution  $y_{\mu}$  of (18) is unique.

**Proposition 2.2.** Assume that  $\mu > 0$  and let  $y_{\mu}$  be defined by (19). Then the solution of (2) is given by  $\mathbf{x}_{\mu} = A^{\mathrm{T}} \mathbf{y}_{\mu}$ .

**Proof.** Multiplication of (18) by  $A^{T}$  from the left yields

 $(A^{\mathrm{T}}A + \mu I)A^{\mathrm{T}}\boldsymbol{y}_{\mu} = A^{\mathrm{T}}\boldsymbol{b}.$ 

Since the system (2) has a unique solution, it is given by  $A^{T}y_{\mu}$ .  $\Box$ 

When  $\mu = 0$ , the solution  $y_{\mu}$  of (18) might not satisfy the discrete Picard condition, see [16, Section 4.5] for a definition, even if the associated solution  $x_{\mu} = A^{T}y_{\mu}$  of (2) does. We therefore only consider Eq. (18) for  $\mu > 0$ .

The L-curve for system (18) is given by

$$\mathbb{L} = \{ (\phi(\|\mathbf{y}_{\mu}\|^{2}), \ \phi(\|\mathbf{d}_{\mu}\|^{2})) \colon \mu > 0 \},$$
(20)

where

$$\boldsymbol{d}_{\boldsymbol{\mu}} = \boldsymbol{A}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{y}_{\boldsymbol{\mu}} - \boldsymbol{b}. \tag{21}$$

The following analog of Proposition 1.1 is valid for the L-curve (20).

**Proposition 2.3.** Let  $q(\mu) = \phi(||\mathbf{y}_{\mu}||^2)$  and  $s(\mu) = \phi(||\mathbf{d}_{\mu}||^2)$ , where  $\mathbf{y}_{\mu}$  and  $\mathbf{d}_{\mu}$  are defined by (19) and (21), respectively, and  $\phi$  is a differentiable monotonically increasing function. Then  $q(\mu)$  is a decreasing function of  $\mu$ ,  $s(\mu)$  is an increasing function of  $\mu$ , and the curve  $\{(q(\mu), s(\mu)): \mu > 0\}$  has negative slope.

**Proof.** The proposition follows by substituting the singular value decomposition of A into (19), similarly as the proof of Proposition 1.1.  $\Box$ 

The above proposition shows that the L-curves (6) and (20) share some properties. However, not all properties shown for the curve (6) hold for the curve (20). For instance, one can construct examples for which the L-curve (20) is not convex when  $\phi(t) = t$ , cf. Proposition 1.1. Moreover, there is no analog of Proposition 1.2 for the L-curve (20), i.e., for an arbitrary vector  $\mathbf{y} \in \mathbb{R}^m$ , the point  $(\phi(||\mathbf{y}||^2), \phi(||AA^T\mathbf{y} - \mathbf{b}||^2))$  may be above, on or below the L-curve (20). Finally, the location of the "vertices" of the L-curves (6) and (20) may differ. Nevertheless, we have found the L-curve (20) to be a valuable aid for determining a suitable value of the regularization parameter  $\mu$  when the norm of the noise in the right-hand side vector  $\mathbf{b}$  is not very small.

In this section, we assume that the "vertex" of the L-curve (20) gives a suitable value of the regularization parameter  $\mu$ , and describe how an approximation of the L-curve, referred to as the L-ribbon, can be computed inexpensively. We then choose a value of the regularization parameter that corresponds to a point close to the "vertex" of the L-ribbon.

We will derive easily computable upper and lower bounds of the norm of the quantities (19) and (21) in Section 2.1.1. These bounds are obtained by first representing  $||\boldsymbol{y}_{\mu}||^2$  and  $||\boldsymbol{d}_{\mu}||^2$  in terms of a Stieltjes integral. We conclude this section by deriving these representations. It follows from (19) that

$$\|\boldsymbol{y}_{\boldsymbol{\mu}}\|^{2} = \boldsymbol{y}_{\boldsymbol{\mu}}^{\mathrm{T}} \boldsymbol{y}_{\boldsymbol{\mu}} = \boldsymbol{b}^{\mathrm{T}} (AA^{\mathrm{T}} + \boldsymbol{\mu}I)^{-2} \boldsymbol{b},$$
(22)

$$\|\boldsymbol{d}_{\mu}\|^{2} = \|AA^{\mathrm{T}}(AA^{\mathrm{T}} + \mu I)^{-1}\boldsymbol{b} - \boldsymbol{b}\|^{2} = \mu^{2}\boldsymbol{b}^{\mathrm{T}}(AA^{\mathrm{T}} + \mu I)^{-2}\boldsymbol{b}.$$
(23)

Define the function

$$\psi_{\mu}(t) = (t+\mu)^{-2}.$$
(24)

Then

$$\|\boldsymbol{y}_{\boldsymbol{\mu}}\|^{2} = \boldsymbol{b}^{\mathrm{T}} \boldsymbol{\psi}_{\boldsymbol{\mu}} (AA^{\mathrm{T}}) \boldsymbol{b}, \quad \|\boldsymbol{d}_{\boldsymbol{\mu}}\|^{2} = \boldsymbol{\mu}^{2} \boldsymbol{b}^{\mathrm{T}} \boldsymbol{\psi}_{\boldsymbol{\mu}} (AA^{\mathrm{T}}) \boldsymbol{b}.$$
<sup>(25)</sup>

Introduce the spectral factorization

$$AA^{\mathrm{T}} = WAW^{\mathrm{T}},\tag{26}$$

where

$$\Lambda = \operatorname{diag}[\lambda_1, \lambda_2, \dots, \lambda_m] \in \mathbb{R}^{m \times m}, \quad W \in \mathbb{R}^{m \times m}, \quad W^{\mathrm{T}}W = I,$$

and let

$$\boldsymbol{h} = [h_1, h_2, \dots, h_m]^{\mathrm{T}} = W^{\mathrm{T}} \boldsymbol{b}$$

Then

$$\boldsymbol{b}^{\mathrm{T}}\psi_{\mu}(\boldsymbol{A}\boldsymbol{A}^{\mathrm{T}})\boldsymbol{b} = \boldsymbol{h}^{\mathrm{T}}\psi_{\mu}(\boldsymbol{\Lambda})\boldsymbol{h} = \sum_{k=1}^{m}\psi_{\mu}(\lambda_{k})h_{k}^{2} = \int\psi_{\mu}(t)\,\mathrm{d}\boldsymbol{\omega}(t),\tag{27}$$

where the right-hand side is a Stieltjes integral with a nondecreasing piecewise constant measure  $\omega(t)$  with jump discontinuities at the eigenvalues  $\lambda_k$ . Substituting (27) into (25) gives the desired representations

$$|\mathbf{y}_{\mu}||^{2} = \int \psi_{\mu}(t) \,\mathrm{d}\omega(t), \quad ||\mathbf{d}_{\mu}||^{2} = \mu^{2} \int \psi_{\mu}(t) \,\mathrm{d}\omega(t).$$
(28)

# 2.1.1. Gauss quadrature for underdetermined systems

The quantities (25) depend on the regularization parameter  $\mu$ . Their computation for many different values of  $\mu$  is not feasible when the matrix A is very large. However, the computation of lower and upper bounds for several values of  $\mu$  can be carried out efficiently by using Gauss and Gauss–Radau quadrature rules. We remark that the application of Gauss-type quadrature rules to compute bounds for certain matrix functionals is well established; see for instance [10] for a thorough treatment. Recently, Golub and von Matt [12] used this approach to develop a method different from ours for

determining a suitable value of the regularization parameter for Tikhonov regularization in standard form (2).

We briefly review some known facts about Gauss quadrature rules and their connection with the Lanczos process. A more detailed treatment is provided by Golub and Meurant [10]. Define the inner product induced by the measure  $\omega(t)$  introduced in (27),

$$\langle f,g\rangle = \int f(t)g(t) \,\mathrm{d}\omega(t) = \sum_{k=1}^{m} f(\lambda_k)g(\lambda_k)h_k^2 = \boldsymbol{h}^{\mathrm{T}}f(\Lambda)g(\Lambda)\boldsymbol{h}_k$$

and let  $\{q_k\}_{k=0}^{m-1}$  be the family of orthonormal polynomials with respect to this inner product, i.e.,

$$\langle q_k, q_j \rangle = \begin{cases} 0, & k \neq j, \\ 1, & k = j. \end{cases}$$

The  $q_k$  satisfy a three-term recurrence relation of the form

$$tq_{k-1}(t) = \beta_k q_k(t) + \alpha_k q_{k-1}(t) + \beta_{k-1} q_{k-2}(t), \quad k = 1, 2, \dots,$$
(29)

where  $q_{-1}(t) = 0$  and  $q_0(t) = \langle 1, 1 \rangle^{-1/2}$ . It is well known that the tridiagonal Lanczos matrix (15), obtained by applying  $\ell$  steps of the Lanczos algorithm to  $AA^{T}$  with initial vector **b**, has the first  $2\ell - 1$  recurrence coefficients for the  $q_i$  as entries, i.e.,

$$T_{\ell} = \begin{bmatrix} \alpha_{1} & \beta_{1} & & & \\ \beta_{1} & \alpha_{2} & \beta_{2} & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{\ell-2} & \alpha_{\ell-1} & \beta_{\ell-1} \\ & & & & \beta_{\ell-1} & \alpha_{\ell} \end{bmatrix},$$
(30)

see, e.g., [10]. We assume that  $\ell$  is sufficiently small so that  $\beta_j > 0$  for  $1 \leq j < \ell$ .

It is convenient to discuss Gauss quadrature in terms of the matrix  $T_{\ell}$  and its Cholesky factor (8). Let the function f be defined and integrable on an interval that contains the support of the measure  $\omega(t)$  and consider the approximation of the integral

$$\mathscr{I}f = \int f(t) \, \mathrm{d}\omega(t)$$

by quadrature rules of Gauss-type. It is well known, see, e.g., Golub and Meurant [10], that the  $\ell$ -point Gauss rule associated with the measure  $\omega(t)$  defined in (27) can be written as

$$\mathscr{G}_{\ell}(f) = \|\boldsymbol{b}\|^2 \boldsymbol{e}_1^{\mathrm{T}} f(T_{\ell}) \boldsymbol{e}_1 = \|\boldsymbol{b}\|^2 \boldsymbol{e}_1^{\mathrm{T}} f(C_{\ell} C_{\ell}^{\mathrm{T}}) \boldsymbol{e}_1,$$
(31)

where  $\boldsymbol{e}_1 = [1, 0, \dots, 0]^T \in \mathbb{R}^{\ell}$  denotes the first axis vector.

Analogously, the  $\ell$ -point Gauss-Radau quadrature rule associated with the measure  $\omega(t)$  with one assigned node at the origin can be written as

$$\mathscr{R}_{\ell}(f) = \|\boldsymbol{b}\|^2 \boldsymbol{e}_1^{\mathrm{T}} f(\bar{\boldsymbol{C}}_{\ell-1} \bar{\boldsymbol{C}}_{\ell-1}^{\mathrm{T}}) \boldsymbol{e}_1, \tag{32}$$

where the matrix  $\bar{C}_{\ell-1}$  is defined by (9). This representation was proposed by Golub and von Matt [12]. A proof can also be found in [6].

We now turn to the integrand  $f = \psi_{\mu}$  defined by (24).

# **Proposition 2.4.** Assume that $\mu > 0$ and let $\psi_{\mu}$ be defined by (24). Then, for $\ell \ge 1$ ,

$$\boldsymbol{b}^{\mathrm{T}}\psi_{\mu}(AA^{\mathrm{T}})\boldsymbol{b} > \mathscr{G}_{\ell}(\psi_{\mu}), \quad \boldsymbol{b}^{\mathrm{T}}\psi_{\mu}(AA^{\mathrm{T}})\boldsymbol{b} < \mathscr{R}_{\ell}(\psi_{\mu}).$$
(33)

**Proof.** We have to determine the sign of the quadrature error when approximating  $\int \psi_{\mu}(t) d\omega(t)$  by Gauss and Gauss-Radau rules, cf. (27). Since for  $\mu > 0$  the derivatives of the integrand do not change sign on the support of the measure  $\omega(t)$ , the sign of the quadrature error can easily be determined by the remainder formulas for Gauss and Gauss-Radau quadrature rules. We refer to [10] or to [6] for details.  $\Box$ 

# 2.1.2. The L-ribbon for underdetermined systems

Proposition 2.4 indicates how a ribbon-like region that contains the L-curve (20) can be computed inexpensively. Introduce, for  $\mu > 0$ ,

$$\begin{aligned} \xi_{\mu}^{-} &= \phi(\mathscr{G}_{\ell}(\psi_{\mu})), \quad \xi_{\mu}^{+} = \phi(\mathscr{R}_{\ell+1}(\psi_{\mu})), \\ \eta_{\mu}^{-} &= \phi(\mu^{2}\mathscr{G}_{\ell}(\psi_{\mu})), \quad \eta_{\mu}^{+} = \phi(\mu^{2}\mathscr{R}_{\ell+1}(\psi_{\mu})). \end{aligned}$$
(34)

Assume that the conditions of Proposition 2.3 are satisfied. Then it follows from the representations (28) and the inequalities (33) that the L-curve (20) is above the curve  $\mathbb{L}^- = \{(\xi_{\mu}^-, \eta_{\mu}^-): \mu > 0\}$  and below the curve  $\mathbb{L}^+ = \{(\xi_{\mu}^+, \eta_{\mu}^+): \mu > 0\}$ . We refer to the ribbon-like region between the curves  $\mathbb{L}^-$  and  $\mathbb{L}^+$  as the L-ribbon.

The width of the ribbon decreases as  $\ell$  increases; when  $\ell$  in Algorithm 1 is sufficiently large, the L-ribbon has zero width and coincides with the L-curve.

Alternatively, we might define the L-ribbon as the union for all  $\mu > 0$  of the rectangular regions with vertices  $\xi^{\pm}_{\mu}$  and  $\eta^{\pm}_{\mu}$ , i.e., the L-ribbon can be defined as

$$\bigcup_{\mu>0} \{ (\xi,\eta) \colon \xi_{\mu}^{-} \leqslant \xi \leqslant \xi_{\mu}^{+}, \ \eta_{\mu}^{-} \leqslant \eta \leqslant \eta_{\mu}^{+} \}.$$

The following algorithm determines rectangles associated with the parameter values  $\mu_j$ ,  $1 \le j \le p$ . These rectangles are plotted in the numerical examples of Section 4.

## Algorithm 2 (L-Ribbon Algorithm).

*Input*:  $\boldsymbol{b} \in \mathbb{R}^m$ ,  $A \in \mathbb{R}^{m \times n}$ ,  $\ell$ ,  $\{\mu_j\}_{j=1}^p$ ; *Output*:  $\{\xi_{\mu_j}^+\}_{j=1}^p$ ,  $\{\xi_{\mu_j}^-\}_{j=1}^p$ ,  $\{\eta_{\mu_j}^+\}_{j=1}^p$ ,  $\{\eta_{\mu_j}^-\}_{j=1}^p$ ;

(i) Apply  $\ell$  steps of Lanczos bidiagonalization of A with initial vector **b** using Algorithm 1 to compute the entries of the bidiagonal matrices  $C_{\ell}$  and  $\bar{C}_{\ell}$ .

(ii) for j = 1, 2, ..., p do

Evaluate  $\mathscr{G}_{\ell}(\psi_{\mu_j})$  and  $\mathscr{R}_{\ell+1}(\psi_{\mu_j})$ ; Compute  $\xi_{\mu_j}^{\pm}$  and  $\eta_{\mu_j}^{\pm}$  defined by (34); *end* j

Thus, Algorithm 2 computes the union of rectangles

$$\bigcup_{j=1}^{p} \{ (\xi, \eta) : \xi_{\mu_{j}}^{-} \leqslant \xi \leqslant \xi_{\mu_{j}}^{+}, \ \eta_{\mu_{j}}^{-} \leqslant \eta \leqslant \eta_{\mu_{j}}^{+} \}.$$
(35)

We turn to the evaluation of the quadrature rules in (34). Formulas (31) and (32) yield

$$\mathscr{G}_{\ell}(\psi_{\mu}) = \|\boldsymbol{b}\|^{2} \boldsymbol{e}_{1}^{\mathrm{T}}(C_{\ell}C_{\ell}^{\mathrm{T}} + \mu I)^{-2}\boldsymbol{e}_{1},$$
(36)

$$\mathscr{R}_{\ell+1}(\psi_{\mu}) = \|\boldsymbol{b}\|^2 \boldsymbol{e}_1^{\mathrm{T}}(\bar{C}_{\ell}\bar{C}_{\ell}^{\mathrm{T}} + \mu I)^{-2} \boldsymbol{e}_1.$$
(37)

We evaluate the Gauss quadrature rule (36) by first determining the solution  $z_{\mu,\ell}$  of the system of equations

$$(\bar{C}_{\ell}\bar{C}_{\ell}^{\mathrm{T}}+\mu I)\boldsymbol{z}=\|\boldsymbol{b}\|\boldsymbol{e}_{1}.$$
(38)

These are the normal equations associated with the least-squares problem

$$\min_{\boldsymbol{z}\in\mathbb{R}^{\ell}}\left\|\begin{bmatrix}\boldsymbol{C}_{\ell}^{T}\\\boldsymbol{\mu}^{1/2}\boldsymbol{I}\end{bmatrix}\boldsymbol{z}-\boldsymbol{\mu}^{-1/2}\|\boldsymbol{b}\|\begin{bmatrix}\boldsymbol{0}\\\boldsymbol{e}_{1}\end{bmatrix}\right\|,\quad\boldsymbol{0}\in\mathbb{R}^{\ell},$$
(39)

and we compute  $z_{\mu,\ell}$  by solving (39). Eldén [8] and Gander [9] describe how (39) can be solved efficiently with the aid of Givens rotations. Note that the matrix  $C_{\ell}$  is independent of the regularization parameter  $\mu$ . Therefore, given this matrix, the Gauss rule (36) can be evaluated in only  $\mathcal{O}(\ell)$ arithmetic floating point operations for each value of  $\mu$ . The scaling factor  $||\boldsymbol{b}||$  in (39) is computed in Algorithm 1. The evaluation of the Gauss–Radau rule (37) can be carried out similarly.

In addition to giving a lower bound for the L-curve, the solution  $z_{\mu,\ell}$  of (39) can be used to determine an approximate solution

$$\mathbf{y}_{\mu,\ell} = U_\ell \mathbf{z}_{\mu,\ell} \tag{40}$$

of (18). The following theorem discusses properties of  $y_{\mu,\ell}$ .

**Theorem 2.5.** Let  $\mu > 0$  be a desired value of the regularization parameter, let  $z_{\mu,\ell}$  solve the linear system of equations (38) and let  $U_{\ell} \in \mathbb{R}^{m \times \ell}$ ,  $u_{\ell+1} \in \mathbb{R}^m$ ,  $C_{\ell} \in \mathbb{R}^{\ell \times \ell}$  and  $\rho_{\ell}, \sigma_{\ell+1} \in \mathbb{R}$  be determined by Algorithm 1. Then  $z_{\mu,\ell}$  solves the Galerkin equations

$$U_{\ell}^{\mathrm{T}}(AA^{\mathrm{T}} + \mu I)U_{\ell}\boldsymbol{z} = U_{\ell}^{\mathrm{T}}\boldsymbol{b},\tag{41}$$

associated with the system (18). Thus, these equations are obtained by projecting the system of equations (18) orthogonally onto the Krylov space (11) and determining an approximate solution (40) of (18) in the same Krylov space. This approximate solution satisfies

$$\|\boldsymbol{y}_{\boldsymbol{\mu},\ell}\|^2 = \mathscr{G}_{\ell}(\boldsymbol{\psi}_{\boldsymbol{\mu}}),\tag{42}$$

$$\|AA^{\mathrm{T}}\boldsymbol{y}_{\mu,\ell} - \boldsymbol{b}\|^{2} = \mu^{2}\mathscr{G}_{\ell}(\psi_{\mu}) + \sigma_{\ell+1}^{2}\rho_{\ell}^{2}\|\boldsymbol{b}\|^{2}|\boldsymbol{e}_{\ell}^{\mathrm{T}}(C_{\ell}C_{\ell}^{\mathrm{T}} + \mu I)^{-1}\boldsymbol{e}_{1}|^{2}.$$
(43)

**Proof.** The formulas (13) and (14) show that the Galerkin equations (41) are equivalent to the system of equations (38). Formula (42) now follows from (40) and (38),

$$\boldsymbol{y}_{\mu,\ell}^{\mathrm{T}}\boldsymbol{y}_{\mu,\ell} = \boldsymbol{z}_{\mu,\ell}^{\mathrm{T}}\boldsymbol{z}_{\mu,\ell} = \|\boldsymbol{b}\|^{2}\boldsymbol{e}_{1}^{\mathrm{T}}(C_{\ell}C_{\ell}^{\mathrm{T}} + \mu I)^{-2}\boldsymbol{e}_{1} = \mathscr{G}_{\ell}(\psi_{\mu}).$$

We turn to the proof of (43). It follows from (13), (14) and  $C_{\ell}e_{\ell} = \rho_{\ell}e_{\ell}$  that

$$AA^{\mathrm{T}}U_{\ell}\boldsymbol{z}_{\mu,\ell} - \boldsymbol{b} = U_{\ell}(C_{\ell}C_{\ell}^{\mathrm{T}}\boldsymbol{z}_{\mu,\ell} - \|\boldsymbol{b}\|\boldsymbol{e}_{1}) + \rho_{\ell}\sigma_{\ell+1}\boldsymbol{u}_{\ell+1}\boldsymbol{e}_{\ell}^{\mathrm{T}}\boldsymbol{z}_{\mu,\ell}.$$
(44)

Substitute  $\mathbf{z}_{\mu,\ell} = \|\mathbf{b}\| (C_{\ell}C_{\ell}^{\mathrm{T}} + \mu I)^{-1} \mathbf{e}_{1}$  into (44) and use the identity

$$M(M + \mu I)^{-1} - I = -\mu(M + \mu I)^{-1},$$

which holds for all matrices  $M \in \mathbb{R}^{\ell \times \ell}$  and scalars  $\mu$  such that  $M + \mu I$  is invertible. We obtain

$$4A^{\mathrm{T}}U_{\ell}\boldsymbol{z}_{\mu,\ell} - \boldsymbol{b} = -\mu \|\boldsymbol{b}\|U_{\ell}(C_{\ell}C_{\ell}^{\mathrm{T}} + \mu I)^{-1}\boldsymbol{e}_{1} + \rho_{\ell}\sigma_{\ell+1}\|\boldsymbol{b}\|\boldsymbol{u}_{\ell+1}\boldsymbol{e}_{\ell}^{\mathrm{T}}(C_{\ell}C_{\ell}^{\mathrm{T}} + \mu I)^{-1}\boldsymbol{e}_{1}.$$

Taking norms shows (43), because the columns of  $U_{\ell}$  are orthonormal as well as orthogonal to the unit vector  $u_{\ell+1}$ .  $\Box$ 

The computation of an approximate solution of (18) may proceed as follows. We increase the number of bidiagonalization steps  $\ell$  in Algorithm 1 until the L-ribbon is sufficiently narrow to allow the determination of the approximate location of the vertex of the L-curve (20), where we use Algorithm 2 to compute approximations of the form (35) of L-ribbons. Plotting these approximations of the L-ribbons helps us determine a value of the regularization parameter, say  $\hat{\mu}$ , that corresponds to a point  $(\phi(||y_{\hat{\mu}}||^2), \phi(||AA^Ty_{\hat{\mu}} - b||^2))$  close to the vertex of the L-curve, where  $y_{\hat{\mu}}$  is defined by (19).

Note that the quantities in the Lanczos decomposition (13) are independent of the regularization parameter  $\mu$ . Therefore they do not have to be recomputed when the value of regularization parameter is changed.

The vector  $y_{\hat{\mu},\ell}$  defined in Theorem 2.5 is an approximate solution of (18). It follows from (42) and (43) that the point  $(\phi(||y_{\hat{\mu},\ell}||^2), \phi(||AA^Ty_{\hat{\mu},\ell} - b||^2))$  is on or above the curve  $\mathbb{L}^-$ . If this point is not sufficiently close to the L-curve, then we increase the number of bidiagonalization steps  $\ell$  in Algorithm 1 until an approximate solution that corresponds to a point sufficiently close to the L-curve is obtained.

#### 2.2. Overdetermined systems

Assume that  $m \ge n$  in (1). Then the standard form of Tikhonov regularization (2) is appropriate. This form has received considerable attention in literature; see e.g., [4] and references therein. A nice survey of solution methods, many of them based on Lanczos bidiagonalization, is presented by Kilmer and O'Leary [21]. An interesting recent approach to locating the vertex of the L-curve is described by Kaufman and Neumaier [20], who combine iterations by the conjugate gradient method for the solution of (2) for a fixed value of the regularization parameter  $\mu$  with a minimization method for computing a value of  $\mu$  that corresponds to a point in the vicinity of the vertex of the L-curve.

This section focuses on the use of an L-ribbon associated with the Tikhonov equations in standard form (2). This ribbon is analogous to the L-ribbon for the system (18) introduced in Section 2.1.2. We refer to [6] for details. Here we only outline the computations required and discuss reorthogonalization.

Assume that the decompositions (13) are available. We seek to determine an approximate solution of (2) of the form

$$\mathbf{x}_{\mu,\ell} = V_\ell \mathbf{z}_{\mu,\ell} \tag{45}$$

and require  $z_{\mu,\ell} \in \mathbb{R}^{\ell}$  to solve the Galerkin equation

$$V_{\ell}^{\mathrm{T}}(A^{\mathrm{T}}A + \mu I)V_{\ell}\boldsymbol{z} = V_{\ell}^{\mathrm{T}}A^{\mathrm{T}}\boldsymbol{b},\tag{46}$$

which is obtained by projecting the linear system of equations (2) orthogonally onto the Krylov space (10) and determining an approximate solution (45) of (2) in the same space.

Identities (13) allow us to simplify Eq. (46) to

$$(\bar{\boldsymbol{C}}_{\ell}^{\mathrm{T}}\bar{\boldsymbol{C}}_{\ell}+\boldsymbol{\mu}\boldsymbol{I})\boldsymbol{z}=\|\boldsymbol{b}\|\bar{\boldsymbol{C}}_{\ell}^{\mathrm{T}}\boldsymbol{e}_{1}.$$
(47)

These are the normal equations associated with the least-squares problem

$$\min_{\boldsymbol{z}\in\mathbb{R}^{\ell}}\left\|\begin{bmatrix}\bar{C}_{\ell}\\\boldsymbol{\mu}^{1/2}\boldsymbol{I}\end{bmatrix}\boldsymbol{z}-\|\boldsymbol{b}\|\boldsymbol{e}_{1}\right\|.$$
(48)

We compute the solution  $z_{\mu,\ell}$  of (46) by solving (48), and then determine the Galerkin solution (45) of (2).

The solution of slight modifications of the least-squares problem (48) yields the values of Gauss and Gauss–Radau quadrature rules for determining an L-ribbon that contains the L-curve (6). The point  $(\phi(||\mathbf{x}_{\mu,\ell}||^2), \phi(||A\mathbf{x}_{\mu,\ell} - \mathbf{b}||^2))$  can be shown to be in this L-ribbon; see [6] for details.

In order to secure that Gauss quadrature rules can be applied to bound the L-curve, we reorthogonalize the columns of  $V_{\ell}$  in the decomposition (13). Note that all columns of  $V_{\ell}$  have to be stored in computer memory until the approximate solution  $\mathbf{x}_{\mu,\ell}$  of (2) has been computed from (45), unless we compute the matrix  $V_{\ell}$  twice. Only a few of the columns of  $U_{\ell+1}$  have to be stored in computer memory simultaneously. Example 4.1 of Section 4 compares the L-ribbons and the computed approximate solutions obtained with and without reorthogonalization of the columns of  $V_{\ell}$ .

## 3. Iterative methods based on the Arnoldi process

Let the matrix A and vector **b** be defined by (1). The Arnoldi process requires that A has the same number of rows and columns. We achieve this by appending rows or columns with zero entries to the given matrix if necessary. Thus, if m > n, then we append m - n columns with zero entries to A to obtain an  $m \times m$  matrix, which we also denote by A. Similarly, when m < n, we append n - m rows with zero entries to A and to the vector **b**. Assume that the matrix A so obtained has k rows and columns and the vector **b** has k rows.

Application of  $\ell$  steps of the Arnoldi process to the matrix A with initial vector **b** yields the decomposition

$$AW_{\ell} = W_{\ell}H_{\ell} + \eta_{\ell+1}\boldsymbol{w}_{\ell+1}\boldsymbol{e}_{\ell}^{\mathrm{T}},\tag{49}$$

where  $W_{\ell} \in \mathbb{R}^{k \times \ell}$ ,  $W_{\ell}^{\mathrm{T}} W_{\ell} = I$ ,  $W_{\ell} e_1 = b/\|b\|$ ,  $w_{\ell+1} \in \mathbb{R}^k$ ,  $W_{\ell}^{\mathrm{T}} w_{\ell+1} = 0$ ,  $\|w_{\ell+1}\| = 1$ ,  $\eta_{\ell+1} \in \mathbb{R}$ and  $H_{\ell} \in \mathbb{R}^{\ell \times \ell}$  is an upper Hessenberg matrix; see, e.g., [11, Chapter 9.4]. We assume that  $\ell$ is sufficiently small so that the decomposition (49) exists. For future reference, we note that the columns of the matrix  $W_{\ell}$  span the Krylov space

$$\mathbb{K}_{\ell}(A, \boldsymbol{b}) = \operatorname{span}\{\boldsymbol{b}, A\boldsymbol{b}, \dots, A^{\ell-1}\boldsymbol{b}\}.$$
(50)

An advantage of the Arnoldi decomposition (49), compared with the Lanczos bidiagonalization (13), is that the former can be determined without using the transpose of A. This is important for problems for which matrix-vector products with the matrix A can be evaluated inexpensively, but matrix-vector products with  $A^{T}$  cannot. This situation may arise when A is defined by the discretization of certain integral or differential operators. Another advantage of the Arnoldi decomposition is that it can be cheaper to compute than the decomposition (13) for fixed  $\ell$ , because the computation
of the Arnoldi decomposition (49) does not require the evaluation of matrix-vector products with the matrix  $A^{T}$ .

This section describes how the decomposition (49) can be used to determine an approximate solution of (2) of the form

$$\mathbf{x}_{\mu,\ell} = W_\ell \mathbf{z}_{\mu,\ell}.\tag{51}$$

Consider the Galerkin equations associated with (2),

$$W_{\ell}^{\mathrm{T}}(A^{\mathrm{T}}A + \mu I)W_{\ell}\boldsymbol{z} = W_{\ell}^{\mathrm{T}}A^{\mathrm{T}}\boldsymbol{b}.$$
(52)

These equations are obtained by projecting the system of equations (2) orthogonally onto the Krylov space (50) and determining an approximate solution (51) of (2) in the same Krylov space. Since the Krylov spaces (50) and (10) differ, the properties of the Galerkin equations (52) and (46) may be different. Theorem 3.1 below sheds some light on the properties of the system of equations (52). Introduce the upper Hessenberg-like matrix

$$\bar{H}_{\ell} = \begin{bmatrix} H_{\ell} \\ \eta_{\ell+1} \boldsymbol{e}_{\ell}^{\mathrm{T}} \end{bmatrix} \in \mathbb{R}^{(\ell+1) \times \ell},\tag{53}$$

where  $H_{\ell}$  and  $\eta_{\ell+1}$  are defined by the Arnoldi decomposition (49), and let  $W_{\ell+1} = [W_{\ell}, w_{\ell+1}] \in \mathbb{R}^{k \times (\ell+1)}$ . Then (49) can be written as

$$AW_{\ell} = W_{\ell+1}\bar{H}_{\ell} \tag{54}$$

and therefore (52) is equivalent to

436

$$(\bar{\boldsymbol{H}}_{\ell}^{\mathrm{T}}\bar{\boldsymbol{H}}_{\ell}+\boldsymbol{\mu}\boldsymbol{I})\boldsymbol{z} = \|\boldsymbol{b}\|\bar{\boldsymbol{H}}_{\ell}^{\mathrm{T}}\boldsymbol{e}_{1}.$$
(55)

**Theorem 3.1.** Let, for  $\ell \ge 1$ , the matrix  $\bar{H}_{\ell}$  be defined by (53) and introduce its singular values  $\sigma_1(\bar{H}_{\ell}) \ge \sigma_2(\bar{H}_{\ell}) \ge \cdots \ge \sigma_\ell(\bar{H}_{\ell}) \ge 0$ . Then

$$\sigma_1(\bar{H}_{\ell+1}) \geqslant \sigma_1(\bar{H}_{\ell}) \geqslant \sigma_2(\bar{H}_{\ell+1}) \geqslant \cdots \geqslant \sigma_\ell(\bar{H}_{\ell+1}) \geqslant \sigma_\ell(\bar{H}_{\ell}) \geqslant \sigma_{\ell+1}(\bar{H}_{\ell+1}).$$
(56)

Let  $\sigma_1(A) \ge \sigma_2(A) \ge \cdots \ge \sigma_k(A)$  denote the singular values of A. Then

$$\sigma_1(A) \ge \sigma_1(H_{\ell+1}), \quad \sigma_{\ell+1}(H_{\ell+1}) \ge \sigma_{\min\{m,n\}}(A).$$
(57)

Assume that A is of rank r and that the Arnoldi decomposition

$$AW_r = W_r H_r \tag{58}$$

exists, where  $W_r \in \mathbb{R}^{k \times r}$ ,  $W_r^T W_r = I$ ,  $W_r \boldsymbol{e}_1 = \boldsymbol{b} / \|\boldsymbol{b}\|$  and  $H_r \in \mathbb{R}^{r \times r}$  is an upper Hessenberg matrix. Then

$$\sigma_j(H_r) = \sigma_j(A), \quad 1 \le j \le r.$$
(59)

**Proof.** We obtain the matrix  $\bar{H}_{\ell+1}$  from  $\bar{H}_{\ell}$  by first appending a zero row to  $\bar{H}_{\ell}$  and then appending a column. The singular values of the matrix do not change when appending a zero row to  $\bar{H}_{\ell}$ . When then appending a column, the singular values of the original and the augmented matrices interlace, see e.g. [11, Corollary 8.6.3], and inequalities (56) follow.

Inequalities (57) follow from (54) and the minimax property of singular values; see [11, Theorem 8.6.1] for the latter. Equalities (59) are a consequence of (58).  $\Box$ 

In general the inequalities in (56) are strict. The properties of the singular values formulated in Theorem 3.1 are analogous to properties of the eigenvalues of the matrices  $\bar{C}_{\ell}^{T}\bar{C}_{\ell}$  in (47), or, equivalently, of the singular values of the matrices  $\bar{C}_{\ell}$ . We therefore would expect that Eqs. (47) and (55) share many numerical properties, however, this issue deserves further investigation.

Eqs. (55) are the normal equations associated with the least-squares problem

$$\min_{\boldsymbol{z}\in\mathbb{R}^{\ell}}\left\|\begin{bmatrix}\bar{H}_{\ell}\\\boldsymbol{\mu}^{1/2}\boldsymbol{I}\end{bmatrix}\boldsymbol{z}-\|\boldsymbol{b}\|\boldsymbol{e}_{1}\right\|.$$
(60)

We compute the solution  $z_{\mu,\ell}$  of (52) by solving (60).

Typically, we would like to solve the least-squares problem (60) for many different values of the regularization parameter  $\mu$ . Therefore, it may be advantageous to first reduce the matrix  $\bar{H}_{\ell}$  to bidiagonal form by application of a judiciously chosen sequence of Givens rotations from the right-hand side and from the left-hand side. This reduction is independent of  $\mu$ . The least-squares problem so obtained is equivalent to (60) and can be solved in only  $\mathcal{O}(\ell)$  arithmetic operations for each value of  $\mu$ , similarly as (39).

For each value of  $\ell \ge 1$ , there is an associated curve

$$\mathbb{L}_{\ell} = \{ (\phi(\|\boldsymbol{x}_{\mu,\ell}\|^2), \phi(\|A\boldsymbol{x}_{\mu,\ell} - \boldsymbol{b}\|^2)) \colon \mu > 0 \}.$$
(61)

It follows from Proposition 1.2 that the curves  $\mathbb{L}_{\ell}$ ,  $\ell \ge 1$ , are above or coincide with the L-curve (6). The curves  $\mathbb{L}_{\ell}$  converge to the L-curve as  $\ell$  increases. Since  $\|\mathbf{x}_{\mu,\ell}\|^2 = \|\mathbf{z}_{\mu,\ell}\|^2$  and  $\|A\mathbf{x}_{\mu,\ell} - \mathbf{b}\|^2 = \|\bar{H}_{\ell}\mathbf{z}_{\mu,\ell} - \|\mathbf{b}\|\mathbf{e}_1\|^2$ , it is quite inexpensive to compute points on the curve  $\mathbb{L}_{\ell}$ . This suggests that we determine points on a sequence of curves  $\mathbb{L}_{\ell}$  for increasing values of  $\ell$  in order to determine the location of the vertex of the L-curve. Let  $\hat{\mu}$  be the value of the regularization parameter so determined. The approximate solution  $\mathbf{x}_{\ell,\hat{\mu}}$  of (2) is then computed using (60) and (51) with  $\mu = \hat{\mu}$ .

#### 4. Computed examples

All computations were carried out using Matlab on a Sun Ultra workstation with unit roundoff  $\varepsilon = 2^{-52} \approx 2 \times 10^{-16}$  except for some computations for Example 4.4. The first three examples are concerned with fairly small test problems and illustrate various aspects of the solution methods discussed. Our fourth example is a fairly large computerized tomography problem, in which an object is reconstructed from its X-ray projections. All plotted L-curves and L-ribbons are for the function  $\phi(t) = \frac{1}{2} \log_{10}(t)$ .

Example 4.1. We consider the solution of the overdetermined system

$$A\boldsymbol{x} = \boldsymbol{b},\tag{62}$$

where the matrix  $A \in \mathbb{R}^{400 \times 200}$  is defined by its singular value decomposition

$$A = U_{400} D V_{200}^{\mathrm{T}}.$$
 (63)

The matrices in this decomposition are determined as follows. Let the matrix  $C_n = [c_{jk}]_{j,k=1}^n$  with entries

$$c_{jk} = \exp\left(\pi \frac{2j-1}{4n-2} \cos\left(\pi \frac{2k-1}{2n-1}\right)\right)$$

have the singular value decomposition  $C_n = U_n D_n V_n^{\mathrm{T}}$ . The matrix  $U_{400}$  in (63) is the left orthogonal matrix in the singular value decomposition of  $C_{400}$  and the matrix  $V_{200}^{\mathrm{T}}$  in (63) is the right orthogonal matrix is the singular value decomposition of  $C_{200}$ . The columns of  $U_{400}$  have the property that the number of sign changes in the sequence  $\{e_j^{\mathrm{T}}U_{400}e_j\}_{j=1}^{400}$  increases with k. The columns of  $V_{200}$  have the same property. This property is typical of matrices obtained by discretizing Fredholm integral equations of the first kind. The entries of the matrix  $D = [d_{jk}] \in \mathbb{R}^{400 \times 200}$  in (63) are given by

$$d_{jk} = \begin{cases} \exp(-\frac{2}{5}(j-1)), & j = k, \\ 0, & j \neq k. \end{cases}$$

Thus, the matrix A has condition number  $d_{11}/d_{200,200} = 4 \cdot 10^{34}$  and therefore is numerically singular.

Let the right-hand side vector in (62) be of the form  $\mathbf{b} = \hat{\mathbf{b}} + \mathbf{e}$ , where  $\hat{\mathbf{b}}$  is thought of as the "exact" right-hand side vector and  $\mathbf{e}$  is thought of as an "error" or "noise" vector. The vector  $\hat{\mathbf{b}}$  is generated so that the linear system of equations

$$A\mathbf{x} = \hat{\mathbf{b}} \tag{64}$$

is consistent. Specifically, we let x be a unit vector with normally distributed random entries with zero mean, and compute  $\hat{b} = Ax$ .

The "noise" vector  $\boldsymbol{e}$  has normally distributed random entries with zero mean. The variance of the entries of  $\boldsymbol{e}$  is chosen so that  $\|\boldsymbol{e}\|/\|\hat{\boldsymbol{b}}\| = 1 \times 10^{-2}$ . We refer to the quotient  $\|\boldsymbol{e}\|/\|\hat{\boldsymbol{b}}\|$  as the noise level.

Fig. 1(a) shows points on the L-curve, marked by "o", for the values

$$\mu_j = 1 \cdot 10^{-7 + (j-1)/4}, \quad 1 \le j \le 22, \tag{65}$$

of the regularization parameter. Small values of *j* correspond to small  $\mu_j$  and give Galerkin solutions  $\mathbf{x}_{\mu_j,\ell}$  of (2), defined by (45), of large norm. The associated discrepancies  $\mathbf{d}_{\mu_j,\ell} = A\mathbf{x}_{\mu_j,\ell} - \mathbf{b}$  are of small norm. Fig. 1(a) also shows 22 rectangles that approximate the L-ribbon. These rectangles are determined by Algorithm 2 with  $\ell = 15$  and are associated with the values (65) of the regularization parameter. The Lanczos bidiagonalization is computed by Algorithm 1 and the columns of the matrix  $V_{15}$  are reorthogonalized in order to secure their numerical orthogonality.

We note that rectangles associated with small values  $\mu_j$  of the regularization parameter are larger than rectangles associated with large  $\mu_j$ . This depends on that the support of the measure  $\omega(t)$ defined by (27) is on the non-negative real axis and the integrand  $\psi_{\mu}$  has a pole at  $t = -\mu$ . Thus, the larger  $\mu > 0$  is, the further away the singularity of the integrand is from the support of the measure  $\omega(t)$  and the more accurate the Gauss and Gauss–Radau rules are. The "vertex" of the L-curve is seen to correspond to roughly the value  $\mu_{14} = 1 \times 10^{-3.75}$  of the regularization parameter; this is easier to see when the figure is enlarged.

The vertices  $\xi_j^-, \eta_j^+$  of the rectangles generated by Algorithm 2 for  $\ell = 15$  are marked by "x" in Fig. 1(a). It is shown in [6, Theorem 5.1] that the Galerkin solutions  $\mathbf{x}_{\mu_j,\ell}$  of (2) determined by (45) and (46) correspond to these vertices. When the rectangular region is "tiny" only the vertex "x" is visible.



Fig. 1. Example 4.1: (a) L-ribbon for 15 bidiagonalization steps with reorthogonalization of the columns of  $V_{15}$ . (b) Relative error  $||\mathbf{x} - \mathbf{x}_{\mu,15}||/|\mathbf{x}||$  as a function of  $\mu$ , where  $\mathbf{x}_{\mu,15}$  denotes the Galerkin solution (45) of the linear system (62) and  $\mathbf{x}$  is the solution of system (64) with noise-free right-hand side.

Fig. 1(b) displays the 10-logarithm of the relative error  $||\mathbf{x}_{\mu,15} - \mathbf{x}||/||\mathbf{x}||$  in the computed Galerkin solution  $\mathbf{x}_{\mu,15}$  as a function of  $\log_{10} \mu$ , where  $\mathbf{x}$  denotes the solution of the linear system (64) with the noise-free right-hand side. In particular, Fig. 1(b) shows that the value  $\mu = \mu_{14}$  yields  $\log_{10}(||\mathbf{x}_{\mu,15} - \mathbf{x}||/||\mathbf{x}||) = 1.5 \times 10^{-2}$ . This value of  $\log_{10}(||\mathbf{x}_{\mu,15} - \mathbf{x}||/||\mathbf{x}||)$  can be seen to be fairly close to minimal.

Fig. 2 is analogous to Fig. 1 and displays the computed L-ribbon and relative error in the computed approximate solutions  $\mathbf{x}_{\mu,15}$  when 15 bidiagonalization steps without reorthogonalization are carried out. The rectangles of the L-ribbon in Fig. 2(a) are much larger than those of Fig. 1(a), and the relative error of the computed approximate solution displayed in Fig. 2(b) looks quite different from the error shown in Fig. 1(b). The "vertex" of the L-curve of Fig. 2(a) is at roughly  $\mu_{16} = 1 \times 10^{-3.25}$  and Fig. 2(b) shows that for  $\mu = \mu_{16}$ , we have  $\log_{10}(||\mathbf{x}_{\mu,15} - \mathbf{x}||/||\mathbf{x}||) = -1.3 \times 10^{-2}$ .

Comparing Figs. 1(a) and 2(a) shows that Gauss quadrature rules give tighter bounds when reorthogonalization is carried out. Reorthogonalization may be crucial for problems with a small noise level, because for these problems the desired value of the regularization parameter is typically small. The effect of the loss of orthogonality of the columns of the matrix  $V_{\ell}$  on the bounds determined by the Gauss rules requires further investigation.

The "vertices" in Figs. 1(a) and 2(a) correspond to different values of the regularization parameter  $\mu$ . Figs. 1(b) and 2(b) show the relative error in the computed approximate solution  $x_{\mu_{14,15}}$  determined with reorthogonalization to be somewhat smaller than the relative error in the computed approximate solution  $x_{\mu_{16,15}}$  determined without reorthogonalization.

Fig. 3 differs from Fig. 2 only in that the number of bidiagonalization steps has been increased from 15 to 22. No reorthogonalization is carried out. As expected, the Gauss rules give tighter bounds when the number of bidiagonalization steps is increased. Fig. 3(a) suggests that the L-curve has its "vertex" at  $\mu = \mu_{15} = 1 \times 10^{-3.5}$ . Fig. 3(b) shows that for this value of  $\mu$ , the approximate solution  $\mathbf{x}_{\mu,22}$  satisfies  $\log_{10}(||\mathbf{x}_{\mu,15} - \mathbf{x}||/||\mathbf{x}||) = -1.4 \times 10^{-2}$ . Thus, the relative error is smaller than



Fig. 2. Example 4.1: (a) L-ribbon for 15 bidiagonalization steps without reorthogonalization. (b) Relative error  $||\mathbf{x} - \mathbf{x}_{\mu,15}||/||\mathbf{x}||$  as a function of  $\mu$ , where  $\mathbf{x}_{\mu,15}$  denotes the Galerkin solution (45) of linear system (62) and  $\mathbf{x}$  is the solution of system (64) with noise-free right-hand side.



Fig. 3. Example 4.1: (a) L-ribbon for 22 bidiagonalization steps without reorthogonalization. (b) Relative error  $||\mathbf{x} - \mathbf{x}_{\mu,22}||/||\mathbf{x}||$  as a function of  $\mu$ , where  $\mathbf{x}_{\mu,22}$  denotes the Galerkin solution (45) of linear system (62) and  $\mathbf{x}$  is the solution of system (64) with noise-free right-hand side.

the relative error obtained with 15 bidiagonalization steps without reorthogonalization, but larger than the relative error achieved with 15 bidiagonalization steps with reorthogonalization.

We remark that when the columns of both  $U_{\ell+1}$  and  $V_{\ell}$  are reorthogonalized, or when the columns of  $U_{\ell+1}$  but not the columns of  $V_{\ell}$  are reorthogonalized, the graphs obtained are identical with the graph of Fig. 1.



Fig. 4. Example 4.2: L-ribbons for underdetermined systems: (a) noise level  $1 \times 10^{-3}$ , (b) noise level  $1 \times 10^{-2}$ .

In summary, the figures of this example show the Gauss rules to give tighter bounds and the relative error in the computed approximate solution to be somewhat smaller when the columns of the matrix  $V_{\ell}$  are reorthogonalized than when they are not.  $\Box$ 

**Example 4.2.** We are concerned with the solution of underdetermined systems of equations  $A^T \mathbf{x} = \mathbf{b}$ , where the matrix A is the same as in Example 4.1. The right-hand side vector is constructed in the same way as in Example 4.1. Fig. 4 shows L-ribbons for different noise levels. All graphs are for  $\ell = 20$  and  $\mu_j = 1 \times 10^{-8+(j-1)/5}$ ,  $1 \le j \le 40$ .

Let  $y_{\mu_j,\ell}$  denote the Galerkin solution (40) of (18) for  $\mu = \mu_j$  discussed in Theorem 2.5 and let  $d_{\mu_j,\ell} = AA^T y_{\mu_j,\ell} - b$  be the associated discrepancy. The points  $(\phi(||y_{\mu_j,\ell}||^2), \phi(||d_{\mu_j,\ell}||^2))$  are marked by "x" in Fig. 4. The figure also shows the rectangles (35) of the L-ribbon and illustrates that the points "x" do not always lie in the L-ribbon, in agreement with Theorem 2.5. In addition Fig. 4 displays points on the L-curve (20) associated with the values  $\mu_j$  of the regularization parameter. These points are marked by "o".

Fig. 4 illustrates the effect of changing the noise level. The location of the "vertex" is quite easily determined in both graphs. The graphs show that an increase in the noise level results in an increase of the value of the regularization parameter determined by the L-curve method.  $\Box$ 

Example 4.3. We solve the integral equation of the first kind

$$\int_{0}^{\pi} \exp(s\cos(t))x(t) \, \mathrm{d}t = 2\frac{\sinh(s)}{s}, \quad 0 \le s \le \frac{\pi}{2},\tag{66}$$

considered by Baart [1]. Eq. (66) is discretized by a Galerkin method with orthonormal box functions. We used the Matlab code by Hansen [15] for computing the nonsymmetric matrix  $A \in \mathbb{R}^{400 \times 400}$  and the exact, i.e., noise-free, right-hand side vector  $\hat{\boldsymbol{b}} \in \mathbb{R}^{400}$ . The matrix A so obtained is numerically singular.



Fig. 5. Example 4.3:  $\mathbb{L}_{\ell}$  curves for  $\ell = 4, 5, 6, 7, 8$  using the Arnoldi decomposition.

We generate a "noise" vector e as described in Example 4.1 and solve the linear system of equations (2) with right-hand side vector  $\mathbf{b} = \hat{\mathbf{b}} + \mathbf{e}$ , where  $\mathbf{e}$  is scaled to give noise level  $\|\mathbf{e}\| / \|\hat{\mathbf{b}}\| = 1 \times 10^{-3}$ .

Fig. 5 shows  $\mathbb{L}_{\ell}$ -curves (61) obtained by Arnoldi decompositions (49) for  $4 \leq \ell \leq 8$ . We use the values  $\mu_j = 1 \times 10^{-8+(j-1)/2}$ ,  $1 \leq j \leq 9$ , of the regularization parameter. The  $\mathbb{L}_{\ell}$ -curves converge to the L-curve (6) as  $\ell$  increases. Points on the  $\mathbb{L}_{\ell}$ -curves are marked by "\*" and connected by straight lines; points on the L-curve are marked by "o" and connected by straight lines. The points "\*" on the curves  $\mathbb{L}_7$  and  $\mathbb{L}_8$  cannot be distinguished. They are very close to the points "o" on the L-curve.

Fig. 6 compares the  $\mathbb{L}_6$ -curve obtained by Arnoldi decomposition to the  $\mathbb{L}_4$ -curve obtained by Lanczos bidiagonalization (13) for the values  $\mu_j = 1 \times 10^{-8+(j-1)/2}$ ,  $1 \le j \le 9$  of the regularization parameter.

We remark that the  $\mathbb{L}_8$ -curve for Arnoldi decomposition (shown in Fig. 5) and  $\mathbb{L}_4$ -curve for Lanczos bidiagonalization require the same number of matrix-vector product evaluations with the matrix A or  $A^T$  and their graphs are identical to plotting accuracy.

**Example 4.4.** We apply the L-ribbon method of Section 2.1 to computerized X-ray tomography. This is a technique for representing a three-dimensional object by means of its two-dimensional cross sections or slices. In this example we show the reconstruction of a slice from projection data.

The region that contains the slice is discretized by a Cartesian grid of  $512 \times 512$  square picture elements (pixels). To each one of 512 X-ray emitters there is an associated detector. The 512 rays between the emitters and detectors form parallel lines of angle  $\theta$  with the horizontal axis. Measurements are made for 90 equidistant angles  $\theta \in [0, \pi)$ .

Let  $b_i$  denote the measured total attenuation of the *i*th ray when it traverses the object at a certain angle. In the present example there are  $512 \times 90 = 46080$  measured attenuations. In the absence of



Fig. 6. Example 4.3:  $\mathbb{L}_{\ell}$  curves: Arnoldi decomposition for  $\ell = 6$ , Lanczos bidiagonalization for  $\ell = 4$ .

measurement errors, the value of  $b_i$  would be the line integral of the unknown X-ray absorption function along the path of the ray. In our computed example, the  $b_i$  are contaminated by errors that give a noise level of  $1 \times 10^{-1}$ . The purpose of the computations is to determine an approximation of the X-ray absorption function from the quantities  $b_i$  and knowledge of the angles of the rays. We display the computed approximation of the X-ray absorption function and refer to it as the reconstructed image of the slice.

The X-ray absorption function is assumed to be constant throughout each pixel. Let  $x_j$  denote the value of the absorption function at pixel *j*. There are  $512^2 = 262144$  pixels. For  $1 \le i \le 46080$  and  $1 \le j \le 262144$ , let  $a_{ij}$  be equal to 1 if the *i*th ray at the given angle intersects the *j*th pixel. Then  $a_{ij}x_j$  represents the attenuation of the *i*th ray by the *j*th pixel, and  $\sum_{j=1}^{262144} a_{ij}x_j$  approximates the total attenuation of the *i*th ray.

Introduce the  $46080 \times 262144$  matrix  $A = [a_{ij}]$  and the right-hand side vector  $\boldsymbol{b} = [b_1, b_2, \dots, b_{46080}]^T$ . The linear system of equations obtained (1) is underdetermined. Therefore we use the solution method described in Section 2.1. The matrix A is referred to as the projection matrix and the solution  $\boldsymbol{x} = [x_1, x_2, \dots, x_{262144}]^T$  as the image vector; see [2,19] for details on image reconstruction from projections.

We show the result of a reconstruction of a slice of a human head. Fig. 7 shows the "original" image that we wish to reconstruct by computerized tomography. This image is represented by  $512 \times 512$  pixels. Note that in "real" applications of computerized tomography the original image is not available; we only know the entries of the matrix *A* and right-hand side *b*. We display the original image to allow comparison with the reconstructed image shown in Fig. 9.

Algorithm 2 is used with  $\ell = 200$  Lanczos steps with reorthogonalization of the columns of the matrix  $U_{\ell+1}$ . Fig. 8 shows the computed L-ribbon. Specifically, the figure displays rectangles (35)



Fig. 7. Example 4.4: Original image.



Fig. 8. Example 4.4: Reconstructed image,  $\ell = 200$ ,  $\mu = 0.06$ , noise level  $1 \times 10^{-1}$ .



Fig. 9. Example 4.4: L-ribbon for the human head section reconstruction;  $\ell = 200$ .

for  $\mu_j = 2 \times 10^{-3+(j-1)/2}$ ,  $1 \le j \le 16$ . The computed L-ribbon is seen to have a "vertex" at roughly  $\mu_4 = 0.06$ .

Let  $\mathbf{y}_{\mu_j,\ell}$  denote the Galerkin solution (40) for  $\mu = \mu_j$  and let  $\mathbf{d}_{\mu_j,\ell} = AA^T \mathbf{y}_{\mu_j,\ell} - \mathbf{b}$  be the associated discrepancy. The points  $(\phi(||\mathbf{y}_{\mu_j,\ell}||^2), \phi(||\mathbf{d}_{\mu_j,\ell}||^2))$  are marked by "x" in Fig. 8. The figure shows that for small values  $\mu_j$  the rectangles (35) are large and the Galerkin solution is quite far away from the L-ribbon. Fig. 9 displays the reconstructed image from projections for  $\ell = 200$  and the value  $\mu_4$  of the regularization parameter.

We remark that the location of "vertex" of the L-curve in Fig. 8 is not clearly discernible by visual inspection and this makes it difficult to accurately determine a value of the regularization parameter which corresponds to a point at or near the "vertex" of the L-curve. Visual inspection of Fig. 8 leads us to choose the value  $\mu_4 = 0.06$  of the regularization parameter. However, since the "vertex" of the L-curve is not very pronounced, one may consider choosing the value  $\mu_5 = 0.2$  instead. It turns out that the reconstructed images obtained with the values  $\mu_4$  and  $\mu_5$  of the regularization parameter look essentially the same. We conclude that the L-ribbon provides a good guideline for how to choose an appropriate value of the regularization parameter. Since the images associated with the values  $\mu_4$  or  $\mu_5$  of the regularization parameter do not differ much, we only show the former.  $\Box$ 

#### Acknowledgements

We would like to thank Per Christian Hansen for comments on a previous version of the paper.

#### References

- M.L. Baart, The use of auto-correlation for pseudo-rank determination in noisy ill-conditioned least-squares problems, IMA J. Numer. Anal. 2 (1982) 241–247.
- [2] M. Bertero, P. Boccacci, Introduction to Inverse Problems in Imaging, Institute of Physics Publ., Bristol, 1998.
- [3] Å. Björck, A bidiagonalization algorithm for solving large and sparse ill-posed systems of linear equations, BIT 18 (1988) 659–670.
- [4] Å. Björck, Numerical Methods for Least Squares Problems, SIAM, Philadelphia, 1996.
- [5] Å. Björck, E. Grimme, P. Van Dooren, An implicit shift bidiagonalization algorithm for ill-posed systems, BIT 34 (1994) 510–534.
- [6] D. Calvetti, G.H. Golub, L. Reichel, Estimation of the L-curve via Lanczos bidiagonalization, BIT 39 (1999) 603–619.
- [7] D. Calvetti, L. Reichel, F. Sgallari, G. Spaletta, A regularizing Lanczos iteration method for underdetermined linear systems, J. Comput. Appl. Math. 20 (1999) 101–120.
- [8] L. Eldén, Algorithms for the regularization of ill-conditioned least squares problems, BIT 17 (1977) 134-145.
- [9] W. Gander, Least squares with a quadratic constraint, Numer. Math. 36 (1981) 291–307.
- [10] G.H. Golub, G. Meurant, Matrices, moments and quadrature, in: D.F. Griffiths, G.A. Watson (Eds.), Numerical Analysis 1993, Longman, Essex, England, 1994, pp. 105–156.
- [11] G.H. Golub, C.F. Van Loan, Matrix Computations, 3rd Edition, Johns Hopkins University Press, Baltimore, 1996.
- [12] G.H. Golub, U. von Matt, Tikhonov regularization for large scale problems, in: G.H. Golub, S.H. Lui, F. Luk, R. Plemmons (Eds.), Workshop on Scientific Computing, Springer, New York, 1997, pp. 3–26.
- [13] M. Hanke, Limitations of the L-curve method in ill-posed problems, BIT 36 (1996) 287–301.
- [14] P.C. Hansen, Analysis of discrete ill-posed problems by means of the L-curve, SIAM Rev. 34 (1992) 561-580.
- [15] P.C. Hansen, Regularization tools: a Matlab package for analysis and solution of discrete ill-posed problems, Numer. Algorithms 6 (1994) 1–35. Software is available in Netlib at http://www.netlib.org.
- [16] P.C. Hansen, Rank-Deficient and Discrete Ill-Posed Problems, SIAM, Philadelphia, 1998.
- [17] P.C. Hansen, The L-curve and its use in the numerical treatment of inverse problems, in: P. Johnston (Ed.), Advances in Biomedicine, Vol. 3, WIT Press, in press.
- [18] P.C. Hansen, D.P. O'Leary, The use of the L-curve in the regularization of discrete ill-posed problems, SIAM J. Sci. Comput. 14 (1993) 1487–1503.
- [19] G.T. Herman, Image Reconstruction from Projections, Academic Press, New York, 1980.
- [20] L. Kaufman, A. Neumaier, Regularization of ill-posed problems by envelope guided conjugate gradients, J. Comput. Graph. Statist. 6 (1997) 451–463.
- [21] M. Kilmer, D.P. O'Leary, Choosing regularization parameters in iterative methods for ill-posed problems, Report, Department of Computer Science, University of Maryland, 1998.
- [22] C.L. Lawson, R.J. Hanson, Solving Least Squares Problems, SIAM, Philadelphia, PA, 1995. First published by Prentice-Hall, 1974.
- [23] C.C. Paige, M.A. Saunders, LSQR: An algorithm for sparse linear equations and sparse least squares, ACM Trans. Math. Software 8 (1982) 43–71.
- [24] T. Reginska, A regularization parameter in discrete ill-posed problems, SIAM J. Sci. Comput. 17 (1996) 740-749.
- [25] C.R. Vogel, Non-convergence of the L-curve regularization parameter selection method, Inverse Problems 12 (1996) 535–547.



Journal of Computational and Applied Mathematics 123 (2000) 447-465

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.nl/locate/cam

# 

Dianne P. O'Leary\*

Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, USA

Received 27 May 1999

#### Abstract

The efficiency and effectiveness of most optimization algorithms hinges on the numerical linear algebra algorithms that they utilize. Effective linear algebra is crucial to their success, and because of this, optimization applications have motivated fundamental advances in numerical linear algebra. This essay will highlight contributions of numerical linear algebra to optimization, as well as some optimization problems encountered within linear algebra that contribute to a symbiotic relationship. © 2000 Elsevier Science B.V. All rights reserved.

# 1. Introduction

The work in any continuous optimization algorithm neatly partitions into two pieces: the work in acquiring information through evaluation of the function and perhaps its derivatives, and the overhead involved in generating points approximating an optimal point. More often than not, this second part of the work is dominated by linear algebra, usually in the form of solution of a linear system or least-squares problem and updating of matrix information.

Thus, members of the optimization community have been *consumers* of linear algebra research, and their needs have set some research directions for the computational linear algebra community. Recently, this relationship has entered a new phase in which optimization problems arising in linear algebra are also setting research agendas for the optimization community.

This essay will highlight the advances in numerical linear algebra that contributed to this symbiotic relationship. First, in Section 2 we review the modeling problems that give rise to linear algebra problems. Least-squares modeling is the subject of Section 3. We turn our attention to the linear algebra of unconstrained optimization problems in Section 4, and then review the simplex method for

 $<sup>^{\</sup>diamond}$  This work was completed at the Departement Informatik, ETH Zürich, Switzerland. This work was also supported in part by the US National Science Foundation under Grant CCR-97-32022.

<sup>\*</sup> Corresponding author.

E-mail address: oleary@cs.umd.edu (D.P. O'Leary).

linear programming in Section 5. Section 6 discusses linear algebra problems arising in interior point methods. Nonlinear problems are briefly considered in Section 7. Section 8 concerns linear algebra problems giving rise to optimization, and Section 9 discusses computational issues in optimization. We summarize our survey in Section 10.

# 2. Linear and quadratic models

The modeling of complex phenomena in science and economics by linear and quadratic models is ubiquitous. It is motivated by the Taylor series expansion of a thrice continuously differentiable function  $f : \mathbb{R}^n \to R$  as

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2}(x - x_0)^{\mathrm{T}} f''(x_0)(x - x_0) + \mathrm{O}(||x - x_0||^3),$$

as well as by the relative ease of handling these models rather than fully-nonlinear ones. Often the full nonlinearity of f is neglected in the modeling process, either because the simpler models yield sufficient accuracy or because the modeling process yields insufficient information about the higher order coefficients.

We often determine the coefficients in a linear model by obtaining the "best-possible" fit to experimental data. The coefficients can be highly dependent on our way of measuring "best." In general, given a model M(t,z) of some function y(t), and data  $(t_i, y_i)$ , i = 1, ..., m, we try to determine the model coefficients  $z \in \mathscr{Z} \subseteq \mathscr{R}^p$  to minimize the norm of the residual vector, whose entries are

 $r_i = y_i - M(t_i, z), \quad i = 1, ..., m.$ 

Common choices of the norm are the 1-norm or infinity-norm, leading to linear programming problems (See Section 5) or the 2-norm, leading to a linear least-squares problem (See Section 3). Narula [68] discusses solution of these various regression problems. If the set  $\mathscr{Z}$  is something other than  $\mathbb{R}^{p}$ , then there are constraints on the minimization problem.

Thus, modeling of physical phenomena leads to optimization problems, but, conversely, algorithms for optimization often lead to linear and quadratic modeling. For instance, an objective function f(x) might be locally modeled as a quadratic function in algorithms such as sequential quadratic programming. As another example, we often temporarily replace a constraint by a local linear model in order to make a subproblem easier to solve (See Section 7).

Perhaps the oldest use of quadratic models to solve nonlinear problems is the iteration of Newton for minimizing a function or finding a zero of a system of nonlinear equations. At each step in the iteration, we construct a quadratic model of the function (or a linear model of the system of equations) and use that model to generate a step in the direction of a better solution. A wonderful survey of Newton's method is given in [96], and we consider this method in Section 4.

### 3. Least squares

Consider the modeling problem

 $\min \|Mz - y\|_2,$ 

448

where  $M \in \mathbb{R}^{m \times n}$ ,  $z \in \mathbb{R}^n$ , and  $y \in \mathbb{R}^m$ . This *linear least-squares problem* was shown by Gauss [39] to produce the z that yields the best linear unbiased estimator of any function  $c^T z_{true}$  whenever the errors in y have mean zero and variance  $\sigma^2 I$ .

The oldest algorithms for solving the linear least-squares problem can be viewed as applying direct or iterative methods to solve the *normal equations* 

$$M^{\mathrm{T}}Mz = M^{\mathrm{T}}y,$$

obtained by setting the derivative of (1) to zero.

Within the past 50 years, advances in the solution of least-squares problems have been of three types: analysis of sensitivity and stability, development of computational tools, and consideration of problem variants.

The lucid textbook by Björck [8] is the definitive reference on the entire subject of numerical solution of least squares problems, and we recommend it for exposition and further references. Higham [48] is an alternate source for the history of sensitivity analysis for these problems.

# 3.1. Sensitivity and stability of least-squares problems

Important contributions to the study of sensitivity of least-squares problems have been made in recent years.

Wedin [94, Theorem 5.1], studied the normwise perturbation of z and the residual r = y - Mzwhen M is perturbed, showing that if the relative perturbations in M and y are less than  $\varepsilon$ , and if the condition number  $\kappa_2(M)$  (the ratio of its largest to its smallest singular value) satisfies  $\kappa_2(M)\varepsilon < 1$ , then

$$\frac{\|z-\hat{z}\|}{\|z\|} \leq \frac{\kappa_2(M)\varepsilon}{1-\kappa_2(M)\varepsilon} \left(2+(\kappa_2(M)+1)\frac{\|r\|_2}{\|M\|_2\|z\|_2}\right),$$
$$\frac{\|r-\hat{r}\|}{\|y\|} \leq (1+2\kappa_2(M))\varepsilon.$$

This result says that if the residual is small, then perturbations are proportional to  $\kappa_2(M)$ , but if the residual is large, then perturbations proportional to  $\kappa_2(M)^2$  might be seen, and that is indeed the case.

Further analysis can be found in [8, Chapter 1], including component-wise bounds on the error [4].

#### 3.2. Computational tools for least-squares problems

The main computational algorithm for least-squares solves the problem by using the QR factorization of the matrix M into the product of a matrix  $Q \in \mathbb{R}^{m \times n}$  with orthogonal columns, and an upper triangular matrix  $R \in \mathbb{R}^{n \times n}$ . Use of this tool was first proposed by Golub [41], but great attention has been given to the relative advantages of factorization using Householder reflections, Givens rotations, or modified Gram–Schmidt [8, Section 2.4]. The first two alternatives were known to have similar desirable error properties, and modified Gram–Schmidt was finally shown stable in a paper of Björck and Paige [10] by exploiting the fact, known to many early practitioners such as Sheffield, that modified Gram-Schmidt is numerically equivalent to Householder QR applied to the matrix

$$\begin{bmatrix} 0\\ M\end{bmatrix}.$$

If the problem is difficult in the sense that M is ill-conditioned, then more refined tools are needed. The QR factorization with column pivoting [41] can be used to try to identify the most linearly independent columns first and perhaps construct a model of reduced size; see [18] for a survey of such rank-revealing QR factorization methods. This is not foolproof, however, and the singular value decomposition [42, Section 2.5] is a more reliable (and more expensive) factorization algorithm for identifying dependencies; see Stewart [85] for historical remarks on the SVD.

The LU factorization of M can also be used to solve least-squares problems [77], but its use is not common except when the matrix M is sparse, with many zero elements. In that case, the QR factors may be quite dense, due to creation of nonzeros in the course of the factorization. To minimize this fill-in, it is important to use the best algorithms for reordering the rows and columns of the matrix [29] before factorization.

The normal equations can be solved by Cholesky factorization into the product of a lower triangular matrix times its transpose, but if the problem is large and sparse, then reordering strategies should again be used to minimize fill [40].

An alternate to factorization for large sparse problems is the use of iterative methods. The preconditioned conjugate gradient algorithm [42] can be used to solve (8), and row-action methods [17] and other specialized methods such as CGLS and LSQR avoid forming the normal equations [8, Chapter 7].

# 3.3. Variants of least-squares problems

Often the matrix M has special structure that can be exploited in order to make solution of the least-squares problem more efficient. One example is the matrix that arises from fitting polynomials using the power basis and equally spaced data points. The resulting matrix for the normal equations, a Vandermonde matrix, has beautiful structure but is quite ill-conditioned [11,9,27,47]. A second example is the band matrix structure that results from fitting functions whose support is local [80,23]. Wavelet [20] and Fourier bases often give matrices with small displacement rank [51] again leading to efficient solution algorithms [86,24,67,44,76].

Some models give rise to nonlinear least-squares problems

$$\min \|r(z)\|,$$

where  $r: \mathbb{R}^n \to \mathbb{R}^m$ . These are usually solved by Newton variants discussed in Section 4.

Constrained least-squares problems also arise frequently in practice. For instance if the parameters z are constrained to be nonnegative, then the resulting least-squares problem is a special case of quadratic programming

$$\min_{z} \quad \frac{1}{2} z^{\mathrm{T}} M z + z^{\mathrm{T}} w, \\
C z \ge d$$
(2)

and efficient algorithms for solving such non-negativity constrained least-squares problems were first proposed by Lawson and Hanson [59]. Alternatively, if the vector z is constrained in 2-norm, then

this results in a quadratic objective function with a single quadratic constraint. This is the situation, for example, in trust region methods for optimization (See Section 4).

Often a sequence of least-squares problems needs to be solved, each representing an update of the previous one due to the addition of new data or the downgrading of the importance of old data. Such problems arise, for example, in signal processing when we try to estimate the position of an unknown number of signal sources (e.g., finding the position of each aircraft within a given zone) given data from a set of receivers. Updating and downdating can be done quite stably if the full QR factorization is saved; in this case, Q is  $m \times m$ . If this is too expensive, then a variety of algorithms have been proposed that have decent numerical properties [8, Chapter 3].

The weighted least-squares problem

$$\min_{z} \|Mz - y\|_{W},$$

where  $||x||_{W}^{2} = x^{T}Wx$ , is also useful in practice. Here W is an estimate of the inverse covariance matrix for the zero-mean errors in measurement of y. The normal equations become

$$M^{\mathrm{T}}WMz = M^{\mathrm{T}}Wy,$$

and if we introduce the residuals s = W(y - Mz), then we can transform the normal equations into an augmented system

$$\begin{bmatrix} W^{-1} & M \\ M^{\mathrm{T}} & 0 \end{bmatrix} \begin{bmatrix} s \\ z \end{bmatrix} = \begin{bmatrix} y \\ 0 \end{bmatrix}.$$
 (3)

We will see this system again in Section 6.

If there are outliers in the data, then the least-squares criterion is not very useful unless the weights are adjusted so that the outliers do not affect the fit very much. This is the goal in *iteratively* reweighted least-squares, or robust regression [50], in which the fixed weight matrix W is replaced by some function of the size of a component of the residual

$$\min_{z}\sum_{i=1}^{m}w(y_i-(Mz)_i).$$

If  $w(u)=u^2$ , then we recover the least-squares problem. Functions that diminish the effects of outliers include Huber's choice [49]

$$w(u) = \begin{cases} u^2/2, & |u| \leq \beta, \\ \beta |u| - \beta^2/2, & |u| > \beta, \end{cases}$$

where  $\beta$  is a problem-dependent parameter. Minimizing Huber's function leads to a quadratic programming problem. Computational issues arising in iteratively reweighted least-squares problems are discussed, for example, in [73].

# 4. Unconstrained optimization

Given a point  $x_0$  and a quadratic model of a function

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2}(x - x_0)^{\mathrm{T}} f''(x_0)(x - x_0),$$

it is natural to approximate the minimizer of f by the minimizer of this model. If  $f''(x_0)$  is positive definite, this minimizer is given by

$$x = x_0 - f''(x_0)^{-1} f'(x_0).$$

This equation motivates Newton's method. Given  $x_0$ , we define a sequence of iterates

$$x_{k+1} = x_k + p_k,$$

where the direction  $p_k$  is the solution to the linear system

$$\bar{B}_k p_k = -f'(x_k) \tag{4}$$

and  $\bar{B}_k = f''(x_k)$ . If f is quadratic, then  $x_1$  is a stationary point of f, a global minimizer if f'' is positive definite. If f is not quadratic, the procedure is convergent at a quadratic rate of convergence to a local minimizer of f under conditions such as those of the Newton-Kantorovich theorem [74, Section 12.6].

Developments of Newton's method during the last 40 years have focussed on improving this method by making it more reliable and by adapting it for use on very large problems.

# 4.1. Making Newton's method more reliable: line searches and trust regions

Two methods have been used to make Newton's method (or its variants) globally convergent to a local minimizer: line searches and trust regions.

In the line search method, the Newton-like direction  $p_k$  is scaled so that

 $x_{k+1} = x_k + \alpha_k p_k,$ 

where  $\alpha_k$  is a parameter chosen to ensure that the objective function f decreases sufficiently in proportion to the size of the step. See [74, Section 14.4.3] for conditions on  $\alpha_k$  that guarantee global convergence (e.g., Wolfe conditions, Goldstein–Armijo conditions).

The trust region method constrains the length of the step so that we do not exit some region in which we "trust" the accuracy of the quadratic model. Thus we solve the problem

 $\min_p M(x_k+p),$ 

 $\|p\| \leq \tau$ ,

where M is the quadratic model and  $\tau$  is the radius of the trust region. If the constraint is active, then the solution to this problem is

$$(B_k + \lambda I)p = -f'(x_k)$$

for some nonnegative parameter  $\lambda$  chosen to make  $||p|| = \tau$ . This problem can be solved by eigendecomposition of  $\overline{B}_k$ , but this is generally too expensive. Often an iterative approach is used; we generate a sequence of approximations to p, stopping and backtracking when the norm of p exceeds  $\tau$ ; see, for example [38]. This does not give a step in the Newton direction unless the radius of the trust region exceeds the norm of the Newton direction  $p_k$  defined in (4).

#### 4.2. Making Newton's method more reliable for nonconvex functions

If the matrix  $\overline{B}_k$  used in the Newton equation is not positive definite, then Newton's method may fail to have a downhill search direction. To remedy this, algorithms based on line search usually diagnose indefiniteness as (4) is solved and cure it by adding a small correction matrix. These techniques are easily incorporated into a Cholesky factorization of the matrix [38].

Another approach to making Newton's method more reliable is to take very small steps – in fact, to follow the path

$$\frac{\mathrm{d}x}{\mathrm{d}t} = -f''(x)^{-1}f'(x),$$

starting with  $x(0) = x_0$ . This is the idea behind methods such as homotopy methods [54], which also introduce a parameterized function in order to locate multiple local minimizers. The linear algebra is heavily drawn from that used in ordinary differential equation solvers [93].

# 4.3. Adapting Newton's method for large problems

Computing, storing, and factoring the Hessian matrix may be impractical if the size is large. *Quasi-Newton* methods mimic Newton's method by generating less expensive approximations  $B_k$  to the matrix  $\overline{B}_k$ . These approximations are generated by updating the approximation for  $\overline{B}_{k-1}$ , and some have come to be interpreted as matrix approximation problems [28]. The most popular quasi-Newton variant is that proposed by Broyden, Fletcher, Goldfarb, and Shanno (BFGS), which is defined by the update formula

$$B_{k+1} = B_k - \frac{B_k s_k s_k^{\mathrm{T}} B_k}{s_k^{\mathrm{T}} B_k s_k} + \frac{y_k y_k^{\mathrm{T}}}{v_k^{\mathrm{T}} s_k},$$

where  $y_k$  is the change in gradient and  $s_k$  is the change in x. Rather than storing and updating  $B_{k+1}$ , we can also store and update its inverse or maintain the Cholesky factors of  $B_{k+1}$ ; see, for example, [37, Section 4.5.2.2].

An alternative is to use  $\bar{B}_k$  but avoid factoring it. This can be achieved by computing an approximate Newton search direction  $p_k$  as the solution to (4) using an iterative method (e.g., conjugate gradients or some other Krylov subspace method) that uses  $\bar{B}_k$  only for matrix-vector products. If a conjugate gradient algorithm is used, then there are very effective ways to handle indefiniteness and to determine the step for the trust region. The iteration can be terminated if the increment *d* to *p* is a direction of negative curvature (i.e.,  $d^T \bar{B}_k d < 0$ ) or if the algorithm steps out of the trust region [38]. Preconditioning can be used to improve convergence of the iterative methods [42, Section 10.3], but how this biases the generation of directions is an open question.

If  $\bar{B}_k$  is too expensive to compute or store, then the necessary products in the iterative method can be approximated by difference quotients

$$\bar{B}_k p = f''(x_k) p \approx \frac{f'(x_k + hp) - f'(x_k)}{h}$$

for a suitably small parameter h. This produces an algorithm that has come to be known as the truncated Newton method [26,72,69].

In very large problems, the updates to the quasi-Newton matrix may prove too numerous to store. In that case we might discard updates as they age, or skip some intermediate updates. These limited memory methods were proposed by Nocedal [70], and the properties of various discarding strategies are studied in [57].

# 4.4. Alternatives to Newton's method for large problems

There is a vast set of low-storage alternatives to Newton-like methods. They sacrifice the superlinear convergence rate that can be achieved under careful implementation of the Newton-like methods [71] in order to avoid storing a matrix approximation. Many of these methods are derived from methods for solving linear systems  $Ax^* = b$  involving a symmetric positive-definite matrix A.

The conjugate gradient method [46] takes a sequence of A-conjugate descent steps for the function  $(x - x^*)^T A(x - x^*)$  beginning with the steepest descent direction. Many authors proposed nonlinear extensions of this method, beginning with Fletcher-Reeves [34]. The algorithms are all equivalent to quasi-Newton algorithms on quadratic functions [33, Chapter 3], but the most robust algorithm for general functions is that of Polak and Ribière [78], restarting with the steepest descent direction in case trouble is diagnosed.

Another set of methods is related to fixed-point methods for solving linear systems. These linear methods are of the form

$$x_{k+1} = Ex_k + c,$$

where  $x^*$  is a fixed point of the iteration and the matrix *E* is chosen so that its spectral radius is small, yielding linear convergence of the sequence to  $x^*$ . Often these methods are derived by some variant of solving the *i*th equation for the *i*th variable, and then using estimates for the other variables to form a new value for the *i*th component of *x*. Examples of such methods are Jacobi, Gauss–Seidel, and SOR. See Varga [90] for a good discussion of such iterations in the linear case, and Ortega and Rheinboldt [74] for the general case.

# 5. Simplex method for linear programming

In the 1940s and 1950s, several events led to an explosion of interest in computational methods. The first was the computational need generated by the participants in World War II. Data fitting (least squares) and logistics support (optimization) created enormous demands for solution to ever-larger models. At the same time, electronic computing machines for the first time made it possible to solve problems larger than those that could be handled by a roomful of human calculators.

George Dantzig was among those who realized the need for automatic algorithms for solving optimization problems, and, working for the US war effort at Rand Corporation, he devised a *tableau* to organize the data in a linear programming problem

$\min_{x} c^{\mathrm{T}} x$	
Ax = b,	(5)
$x \ge 0$ ,	

where  $A \in \mathscr{R}^{m \times n}$ , with m < n. The tableau of numbers could be stored and modified systematically to produce an optimal solution to the problem, as well as information on the sensitivity of the solution to the data in A, b, and c [25]. Not only was this scheme well adapted to single or multiple human calculators, but it could also be implemented for large problems on electronic computers, enabling logistics planning that was unthinkable a few years earlier. The solution of linear programs, which earlier could in general be done only approximately by heuristic methods, could now be automated.

Dantzig's simplex algorithm was based on generating a path through the feasible set  $\mathscr{G} = \{x \ge 0: Ax = b\}$  through a set of vertices (i.e., points x for which at most m components are nonzero) that are adjacent (i.e., have all but one zero component in common). Along this path, the objective function  $c^{T}x$  usually decreases, but in any case does not increase. Once an *anti-cycling* safeguard is added that prevents any vertex from being visited too many times, the algorithm can be proven to converge, because there are only a finite number of vertices and it can be shown that one of them is an optimal solution.

For a given vertex x, we let  $\mathscr{I}$  denote the set of indices *i* for which  $x_i$  is nonzero. Then, if  $A_{\mathscr{I}}$  denotes the set of columns of A corresponding to indices in  $\mathscr{I}$ , we see that the nonzero components  $x_{\mathscr{I}}$  are defined by

$$A_{\mathscr{I}} x_{\mathscr{I}} = b. \tag{6}$$

In order to step from this vertex to an adjacent one, we replace one index in  $\mathscr{I}$  by an index not in  $\mathscr{I}$ , and the index is determined by solving a linear system involving the matrix  $A_{\mathscr{I}}^{T}$ . In order to compute the *x* corresponding to this step, we must modify our coefficient matrix by replacing one column with a new one. Dantzig proposed accumulating the matrix inverse and updating it using elementary row operations. Equivalently, his algorithm can be viewed as Gauss–Jordan elimination without pivoting [5]. This algorithm is numerically unstable, and simplex practitioners use periodic *reinversions* to recalculate the matrix inverse directly and eliminate the accumulated inaccuracies. This is still not unconditionally stable, but for many applications it works well. For dense matrices, the initial factorization cost is  $O(m^3)$  and the cost of each update is  $O(m^2)$ . Typically the simplex algorithm takes a reasonably small number of iterations – a small multiple of *m* [25, p. 160] – but in the worst case the algorithm can visit every vertex [56], so the bound on the cost is exponential in the problem size.

In the last half of the 20th century, the dimensions of linear programming problems have become much larger than first envisioned. At the same time, the matrices of interest have tended to become more *sparse*, with many zero elements. Consequently, even though the original matrix has large dimension, it usually can be stored in a small amount of memory. But the original simplex algorithm explicitly stores the matrix inverse, which is usually completely dense. Thus various modifications were made to the linear algebra of the algorithm to make it less of a storage hog. In the *revised simplex algorithm* with product form of the inverse, the inverse matrix is stored as a product of updates: a matrix  $A_{\text{f}}$  is represented as

$$A_{\mathscr{I}}^{-1} = R_{k-1} \dots R_1,$$

where each update matrix  $R_i$  differs from the identity by one column. This form is easily updated by accumulating additional matrices R, but when the storage for the updates becomes prohibitive, or when inaccuracies accumulate, reinversion is performed. The computational linear algebra community became interested in the simplex method in the late 1960s. Bartels and Golub [6,5] showed how the updating algorithm could be made stable through the use of partial pivoting, the interchange of rows of the matrix in order to bring the largest magnitude element in the current column to the main diagonal at every stage of the factorization. By computing in this way, it is possible to bound the error in the computed solution in two important ways: the computed solution solves a nearby problem, and the computed solution is close to the true solution [48, Chapter 9]. Neither of these properties is guaranteed to hold for earlier variants of the simplex algorithm. Still, the use of this stabilized algorithm met with resistance. Pivoting makes the implementation of updating much more costly, and for sparse matrices, it makes the data handling more difficult and the storage space generally higher.

The QR algorithm is an alternate matrix factorization that does not require pivoting for stability, but its fill-in often makes it prohibitively expensive for sparse matrices, so it was never widely used.

Much research in matrix reordering was spurred in part by the simplex algorithm. See [29] for more information on reordering.

Although iterative methods could be used to solve the linear systems in the simplex method, they have been proposed only for some special applications.

# 6. Interior point methods for linear programming

The proof by Khachian [55] that linear programming problems can be solved in polynomial time began a new era in the solution of optimization problems. Khachian's algorithm was not practical for computation, but suddenly a great deal of attention was focused on the *interior point method* (IPM), algorithms in which the path of the iterates stays in the relative interior of the feasible set rather than marching around the boundary from vertex to vertex. Karmarkar [52] was the first to propose a relatively practical interior point algorithm that had polynomial complexity, and that announcement spurred a flurry of work on new methods, as well as further work on older proposals such as the SUMT technique of Fiacco and McCormick [31].

The structure of IPMs is quite different from that of the simplex algorithm, but one similarity remains: the main computational work in the algorithm is the solution of a linear system of equations. Unlike the simplex algorithm, however, this linear system arises from a linear least-squares problem, and this extra structure can be quite useful. Further, although the sparsity structure of the matrix in the simplex algorithm changes from iteration to iteration, the structure of the matrix in the IPM is constant, and only the weights in a diagonal matrix are changing. This fact makes data management much easier.

Consider our linear programming problem (5). Gonzaga [43] and Wright [95] surveyed interior point methods, and many computational issues are addressed by Lustig et al. [64] and Andersen et al. [1]. The basic idea is to replace the linear program by a nonlinear problem formed by using Lagrange multipliers y to handle the linear equality constraints, and using barrier functions to avoid violating the nonnegativity constraints. One popular barrier function is the logarithmic barrier,  $\ln x_j$ , which goes to  $-\infty$  as  $x_j \rightarrow 0^+$ . The resulting Lagrange-barrier function is

$$L(x, y, \mu) = c^{\mathrm{T}}x - y^{\mathrm{T}}(Ax - b) - \mu \sum_{j=1}^{n} \ln x_{j}.$$

457

The solution to our linear programming problem (5) is the limit of the saddle-points of L as  $\mu \to 0$ . If we set the derivative of L equal to zero, we obtain necessary (first-order) conditions for a solution  $(x, y, \mu)$  to be optimal:

$$Ax = b,$$
  

$$c - A^{T}y - z = 0,$$
  

$$XZe = \mu e.$$

Here, *e* denotes a vector of all ones, and upper-case letters X and Z denote diagonal matrices created from the entries of the vectors x and z, respectively. In some sense this is a relaxation of the linear program, since these are optimality conditions for the linear program if we take z = 0 and  $\mu = 0$ . The idea is to solve a sequence of problems; initially,  $\mu$  is taken large in order to easily obtain a solution, and then  $\mu$  is reduced.

The introduction of the variables z makes the first two equations linear, and the Lagrange multipliers y can also be interpreted as the solution to the linear programming problem that is *dual* to (5). The most successful IPMs have been those that preserve primal feasibility by keeping Ax = b while at the same time maintaining dual feasibility by keeping  $c - A^{T}y \ge 0$ .

We now have a system of nonlinear equations to solve, and we can apply Newton's method. We compute the Newton direction by solving the KKT (Karush-Kuhn-Tucker) system

$$\begin{pmatrix} -X^{-1}Z & A^{\mathrm{T}} \\ A & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} r_d + Ze - \mu X^{-1}e \\ r_p \end{pmatrix},$$
(7)

or by solving the equations formed by eliminating  $\Delta x$  from this system. Defining  $r_p = b - Ax$ ,  $r_d = c - A^T y - z$ , and  $D^2 = Z^{-1}X$ , we obtain the normal equations

$$(AD^{2}A^{T})\Delta y = AD^{2}(r_{d} + Ze - \mu x^{-1}e) + r_{p}.$$
(8)

Once  $\Delta y$  is determined,  $\Delta x$  may be easily computed from

$$-(X^{-1}Z)\varDelta x + A^{\mathrm{T}}\varDelta y = r_d + Ze - \mu X^{-1}e.$$

Solving either Eq. (7) or (8), then, is the central computational problem in applying IPMs to linear programming problems. The remaining considerations are what sequence of  $\mu$  values to use, how accurately to solve intermediate problems, and when to terminate the algorithm or switch to direct solution once the optimal vertex is identified. For more information on these aspects, see, for example, [64,1]. Here we concentrate on the issues involved in solving (7) or (8).

The normal Eqs. (8) involve a symmetric positive semi-definite matrix (positive definite if A is full rank), and the Cholesky factorization is an efficient tool for solving such systems. If the matrix is sparse, though, then the Cholesky factor has the same sparsity structure as the triangular factor in the QR factorization of A, and this can be quite dense. We observe that

$$A^{\mathrm{T}}DA = \sum_{j=1}^{n} a_j d_{jj} a_j^{\mathrm{T}},$$

where  $a_j$  is a column of A. If a small number of columns are causing excessive fill-in, then these columns can be omitted from the factorization, and the Sherman–Morrison–Woodbury formula [42, Section 2.1.3] can be used to correct for their absence [3].

Simplex method	Interior point methods
Nonsymmetric	Symmetric positive definite (normal equations) or indefinite (augmented system)
Changing sparsity pattern	Fixed sparsity pattern
Usually well conditioned	Can become increasingly ill-conditioned as $\mu$ becomes smaller
Matrix changes by rank-2 update	Matrix changes completely, but only because $D$ is changing

Table 1

Comparison	of	matrix	problems	in	the	simplex	algorithm	and	in	IPMs
						-				

Table 1 compares the features of the matrix problems in the simplex algorithm and in IPMs.

If the matrix is ill-conditioned (which often happens at the end-stage, when  $\mu$  is small and some solution components go to zero) or rank-deficient (perhaps due to omission of columns in the factorization), it may be desirable to add a diagonal correction to  $A^TDA$  so that a factorization of a better conditioned matrix is computed. This technique of Stewart [84] has been used by Andersen [3] and others.

The matrix of the KKT system (7) is always symmetric indefinite. We also saw this matrix in optimality conditions (3) for weighted least-squares problems. Cholesky factorization is unstable for indefinite matrices, so other alternatives must be applied. The Bunch–Kaufman–Parlett factorization [14,13] into the product  $LSL^{T}$ , where L is lower triangular and S is block diagonal with  $1 \times 1$  or  $2 \times 2$  blocks, is a convenient tool for such problems.

If A is large and sparse, then the factorizations for (7) or (8) usually include sparsity considerations in the choice of pivot order.

Iterative methods for solving the linear systems in IPMs have received a great deal of attention but rather limited success. The key problem is the choice of preconditioner. Chin and Vannelli [19] solved the KKT system (7) using an incomplete factorization as a preconditioner, while Freund and Jarre [36] proposed SSOR preconditioners. Most of the preconditioning work has been on the normal equation formulation (8). Karmarkar and Ramakrishnan [53] used the factorization of the matrix for one value of  $\mu$  to precondition the problem when  $\mu$  is changed. Mehrotra [65] used an incomplete Cholesky factorization as a preconditioner, recomputing the factorization for each new  $\mu$ value. Carpenter and Shanno [16] used diagonal preconditioning, and Portugal, Resende, Veiga, and Júdice [79] also used spanning-tree preconditioners.

The best solution algorithm will surely be a hybrid approach that sometimes chooses direct solvers and sometimes iterative ones. Wang and O'Leary [92,91] proposed an adaptive algorithm for determining whether to use a direct or iterative solver, whether to reinitialize or update the preconditioner, and how many updates to apply, but further work remains.

The ill-conditioning of the matrices has stimulated a lot of work in trying to understand why the algorithms work as well as they do. Saunders [81] sets forth a set of reliable solution strategies, and a stability analysis is presented in [35].

# 7. Nonlinear programming

Optimization problems with nonlinearities in their objective function or their constraints can be more difficult to solve than linear programming. We survey selected nonlinear programming problems and strategies that make use of linear algebra.

Linear algebra plays a key role in the solution of quadratic programming problems (2) and of linear complementarity problems (LCP)

$$Ax - b = z,$$
$$x^{T}z = 0,$$
$$x \ge 0, z \ge 0.$$

Approaches include variations of the simplex algorithm, extensions of linear iterations such as Jacobi and Gauss–Seidel, descent methods such as conjugate gradient, and interior point algorithms. See [66] for a comprehensive discussion. Questions of existence and uniqueness of solutions to LCP spurred work in matrix theory on matrix cones [22].

In the past, two popular methods were used to handle constraints in nonlinear programming problems [62, Chapter 11]. In the first, certain constraints were held *active* for a portion of the iteration, and the iterates were not permitted to depart from them. Any descent step was augmented by a step back to the active constraints. In the second, descent directions were projected onto equality constraints before the step was taken; thus, steps were computed relative to a *reduced gradient* that corresponded to the gradient on the constraint surface. The computations were performed by projection matrices. Both of these strategies are currently in eclipse, due to the advent of sequential quadratic programming (SQP) and interior point methods.

In SQP, we solve a sequence of quadratic programming problems (2) arising from quadratic models of the original constrained problem, using IPM or simplex-based methods for the subproblems. Again we need modifications to maintain positive definiteness. Boggs and Tolle [12] give an excellent survey of these methods.

Interior point methods are also applied to nonlinear optimization problems directly [87]. The matrix in the augmented system (7) becomes somewhat more complicated than in the linear case; the lower right block can become nonzero, the upper left block may be full instead of diagonal, and in many formulations, the matrix is increasingly ill-conditioned [95,83]. The structure of this ill-conditioning is now somewhat understood, though, and, with care, the linear systems can be solved successfully [81].

Even these newer approaches, SQP and IPM, are not completely satisfactory, especially when the constraints are ill behaved [21].

A rather different approach to some classes of optimization problems is the use of neural networks [30]. These networks fit a surface to a function of many variables. There are various viewpoints for interpreting such methods, but one way is that the training of the network corresponds to optimizing parameters in a function that discriminates among different classes of points. The functional form is predetermined, and the optimization problem is generally non-convex, with many local solutions. To overcome this difficulty, Vapnik and colleagues proposed *support vector machines*, a more limited set of functional forms that are easier to analyze; see,

for example [89]. Many useful choices lead to convex optimization problems – in fact, to very large dense least squares or quadratic programming problems (2). Burges [15] provides a good introduction to the concepts and computational issues, while [88] is a more detailed study.

### 8. Matrix and eigenvalue optimization problems

And now we come full circle. We have seen how computational linear algebra has enabled efficiency advances in computational optimization. We have seen that the optimization community has raised interesting questions about matrix theory and about stability analysis of linear algorithms. Now we discuss a set of optimization problems that arise from linear algebra and have motivated the development of important optimization algorithms and advances in the understanding of duality theory for optimization problems.

These problems involve eigenvalue optimization [60]. An important subclass is the class of *semidefinite programs*. Superficially, they resemble linear programming problems (5), since they can be written as

$$\min_{\substack{X\\AX = B,}} C \bullet X,$$

$$X \ge 0,$$
(9)

but here *C* and *X* are symmetric  $n \times n$  matrices,  $C \bullet X = \text{trace}(CX)$ , and  $X \ge 0$  means that *X* is positive semidefinite. This problem is convex but nonlinear. The duality structure for semidefinite programming, the existence and construction of another problem that has the same optimality conditions as (9), is not as satisfying as that for linear programming. Despite the differences between the two classes of problems, linear programming gives much insight here, both for the theory and for the algorithms, and interior point methods that are direct generalizations of those for linear programming are the methods of choice.

Thus, semidefinite programming problems are eigenvalue optimization problems, and these problems have important linear algebra applications in control, in minimizing the condition number of a matrix by diagonal scaling, and in solving Lyapunov inequalities. Further information can be obtained from a review article of Lewis and Overton [60], a review article of Lobo, Vandenberghe, Boyd, and Lebret describing a subclass known as second-order cone programming [61], and a collection of papers [75].

# 9. Computational trends

Optimization algorithms can consume a great deal of computational resources, and they have always been run on state-of-the-art computer architectures. More and more, these algorithms are packaged and portable. There is reliable software for least-squares problems on parallel computers [82], and significant work has been done with neural networks [30, Section 4.1] and systolic arrays [58]. But there is limited experience with parallelization of constrained optimization codes. A notable effort is the parallel version of the CPLEX code by Lustig and Rothberg [63].

A second computational trend is the development of software that performs more of the drudgery for the user. Problem generators have been widely available for many years, but new tools are also being developed. Programs for automatic differentiation, for example, have contributed to the practical application of optimization techniques to a much larger set of problems. An automatic differentiation program uses the computational definition of a function in some programming language to generate a program for evaluating the derivative of the function. There are two basic strategies, both involving repeated applications of the chain rule. The forward mode consumes a great deal of intermediate storage, while the backward mode generally takes more time. Practical implementations generally use a combination of the two strategies, guided by linear algebra tools such as sparsity structure analysis and the construction of structurally orthogonal basis vectors [7].

# 10. Conclusions

Major developments in the basic linear algebra of optimization algorithms in the 20th century include:

- Invention of the simplex algorithm, based on Gauss-Jordan elimination and updating.
- Learning to implement the simplex algorithm in a stable way while preserving sparsity.
- Development and understanding of Newton alternatives: truncated Newton for use when derivatives are not available, quasi-Newton for use when second derivatives are not available, limited-memory and conjugate gradient methods for large problems.
- Development of least-squares algorithms for solving dense and sparse problems in a provably stable way.
- Development of algorithms for a wider range of constrained optimization problems, including those involving eigenvalue placement.
- Making automatic differentiation practical.
- Understanding the sensitivity of linear [48] and constrained problems [25, Section 12.4] [32] to perturbations in the data.

In addition, the development of efficient "off-the-shelf" packages of reliable software for dense linear algebra (LAPACK) [2] and sparse linear algebra (e.g., Harwell codes [45]) makes the development of efficient and reliable optimization software much easier, and most optimization packages do make use of this linear algebra basis.

Research areas that will remain active in the next century include:

- Hybrid algorithms for solving the linear systems from IPMs and other sources, involving automatic preconditioning.
- More effective algorithms for global optimization.
- More effective algorithms for nonlinear constraints.
- Sensitivity analysis.

Much progress in linear algebra in the 20th century has been motivated, at least in part, by optimization problems. This progress includes matrix up- and down-dating, sparse direct and iterative methods for linear systems, and solution of least squares problems. Conversely, progress in

optimization enables many previously intractable linear algebra problems to be solved, especially those related to eigenvalue placement. During the next century, this symbiosis will undoubtably continue. Progress in optimization will inevitably be linked with progress in linear algebra.

### Acknowledgements

I am grateful for the hospitality provided by Professor Walter Gander and the Departement Informatik, ETH Zürich, Switzerland, which enabled this work to be completed. I also appreciate helpful comments on a draft of this article from Tamara G. Kolola.

### References

- E.D. Andersen, J. Gondzio, C. Mészáros, X. Xu, Implementation of interior point methods for large scale linear programs, in: T. Terlaky (Ed.), Interior Point Methods of Mathematical Programming, Kluwer Academic Publishers, Boston, 1996, pp. 189–252.
- [2] E. Anderson, Z. Bai, C. Bischof, L.S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, D. Sorensen, LAPACK Users' Guide, SIAM, Philadelphia, 2000.
- [3] K.D. Andersen, A modified Schur-complement method for handling dense columns in interior point methods for linear programming, ACM Trans. Math. Software 22 (1996) 348–356.
- [4] M. Arioli, J.W. Demmel, I.S. Duff, Solving sparse linear systems with sparse backward error, SIAM J. Matrix Anal. Appl. 10 (1989) 165–190.
- [5] R.H. Bartels, A stablization of the simplex method, Numer. Math. 16 (1971) 414-434.
- [6] R.H. Bartels, G.H. Golub, The Simplex method of linear programming using LU decomposition, Comm. ACM 12 (1969) 266–268.
- [7] C. Bischof, A. Bouaricha, P. Khademi, J. Moré, Computing gradients in large-scale optimization using automatic differentiation, INFORMS J. Comput. 9 (1997) 185–194.
- [8] Å. Björck, Numerical Methods for Least Squares Problems, SIAM, Philadelphia, PA, 1996.
- [9] Å. Björck, T. Elfving, Algorithms for confluent Vandermonde systems, Numer. Math. 21 (1973) 130-137.
- [10] Å. Björck, C.C. Paige, Loss and recapture of orthogonality in the modified Gram–Schmidt algorithm, SIAM J. Matrix Anal. Appl. 13 (1992) 176–190.
- [11] Å. Björck, V. Pereyra, Solution of Vandermonde system of equations, Math. Comp. 24 (1970) 893-903.
- [12] P.T. Boggs, J.W. Tolle, Sequential quadratic programming, Acta Numerica 4 (1995) 1-51.
- [13] J.R. Bunch, L. Kaufman, B.N. Parlett, Decomposition of a symmetric matrix, Numer. Math. 27 (1976) 95-109.
- [14] J.R. Bunch, B.N. Parlett, Direct methods for solving symmetric indefinite systems of linear equations, SIAM J. Numer. Anal. 8 (1971) 639–655.
- [15] C.J.C. Burges, A tutorial on support vector machines for pattern recognition, Data Mining Knowledge Discovery 2 (1998) 121–167.
- [16] T.J. Carpenter, D.F. Shanno, An interior point method for quadratic programs based on conjugate projected gradients, Comput. Optim. Appl. 2 (1993) 5–28.
- [17] Y. Censor, Row-action methods for huge and sparse systems and their applications, SIAM Rev. 23 (1981) 444-466.
- [18] S. Chandrasekaran, I.C.F. Ipsen, On rank-revealing factorisations, SIAM J. Matrix Anal. Appl. 15 (1994) 592-622.
- [19] P. Chin, A. Vannelli, Iterative methods for the augmented equations in large-scale linear programming, Technical Report UWE& CE-94-01, Department of Electrical and Computer Engineering, University of Waterloo, October 1994.
- [20] C.K. Chui, An Introduction to Wavelets, Academic Press, New York, 1992.
- [21] A.R. Conn, N.I.M. Gould, P.L. Toint, Methods for nonlinear constraints in optimization calculations, in: I.S. Duff, G.A. Watson (Eds.), The State of the Art in Numerical Analysis, Clarendon Press, Oxford, England, 1997, pp. 363–390.

- [22] R.W. Cottle, J.-S. Pang, R.E. Stone, The Linear Complementarity Problem, Academic Press, New York, 1992.
- [23] M.G. Cox, The least squares solution of overdetermined linear equations having band or augmented band structure, IMA J. Numer. Anal. 1 (1981) 3–22.
- [24] G. Cybenko, Fast Toeplitz orthogonalization using inner products, SIAM J. Sci. Statist. Comput. 8 (1987) 734-740.
- [25] G.B. Dantzig, Linear Programming and Extensions, Princeton University Press, Princeton, NJ, 1963.
- [26] R.S. Dembo, S.C. Eisenstat, T. Steihaug, Inexact Newton methods, SIAM J. Numer. Anal. 19 (1982) 400-408.
- [27] C.J. Demeure, Fast QR factorization of Vandermonde matrices, Linear Algebra Appl. 122 (3/4) (1989) 165–194.
- [28] J.E. Dennis Jr., J.J. Morè, Quasi-Newton methods, motivation and theory, SIAM Rev. 19 (1977) 46-89.
- [29] I.S. Duff, A.M. Erisman, J.K. Reid, Direct Methods for Sparse Matrices, Clarendon Press, Oxford, 1986.
- [30] S.W. Ellacott, Aspects of the numerical analysis of neural networks, Acta Numer. 3 (1994) 145–202.
- [31] A.V. Fiacco, G.P. McCormick, Nonlinear Programming : Sequential Unconstrained Minimization Techniques, Wiley, New York, 1968 Reprint: Volume 4 of SIAM Classics in Applied Mathematics, SIAM Publications, Philadelphia, PA 19104–2688, USA, 1990).
- [32] A.V. Fiacco (Ed.), Mathematical Programming with Data Perturbations, Marcel Dekker, New York, 1998.
- [33] R. Fletcher, Practical Methods of Optimization, Wiley, New York, 1987.
- [34] R. Fletcher, C.M. Reeves, Function minimization by conjugate gradients, Comput. J. 7 (1964) 149-154.
- [35] A. Forsgren, P.E. Gill, J.R. Shinnerl, Stability of symmetric ill-conditioned systems arising in interior methods for constrained optimization, SIAM J. Matrix Anal. Appl. 17 (1996) 187–211.
- [36] R.W. Freund, F. Jarre, A QMR-based interior-point algorithm for solving linear programs, Technical Report, AT& T Bell Laboratories and Institut für Angewandte Mathematik und Statistik, 1995.
- [37] P.E. Gill, W. Murray, M.H. Wright, Practical Optimization, Academic Press, New York, 1981.
- [38] J. Nocedal, Large Scale Unconstrained Optimization, in: I.S. Duff, G.A. Watson (Eds.), The State of the Art in Numerical Analysis, Clarendon, Oxford, UK, 1997, pp. 311–338.
- [39] C.F. Gauss, Theory of the Combination of Observations Least Subject to Errors, Part One, Part Two, Supplement, SIAM, Philadelphia, PA, 1995 (Translated from the Latin by G. W. Stewart).
- [40] J.A. George, J.W.-H. Liu, Computer Solution of Large Sparse Positive Definite Systems, Prentice-Hall, Englewood Cliffs, NJ, 1981.
- [41] G. Golub, Numerical methods for solving least squares problems, Numer. Math. 7 (1965) 206-216.
- [42] G.H. Golub, C.F. Van Loan, Matrix Computations, Johns Hopkins University Press, Baltimore, MD, 1996.
- [43] C.C. Gonzaga, Path-following methods for linear programming, SIAM Rev. 34 (1992) 167-224.
- [44] P.C. Hansen, H. Gesmar, Fast orthogonal decomposition of rank deficient Toeplitz matrices, Numer. Algorithms 4 (1993) 151–166.
- [45] Harwell subroutine library, HSL Office, Culham, Oxon OX14 3ED, United Kingdom, http://www.cse. clrc.ac.uk/Activity/HSL.
- [46] M.R. Hestenes, E. Stiefel, Methods of conjugate gradients for solving linear systems, J. Res. Natl. Bur. Standards 49 (1952) 409–436.
- [47] N.J. Higham, Error analysis of the Björck-Pereyra algorithms for solving Vandermonde systems, Numer. Math. 50 (1987) 613–632.
- [48] N.J. Higham, Accuracy and Stability of Numerical Algorithms, SIAM, Philadelphia, PA, 1996.
- [49] P.J. Huber, Robust estimation of a location parameter, Ann. Math. Statist. 35 (1964) 73–101.
- [50] P.J. Huber, Robust Statistics, Wiley, New York, 1981.
- [51] T. Kailath, A.H. Sayed, Displacement structure: theory and applications, SIAM Rev. 37 (1995) 297-386.
- [52] N.K. Karmarkar, A new polynomial-time algorithm for linear programming, Combinatorica 4 (1984) 373–395.
- [53] N.K. Karmarkar, K.G. Ramakrishnan, Computational results of an interior point algorithm for large scale linear programming, Math. Programming 52 (1991) 555–586.
- [54] R.B. Kellogg, T.-Y. Li, J.A. Yorke, A constructive proof of the Brouwer fixed-point theorem and computational results, SIAM J. Numer. Anal. 13 (1976) 473–483.
- [55] L.G. Khachian, A polynomial algorithm in linear programming, Dokl. Akad. Nauk SSSR 244 (1979) 1093–1096.
- [56] V. Klee, G.J. Minty, How good is the simplex algorithm? In: O. Shisha, (Ed.), Inequalities III, Academic Press, New York, 1972, pp. 159–175.
- [57] T. Kolda, D.P. O'Leary, L. Nazareth, BFGS with update skipping and varying memory, SIAM J. Optim. 8 (1998) 1060–1083.

- [58] H.T. Kung, C.E. Leiserson, Introduction to VLSI systems, Addison-Wesley, Reading, MA, 1980.
- [59] C.L. Lawson, R.J. Hanson, Solving Least Squares Problems, Prentice-Hall, Englewood Cliffs, NJ, 1974; reprinted by SIAM, Philadelphia, PA, 1995.
- [60] A.S. Lewis, M.L. Overton, Eigenvalue optimization, Acta Numer. 5 (1996) 149-190.
- [61] M. Lobo, L. Vandenberghe, S. Boyd, H. Lebret, Applications of second-order cone programming, Linear Algebra Appl. 284 (1998) 193–228.
- [62] D.G. Luenberger, Linear and Nonlinear Programming, Addison-Wesley, Reading, MA, 1984.
- [63] I.J. Lustig, E. Rothberg, Gigaflops in linear programming, Technical Report, Silicon Graphics, 1995.
- [64] I.J. Lustig, R.E. Marsten, D.F. Shanno, Interior point methods for linear programming: computational state of the art, ORSA J. Comput. 6 (1) (1994) 1–14.
- [65] S. Mehrotra, Implementation of affine scaling methods: Approximate solutions of systems of linear equations using preconditioned conjugate gradient methods, ORSA J. Comput. 4 (2) (1992) 103–118.
- [66] K.G. Murty, Linear Complementarity, Linear and Nonlinear Programming, Heldermann Verlag, Berlin, Germany, 1988.
- [67] J.G. Nagy, Toeplitz least squares computations, Ph.D. Thesis, North Carolina State University, Raleigh, NC, 1991.
- [68] S.C. Narula, Optimization techniques in linear regression: a review, TIMS/Stud. Management Sci. 19 (1982) 11-29.
- [69] S.G. Nash, Preconditioning of truncated-Newton methods, SIAM J. Sci. Statist. Comput. 6 (1985) 599-616.
- [70] J. Nocedal, Updating quasi-Newton matrices with limited storage, Math. Comp. 35 (1980) 773-782.
- [71] J. Nocedal, Theory of algorithms for unconstrained optimization, Acta Numerica 1 (1992) 199-242.
- [72] D.P. O'Leary, A discrete Newton algorithm for minimizing a function of many variables, Math. Programming 23 (1982) 20–33.
- [73] D.P. O'Leary, Robust regression computation using iteratively reweighted least squares, SIAM J. Matrix Anal. Appl. 11 (1990) 466–480.
- [74] J.M. Ortega, W.C. Rheinboldt, Iterative Solution of Nonlinear Equations in Several Variables, Academic Press, New York, 1970.
- [75] M. Overton, H. Wolkowicz, Forward: special issue on semidefinite programming, Math. Programming 77 (1997) 105–110.
- [76] H. Park, L. Eldén, Stability analysis and fast algorithms for triangularization of Toeplitz matrices, Numer. Math. 76 (1997) 383–402.
- [77] G. Peters, J.H. Wilkinson, The least squares problem and pseudo-inverses, Comput. J. 13 (1970) 309-316.
- [78] E. Polak, G. Ribière, Note sur la convergence de methodes de directions conjugées, Rev. Francaise Informat Recherche Operationelle 3eAnnée 16 (1969) 35–43.
- [79] L.F. Portugal, M.G.C. Resende, G. Veiga, J.J. Júdice, A truncated primal-infeasible dual-feasible network interior point method, Networks (2000), to appear.
- [80] J.K. Reid, A Note on the least squares solution of a band system of linear equations by Householder reductions, Comput J. 10 (1967) 188–189.
- [81] M.A. Saunders, Cholesky-based methods for sparse least squares: the benefits of regularization, in: L. Adams, J.L. Nazareth (Eds.), Linear and Nonlinear Conjugate Gradient-Related Methods, SIAM, Philadelphia, PA, 1996, pp. 92–100.
- [82] L.S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, R.C. Whaley, ScaLAPACK Users' Guide, SIAM, Philadelphia, PA, 1997; http://www.netlib.org/scalapack/index.html.
- [83] D.F. Shanno, E.M. Simantiraki, Interior point methods for linear and nonlinear programming, in: I.S. Duff, G.A. Watson (Eds.), The State of the Art in Numerical Analysis, Clarendon Press, Oxford, England, 1997, pp. 339–362.
- [84] G.W. Stewart, Modifying pivot elements in Gaussian elimination, Math. Comp. 28 (126) (1974) 537-542.
- [85] G.W. Stewart, On the early history of the singular value decomposition, SIAM Rev. 35 (1993) 551-566.
- [86] D.R. Sweet, Fast Toeplitz orthogonalization, Numer. Math. 43 (1984) 1-21.
- [87] T. Terlaky (Ed.), Interior Point Methods of Mathematical Programming, Kluwer Academic Publishers, Boston, 1996.
- [88] V. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.
- [89] V. Vapnik, A. Ya. Lerner, Pattern recognition using generalized portraits, Automat. Remote Control 24 (1963) 709–719; transl. from Avtomatika i Telemekhanika 24 (6) (1963) 774–780.
- [90] R.S. Varga, Matrix Iterative Analysis, Prentice-Hall, Englewood Cliffs, NJ, 1962.

- [91] W. Wang, D.P. O'Leary, Adaptive use of iterative methods in predictor-corrector interior point methods for linear programming, Technical Report CS-TR-4011, Computer Science Department, University of Maryland, College Park, MD, April 1999, Numer. Algorithms, to appear.
- [92] W. Wang, D.P. O'Leary, Adaptive use of iterative methods in interior point methods for linear programming, Technical Report CS-TR-3560, Computer Science Department, University of Maryland, November 1995, http://www.cs.umd.edu/Dienst/UI/2.0/Describe/ncstrl.umcp/CS-TR-3560.
- [93] L.T. Watson, M. Sosonkina, R.C. Melville, A.P. Morgan, H.F. Walker, Algorithm 777: HOMPACK90: a suite of Fortran 90 codes for globally convergent homotopy algorithms, ACM Trans. Math. Software 23 (1997) 514–549.
- [94] P.-Å. Wedin, Perturbation theory for pseudo-inverses, BIT 13 (1973) 217–232.
- [95] M.H. Wright, Interior methods for constrained optimization, Acta Numerica 1 (1992) 341-407.
- [96] T.J. Ypma, Historical development of the Newton-Raphson method, SIAM Rev. 37 (1995) 531-551.



Journal of Computational and Applied Mathematics 123 (2000) 467-487

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.nl/locate/cam

# Some computational problems arising in adaptive optics imaging systems

Robert J. Plemmons<sup>a, \*, 1</sup>, Victor P. Pauca<sup>b, 2</sup>

<sup>a</sup>Departments of Mathematics and Computer Science, Box 7388, Wake Forest University, Winston-Salem, NC 27109,

USA

<sup>b</sup>Department of Computer Science, Box 90129, Duke University, Durham, NC 27708, USA

Received 31 May 1999

#### Abstract

Recently there has been growing interest and progress in using numerical linear algebra techniques in adaptive optics imaging control computations. Real-time adaptive optics is a means for enhancing the resolution of ground based, optical telescopes beyond the limits previously imposed by the turbulent atmosphere. An adaptive optics system automatically corrects for light distortions caused by the medium of transmission. The system measures the characteristics of the phase of the arriving wavefront and corrects for the degradations by means of one or more deformable mirrors controlled by special purpose computers.

No attempt is made in this paper to give a comprehensive survey of recent numerical linear applications in optical imaging. Rather, two fairly representative applications are discussed in some detail. The following research topics in the area of adaptive optics control systems, each involving the formulation and numerical solution of difficult problems in numerical linear algebra, are described: (1) Jacobi-like eigenvalue computations for multiple bandwidth deformable mirror control methods, and (2) covariance matrix computations for performance modeling of adaptive optics systems using fast Hankel transforms. © 2000 Elsevier Science B.V. All rights reserved.

*Keywords:* Imaging through turbulence; Adaptive optics control; Eigenvalues; Covariance matrix computations; Fast Hankel transforms; Structured matrices

<sup>\*</sup> Corresponding author.

*E-mail addresses:* plemmons@wfu.edu. Web: http://www.mthcsc.wfu.edu (R.J. Plemmons), pauca@cs.duke.edu. Web: http://www.cs.duke.edu (V.P. Pauca).

<sup>&</sup>lt;sup>1</sup>Research supported in part by the National Science Foundation.

<sup>&</sup>lt;sup>2</sup>Research supported in part by the Defense Advanced Research Projects Agency and by the National Science Foundation.



Fig. 1. Typical adaptive optics system.

# 1. Introduction

### 1.1. Atmospheric imaging and adaptive optics systems

Adaptive optics (AO) has emerged as the technique of choice to mitigate the blurring caused by atmospheric turbulence in large aperture imaging systems that allow extremely dim objects to be observed [11,15,18,20]. AO techniques compensate for degradations added along the path of the light from the object being imaged, prior to the formation of the image of the object.

AO systems are designed to measure errors in the phase of the arriving wavefront continually and correct them automatically. A typical AO imaging system consists of wave front sensors (WFS) and deformable mirrors (DM), in addition to an imaging system. See for example Fig. 1. Basically, the turbulence-induced wave front deformations, or aperture-averaged phase variance, are sensed by a WFS and compensated by a DM. The surface of the DM must be adjusted, using a vector of DM control commands, to be phase-conjugate to the incident wave front in a time commensurate with the rate of the turbulence change. Any AO systems attempting to compensate for atmospheric turbulence must address such *real-time* DM control problem. Typically, deformable mirrors operate in a closed loop as shown in Fig. 1, and can partially compensate for the degradation effects of atmospheric turbulence. However, to be effective, the DM control command vector must be computed and issued to the hardware at real-time speed. The DM control commands are usually calculated as some function of measurements of the incoming wavefront captured by the WFS, and parameters specific to the particular AO hardware being used, e.g. aperture diameter, WFS and DM geometry, level of sensor noise, etc. Exactly how to compute the DM control commands given the time and system constrains is an important topic of investigation that is addressed in this paper. Another related problem concerns performance modeling and design of adaptive optics systems. AO systems modeling and evaluation are essential to assessment of performance capacities and limitations and to the effective design of AO systems for large telescopes such as Gemini, Keck, Subaru, and VLT [20]. The objective of AO system design is to select hardware parameters and control approaches that will optimize system performance for the expected operating conditions subject to the resources available, and maximize the resulting payoff for the intended scientific applications.

The large range of possible observing scenarios and AO system parameters entail numerous cases to be considered, and fast computational approaches for performance evaluation and optimization are highly desirable. According to the conventional theory [17], turbulence is stochastic in nature. Hence, in most cases, modeling requires intensive computations involving the covariance matrices for the statistical relationship between phase gradient measurements and the DM control commands. To cover a parameter sampling space of a reasonable size, there may be hundreds or thousands of such covariance matrices to generate and compute with using today's computational practice for performance evaluation. Two-parameter Hankel transforms arise in the modeling of each covariance matrix entry for performance evaluation and design of AO imaging systems. The fast computation of such two-parameter Hankel transforms is addressed as well in this paper.

In modern facilities, AO-compensated image data is further processed by off-line *image restoration* (post-processing) tools that can scrub the captured optical images even cleaner. Image post-processing, which will not be discussed in detail in this paper, involves further removal or minimization of degradation (blur, clutter, noise, etc.) in an image using a priori knowledge about the degradation phenomena. Post-processing may restore the adaptive optics recorded image to a state even closer to perfection by filtering out any remaining noise and blur that can be distinguished from the image. The classic tool is regularized least squares; one of the newest techniques employed is based on the solution of a nonlinear partial differential equations. The power of these tools can be substantial. One of our simulations, for example, shows them improving the resolution of a telescope from being barely able to spot an object the size of a house trailer in earth's orbit to detecting a hand waving from the trailer's window!

Adaptive optics compensation plays an essential role in current state-of-the-art atmospheric telescope imaging technology. The ideal earth-based astronomical telescope is built on bedrock, high on a remote mountain. The solid foundation partially stabilizes the telescope against wind and other potential causes of vibration, while the altitude and isolation minimize atmospheric degradation. The Hubble space telescope carries this logic to its natural extreme, but even the Hubble's accuracy is limited by the effects of thermal stresses and other forces that shift the phase of the incoming light ever so slightly. Adaptive optics corrects the higher frequency errors caused by atmospheric irregularities and telescope vibration. With adaptive optics, instruments like the 3.5-m telescope at the Starfire Optical Range of the US Air Force Research Laboratory in New Mexico, can partially correct the image before it is recorded. Note that this real-time control requires extraordinarily high-speed computation – up to 10 billion floating point operations per second [15].

# 1.2. Linear algebra computations in optical imaging

We survey two topics concerning work in numerical linear algebra in investigations into algorithms and software for high-performance, high-resolution adaptive optics imaging applications: (i) eigenvalue computations in multiple bandwidth control in adaptive optics, and (ii) computation of fast transforms in performance modeling and adaptive optics system design.

The AO control problem includes the determination of one or more optimal reconstructor matrices for phase reconstruction as well as a set of commands to control the surface of the DM. Recently, AO systems have been proposed that use multiple control bandwidths [6,14] and corresponding techniques that may improve considerably AO compensation performance. These techniques concern an important eigenvalue problem related to a matrix trace maximization optimization problem.

In ground-based atmospheric imaging, a closed-loop adaptive optics systems must compensate for time-varying wavefront distortions on the basis of noisy sensor measurements. Time-varying distortions are most accurately corrected by a system with a high control bandwidth. However, the noise in wave front sensor measurements is best rejected by reducing the control bandwidth and temporally smoothing the wave front correction to be applied. The optimal control bandwidth minimizes the sum of these two effects and depends upon the wave front sensor noise and the temporal characteristics of the wave front errors to be corrected.

Most adaptive optics systems developed to date have addressed this tradeoff using a *common* control bandwidth for all components of the wave front distortion profile [3]. A system employing several different control bandwidths for separate wave front components has recently been tested [6]. Because wave front sensor noise statistics and the temporal dynamics of the wave front distortion to be corrected vary as a function of spatial frequency [20], it should, in principle, be possible to improve adaptive optics performance by using the more sophisticated multiple control bandwidth approach.

On the performance modeling and design problem, various modeling approaches that consider the stochastic nature of turbulence have been proposed. The linear systems model framework developed by Ellerbroek [4] provides first-order performance estimates which account for the full range of fundamental adaptive optics error sources and their interacting effects. It is being applied in the evaluation of existing and future AO facilities [5]. The modeling requires intensive computations involving the covariance matrices for the statistical relationship between the sensed phase gradient measurements and the DM actuator commands.

The computation is intensive in two aspects. First, the computation for each given parameter set includes generating all entries of certain covariance matrices, where each entry requires the evaluation of a multiple integral. Subsequent computations with the covariance matrices require numerous inversions and multiplications of large matrices [4] to obtain performance estimates such as the residual mean-square phase error and the associated optical transfer function of the telescope. Secondly, such computations are carried out many times over a large sample space of AO performance parameters. We describe briefly the parameters according to their physical meaning: (a) observing scenario parameters such as wavelength, aperture diameter, and zenith angle; (b) assumed atmosphere characteristics such as the wind profile, the refractive index structure constant of turbulence, and the turbulence outer scale; (c) architecture specifics of wave front sensing such as the WFS sub-aperture geometry, beacon geometry and conjugate range; (d) architecture specifics of deformable mirrors such as the actuator geometry and conjugate range, and (e) level of noise, degree of hardware imperfection or limitations.

In terms of the computation complexity, the size of the covariance matrices is proportional to the number of deformable mirror actuators and the number of wavefront sensor measurements. Typically, the matrices are of order from 100 to 5000 for high-order AO systems designed to compensate for

turbulence on very large telescopes. Present computational approaches for evaluating the integral for each matrix entry may sample the integrand at up to  $10^4$  points, where the integrand itself may be represented as an integral or a series [4].

# 1.3. Overview of the paper

In the following sections information is provided concerning work in numerical linear algebra in investigations into algorithms and software for high-performance, high-resolution imaging applications. Much of the work is particularly directed toward the development and implementation of innovative algorithms and techniques for optical imaging. The following research topics (each involving the formulation and numerical solution of difficult problems in numerical linear algebra) are described in this paper:

- Jacobi-like eigenvalue computations for multiple bandwidth control.
- Covariance matrix computations for performance modeling of adaptive optics systems using fast Hankel transforms.

Real-time adaptive-optics is a means for enhancing the resolution of ground based, optical telescopes beyond the limits previously imposed by the turbulent atmosphere. Our purpose in recent work on this topic has been to apply numerical linear algebra to investigate the areas of adaptive closed-loop deformable mirror control systems. Section 2 concerns an important eigenvalue problem related to a trace maximization optimization problem, approached using a Jacobi-like spectral algorithm [14]. In Section 3 fast integral transform methods [12] are applied to covariance matrix computations useful in the important area of adaptive optics systems performance analysis and design.

# 2. An eigenvalue problem in adaptive optics

We are concerned here with a non-smooth optimization problem arising in adaptive optics, which involves the real-time control of a deformable mirror designed to compensate for atmospheric turbulence and other dynamic image degradation factors [3]. One formulation of this problem yields a functional

$$f(U) = \sum_{i=1}^{n} \max_{j} \{ (U^{\mathrm{T}} M_{j} U)_{ii} \},$$
(1)

to be maximized over orthogonal matrices U for a fixed collection of  $n \times n$  symmetric matrices  $M_j$ . A study is made in [14] of eigenvalue computations used to solve this "trace maximization" problem. The reader interested in the derivation of the optimization problem or additional background on adaptive optics is referred to this paper.

First, the situation which can arise in practical applications where the matrices  $M_j$  are "nearly" pairwise commutative is considered. Besides giving useful bounds, results for this case lead to a theoretical closed-form solution for globally maximizing f. However, even here conventional optimization methods for maximizing f are not practical in a real-time environment. The general optimization problem is quite difficult but can be approached using a heuristic Jacobi-like algorithm.

Numerical tests indicate that the algorithm provides an effective means to optimize performance for some important adaptive-optics systems [14].

# 2.1. The nearly diagonalizable case

For computational convenience, we restrict our attention to the case of real  $n \times n$  matrices  $M_j$  and orthogonal U. The results given in this section extend in a natural way to the case where the  $M_j$  are Hermitian and U is unitary.

It has been observed in practical data [6], that the  $M_j$  in (1) are nearly diagonal matrices, and thus, of course, are almost simultaneously diagonalizable. The computations reported in [6] were performed with such data. Below is given a bound that sheds light on this situation and clarifies some observations reported in that paper. It also follows as a corollary to the theorem below that if the matrices are, in fact, simultaneously diagonalizable, then any orthogonal matrix Q which simultaneously diagonalizes the  $M_j$  globally maximizes f.

For notation purposes let

$$D_i = \operatorname{diag}(M_i)$$

for each j, and define the functional

$$h(U) = \sum_{i=1}^{n} \max_{j} \{ [U^{\mathrm{T}}(M_{j} - D_{j})U]_{ii} \}.$$
(2)

Let  $f_{\text{max}}$  denote the *global maximum* of f(U) over all orthogonal U. We say that the  $\{M_j\}$  form a "nearly optimal" set of matrices if

$$\sum_{i=1}^n \max_j \{(M_j)_{ii}\}$$

is close to  $f_{\text{max}}$ .

It was conjectured by the authors of [6] that f in (1) is maximized by a particular orthogonal matrix Q in the special case where the  $M_j$  are simultaneously diagonalized by Q. We now formally state and prove this result through the following technical lemma and resulting theorem.

**Lemma 1.** Suppose  $\{M_j\}, 1 \le j \le k$ , is a collection of n-by-n symmetric matrices. Then for any *n*-by-n orthogonal matrix U

$$f(I) \ge f(U) - h(U).$$

It follows that

$$\sum_{i=1}^{n} \max_{j} \{ (M_{j})_{ii} \} \ge f_{\max} - h(U),$$

for any U that globally maximizes f(U).

**Proof.** The proof is quite technical and can be found in [14].
Let U denote the global maximizer of f(U). Then the Lemma tells us that

$$f(I) + h(U) \ge f(U) = f_{\max}$$

It follows that

 $f(I) + \max_{U} h(U) \ge f_{\max}.$ 

That is, the worst case of our bound will be

 $\max_{U} h(U) \ge f_{\max} - f(I) \ge 0.$ 

Thus, if the  $M_j$  are "close" to diagonal, then since h(U) is "small", the  $M_j$  are "nearly" optimal. In the limiting case, we have the following theoretical result.

**Theorem 1.** Suppose  $\{M_j\}$ ,  $1 \le j \le k$ , is a collection of symmetric pairwise commuting matrices. Let Q be any orthogonal matrix which simultaneously diagonalizes the  $M_j$ . Then Q is a global maximizer of the functional f(U) given in (1).

**Proof.** Let  $B_j = U^T M_j U$  where U is an arbitrary orthogonal matrix. We can rewrite  $B_j$  using the orthogonal matrix  $V = Q^T U$ :

$$B_j = U^{\mathrm{T}} M_j U = (QV)^{\mathrm{T}} M_j (QV) = V^{\mathrm{T}} Q^{\mathrm{T}} M_j QV = V^{\mathrm{T}} D_j V,$$

where  $D_i = Q^{\mathrm{T}} M_i Q$  is the diagonalization of  $M_i$  using Q. Observe that

$$f(U) = \sum_{i=1}^{n} \max_{j} \{ (B_j)_{ii} \}$$

and that

$$f(Q) = \sum_{i=1}^{n} \max_{j} \{ (D_j)_{ii} \}.$$

Thus without loss of generality, we can assume the  $M_j$  are already diagonal matrices. We have to show that  $f(Q) \ge f(U)$ . But this follows from the Theorem since in this case h(U) = 0.  $\Box$ 

Besides giving interesting bounds, results in this section lead to a result providing a theoretical closed-form solution for globally maximizing f if the  $M_j$  are simultaneously diagonalizable. An algorithm that "nearly" simultaneously diagonalizes the matrices  $M_j$  will "approximately" maximize f(U).

Although Theorem 1 identifies an orthogonal matrix Q that globally maximizes the functional f(U) where the  $M_j$  are pairwise commutative, the process of computing the simultaneous diagonalizer Q can be quite nontrivial [1]. Thus, even here conventional optimization methods for maximizing f are not practical in this real-time environment. Instead, the authors of [14] pursue a fast heuristic approach to computing an acceptable U for our application. Further, the matrices  $M_j$  in their case are not necessarily simultaneously diagonalizable. The general optimization problem is quite difficult.

# 2.2. The general trace maximization problem

In this section we consider the general problem of maximizing f(U), where the  $M_j$  are generally not simultaneously diagonalizable.

In view of Theorem 1, one obvious approach for approximating a maximizer U for the general case would be to apply an extension of the simultaneous diagonalization of matrix pairs algorithm, e.g. the algorithm in [1], to the  $M_1, \ldots, M_k$ , until the transformed matrices are "close to" diagonal. One might measure the progress toward simultaneous diagonalization by a quantity such as

off
$$(M_1,...,M_k) = \sum_{i < j} [M_1]_{ij}^2 + \cdots + \sum_{i < j} [M_k]_{ij}^2$$

But the simultaneous diagonalization of more than two matrices is not an easy task to formulate algorithmically, and any such scheme would be very intensive computationally [1]. The authors of [14] have not implemented this approximate simultaneous diagonalization type of approach. Rather they choose a faster heuristic scheme described in the following sections, which appears to perform quite well on practical adaptive optics problems.

The general matrix optimization problem is considered next. Here, we describe a heuristic trace maximization approach based on a hill climbing scheme for maximizing f(U) relative to pairs of matrices. The case of k = 2 matrices is considered first, the solution of which leads to a heuristic algorithm for general k.

#### 2.2.1. A two-matrix algorithm

Let k = 2 and  $F = M_1$  and  $G = M_2$ . Suppose the orthogonal matrix U with columns  $[u_1 u_2 ... u_n]$  is the maximizer. Without loss of generality, the columns of U can be ordered in such a way that Eq. (1) can be written as

$$f(U) = \sum_{i=1}^{r} u_i^{\mathrm{T}} F u_i + \sum_{i=r+1}^{n} u_i^{\mathrm{T}} G u_i,$$
(3)

where r is the number of the diagonal elements of the product  $U^{T}FU$  that are larger than the corresponding diagonal elements of  $U^{T}GU$ . Let  $U = [U_1|U_2]$  with  $U_1 = [u_1 u_2 \dots u_r]$ . Since U is orthogonal, it follows that

$$UU^{\mathrm{T}} = U_1 U_1^{\mathrm{T}} + U_2 U_2^{\mathrm{T}} = I.$$

Using this and the trace properties

$$\operatorname{tr}(MN) = \operatorname{tr}(NM)$$
 and  $\operatorname{tr}(M+N) = \operatorname{tr}(M) + \operatorname{tr}(N)$ ,

we can rewrite Eq. (3) as follows:

$$f(U) = tr(U_1^T F U_1) + tr(U_2^T G U_2)$$
  
= tr(FU\_1 U\_1^T) + tr(GU\_2 U\_2^T)  
= tr(FU\_1 U\_1^T) + tr(G(I - U\_1 U\_1^T))  
= tr(FU\_1 U\_1^T) + tr(G - GU\_1 U\_1^T)

$$= \operatorname{tr}(FU_1U_1^{\mathrm{T}} - GU_1U_1^{\mathrm{T}}) + \operatorname{tr}(G)$$
$$= \operatorname{tr}((F - G)U_1U_1^{\mathrm{T}}) + \operatorname{tr}(G)$$
$$= \operatorname{tr}(U_1^{\mathrm{T}}(F - G)U_1) + \operatorname{tr}(G).$$

Therefore, the maximizer of f(U) is the maximizer of the term  $tr(U_1^T(F - G)U_1)$ , by taking as  $U_1$  the eigenvectors of F - G that correspond to positive eigenvalues, a direct consequence of Theorem 1 for the simultaneously diagonalizable matrices F - G and G - G. This computation involves the Schur decomposition of the symmetric matrix F - G. For a description of the Schur decomposition algorithm see [7].

#### 2.2.2. A Jacobi-like spectral algorithm

With the two-matrix maximization algorithm one can solve the general k-matrix problem with a Jacobi-like approach. If M is a  $n \times n$  matrix and s is a subset of the integers from 1 to n, we use the notation M(s,s) to denote the sub-matrix of M with rows and columns in s. M(:,s) denotes the sub-matrix with columns in s.

Let  $U = U_0$  (an initial guess).

While successive values of f(U) did not converge do

Let  $B_j$  be  $U^T M_j U$  for all  $j = 1 \dots k$ .

Choose a pair of matrices  $B_l, B_m$ .

Let *u* the set of indices *i* such that either  $B_l(i,i)$  or  $B_m(i,i)$ 

is the maximum element over all  $B_j(i,i)$  for j = 1...k.

Let  $U_1$  be the optimizer of the two-matrix subproblem  $B_l(u,u), B_m(u,u).$ Update U(:,u) with  $U(:,u)U_1$ .

Since  $M_j$  is symmetric, then the sub-matrix  $M_j(u, u)$  is symmetric, for any set of indices u. The two-matrix subproblem contains the current maximum elements for the indices in u, so for any increase to the value of the sum of the maximum diagonal elements, there will be at least as big an increase to the value of f(U). That means that the sequence of matrices  $\{U^{(j)}\}\$ , where  $U^{(j)}$  is the matrix U on the *j*th iteration of the main loop of the algorithm, defines a non-decreasing sequence of values  $\{f(U^{(j)})\}\$ . The algorithm terminates when the sequence  $\{f(U^{(j)})\}\$  converges. The work requirements in using this algorithm are  $O(kn^3)$  operations per sweep. Experience with some practical adaptive optics problems reported in [6] is that only a few sweeps are necessary for convergence.

The strategy for choosing the pair of matrices to improve upon leads to different algorithms. The simplest one is a sweep of all possible pairs of matrices (*Hill Climbing*). An alternate strategy is to choose the pair that gives the biggest increase in the value of f(U) (*Steepest-Ascent Hill Climbing*). Other techniques of heuristic search can be applied too.

The advantage of search algorithms is that they can improve the result of any other algorithm. One can consider any simpler heuristic algorithm as a preprocessing step for the search algorithms. The preprocessing algorithms can help us choose the initial orthogonal matrix  $U_0$ , instead of starting from the identity or a completely random orthogonal matrix. Such possible preprocessing algorithms

# could be:

- Find the diagonalizer of the matrix  $M_i$  with the largest trace to form  $U_0$ .
- For matrices that are "close to" simultaneously diagonalizable, one can use the almost diagonalizing U as  $U_0$ .
- Find the matrix  $M_j$  that has the largest sum of positive eigenvalues, and use the eigenvectors corresponding to this matrix in order to form an initial  $U_0$ .

# 2.3. Sample computations

Given next is a summary of the results of sample numerical experiments conducted to assess the utility of the Jacobi-like optimization algorithm for an actual adaptive optics application. We recall from the introduction that the motivation for studying the functional f(U) is to determine a set of wavefront control modes and associated temporal filters to optimize adaptive optics system performance, especially in cases where optical disturbances arise from several different error sources with significantly different temporal characteristics. The adaptive-optics scenario for these calculations is based upon parameters taken from the Gemini-North 8-m telescope in Hawaii. The optical disturbances to be corrected include atmospheric turbulence and telescope vibrations driven by ground-level winds. Telescope vibrations introduce optical distortions which may vary at a much higher frequency than atmospheric turbulence, but these distortions are restricted to two characteristic wavefront modes associated with misaligned mirrors. It is important to see if the Jacobi-like optimization method could identify these modes based solely upon the values of the matrices  $M_k$  and adjust their temporal filters accordingly. The performance of the modal control algorithm computed using the Jacobi-like method was scored by comparing it against the best possible control algorithm restricted to using a common temporal filter for all modes of the optical distortion. As defined previously in [6], these two control approaches will be referred to as the MB (multiple bandwidth) and SB (reduced range single bandwidth) control algorithms.

Adaptive-optics system performance is frequently quantified in terms of the residual mean-square phase error  $\sigma^2$  which remains when the adaptive-optics loop is closed.  $\sigma^2$  is related to the value of the functional f(U) in (1) by

$$\sigma^2 = \sigma_0^2 - f(U), \tag{4}$$

where  $\sigma_0^2$  is the "open-loop" mean-square phase error due to turbulence and telescope vibration with no adaptive optics at all. The value of f(U) can be quite close to  $\sigma_0^2$  when the adaptive optics system is correcting most of the optical disturbance, so that even a small (absolute) change in f(U)can make a big (relative) difference in the value of the residual error  $\sigma^2$ .

Sample values of  $\sigma^2$  for the MB and SB control algorithms are presented in Table 1 as a function of the level of noise in the wavefront sensor measurements. The performance of the two control algorithms is nearly identical when the wavefront sensor measurement noise is low, since in this case the best control strategy is to apply little or no temporal filtering and respond to the measured optical distortions as rapidly as possible. The performance of both control algorithms degrades with increasing WFS noise, but the MB control algorithm degrades less rapidly. For a high noise level the MB algorithm (as found using the Jacobi-like method) applies minimal filtering to the two wavefront modes associated with telescope vibration and a much greater amount of smoothing to

Table	1
-------	---

Residual errors  $\sigma^2$  for the SB (single bandwidth) and MB (multiple bandwidth) control algorithms as a function of WFS noise level for sample adaptive optics parameters

Noise level	SB	MB
0.00	0.009253	0.009252
0.05	0.011805	0.011267
0.10	0.017922	0.014464
0.20	0.037067	0.023678
0.40	0.075825	0.051618

the remaining modes resulting from atmospheric turbulence. For these conditions the SB algorithm faces a dilemma: A uniform temporal filter for all modes either fails to compensate the high-frequency telescope vibrations, or adds unnecessary noise to the correction of low-frequency errors associated with atmospheric turbulence. The resulting increase in  $\sigma^2$  can be larger than a factor of 1.5 at the higher noise levels.

An analysis, both theoretical and practical, has been given in this section for a difficult optimization problem arising in adaptive optics. This paper extends earlier work on an optimization method from [6] for the real-time control of deformable mirrors with the use of multiple bandwidths for adaptive-optics applications. See also [14] for more details.

Numerical tests, reported in Table 1, using a Jacobi-like algorithm to compare single bandwidth with our multiple bandwidth control methods indicate that the performance of a closed-loop adaptive-optics system which must compensate for the effects of both atmospheric turbulence and telescope vibration may be *greatly improved* by the selection of distinct and independently optimized multiple control bandwidths for separate modes of the wave-front-distortion profile. The Jacobi algorithm is well known to be highly parallelizable [7].

Although the simulation tests reported here are for problems of modest dimensions, future tests will involve adaptive optics systems utilizing deformable mirrors with many more degrees of freedom. The adaptive-optics model assumed for the calculations reported here is based upon the current Gemini design and includes a Shack–Hartmann wavefront sensor with  $8 \times 8$  subapertures and a continuous facesheet deformable mirror with  $9 \times 9$  actuators. Segmented mirror systems with very large numbers of degrees of freedom also are of interest. For example, the SELENE system [10] with segmented rather than continuous facesheet mirrors, is envisioned to have about 250,000 subapertures. Designing an adaptive optics system with a very large number of degrees of freedom taxes many areas, including the development of fast numerical linear algebra techniques.

# 3. Covariance matrix computations in adaptive optics

This section is concerned with the computation of covariance matrices useful in adaptive optics systems performance evaluations under various atmospheric conditions and hardware configurations. To illustrate the basic ideas, we sketch in more detail the main components of an AO system in Fig. 2. These include the *deformable mirror* (DM), the *wave front sensor* (WFS), and the *actuator command computer*. Light in a narrow spectral band passing through the atmosphere is modeled by a plane wave. When traveling through the atmosphere that does not have a uniform index of refraction,



Fig. 2. A simplified closed-loop AO system with main components.

light waves are aberrated and no longer planar. In a closed-loop AO system, this aberrated light is first reflected from the DM. Some of this light is focused to form an image, and some is diverted to the WFS that measures the wave front phase deformation. The actuator command computer takes measurements from the WFS and map them into real time control commands for the DM. How this translation is done depends on the criterion selected. In general, wave front sensing is a key aspect of many optical techniques and systems such as optical shop testing, interferometry and imaging through random media such as the earth's atmosphere.

#### 3.1. Modeling and adaptive optics system design

We present methods for the fast computation of two-parameter Hankel transforms arising in the modeling and performance evaluation of adaptive optics (AO) imaging systems. The objective of AO system design is to select hardware parameters and control approaches that will optimize system performance for the expected operating conditions subject to the resources available, and maximize the resulting payoff for the intended scientific applications.

The main hardware components of an AO imaging system consist of wave front sensors (WFS) and deformable mirrors (DM) system (see Fig. 2). We let s denote the discrete WFS *measurement vector* and c denote the DM *actuator command vector*. A linear systems model framework developed by Ellerbroek [3] provides first-order performance estimates which account for the full range of fundamental adaptive optics error sources and their interacting effects. The modeling requires

intensive computations involving the covariance matrices for the statistical relationship between the sensed phase gradient measurements s and the DM actuator commands c.

We describe the problem of computing the two-parameter Hankel transform that arises in Ellerbroek's model and many statistical models, see for example [17] and the references therein. In Section 3.3, we present two alternative approaches to representing the transform approximately by structured matrices. The matrix structures are to be exploited in the subsequent computations to reduce computation cost significantly. Recent studies on this topic include [3,4,12,13].

# 3.2. Hankel transform methods in adaptive optics covariance computations

The basic computational quantities for AO system modeling and evaluation are the covariance matrices which we express as

$$A = \operatorname{cov}(c, c), \quad B = \operatorname{cov}(c, s), \quad C = \operatorname{cov}(s, s).$$
(5)

These matrices describe the second-order statistics of the DM actuator command vector c, which optimally compensate for turbulence-induced phase distortions, and the temporally filtered WFS measurements vector s. The scalar components  $c_i$  of c can be represented as integrals of the form

$$c_i = \int_D \omega_i(r)\phi_i(r)\,\mathrm{d}r,\tag{6}$$

where r denotes coordinates in the aperture plane D,  $\phi_i(r)$  is, for example, an induced wave front propagating (see Fig. 2) from a point source, and  $\omega_i(r)$  is a weighting function. A similar integral representation exists for the  $s_i$  components.

Ellerbroek's derivation of (6), and hence (5), assumes isotropic turbulence with a Kolmogorov or von Kármán spectrum, Taylor's frozen-flow hypothesis, and no scintillation. In particular, if outer scale effects are considered and the direction of the atmospheric wind velocity is random and uniformly distributed,  $cov(c_i, c_j)$  may be represented mathematically as follows:

$$\operatorname{cov}(c_{i},c_{j}) = \gamma \left[ \int_{0}^{\infty} C_{n}^{2}(h) \, \mathrm{d}h \right]^{-1} \int \int w_{i}(r) w_{j}(r') \int_{0}^{H} C_{n}^{2}(h) \\ \times \int_{0}^{\infty} f(x) \left[ J_{0} \left( \frac{2\pi y}{L_{0}} x \right) J_{0} \left( \frac{2\pi y'}{L_{0}} x \right) - 1 \right] \, \mathrm{d}x \, \mathrm{d}h \, \mathrm{d}r \, \mathrm{d}r'.$$
(7)

Here, h is the range along the optical axis of the telescope, H is the range to the nearer of the two sources for the wavefronts  $\phi_i$  and  $\phi_j$ ,  $J_0$  is the zero-order Bessel function of the first kind,  $C_n^2$  is the index of refraction structure constant, which specifies the steady variation of the turbulence strength at altitude h, y is a scalar function of h, r and r', y' is a scalar function depending on the wind velocity at h,  $L_0$  is the outer scale of atmospheric turbulence at h, and  $\gamma$  is a normalization factor. The scalar function f(x) in (7) is defined as

$$f(x) = \frac{x}{(x^2 + 1)^{11/6}}$$
(8)

and is related to the von Kármán spectrum. In general, it depends on the characteristics of the modeling problem being studied. We use (8) throughout the paper for illustration purposes.

J

Evidently, efficient and accurate evaluation of the inner most integral in (7) is highly desirable. Following the notation in [2], we denote the *two-parameter Hankel transform* of a function f(x) by

$$h(\alpha,\beta) = \int_0^\infty f(x) J_0(\alpha x) J_0(\beta x) \,\mathrm{d}x,\tag{9}$$

where  $\alpha$  and  $\beta$  are nonnegative real numbers. For the inner most integral along x in (7), the parameters

$$\alpha = \frac{2\pi y}{L_0}, \qquad \beta = \frac{2\pi y'}{L_0},\tag{10}$$

depend on the function y and y'. Thus, use of numerical quadratures for the outer integrals require that the two-parameter Hankel transform f be evaluated at multiple points. As mentioned in the introduction section, the number of quadrature nodes may be at the order of  $10^4$ . Moreover, the change in y and/or y' may result in a different set of parameters.

Let M > 0 and N > 0. Given M points a = a(1 : M) and N points b = b(1 : N), we denote by

$$H(a,b), \quad a \in \mathbb{R}^M, \ b \in \mathbb{R}^N.$$

$$\tag{11}$$

the  $M \times N$  matrix with entries  $h(a_i, b_j)$ , i = 1 : M, j = 1 : N. For simplicity, we may assume M = N. By the modeling framework, if, instead, the direction of the atmospheric wind velocity is assumed to be known, then  $h(\alpha, \beta)$  becomes a function of a single parameter,  $h(\alpha)$  [4], i.e. a single-parameter Hankel transform.

Two computational approaches for evaluating H(a, b) have been presented in previous work. One approach simply employs numerical quadrature to the integration, requiring the evaluation of the Bessel function  $J_0$  at many quadrature points, which in turn may be computed either by the integral representation of  $J_0$ ,

$$J_0(\alpha) = \frac{1}{\pi} \int_0^{\pi} \cos(\alpha \cos \theta) \,\mathrm{d}\theta. \tag{12}$$

or by the series representation of  $J_0$ ,

$$J_0(\alpha) = \sum_{k=0}^{\infty} (-1)^k \left(\frac{\alpha^k}{2^k k!}\right)^2.$$

The alternative approach, which was proposed recently by Ellerbroek [3], applies the Mellin transform convolution theorem and the Cauchy residue theorem to convert  $h(\alpha, \beta)$  from integral form to a double power series in two parameters when  $\alpha < \beta$ . The resulting expression is

$$h(\alpha,\beta) = \gamma_1 \sum_{n=0}^{\infty} \sum_{m=0}^{\infty} \frac{u^n}{(n!)^2} \frac{(n+m)!}{(\frac{1}{6})_{n+m}} \frac{v^m}{(m!)^2} + \gamma_2 v^{5/6} \sum_{n=0}^{\infty} \sum_{m=0}^{\infty} \frac{u^n}{(n!)^2} \frac{(\frac{11}{6})_{n+m}}{(n+m)!} \frac{v^m}{[(\frac{11}{6})_m]^2} + \gamma_2 v^{5/6} \sum_{n=0}^{\infty} \sum_{m=1}^{\infty} \frac{u^n (\frac{11}{6})_n}{n!} \frac{1}{[(n+m)!]^2} (u/v)^m [(-\frac{5}{6})_m]^2,$$
(13)

where  $u = \alpha^2/4$ ,  $v = \beta^2/4$ ,  $\gamma_1 = \frac{3}{5}$ , and  $\gamma_2 = (\Gamma(-\frac{5}{6}))/(2\Gamma(\frac{11}{6}))$ . Here  $\Gamma$  is the Gamma function [2], and  $(z)_k = (\Gamma(z+k))/\Gamma(z)$  is a Gamma ratio. With either approach, the elements of the matrix

H are computed approximately. Nevertheless, the errors introduced by numerical quadratures or by truncating the power series may be made below a pre-specified threshold. Both approaches, however, are used at the element-wise level only, in the previous work [4]. In the next two sections, we introduce matrix factorization approaches that exploit the geometry of the parameter points and the smoothness of the integrand. These approaches lead to efficient algorithms for the subsequent computations involving H. Preliminary work by the authors on this topic appeared in [12].

#### 3.3. Compact-form representations using structured matrices

We present in this section two approaches for obtaining a compact-form representation of H(a, b) for efficient subsequent computations with H(a, b). The first approach is based on numerical quadratures, and the second one is based on the power expansion representation (13).

# 3.3.1. Numerical quadratures approach.

A fast single-parameter Hankel transform. We introduce first a fast approach for the single-parameter Hankel transform,

$$h(\alpha) = \frac{1}{\pi} \int_{-\alpha}^{\alpha} \frac{F(u)}{\sqrt{a^2 - u^2}} \,\mathrm{d}u,$$
(14)

where

$$F(u) \int_0^\infty f(x) \cos(ux) \,\mathrm{d}x \tag{15}$$

is the Fourier cosine transform of f and is even in u. Recall that this is related to the AO performance evaluation case when the wind velocity is known. The fast transform approach comes from the work of Kapur and Rokhlin in [9]. With properly designed numerical quadratures for the computation of the integrals in (15) and (14), the transform at N equally spaced points can be done with  $O(N \log N)$ arithmetic operations.

To compute (15), the integration range is first truncated to [0,X] for some real number X, independent of u, so that  $\int_X^{\infty} f(x) dx < \tau/2$ , where  $\tau$  is a given bound on approximation error. Next, we use a Newton-Cotes quadrature with N equally spaced nodes. Let  $h_x = X/(N-1)$ . Then,

$$F(u) = h_x \sum_{j=0}^{N-1} f(x_j) \cos(ux_j) \omega_j + \eta(u) = \tilde{F}(u) + \eta(u),$$
(16)

where  $x_j = jh_x$ ,  $\omega_j$  are quadrature weights at  $x_j$ , and  $\eta(u)$  is the approximation error in  $\tilde{F}(u)$ , which is bounded by  $\tau$ .

Let a(0: N-1) be an equally spaced set of evaluation points for the single-parameter transform (14). We assume that  $a(i) = ih_a$  with  $h_a = \pi/X$ . Let u = a. Then  $\tilde{F}$  in (16) at the N points of u can be written in matrix form

$$\tilde{F}(u) = Cv, \qquad C(i,j) = \cos(ij\pi/(N-1)), \qquad v(j) = h_x f(x_j)\omega_j.$$
 (17)

The matrix-vector multiplication Cv can be done with  $O(N \log N)$  operations with a fast discrete cosine transform algorithm.

We consider next the computation of (14), assuming F(u) is available. Notice first that  $h(0) = \int_0^\infty f(x) dx = F(0)$ . For i > 0, using the Trapezoidal rule with end-point correction  $(E_C)$  by Rokhlin [16], we have

$$h(a_i) = h_a \sum_{l=-(i-1)}^{i-1} \frac{F(u_l)}{\sqrt{a_i^2 - u_l^2}} + E_{\rm C}(a_i, k) + \varepsilon(a_i, k), \quad i = 1: N-1,$$
(18)

where the term

$$E_{\rm C}(a_i,k)h_a \sum_{l=-k,l\neq 0}^k v(i,l) \frac{F(a_i+lh_a)}{\sqrt{|a_i^2 - (a_i+lh_a)^2|}}$$
(19)

deals with the singularity of the integrand in (14) at the end points. With the end correction, the error term  $\varepsilon(a_i)$  is bounded as follows:

$$|\varepsilon(a_i,k)| \leq \frac{\gamma}{i^{4k-2}},$$

where  $\gamma$  is a constant, and k depends only on the required accuracy. The end-point correction weights v(i, l) may be pre-computed with O(kN) operations, and be reused. An easy modification to the quadrature can make the errors at all points  $a_i$  uniformly bounded by a pre-specified threshold, see [19]. Since k is independent of N and  $k \ll N$ , the cost for the end-point correction is O(N).

We now turn to the computation of the first term in (18) at all points  $a_i$ . It is easy to check that

$$H(a) = MDF(u) + E_{\rm C}(a,k) + \varepsilon(a,k), \tag{20}$$

where  $D = 2I - e_1 e_1^T$ , and

$$M(i,j) = \begin{cases} 1, & i = j = 0, \\ \frac{1}{\sqrt{i^2 - j^2}}, & j < i, i > 0, \\ 0, & \text{otherwise} \end{cases}$$
(21)

is a lower triangular matrix. The multiplication of M with a vector may take only  $O(N \log N)$  operations with an algorithm based on the same idea behind the fast multipole algorithm [8].

In summary, we have

$$H(a) = \tilde{H}(a) + MD\eta(u) + \varepsilon(a,k), \qquad \tilde{H}(a) = MDCv + E_{\rm C}(a,k).$$

The total cost for computing the approximate quantity  $\tilde{H}(a)$  is  $O(N \log N)$  operations. Since the matrices C and M need not be formed element-wise explicitly, the computation may take only O(N) memory units.

Two-parameter Hankel transform and compact form representation. The single-parameter Hankel transform approach can be efficiently extended to the two-parameter Hankel transform. Substituting the integral representation of  $J_0$  (12) in (9) and changing the variables  $u = \alpha \cos \theta$ ,  $v = \beta \cos \phi$ , we have

$$h(\alpha,\beta) = \frac{1}{\pi^2} \int_0^\infty \int_0^\pi \int_0^\pi f(x) \cos(\alpha x \cos\theta) \cos(\beta x \cos\phi) \,\mathrm{d}\theta \,\mathrm{d}\phi \,\mathrm{d}x$$
$$= \frac{1}{\pi^2} \int_{-\alpha}^\alpha \int_{-\beta}^\beta \int_0^\infty \frac{f(x) \cos(ux) \cos(vx)}{\sqrt{\alpha^2 - u^2} \sqrt{\beta^2 - v^2}} \,\mathrm{d}x \,\mathrm{d}u \,\mathrm{d}v.$$

Let

$$F(u,v)\int_0^\infty f(x)\cos(ux)\cos(vx)\,\mathrm{d}x,\tag{22}$$

then F(u, v) is well defined for the function f in (8). The two-parameter Hankel transform becomes

$$h(\alpha,\beta) = \frac{1}{\pi^2} \int_{-\alpha}^{\alpha} \int_{-\beta}^{\beta} \frac{F(u,v)}{\sqrt{\alpha^2 - u^2}\sqrt{\beta^2 - v^2}} \,\mathrm{d}u \,\mathrm{d}v.$$
(23)

The integrand of (23) is even in u and v, and is singular at  $\pm \alpha$  and  $\pm \beta$ .

Similar to the computation of the single-parameter transform, we compute F(u, v) in (22) at all evaluation points needed for the computation of H(a, b). For simplicity, we assume a = b = u = v. Thus,  $h_a = h_b = h$ . In matrix form

$$F(u,v) = \tilde{F}(u,v) + \eta(u,v), \qquad \tilde{F}(u,v) = CD_f C^{\mathrm{T}},$$
(24)

where  $D_f = h_x \operatorname{diag}(f(x_j)\omega_j)$ . That is, the approximate matrix  $\tilde{F}(u, v)$  is the two dimensional discrete cosine transform of  $D_f$ . We note that  $\tilde{F}(u, v)$  has a compact representation of O(N) storage complexity by using the structure of its matrix factors.

As in the single parameter case, the Trapezoidal rule with end correction can be applied to the integral in (23) in both directions u and v. The matrix form equivalent to (20) is as follows

$$H(a,b) = MDF(u,v)DM^{\mathrm{T}} + E_{\mathrm{C}}(a,b,k) + \varepsilon(a,b,k),$$

where  $E_C(a, b, k) = B_k \tilde{F}(u, v) B_k^T + E$ ,  $B_k$  is narrow banded, E is of low rank, and k > 0, is a constant independent of N. Since  $\tilde{F}(u, v)$  has the compact representation in (24), the approximate to H(a, b)

$$\tilde{H}(a,b) = MD\tilde{F}(u,v)DM^{\mathrm{T}} + E_{\mathrm{C}}(a,b,k),$$

has a compact-form representation of storage complexity O(N) as well.

#### 3.3.2. Power-series expansion approach

We now employ Ellerbrock's power-series expansion (13) to obtain another structured matrix representation. First, the three double power series can be truncated so that the truncated errors are below a pre-specified bound. For convenience, we use  $K(a_i, b_j)$ , or simply K, to denote the truncation point, i.e., the largest number of remaining terms in the truncated sums for the (i, j) element. Denote by vand( $\alpha$ ) the Vandermonde vector

vand(
$$\alpha$$
) = [ $a, \alpha, \alpha^2, \ldots, \alpha^{K-1}$ ]<sup>T</sup>,

at node  $\alpha$  of length K. Based on (13), we can write element  $H(a_i, b_j)$  as follows:

$$H(a_{i}, b_{j}) = c_{1} \operatorname{vand}^{T}(u_{i})H_{1} \operatorname{vand}(v_{j}),$$
  
+  $c_{2} \operatorname{vand}^{T}(u_{i})H_{2} \operatorname{vand}(v_{j})v_{j}^{5/6}$   
+  $c_{2} \operatorname{vand}^{T}(u_{i})H_{3} \operatorname{vand}(u_{j}/v_{j})v_{j}^{5/6} + \eta(a_{i}, b_{j}) \quad a_{i} \leq b_{j},$  (25)

where  $c_1$  and  $c_2$  are constants,  $H_1$ ,  $H_2$ , and  $H_3$  are  $K \times K$  Hankel matrices, up to diagonal scalings, the elements of which are independent of the values  $u_i = (a_i/2)^2$  and  $v_j = (b_j/2)^2$ .



Fig. 3. 3-D view (left) of the approximation of (, ), with  $h = \frac{3}{256}$ , and (right) a corresponding hierarchical matrix partition. The top left corner in the matrix partition corresponds to the position of the peak in the 3-D plot.

For simplicity, we assume that  $a_i, b_j \in [0, 1]$  and the elements of a and b are in increasing order. Consider first the special case that  $a_{N-1} \leq b_0$ . We get directly from (25)

$$H(a,b) = c_1 \operatorname{Vand}^1(u) H_1 \operatorname{Vand}(v),$$
  
+  $c_2 \operatorname{Vand}^T(u) H_2 \operatorname{Vand}(v) D(v)^{5/6}$   
+  $c_2 \operatorname{Vand}^T(u) H_3 \operatorname{Vand}(u./v) D(v)^{5/6} + \eta(a,b),$  (26)

where *u* is the vector with components  $u_i$  and *v* is the vector with components  $v_j$ , Vand((1:*N*)) denotes the Vandermonde matrix [vand( $x_1$ ),...,vand( $x_N$ )], *D*() is the diagonal matrix with the elements of placed on the diagonal. Let K(a,b) > 0 be the truncation length for all elements so that  $\eta(a,b)$  is element-wise below a pre-specified bound. It can be seen from the third sum of (25) that such a *K* decreases with the ratio  $r = a_{N-1}/b_0$ . Thus, for small *K*, H(a,b) can be approximated by a matrix of rank 3*K*. Notice that as long as the ratio *r* remains the same, the rank of the approximate matrix does not increase with *N*, and the matrix has a lower rank representation. In other words, the  $N \times N$  matrix has a representation of storage complexity O(N).

We are now in a position to show that for the case a = b, the matrix has a representation of storage complexity  $O(N \log N)$ . In this case, H(a,b) is symmetric. In the upper part of the matrix, all elements satisfy the expansion condition  $a_i \leq b_j$  as in (13). We partition the upper part into blocks so that each block can be approximated by a lower rank matrix, and the number of blocks is O(N). Let the ratio  $a_i/b_j$  in each block be bounded by 1. Let K be the truncation point. We illustrate such a partition for the case that the  $a_i$  are equally spaced in Fig. 3. Suppose the blocks on the diagonal are of  $m \times m$ ,  $m \leq K$ . The size of the blocks is doubled along the anti-diagonal. Each block can be approximated by a matrix of rank  $\leq 3K$ . There are  $\frac{3}{2}N/m$  blocks of order m,  $\frac{3}{2}((N/2m) - 1)$  blocks of order 4m, and so on. Thus, there are about 3N/m blocks in the upper part of the matrix. Let  $N = m2^p$  for some p > 0. Then, the representation storage complexity is proportional to  $\sum_{i=0}^{p-1} [N/(m2^i) - 1]m2^i = pN - (N - m) < N \log N$ .

### 3.4. Computations with the compact form matrix representations

The numerical computation of outer integrals in the generation of a covariance matrix element described by the integral (7) amounts to the multiplication of a matrix H(a, b) with a weight vector followed by a dot product. Each covariance matrix element is the sum of many such dot products. Using the structured matrix representation of H(a, b) via numerical quadratures, the multiplication of H(a, b) with a vector is obtained by a few matrix–vector multiplications and a vector summation. The matrices are either diagonal, banded, the discrete cosine transform matrix of the first type, or the multiplication can be done with O(N) or  $O(N \log N)$  operations. Thus, the cost for the multiplication of H(a, b) with a vector is  $O(N \log N)$ .

By the structured matrix representation via the power series, the matrix H(a, b) can be viewed as a sum of block banded matrices along  $O(\log N)$  diagonals. The multiplication of each block banded matrix with a vector involves O(N) operations. The total matrix-vector multiplication cost is thus  $O(N \log N)$ . We note that, although the two compact representations are different in storage requirement by a factor of  $O(\log N)$ , both lead to the matrix-vector multiplications with  $O(N \log N)$ arithmetic operations.

# 3.5. Additional comments

There is still room for reduction in computation cost. A study on the accuracy and efficiency for computing single-parameter Hankel transform on non-uniform sampling points as opposed to equally spaced points is reported in [19], based on the use of numerical quadratures. A similar study of two-parameter Hankel transforms is yet to be done.

The techniques to obtain power-series expansions can be extended to a larger class of computational problems with Bessel functions of the first kind as the inner most integrand. Numerical experiments show, however, that the approach via numerical quadrature is less sensitive to rounding errors than the power series approach. Our numerical experiments show that with the same partition of the H(a, b), the numerical rank of each block in double precision is much lower than that suggested by the truncation point in the power series expansion.

# 4. Summary of the paper

To summarize this paper, we have considered certain representative numerical linear algebra problems arising in deformable mirror control computations in adaptive-optics compensated imaging. The randomness and time evolution of the atmospheric inhomogeneities make imaging through turbulence a difficult and challenging problem. Adaptive optics techniques afford a mechanical means of sensing and correcting for turbulence effects as they occur. High-resolution images are essential in many important applications in defense, science, engineering, law enforcement, and medicine. The need to extract meaningful information from degraded images is especially vital for such applications as aero-optics imaging, surveillance photography, and other modern imaging systems. The goals of this paper were to describe two recent innovative adaptive-optics deformable mirror control algorithms, with concentration on performance analysis, software, for on-line adaptive-optics systems for imaging through turbulence. We feel that it is important to exploit the general mathematical structure of computational matrix problems found in several optical imaging and other signal processing problems. This may lead to a family of methods and routines that are adaptable to several scenarios in obtaining high-resolution images using modern adaptive optics systems.

#### Acknowledgements

The authors would like to thank Dr. Brent Ellerbroek at the Air Force Research Laboratory Starfire Optical Range, NM for providing motivation, encouragement, and very helpful advice for this study. Bob Plemmons, would like to thank the Computer Science Department at Duke University for hosting his sabbatical in Fall 1998, at which time he participated in a series of seminars on linear algebra in adaptive optics with Moody Chu from NC State University, Nikos Pitsianis from BOPS, Inc., and with Paul Pauca and Xiaobai Sun from Duke University.

# References

- A. Bunse-Gerstner, R. Byers, V. Mehrmann, Numerical methods for simultaneous diagonalization, SIAM. J. Matrix Anal. Appl. 14 (1993) 927–949.
- [2] L. Debnath, Integral Transforms and their Applications, CRC Press, New York, 1995.
- [3] B.L. Ellerbroek, First-order performance evaluation of adaptive optics systems for atmospheric turbulence compensation in extended field-of-view astronomical telescopes, J. Opt. Soc. Am. A 11 (1994) 783–805.
- [4] B.L. Ellerbroek, Including outer scale effects in zonal adaptive optics calculations, Appl. Opt. 36 (36) (1997) 9456–9467.
- [5] B.L. Ellerbroek, D.W. Tyler, Adaptive optics sky coverage calculations for the gemini-north telescope, Proc. Astron. Soc. Pac. 110 (1998) 165–185.
- [6] B.L. Ellerbroek, C. Van Loan, N. Pitsianis, R.J. Plemmons, Optimizing closed-loop adaptive optics performance using multiple control bandwidths, J. Opt. Soc. Am. A 11 (1994) 2871–2886.
- [7] G.H. Golub, C. Van Loan, Matrix Computations, 3rd Edition, The Johns Hopkins University Press, Baltimore, MD, 1996.
- [8] L. Greengard, V. Rokhlin, A fast algorithm for particle simulations, J. Comput. Phys. 135 (1997) 280-292.
- [9] S. Kapur, V. Rokhlin, An algorithm for the fast Hankel transform, Technical Report 1045, Computer Science Department, Yale University, 1995.
- [10] M. Milman, A. Fijany, Wave-front control algorithms and analysis for a dense adaptive optics system with segmented deformable mirror, J. Opt. Soc. Am. A 13 (1996) 365–378.
- [11] J. Nelson, Reinventing the telescope, Popular Science 85 (1995) 57-59.
- [12] V.P. Pauca, B.L. Ellerbroek, N. Pitsianis, R.J. Plemmons, X. Sun, Performance modeling of adaptive-optics imaging systems using fast Hankel transforms, Adv. Signal Process. Algorithms, Architectures, and Implementations VIII 3461 (1998) 339–347.
- [13] V.P. Pauca, R.J. Plemmons, B.L. Ellerbroek, X. Sun, Efficient two-parameter Hankel transforms in adaptive optics system evaluations, Lin. Algebra Appl., to appear.
- [14] N.P. Pitsianis, B.L. Ellerbroek, C. Van Loan, R.J. Plemmons, Jacobi-like method for a control algorithm in adaptive-optics imaging, Adv. Signal Process. Algorithms, Architectures, and Implementations VIII 3461 (1998) 296–307.
- [15] M.C. Roggemann, B. Welsh, Imaging Through Turbulence, CRC Press, Boca Raton, FL, 1996.
- [16] V. Rokhlin, End-point corrected trapezoidal quadrature rules for singular functions, Comput. Math. Appl. 20 (1990) 51-62.
- [17] R.J. Sasiela, Electromagnetic Wave Propagation in Turbulence, Springer, Berlin, 1994.

- [18] R.A. Smith, Seeing the universe through new eyes, National Geographic Mag. 185 (1) (1994) 2-41.
- [19] X. Sun, V.P. Pauca, A note on fast Hankel transforms, Technical report, Duke University, 1999.
- [20] R.K. Tyson, Principles of Adaptive Optics, 2nd Edition, Academic Press, San Diego, 1998.



Journal of Computational and Applied Mathematics 123 (2000) 489-514

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.nl/locate/cam

# Numerical linear algebra algorithms and software

Jack J. Dongarra<sup>a,b,\*</sup>, Victor Eijkhout<sup>a</sup>

<sup>a</sup>Department of Computer Science, University of Tennessee, 107 Ayres Hall, Knoxville, TN 37996-1301, USA <sup>b</sup>Mathematical Sciences Section, Oak Ridge National Laboratory, P.O. Box 2008, Bldg. 6012, Oak Ridge,

TN 37831-6367, USA

Received 12 July 1999; received in revised form 16 August 1999

#### Abstract

The increasing availability of advanced-architecture computers has a significant effect on all spheres of scientific computation, including algorithm research and software development in numerical linear algebra. Linear algebra – in particular, the solution of linear systems of equations – lies at the heart of most calculations in scientific computing. This paper discusses some of the recent developments in linear algebra designed to exploit these advanced-architecture computers. We discuss two broad classes of algorithms: those for dense, and those for sparse matrices. © 2000 Elsevier Science B.V. All rights reserved.

# 1. Introduction

The increasing availability of advanced-architecture computers has a significant effect on all spheres of scientific computation, including algorithm research and software development in numerical linear algebra. Linear algebra – in particular, the solution of linear systems of equations – lies at the heart of most calculations in scientific computing. This article discusses some of the recent developments in linear algebra designed to exploit these advanced-architecture computers. We discuss two broad classes of algorithms: those for dense, and those for sparse matrices. A matrix is called sparse if it has a substantial number of zero elements, making specialized storage and algorithms necessary.

Much of the work in developing linear algebra software for advanced-architecture computers is motivated by the need to solve large problems on the fastest computers available. In this article, we focus on four basic issues: (1) the motivation for the work; (2) the development of standards

\* Corresponding author.

E-mail address: dongarra@cs.utk.edu (J.J. Dongarra).

for use in linear algebra and the building blocks for libraries; (3) aspects of algorithm design and parallel implementation; and (4) future directions for research.

As representative examples of dense matrix routines, we will consider the Cholesky and LU factorizations, and these will be used to highlight the most important factors that must be considered in designing linear algebra software for advanced-architecture computers. We use these factorization routines for illustrative purposes not only because they are relatively simple, but also because of their importance in several scientific and engineering applications that make use of boundary element methods. These applications include electromagnetic scattering and computational fluid dynamics problems, as discussed in more detail in Section 2.1.2.

For the past 15 years or so, there has been a great deal of activity in the area of algorithms and software for solving linear algebra problems. The goal of achieving high performance on codes that are portable across platforms has largely been realized by the identification of linear algebra kernels, the basic linear algebra subprograms (BLAS). We will discuss the EISPACK, LINPACK, LAPACK, and ScaLAPACK libraries which are expressed in successive levels of the BLAS.

The key insight of our approach to designing linear algebra algorithms for advanced architecture computers is that the frequency with which data are moved between different levels of the memory hierarchy must be minimized in order to attain high performance. Thus, our main algorithmic approach for exploiting both vectorization and parallelism in our implementations is the use of block-partitioned algorithms, particularly in conjunction with highly tuned kernels for performing matrix–vector and matrix–matrix operations (the Level 2 and 3 BLAS).

# 2. Dense linear algebra algorithms

# 2.1. Overview of dense algorithms

Common operations involving dense matrices are the solution of linear systems

$$Ax = b$$
,

the least-squares solution of over- or underdetermined systems

$$\min_{x} ||Ax - b||$$

and the computation of eigenvalues and -vectors

 $Ax = \lambda x.$ 

Although these problems are formulated as matrix-vector equations, their solution involves a definite matrix-matrix component. For instance, in order to solve a linear system, the coefficient matrix is first factored as

$$A = LU$$

(or  $A = U^{T}U$  in the case of symmetry) where L and U are lower and upper triangular matrices, respectively. It is a common feature of these matrix-matrix operations that they take, on a matrix of size  $n \times n$ , a number of operations proportional to  $n^{3}$ , a factor n more than the number of data elements involved.

Thus, we are led to identify three levels of linear algebra operations:

- Level 1: vector-vector operations such as the update  $y \leftarrow y + \alpha x$  and the inner product  $d = x^T y$ . These operations involve (for vectors of length *n*) O(*n*) data and O(*n*) operations.
- Level 2: matrix-vector operations such as the matrix-vector product y = Ax. These involve  $O(n^2)$  operations on  $O(n^2)$  data.
- Level 3: matrix-matrix operations such as the matrix-matrix product C = AB. These involve  $O(n^3)$  operations on  $O(n^2)$  data.

These three levels of operations have been realized in a software standards known as the basic linear algebra subprograms (BLAS) [17,18,46]. Although BLAS routines are freely available on the net, many computer vendors supply a tuned, often assembly coded, BLAS library optimized for their particular architecture, see also Section 4.3.

The relation between the number of operations and the amount of data is crucial for the performance of the algorithm. We discuss this in detail in Section 3.1.

# 2.1.1. Loop rearranging

The operations of BLAS levels 2 and 3 can be implemented using doubly and triply nested loops, respectively. With simply modifications, this means that for level 2 each algorithms has two, and for level 3 six different implementations [20]. For instance, solving a lower triangular system Lx = y is mostly written

for 
$$i = 1 \dots n$$
  
 $t = 0$   
for  $j = 1 \dots i - 1$   
 $t \leftarrow t + \ell_{ij}x_j$   
 $x = \ell_{ii}^{-1}(y_i - t)$ 

but can also be written as

for 
$$j = 1 \dots n$$
  
 $x_j = \ell_{jj}^{-1} y_j$   
for  $i = j + 1 \dots n$   
 $y_i \leftarrow y_i - \ell_{ij} x_j$ 

(The latter implementation overwrites the right-hand side vector y, but this can be eliminated.)

While the two implementations are equivalent in terms of number of operations, there may be substantial differences in performance due to architectural considerations. We note, for instance, that the inner loop in the first implementation uses a row of L, whereas the inner loop in the second traverses a column. Since matrices are usually stored with either rows or columns in contiguous locations, column storage the historical default inherited from the FORTRAN programming language, the performance of the two can be radically different. We discuss this point further in Section 3.1.

# 2.1.2. Uses of LU factorization in science and engineering

A major source of large dense linear systems is problems involving the solution of boundary integral equations [26]. These are integral equations defined on the boundary of a region of interest. All examples of practical interest compute some intermediate quantity on a two-dimensional boundary

and then use this information to compute the final desired quantity in three-dimensional space. The price one pays for replacing three dimensions with two is that what started as a sparse problem in  $O(n^3)$  variables is replaced by a dense problem in  $O(n^2)$ .

Dense systems of linear equations are found in numerous applications, including:

- airplane wing design;
- radar cross-section studies;
- flow around ships and other off-shore constructions;
- diffusion of solid bodies in a liquid;
- noise reduction; and
- diffusion of light through small particles.

The electromagnetics community is a major user of dense linear systems solvers. Of particular interest to this community is the solution of the so-called radar cross-section problem. In this problem, a signal of fixed frequency bounces off an object; the goal is to determine the intensity of the reflected signal in all possible directions. The underlying differential equation may vary, depending on the specific problem. In the design of stealth aircraft, the principal equation is the Helmholtz equation. To solve this equation, researchers use the *method of moments* [37,62]. In the case of fluid flow, the problem often involves solving the Laplace or Poisson equation. Here, the boundary integral solution is known as the *panel method* [38,39], so named from the quadrilaterals that discretize and approximate a structure such as an airplane. Generally, these methods are called *boundary element methods*.

Use of these methods produces a dense linear system of size  $O(N) \times O(N)$ , where N is the number of boundary points (or panels) being used. It is not unusual to see size  $3N \times 3N$ , because of three physical quantities of interest at every boundary element.

A typical approach to solving such systems is to use LU factorization. Each entry of the matrix is computed as an interaction of two boundary elements. Often, many integrals must be computed. In many instances, the time required to compute the matrix is considerably larger than the time for solution.

The builders of stealth technology who are interested in radar cross-sections are using direct Gaussian elimination methods for solving dense linear systems. These systems are always symmetric and complex, but not Hermitian.

For further information on various methods for solving large dense linear algebra problems that arise in computational fluid dynamics, see the report by Alan Edelman [26].

# 2.2. Block algorithms and their derivation

It is comparatively straightforward to recode many of the dense linear algebra algorithms so that they use level 2 BLAS. Indeed, in the simplest cases the same floating-point operations are done, possibly even in the same order: it is just a matter of reorganizing the software. To illustrate this point, we consider the Cholesky factorization algorithm, which factors a symmetric positive-definite matrix as  $A = U^T U$ . We consider Cholesky factorization because the algorithm is simple, and no pivoting is required on a positive-definite matrix.

Suppose that after j - 1 steps the block  $A_{00}$  in the upper left-hand corner of A has been factored as  $A_{00} = U_{00}^{T}U_{00}$ . The next row and column of the factorization can then be computed by writing

$$= U^{\mathrm{T}}U \text{ as}$$

$$\begin{pmatrix} A_{00} \ b_{j} \ A_{02} \\ . \ a_{jj} \ c_{j}^{\mathrm{T}} \\ . \ . \ A_{22} \end{pmatrix} = \begin{pmatrix} U_{00}^{\mathrm{T}} \ 0 \ 0 \\ v_{j}^{\mathrm{T}} \ u_{jj} \ 0 \\ U_{02}^{\mathrm{T}} \ w_{j} \ U_{22}^{\mathrm{T}} \end{pmatrix} \begin{pmatrix} U_{00} \ v_{j} \ U_{02} \\ 0 \ u_{jj} \ w_{j}^{\mathrm{T}} \\ 0 \ 0 \ U_{22} \end{pmatrix},$$

where  $b_j, c_j, v_j$ , and  $w_j$  are column vectors of length j - 1, and  $a_{jj}$  and  $u_{jj}$  are scalars. Equating coefficients on the *j*th column, we obtain

$$b_j = U_{00}^{\mathrm{T}} v_j, \qquad a_{jj} = v_j^{\mathrm{T}} v_j + u_{jj}^2.$$

A

Since  $U_{00}$  has already been computed, we can compute  $v_j$  and  $u_{jj}$  from the equations

$$U_{00}^{\mathrm{T}} v_j = b_j, \qquad u_{jj}^2 = a_{jj} - v_j^{\mathrm{T}} v_j.$$

The computation of  $v_j$  is a triangular system solution, a BLAS level 2 operation. Thus, a code using this will have a single call replacing a loop of level 1 calls or a doubly nested loop of scalar operations.

This change by itself is sufficient to result in large gains in performance on a number of machines – for example, from 72 to 251 megaflops for a matrix of order 500 on one processor of a CRAY Y-MP. Since this is 81% of the peak speed of matrix–matrix multiplication on this processor, we cannot hope to do very much better by using level 3 BLAS.

We can, however, restructure the algorithm at a deeper level to exploit the faster speed of the level 3 BLAS. This restructuring involves recasting the algorithm as a *block algorithm* – that is, an algorithm that operates on *blocks* or submatrices of the original matrix.

#### 2.2.1. Deriving a block algorithm

To derive a block form of Cholesky factorization, we partition the matrices as shown in Fig. 1, in which the diagonal blocks of A and U are square, but of differing sizes. We assume that the first block has already been factored as  $A_{00} = U_{00}^{T}U_{00}$ , and that we now want to determine the second block column of U consisting of the blocks  $U_{01}$  and  $U_{11}$ . Equating submatrices in the second block of columns, we obtain

$$A_{01} = U_{00}^{\mathrm{T}} U_{01},$$

$$A_{11} = U_{01}^{\mathrm{T}} U_{01} + U_{11}^{\mathrm{T}} U_{11}$$

A <sub>00</sub>	A <sub>01</sub>	A <sub>02</sub>	$U_{00}^{T}$	0	0		U <sub>00</sub>	U <sub>01</sub>	U <sub>02</sub>
$A_{01}^T$	A <sub>11</sub>	A <sub>12</sub>	$U_{01}^T$	$U_{11}^T$	0	*	0	U <sub>11</sub>	U <sub>12</sub>
$A_{\scriptscriptstyle 02}^{T}$	$A_{12}^{T}$	A <sub>22</sub>	$U_{02}^{T}$	U <sup>T</sup> <sub>12</sub>	$U_{22}^{T}$	Φ	0	0	U <sub>22</sub>

Fig. 1. Partitioning of A,  $U^{T}$ , and U into blocks. It is assumed that the first block has already been factored as  $A_{00} = U_{00}^{T} U_{00}$ , and we next want to determine the block column consisting of  $U_{01}$  and  $U_{11}$ . Note that the diagonal blocks of A and U are square matrices.

	CRAY T-90	CRAY T-90		
	1 proc.	4 proc.		
<i>j</i> -variant: LINPACK	376	392		
<i>j</i> -variant: using level 3 BLAS	1222	2306		
i-variant: using level 3 BLAS	1297	3279		

Table 1 Speed (Megaflops) of Cholesky factorization  $A = U^{T}U$  for n = 500

Hence, since  $U_{00}$  has already been computed, we can compute  $U_{01}$  as the solution to the equation

$$U_{00}^{\mathrm{T}}U_{01} = A_{01}$$

by a call to the level 3 BLAS routine STRSM; and then we can compute  $U_{11}$  from

$$U_{11}^{\mathrm{T}}U_{11} = A_{11} - U_{01}^{\mathrm{T}}U_{01}.$$

This involves first updating the symmetric submatrix  $A_{11}$  by a call to the level 3 BLAS routine SSYRK, and then computing its Cholesky factorization. Since Fortran does not allow recursion, a separate routine must be called, using level 2 BLAS rather than level 3. In this way, successive blocks of columns of U are computed.

But that is not the end of the story, and the code given above is not the code actually used in the LAPACK routine SPOTRF. We mentioned earlier that for many linear algebra computations there are several algorithmic variants, often referred to as i-, j-, and k-variants, according to a convention introduced in [15,20] and explored further in [53,54]. The same is true of the corresponding block algorithms.

It turns out that the *j*-variant chosen for LINPACK, and used in the above examples, is not the fastest on many machines, because it performs most of the work in solving triangular systems of equations, which can be significantly slower than matrix-matrix multiplication. The variant actually used in LAPACK is the *i*-variant, which relies on matrix-matrix multiplication for most of the work.

Table 1 summarizes the results.

#### 3. The influence of computer architecture on performance

#### 3.1. Discussion of architectural features

In Section 2.1.1 we noted that for BLAS levels 2 and 3 several equivalent implementations of the operations exist. These differ, for instance, in whether they access a matrix operand by rows or columns in the inner loop. In FORTRAN, matrices are stored by columns, so accessing a column corresponds to accessing consecutive memory elements. On the other hand, as one proceeds across a row, the memory references jump across memory, the length of the jump being proportional to the length of a column.

We will now give a simplified discussion on the various architectural issues that influence the choice of algorithm. The following is, of necessity, a simplified account of the state of affairs for any particular architecture.

At first, we concentrate only on 'nonblocked' algorithms. In blocked methods, discussed in more detail below, every algorithm has two levels on which we can consider loop arranging: the block level, and the scalar level. Often, the best arrangement on one level is not the best on the other. The next two subsections concern themselves with the scalar level.

#### 3.1.1. Using consecutive elements

The decision how to traverse matrix elements should usually be taken so as to use elements that are consecutive in storage. There are at least three architectural reasons for this.

*Page swapping*: By using consecutive memory elements, instead of ones at some stride distance of each other, the amount of memory page swapping is minimized.

*Memory banks*: If the processor cycle is faster than the memory cycle, and memory consists of interleaved banks, consecutive elements will be in different banks. By contrast, taking elements separated a distance equal to the number of banks, all elements will come from the same bank. This will reduce the effective performance of the algorithm to the memory speed instead of the processor speed.

*Cache lines*: Processors with a memory cache typically do not bring in single elements from memory to cache, but move them one 'cache line' at a time. A cache line consists of a small number of consecutive memory elements. Thus, using consecutive memory storage elements means that a next element will already be in cache and does not have to be brought into cache. This cuts down on memory traffic.

Whether consecutive elements correspond to rows or columns in a matrix depends on the programming language used. In Fortran, columns are stored consecutively, whereas C has row elements contiguous in memory.

The effects of column orientation are quite dramatic: on systems with virtual or cache memories, the LINPACK library codes (Section 4.4.2), which are written in FORTRAN and which are column-oriented, will significantly outperform FORTRAN codes that are not column-oriented. In the C language, however, algorithms should be formulated with row-orientation. We note that textbook examples of matrix algorithms are usually given in a row-oriented manner.

# 3.1.2. Cache reuse

In many contemporary architectures, memory bandwidth is not enough to keep the processor working at its peak rate. Therefore, the architecture incorporates some cache memory, a relatively small store of faster memory. The memory bandwidth problem is now shifted to bringing the elements into cache, and this problem can be obviated almost entirely if the algorithm can re-use cache elements.

Consider for instance a matrix-vector product y = Ax. The doubly nested loop has an inner statement

$$y_i \leftarrow y_i + a_{ij}a_j$$

implying three reads and one write from memory for two operations. If we write the algorithm as

$$y_* = x_1 a_{1*} + x_2 a_{2*} + \cdots,$$

we see that, keeping y in cache<sup>1</sup> and reusing the elements of x, we only need to load the column of A, making the asymptotic demand on memory one element load once x and y have been brought into cache.

#### 3.1.3. Blocking for cache reuse

Above, we saw in the Cholesky example how algorithms can naturally be written in terms of level 2 operations. In order to use level 3 operations, a more drastic rewrite is needed.

Suppose we want to perform the matrix-matrix multiplication C = AB, where all matrices are of size  $n \times n$ . We divide all matrices in subblocks of size  $k \times k$ , and let for simplicity's sake k divide n: n = km. Then the triply nested scalar loop becomes, in one possible rearrangement

for 
$$i = 1 \dots m$$
  
for  $k = 1 \dots m$   
for  $j = 1 \dots m$   
 $C_{ij} \leftarrow C_{ij} + A_{ik}B_{kj}$ 

where the inner statement is now a size k matrix-matrix multiplication.

If the cache is now large enough for three of these smaller matrices, we can keep  $C_{ij}$  and  $A_{ik}$  in cache,<sup>2</sup> while successive blocks  $B_{kj}$  are being brought in. The ratio of memory loads to operations is then (ignoring the loads of the elements of C and A, which is amortised)  $k^2/k^3$ , that is, 1/k.

Thus, by blocking the algorithm, and arranging the loops so that blocks are reused in cache, we can achieve high performance in spite of a low-memory bandwidth.

# 3.2. Target architectures

The EISPACK and LINPACK software libraries were designed for supercomputers used in the 1970s and early 1980s, such as the CDC-7600, Cyber 205, and Cray-1. These machines featured multiple functional units pipelined for good performance [41]. The CDC-7600 was basically a high-performance scalar computer, while the Cyber 205 and Cray-1 were early vector computers.

The development of LAPACK in the late 1980s was intended to make the EISPACK and LIN-PACK libraries run efficiently on shared memory, vector supercomputers. The ScaLAPACK software library will extend the use of LAPACK to distributed memory concurrent supercomputers. The development of ScaLAPACK began in 1991 and is had its first public release of software by the end of 1994.

The underlying concept of both the LAPACK and ScaLAPACK libraries is the use of blockpartitioned algorithms to minimize data movement between different levels in hierarchical memory. Thus, the ideas discussed in this chapter for developing a library for dense linear algebra computations are applicable to any computer with a hierarchical memory that (1) imposes a sufficiently large startup cost on the movement of data between different levels in the hierarchy, and for which (2) the cost of a context switch is too great to make fine grain size multithreading worthwhile. Our target machines are, therefore, medium and large grain size advanced-architecture computers. These

<sup>&</sup>lt;sup>1</sup>Since many level-1 caches are write-through, we would not actually keep y in cache, but rather keep a number of elements of it in register, and reuse these registers by unrolling the '\*' loop.

<sup>&</sup>lt;sup>2</sup> Again, with a write-through level-1 cache, one would try to keep  $C_{ij}$  in registers.

include "traditional" shared memory, vector supercomputers, such as the Cray C-90 and T-90, and MIMD distributed memory concurrent supercomputers, such as the SGI Origin 2000, IBM SP, Cray T3E, and HP/Convex Exemplar concurrent systems.

Future advances in compiler and hardware technologies are expected to make multithreading a viable approach for masking communication costs. Since the blocks in a block-partitioned algorithm can be regarded as separate threads, our approach will still be applicable on machines that exploit medium and coarse grain size multithreading.

# 4. Dense linear algebra libraries

# 4.1. Requirements on high-quality, reusable, mathematical software

In developing a library of high-quality subroutines for dense linear algebra computations the design goals fall into three broad classes:

- performance,
- ease-of-use,
- range-of-use.

# 4.1.1. Performance

Two important performance metrics are *concurrent efficiency* and *scalability*. We seek good performance characteristics in our algorithms by eliminating, as much as possible, overhead due to load imbalance, data movement, and algorithm restructuring. The way the data are distributed (or decomposed) over the memory hierarchy of a computer is of fundamental importance to these factors. Concurrent efficiency,  $\varepsilon$ , is defined as the concurrent speedup per processor [32], where the concurrent speedup is the execution time,  $T_{seq}$ , for the best sequential algorithm running on one processor of the concurrent computer, divided by the execution time, T, of the parallel algorithm running on  $N_p$  processors. When direct methods are used, as in LU factorization, the concurrent efficiency depends on the problem size and the number of processors, so on a given parallel computer and for a fixed number of processors, the running time should not vary greatly for problems of the same size. Thus, we may write

$$\varepsilon(N, N_{\rm p}) = \frac{1}{N_{\rm p}} \frac{T_{\rm seq}(N)}{T(N, N_{\rm p})},\tag{1}$$

where N represents the problem size. In dense linear algebra computations, the execution time is usually dominated by the floating-point operation count, so the concurrent efficiency is related to the performance, G, measured in floating-point operations per second by

$$G(N, N_{\rm p}) = \frac{N_{\rm P}}{t_{\rm calc}} \varepsilon(N, N_{\rm p}), \tag{2}$$

where  $t_{calc}$  is the time for floating-point operation. For iterative routines, such as eigensolvers, the number of iterations, and hence the execution time, depends not only on the problem size, but also on other characteristics of the input data, such as condition number. A parallel algorithm is said to be scalable [34] if the concurrent efficiency depends on the problem size and number of processors

only through their ratio. This ratio is simply the problem size per processor, often referred to as the granularity. Thus, for a scalable algorithm, the concurrent efficiency is constant as the number of processors increases while keeping the granularity fixed. Alternatively, Eq. (2) shows that this is equivalent to saying that, for a scalable algorithm, the performance depends linearly on the number of processors for fixed granularity.

### 4.1.2. Ease-of-use

Ease-of-use is concerned with factors such as portability and the user interface to the library. Portability, in its most inclusive sense, means that the code is written in a standard language, such as Fortran or C, and that the source code can be compiled on an arbitrary machine to produce a program that will run correctly. We call this the "mail-order software" model of portability, since it reflects the model used by software servers such as *netlib* [19]. This notion of portability is quite demanding. It requires that all relevant properties of the computer's arithmetic and architecture be discovered at runtime within the confines of a compilable Fortran code. For example, if it is important to know the overflow threshold for scaling purposes, it must be determined at runtime *without overflowing*, since overflow is generally fatal. Such demands have resulted in quite large and sophisticated programs [24,44], which must be modified frequently to deal with new architectures and software releases. This "mail-order" notion of software portability also means that codes generally must be written for the worst possible machine expected to be used, thereby often degrading performances on all others. Ease-of-use is also enhanced if implementation details are largely hidden from the user, for example, through the use of an object-based interface to the library [22].

# 4.1.3. Range-of-use

Range-of-use may be gauged by how numerically stable the algorithms are over a range of input problems, and the range of data structures the library will support. For example, LINPACK and EISPACK deal with dense matrices stored in a rectangular array, packed matrices where only the upper- or lower-half of a symmetric matrix is stored, and banded matrices where only the nonzero bands are stored. In addition, some special formats such as Householder vectors are used internally to represent orthogonal matrices. In the second half of this paper we will focus on sparse matrices, that is matrices with many zero elements, which may be stored in many different ways.

# 4.2. Portability, scalability, and standards

Portability of programs has always been an important consideration. Portability was easy to achieve when there was a single architectural paradigm (the serial von Neumann machine) and a single programming language for scientific programming (Fortran) embodying that common model of computation. Architectural and linguistic diversity have made portability much more difficult, but no less important, to attain. Users simply do not wish to invest significant amounts of time to create large-scale application codes for each new machine. Our answer is to develop portable software libraries that hide machine-specific details.

In order to be truly portable, parallel software libraries must be *standardized*. In a parallel computing environment in which the higher-level routines and/or abstractions are built upon lower-level computation and message-passing routines, the benefits of standardization are particularly apparent.

Furthermore, the definition of computational and message-passing standards provides vendors with a clearly defined base set of routines that they can implement efficiently.

From the user's point of view, portability means that, as new machines are developed, they are simply added to network, supplying cycles where they are most appropriate.

From the mathematical software developer's point of view, portability may require significant effort. Economy in development and maintenance of mathematical software demands that such development effort be leveraged over as many different computer systems as possible. Given the great diversity of parallel architectures, this type of portability is attainable to only a limited degree, but machine dependences can at least be isolated.

LAPACK is an example of a mathematical software package whose highest-level components are portable, while machine dependences are hidden in lower-level modules. Such a hierarchical approach is probably the closest one can come to software portability across diverse parallel architectures. And the BLAS that are used so heavily in LAPACK provide a portable, efficient, and flexible standard for applications programmers.

Like portability, *scalability* demands that a program be reasonably effective over a wide range of number of processors. Maintaining scalability of parallel algorithms, and the software libraries implementing them, over a wide range of architectural designs and numbers of processors will likely require that the fundamental granularity of computation be adjustable to suit the particular circumstances in which the software may happen to execute. Our approach to this problem is block algorithms with adjustable block size. In many cases, however, polyalgorithms<sup>3</sup> may be required to deal with the full range of architectures and processor multiplicity likely to be available in the future.

Scalable parallel architectures of the future are likely to be based on a distributed memory architectural paradigm. In the longer term, progress in hardware development, operating systems, languages, compilers, and communications may make it possible for users to view such distributed architectures (without significant loss of efficiency) as having a shared memory with a global address space. For the near term, however, the distributed nature of the underlying hardware will continue to be visible at the programming level; therefore, efficient procedures for explicit communication will continue to be necessary. Given this fact, standards for basic message passing (send/receive), as well as higher-level communication constructs (global summation, broadcast, etc.), become essential to the development of scalable libraries that have any degree of portability. In addition to standardizing general communication primitives, it may also be advantageous to establish standards for problem-specific constructs in commonly occurring areas such as linear algebra.

The basic linear algebra communication subprograms (BLACS) [16,23] is a package that provides the same ease of use and portability for MIMD message-passing linear algebra communication that the BLAS [17,18,46] provide for linear algebra computation. Therefore, we recommend that future software for dense linear algebra on MIMD platforms consist of calls to the BLAS for computation and calls to the BLACS for communication. Since both packages will have been optimized for a particular platform, good performance should be achieved with relatively little effort. Also, since both packages will be available on a wide variety of machines, code modifications required to change platforms should be minimal.

<sup>&</sup>lt;sup>3</sup> In a polyalgorithm the actual algorithm used depends on the computing environment and the input data. The optimal algorithm in a particular instance is automatically selected at runtime.

# 4.3. The BLAS as the key to portability

At least three factors affect the performance of compilable code:

- 1. *Vectorization/cache reuse*: Designing vectorizable algorithms in linear algebra is usually straightforward. Indeed, for many computations there are several variants, all vectorizable, but with different characteristics in performance (see, for example, [15]). Linear algebra algorithms can approach the peak performance of many machines – principally because peak performance depends on some form of chaining of vector addition and multiplication operations or cache reuse, and this is just what the algorithms require. However, when the algorithms are realized in straightforward Fortran77 or C code, the performance may fall well short of the expected level, usually because Fortran compilers fail to minimize the number of memory references – that is, the number of vector load and store operations or effectively reuse cache.
- 2. *Data movement*: What often limits the actual performance of a vector, or scalar, floating-point unit is the rate of transfer of data between different levels of memory in the machine. Examples include the transfer of vector operands in and out of vector registers, the transfer of scalar operands in and out of a high speed cache, the movement of data between main memory and a high-speed cache or local memory, paging between actual memory and disk storage in a virtual memory system, and interprocessor communication on a distributed memory concurrent computer.
- 3. *Parallelism*: The nested loop structure of most linear algebra algorithms offers considerable scope for loop-based parallelism. This is the principal type of parallelism that LAPACK and ScaLA-PACK presently aim to exploit. On shared memory concurrent computers, this type of parallelism can sometimes be generated automatically by a compiler, but often requires the insertion of compiler directives. On distributed memory concurrent computers, data must be moved between processors. This is usually done by explicit calls to message passing routines, although parallel language extensions such as and Coherent Parallel C [30] and Split-C [13] do the message passing implicitly.

These issues can be controlled, while obtaining the levels of performance that machines can offer, through use of the BLAS, introduced in Section 2.1.

Level 1 BLAS are used in LAPACK, but for convenience rather than for performance: they perform an insignificant fraction of the computation, and they cannot achieve high efficiency on most modern supercomputers. Also, the overhead entailed in calling the BLAS reduces the efficiency of the code. This reduction is negligible for large matrices, but it can be quite significant for small matrices. Fortunately, level 1 BLAS can be removed from the smaller, more frequently used LAPACK codes in a short editing session.

Level 2 BLAS can achieve near-peak performance on many vector processors, such as a single processor of a CRAY X-MP or Y-MP, or Convex C-2 machine. However, on other vector processors such as a CRAY-2 or an IBM 3090 VF, the performance of level 2 BLAS is limited by the rate of data movement between different levels of memory.

Level 3 BLAS overcome this limitation. Level 3 of BLAS performs  $O(n^3)$  floating-point operations on  $O(n^2)$  data, whereas level 2 BLAS perform only  $O(n^2)$  operations on  $O(n^2)$  data. Level 3 BLAS also allow us to exploit parallelism in a way that is transparent to the software that calls them. While Level 2 BLAS offer some scope for exploiting parallelism, greater scope is provided by Level 3 BLAS, as Table 2 illustrates.

16
10
11207
14282
6953
13281
1558
14009

Table 2 Speed in Mflop/s of level 2 and level 3 BLAS operations on a CRAY C90 (all matrices are of order 1000; U is upper triangular)

The BLAS can provide portable high performance through being a standard that is available on many platforms. Ideally, the computer manufacturer has provided an assembly coded BLAS tuned for that particular architecture, but there is a standard implementation available that can simply be compiled and linked. Using this standard BLAS may improve the efficiency of programs when they are run on nonoptimizing compilers. This is because doubly subscripted array references in the inner loop of the algorithm are replaced by singly subscripted array references in the appropriate BLAS. The effect can be seen for matrices of quite small order, and for large orders the savings are quite significant.

# 4.4. Overview of dense linear algebra libraries

Over the past 25 years, we have been directly involved in the development of several important packages of dense linear algebra software: EISPACK, LINPACK, LAPACK, and the BLAS. Most recently, we have been involved in the development of ScaLAPACK, a scalable version of LA-PACK for distributed memory concurrent computers. In this section, we give a brief review of these packages – their history, their advantages, and their limitations on high-performance computers.

# 4.4.1. EISPACK

EISPACK is a collection of Fortran subroutines that compute that eigenvalues and eigenvectors of nine classes of matrices: complex general, complex Hermitian, real general, real symmetric, real symmetric banded, real symmetric tridiagonal, special real tridiagonal, generalized real, and generalized real symmetric matrices. In addition, two routines are included that use singular value decomposition to solve certain least-squares problems.

EISPACK is primarily based on a collection of Algol procedures developed in the 1960s and collected by J.H. Wilkinson and C. Reinsch in a volume entitled *Linear Algebra* in the *Handbook for Automatic Computation* [64] series. This volume was not designed to cover every possible method of solution; rather, algorithms were chosen on the basis of their generality, elegance, accuracy, speed, or economy of storage.

Since the release of EISPACK in 1972, over 10000 copies of the collection have been distributed worldwide.

# 4.4.2. LINPACK

LINPACK is a collection of Fortran subroutines that analyze and solve linear equations and linear least-squares problems. The package solves linear systems whose matrices are general, banded, symmetric indefinite, symmetric positive-definite, triangular, and tridiagonal square. In addition, the package computes the QR and singular-value decompositions of rectangular matrices and applies them to least-squares problems.

LINPACK is organized around four matrix factorizations: LU factorization, pivoted Cholesky factorization, QR factorization, and singular value decomposition. The term LU factorization is used here in a very general sense to mean the factorization of a square matrix into a lower triangular part and an upper triangular part, perhaps with pivoting. Next, we describe the organization and factors influencing LINPACK's efficiency.

LINPACK uses column-oriented algorithms to increase efficiency by preserving locality of reference. By column orientation we mean that the LINPACK codes always reference arrays down columns, not across rows. This works because Fortran stores arrays in column major order. This means that as one proceeds down a column of an array, the memory references proceed sequentially in memory. Thus, if a program references an item in a particular block, the next reference is likely to be in the same block. See further Section 3.1.1. LINPACK uses level 1 BLAS; see Section 4.3.

Since the release of LINPACK, over 20000 copies of the collection have been distributed world-wide.

# 4.4.3. LAPACK

LAPACK [14] provides routines for solving systems of simultaneous linear equations, least-squares solutions of linear systems of equations, eigenvalue problems, and singular-value problems. The associated matrix factorizations (LU, Cholesky, QR, SVD, Schur, generalized Schur) are also provided, as are related computations such as reordering of the Schur factorizations and estimating condition numbers. Dense and banded matrices are handled, but not general sparse matrices. In all areas, similar functionality is provided for real and complex matrices, in both single and double precision.

The original goal of the LAPACK project was to make the widely used EISPACK and LIN-PACK libraries run efficiently on shared-memory vector and parallel processors. On these machines, LINPACK and EISPACK are inefficient because their memory access patterns disregard the multilayered memory hierarchies of the machines, thereby spending too much time moving data instead of doing useful floating-point operations. LAPACK addresses this problem by reorganizing the algorithms to use block matrix operations, such as matrix multiplication, in the innermost loops [2,14]. These block operations can be optimized for each architecture to account for the memory hierarchy [1], and so provide a transportable way to achieve high efficiency on diverse modern machines. Here, we use the term "transportable" instead of "portable" because, for fastest possible performance, LAPACK requires that highly optimized block matrix operations be already implemented on each machine. In other words, the correctness of the code is portable, but high performance is not – if we limit ourselves to a single Fortran source code.

LAPACK can be regarded as a successor to LINPACK and EISPACK. It has virtually all the capabilities of these two packages and much more besides. LAPACK improves on LINPACK and EISPACK in four main respects: speed, accuracy, robustness and functionality. While LINPACK

and EISPACK are based on the vector operation kernels of level 1 BLAS, LAPACK was designed at the outset to exploit level 3 BLAS – a set of specifications for Fortran subprograms that do various types of matrix multiplication and the solution of triangular systems with multiple right-hand sides. Because of the coarse granularity of level 3 BLAS operations, their use tends to promote high efficiency on many high-performance computers, particularly if specially coded implementations are provided by the manufacturer.

LAPACK is designed to give high efficiency on vector processors, high-performance "superscalar" workstations, and shared memory multiprocessors. LAPACK in its present form is less likely to give good performance on other types of parallel architectures (for example, massively parallel SIMD machines, or MIMD distributed memory machines), but the ScaLAPACK project, described in Section 4.4.4, is intended to adapt LAPACK to these new architectures. LAPACK can also be used satisfactorily on all types of scalar machines (PCs, workstations, mainframes).

LAPACK, like LINPACK, provides LU and Cholesky factorizations of band matrices. The LIN-PACK algorithms can easily be restructured to use level 2 BLAS, though restructuring has little effect on performance for matrices of very narrow bandwidth. It is also possible to use level 3 BLAS, at the price of doing some extra work with zero elements outside the band [21]. This process becomes worthwhile for large matrices and semi-bandwidth greater than 100 or so.

### 4.4.4. ScaLAPACK

The ScaLAPACK software library extends the LAPACK library to run scalably on MIMD, distributed memory, concurrent computers [10,11]. For such machines the memory hierarchy includes the off-processor memory of other processors, in addition to the hierarchy of registers, cache, and local memory on each processor. Like LAPACK, the ScaLAPACK routines are based on block-partitioned algorithms in order to minimize the frequency of data movement between different levels of the memory hierarchy. The fundamental building blocks of the ScaLAPACK library are distributed memory versions of levels 2 and 3 BLAS, and a set of BLACS [16,23] for communication tasks that arise frequently in parallel linear algebra computations. In the ScaLA-PACK routines, all interprocessor communication occurs within the distributed BLAS and BLACS, so the source code of the top software layer of ScaLAPACK looks very similar to that of LAPACK.

#### 5. Future research directions in dense algorithms

Traditionally, large, general-purpose mathematical software libraries have required users to write their own programs that call library routines to solve specific subproblems that arise during a computation. Adapted to a shared-memory parallel environment, this conventional interface still offers some potential for hiding underlying complexity. For example, the LAPACK project incorporates parallelism in level 3 BLAS, where it is not directly visible to the user.

But when going from shared-memory systems to the more readily scalable distributed memory systems, the complexity of the distributed data structures required is more difficult to hide from the user. Not only must the problem decomposition and data layout be specified, but different phases of the user's problem may require transformations between different distributed data structures.

These deficiencies in the conventional user interface have prompted extensive discussion of alternative approaches for scalable parallel software libraries of the future. Possibilities include:

- 1. Traditional function library (i.e., minimum possible change to the status quo in going from serial to parallel environment). This will allow one to protect the programming investment that has been made.
- 2. Reactive servers on the network. A user would be able to send a computational problem to a server that was specialized in dealing with the problem. This fits well with the concepts of a networked, heterogeneous computing environment with various specialized hardware resources (or even the heterogeneous partitioning of a single homogeneous parallel machine).
- 3. General interactive environments like Matlab or Mathematica, perhaps with "expert" drivers (i.e., knowledge-based systems). With the growing popularity of the many integrated packages based on this idea, this approach would provide an interactive, graphical interface for specifying and solving scientific problems. Both the algorithms and data structures are hidden from the user, because the package itself is responsible for storing and retrieving the problem data in an efficient, distributed manner. In a heterogeneous networked environment, such interfaces could provide seamless access to computational engines that would be invoked selectively for different parts of the user's computation according to which machine is most appropriate for a particular subproblem.
- 4. Domain-specific problem solving environments, such as those for structural analysis. Environments like Matlab and Mathematica have proven to be especially attractive for rapid prototyping of new algorithms and systems that may subsequently be implemented in a more customized manner for higher performance.
- 5. Reusable templates (i.e., users adapt "source code" to their particular applications). A template is a description of a general algorithm rather than the executable object code or the source code more commonly found in a conventional software library. Nevertheless, although templates are general descriptions of key data structures, they offer whatever degree of customization the user may desire.

Novel user interfaces that hide the complexity of scalable parallelism will require new concepts and mechanisms for representing scientific computational problems and for specifying how those problems relate to each other. Very high level languages and systems, perhaps graphically based, not only would facilitate the use of mathematical software from the user's point of view, but also would help to automate the determination of effective partitioning, mapping, granularity, data structures, etc. However, new concepts in problem specification and representation may also require new mathematical research on the analytic, algebraic, and topological properties of problems (e.g., existence and uniqueness).

We have already begun work on developing such templates for sparse matrix computations. Future work will focus on extending the use of templates to dense matrix computations.

We hope the insight we gained from our work will influence future developers of hardware, compilers and systems software so that they provide tools to facilitate development of high quality portable numerical software.

The EISPACK, LINPACK, and LAPACK linear algebra libraries are in the public domain, and are available from *netlib*. For example, for more information on how to obtain LAPACK, send the following one-line email message to netlib@ornl.gov:

send index from lapack

or visit the web site at *http://www.netlib.org/lapack/*. Information for EISPACK, LINPACK, and ScaLAPACK can be similarly obtained.

# 6. Sparse linear algebra methods

# 6.1. Origin of sparse linear systems

The most common source of sparse linear systems is the numerical solution of partial differential equations. Many physical problems, such as fluid flow or elasticity, can be described by partial differential equations. These are implicit descriptions of a physical model, describing some internal relation such as stress forces. In order to arrive at an explicit description of the shape of the object or the temperature distribution, we need to solve the PDE, and for this we need numerical methods.

### 6.1.1. Discretized partial differential equations

Several methods for the numerical solution of PDEs exist, the most common ones being the methods of finite elements, finite differences, and finite volumes. A common feature of these is that they identify discrete points in the physical object, and give a set of equations relating these points.

Typically, only points that are physically close together are related to each other in this way. This gives a matrix structure with very few nonzero elements per row, and the nonzeros are often confined to a 'band' in the matrix.

#### 6.1.2. Sparse matrix structure

Matrices from discretized partial differential equations contain so many zero elements that it pays to find a storage structure that avoids storing these zeros. The resulting memory savings, however, are offset by an increase in programming complexity, and by decreased efficiency of even simple operations such as the matrix–vector product.

More complicated operations, such as solving a linear system, with such a sparse matrix present a next level of complication, as both the inverse and the LU factorization of a sparse matrix are not as sparse, thus needing considerably more storage. Specifically, the inverse of the type of sparse matrix we are considering is a full matrix, and factoring such a sparse matrix fills in the band completely.

**Example.** Central differences in *d* dimensions, *n* points per line, matrix size  $N = n^d$ , bandwidth  $q = n^{d-1}$  in natural ordering, number of nonzero  $\sim n^d$ , number of matrix elements  $N^2 = n^{2d}$ , number of elements in factorization  $N^{1+(d-1)/d}$ .

# 6.2. Basic elements in sparse linear algebra methods

Methods for sparse systems use, like those for dense systems, vector-vector, matrix-vector, and matrix-matrix operations. However, there are some important differences.

For iterative methods, discussed in Section 8, there are almost no matrix-matrix operations. See [43] for an exception. Since most modern architectures prefer these level 3 operations, the performance of iterative methods will be limited from the outset.

An even more serious objection is that the sparsity of the matrix implies that indirect addressing is used for retrieving elements. For example, in the popular row-compressed matrix storage format, the matrix–vector multiplication looks like

for 
$$i = 1 \dots n$$
  
 $p \leftarrow$  pointer to row  $i$   
for  $j = 1, n_i$   
 $y_i \leftarrow y_i + a(p+j)x(c(p+j))$ 

where  $n_i$  is the number of nonzeros in row *i*, and  $p(\cdot)$  is an array of column indices. A number of such algorithms for several sparse data formats are given in [6].

Direct methods can have a BLAS 3 component if they are a type of dissection method. However, in a given sparse problem, the more dense the matrices are, the smaller they are on average. They are also not general full matrices, but only banded. Thus, we do not expect very high performance on such methods either.

### 7. Direct solution methods

For the solution of a linear system one needs to factor the coefficient matrix. Any direct method is a variant of Gaussian elimination. As remarked above, for a sparse matrix, this fills in the band in which the nonzero elements are contained. In order to minimize the storage needed for the factorization, research has focused on finding suitable orderings of the matrix. Re-ordering the equations by a symmetric permutation of the matrix does not change the numerical properties of the system in many cases, and it can potentially give large savings in storage. In general, direct methods do not make use of the numerical properties of the linear system, and thus their execution time is affected in a major way by the structural properties of the input matrix.

# 7.1. Matrix graph theory

The most convenient way of talking about matrix orderings or permutations is to consider the matrix 'graph' [55]. We introduce a node for every physical variable, and nodes i and j are connected in the graph if the (i, j) element of the matrix is nonzero. A symmetric permutation of the matrix then corresponds to a numbering of the nodes, while the connections stay the same. With these permutations, one hopes to reduce the 'bandwidth' of the matrix, and thereby the amount of fill generated by the factorization.

# 7.2. Cuthill–McKee ordering

A popular ordering strategy is the Cuthill–McKee ordering, which finds levels or wavefronts in the matrix graph. This algorithm is easily described:

- 1. Take any node as starting point, and call that 'level 0'.
- 2. Now successively take all nodes connected to the previous level, and group them into the next level.

3. Iterate this until all nodes are grouped into some level; the numbering inside each level is of secondary importance.

This ordering strategy often gives a smaller bandwidth than the natural ordering and there are further advantages to having a level structure, e.g., for out-of-core solution or for parallel processing. Often, one uses the 'reverse Cuthill–Mckee' ordering [50].

#### 7.3. Minimum degree

An explicit reduction of bandwidth is effected by the minimum degree ordering, which at any point in the factorization chooses the variable with the smallest number of connections. Considering the size of the resulting fill-in is used as a tie breaker.

# 7.4. Nested dissection

Instead of trying to minimize fill-in by reducing the bandwidth, one could try a direct approach. The 'nested dissection' ordering recursively splits the matrix graph in two, thus separating it into disjoint subgraphs. Somewhat more precisely, given a graph, this algorithm relies on the existence of a 'separator': a set of nodes such that the other nodes fall into two mutually unconnected subgraphs. The fill from first factoring these subgraphs, followed by a factorization of the separator, is likely to be lower than for other orderings.

It can be shown that for PDEs in two space dimensions this method has a storage requirement that is within a log factor of that for the matrix itself, that is, very close to optimal [33]. This proof is easy for PDEs on rectangular grids, but with enough graph theory it can be generalized [48,49]. However, for problems in three space dimensions, the nested dissection method is no longer optimal.

An advantage of dissection-type methods is that they lead to large numbers of uncoupled matrix problems. Thus, to an extent, parallelization of such methods is easy. However, the higher levels in the tree quickly have fewer nodes than the number of available processors. In addition to this, they are also the larger subproblems in the algorithm, thereby complicating the parallelization of the method.

Another practical issue is the choice of the separator set. In a model case this is trivial, but in practice, and in particular in parallel, this is a serious problem, since the balancing of the two resulting subgraphs depends on this choice. Recently, the so-called 'second eigenvector methods' have become popular for this [56].

#### 8. Iterative solution methods

Direct methods, as sketched above, have some pleasant properties. Foremost is the fact that their time to solution is predictable, either a priori, or after determining the matrix ordering. This is due to the fact that the method does not rely on numerical properties of the coefficient matrix, but only on its structure. On the other hand, the amount of fill can be substantial, and with it the execution time. For large-scale applications, the storage requirements for a realistic size problem can simply be prohibitive.

Iterative methods have far lower storage demands. Typically, the storage, and the cost per iteration with it, is of the order of the matrix storage. However, the number of iterations strongly depends on properties of the linear system, and is at best known up to an order estimate; for difficult problems the methods may not even converge due to accumulated round-off errors.

#### 8.1. Basic iteration procedure

In its most informal sense, an iterative method in each iteration locates an approximation to the solution of the problem, measures the error between the approximation and the true solution, and based on the error measurement improves on the approximation by constructing a next iterate. This process repeats until the error measurement is deemed small enough.

#### 8.2. Stationary iterative methods

The simplest iterative methods are the 'stationary iterative methods'. They are based on finding a matrix M that is, in some sense, 'close' to the coefficient matrix A. Instead of solving Ax = b, which is deemed computationally infeasible, we solve  $Mx_1 = b$ . The true measure of how well  $x_1$ approximates x is the error  $e_1 = x_1 - x$ , but, since we do not know the true solution x, this quantity is not computable. Instead, we look at the 'residual':  $r_1 = Ae_1 = Ax_1 - b$ , which is a computable quantity. One easily sees that the true solution satisfies  $x = A^{-1}b = x_1 - A^{-1}r_1$ , so, replacing  $A^{-1}$ with  $M^{-1}$  in this relation, we define  $x_2 = x_1 - M^{-1}r_1$ .

Stationary methods are easily analyzed: we find that  $r_i \rightarrow 0$  if all eigenvalues  $\lambda = \lambda (I - AM^{-1})$  satisfy  $|\lambda| < 1$ . For certain classes of A and M this inequality is automatically satisfied [36,61].

#### 8.3. Krylov space methods

The most popular class of iterative methods nowadays is that of 'Krylov space methods'. The basic idea there is to construct the residuals such that *n*th residual  $r_n$  is obtained from the first by multiplication by some polynomial in the coefficient matrix A, that is,

$$r_n = P_{n-1}(A)r_1.$$

The properties of the method then follow from the properties of the actual polynomial [3,7,9].

Most often, these iteration polynomials are chosen such that the residuals are orthogonal under some inner product. From this, one usually obtains some minimization property, though not necessarily a minimization of the *error*.

Since the iteration polynomials are of increasing degree, it is easy to see that the main operation in each iteration is one matrix-vector multiplication. Additionally, some vector operations, including inner products in the orthogonalization step, are needed.

#### 8.3.1. The issue of symmetry

Krylov method residuals can be shown to satisfy the equation

 $r_n \in \operatorname{span}\{Ar_{n-1}, r_{n-1}, \ldots, r_1\}.$ 

This brings up the question whether all  $r_{n-1}, \ldots, r_1$  need to be stored in order to compute  $r_n$ . The answer is that this depends on the symmetry of the coefficient matrix. For a symmetric problem, the  $r_n$  vectors satisfy a three-term recurrence. This was the original conjugate gradient method [40].

For nonsymmetric problems, on the other hand, no short recurrences can exist [29], and therefore, all previous residuals need to be stored. Some of these methods are OrthoDir and OrthoRes [65].

If the requirement of orthogonality is relaxed, one can derive short-recurrence methods for nonsymmetric problems [31]. In the biconjugate gradient method, two sequences  $\{r_n\}$  and  $\{s_n\}$  are derived that are mutually orthogonal, and that satisfy three-term recurrences.

A disadvantage of this latter method is that it needs application of the transpose of the coefficient matrix. In environments where the matrix is only operatively defined, this may exclude this method from consideration. Recently developed methods, mostly based on the work of [59,60], obviate this consideration.

#### 8.3.2. True minimization

The methods mentioned so far minimize the error (over the subspace generated) in some matrixrelated norm, but not in the Euclidean norm. We can effect a true minimization by collecting the residuals generated so far, and finding a minimizing convex combination. This leads to one of the most popular methods nowadays: GMRES [58]. It will always generate the optimal iterate, but for this it requires storage of all previous residuals. In practice, truncated or restarted version of GMRES are popular.

# 8.4. Preconditioners

The matrix M that appeared in the section on stationary iterative methods can play a role in Krylov space methods too. There, it is called a 'preconditioner', and it acts to improve spectral properties of the coefficient matrix that determine the convergence speed of the method. In a slight simplification, one might say that we replace the system Ax = b by

$$(AM^{-1})(Mx) = b.$$

(Additionally, the inner product is typically changed.) It is generally recognized that a good preconditioner is crucial to the performance of an iterative method.

The requirements on a preconditioner are that it should be easy to construct, a system Mx = b should be simple to solve, and in some sense M should be an approximation to A. These requirements need to be balanced: a more accurate preconditioner is usually harder to construct and more costly to apply, so any decrease in the number iterations has to be set against a longer time per iteration, plus an increased setup phase.

The holy grail of preconditioners is finding an 'optimal' preconditioner: one for which the number of operations for applying it is of the order of the number of variables, while the resulting number of iterations is bounded in the problem size. There are very few optimal preconditioners.

# 8.4.1. Simple preconditioners

Some preconditioners need no construction at all. For instance, the Jacobi preconditioner consists of simply the matrix diagonal  $D_A$ . Since in PDE applications the largest elements are on the diagonal, one expects some degree of accuracy from this. Using not just the diagonal, but the whole lower
triangular part  $D_A + L_A$  of the coefficient matrix, an even more accurate method results. Since this triangular matrix is nonsymmetric, it is usually balanced with the upper triangular part as  $(D_A + L_A)D_A^{-1}(D_A + U_A)$ .

#### 8.4.2. Incomplete factorizations

A successful strategy for preconditioners results from mimicking direct methods, but applying some approximation process to them. Thus, the so-called 'incomplete factorization' methods ignore fill elements in the course of the Gaussian elimination process. Two strategies are to ignore elements in fixed positions, or to drop elements that are deemed small enough to be negligible. The aim is here to preserve at least some of the sparsity of the coefficient matrix in the factorization, while giving something that is close enough to the full factorization.

Incomplete factorizations can be very effective, but there are a few practical problems. For the class of M-matrices, these methods are well defined [52], but for other, even fairly common classes of matrices, there is a possibility that the algorithm breaks down [42,45,51].

Also, factorizations are inherently recursive, and coupled with the sparseness of the incomplete factorization, this gives very limited parallelism in the algorithm using a natural ordering of the unknowns. Different orderings may be more parallel, but take more iterations [25,27,43].

#### 8.4.3. Analytically inspired preconditioners

In recent years, a number of preconditioners have gained in popularity that are more directly inspired by the continuous problem. First of all, for a matrix from an elliptic PDE, one can use a so-called 'fast solver' as preconditioner [12,28,63].

A particularly popular class of preconditioners based on the continuous problem, is that of 'domain decomposition' methods. If the continuous problem was elliptic, then decomposing the domain into simply connected pieces leads to elliptic problems on these subdomains, tied together by internal boundary conditions of some sort.

For instance, in the Schur complement domain decomposition method [8], thin strips of variables are assigned a function as interface region, and the original problem reduces to fully independent problems on the subdomains, connected by a system on the interface that is both smaller and better conditioned, but more dense, than the original one. While the subdomains can trivially be executed in parallel, the interface system poses considerable problems.

Choosing overlapping instead of separated subdomains leads to the class of Schwarz method [47]. The original Schwarz method on two domains proposed solving one subdomain, deriving interface conditions from it for the other subdomain, and solving the system there. Repetition of this process can be shown to converge. In a more parallel variant of this method, all subdomains solve their system simultaneously, and the solutions on the overlap regions are added together.

Multilevel methods do not operate by decomposing the domain. Rather, they work on a sequence of nested discretization, solving the coarser ones as a starting point for solving the finer levels. Under certain conditions such methods can be shown to be close to optimal [4,35]. However, they require explicit knowledge of the operator and boundary conditions. For this reason, people have investigated algebraic variants [5,57]. In both cases, these methods can be parallelised by distributing each level over the processors, but this may not be trivial.

#### 9. Libraries and standards in sparse methods

Unlike in dense methods, there are few standards for iterative methods. Most of this is due to the fact that sparse storage is more complicated, admitting of more variation, and therefore less standardised. Whereas the (dense) BLAS has been accepted for a long time, sparse BLAS is not more than a proposal under research.

#### 9.1. Storage formats

As is apparent from the matrix-vector example in Section 6.2, storage formats for sparse matrices include not just the matrix elements, but pointer information describing where the nonzero elements are placed in the matrix. A few storage formats are in common use (for more details see [6]):

Aij format: In the 'Aij' format, three arrays of the same length are allocated: one containing the matrix elements, and the other two containing the i and j coordinates of these elements. No particular ordering of the elements is implied.

*Row/column-compressed*: In the row-compressed format one array of integers is allocated in addition to the matrix element, giving the column indices of the nonzero elements. Since all elements in the same row are stored contiguously, a second, smaller, array is needed giving the start points of the rows in the two larger arrays.

*Compressed diagonal*: If the nonzero elements of the matrix are located, roughly or exactly, along subdiagonals, one could use contiguous storage for these diagonals. There are several diagonal storage formats. In the simplest, describing a contiguous block of subdiagonals, only the array of matrix elements is needed; two integers are sufficient to describe which diagonals have been stored.

There exist blocked versions of these formats, for matrices that can be partitioned into small square subblocks.

#### 9.2. Sparse libraries

Since sparse formats are more complicated than dense matrix storage, sparse libraries have an added level of complexity. This holds even more so in the parallel case, where additional indexing information is needed to specify which matrix elements are on which processor.

There are two fundamentally different approaches for handling this complexity. Some sparse libraries require the user to set up the matrix and supply it to the library, while all handling is performed by the library. This requires the user to store data in a format dictated by the library, which might involve considerable work.

On the other hand, the library might do even the matrix setup internally, hiding all data from the user. This gives total freedom to the user, but it requires the library to supply sufficient access functions so that the user can perform certain matrix operations, even while not having access to the object itself.

#### References

[1] E. Anderson, J. Dongarra, Results from the initial release of LAPACK, Technical Report LAPACK Working Note 16, Computer Science Department, University of Tennessee, Knoxville, TN, 1989. http://www.netlib.org/lapack/lawns/lawn16.ps.

- [2] E. Anderson, J. Dongarra, Evaluating block algorithm variants in LAPACK, Technical Report LAPACK Working Note 19, Computer Science Department, University of Tennessee, Knoxville, TN, 1990. http://www.netlib.org/lapack/lawns/lawn19.ps.
- [3] O. Axelsson, A.V. Barker, Finite Element Solution of Boundary Value Problems, Theory and Computation, Academic Press, Orlando, FL, 1984.
- [4] O. Axelsson, P. Vassilevski, Algebraic multilevel preconditioning methods, I, Numer. Math. 56 (1989) 157-177.
- [5] O. Axelsson, V. Eijkhout, The nested recursive two-level factorization method for nine-point difference matrices, SIAM J. Sci. Statist. Comput. 12 (1991) 1373–1400.
- [6] R. Barrett, M. Berry, T.F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H. van der Vorst, Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, SIAM, Philadelphia, PA, 1994. http://www.netlib.org/templates/templates.ps.
- [7] G. Birkhoff, R.E. Lynch, Numerical Solution of Elliptic Problems, SIAM, Philadelphia, PA, 1984.
- [8] P. Bjørstad, O. Widlund, Iterative methods for the solution of elliptic problems on regions partitioned into substructures, SIAM J. Numer. Anal. 23 (1986) 1097–1120.
- [9] T. Chan, Henk van der Vorst, Linear system solvers: sparse iterative methods, in: D. Keyes et al (Eds.), Parallel Numerical Algorithms, Proceedings of the ICASW/LaRC Workshop on Parallel Numerical Algorithms, May 23–25, 1994, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1997, pp. 91–118.
- [10] J. Choi, J.J. Dongarra, R. Pozo, D.W. Walker, Scalapack: a scalable linear algebra library for distributed memory concurrent computers, Proceedings of the Fourth Symposium on the Frontiers of Massively Parallel Computation, IEEE Computer Society Press, Silver Spring, MD, 1992, pp. 120–127.
- [11] J. Choi, J.J. Dongarra, D.W. Walker, The design of scalable software libraries for distributed memory concurrent computers, in: J.J. Dongarra, B. Tourancheau (Eds.), Environments and Tools for Parallel Scientific Computing, Elsevier Science Publishers, Amsterdam, 1993.
- [12] P. Concus, Gene H. Golub, Use of fast direct methods for the efficient numerical solution of nonseparable elliptic equations, SIAM J. Numer. Anal. 10 (1973) 1103–1120.
- [13] D.E. Culler, A. Dusseau, S.C. Goldstein, A. Krishnamurthy, S. Lumetta, T. von Eicken, K. Yelick, Introduction to Split-C: Version 0.9, Technical Report, Computer Science Division – EECS, University of California, Berkeley, CA 94720, February 1993.
- [14] J. Demmel, LAPACK: a portable linear algebra library for supercomputers, Proceedings of the 1989 IEEE Control Systems Society Workshop on Computer-Aided Control System Design, December 1989.
- [15] J.J. Dongarra, Increasing the performance of mathematical software through high-level modularity, Proceedings of the Sixth International Symposium Comp. Methods in Engineering & Applied Sciences, Versailles, France, North-Holland, Amsterdam, 1984, pp. 239–248.
- [16] J.J. Dongarra, LAPACK Working Note 34: Workshop on the BLACS, Computer Science Department, Technical Report CS-91-134, University of Tennessee, Knoxville, TN, May 1991. http://www.netlib.org/lapack/lawns/ lawn16.ps.
- [17] J.J. Dongarra, J. Du Croz, S. Hammarling, I. Duff, A set of level 3 basic linear algebra subprograms, ACM Trans. Math. Software 16 (1) (1990) 1–17.
- [18] J.J. Dongarra, J. Du Croz, S. Hammarling, R. Hanson, An extended set of Fortran basic linear algebra subroutines, ACM Trans. Math. Software 14 (1) (1988) 1–17.
- [19] J.J. Dongarra, E. Grosse, Distribution of mathematical software via electronic mail, Comm. ACM 30 (5) (1987) 403–407.
- [20] J.J. Dongarra, F.C. Gustavson, A. Karp, Implementing linear algebra algorithms for dense matrices on a vector pipeline machine, SIAM Rev. 26 (1984) 91–112.
- [21] J.J. Dongarra, P. Mayes, Giuseppe Radicati di Brozolo, The IBM RISC System/6000 and linear algebra operations, Supercomputer 44 (VIII-4) (1991) 15–30.
- [22] J.J. Dongarra, R. Pozo, D.W. Walker, An object oriented design for high performance linear algebra on distributed memory architectures, Proceedings of the Object Oriented Numerics Conference, 1993.
- [23] J.J. Dongarra, R.A. van de Geijn, Two-dimensional basic linear algebra communication subprograms, Technical Report LAPACK Working Note 37, Computer Science Department, University of Tennessee, Knoxville, TN, October 1991.

- [24] J. Du Croz, M. Pont, The development of a floating-point validation package, in: M.J. Irwin, R. Stefanelli (Eds.), Proceedings of the Eighth Symposium on Computer Arithmetic, Como, Italy, May 19–21, 1987, IEEE Computer Society Press, Silver Spring, MD, 1987.
- [25] I.S. Duff, G.A. Meurant, The effect of ordering on preconditioned conjugate gradients, BIT 29 (1989) 635-657.
- [26] A. Edelman, Large dense numerical linear algebra in 1993: the parallel computing influence, Int. J. Supercomput. Appl. 7 (1993) 113–128.
- [27] V. Eijkhout, Analysis of parallel incomplete point factorizations, Linear Algebra Appl. 154–156 (1991) 723–740.
- [28] H.C. Elman, M.H. Schultz, Preconditioning by fast direct methods for non self-adjoint nonseparable elliptic equations, SIAM J. Numer. Anal. 23 (1986) 44–57.
- [29] V. Faber, T. Manteuffel, Orthogonal error methods, SIAM J. Numer. Anal. 24 (1987) 170-187.
- [30] E.W. Felten, S.W. Otto, Coherent parallel C, in: G.C. Fox (Ed.), Proceedings of the Third Conference on Hypercube Concurrent Computers and Applications, ACM Press, New York, 1988, pp. 440–450.
- [31] R. Fletcher, Conjugate gradient methods for indefinite systems, in: G.A. Watson (Ed.), Numerical Analysis Dundee, 1975, Springer, New York, 1976, pp. 73–89.
- [32] G.C. Fox, M.A. Johnson, G.A. Lyzenga, S.W. Otto, J.K. Salmon, D.W. Walker, Solving Problems on Concurrent Processors, Vol. 1, Prentice-Hall, Englewood Cliffs, NJ, 1988.
- [33] A. George, H.-W. Liu, Computer Solution of Large Sparse Positive Definite Systems, Prentice-Hall, Englewood Cliffs, NJ, 1981.
- [34] A. Gupta, V. Kumar, On the scalability of FFT on parallel computers, Proceedings of the Frontiers 90 Conference on Massively Parallel Computation, IEEE Computer Society Press, 1990. Also available as Technical Report TR 90-20 from the Computer Science Department, University of Minnesota, Minneapolis, MN 55455.
- [35] W. Hackbusch, Multi-Grid Methods and Applications, Springer, Berlin, 1985.
- [36] L.A. Hageman, D.M. Young, Applied Iterative Methods, Academic Press, New York, 1981.
- [37] W. Croswell, Origin and development of the method of moments for field computation, IEEE Antennas Propagation Mag. 32 (1990) 31–34.
- [38] J.L. Hess, Panel methods in computational fluid dynamics, Annu. Rev. Fluid Mech. 22 (1990) 255-274.
- [39] J.L. Hess, M.O. Smith, Calculation of potential flows about arbitrary bodies, in: D. Küchemann (Ed.), Progress in Aeronautical Sciences, Vol. 8, Pergamon Press, Oxford, 1967.
- [40] M.R. Hestenes, E. Stiefel, Methods of conjugate gradients for solving linear systems, Nat. Bur. Stand. J. Res. 49 (1952) 409–436.
- [41] R.W. Hockney, C.R. Jesshope, Parallel Computers, Adam Hilger, Bristol, UK, 1981.
- [42] A. Jennings, G.M. Malik, Partial elimination, J. Inst. Math. Appl. 20 (1977) 307-316.
- [43] M.T. Jones, P.E. Plassmann, Parallel solution of unstructured, sparse systems of linear equations, in: R.F. Sincovec, D.E. Keyes, M.R. Leuze, L.R. Petzold, D.A. Reed (Eds.), Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing, SIAM, Philadelphia, PA, pp. 471–475.
- [44] W. Kahan, Paranoia, Available from netlib [19]: http://www.netlib.org/paranoia.
- [45] D.S. Kershaw, The incomplete cholesky-conjugate gradient method for the iterative solution of systems of linear equations, J. Comput. Phys. 26 (1978) 43–65.
- [46] C. Lawson, R. Hanson, D. Kincaid, F. Krogh, Basic linear algebra subprograms for Fortran usage, ACM Trans. Math. Software 5 (1979) 308–323.
- [47] P.L. Lions, On the Schwarz alternating method. i., in: R. Glowinski, G.H. Golub, G. Meurant, J. Periaux (Eds.), Domain Decomposition Methods for Partial Differential Equations, Proceedings of the First Internation Symposium, Paris, January 7–9, 1987, SIAM, Philadelphia, PA, 1988, pp. 1–42.
- [48] R.J. Lipton, D.J. Rose, R.E. Tarjan, Generalized nested dissection, SIAM J. Numer. Anal. 16 (1979) 346-358.
- [49] R.J. Lipton, R.E. Tarjan, A separator theorem for planar graphs, SIAM J. Appl. Math. 36 (1979) 177–189.
- [50] J.W-H. Liu, A.H. Sherman, Comparative analysis of the Cuthill–McKee and the reverse Cuthill–McKee ordering algorithms for sparse matrices, SIAM J. Numer. Anal. 13 (1973) 198–213.
- [51] T.A. Manteuffel, An incomplete factorization technique for positive definite linear systems, Math. Comp. 34 (1980) 473–497.
- [52] J.A. Meijerink, H.A. van der Vorst, An iterative solution method for linear systems of which the coefficient matrix is a symmetric *m*-matrix, Math. Comp. 31 (1977) 148–162.
- [53] J.M. Ortega, The *ijk* forms of factorization methods I, Vector computers, Parallel Comput. 7 (1988) 135–147.

- [54] J.M. Ortega, C.H. Romine, The *ijk* forms of factorization methods II, Parallel systems, Parallel Comput. 7 (1988) 149–162.
- [55] S.V. Parter, The use of linear graphs in Gaussian elimination, SIAM Rev. 3 (1961) 119–130.
- [56] A. Pothen, H.D. Simon, Kang-Pu Liou, Partitioning sparse matrices with eigenvectors of graphs, SIAM J. Matrix Anal. Appl. 11 (3) (1990) 430–452.
- [57] J.W. Ruge, K. Stüben, Algebraic multigrid, in: S.F. McCormick (Ed.), Multigrid Methods, SIAM, Philadelphia, PA, 1987, (Chapter 4).
- [58] Y. Saad, M.H. Schultz, GMRes: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Statist. Comput. 7 (1986) 856–869.
- [59] P. Sonneveld, CGS, a fast Lanczos-type solver for nonsymmetric linear systems, SIAM J. Sci. Statist. Comput. 10 (1989) 36–52.
- [60] H. van der Vorst, Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems, SIAM J. Sci. Statist. Comput. 13 (1992) 631–644.
- [61] R.S. Varga, Matrix Iterative Analysis, Prentice-Hall, Englewood Cliffs, NJ, 1962.
- [62] J.J.H. Wang, Generalized Moment Methods in Electromagnetics, Wiley, New York, 1991.
- [63] O. Widlund, On the use of fast methods for separable finite difference equations for the solution of general elliptic problems, in: D.J. Rose, R.A. Willoughby (Eds.), Sparse Matrices and their Applications, Plenum Press, New York, 1972, pp. 121–134.
- [64] J. Wilkinson, C. Reinsch, Handbook for Automatic Computation: Vol. II Linear Algebra, Springer, New York, 1971.
- [65] D.M. Young, K.C. Jea, Generalized conjugate-gradient acceleration of nonsymmetrizable iterative methods, Linear Algebra Appl. 34 (1980) 159–194.



Journal of Computational and Applied Mathematics 123 (2000) 515-530

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.nl/locate/cam

# The impact of high-performance computing in the solution of linear systems: trends and problems $\stackrel{\text{trends}}{\approx}$

Iain S. Duff\*

Atlas Centre, Rutherford Appleton Laboratory, Computating & Information System Department, Oxon OX11 0QX, UK

Received 29 August 1999; received in revised form 25 October 1999

#### Abstract

We review the influence of the advent of high-performance computing on the solution of linear equations. We will concentrate on direct methods of solution and consider both the case when the coefficient matrix is dense and when it is sparse. We will examine the current performance of software in this area and speculate on what advances we might expect in the early years of the next century. © 2000 Elsevier Science B.V. All rights reserved.

#### MSC: 65F05; 65F50

Keywords: Sparse matrices; Direct methods; Parallelism; Matrix factorization; Multifrontal methods

#### 1. Introduction

In view of the other papers appearing in this volume, we will study only the solution of linear equations

$$Ax = b \tag{1.1}$$

using direct methods based on a factorization of the coefficient matrix A. We will consider both the case when A is dense and when it is sparse although we will concentrate more on the latter.

Although there are several ways to factorize a matrix, we will use the LU factorization

$$PAQ = LU, (1.2)$$

<sup>☆</sup> Current reports available by anonymous ftp to ftp.numerical.rl.ac.uk in directory pub/reports. This report is in file duffRAL99072.ps.gz. Report also available through URL http://www.numerical.rl.ac.uk/reports/reports.html. \* Tel.: +44-1235-445803; fax: +44-1235-446626.

*E-mail address:* i.s.duff@rl.ac.uk (I.S. Duff).

0377-0427/00/\$ - see front matter  $\odot$  2000 Elsevier Science B.V. All rights reserved. PII: \$0377-0427(00)00401-5

where P and Q are permutation matrices, L is a unit lower triangular matrix, and U is an upper triangular matrix. When A is a symmetric matrix, we use the analogous factorization

$$PAP^{\mathrm{T}} = LDL^{\mathrm{T}},\tag{1.3}$$

where D is a diagonal matrix, or possibly block diagonal (with blocks of order 1 and 2) if we want a stable factorization of an indefinite matrix [21].

We discuss the building blocks for both sparse and dense factorization in Section 2 and illustrate their use in dense factorization in Section 3. We then show how such building blocks can be used in sparse factorization in Section 4 indicating how this has revolutionized the performance of sparse codes. We discuss recent attempts to harness the power of parallel computers in Section 5 before examining the current power and limitations of direct methods in Section 6. We conclude with some remarks on the future in Section 7.

A wide range of iterative, direct, and preconditioning techniques with an emphasis on the exploitation of parallelism is considered at length in the recent book by Dongarra et al. [34]. A more recent bibliographic tour is presented by Duff and van der Vorst [44].

#### 2. Building blocks

A common feature of current high-performance machines is that the main obstacle to obtaining high performance is the bottleneck in getting data from the main memory to the functional units. This is true whether they are built from custom-made silicon or commodity chips and whether they are RISC processor workstations, pentium-based PCs, vector processors, or shared or distributed memory parallel computers. Most machines use a high-speed cache as a staging post. Data in this cache (many machines have multiple caches usually organized hierarchically but here we talk about the highest level cache) can be transferred at low latency and high bandwidth to the functional units but the amount of data that can be stored in the cache is quite small (often less than one Mbyte).

This means that if we want to obtain high performance relative to the peak of the machine, it is necessary to reuse data in the cache as much as possible to amortize the cost of getting it to the cache from main memory. The most suitable and widely used kernels for doing this are the Level 3 BLAS for  $O(n^3)$  operations involving matrices of order *n*. There are nine Level 3 BLAS kernels but the two that are most used in routines based on LU factorization are the matrix-matrix multiplication routine \_GEMM and the solution of a block of right-hand sides by a triangular system, \_TRSM, although the symmetric update routine, \_SYRK, can be used in a symmetric factorization.

We show, in Table 1, the performance of the Level 3 BLAS kernel \_GEMM on a range of computers with various floating-point chips and memory organizations. In many cases, this kernel attains about 90% or more of the peak performance of the chip and in every case more than 66% of the peak is achieved.

These building blocks have been discussed in detail in the paper by Dongarra and Eijkhout [32] so we do not discuss them further here other than to say that they can be used in factorization algorithms so that asymptotically the floating-point operations are all performed using these kernels.

Machine	Peak	_GEMM
Meiko CS2-HA	100	88
IBM SP2	266	232
Intel PARAGON	75	68
DEC Turbo Laser	600	450
CRAY C90	952	900
CRAY T3D	150	102

Performance of $\_GEMM$ kernel in Mflop/s on a	range of machines (single processor perfor-
mance). Matrices of order 500	

#### 3. Factorization of dense matrices

Table 1

To some extent, the algorithm and code development for numerical linear algebra have always been driven by developments in computer architectures. The first real library of subroutines for linear algebra on dense matrices was developed in Algol by Wilkinson and Reinsch [87]. These were used as the basis for the LINPACK project where a wide range of software for solving dense systems of equations was developed in Fortran and is described in the LINPACK book [30]. The LU factorization code has been used as a basis for the benchmarking of computers with the latest results being available on the World Wide Web [29]. The codes in the LINPACK package used Level 1 BLAS [68] and were portable over a wide range of machines. Although the Level 1 BLAS were ostensibly for vector operations, the LINPACK codes performed poorly on vector or cache-based machines. This was addressed in the development of the LAPACK package for linear algebra [14]. Codes in this package incorporated Level 2 and Level 3 BLAS ([31,33] respectively) and had a much improved performance on modern architectures. Many vendors of shared memory computers offer parallel versions of the BLAS and so, at this level, parallelization is trivial. However, LAPACK was not designed for parallel machines and, in particular, not for machines with distributed memory that use message passing to communicate data. This last class of machines is targeted by the ongoing ScaLAPACK project [19] that supports distributed computation using tools like the Basic Linear Algebra Communications Routines (BLACS) [86].

If we view the LU factorization in a blocked or partitioned way, it becomes relatively simple to show how Level 3 BLAS can be used. We show a schematic of block LU factorization in Fig. 1. These diagrams represent a single block stage of the factorization using three different approaches. Factorization operations are performed on the hatched regions and access is required to the other regions shaded in the matrix. For example, in the block right-looking LU factorization, the hatched block column is first factorized using the Level 2 BLAS algorithm described in Fig. 2, the hatched block row of U is computed using the Level 3 BLAS kernel \_TRSM and the shaded portion of the matrix updated using the \_GEMM kernel to multiply the hatched part of the block column beneath the diagonal block with this newly computed block row of U. Algorithms of this kind are the bases for the factorization routines within the LAPACK suite that are discussed in the article by Dongarra and Eijkhout [32].

Recently, Gustavson and his collaborators [13,58,59] have developed a recursive way of looking at the factorizations which has the effect of increasing the proportion of Level 3 operations and avoids



Fig. 1. Block variants of LU decomposition.

For each column, j, of the rectangular matrix in turn do Update the part of column j above the diagonal by solving a system whose right-hand side is the corresponding part of the original column and whose coefficient matrix is the lower triangular matrix from the previously computed columns in the block, using the Level 2 BLAS kernel \_TRSV. Update the lower part of column j by using the Level 2 BLAS kernel \_GEMV to multiply the rectangular matrix corresponding to rows jto n and columns 1 to j - 1 with the newly computed vector (and subtract this from the lower part of column j). Choose the pivot from this newly computed lower part of column j, swop its row with row j and scale the column below the diagonal.

Fig. 2. Level 2 factorization of rectangular block.

the necessity for choosing block sizes as in the abovementioned block algorithms. The recursive approach can be thought of by looking at the factorization at the halfway point so that the matrix can be partitioned as

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix},$$

where matrices  $A_{11}$  and  $A_{21}$  are factorized. At this stage, a Level-3-type algorithm can be used to update the blocks  $A_{12}$  and  $A_{22}$ , and  $A_{22}$  can then be factorized using a similar recursive algorithm. Of course, the first block columns were also factorized recursively in similar fashion. An added bonus of the recursive algorithm is that access to the blocks for Level 3 computations can be organized on contiguous data. For the earlier algorithms, the leading dimension of the arrays corresponding to the blocks is not equal to the block size. This is more noticeable in the recursive form of the Cholesky factorization.

#### 4. Factorization of sparse matrices

The two main books discussing the direct solution of sparse linear equations are those by George and Liu [50] and by Duff et al. [39]. The former restricts its discussion to symmetric positive-definite systems and emphasizes graph theoretic aspects, while the latter considers both symmetric and unsymmetric systems and includes a discussion of some of the algorithms used in the Harwell subroutine library (HSL) [64]. The HSL has arguably the largest number of direct sparse codes in a single library and has a few codes for iterative solution also. Information on this Library can be found in the Web pages http://www.cse.clrc.ac.uk/Activity/HSL. A subset of HSL is marketed by NAG as the Harwell Sparse Matrix Library (HSML). Other sparse direct software can be found through the netlib repository http://www.netlib.org.

When factorizing sparse matrices, it is crucial that the permutation matrices of (1.2) and (1.3) are chosen to preserve sparsity in the factors as well as to maintain stability and many algorithms have been developed to achieve this. In the general unsymmetric case, this leads to a need to compromise the numerical pivoting strategy in order to choose pivots to limit the fill-in. A common strategy for limiting fill-in, due to Markowitz [72], chooses entries so that the product of the number of other entries in the row and column of the candidate pivot is minimized. An entry is accepted as a pivot only if it is within a threshold of the largest in its column. The threshold is often an input parameter and a typical value for it is 0.1. This Markowitz-threshold strategy and a range of other similar possibilities are discussed in detail in [39]. Data structures are designed so that only the nonzero entries of the matrix and of the factors need to be held. This, coupled with the fact that it is often nontrivial to determine what part of the matrix is updated at each pivot step, has led to complicated algorithms and codes that are hard to implement efficiently on modern architectures [39].

In the symmetric case, the Markowitz analogue is *minimum degree* where one chooses as pivot a diagonal entry with the least number of entries in its row. This criterion was first proposed in 1967 [84] and has stood the test of time well. George [48] proposed a different class of orderings based on a nonlocal strategy of dissection. In his *nested dissection* approach, a set of nodes is selected to partition the graph, and this set is placed at the end of the pivotal sequence. The subgraphs corresponding to the partitions are themselves similarly partitioned and this process is nested with pivots being identified in reverse order. Minimum degree, nested dissection and several other symmetric orderings were included in the SPARSPAK package [49,51]. Many experiments were performed using the orderings in SPARSPAK and elsewhere, and the empirical experience at the beginning of the 1990s indicated that minimum degree was the best ordering method for general symmetric problems. We will return to this issue of ordering when we consider parallelism in Section 5.

It is not immediately or intuitively apparent that the kernels discussed in Section 2 can be used in the factorization of sparse matrices and indeed much of the work and heuristics developed in the 1970s attempted to do just the opposite, namely to perform the basic elimination operations on as sparse vectors as possible.

The most obvious way of using dense kernels in sparse factorization is to order the sparse matrix so that its nonzero entries are clustered near the diagonal (called bandwidth minimization) and then regard the matrix as banded, since zeros within the band soon fill-in. However, this is normally too wasteful as even the high computational rate of the Level 3 BLAS does not compensate for the extra work. A variable band format is used to extend the range of applicability of this technique. A related, but more flexible scheme, is the frontal method (for example, [36]) which owes its origin

to computations using finite elements. However, all these techniques require that the matrix can be ordered to obtain a narrow band or frontwidth. Duff [35] gives several instances of how dense techniques can be used in sparse factorizations including the then newly developed multifrontal techniques. The principal advantage of multifrontal techniques over a (uni)frontal approach is that they can be used in conjunction with any ordering scheme so that sparsity can be preserved.

A fundamental concept in sparse matrix factorization is an *elimination tree*. The elimination tree is defined for any sparse matrix whose sparsity pattern is symmetric. For a sparse matrix of order n, the elimination tree is a tree on n nodes such that node j is the father of node i if entry (i, j), j > i is the first entry below the diagonal in column i of the lower triangular factor. An analogous graph for an unsymmetric patterned sparse matrix is the directed acyclic graph [24,54].

Sparse Cholesky factorization by columns can be represented by an elimination tree. This can either be a left-looking (or fan-in) algorithm, where updates are performed on each column in turn by all the previous columns that contribute to it, then the pivot is chosen in that column and the multipliers calculated; or a right-looking (or fan-out) algorithm where, as soon as the pivot is selected and multipliers calculated, that column is immediately used to update all future columns that it modifies. The terms left-looking and right-looking are discussed in detail in the book [34]. Either way, the dependency of which columns update which columns is determined by the elimination tree. If each node of the tree is associated with a column, a column can only be modified by columns corresponding to nodes that are descendants of the corresponding node in the elimination tree.

One approach to using higher level BLAS in sparse direct solvers is a generalization of a sparse column factorization. Higher level BLAS can be used if columns with a common sparsity pattern are considered together as a single block or supernode and algorithms are termed column–supernode, supernode–column, and supernode–supernode depending on whether target, source, or both are supernodes (for example, [27]).

An alternative to the supernodal approach for utilizing Level 3 BLAS within a sparse direct code is a multifrontal technique [43]. In this approach, the nonzero entries of the pivot row and column are held in the first row and column of a dense array and the outer-product computation at that pivot step is computed within that dense submatrix. The dense submatrix is called a *frontal matrix*. Now, if a second pivot can be chosen from within this dense matrix (that is there are no nonzero entries in its row and column in the sparse matrix that lie outside this frontal matrix), then the operations for this pivot can again be performed within the frontal matrix. In order to facilitate this multiple elimination within a frontal matrix, an assembly tree is preferred to an elimination tree where, for example, chains of nodes are collapsed into a single node so that each node can represent several eliminations. Indeed sometimes we artificially enlarge the frontal matrices so that more pivots are chosen at each node and the Level 3 BLAS component is higher. Thus, the kernel of the multifrontal scheme can be represented by the computations

$$F_{11} = L_1 U_1 \tag{4.4}$$

and

$$F_{22}' \leftarrow F_{22} - F_{21} U_1^{-1} L_1^{-1} F_{12} \tag{4.5}$$

performed within the dense frontal matrix

$$\begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix}.$$

Table 2

Performance in Mflop/s of multifrontal code MA41 on matrix BCSSTK15, from the Rutherford-
Boeing Collection [40], on a single processor of a range of RISC processors. For comparison
the performance of DGEMM on a matrix of order 500 is given

Computer	Peak	DGEMM	MA41
DEC 3000/400 AXP	133	49	34
HP 715/64	128	55	30
IBM RS6000/750	125	101	64
IBM SP2 (Thin node)	266	213	122
MEIKO CS2-HA	100	43	31

The Schur complement,  $F'_{22}$  (4.5), is then sent to the parent node in the tree where it is summed with contributions from the original matrix and the other children to form another dense submatrix on which similar operations are performed at the father node. The effectiveness of this approach on RISC-based machines can be seen from the results in Table 2 where the code MA41 is a multifrontal code in the HSL [8]. Here the overall performance of the sparse code is always more than half that of the DGEMM kernel.

Several authors have experimented with these different algorithms (right-looking, left-looking, and multifrontal) and different blockings. Ng and Peyton [74] favour the left-looking approach and Amestoy and Duff [8] show the benefits of Level 3 BLAS within a multifrontal code on vector processors. Rothberg and Gupta [81] find that, on cache-based machines, it is the blocking that affects the efficiency (by a factor of 2–3) and the algorithm that is used has a much less significant effect. Demmel et al. [27] have extended the supernodal concept to unsymmetric systems although, for general unstructured matrices, they cannot use regular supernodes for the target columns and so they resort to Level 2.5 BLAS, which is defined as the multiplication of a set of vectors by a matrix where the vectors cannot be stored in a two-dimensional array. By doing this, the source supernode can be held in cache and applied to the target columns or blocks of columns of the "irregular" supernode, thus getting a high degree of reuse of data and a performance similar to the Level 3 BLAS.

It is very common to solve sparse least-squares problems by forming the normal equations

$$A^{\mathrm{T}}Ax = A^{\mathrm{T}}b \tag{4.6}$$

and to use a sparse solution scheme for symmetric positive-definite systems on these resulting equations. There are, however, other methods for solving the least-squares problem. The most robust uses a QR factorization of the coefficient matrix. This factorization can also be implemented as a multifrontal method and codes have been developed by [1,12,73].

#### 5. Parallel computation

In contrast to the situation with iterative methods where, in addition to vector operations, often only matrix-vector products are required, the kernel computations for sparse direct methods are far more complicated. Nevertheless, the benefits that can be obtained from successful parallelization can be much greater. Indeed, Duff and van der Vorst [44] claim that the ratio of this benefit is in proportion to 1:5:10 for iterative methods, direct sparse factorizations, and direct dense factorizations respectively. That is, we might expect gains five times as great due to the parallelization of a direct sparse code than an iterative one. The reason for this is similar to the reason why direct methods, when properly formulated, can be so efficient on RISC-based or vector machines. This is due to the kernel (as we discussed in the last three sections) being a dense matrix – dense matrix multiply. We saw the benefit of using Level 3 BLAS in sparse factorization for RISC-based machines in Section 4. It is also of benefit in a parallel setting to combine pivot steps and to work not with rows and columns but with block rows and columns. Clearly, the use of such block techniques and higher level BLAS allow us to obtain parallelism at the level of the elimination operations themselves.

There is also a very coarse level at which parallelism can be exploited. At this coarsest level, which is similar to the subdivision of a problem by domain decomposition, we use techniques for partitioning the matrix. These are often designed for parallel computing and are particularly appropriate for distributed memory computers. Indeed, partitioning methods are often only competitive when parallelism is considered. The PARASPAR package [88] uses a preordering to partition the original problem. The MCSPARSE package [47,53] similarly uses a coarse matrix decomposition to obtain an ordering to bordered block triangular form.

However, the main level of parallelism that we wish to discuss here is at a level intermediate between these two and is due to the sparsity of the matrix being factorized. Clearly, there can be substantial independence between pivot steps in sparse elimination. For example, if the matrix were a permutation of a diagonal matrix all operations could be performed in parallel. Two matrix entries  $a_{ij}$  and  $a_{rs}$  can be used as pivots simultaneously if  $a_{is}$  and  $a_{rj}$  are zero. These pivots are termed *compatible*. This observation [22] has been the basis for several algorithms and parallel codes for general matrices. When a pivot is chosen all rows with entries in the pivot column and all columns with entries in the pivot row are marked as ineligible and a subsequent pivot can only be chosen from the eligible rows and columns. In this way, a set of say k independent pivots is chosen. If the pivots were permuted to the beginning of the matrix, this  $k \times k$  pivot block would be diagonal. The resulting elimination operations are performed in parallel using a rank k update. The procedure is repeated on the reduced matrix. The algorithms differ in how the pivots are selected (clearly one must compromise criteria for reducing fill-in in order to get a large compatible pivot set) and in how the update is performed.

Alaghband [2] uses compatibility tables to assist in the pivot search. She uses a two-stage implementation where first pivots are chosen in parallel from the diagonal and then off-diagonal pivots are chosen sequentially for stability reasons. She sets thresholds for both sparsity and stability when choosing pivots. Davis and Yew [25] perform their pivot selection in parallel, which results in the nondeterministic nature of their algorithm because the compatible set will be determined by the order in which potential compatible pivots are found. Their algorithm, D2, was designed for shared-memory machines and was tested extensively on an Alliant FX/8.

The Y12M algorithm [89] extends the notion of compatible pivots by permitting the pivot block to be upper triangular rather than diagonal, which allows them to obtain a larger number of pivots, although the update is more complicated. For distributed memory architectures, van der Stappen et al. [85] distribute the matrix over the processors in a grid fashion, perform a parallel search for compatible pivots, choosing entries of low Markowitz cost that satisfy a pivot threshold, and perform a parallel rank-k update of the reduced matrix, where k is the number of compatible pivots chosen. They show high speedups (about 100 on 400 processors of a PARSYTEC SuperCluster FT-400)

although the slow processor speed masks the communication costs on this machine. Their code was originally written in OCCAM, but they have since developed a version using PVM [67].

In the context of reduced stability of the factorization due to the need to preserve sparsity and exploit parallelism, it is important that sparse codes offer the possibility of iterative refinement both to obtain a more accurate answer and to provide a measure of the backward error. Demmel and Li [69] try to avoid the dynamic data structures required by numerical pivoting by using the algorithm of Duff and Koster [41,42] to permute large entries to the diagonal prior to starting the factorization. They also suggest computing in increased precision to avoid some of the problems from this compromise to stability pivoting.

An important aspect of these approaches is that the parallelism is obtained directly because of the sparsity in the system. In general, we exhibit this form of parallelism through the assembly tree of Section 4 where operations at nodes which are not on the same (unique) path to the root (that is none is a descendant of another) are independent and can be executed in parallel (see, for example, [37,70]). The set of pivots discussed above could correspond to leaf nodes of such a tree. The tree can be used to schedule parallel tasks. For shared memory machines, this can be accomplished through a shared pool of work with fairly simple synchronizations that can be controlled using locks protecting critical sections of the code [6,37,38]. One of the main issues for an efficient implementation on shared memory machines concerns the management of data, which must be organized so that book-keeping operations such as garbage collection do not cause too much interference with the parallel processing.

A problem with the minimum degree ordering is that it tends to give elimination trees that are not well balanced and so not ideal for use as a computational graph for driving a parallel algorithm. The elimination tree can be massaged [71] so that it is more suitable for parallel computation but the effect of this is fairly limited for general matrices. The beauty of dissection orderings is that they take a global view of the problem; their difficulty until recently has been the problem of extending them to unstructured problems. Recently, there have been several tools and approaches that make this extension more realistic [76]. The essence of a dissection technique is a bisection algorithm that divides the graph of the matrix into two partitions. If node separators are used, a third set will correspond to the node separators. Recently, there has been much work on obtaining better bisections even for irregular graphs. Perhaps the bisection technique that has achieved the most fame has been spectral bisection [76]. In this approach, use is made of the Laplacian matrix that is defined as a symmetric matrix whose diagonal entries are the degrees of the nodes and whose off-diagonals are -1 if and only if the corresponding entry in the matrix is nonzero. This matrix is singular because its row sums are all zero but, if the matrix is irreducible, it is positive semidefinite with only one zero eigenvalue. Often the same software is used for the dissection orderings as for graph partitioning. Two of the major software efforts in this area are CHACO [62] and METIS [66].

A currently favoured approach is for the dissection technique to be used only for the top levels and the resulting subgraphs to be ordered by a minimum degree scheme. This hybrid technique was described some time ago [52] but was discredited because of the poor performance of nested dissection techniques on irregular graphs at that time. However, because of the much better implementations of dissection orderings as discussed above, this hybrid technique is included in many current implementations (for example, [17,63]).

Current empirical evidence would suggest that these schemes are at least competitive with minimum degree on some large problems from structural analysis [17,79] and from financial modelling

Ta	bl	e	3

Effect of ordering on Problem BMWCRA\_1 from the PARASOL test set. Matrix of order 148,770 with 5,396,386 entries in the lower triangle. Elapsed times in seconds on an ORIGIN 2000. Speedups in parentheses

		Analysis phase entries in factors		Operations
AMD HYBRID		$\begin{array}{c} 1.13 \times 10^8 \\ 8.53 \times 10^7 \end{array}$		$\begin{array}{c} 1.28 \times 10^{11} \\ 6.72 \times 10^{10} \end{array}$
	Factorization		Solve	
No. procs	AMD	HYBRID	AMD	HYBRID
1	687	307	12.0	10.1
2	408(1.7)	178(1.7)	7.5(1.6)	5.4(1.9)
4	236(2.9)	82(3.7)	6.7(1.8)	4.2(2.4)
8	143(4.8)	58(5.3)	4.2(2.9)	2.6(3.9)
16	112(6.1)	36(8.5)	2.9(4.1)	1.9(5.3)

[18]. In these studies, dissection techniques outperform minimum degree by on average about 15% in terms of floating-point operations for Cholesky factorization using the resulting ordering, although the cost of these orderings can be several times that of minimum degree and may be a significant proportion of the total solution time [17]. We show, in Table 3 the effect of the hybrid ordering within the MUMPS code (see Section 6) on one of the PARASOL test examples. AMD is an ordering produced by the approximate minimum degree code of Amestoy et al. [7], and HYBRID is an ordering from METIS that combines nested dissection and minimum degree. We see that the gains from the HYBRID ordering are even more dramatic than those mentioned above with about half the number of operations required for factorization with the HYBRID ordering than with AMD. We also note, from the results in Table 3, that the parallelism is better for the HYBRID ordering.

In recent years, the performance of sparse direct codes has increased considerably. The improvement is not from the approach used (fan-in, fan-out, multifrontal) but rather because of the use of blocking techniques and two-dimensional mappings. The benefit of using higher level BLAS kernels, coupled with increases in local memory and the communication speed of parallel processors, have at last made the solution of large sparse systems feasible on such architectures. We now review some of the recent performance figures from several different implementations. Dumitrescu et al. [45] record a performance of over 360 Mflop/s on 32 nodes of an IBM SP1 using a two-dimensional block fan-in algorithm. Rothberg [80] has implemented a block fan-out algorithm using two-dimensional blocking and obtains a performance of over 1.7 Gflop/s on 128 nodes of an Intel Paragon, which is about 40% of the performance of the \_GEMM kernel on that machine. A 2-D block fan-out algorithm has been coupled with some block mapping heuristics to obtain a performance of over 3 Gflop/s for a 3-D grid problem on a 196-node Intel Paragon [82]. A similar type of two-dimensional mapping is used [56] in an implementation of a multifrontal method, where much of the high performance is obtained through balancing the tree near its root and using a highly tuned mapping of the dense matrices near the root to allow a high level of parallelism to be maintained. Although the headline figure of nearly 20 Gflop/s on the CRAY T3D was obtained on a fairly artificial and essentially dense problem, large sparse problems from structural analysis were factorized at between 8 and 15 Gflop/s on the same machine for which a tuned \_GEMM code will execute at around 50 Gflop/s. This code is available in compiled form on an IBM SP2 [55] and source code versions of a portable implementation are available from the authors [57]. More recently, Li and Demmel [69] have been developing a version of the SuperLU code [26] for distributed memory machines and the MUMPS multifrontal code [10], developed within the EU PARASOL Project, also targets message passing architectures.

Partly because of the success of fast and parallel methods for performing the numerical factorization, other phases of the solution are now becoming more critical on parallel computers. The package [61] executes all phases in parallel, and there has been much recent work in finding parallel methods for performing the reordering. This has been another reason for the growth in dissection approaches (for example, see [65,78]). Parallelism in the triangular solve can be obtained either using the identical tree to the numerical factorization [12] or by generating a tree from the sparsity pattern of the triangular factor [15]. However, in order to avoid the intrinsically sequential nature of a sparse triangular solve, it is possible to hold the denser but still sparse  $L^{-1}$ , or better a partitioned form of this to avoid some of the fill-in that would be associated with forming  $L^{-1}$  explicitly [5]. Various schemes for this partitioning have been proposed to balance the parallelism (limited by the number of partitions) with the fill-in (for example, [3,4,75]) and, more recently, the selective inversion of submatrices produced by a multifrontal factorization algorithm has been proposed [77].

#### 6. Current situation

There is no question that direct sparse matrix algorithms and codes based on them have "come of age". Gone are the days when the only sparse codes that were generally available could be found in the HSL. We have already remarked on the PSPASES code for symmetric positive-definite systems by Gupta and others [55,57] and the SuperLU code for general unsymmetric sparse systems by Demmel and Li [27,69]. Both these projects have developed code for distributed memory computers.

The MUMPS code [9,11] implements a parallel multifrontal technique for distributed memory computers and is part of the EU PARASOL Project<sup>1</sup> whose goal was to build and test a portable library for solving large sparse systems of equations on distributed memory systems. The PARA-SOL software is written in Fortran 90, uses MPI for message passing, and is available from teh Web page http://www.pallas.de/parasol/. The solvers developed in this Project are two domain decomposition codes by Bergen and ONERA, a multigrid code by GMD, and the MUMPS code.

Dobrian et al. [28] have studied the use of an object-oriented approach to design sparse direct solvers and O-O is used by Ashcraft in his SPOOLES package [16]. Yang and his co-workers have developed a sparse direct package  $S^*$  for distributed memory computers [46] and there are a number of commercial offerings that can be found through Web searches.

<sup>&</sup>lt;sup>1</sup>For more information on the PARASOL project, see the web site at http://www.genias.de/projects/parasol/index.html.

#### 7. Future trends

There seems a never ending demand for the solution of larger and larger problems. For example, some problems from the ASCI program in the United States have dimensions of several million and animal breeders are now solving systems of 20–30 million degrees of freedom.

Clearly, the size of problem that can be solved by a direct method is very dependent on the matrix structure. For example, a diagonal or tridiagonal system pose no problems when the dimension increases and indeed, if fill-in can be kept low, it is usually possible to solve very large problems by sparse direct factorization. However, for the discretization of three-dimensional partial differential equations, the limitations of direct methods become all too apparent. Although problems from finite-element discretizations of order nearly one million have been solved by MUMPS [11], in my opinion, the most promising set of techniques for the solution of really large problems are those that combine both direct and iterative methods. This can be viewed as a sophisticated preconditioning for an iterative method and is discussed in greater length in the article by Saad and van der Vorst [83].

One of the most promising techniques uses graph partitioning to subdivide the problem, solves the local subproblems by direct methods and uses an iterative method to couple the blocks in the partitioning. This approach is very similar to methods used in the solution of problems from discretizations of PDEs using domain decomposition which can be viewed as permuting the matrix to bordered block diagonal form. However, additional preconditioners are used both for the Schur complement and also a coarse preconditioner for the overall problem. A good discussion of these preconditioners can be found in the thesis of Luiz Carvalho [23].

It is interesting to surmise what the trends will be. Personally, I feel that languages like Fortran 90 combine sufficient object orientation with clarity and efficiency although there is certainly an increasing population who find the lure of novel object-oriented languages irresistible. Old techniques continue to be rediscovered as in the revamping of interpretative code approaches, originally developed in the 1970s [60], by Grund [20] who has had some success in solving problems from chemical engineering. The exploitation of sparsity on the right-hand side, for some time pursued in the context of electronic circuit applications and power systems, is now becoming a very powerful tool in the rehabilitation of simplex methods for linear programming.

#### Acknowledgements

I would like to thank my colleagues Patrick Amestoy, Jacko Koster, Xiaoye Li, John Reid, and Jennifer Scott for some helpful remarks on a draft of this paper.

#### References

- M. Adlers, Computing sparse orthogonal factors in MATLAB, Technical Report LiTH-MAT-R-1998-19, Linköping University, 1999.
- [2] G. Alaghband, Parallel sparse matrix solution and performance, Parallel Comput. 21 (9) (1995) 1407-1430.
- [3] F.L. Alvarado, A. Pothen, R. Schreiber, Highly parallel sparse triangular solution, in: A. George, J.R. Gilbert, J.W.H. Liu (Eds.), Graph Theory and Sparse Matrix Computation, Springer, Berlin, 1993.

- [4] F.L. Alvarado, R. Schreiber, Optimal parallel solution of sparse triangular systems, SIAM J. Sci. Comput. 14 (1993) 446–460.
- [5] F.L. Alvarado, D.C. Yu, R. Betancourt, Partitioned sparse  $A^{-1}$  methods, IEEE Trans. Power Systems 3 (1990) 452–459.
- [6] P.R. Amestoy, Factorization of large sparse matrices based on a multifrontal approach in a multiprocessor environment, INPT Ph.D. Thesis TH/PA/91/2, CERFACS, Toulouse, France, 1991.
- [7] P.R. Amestoy, T.A. Davis, I.S. Duff, An approximate minimum degree ordering algorithm, SIAM J. Matrix Anal. Appl. 17 (4) (1996) 886–905.
- [8] P.R. Amestoy, I.S. Duff, Vectorization of a multiprocessor multifrontal code, Internat. J. SuperComput. Appl. 3 (1989) 41–59.
- [9] P.R. Amestoy, I.S. Duff, J.-Y. L'Excellent, Multifrontal parallel distributed symmetric and unsymmetric solvers, Comput. Methods Appl. Mech. Eng. 184 (2000) 501–520.
- [10] P.R. Amestoy, I.S. Duff, J.-Y. L'Excellent, Multifrontal solvers within the PARASOL environment, in: B. Kagstrom, J. Dongarra, E. Elmroth, J. Wasniewski (Eds.), Applied Parallel Computing, PARA'98, Lecture Notes in Computer Science, Vol. 1541, Springer, Berlin, 1998, pp. 7–11.
- [11] P.R. Amestoy, I.S. Duff, J.-Y. L'Excellent, J. Koster, A fully asynchronous multifrontal solver using distributed dynamic scheduling, Technical Report RAL-TR-1999-059, Rutherford Appleton Laboratory, 1999.
- [12] P.R. Amestoy, I.S. Duff, C. Puglisi, Multifrontal QR factorization in a multiprocessor environment, Numer. Linear Algebra Appl. 3 (4) (1996) 275–300.
- [13] B.S. Andersen, F. Gustavson, A. Karaivanov, J. Wasniewski, P.Y. Yalamov, LAWRA Linear Algebra With Recursive Algorithms, Technical Report UNIC-99-01, UNI-C, Lyngby, Denmark, 1999.
- [14] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, D. Sorensen, LAPACK Users' Guide, 2nd Edition, SIAM Press, Philadelphia, PA, 1995.
- [15] E.C. Anderson, Y. Saad, Solving sparse triangular systems on parallel computers, Internat. J. High Speed Comput. 1 (1989) 73–95.
- [16] C. Ashcraft, R. Grimes, SPOOLES: An object-oriented sparse matrix library. Proceedings of the Ninth SIAM Conference on Parallel Processing, 1999. See http://www.netlib.org/linalg/spooles.
- [17] C. Ashcraft, J.W.H. Liu, Robust ordering of sparse matrices using multisection, SIAM J. Matrix Anal. Appl. 19 (1998) 816–832.
- [18] A. Berger, J. Mulvey, E. Rothberg, R. Vanderbei, Solving multistage stochastic programs using tree dissection, Technical Report SOR-97-07, Programs in Statistics and Operations Research, Princeton University, Princeton, NJ, 1995.
- [19] L.S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, R.C. Whaley, ScaLAPACK Users' Guide, SIAM Press, Philadelphia, PA, 1997.
- [20] J. Borchardt, F. Grund, D. Horn, Parallel numerical methods for large systems of differential-algebraic equations in industrial applications, Technical Report 382, Weierstraß-Institut f
  ür Angewandte Analysis und Stochastik, Berlin, 1997.
- [21] J.R. Bunch, L. Kaufman, B.N. Parlett, Decomposition of a symmetric matrix, Numer. Math. 27 (1976) 95-110.
- [22] D.A. Calahan, Parallel solution of sparse simultaneous linear equations, Proceedings 11th Annual Allerton Conference on Circuits and System Theory, University of Illinois, 1973, pp. 729–735.
- [23] L.M. Carvalho, Preconditioned Schur complement methods in distributed memory environments, INPT Ph.D. Thesis TH/PA/97/41, CERFACS, Toulouse, France, 1997.
- [24] T.A. Davis, I.S. Duff, An unsymmetric-pattern multifrontal method for sparse LU factorization, SIAM J. Matrix Anal. Appl. 18 (1) (1997) 140–158.
- [25] T.A. Davis, P.C. Yew, A nondeterministic parallel algorithm for general unsymmetric sparse LU factorization, SIAM J. Matrix Anal. Appl. 11 (1990) 383–402.
- [26] J.W. Demmel, J.R. Gilbert, X.S. Li, SuperLU users' guide, Technical Report, Computer Science Division, U.C. Berkeley, Berkeley, California, February 1995. (available from netlib).
- [27] J.W. Demmel, S.C. Eisenstat, J.R. Gilbert, X.S. Li, W.H. Liu, A supernodal approach to sparse partial pivoting, SIAM J. Matrix Anal. Appl. 20 (1999) 720–755.

- [28] F. Dobrian, G. Kumfert, A. Pothen, Object-oriented design for sparse direct solvers, Technical Report 99-2, Institute for Computer Applications in Science and Engineering (ICASE), MS 132C, NASA Langley Research Center, Hampton, VA 23681-0001, USA, 1999.
- [29] J.J. Dongarra, Performance of various computers using standard linear algebra software, Technical Report CS-89-85, University of Tennessee, Knoxville, Tennessee, 1999. Updated version at Web address http://www.netlib.org/benchmark/performance.ps.
- [30] J.J. Dongarra, J.R. Bunch, C.B. Moler, G.W. Stewart, LINPACK User's Guide, SIAM Press, Philadelphia, PA, 1979.
- [31] J.J. Dongarra, J.J. Du Croz, S. Hammarling, R.J. Hanson, An extented set of Fortran Basic Linear Algebra Subprograms, ACM Trans. Math. Software 14 (1988) 1–17.
- [32] J.J. Dongarra, V. Eijkhout, Numerical linear algebra algorithms and software, this issue, J. Comput. Appl. Math. 123 (2000) 489–514.
- [33] J.J. Dongarra, J. Du Croz, I.S. Duff, S. Hammarling, A set of Level 3 basic linear algebra subprograms, ACM Trans. Math. Software 16 (1990) 1–17.
- [34] J.J. Dongarra, I.S. Duff, D.C. Sorensen, H.A. van der Vorst, Numerical Linear Algebra for High-Performance Computers, SIAM Press, Philadelphia, PA, 1998.
- [35] I.S. Duff, Full matrix techniques in sparse Gaussian elimination, in: G.A. Watson (Ed.), Numerical Analysis Proceedings, Dundee 1981, Lecture Notes in Mathematics, Vol. 912, Springer, Berlin, 1981, pp. 71–84.
- [36] I.S. Duff, Design features of a frontal code for solving sparse unsymmetric linear systems out-of-core, SIAM J. Sci. Statist. Comput. 5 (1984) 270–280.
- [37] I.S. Duff, Parallel implementation of multifrontal schemes, Parallel Comput. 3 (1986) 193–204.
- [38] I.S. Duff, Multiprocessing a sparse matrix code on the Alliant FX/8, J. Comput. Appl. Math. 27 (1989) 229-239.
- [39] I.S. Duff, A.M. Erisman, J.K. Reid, Direct Methods for Sparse Matrices, Oxford University Press, Oxford, England, 1986.
- [40] I.S. Duff, R.G. Grimes, J.G. Lewis, The Rutherford–Boeing sparse matrix collection, Technical Report RAL-TR-97-031, Rutherford Appleton Laboratory, 1997. Also Technical Report ISSTECH-97-017 from Boeing Information & Support Services, Seattle and Report TR/PA/97/36 from CERFACS, Toulouse.
- [41] I.S. Duff, J. Koster, The design and use of algorithms for permuting large entries to the diagonal of sparse matrices, SIAM J. Matrix Anal. Appl. 20 (4) (1999) 889–901.
- [42] I.S. Duff, J. Koster, On algorithms for permuting large entries to the diagonal of a sparse matrix, Technical Report RAL-TR-1999-030, Rutherford Appleton Laboratory, 1999. Also appeared as Report TR/PA/99/13, CERFACS, Toulouse, France.
- [43] I.S. Duff, J.K. Reid, The multifrontal solution of indefinite sparse symmetric linear systems, ACM Trans. Math. Software 9 (1983) 302–325.
- [44] I.S. Duff, H.A. van der Vorst, Developments and trends in the parallel solution of linear systems, Parallel Comput. 25 (1999) 1931–1970.
- [45] B. Dumitrescu, M. Doreille, J.-L. Roch, D. Trystram, Two-dimensional block partitionings for the parallel sparse Cholesky factorization, Numer. Algorithms 16 (1) (1997) 17–38.
- [46] C. Fu, X. Jiao, T. Yang, Efficient sparse LU factorization with partial pivoting on distributed memory architectures, Technical report, University of California at Santa Barbara, 1997; IEEE Trans. Parallel Distrib. Systems, submitted for publication.
- [47] K.A. Gallivan, B.A. Marsolf, H.A.G. Wijshoff, Solving large nonsymmetric sparse linear systems using MCSPARSE, Parallel Comput. 22 (1996) 1291–1333.
- [48] A. George, Nested dissection of a regular finite element mesh, SIAM J. Numer. Anal. 10 (1973) 345-363.
- [49] A. George, J.W.H. Liu, The design of a user interface for a sparse matrix package, ACM Trans. Math. Software 5 (2) (1979) 139–162.
- [50] A. George, J.W.H. Liu, Computer Solution of Large Sparse Positive Definite Systems, Prentice-Hall, Englewood Cliffs, NJ, 1981.
- [51] A. George, J.W.H. Liu, E.G. Ng, User's guide for SPARSPAK: Waterloo sparse linear equations package, Technical Report CS-78-30 (Revised), University of Waterloo, Canada, 1980.
- [52] A. George, J.W. Poole, R. Voigt, Incomplete nested dissection for solving  $n \times n$  grid problems, SIAM J. Numer. Anal. 15 (1978) 663–673.

- [53] J.P. Geschiere, H.A.G. Wijshoff, Exploiting large grain parallelism in a sparse direct linear system solver, Parallel Comput. 21 (8) (1995) 1339–1364.
- [54] J.R. Gilbert, J. W.H. Liu, Elimination structures for unsymmetric sparse LU factors, SIAM J. Matrix Anal. Appl. 14 (1993) 334–354.
- [55] A. Gupta, M. Joshi, V. Kumar, WSSMP: Watson Symmetric Sparse Matrix Package. Users Manual: Version 2.0β, Technical Report RC 20923 (92669), IBM T. J. Watson Research Centre, P.O. Box 218, Yorktown Heights, NY 10598, July 1997.
- [56] A. Gupta, G. Karypis, V. Kumar, Highly scalable parallel algorithms for sparse matrix factorization, Technical Report TR-94-63, Department of Computer Science, University of Minnesota, 1994.
- [57] A. Gupta, G. Karypis, V. Kumar, Highly scalable parallel algorithms for sparse matrix factorization, IEEE Trans. Parallel Distrib. Systems 8 (1997) 502–520.
- [58] F.G. Gustavson, Recursion leads to automatic variable blocking for dense linear algebra algorithms, IBM J. Res. Develop. 41 (1997) 737–755.
- [59] F. Gustavson, A. Henriksson, I. Jonsson, B. Kågström, P. Ling, Recursive blocked data formats and BLASs for dense linear algebra algorithms, in: B. Kågstrom, J. Dongarra, E. Elmroth, J. Wasniewski (Eds.), Applied Parallel Computing, Fourth International Workshop, PARA'98, Springer, Berlin, 1998, pp. 195–206.
- [60] F.G. Gustavson, W.M. Liniger, R.A. Willoughby, Symbolic generation of an optimal Crout algorithm for sparse systems of linear equations, J. ACM 17 (1) (1970) 87–109.
- [61] M.T. Heath, P. Raghavan, Performance of a fully parallel sparse solver, Proceedings of SHPCC '94, Scalable High-Performance Computing Conference, May 23–25, 1994, Knoxville, Tennessee, Los Alamitos, California, IEEE Computer Society Press, Silver Spring, MD, 1994, pp. 334–341.
- [62] B. Hendrickson, R. Leland, The CHACO User's Guide, Version 2.0, Technical Report SAND94-2692, Sandia National Laboratories, Albuquerque, October 1994.
- [63] B. Hendrickson, E. Rothberg, Improving the runtime and quality of nested dissection ordering, SIAM J. Sci. Comput. 20 (1998) 468–489.
- [64] HSL, Harwell Subroutine Library, A Catalogue of Subroutines (Release 12). AEA Technology, Harwell Laboratory, Oxfordshire, England, 1996. For information concerning HSL contact: Dr Nick Brealey, HSL Manager, AEA Technology Products & Systems, Culham Science Centre, Oxon OX14 3ED, England (tel: +44-1235-463404, fax: +44-1235-463480, email: hsl@aeat.co.uk).
- [65] G. Karypis, V. Kumar, Parallel multilevel graph partitioning, Technical Report TR-95-036, Department of Computer Science, University of Minnesota, May 1995.
- [66] G. Karypis, V. Kumar, METIS: Unstructured graph partitioning and sparse matrix ordering system, Technical Report TR 97-061, Department of Computer Science, University of Minnesota, Minnesota, 1997.
- [67] J. Koster, On the parallel solution and the reordering of unsymmetric sparse linear systems. INPT Ph.D. Thesis TH/PA/97/51, CERFACS, Toulouse, France, 1997.
- [68] C.L. Lawson, R.J. Hanson, D.R. Kincaid, F.T. Krogh, Basic linear algebra subprograms for Fortran usage, ACM Trans. Math. Software 5 (1979) 308–323.
- [69] X.S. Li, J.W. Demmel, Making sparse Gaussian elimination scalable by static pivoting. Proceedings of Supercomputing, Orlando, Florida, November 1998.
- [70] J.W.H. Liu, On the storage requirement in the out-of-core multifrontal method for sparse factorization, ACM Trans. Math. Software 12 (1987) 249–264.
- [71] J.W.H. Liu, Reordering sparse matrices for parallel elimination, Parallel Comput. 11 (1989) 73-91.
- [72] H.M. Markowitz, The elimination form of the inverse and its application to linear programming, Management Sci. 3 (1957) 255–269.
- [73] P. Matstoms, Parallel sparse QR factorization on shared memory architectures, Parallel Comput. 21 (1995) 473-486.
- [74] E.G. Ng, B.W. Peyton, Block sparse Cholesky algorithms on advanced uniprocessor computers, SIAM J. Sci. Comput. 14 (1993) 1034–1056.
- [75] B.W. Peyton, A. Pothen, X. Yuan, Partitioning a chordal graph into transitive subgraphs for parallel sparse triangular solution, Technical Report ORNL/TM-12270, Engineering Physics and Mathematics Division, Oak Ridge National Laboratory, Tennessee, December 1992.
- [76] A. Pothen, H.D. Simon, K.P. Liou, Partitioning sparse matrices with eigenvectors of graphs, SIAM J. Matrix Anal. Appl. 11 (3) (1990) 430–452.

- [77] P. Raghavan, Efficient parallel sparse triangular solution using selective inversion, Technical Report CS-95-314, Department of Computer Science, University of Tennessee, Knoxville, Tennessee, 1995.
- [78] P. Raghavan, Parallel ordering using edge contraction, Parallel Comput. 23 (8) (1997) 1045–1067.
- [79] E. Rothberg, Exploring the tradeoff between imbalance and separator size in nested dissection ordering, Technical Report Unnumbered, Silicon Graphics Inc, 1996.
- [80] E. Rothberg, Performance of panel and block approaches to sparse Cholesky factorization on the iPSC/860 and Paragon multicomputers, SIAM J. Sci. Comput. 17 (3) (1996) 699–713.
- [81] E. Rothberg, A. Gupta. An evaluation of left-looking, right-looking and multifrontal approaches to sparse Cholesky factorization on hierarchical-memory machines, Technical Report STAN-CS-91-1377, Department of Computer Science, Stanford University, 1991.
- [82] E. Rothberg, R. Schreiber, Improved load distribution in parallel sparse Cholesky factorization, Technical Report 94-13, Research Institute for Advanced Computer Science, 1994.
- [83] Y. Saad, H.A. van der Vorst, Iterative solution of linear systems in the 20th century, this issue, J. Comput. Appl. Math. 123 (2000) 1–33.
- [84] W.F. Tinney, J.W. Walker, Direct solutions of sparse network equations by optimally ordered triangular factorization, Proc. IEEE 55 (1967) 1801–1809.
- [85] A.F. van der Stappen, R.H. Bisseling, J.G.G. van de Vorst, Parallel sparse LIU decomposition on a mesh network of transputers, SIAM J. Matrix Anal. Appl. 14 (1993) 853–879.
- [86] R.C. Whaley, LAPACK Working Note 73: Basic Linear Algebra Communication Subprograms: analysis and implementation across multiple parallel architectures, Technical Report CS-94-234, Computer Science Department, University of Tennessee, Knoxville, Tennessee, May 1994.
- [87] J.H. Wilkinson, C. Reinsch, Handbook for Automatic Computation, Volume II Linear Algebra, Springer, Berlin, 1971.
- [88] Z. Zlatev, J. Waśniewski, P.C. Hansen, Tz. Ostromsky, PARASPAR: a package for the solution of large linear algebraic equations on parallel computers with shared memory, Technical Report 95-10, Technical University of Denmark, Lyngby, 1995.
- [89] Z. Zlatev, J. Waśniewski, K. Schaumburg. Introduction to PARASPAR, Solution of large and sparse systems of linear algebraic equations, specialised for parallel computers with shared memory, Technical Report 93-02, Technical University of Denmark, Lyngby, 1993.



JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.nl/locate/cam

### Journal of Computational and Applied Mathematics 123 (2000) 531

## Author Index Volume 123 (2000)

matrix and polynomial approaches to Lanczos-	
type algorithms	241-260
Bridson, R. and WP. Tang, Refining an approximate	
inverse	293-306
Calvetti, D. and L. Reichel, Iterative methods for	
large continuation problems	217-240
Calvetti, D., S. Morigi, L. Reichel and F. Sgallari, Tikhonov regularization and the L-curve for large	
discrete ill-posed problems	423-446
<b>Chan, T.F.</b> and <b>W.L. Wan,</b> Robust multigrid methods	125 110
for nonsmooth coefficient elliptic linear systems	323-352
Dongarra, J.J. and V. Eijkhout, Numerical linear	
algebra algorithms and software	489–514
Duff, I.S., The impact of high-performance comput-	
ing in the solution of linear systems: trends and	
problems	515-530
Eiermann, M., O.G. Ernst and O. Schneider, Analysis	
of acceleration strategies for restarted minimal	
residual methods	261-292
Eijkhout, V., see Dongarra, J.J.	489-514
Ernst, O.G., see Eiermann, M.	261-292
Freund, R.W., Krylov-subspace methods for reduced-	
order modeling in circuit simulation	395-421
Frommer, A. and D.B. Szyld, On asynchronous iter-	
ations	201-216
Golub, G.H. and H.A. van der Vorst, Eigenvalue	
computation in the 20th century	35- 65
Hadjidimos, A., Successive overrelaxation (SOR) and	
related methods	177-199
<b>Insen. I.C.F</b> An overview of relative $\sin \Theta$ theorems	
for invariant subspaces of complex matrices	131-153
Koning, J., G. Rodrigue and D. White, Scalable pre-	
conditioned conjugate gradient inversion of vec-	
tor finite element mass matrices	307-321
Mehrmann, V. and H. Xu, Numerical methods in	
control	371-394
Morgan, R.B., Preconditioning eigenvalues and some	
comparison of solvers	101-115

Brezinski, C., M. Redivo-Zaglia and H. Sadok, The

Morigi, S., see Calvetti, D.	423-446
Neal, L., see Poole, G.	353-369
O'Leary, D.P., Symbiosis between linear algebra and optimization	447-465
<ul> <li>Parlett, B.N., For tridiagonals T replace T with LDL<sup>1</sup></li> <li>Pauca, V.P., see Plemmons, R.J.</li> <li>Plemmons, R.J. and V.P. Pauca, Some computational problems arising in adaptive optics imaging systems</li> <li>Poole, G. and L. Neal, The Rook's pivoting strategy</li> </ul>	117-130 467-487 467-487 353-369
Redivo-Zaglia, M., see Brezinski, C. Reichel, L., see Calvetti, D. Reichel, L., see Calvetti, D. Rodrigue, G., see Koning, J.	241-260 217-240 423-446 307-321
<ul> <li>Saad, Y. and H.A. van der Vorst, Iterative solution of linear systems in the 20th century</li> <li>Sadok, H., see Brezinski, C.</li> <li>Sameh, A. and Z. Tong, The trace minimization method for the symmetric generalized eigenvalue problem</li> <li>Schneider, O., see Eiermann, M.</li> <li>Sgallari, F., see Calvetti, D.</li> <li>Szyld, D.B., see Frommer, A.</li> </ul>	1- 33 241-260 155-175 261-292 423-446 201-216
Tang, WP., see Bridson, R. Tong, Z., see Sameh, A.	293–306 155–175
van der Vorst, H.A., see Saad, Y. van der Vorst, H.A., see Golub, G.H. van Loan, C.F., The ubiquitous Kronecker product	1- 33 35- 65 85-100
<ul><li>Wan, W.L., see Chan, T.F.</li><li>Watkins, D.S., <i>QR</i>-like algorithms for eigenvalue problems</li><li>White, D., see Koning, J.</li></ul>	323–352 67– 83 307–321
Xu, H., see Mehrmann, V.	371-394

0377-0427/00/\$ - see front matter  $\odot$  2000 Elsevier Science B.V. All rights reserved. PII: S 0 3 7 7 - 0 4 2 7 ( 0 0 ) 0 0 5 6 8 - 9