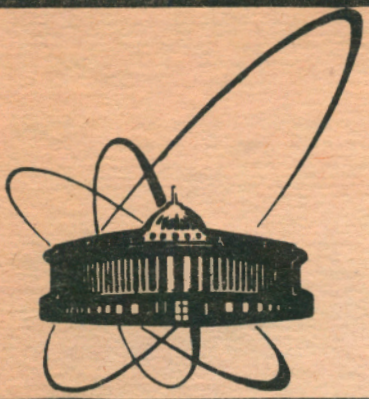

сообщения
объединенного
института
ядерных
исследований
дубна



E11-90-391

E.G.Nadjakov

# MATRIX DIAGONALISATION ON PERSONAL COMPUTERS

1990

1. Algorithm

There is a great variety of algorithms and programs for solving this problem due to its importance in physics, and in particular in quantum many body problems. As much as the algorithms are concerned, a detailed survey may be found in [1], together with instructions for programs constructions. As well known, a high order matrix is a rather complicated object, and may present various intricated special cases. Therefore so many algorithms have been developed, in order to be able to deal with different matrix types and with their intricated cases. The programs based on them have been designed mostly for big computers.

Our aim has been to suggest a method, universal enough to be able to treat different matrix types together with their complications. At the same time it should be able to compete in reasonable limits with big computer methods, with respect to high execution speed and matrix order. And its program should be reasonably concise to fit into a personal computer with a 640 and even a 256 kb memory. The idea to achieve this has been not to apply complicated methods devoted to knotty special cases. It has been to choose a variety of most reliable methods, with the possibility to change them during performance. Thus a universal method is found. It allows the user to make a compromise between accuracy and speed during execution, depending on the results.

If one looks at the known algorithms, one can see that they include different initial matrix types, different matrix form transformations to increase the speed, different iteration procedures to find the eigenvalues, procedures to find the eigenstates, and different procedures to decompose a matrix to find its determinant and inverse matrix and/or its eigenvalues and eigenstates. As much as the matrix types are concerned, we have included any real matrices,

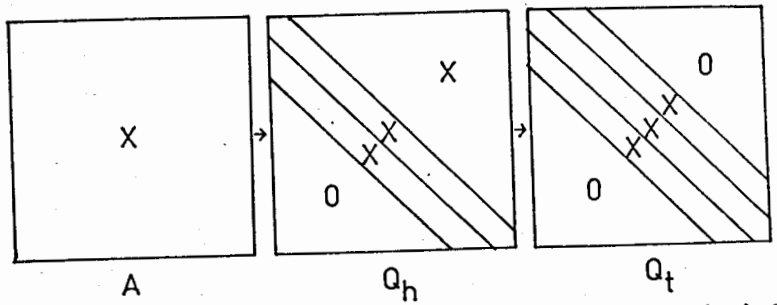


Figure 1. Original (A), hessenberg ($Q_h$) and tridiagonal ($Q_t$) forms of a matrix, the second and third one obtained by known successive transformations [1].

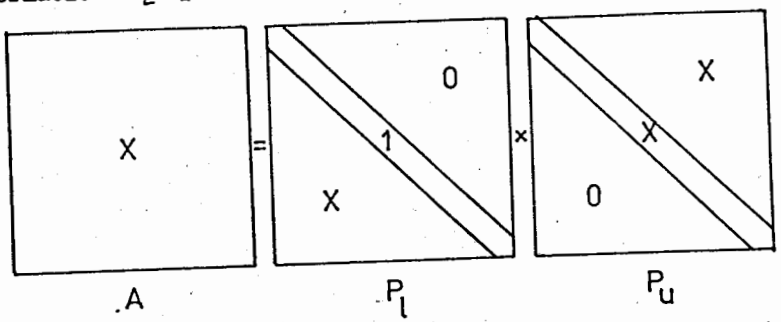


Figure 2. Decomposition of a matrix (A or Q) into a product $A=P_l \times P_u$ of a lower ($P_l$) and upper ($P_u$) triangular matrices. The determinant D of A is equal to the product of the diagonal matrix elements of $P_u$. The inverse matrix $A^{-1} = P_u^{-1} \times P_l^{-1}$ is obtained by a simple procedure [2].

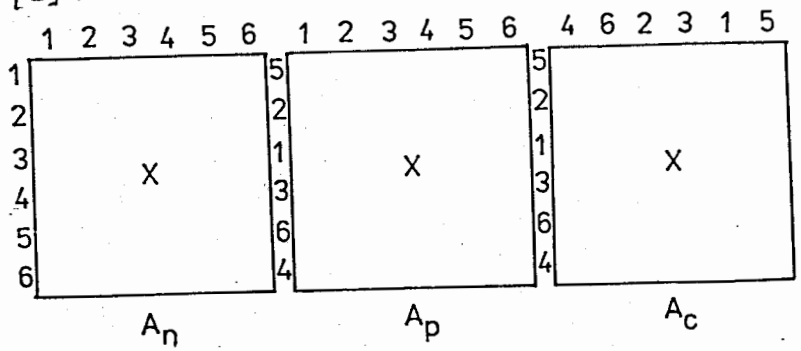


**Figure 3.** No ($A_n$), partial ($A_p$) and complete ($A_c$) pivoting of a matrix (A or Q) used to make possible and/or numerically stable and accurate the transformation of figure 1 and the decomposition of figure 2 [1,2].



independent of symmetry (hermiticity). As much as matrix form transformations are concerned, we have combined the possibility to apply 4 forms: original, hessenberg, tridiagonal accepted and another tridiagonal modified (figure 1); the last one is good for low order matrices only, but worse for high order ones. Thus we increase the execution speed very much, especially for high order matrices, since the theoretical execution times increase with matrix order N roughly as $N^4$ for original, $N^3$ for hessenberg and $N^2$ for tridiagonal forms.

For finding the eigenvalues we have also used the $A - E = P_l \times P_u$ decomposition (figure 2), which is the foundation of the LR and its modification QR methods [1]. As well known, this decomposition does not apply to some special matrices and is not numerically stable for many others [2], if pivoting is not applied (figure 3). No pivoting means leaving the matrix form as it is, partial pivoting: interchanging its rows only, complete pivoting: interchanging its rows and columns as well, in an optimal for form transformation and matrix decomposition way [1,2]. Then we have made use of an iteration procedure based on the bisection method (figure 4). This method is most reliable as much as accuracy is concerned.

For finding the eigenstates after the steps of figures 1,2,3,4 have been completed, we have applied a method explained in figure 5. It allows one to find the multiplicity of a degenerate eigenvalue, making special use of the complete pivoting in figure 3 during the last step in figure 4. Generally this is a difficult problem [1]. Thereafter we find all the eigenstates belonging to this eigenvalue (figure 5) and orthonomalise them by a Gram - Schmidt procedure.

Thus our algorithm contains a combination of methods of the two main classes [1]: form transformations and LR, QR methods.

## 2. Program

The program has been realised in turbo basic language. The choice

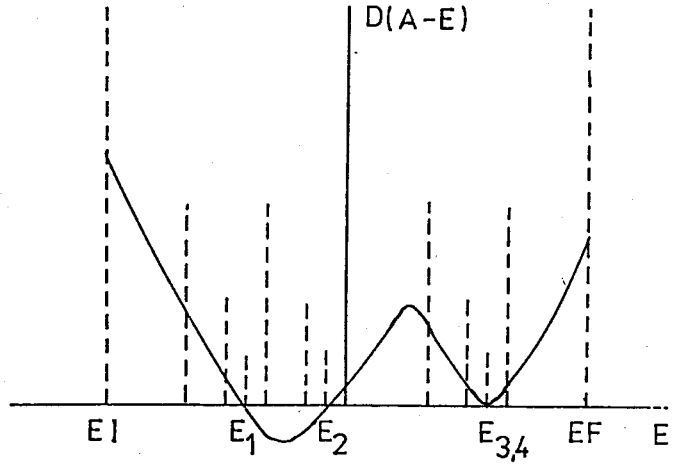


**Figure 4.** Iteration procedure for finding the eigenvalues of a matrix A by the zeros $E_i$ of the determinant D=D(A-E) of A-E after the transformation of figure 1 and the decomposition of figure 2 with pivoting of figure 3 by a bisection method [1].

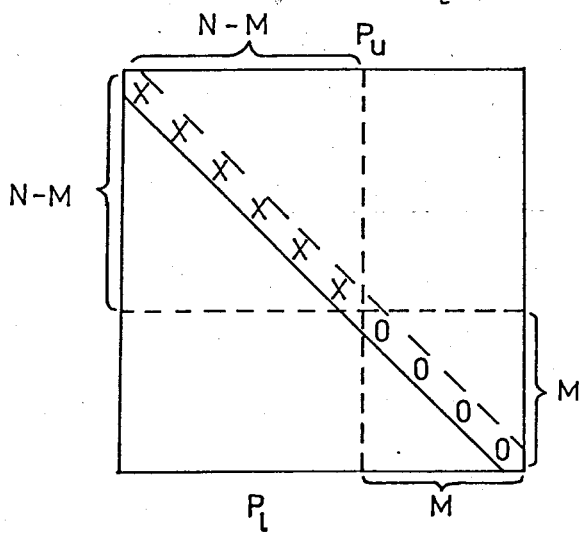


**Figure 5.** Method of obtaining the degeneracy (multiplicity) M of an eigenvalue E, after the transformation (figure 1), by the decomposition $P=(P_l , P_u)$ of A-E (figure 2), using its complete pivoting (figure 3), and after the iteration procedure for finding E (figure 4) leaving M zeros in the lower right end of the main diagonal of P. All the M eigenstates of E are obtained by solving the N-M order upper left determinant equations, with free terms: any of the M last right columns.



of this language may be well understood. First of all it preserves the possibilities of the usual basic for a dialogue between computer and user. As we are going to see, this makes the many branching commands during performance very convenient. On the other side it is much better than the usual basic and has most of the possibilities of the richer languages, e.g. fortran. In particular it automatically includes the use of a mathematical coprocessor if it is available to increase the execution speed. At the same time it also automatically includes the use of the whole memory, especially if an extension of it is available. However if one or both of these accessories are not available, the program can be used without them as well.

We are going to describe the program in a schematic way only. The input of the initial matrix A (figure 6) may be done from a file, prepared by the user after having calculated the matrix elements earliear, or manually directly. A possibility is included either to use this matrix as the ready one, or to construct by it a test matrix $\tilde{A}$ E $\tilde{A}^{-1}$ and exchange it with A , after input of known eigenvalues E (table 1) which are forgotten by the computer and may be found later by its usual procedure. Then the original matrix form can be changed from A into Q (figure 1) in order to increase execution speed.

Later on two main branches are open. In the r.h.s. of figure 6 is a short branch which might be called determinant (equations) solution (deso). After eventually having changed A to Q (figure 1) and then A to A-E, it fulfills the decomposition (figure 2) with pivoting (figure 3), finds the determinant D = D(A) , solves a determinant equation $A\times\bar{B} = B$ by finding the solution column $\bar{B}$ for any free column B , and finds the inverse matrix $S = A^{-1}$ (table 1) together with the inversion check $A\times S = I$ (table 2).

In the l.h.s. of figure 6 is shown the long branch which might be called matrix diagonalisation (madi). A short program finds the limits of the eigenvalues from below -EM and from above +EM . They can be

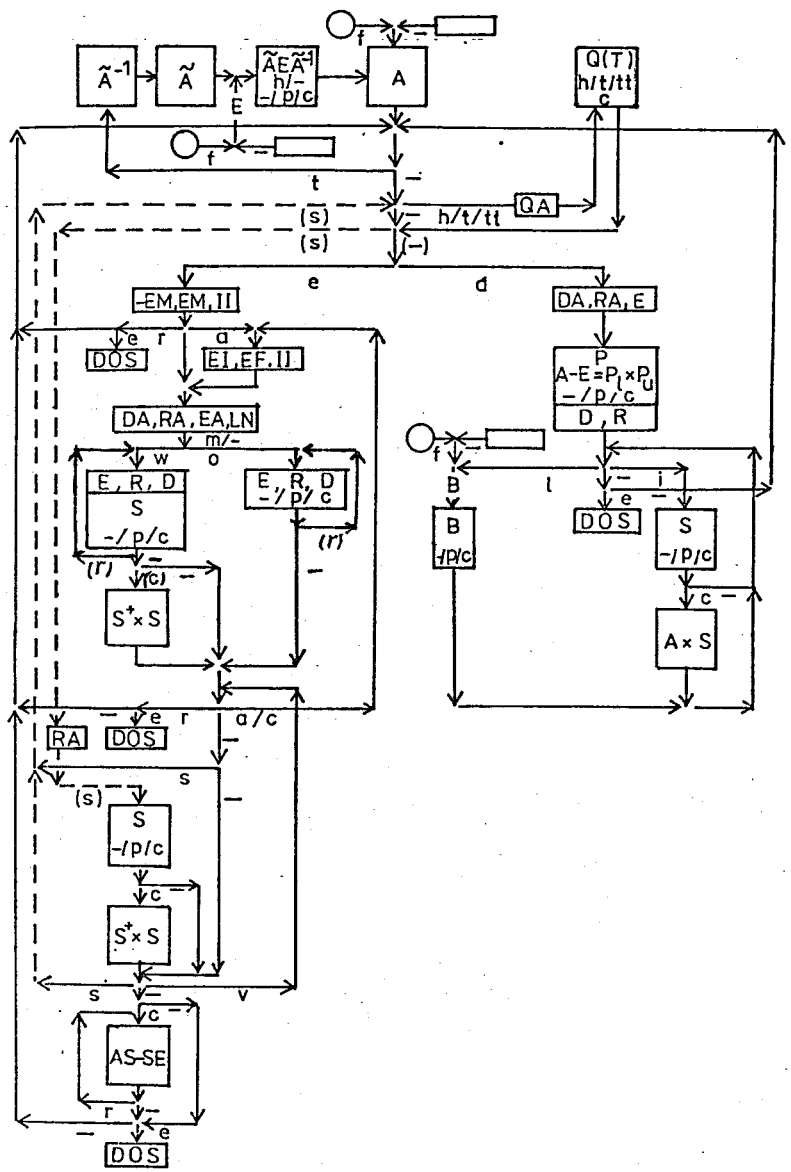


**Figure 6**. Program scheme; see table 1: notations, table 2: checks, table 3: commands, table 4: parameters.



changed by user to EI and respectively EF , if he has additional information about them and/or wishes to obtain only several of the eigenvalues situated in the interval (EI,EF) . By using all the programs and subprograms of the r.h.s. and additionally the iteration procedure we find the eigenvalues E (figure 4) , with or without the eigenstates S (table 1). The last ones may be found thereafter too. Any pivoting (figure 3) made during form transformation (figure 1) and matrix decomposition (figure 2) has been remembered and is restored at this step. If there are degenerate eigenvalues, to obtain their actual multiplicity and all their eigenstates, complete pivoting (figure 3) is usually necessary at the final iteration step (figure 4) with the method of figure 5 . An orthonormalisation check $S^+xS = I$ is proposed for hermitian matrices A . It may also play the role of checking if all eigenstates are independent for nonhermitian matrices A , for which generally $S^+xS \neq I$ . A final diagonalisation check $AxS - SxE = 0$ is offered too. It proves the actual validity of all the eigenvalues E as well of all the eigenstates S directly for the original form A of the matrix (table 2).

## 3. Instructions to users

These instructions are made evident in the program scheme presented in figure 6, together with the explanations in tables 1,2,3,4 . The input of the matrix is shown at the top of figure 6. The deso branch described in section 2 is presented in the r.h.s. of figure 6. The madi branch of section 2 is presented in the l.h.s. of figure 6. In addition to the program notations shown in table 1 and program checks shown in table 2, both of them commented in section 2, we must say a few words about the branching commands in table 3 and input parameters in table 4.

The branching commands (table 3) are used to change or repeat the way of performance. The user does not need table 3 except for planning

Table 1. Program notations:

A: ready or test matrix in original form; $\widetilde{A}$: its transformation into an orthogonal (unitary) matrix used to obtain symmetric (hermitian) test matrix $\widetilde{A} E \widetilde{A}^{-1} \longrightarrow A$ , or $\widetilde{A} = A$ if hermiticity is not required.

Q: hessenberg, tridiagonal accepted or tridiagonal modified form of A, together with the matrix transforming back the hessenberg or tridiagonal form (in the last case a matrix diagonal T is added) into original form, situated in the free Q (0 elements) memory space (figure 1).

P: matrix decomposition of $A-E = P_l \times P_u$ into a product of a lower $P_l$ and upper $P_u$ triangular matrices; for other than original forms $A \longrightarrow Q$ (figure 2).

S: inverse matrix $S = A^{-1}$ in the r.h.s., eigenstates matrix $AxS = SxE$ in the l.h.s.

B: linear equations free terms input vector column initially, their solution output vector column finally, in the r.h.s.

E: displacement in the r.h.s., eigenvalue in the l.h.s., multiplied by the unit matrix; eigenvalues diagonal matrix, input for test matrix construction $\widetilde{A} E \widetilde{A}^{-1} \longrightarrow A$ for the r.h.s. and l.h.s. initially, output after matrix diagonalisation in the l.h.s. finally.

D: determinant of A-E equal to the product of the $P_u$ diagonal elements after decomposition in the r.h.s. and l.h.s.

R: ratio, geometric average of the lower than RA diagonal elements of $P_u$ in the r.h.s. and l.h.s.

DOS: quit program.

Table 2. Program checks:

Inversion check in the r.h.s.: $AxS = I$ (unit matrix).
Orthonormalisatian check in the l.h.s.: $S^+xS = I$ if A is hermitian.
Diagonalisatian check in the l.h.s.: $AxS - SxE = 0$ (zero matrix).



his future actions, since at each branching point he obtains the whole text of the possible branching commands, together with the corresponding letters he must choose to press, on his screen as shown in table 3.

The input parameters (table 4) are demanded on the screen in the same way as listed in table 4. However they need some explanations since they may influence the accuracy and the speed. The transform accuracy QA is needed only when a transformation to a Q hessenberg or tridiagonal form is required. If the transformed form is good (during transformation no variable, by which one must divide [1] , happens to be neer to 0) then QA = 0 will be OK. Otherwise one might improve the Q form by taking a small number QA > 0 .

The ratio accuracy RA is one of the most important parameters for the madi l.h.s. branch. It determines the degeneracy of the eigenvalue M (figure 5). If it is too low, the eigenvalue will be found, but its degeneracy may be decreased even down to 0 . In the last case this means that it will not be recognised (numerated). If it is too high, the degeneracy of the eigenvalue may be increased, and thus other eigenvalues might be skipped. Happily the useful interval for RA is several orders of magnitude, e.g. roughly between the eigenvalue distance ED (figure 4) and the eigenvalue accuracy EA .

The eigenvalue accuracy EA is exactly what this means. A lower value of EA , meaning higher accuracy, at the same time increases the execution time, but only logarithmically.

The interval number LN must be chosen so that the eigenvalue distance ED = (EF - EI)/LN is lower than the actual minimal eigenvalue distance. It is not recommended to choose this LN much higher, correspondingly ED much lower, since this increases the execution time unnecessarily.

All other parameters of table 4 are used in special cases only which will be understood without explanations. E.g. the determinant

Table 3. Branching commands:

any other sign: .../-
input via file/direct: f/-
test/ready matrix: t/-
hermitian/any matrix: h/-
original/hessenberg/tridiagonal accepted/tridiagonal modified form: -/h/t/tt
pivoting no/partial/complete: -/p/c (figure 3)
eigenvalues/determinant: e/d (l.h.s./r.h.s.)
linear equations/inverse matrix/not: l/i/-
with/out eigenstates: w/o
return/alternate/unique: r/a/-
return/again/continue/not: r/a/c/-
states/not: s/-
values/states/not: v/s/-
manual/aut correction: m/-
change .../not: c/-
abs/rel correction: a/-
repeat .../not: r/-
... check/not: c/-
end/not: e/-

Table 4. Input parameters:

transform accuracy: QA (r.h.s. and l.h.s.)
determinant accuracy: DA (r.h.s. and l.h.s.)
ratio accuracy: RA (r.h.s. and l.h.s.)
displacement: E (r.h.s.)
eigenvalue accuracy: EA (l.h.s.)
interval number: LN (l.h.s.)
eigenvalue initial point: EI (l.h.s.) , instead of evaluated -EM
eigenvalue final point: EF (l.h.s.) , instead of evaluated +EM
initial number: II (l.h.s.) , instead of accepted 1
correction (abs corr): R (l.h.s.) , instead of evaluated R
multiplier (rel corr): D (l.h.s.) , DR instead of evaluated R



accuracy DA may in almost all cases be put DA = 0 . DA > 0 plays a role only to shorten the time of the eigenvalue calculation sometimes, but on the other hand this is dangerous since it might decrease the accuracy.

## 4. Files and tests

In the case of accepting this work as a JINR preprint, anybody interested will get a copy of a diskette containing the files necessary for use and test.

First of all, two files mainev.bas and veinev.bas are included. They have been prepared in usual basic language, since they must be accessible for changes to the particular needs of any user. The first one creates the input matrix A and outputs it into the changeable file matrix.dat (top of figure 6). The second one creates the free terms column B (r.h.s. of figure 6), or the known eigevalues row E (l.h.s. of figure 6), and outputs them into the changeable file vector.dat . Both of them can be used either manually-directly or if an authors subprogram starting from line number 100 is added, calculating the corresponding elements according to the user's wishes. If unchanged, after command - they will create the files introduced manually, but after command e they will create corresponding files with 0 elements only.

Then three files deso39.exe , deso67.exe , deso89.exe follow, compiled in turbo basic language. The numbers denote the maximal order of the matrix accessible for each file. All the three of them correspond to the r.h.s. of figure 6 only (its deso part) without the top of it except the input of a ready matrix A directly or from a file (i.e. without the creation of a test matrix and without its transformation into another form Q ) and without the l.h.s.: its madi part.

Thereafter the basic three files madi39.exe , madi67.exe ,

madi89.exe follow, compiled in turbo basic language as well, the numbers denoting the same maximal order. All the three of them correspond to the complete figure 6 (with its top part, its r.h.s. deso part and its l.h.s. madi part). Three maximal orders have been chosen due to the fact that turbo basic makes the performance for a matrix of the same order a little longer if a higher maximal order is chosen, and due to the fact that order 89 is inaccessible for a 256 kb memory, but accessible for a 640 kb memory, and order 67 is accessible for a 256 kb memory.

At last three matrix files matr06.dat , matr12.dat , matr24.dat and three vector files vect06.dat , vect12.dat , vect24.dat are available for tests of matrices of order 6 , 12 , 24 corresponingly. E.g. to use the test matrix and test vector of order 24 one must copy matr24.dat onto the changeable file matrix.dat, and vect24.dat onto the changeable file vector.dat. With each of them we can suggest many test combinations. In fact, we have 4 initial and 4 final forms, 3 initial and 3 final pivotings, i.e. $4^2 3^2 = 144$ form and pivoting choices, times an unknown set number of 4 important parameters (section 3).

With the above mentioned 24 order matrix we recommend the following tests. After the calling command: madi39, we choose the following branching commands and input parameters: f;-,d,-,c,0,0,0, and obtain the determinant of the ready matrix in its original form:
D = -5.120 764 213 753 242 ; then after: 1,f we obtain the solution of the determinant equation , its last element being
B(24) = -2.445 755 752 506 978 ; and after: i we obtain the inverse matrix with its last element S(24,24) = 3.851 300 418 968 261 E-003 . The inversion check is satisfied. After the following commands and parameters: madi39,f;-,e,o,t,-,0,t,p;-,-,0,1, .000 000 01, 400,- we get all the eigenvalues of the ready matrix, the last of which is E(24) = 7.962 485 292 138 446 , all of them with multiplicity M = 1 ;



after: -,-,s,t,p,0,1 we get all eigenstates components with the last one: S(24,24) = .228 704 973 291 901 3 .

In order to show the possibility to find degenerate eigenvalues with all their eigenstates, we construct a test hermitian matrix and diagonalise it by: madi39,f;t,h,c,0,0,0;f,-,-,-;e,o,t,-,0,t,c; -,-,0,.01, .000 000 01, 200,- obtaining all its eigenvalues E , with their multiplicities M , e.g. E(16-21) = 2.099 999 997 594 839 with the highest M = 6 and the last E(22-24) = 3.100 000 001 093 566 with M = 3 ; after: -,-,s,t,c,0,.01 we obtain all eigenstates components with the last one: S(24,24) = -.176 446 704 683 418 $_0$ .

To show the diagonalisation of a nonhermitian matrix, we construct a test nonhermitian matrix and diagonalise it by: madi39,f;t,-,c,0,0,0; f,-,-,-;e,o,t,-,0,t,c;-,a,-5.1,5.1,-,0,.01,.000 000 01, 200,- and find all the eigenvalues E with their multiplicities M , e.g. E(16-21) = 2.100 000 002 503 403 with the highest M = 6 and the last E(22-24) = 3.100 000 003 695 5$_{00}$ with M = 3 ; after : -,-,s,-,c,.01 we find all eigenstate components with the last one S(24,24) = 5.833 837 799 603 148 E-002 . With the more usual commands and parameters after the last ";" sign of the first sequence: -,-,0,.01, .000 000 01, 400,- one finds: E(16-21) = 2.099 999 998 669 201 and E(22-24) = 3.100 000 005 113 749 and with the same second sequence: S(24,24) = 5.833 837 821 984 524 E-002 . Results coinciding with the first ones for the same matrix will be obtained with the second tridiagonal modified form if we change the first sequence after the third ";" sign to: e,o,tt,-, .001, tt,c;-,a,-5.1,5.1,-,0,.001 , .000 000 01, 200,- and if we do not change the second sequence. Due to the nonhermiticity of A the orthonormalisation check shows that the eigenstates are normalised but generally they are nonorthogonal. On the other hand the diagonalisation check is satisfied, as in the other cases with good commands and parameters.

## 5. Characteristics

The execution time of obtaining all the eigenvalues of the 24 order matrix with accuracy $10^{-8}$ is $1^m47^s$ for a Pravetz-16 with coprocessor, about 5 times longer without coprocessor, and about 10 times shorter for a Microway. This time depends on the initial and final matrix forms and pivotings, on the eigenvalue accuracy EA and interval number LN , and strongly on the matrix order N . E.g. if EA = $10^{-8}$ , LN = 400 and N = 24 , the dependence of the time on the form is as follows. For t,p,t,p initial form and pivoting, final form and pivoting it will be: $4^m02^s$ , h,p,h,p : $5^m28^s$ , -,p,-,p : $30^m56^s$ . Its dependence on the pivoting is as follows. For t,-,t,- it will be: $1^m43^s$ , t,p,t,p : $4^m02^s$ as above, t,c,t,c : $68^m54^s$ . The time for t,c is almost the same as that for h,c and -,c . For the most relevant mixed cases like t,-,t,p it will be: $1^m48^s$ , t,-,t,c : $3^m33^s$ , t,p,t,c : $5^m48^s$ . For t,- and t,p it must increase with N as about $N^2$ , for h,- and h,p : as about $N^3$ , for t,c; h,c; -,-; -,p and -,c : as about $N^4$ .

In conclusion we summarise the main characteristics of the method:

1) Universality: a variety of methods with different accuracy and speed may be chosen by combinations of several possible initial and final forms, initial and final pivotings and parameter sets.

2) All these methods may be chosen during performance, depending on the results.

3) Reliability, accuracy and speed may be optimised due to 1) and 2). The usual speed of the diagonalisation of a 24 order matrix is equal to finding all eigenvalues in 110 s for Pravetz-16 with coprocessor, about 5 times slower without it and about 10 times faster for Microway. It decreases proportionally to about the square of the order.

4) The highest possible matrix order is 67 for a 256 kb memory or 89 for a 640 kb memory. The last number 89 may be increased about $\sqrt{2}$ times to about 125 by a minor change in the program.



5) Matrices with degenerate eigenvalues are accessible to the method.

6) The eigenvalues are found in the order of increase. This makes it possible not to search for all of them, but only for several in a definite interval, and thus to decrease the execution time considerably.

## References

[1] Wilkinson J H 1965 The algebraic eigenvalue problem (Oxford: Clarendon Press)

[2] Dahlquist G and Bjork A 1974 Numerical methods (London: Prentice Hall)

Received by Publishing Department
on June 8, 1990.

Наджаков Е.Г. E11-90-391

Диагонализация матриц на персональных компьютерах.

Предложены алгоритм и программа для диагонализации реальных, но не обязательно эрмитовых матриц. Выбран компромисс, делающий их универсальными, со многими выборами доступными во время исполнения, надежными, точными, быстрыми и приложимыми к матрицам высокого порядка в то же самое время. Для алгоритма это достигнуто путем незнакомой комбинации знакомых современных вычислительных методов. Применен новый метод для обработки матриц с вырожденными собственными значениями. Программа реализована на языке турбо бейсик. Это позволяет использовать возможности как математического копроцессора для более быстрого выполнения, так и расширенной памяти для приложения к матрицам более высокого порядка, если эти дополнения есть, или применять программу без них если их нет.

Работа выполнена в Лаборатории теоретической физики ОИЯИ.

Сообщение Объединенного института ядерных исследований. Дубна 1990

Nadjakov E.G. E11-90-391

Matrix Diagonalisation on Personal Computers

Algorithm and program for real, but not necessarily hermitian, matrix diagonalisation are suggested. A compromise has been chosen which makes them universal, with many choices accessible to user during perfomance, reliable, accurate, fast and applicable to high order matrices at the same time. For the algorithm this has been achieved by an unknown combination of known contemporary computational methods. A new method has been applied to treat matrices with degenerate eigenvalues. The program has been realised in turbo basic language. This makes it able to use the possibilities of both mathematical coprocessor for faster execution and of extended memory for application to higher order matrices if these accessories are available, or to apply the program without them if they are not available.

The investigation has been performed at the Laboratory of Theoretical Physics, JINR.

Communication of the Joint Institute for Nuclear Research. Dubna 1990