сообщения
объединенного
института
ядерных
исследований
дубна

Р11-88-909

В.И.Коробов

МЕТОД ОБРАТНОЙ ИТЕРАЦИИ С РЕГУЛЯРИЗАЦИЕЙ.

АЛГОРИТМЫ И ПРОГРАММЫ

1988

В работе рассматривается численная реализация метода обратной итерации с регуляризацией для симметричной обобщенной задачи на собственные значения

$$Ax = \lambda Bx . \qquad (I)$$

Относительно матриц A и B предполагается, что они симметричны и матрица B положительно определена.

Метод обратной итерации хорошо известен в литературе /I/ как эффективный метод решения задачи на собственные значения в случае, если требуется найти одно собственное значение и соответствующий ему собственный вектор. Описываемая ниже схема использует процедуру декомпозиции Банча /2/ (см. также /6/), необходимую для решения системы уравнений с симметричной, но не положительно определенной матрицей. Для получения устойчивых решений в алгоритме программы предусмотрена возможность регуляризации задачи /3/. В заключение приводятся тексты программ на фортране.

Схема метода обратной итерации описывается уравнениями

$$(A-\sigma_k B)u_{k+1} = Bv_k, \quad v_k = u_k/\|u_k\| . \qquad (2)$$

Здесь σ_k и u_k – приближения к собственному значению λ_i и собственному вектору x_i задачи (I), и при надлежащем выборе σ_k, u_k сходится к x_i .

Наиболее известны две стратегии выбора смещения σ_k . По первой стратегии σ_k выбирается равным отношению Релея вектора u_k /I/, $\sigma_k = \rho = (u_k^T Au_k)/(u_k^T Bu_k)$. Как следует из оценки Темпла /I/, приближение, использующее отношение Релея, имеет асимптотически квадратичную зависимость от нормы вектора невязки $r = Au - \rho Bu$. В окрестности решения процесс обладает кубической скоростью сходимости.

По второй стратегии σ_k выбирается постоянной, близкой к искомому собственному значению, при этом сходимость процесса к решению линейна. Если приближение к собственному значению выбрано удачно, то достаточно нескольких итераций для получения требуемого решения.

Оценим стоимость метода обратной итерации, описываемого схемой (2). Умножение вектора на матрицу B имеет сложность n^2 умножений. Решение системы уравнений с симметричной матрицей ($A-\sigma_k B$) распадается на два этапа: декомпозицию матрицы и собственно решение системы $(A-\sigma_k B)u=b$ с учетом разложения матрицы на треугольные .

Стоимость первой операции с использованием методов для симметричных матриц эквивалентна приблизительно $n^3/6$ умножений. Вторая же операция выполняется за n^2 умножений. Отсюда видно, что основное время при использовании метода обратной итерации должно расходоваться на декомпозицию матриц. С учетом сказанного можно сделать вывод, что выбор второй стратегии предпочтителен,так как при этом декомпозиция матрицы выполняется только один раз.

Стандартный метод обратной итерации используется для получения одного собственного значения и собственного вектора. В случае, когда требуется найти более одного собственного значения, можно использовать обратную итерацию с итерированием на подпространстве /4/.

В работе /3/ отмечается, что метод обратной итерации является наиболее устойчивым методом для решения задачи (I) с плохообусловленной парой (A,B). Используя обратный анализ ошибок /5/, можно показать, что матрица ошибок, эквивалентных ошибкам, возникающим в результате погрешности вычислений, имеет величину порядка точности округлений машинного представления чисел – элементов,входящих в матрицы A и B. Она существенно меньше, чем соответствующие матрицы ошибок, получаемые при использовании метода приведения задачи (I) к стандартному виду или QZ – алгоритма /I/.

Устойчивость приближения зависит от выбора параметра смещения σ. Это видно из следующих рассуждений.

Абсолютная погрешность решения системы $(A-\sigma B)y=g$ приближенно равна $\|(A-\sigma B)^{-1}\|\cdot\|H\|\cdot\|g\|$, где H –матрица возмущений, компенсирующая погрешности вычисления декомпозиции матрицы. Относительная погрешность решения системы $\|(A-\sigma B)^{-1}\|\cdot\|H\|\cdot\|g\|/\|y\|$ минимизируется с ростом $\|y\|$. Это достигается за счет приближения смещения σ к точному собственному значению, так как в этом случае компонента решения, отвечающая соответствующему собственному вектору, становится доминирующей. Таким образом, подбирая начальное приближение как можно ближе к собственному значению, мы увеличиваем относительную точность решения системы $(A-\sigma B)y=g$ и соответственно относительную точность нового приближения к собственному значению, полученного в результате использования метода обратной итерации. Практические вычисления подтверждают эту закономерность.

В случае, когда пара матриц (A,B) является плохо обусловленной, возможно использование методов регуляризации задачи /3/. Регуляризованные матрицы имеют вид

$$A_R=A+s\alpha D\ ,\qquad B_R=B+s\beta D\ ,$$

где (α,β) – вектор направления регуляризации ($\alpha^2+\beta^2=1$ и $\beta\geq 0$), s – параметр регуляризации, а D – диагональная матрица, имеющая положительные элементы на своей главной диагонали. В программных реализациях алгоритма,рассматриваемых ниже,в качестве такой матрицы используется матрица $D=diag/(A-\sigma B)/$, состоящая из абсолютных значений диагональных элементов матрицы ($A-\sigma B$) и нулевых внедиагональных элементов. При этом вектор регуляризации $(\alpha,\beta)=(1,0)$, а параметр регуляризации s задается при вызове подпрограммы.

Ключевым моментом в разработке эффективной программы метода обратной итерации является реализация алгоритма решения систем линейных уравнений $Ax=b$ с симметричной матрицей A. Используемая при этом матрица может не являться положительно (отрицательно) определенной, это делает применение схемы Холецкого декомпозиции матриц нежелательным из-за неустойчивости процедуры. В работах /2,7/ предложены различные устойчивые схемы разложения симметричных матриц, обладающие той же вычислительной сложностью,что и схема Холецкого. В рассматриваемой нами реализации используется схема декомпозиции Банча /2/.

Алгоритм Банча имеет много программных реализаций, работающих с матрицами, хранящимися в симметричной форме. Однако использование традиционной схемы редукции матрицы сверху вниз приводит к увеличению времени счета в сравнении с методом Холецкого в I,5–2 раза (подобное увеличение времени имеет место, например, в программе LEQ1S из библиотеки IMSL). В Приложении приводится текст программы с именем LEQ1S , использующей редукцию матрицы снизу вверх и имеющей время выполнения всего на 5% больше, чем программы, реализующие метод Холецкого.

Подобная разница во времени связана с тем, что в последнем случае выборка элементов из памяти ЭВМ происходит последовательно, и это дает ускорение процесса выборки элементов на аппаратном уровне.

Тексты описываемых ниже программ INVIT и INVITD имеются в Приложении.

Программа INVIT реализует метод обратной итерации в оперативной памяти ЭВМ. Она использует симметричную форму хранения матриц A и B . Элементы матрицы при симметричной форме хранения расположены в линейном массиве в следующем порядке: $a_{11}, a_{12}, a_{22}, a_{13}, a_{23}, a_{33}, \ldots, a_{nn}$, всего $n(n+1)/2$ элементов на одну матрицу.

Алгоритм INVIT :

I. Инициализировать начальное приближение к собственному вектору $y_0=(1,1,1,\ldots,1)$ и собственному значению $Q_0=\sigma$.

2. Заменить матрицу A матрицей ($A-\sigma B$) и ввести регуляризующее смещение, если это необходимо.

3. Сделать декомпозицию матрицы ($A-\sigma B$).

Начало цикла обратной итерации.

4. Нормировать приближение к собственному вектору $x_k=y_k/\|y_k\|$.

5. Решить систему $(A-\sigma B)y_{k+1}=Bx_k$.

6. Вычислить приближение к собственному значению

$\omega_{k+1} = \sigma + 1/(y_{k+1}, x_k)$.

7. Подвергнуть ω_{k+1} тесту на сходимость. Если сходимость не достигнута, перейти к пункту 4. Конец цикла обратной итерации.

8. Положить $\lambda = \omega_{k+1}$, $v = y_{k+1} / \| y_{k+1} \|_B$, где $\|y\|_B = \sqrt{(y, By)}$, и выйти.

После выполнения программы INVIT матрица A не сохраняется. Требуемый объем оперативной памяти $\sim n^2+4n$. Подробная информация о передаваемых параметрах содержится в тексте программы.

Программа INVITD использует матрицы A и B , хранящиеся на внешних носителях. При активации программы в оперативную память ЭВМ загружается матрица ($A-\sigma B$) с учетом регуляризации, и все дальнейшие манипуляции (декомпозиция, решение систем уравнений) производятся только с этой матрицей, оставляя матрицы A и B без изменений. Для обращения к матрицам A и B используется подпрограмма MVMUL , реализующая умножение матрицы на вектор для матриц, находящихся на внешних носителях.

Алгоритм INVITD получается из алгоритма INVIT , если заменить пункты 2 и 8 пунктами:

2'. Загрузить матрицу ($A-\sigma B$) в оперативную память и ввести регуляризующее смещение, если это необходимо.

8'. Положить λ равным отношению Релея $\lambda = (y_{k+1}, Ay_{k+1})/(y_{k+1}, By_{k+1})$, нормировать собственный вектор $v = y_{k+1} / \| y_{k+1} \|_B$, где $\| y \|_B = \sqrt{y, By}$, и выйти.

В последнем пункте алгоритма для оценки собственного значения по отношению Релея используются нерегуляризованные матрицы A и B . Это позволяет уменьшить систематическое смещение оценки, вносимое регуляризатором.

Использование программы INVITD сокращает объем требуемой оперативной памяти до $\sim n(n+1)/2+3n$, оставляя характеристики времени выполнения программ без изменений. Информация о передаваемых параметрах содержится в тексте программы в Приложении.

Практическое использование метода обратной итерации с регуляризацией показало, что для получения устойчивых собственных значений целесообразно использовать параметр регуляризации величиной порядка машинного эпсилон, а в случае хорошо определенной задачи его значение должно быть равным нулю.

Эффективным способом борьбы с неустойчивостью является переход от двойной точности к четвертой. Реализации программ, использующие арифметику четвертой точности и аналогичные программам INVIT и INVITD , имеются. Однако постоянная эксплуатация этих программ очень обременительна, поскольку время счета возрастает более чем в 4 раза, а требование к памяти - в 2 раза.

Приложение

В этом приложении приводятся тесты программ INVIT и INVITD , реализующих метод обратной итерации, и текст программы LEQ1S решения систем линейных уравнений $Ax=b$ с симметричной матрицей A .

```
      SUBROUTINE INVIT(A,B,NDIM,EIG,EPS,X,WA)
C-----------------------------------------------------------------
C
C     COMPUTER            - IBM/DOUBLE
C
C     COMPILER            - FORTRAN VS
C
C     LATEST REVISION     - JULY 03, 1987
C
C     PERPOSE             - INVERSE ITERATION FOR THE
C                             GENERALIZED EIGENVALUE PROBLEM
C                             A*X = E*B*X - SYMMETRIC STORAGE
C                             MODE.
C
C     USAGE               - CALL INVIT(A,B,NDIM,EIG,EPS,X,WA)
C
C     ARGUMENTS      A    - LINEAR ARRAY, ON INPUT A CONTAINS
C                             THE ELEMENTS OF MATRIX A IN
C                             SYMMETRIC STORAGE MODE, THE LENGTH
C                             OF A IS NDIM*(NDIM+1)/2, ON OUTPUT
C                             A IS DESTROIED.
C                    B    - LINEAR ARRAY, ON INPUT B CONTAINS THE
C                             ELEMENTS OF MATRIX B IN SYMMETRIC
C                             STORAGE MODE.
C                    NDIM - DIMENSION OF MATRIX A AND B.
C                    EIG  - ON INPUT, INITIAL APPROATCH TO THE
C                             EIGENVALUE, ON OUTPUT, REFINED
C                             ESTIMATE OF THE EIGENVALUE.
C                    EPS  - PARAMETER OF REGULARIZATION,
C                             IF EIG IS ONE OF THE LOWEST
C                               EIGENVALUES, THEN EPS MUST BE
C                               GREATER OR EQUAL TO ZERO,
C                             IF EIG IS ONE OF TNE UPPEST
C                               EIGENVALUES, THEN EPS MUST BE
C                               LESS OR EQUAL TO ZERO,
C                             IF THE PROBLEM IS WELL DEFINED,
C                               THEN IT IS BETTER TO SET THE
C                               VALUE OF EPS TO ZERO.
C                    X    - ON OUTPUT, THE EIGENVECTOR WITH
C                             EQUAL TO ONE B-NORM, (X,B*X)=1.
C                    WA   - WORK ARRAY OF LENGTH 2*NDIM
C
C     REQD. ROUTINES      - LEQ1S,MVMUL
C
C-----------------------------------------------------------------
      IMPLICIT REAL*8     (A-H,O-Z)
      DIMENSION           A(1),B(1),X(1),WA(1)
      DATA                ZERO/0.D0/,ONE/1.D0/,SIXTN/16.D0/
C
      NDET = NDIM+1
      REPS = ONE
    5 REPS = REPS/2
      IF (ONE.LT.ONE+REPS) GOTO 5
      ERREST = MAX(SIXTN*NDIM*REPS,ABS(EPS))
C
```

```
C                         EIGENVECTOR INITIALIZATION
C
      DO 10 I=1,NDIM
         X(I) = ONE
   10 CONTINUE
C
C                         REPLACE MATRIX A WITH MATRIX
C                           (A-EIG*B)
C
      EOLD = EIG
      II = 0
      DO 30 I=1,NDIM
         IM1 = I-1
         DO 20 J=1,IM1
            II = II+1
            A(II) = A(II)-EIG*B(II)
   20    CONTINUE
         II = II+1
         A(II) = A(II)-EIG*B(II)
         A(II) = A(II)+EPS*ABS(A(II))
   30 CONTINUE
C
C                         INVERSE ITERATION
C
      DO 70 IT=1,10
         IJOB = 2
         IF (IT.EQ.1) IJOB = 0
C
C                         MULTIPLY ON MATRIX B AND NORMALIZE
C                           VECTOR X
C
         CALL MVMUL(B,NDIM,X,WA)
         SUM = ZERO
         DO 40 I=1,NDIM
            SUM = SUM+X(I)*X(I)
   40    CONTINUE
         SUM = ONE/SQRT(SUM)
         DO 50 I=1,NDIM
            T = WA(I)*SUM
            WA(I) = X(I)*SUM
            X(I) = T
   50    CONTINUE
C
C                         MULTIPLY ON MATRIX INV(A-EIG*B)
C
         CALL LEQ1S(A,NDIM,X,1,NDIM,IJOB,WA(NDET),IER)
C
C                         CALCULATE REFINED EIGENVALUE
C
         SUM = ZERO
         DO 60 I=1,NDIM
            SUM = SUM+X(I)*WA(I)
   60    CONTINUE
         EPREV = EIG
         EIG = EOLD+ONE/SUM
```



```
         IF (ABS(EPREV-EIG).LT.ERREST*ABS(EIG)) GOTO 75
C
   70 CONTINUE
C
   75 CALL MVMUL(B,NDIM,X,WA)
      SUM = ZERO
      DO 80 I=1,NDIM
         SUM = SUM+WA(I)*X(I)
   80 CONTINUE
      SUM = ONE/SQRT(SUM)
      DO 90 I=1,NDIM
         X(I) = X(I)*SUM
   90 CONTINUE
C
C                         EXIT
C
      RETURN
      END
      SUBROUTINE MVMUL(A,N,V,VM)
C-----------------------------------------------------------------
C
C   COMPUTER             - IBM/DOUBLE
C
C   COMPILER             - FORTRAN VS
C
C   LATEST REVISION      - JULY 03, 1987
C
C   PERPOSE              - MULTIPLY ON MATRIX A, NUCLEUS FOR
C                            INVIT.
C
C   REQD. ROUTINES       - NONE REQUIRED.
C
C-----------------------------------------------------------------
      IMPLICIT REAL*8     (A-H,O-Z)
      DIMENSION           V(1),VM(1),A(1)
      DATA                ZERO/0.0D0/
C
      VM(1) = V(1)*A(1)
      II = 1
      DO 20 I=2,N
         IM1 = I-1
         VM(I) = ZERO
         DO 10 J=1,IM1
            VM(I) = VM(I)+A(II+J)*V(J)
            VM(J) = VM(J)+A(II+J)*V(I)
   10    CONTINUE
         II = II+I
         VM(I) = VM(I)+A(II)*V(I)
   20 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE INVITD(NDIM,EIG,EPS,X,WA)
C-----------------------------------------------------------------------
C
C     COMPUTER             - IBM/DOUBLE
C
C     COMPILER             - FORTRAN VS
C
C     LATEST REVISION      - AUGUST 6, 1987
C
C     PERPOSE              - INVERSE ITERATION FOR THE
C                              GENERALIZED EIGENVALUE PROBLEM
C                              A*X = E*B*X - SYMMETRIC STORAGE
C                              MODE - EXTERNAL STORAGE.
C
C     USAGE                - CALL INVITD(NDIM,EIG,EPS,X,WA)
C
C     ARGUMENTS      NDIM  - DIMENSION OF MATRIX A AND B.
C                    EIG   - ON INPUT, INITIAL APPROATCH TO THE
C                              EIGENVALUE, ON OUTPUT, REFINED
C                              ESTIMATE OF THE EIGENVALUE.
C                    EPS   - PARAMETER OF REGULARIZATION,
C                              IF EIG IS ONE OF THE LOWEST
C                                EIGENVALUES, THEN EPS MUST BE
C                                GREATER OR EQUAL TO
C                                ZERO,
C                              IF EIG IS ONE OF TNE UPPEST
C                                EIGENVALUES, THEN EPS MUST BE
C                                LESS OR EQUAL TO ZERO,
C                              IF THE PROBLEM IS WELL DEFINED,
C                                THEN IT IS BETTER TO SET THE
C                                VALUE OF EPS TO ZERO.
C                    X     - ON OUTPUT, THE EIGENVECTOR WITH
C                              EQUAL TO ONE B-NORM, (X,B*X)=1.
C                    WA    - WORK ARRAY OF LENGTH
C                              NDIM*(NDIM+1)/2+2*NDIM.
C
C     REQD. ROUTINES       - AMSB,IOBUFF,LEQ1S,MVMUL
C
C     REMARKS              - MATRICES A AND B STORED ON EXTERNAL
C                              UNITS WITH UNIT NUMBERS 15 AND 10
C                              RESPECTIVELY.
C
C-----------------------------------------------------------------------
      IMPLICIT                  REAL*8 (A-H,O-Z)
      DIMENSION                 X(1),WA(1)
      DATA                      ZERO/0.D0/,ONE/1.D0/,SIXTN/16.D0/
C
C                               FIRST EXECUTABLE STATEMENT
C
      ND = NDIM+1
      NA = ND+NDIM
      REPS = ONE
    5 REPS = REPS/2
      IF (ONE.LT.ONE+REPS) GOTO 5
      ERREST = MAX(SIXTN*NDIM*REPS,ABS(EPS))
      EOLD = EIG
C
C                               OPEN FILES
```



```
C
      IUNIT1 = 10
      IUNIT2 = 15
      OPEN (UNIT=10,IOSTAT=IOVAL,ACCESS='DIRECT',
     *      STATUS='OLD',RECL=13024,FILE='FCOMPB')
      IF (IOVAL.NE.0) STOP
      OPEN (UNIT=15,IOSTAT=IOVAL,ACCESS='DIRECT',
     *      STATUS='OLD',RECL=13024,FILE='FCOMPA')
      IF (IOVAL.NE.0) STOP
C
C                               EIGENVECTOR INITIALIZATION
C
      DO 10 I=1,NDIM
         X(I) = ONE
   10 CONTINUE
C
C                               SET MATRIX (A-EIG*B) IN WA(NA)
C
      CALL AMSB(WA(NA),NDIM,EIG,EPS,IUNIT1,IUNIT2)
C
C                               INVERSE ITERATION
C
      CALL LEQ1S(WA(NA),NDIM,X,1,NDIM,1,WA(ND),IER)
      DO 50 IT=1,10
C
C                               MULTIPLY ON MATRIX B AND NORMALIZE
C                                 VECTOR X
C
         CALL MVMUL(NDIM,WA,X,IUNIT1)
         SUM = ZERO
         DO 20 I=1,NDIM
            SUM = SUM+X(I)*X(I)
   20    CONTINUE
         SUM = ONE/SQRT(SUM)
         DO 30 I=1,NDIM
            TEMP = WA(I)*SUM
            WA(I) = X(I)*SUM
            X(I) = TEMP
   30    CONTINUE
C
C                               MULTIPLY ON MATRIX INV(A-ESM*B)
C
         CALL LEQ1S(WA(NA),NDIM,X,1,NDIM,2,WA(ND),IER)
C
C                               CALCULATE REFINED EIGENVALUE
C
         SUM = ZERO
         DO 40 I=1,NDIM
            SUM = SUM+X(I)*WA(I)
   40    CONTINUE
         EPREV = EIG
         EIG = EOLD+ONE/SUM
         IF (ABS(EPREV-EIG).LT.ERREST*ABS(EIG)) GOTO 55
C
   50 CONTINUE
C
   55 CALL MVMUL(NDIM,WA,X,IUNIT2)
      TEMP = ZERO
```

```
      DO 60 I=1,NDIM
         TEMP = TEMP+WA(I)*X(I)
   60 CONTINUE
      CALL MVMUL(NDIM,WA,X,IUNIT1)
      SUM = ZERO
      DO 65 I=1,NDIM
         SUM = SUM+WA(I)*X(I)
   65 CONTINUE
      EIG = TEMP/SUM
      SUM = ONE/SQRT(SUM)
      DO 70 I=1,NDIM
         X(I) = X(I)*SUM
   70 CONTINUE
C
C                            EXIT
C
      RETURN
      END
      SUBROUTINE AMSB(A,NDIM,SHFT,EPS,IUNIT1,IUNIT2)
C-------------------------------------------------------------------
C
C   COMPUTER                 - IBM/DOUBLE
C
C   COMPILER                 - FORTRAN VS
C
C   LATEST REVISION          - AUGUST 6, 1987
C
C   PERPOSE                  - SET MATRIX (A-SHFT*B) IN REAL
C                                MEMORY, NUCLEUS FOR INVITD.
C
C   USAGE                    - CALL AMSB(A,NDIM,SHFT,EPS,
C                                        IUNIT1,IUNIT2)
C
C   ARGUMENTS      A         - LINEAR ARRAY, WHERE MATRIX
C                                (A-SHFT*B) STORED.
C                  NDIM      - ORDER OF MATRICES.
C                  SHFT      - SHIFT COEFFICIENT.
C                  EPS       - REGULARIZATOR.
C                  IUNIT1    - UNIT NUMBER FOR FILE WITH MATRIX B.
C                  IUNIT2    - UNIT NUMBER FOR FILE WITH MATRIX A.
C
C   REQD. ROUTINES           - IOBUFF
C
C-------------------------------------------------------------------
      IMPLICIT                REAL*8 (A-H,O-Z)
      DIMENSION               BUF1(1628),BUF2(1628),A(1)
C
C                             FIRST EXECUTABLE STATEMENT
C
      II = 0
      M = 1628
      IREC = 1
      DO 20 I=1,NDIM
         DO 10 J=1,I
            II = II+1
            IF (M.EQ.1628) THEN
               M = 0
               CALL IOBUFF(IREC,BUF1,1,IUNIT1)
               CALL IOBUFF(IREC,BUF2,1,IUNIT2)
               IREC = IREC+1
            END IF
            M = M+1
            A(II) = BUF2(M)-SHFT*BUF1(M)
   10    CONTINUE
         A(II) = A(II)+EPS*ABS(A(II))
   20 CONTINUE
C
      RETURN
      END
      SUBROUTINE IOBUFF(IREC,BUFF,IJOB,IUNIT)
C-------------------------------------------------------------------
C
C   COMPUTER                 - IBM/DOUBLE
C
C   COMPILER                 - FORTRAN VS
C
C   LATEST REVISION          - AUGUST 6, 1987
C
C   PERPOSE                  - READ AND/OR WRITE THE BUFFER BUFF
C                                OF/TO THE EXTERNAL FILE.
C
C   USAGE                    - CALL IOBUFF(IREC,BUFF,IJOB,IUNIT)
C
C   ARGUMENTS      IREC      - RECORD NUMBER.
C                  BUFF      - BUFFER FOR INPUT/OUTPUT.
C                  IJOB      - INPUT OPTION PARAMETER,
C                                IF IJOB=1,THEN READ BUFF FROM
C                                  THE EXTERNAL FILE,
C                                IF IJOB=2,THEN WRITE BUFF TO THE
C                                  EXTERNAL FILE,
C                                IF IJOB=3,THEN WRITE PRECEEDING
C                                  BUFFER TO THE EXTERNAL FILE AND
C                                  READ NEW BUFFER WITH RECORD
C                                  NUMBER IREC.
C                  IUNIT     - UNIT NUMBER.
C
C   REQD. ROUTINES           - NONE REQUIRED.
C
C-------------------------------------------------------------------
      IMPLICIT                REAL*8 (A-H,O-Z)
      DIMENSION               BUFF(1628)
C
C                             FIRST EXECUTABLE STATEMENT
C
      IWR = IREC
      IF (IJOB.EQ.3) IWR = IWR-1
      IF (IJOB.EQ.1) GOTO 10
      WRITE (UNIT=IUNIT,REC=IWR,IOSTAT=IOVAL) BUFF
      IF (IOVAL.NE.0) STOP
      IF (IJOB.EQ.2) RETURN
   10 CONTINUE
      READ (UNIT=IUNIT,REC=IREC,IOSTAT=IOVAL) BUFF
      IF (IOVAL.NE.0) STOP
C
      RETURN
      END
```

```
      SUBROUTINE MVMUL(NDIM,XX,X,IUNIT)
C-----------------------------------------------------------------
C
C   COMPUTER                 - IBM/DOUBLE
C
C   COMPILER                 - FORTRAN VS
C
C   LATEST REVISION          - AUGUST 6, 1987
C
C   PERPOSE                  - MULTIPLY ON MATRIX THAT STORED ON
C                                EXTERNAL UNIT IUNIT, NUCLEUS
C                                FOR INVITD.
C
C   USAGE                    - CALL MVMUL(NDIM,XX,X,IUNIT)
C
C   ARGUMENTS      NDIM      - ORDER OF MATRIX AND LENGTH OF
C                                VECTORS XX AND X.
C                  XX        - OUTPUT VECTOR (XX=A*X).
C                  X         - INPUT VECTOR.
C                  IUNIT     - EXTERNAL UNIT NUMBER.
C
C   REQD. ROUTINES           - IOBUFF
C
C-----------------------------------------------------------------
      IMPLICIT                REAL*8 (A-H,O-Z)
      DIMENSION               XX(1),X(1),BUFF(1628)
      DATA                    ZERO/0.D0/
C
C                             FIRST EXECUTABLE STATEMENT
C
      IREC = 1
      M = 1
      CALL IOBUFF(IREC,BUFF,1,IUNIT)
      XX(1) = BUFF(M)*X(1)
      DO 20 I=2,NDIM
         IM1 = I-1
         XX(I) = ZERO
         DO 10 J=1,IM1
            M = M+1
            XX(J) = XX(J)+BUFF(M)*X(I)
            XX(I) = XX(I)+BUFF(M)*X(J)
            IF (M.LT.1628) GOTO 10
            M = 0
            IREC = IREC+1
            CALL IOBUFF(IREC,BUFF,1,IUNIT)
   10    CONTINUE
         M = M+1
         XX(I) = XX(I)+BUFF(M)*X(I)
         IF(M.LT.1628) GOTO 20
         M = 0
         IREC = IREC+1
         CALL IOBUFF(IREC,BUFF,1,IUNIT)
   20 CONTINUE
C
      RETURN
      END
```



```
      SUBROUTINE LEQ1S (A,N,B,M,IB,IJOB,WA,IER)
C-----------------------------------------------------------------
C
C   COMPUTER                 - IBM/DOUBLE
C
C   COMPILER                 - FORTRAN VS
C
C   LATEST REVISION          - DECEMBER 3, 1986
C
C   PURPOSE                  - LINEAR EQUATION SOLUTION -
C                                INDEFINITE MATRIX - SYMMETRIC
C                                STORAGE MODE - SPACE ECONOMIZER
C                                SOLUTION
C
C   USAGE                    - CALL LEQ1S (A,N,B,M,IB,IJOB,WA,IER)
C
C   ARGUMENTS      A         - INPUT/OUTPUT VECTOR OF DIMENSION
C                                N*(N+1)/2. SEE PARAMETER IJOB.
C                  N         - ORDER OF MATRIX A AND THE NUMBER OF
C                                ROWS IN B. (INPUT)
C                  B         - INPUT/OUTPUT MATRIX OF DIMENSION N
C                                BY M. ON INPUT, B CONTAINS THE M
C                                RIGHT-HAND SIDES OF THE EQUATION
C                                AX = B. ON OUTPUT, THE SOLUTION
C                                MATRIX X REPLACES B. IF IJOB = 1,
C                                B IS NOT USED.
C                  M         - NUMBER OF RIGHT HAND SIDES (COLUMNS
C                                IN B). (INPUT)
C                  IB        - ROW DIMENSION OF B EXACTLY AS
C                                SPECIFIED IN THE DIMENSION
C                                STATEMENT IN THE CALLING PROGRAM.
C                                (INPUT)
C                  IJOB      - INPUT OPTION PARAMETER. IJOB = I
C                                IMPLIES WHEN
C                                I = 0, FACTOR THE MATRIX A AND
C                                  SOLVE THE EQUATION AX = B. ON
C                                  INPUT, A CONTAINS THE COEFFICIENT
C                                  MATRIX OF THE EQUATION AX = B,
C                                  WHERE A IS ASSUMED TO BE AN
C                                  N BY N SYMMETRIC MATRIX. A IS
C                                  STORED IN SYMMETRIC STORAGE MODE
C                                  AND THEREFORE HAS DIMENSION
C                                  N*(N+1)/2. ON OUTPUT, A IS
C                                  REPLACED BY ITS FACTORIZED FORM.
C                                I = 1, FACTOR THE MATRIX A.
C                                  A CONTAINS THE SAME INPUT/OUTPUT
C                                  INFORMATION AS IF IJOB = 0. B IS
C                                  NOT USED.
C                                I = 2, SOLVE THE EQUATION AX = B.
C                                  THIS OPTION IMPLIES THAT LEQ1S
C                                  HAS ALREADY BEEN CALLED USING
C                                  IJOB = 0 OR 1 SO THAT THE MATRIX
C                                  A HAS ALREADY BEEN FACTORED.
C                                  IN THIS CASE, OUTPUT MATRIX A
C                                  MUST HAVE BEEN SAVED FOR REUSE
C                                  IN THE CALL TO LEQ1S.
```

```
C                WA      - WORK AREA OF LENGTH N.
C                IER     - ERROR PARAMETER. (OUTPUT)
C                          WARNING ERROR (WITH FIX)
C                            IER = 65 INDICATES THAT MATRIX A
C                              IS ALGORITHMICALLY SINGULAR.
C                          TERMINAL ERROR
C                            IER = 129 INDICATES THAT MATRIX A
C                              IS EXACTLY SINGULAR ON ONE STEP
C                              OF PROCEDURE.
C
C-----------------------------------------------------------------
      IMPLICIT           REAL*8 (A-H,O-Z)
      DIMENSION          A(1),B(IB,1),WA(1)
      DATA               ZERO/0.0D0/,SIXTN/16.0D0/
      DATA               ALPHA/0.6403882D0/
C
C                                  FIRST EXECUTABLE STATEMENT
C                                    INITIALIZE IER
C
      IER = 0
C
C   ****  FIRST STAGE  ****        DECOMPOSE A INTO THE
C                                    PRODUCT M*D*M-TRANSPOSE
C                                    WHERE M IS UNIT LOWER
C                                    TRIANGULAR AND D IS BLOCK
C                                    DIAGONAL WITH BLOCKS OF
C                                    ORDER 1 OR 2.
C                                    M AND D ARE WRITTEN OVER A.
C
      IF (N.LE.0) GOTO 9005
      IF (IJOB.EQ.2) GOTO 80
C
C                                  CALCULATE EQUILIBRATION
C                                    FACTORS
C
      RN = SIXTN*N
      LL = 1
      DO 10 I=1,N
         WA(I) = ZERO
         L = LL
         DO 5 J=1,N
            TEMP = ABS(A(L))
            IF (TEMP.GT.WA(I)) WA(I) = TEMP
            IF (J.LT.I) THEN
               L = L+1
            ELSE
               L = L+J
            END IF
    5    CONTINUE
         WA(I) = WA(I)*RN
         LL = LL+I
   10 CONTINUE
      I = N
      IF (I.EQ.1) GOTO 75
C
```



```
C                                  ****  BEGIN LOOP  ****
C
   15 IM1 = I-1
      IM2 = I-2
      IX = I*IM1/2
      IXI = IX+I
C
C                                  CALCULATE MAXIMUM OFF
C                                    DIAGONAL ELEMENT IN ROW I
C
      AII = ABS(A(IXI))
      SAVE = ZERO
      DO 20 L=1,IM1
         TEMP = ABS(A(IX+L))
         IF (TEMP.GT.SAVE) THEN
            SAVE = TEMP
            J = L
         END IF
   20 CONTINUE
      ICHANG = I
      IF (AII.GE.ALPHA*SAVE) GOTO 35
C
C                                  CALCULATE MAXIMUM OFF
C                                    DIAGONAL ELEMENT IN ROW J
C
      SIGMA = SAVE
      JX = J*(J-1)/2
      JXJ = JX+J
      JXT = JX+1
      AJJ = ABS(A(JXJ))
      DO 25 L=1,IM1
         TEMP = ABS(A(JXT))
         IF (TEMP.GT.SIGMA.AND.L.NE.J) SIGMA = TEMP
         IF (L.LT.J) THEN
            JXT = JXT+1
         ELSE
            JXT = JXT+L
         END IF
   25 CONTINUE
C
C                                  CHOOSE A STRATEGY FOR THE
C                                    PIVOTING
C
      IF (AII*SIGMA.GE.ALPHA*SAVE*SAVE) GOTO 35
C
      IF (AJJ.LT.ALPHA*SIGMA) GOTO 50
C
C                                  INTERCHANGE ROWS I AND J
C
      JXT = JX+1
      DO 30 L=1,IM1
         TEMP = A(IX+L)
         A(IX+L) = A(JXT)
         A(JXT) = TEMP
         IF (L.LT.J) THEN
            JXT = JXT+1
```

```
         ELSE
            JXT = JXT+L
         END IF
   30 CONTINUE
      TEMP = A(JXJ)
      A(JXJ) = A(IXI)
      A(IXI) = A(IX+J)
      A(IX+J) = TEMP
      TEMP = WA(I)
      WA(I) = WA(J)
      WA(J) = TEMP
      AII = AJJ
      ICHANG = J
C
C                                        WE USE A 1 BY 1 PIVOT
C
   35 IF (WA(I)+AII.LE.WA(I)) IER = 65
      IF (AII.EQ.ZERO) GOTO 9005
      WA(I) = ICHANG
C
      AII = A(IXI)
      IXT = IX+1
      DO 45 J=IM1,1,-1
         SAVE = -A(IX+J)/AII
         IXL = IXT-1
         IXT = IXT-J
         ISH = IX-IXT+1
         MM = MOD(J,5)
         IF (MM.NE.0) THEN
            DO 39 K=IXT,IXT+MM-1
               A(K) = A(K)+SAVE*A(ISH+K)
   39       CONTINUE
         END IF
         IXF = IXT+MM
         DO 40 K=IXF,IXL,5
            A(K) = A(K)+SAVE*A(ISH+K)
            A(K+1) = A(K+1)+SAVE*A(ISH+K+1)
            A(K+2) = A(K+2)+SAVE*A(ISH+K+2)
            A(K+3) = A(K+3)+SAVE*A(ISH+K+3)
            A(K+4) = A(K+4)+SAVE*A(ISH+K+4)
   40    CONTINUE
         A(IX+J) = SAVE
   45 CONTINUE
      I = IM1
      IF (I.GT.1) GOTO 15
      GOTO 75
C
C                                        WE USE A 2 BY 2 PIVOT
C
   50 IX1 = IX-IM1
      IF (J.EQ.IM1) GOTO 60
C
C                                        INTERCHANGE ROWS I-1 AND J
C
      JXT = JX+1
```



```
      DO 55 L=1,IM2
         TEMP = A(IX1+L)
         A(IX1+L) = A(JXT)
         A(JXT) = TEMP
         IF (L.LT.J) THEN
            JXT = JXT+1
         ELSE
            JXT = JXT+L
         END IF
   55 CONTINUE
      TEMP = A(JXJ)
      A(JXJ) = A(IX)
      A(IX) = A(IX1+J)
      A(IX1+J) = TEMP
      TEMP = A(IX+J)
      A(IX+J) = A(IX+IM1)
      A(IX+IM1) = TEMP
      TEMP = WA(IM1)
      WA(IM1) = WA(J)
      WA(J) = TEMP
C
C                                        DET MUST BE NEGATIVE
C
   60 DET = A(IXI)*A(IX)-A(IXI-1)**2
C
      TEMP = MAX(WA(I),WA(IM1))
      IF (TEMP+ABS(DET).LE.TEMP) IER = 65
      IF (DET.GE.ZERO) GOTO 9005
      WA(I) = J
      WA(IM1) = DET
C
      AIM1I = A(IXI-1)/DET
      AII = A(IXI)/DET
      AIM1 = A(IX)/DET
      IF (IM2.EQ.0) RETURN
      IXT = IX1
      DO 70 J=IM2,1,-1
         SAVE = AIM1I*A(IX1+J)-AIM1*A(IX+J)
         TEMP = AIM1I*A(IX+J)-AII*A(IX1+J)
         IXT = IXT-J
         DO 65 K=1,J
            A(IXT+K) = A(IXT+K)+A(IX+K)*SAVE+A(IX1+K)*TEMP
   65    CONTINUE
         A(IX+J) = SAVE
         A(IX1+J) = TEMP
   70 CONTINUE
      I = IM2
      IF (I.GT.1) GOTO 15
   75 IF (I.LE.0) GOTO 80
      IF (WA(1)+ABS(A(1)).LE.WA(1)) IER = 65
      IF (A(1).EQ.ZERO) GOTO 9005
      WA(1) = 1
C
C  ****  SECOND STAGE  ****        SOLVE M*D*MT*X = B WHERE
C                                    MT = M-TRANSPOSE
C
```

```
   80 IF (IJOB.EQ.1) GO TO 9000
C
      DO 120 JC=1,M
C
C                                   SOLVE M*D*Y = B AND STORE
C                                     Y IN B
C
         I = N
   85    IF (I.LE.1) GOTO 100
         IM1 = I-1
         IX = I*IM1/2
         IXI = IX+I
         ICHANG = WA(I)
         SAVE = B(ICHANG,JC)
         IF (WA(IM1).GT.ZERO) THEN
C
C                                   WE USE A 1 BY 1 PIVOT
C
            B(ICHANG,JC) = B(I,JC)
            B(I,JC) = SAVE/A(IXI)
            DO 90 J=1,IM1
               B(J,JC) = B(J,JC)+A(IX+J)*SAVE
   90       CONTINUE
            I = IM1
         ELSE
C
C                                   WE USE A 2 BY 2 PIVOT
C
            TEMP = B(I,JC)
            B(ICHANG,JC) = B(IM1,JC)
            DET = WA(IM1)
            IX1 = IX-IM1
            IM2 = I-2
            B(I,JC) = (TEMP*A(IX)-SAVE*A(IX+IM1))/DET
            B(IM1,JC) = (SAVE*A(IXI)-TEMP*A(IX+IM1))/DET
            DO 95 J=1,IM2
               B(J,JC) = B(J,JC)+A(IX+J)*TEMP+A(IX1+J)*SAVE
   95       CONTINUE
            I = IM2
         END IF
         GOTO 85
  100    IF (I.EQ.1) THEN
            B(1,JC) = B(1,JC)/A(1)
            I = 2
         ELSE
            I = 3
         END IF
C
C                                   SOLVE MT*X = Y AND STORE
C                                     X IN B
C
  105    IF (I.GT.N) GOTO 120
            II = I
            IF (WA(I).LE.ZERO) II = I+1
            IM1 = I-1
            IX = I*IM1/2
```



```
            DO 115 K=I,II
               SAVE = B(K,JC)
               DO 110 J=1,IM1
                  SAVE = SAVE+A(IX+J)*B(J,JC)
  110          CONTINUE
               B(K,JC) = SAVE
               IX = IX+K
  115       CONTINUE
            ICHANG = WA(II)
            SAVE = B(I,JC)
            B(I,JC) = B(ICHANG,JC)
            B(ICHANG,JC) = SAVE
            I = II+1
         GOTO 105
  120 CONTINUE
C
C                                   EXIT
C
 9000 IF (IER.EQ.0) RETURN
      IF (IER.EQ.65) PRINT 9065
 9065 FORMAT(/5X,'WARNING ERROR  IER=65  IN  *** LEQ1S ***'/
     *       7X,'MATRIX A IS ALGORITHMICALLY SINGULAR.'/)
      RETURN
 9005 IER = 129
      PRINT 9129
 9129 FORMAT(/5X,'TERMINAL ERROR  IER=129  IN  *** LEQ1S ***'/
     *       7X,'MATRIX A IS EXACTLY SINGULAR.'/)
      RETURN
      END
```

Литература

1. Парлетт Б. Симметричная проблема собственных значений. Численные методы. – М., Мир, 1983.
2. Bunch J.R., Kaufman L. Some stable methods for calculating inertia and solving symmetricl linear systems. – Math. Comp., 1977, v.31, No 137, p.163-179.
3. Коробов В.И. Регуляризация экстремальных собственных значений для симметричной обобщенной задачи. – Препринт ОИЯИ, Р11-87-351, Дубна, 1987.
4. Bathe K.J., Wilson E., 1976. Numerical Methods in Finite Element Analysis.- Englewood Cliffs, N.J.: Prentice-Hall.
5. Уилкинсон Дж. Алгебраическая проблема собственных значений. – М.: Наука, 1970.
6. Икрамов Х.Д. Численные методы для симметричных линейных систем. – М.: Наука, 1988.
7. Aasen J.O. On the reduction of a symmetric matrix to tridiagonal form. – BIT, 1971, v.2, No3, p.233-242.

Рукопись поступила в издательский отдел
28 декабря 1988 года.