20. COMPUTER PROGRAMS

In this section we give a typical computer program which implements one of the algorithms of Section 17 with the help of the discretization procedure stated in Section 18. The program is written in FORTRAN 77. We also show how this program can be modified to cover other cases.

We begin by documenting the program.

PROGRAM ITEIG

* PURPOSE *

COMPUTATION OF ITERATES FOR APPROXIMATING A SIMPLE EIGENVALUE AND A CORRESPONDING EIGENVECTOR OF AN INTEGRAL OPERATOR BY THE RAYLEIGH-SCHRÖDINGER SCHEME USING THE FREDHOLM METHOD(2)

* REFERENCES *

ALGORITHM 17.8 AND TABLE 19.1 ALONG WITH THE DISCRETIZATION PROCEDURE OF SECTION 18 IN THE MONOGRAPH ENTITLED SPECTRAL PERTURBATION AND APPROXIMATION WITH NUMERICAL EXPERIMENTS BY B.V. LIMAYE. THE PROGRAM WAS WRITTEN BY R.P. KULKARNI AND B.V. LIMAYE.

* PARAMETERS *

L	- THE DESIRED NUMBER OF ITERATIONS
M	- THE ORDER OF THE MATRIX TM WHICH DISCRETIZES AN INTEGRAL OPERATOR T
N	- THE ORDER OF THE MATRIX A
N1	- THE SERIAL NUMBER OF THE SELECTED EIGENVALUE OF A

* MAJOR DATA STRUCTURES *

TM	- M BY M MATRIX WHICH DISCRETIZES THE INTEGRAL OPERATOR T
A	- N BY N REAL SYMMETRIC MATRIX FOR WHICH WE INITIALLY SOLVE AN EIGENVALUE PROBLEM

D	-	N VECTOR CONTAINING EIGENVALUES OF A IN ASCENDING ORDER
Z	-	N BY N MATRIX WHOSE I-TH COLUMN CONTAINS AN EIGENVECTOR OF A CORRESPONDING TO D(I); AND HAS EUCLIDEAN NORM ONE
LAM	-	L+1 VECTOR CONTAINING THE SELECTED NONZERO SIMPLE EIGENVALUE OF A IN LAM(0) AND THE SUCCESSIVE EIGENVALUE ITERATES IN LAM(1) TO LAM(L)
U	-	AN EIGENVECTOR OF A CORRESPONDING TO LAM(0)
V	-	THE EIGENVECTOR OF THE CONJUGATE TRANSPOSE OF A, WHICH EQUALS A, CORRESPONDING TO LAM(0) AND HAVING ITS INNER PRODUCT WITH U EQUAL TO 1/LAM(0)
PH	-	M BY L+1 MATRIX CONTAINING THE INITIAL EIGENVECTOR IN THE FIRST COLUMN AND THE SUCCESSIVE EIGENVECTOR ITERATES IN THE REMAINING COLUMNS
AV	-	M BY N MATRIX WHICH TRANSFORMS CERTAIN N VECTORS ASSOCIATED WITH A INTO M VECTORS
IV	-	M BY N MATRIX WHICH TRANSFORMS N VECTORS INTO M VECTORS BY USING LINEAR INTERPOLATION OF FUNCTIONS
KM,KN, KH,KV	-	M BY M, N BY N, N BY M, M BY N MATRICES RESPECTIVELY, WHICH STORE THE WEIGHTED VALUES OF THE KERNEL OF THE INTEGRAL OPERATOR T AT VARIOUS NODES
TAH	-	N BY M MATRIX USED FOR CALCULATING EIGENVALUE ITERATES
TAHPH	-	N VECTOR DENOTING THE PRODUCT OF TAH AND A COLUMN OF PH
C	-	N+1 BY N MATRIX CONTAINING THE COEFFICIENTS OF A LINEAR SYSTEM USED FOR CALCULATING EIGENVECTOR ITERATES
ZETA	-	SCALING FACTOR FOR THE FIRST ROW OF C
BETA	-	N+1 VECTOR CONTAINING THE RIGHT HAND SIDE OF A LINEAR SYSTEM WHOSE COEFFICIENT MATRIX IS C
SUM	-	N VECTOR USED IN CALCULATING BETA

SOL - LEAST SQUARES SOLUTION OF A LINEAR SYSTEM WHOSE COEFFICIENT MATRIX IS C

ALPHA - N BY L+1 MATRIX CONTAINING LAM(0)*U IN THE FIRST COLUMN AND THE SUCCESSIVE SOLUTION VECTORS SOL IN THE REMAINING COLUMNS

AVAL - M VECTOR DENOTING THE PRODUCT OF AV AND A COLUMN OF ALPHA

PRIT - M VECTOR DENOTING A WEIGHTED SUM OF PREVIOUS EIGENVECTOR ITERATES

TMPH - M VECTOR DENOTING THE PRODUCT OF TM AND A COLUMN OF PH

RESID - MAXIMUM NORM OF THE RESIDUAL, USED IN STOPPING CRITERIA

RELIN - RELATIVE INCREMENT IN AN EIGENVECTOR ITERATE, USED IN STOPPING CRITERIA

* SUBROUTINES CALLED *

EIGRS - COMPUTES THE EIGENVALUES AND EIGENVECTORS OF A REAL SYMMETRIC MATRIX (ROUTINE IN IMSL LIBRARY EDITION 9.2, EQUIVALENT TO ROUTINE EVCSF IN IMSL MATH/LIBRARY EDITION 10.0)

LLBQF - COMPUTES THE HIGH ACCURACY SOLUTION OF A LINEAR LEAST SQUARES PROBLEM (ROUTINE IN IMSL LIBRARY, EDITION 9.2, EQUIVALENT TO ROUTINE LSBRR IN IMSL MATH/LIBRARY, EDITION 10.0)

* FUNCTIONS CALLED *

KERNEL - REAL FUNCTION WHICH YIELDS KERNEL OF THE INTEGRAL OPERATOR T

NODE - REAL FUNCTION WHICH YIELDS NODES FOR GENERATING THE MATRICES KM, KN, KH, KV AND IV

WEIGHT - REAL FUNCTION WHICH YIELDS WEIGHTS FOR GENERATING THE MATRICES KM, KN, KH AND KV

MAXNORM - REAL FUNCTION WHICH YIELDS THE MAXIMUM NORM OF A VECTOR

```
      PROGRAM   ITEIG(TAPE1,TAPE2)

      PARAMETER (L=30, M=100, N=10, N1=10)

      INTEGER   L,M,N,N1,I,J,K,JOBN,IZ,IER,IA,NN,IB,NB,IND,IX
      REAL      KM(M,M),KN(N,N),KH(N,M),KV(M,N),IV(M,N),
     1          A(N,N),D(N),Z(N,N),WK((2*N+1)*(N+3)+N),
     1          LAM(0:L),U(N),V(N),AV(M,N),PH(M,0:L),ALPHA(N,0:L),
     1          C(N+1,N),TAH(N,M),TM(M,M),
     1          TAHPH(N),SUM(N),BETA(N+1),CC(4),SOL(N),IWK(N),
     1          AVAL(M),PRIT(M),TMPH(M),X(M),Y(M),
     1          ZETA,RESID,RELIN,
     1          KERNEL,NODE,WEIGHT,MAXNORM


      WRITE(2,10)
 10   FORMAT(1H ,5X,'RAYLEIGH-SCHRODINGER SCHEME',/)
      WRITE(2,20)
 20   FORMAT(1H ,5X,'FREDHOLM METHOD(2)',/)
      WRITE(2,30)
 30   FORMAT(1H ,5X,'KERNEL:EXP(S*T)',/)
      WRITE(2,40)
 40   FORMAT(1H ,5X,'NODES:GAUSS TWO POINTS',/)
      WRITE(2,50)
 50   FORMAT(1H ,5X,'WEIGHTS:1/N',/)
      WRITE(2,60)N,N1,M
 60   FORMAT(1H ,5X,'N=',I2,3X,'N1=',I2,3X,'M=',I3,/)
      WRITE(2,70)
 70   FORMAT(1H ,5X,'PRECISION FOR STOPPING CRITERIA:1.0E-12',/)


*     GENERATION OF KM, KN, KH AND KV
        DO 110 I=1,M
           DO 120 J=1,M
              KM(I,J) = WEIGHT(J,M)*KERNEL(NODE(I,M),NODE(J,M))
 120       CONTINUE
 110    CONTINUE

        DO 130 I=1,N
           DO 140 J=1,N
              KN(I,J) = WEIGHT(J,N)*KERNEL(NODE(I,N),NODE(J,N))
 140       CONTINUE
 130    CONTINUE

        DO 150 I=1,N
           DO 160 J=1,M
              KH(I,J) = WEIGHT(J,M)*KERNEL(NODE(I,N),NODE(J,M))
 160       CONTINUE
 150    CONTINUE

        DO 170 I=1,M
           DO 180 J=1,N
              KV(I,J) = WEIGHT(J,N)*KERNEL(NODE(I,M),NODE(J,N))
 180       CONTINUE
 170    CONTINUE
```

```
*     GENERATION OF IV
        J=1
        DO 190 I=1,M
           IF (NODE(I,M).LT.NODE(1,N)) THEN
              IV(I,J) = 1.0
           ELSEIF (NODE(I,M).LT.NODE(2,N)) THEN
              IV(I,J) = (NODE(2,N)-NODE(I,M))
     1                  /(NODE(2,N)-NODE(1,N))
           ELSE
              IV(I,J) = 0.0
           ENDIF
 190    CONTINUE
        DO 200 J=2,N-1
           DO 210 I=1,M
              IF (NODE(I,M).LT.NODE(J-1,N)) THEN
                 IV(I,J) = 0.0
              ELSEIF (NODE(I,M).LT.NODE(J,N)) THEN
                 IV(I,J)=(NODE(J-1,N)-NODE(I,M))
     1                   /(NODE(J-1,N)-NODE(J,N))
              ELSEIF (NODE(I,M).LT.NODE(J+1,N)) THEN
                 IV(I,J)=(NODE(J+1,N)-NODE(I,M))
     1                   /(NODE(J+1,N)-NODE(J,N))
              ELSE
                 IV(I,J) = 0.0
              ENDIF
 210       CONTINUE
 200    CONTINUE
        J=N
        DO 220 I=1,M
           IF (NODE(I,M).LT.NODE(N-1,N)) THEN
              IV(I,J) = 0.0
           ELSEIF (NODE(I,M).LT.NODE(N,N)) THEN
              IV(I,J) = (NODE(N-1,N)-NODE(I,M))
     1                  /(NODE(N-1,N)-NODE(N,N))
           ELSE
              IV(I,J) = 1.0
           ENDIF
 220    CONTINUE


*     STEP 1(I): EIGENELEMENTS OF A

        DO 310 I=1,N
           DO 320 J=1,N
              A(I,J) = KN(I,J)
 320       CONTINUE
 310    CONTINUE

        JOBN = 12
        IZ = N
      CALL EIGRS(A,N,JOBN,D,Z,IZ,WK,IER)

        WRITE(2,330)
 330    FORMAT(1H ,5X,'EIGENVALUES OF A',/)
        WRITE(2,340)(D(I),I=1,N)
 340    FORMAT(1H ,3X,3E21.13)
```

```
      LAM(0) = D(N1)
      DO 350 I=1,N
         U(I) = Z(I,N1)
 350  CONTINUE

*     STEP 1(II): EIGENVECTOR OF CONJUGATE TRANSPOSE OF A

      DO 360 I=1,N
         V(I) = Z(I,N1)/LAM(0)
 360  CONTINUE


*     STEP 2

*     GENERATION OF AV
      DO 410 I=1,M
         DO 420 J=1,N
            AV(I,J) = 0.0
            DO 430 K=1,N
               AV(I,J) = AV(I,J)+IV(I,K)*KN(K,J)
 430        CONTINUE
 420     CONTINUE
 410  CONTINUE

*     COMPUTATION OF PH(0)
      DO 440 I=1,M
         PH(I,0) = 0.0
         DO 450 J=1,N
            PH(I,0) = PH(I,0)+AV(I,J)*U(J)
 450     CONTINUE
 440  CONTINUE

*     COMPUTATION OF ALPHA(0)
      DO 460 I=1,N
         ALPHA(I,0) = LAM(0)*U(I)
 460  CONTINUE


*     GENERATION OF C
      ZETA = 0.0
      DO 470 I=1,N
         IF (ZETA .LT. ABS(D(I)-D(N1))) THEN
             ZETA = ABS(D(I)-D(N1))
         ENDIF
 470  CONTINUE
      ZETA = ZETA*LAM(0)

      DO 480 J=1,N
         C(1,J) = ZETA*V(J)
 480  CONTINUE
      DO 490 I=2,N+1
         DO 500 J=1,N
            C(I,J) = A(I-1,J)
 500     CONTINUE
 490  CONTINUE
      DO 510 I=1,N
         C(I+1,I) = A(I,I)-LAM(0)
 510  CONTINUE
```

```
*     GENERATION OF TAH
        DO 520 I=1,N
           DO 530 J=1,M
              TAH(I,J) = KH(I,J)
 530       CONTINUE
 520    CONTINUE

*     GENERATION OF TM
        DO 540 I=1,M
           DO 550 J=1,M
              TM(I,J) = KM(I,J)
 550       CONTINUE
 540    CONTINUE

        WRITE (2,690)
 690    FORMAT (/,1H ,6X,'J',9X,'LAM(J)',10X,'RESID',5X,'RELIN')
        J = 0
        WRITE (2,700) J,LAM(J)
 700    FORMAT (/,1H ,5X,I2,2X,E19.13,2E10.2)

*     THE ITERATION STARTS

        DO 710 J=1,L

*     STEP 2(I):COMPUTATION OF J-TH EIGENVALUE ITERATE

           DO 720 I=1,N
              TAHPH(I) = 0.0
              DO 730 K=1,M
                 TAHPH(I) = TAHPH(I)+TAH(I,K)*PH(K,J-1)
 730          CONTINUE
 720       CONTINUE

           LAM(J) = 0.0
           DO 740 I=1,N
              LAM(J) = LAM(J)+TAHPH(I)*V(I)
 740       CONTINUE


*     STEP 2(II):SOLUTION OF (N+1)*N LINEAR SYSTEM

*     CALCULATION OF RIGHT HAND SIDE
           DO 810 I=1,N
              SUM(I) = 0.0
              DO 820 K=0,J-1
                 SUM(I) = SUM(I)+LAM(J-K)*ALPHA(I,K)
 820          CONTINUE
 810       CONTINUE

           BETA(1) = 0.0
           DO 830 I=1,N
              BETA(I+1) = -TAHPH(I)+SUM(I)
 830       CONTINUE
```

```
*      LEAST SQUARES SOLUTION
             IA = N+1
             NN = N+1
             IB = N+1
             NB = 1
             IND = 0
             IX = N
       CALL LLBQF(C,IA,NN,N,BETA,IB,NB,IND,CC,SOL,IX,IWK,WK,IER)
             DO 840 I=1,N
                ALPHA(I,J) = SOL(I)
 840         CONTINUE

*      STEP 2(III): COMPUTATION OF THE J-TH EIGENVECTOR ITERATE

             DO 910 I=1,M
                AVAL(I) = 0.0
                DO 920 K=1,N
                   AVAL(I) = AVAL(I)+ AV(I,K)*ALPHA(K,J)
 920            CONTINUE
 910         CONTINUE

             DO 930 I=1,M
                PRIT(I) = 0.0
                DO 940 K=1,J
                   PRIT(I) = PRIT(I)+(LAM(K-1)-LAM(K))*PH(I,J-K)
 940            CONTINUE
 930         CONTINUE

             DO 950 I=1,M
                TMPH(I) = 0.0
                DO 960 K=1,M
                   TMPH(I) = TMPH(I)+TM(I,K)*PH(K,J-1)
 960            CONTINUE
 950         CONTINUE

             DO 970 I=1,M
                PH(I,J) = (AVAL(I)+PRIT(I)+TMPH(I))/LAM(0)
 970         CONTINUE


*      CALCULATION OF RESIDUAL AND RELATIVE INCREMENT
             DO 980 I=1,M
                X(I) = TMPH(I)-LAM(J)*PH(I,J-1)
 980         CONTINUE

             RESID = MAXNORM(X,M)

             DO 990 I=1,M
                X(I) = PH(I,J)-PH(I,J-1)
                Y(I) = PH(I,J)
 990         CONTINUE

             RELIN = MAXNORM(X,M)/MAXNORM(Y,M)

             WRITE(2,700) J,LAM(J),RESID,RELIN
```

```
*     STOPPING CRITERIA
           IF (RESID.LT.1.0E-12) THEN
               WRITE(2,1000)
 1000          FORMAT(/,1H ,5X, 'RESID.LT.1.0E-12')
           ENDIF
           IF (RELIN.LT.1.0E-12) THEN
               WRITE(2,1010)
 1010          FORMAT(/,1H ,5X,'RELIN.LT.1.0E-12')
           ENDIF
           IF (RESID.LT.1.0E-12.AND.RELIN.LT.1.0E-12) THEN
               GO TO 1100
           ENDIF

 710    CONTINUE


 1100   CONTINUE
        STOP
        END


      REAL FUNCTION KERNEL(S,T)
      REAL S,T
      KERNEL = EXP(S*T)
      RETURN
      END

      REAL FUNCTION NODE(I,N)
      INTEGER  I,I1,I2,N
      I1 = I/2
      I2 = I-I1*2
      IF (I2.NE.0) THEN
         NODE = (FLOAT(I)-1.0/SQRT(3.0))/N
      ELSE
         NODE = (FLOAT(I)-1.0+1.0/SQRT(3.0))/N
      ENDIF
      RETURN
      END

      REAL FUNCTION WEIGHT(I,N)
      INTEGER  I,N
      WEIGHT = 1.0/FLOAT(N)
      RETURN
      END

      REAL FUNCTION MAXNORM(X,M)
      INTEGER M
      REAL X(M)
      MAXNORM = 0.0
      DO 1110 I=1,M
         IF (MAXNORM.LT.ABS(X(I))) THEN
            MAXNORM = ABS(X(I))
         ENDIF
 1110 CONTINUE
      RETURN
      END
```

OUTPUT OF PROGRAM ITEIG

RAYLEIGH-SCHRODINGER SCHEME

FREDHOLM METHOD(2)

KERNEL:EXP(S*T)

NODES:GAUSS TWO POINTS

WEIGHTS:1/N

N=10 N1=10 M=100

PRECISION FOR STOPPING CRITERIA:1.0E-12

EIGENVALUES OF A

-.1133820135241E-14	.1047006402339E-14	.2699346339808E-12
.4796177191072E-10	.9238598296043E-08	.1050078986292E-05
.7441265877077E-04	.3552405730829E-02	.1059756286955E+00
.1353028494291E+01		

J	LAM(J)	RESID	RELIN
0	.1353028494291E+01		
1	.1352614455737E+01	.18E-01	.23E-01
2	.1353030065281E+01	.16E-03	.23E-03
3	.1353030261682E+01	.39E-05	.50E-05
4	.1353030164665E+01	.92E-07	.13E-06
5	.1353030164536E+01	.15E-08	.20E-08
6	.1353030164578E+01	.60E-10	.86E-10
7	.1353030164578E+01	.69E-12	.84E-12

RESID.LT.1.0E-12

RELIN.LT.1.0E-12

Computation of actual accuracy

As we have seen in Section 18, the computed eigenvalue iterates λ_j = LAM(J) will converge, under suitable conditions, to the simple eigenvalue $\lambda^{(M)}$ of the matrix [TM] which is nearest to λ_0 = LAM(0) , and the computed eigenvector iterates $\underset{\sim}{c}_j$ will converge to the corresponding eigenvector $\underset{\sim}{c}^{(M)}$ of [TM] which satisfies

$$\langle [TAH]\underset{\sim}{c}^{(M)}, V\rangle = \lambda^{(M)} .$$

We consider some additions to the program ITEIG which allow us to find the actual accuracy reached at each iterate by computing $\lambda^{(M)}$, $\underset{\sim}{c}^{(M)}$, $\lambda^{(M)} - \lambda_j$ and the maximum norm of $\underset{\sim}{c}^{(M)} - \underset{\sim}{c}_j$, $j = 0,1,\ldots,L$. This is done only for illustrative purposes. The whole point of PROGRAM ITEIG is to avoid calculating $\lambda^{(M)}$ and $\underset{\sim}{c}^{(M)}$.

```
* MAJOR DATA STRUCTURES *
M1          - THE SERIAL NUMBER OF THE EIGENVALUE OF TM NEAREST TO LAM(0)
DD          - M VECTOR CONTAINING EIGENVALUES OF TM
ZZ          - M BY M MATRIX WHOSE I-TH COLUMN CONTAINS AN EIGENVECTOR OF
              TM CORRESPONDING TO DD(I)
TAHZZ       - N VECTOR DENOTING THE PRODUCT OF TAH AND THE M1-TH COLUMN
              OF ZZ AND HAS EUCLIDEAN NORM ONE
SCP         - THE SCALAR PRODUCT OF TAHZZ AND V
PHI         - THE EIGENVECTOR OF TM CORRESPONDING TO DD(M1) WHOSE INNER
              PRODUCT WITH V EQUALS DD(M1)
```

We declare in the beginning of PROGRAM ITEIG

```
INTEGER   M1
REAL      DD(M), ZZ(M,M), WWK(M+M*(M+1)/2), TAHZZ(N), SCP, PHI(M)
```

and add the following lines at places indicated by the statement numbers; the WRITE statements and their formats 690 and 700 are also changed.

```
*     ADDENDUM TO PROGRAM ITEIG

*     EIGENELEMENTS OF TM FOR COMPARISON
        JOBN = 12
        IZ = M
      CALL EIGRS (TM,M,JOBN,DD,ZZ,IZ,WWK,IER)

*     EIGENVALUE OF TM NEAREST TO LAM(0)
        DO 615 I=M,1,-1
           IF (DD(I).LE.LAM(0)) THEN
              M1 = I
              IF ( M1.EQ.M) THEN
                 GO TO 635
              ELSE
                 GO TO 625
              ENDIF
           ENDIF
 615    CONTINUE
        M1 = 1
 625    IF (ABS(LAM(0)-DD(M1)).GT. ABS(LAM(0)-DD(M1+1))) THEN
           M1 = M1+1
        ENDIF
 635    CONTINUE

        WRITE (2,645) M1,DD(M1)
 645    FORMAT (/,1H ,5X,'M1=',I3,5X,'LAM =',E19.13,/,/)

        DO 655 I=1,N
           TAHZZ(I) = 0.0
           DO 665 J=1,M
              TAHZZ(I) = TAHZZ(I)+TAH(I,J)*ZZ(J,M1)
 665       CONTINUE
 655    CONTINUE

        SCP = 0.0
        DO 675 I=1,N
           SCP = SCP+TAHZZ(I)*V(I)
 675    CONTINUE

        DO 685 I=1,M
           PHI(I) = ZZ(I,M1)/SCP*DD(M1)
           X(I) = PHI(I)-PH(I,0)
 685    CONTINUE

        WRITE (2,690)
 690    FORMAT (1H ,6X,'J',9X,'LAM(J)',8X,'LAM-LAM(J)',1X,
     1          'PH-PH(J)',3X,'RESID',5X,'RELIN')
        J = 0
        WRITE (2,700)J,LAM(J),DD(M1)-LAM(J),MAXNORM(X,M)
 700    FORMAT (/,1H ,5X,I2,2X,E19.13,4E10.2)


           DO 995 I=1,M
              X(I) = PHI(I)-PH(I,J)
 995       CONTINUE

           WRITE(2,700) J,LAM(J),DD(M1)-LAM(J),MAXNORM(X,M),
     1                  RESID,RELIN
```

OUTPUT OF PROGRAM ITEIG WITH THE ADDENDUM

RAYLEIGH-SCHRODINGER SCHEME

FREDHOLM METHOD(2)

KERNEL:EXP(S*T)

NODES:GAUSS TWO POINTS

WEIGHTS:1/N

N=10 N1=10 M=100

PRECISION FOR STOPPING CRITERIA:1.0**-12

EIGENVALUES OF A

-.1133820135241E-14	.1047006402339E-14	.2699346339808E-12
.4796177191072E-10	.9238598296043E-08	.1050078986292E-05
.7441265877077E-04	.3552405730829E-02	.1059756286955E+00
.1353028494291E+01		

M1=100 LAM = .1353030164578E+01

J	LAM(J)	LAM-LAM(J)	PH-PH(J)	RESID	RELIN
0	.1353028494291E+01	.17E-05	.13E-01		
1	.1352614455737E+01	.42E-03	.13E-03	.18E-01	.23E-01
2	.1353030065281E+01	.99E-07	.29E-05	.16E-03	.23E-03
3	.1353030261682E+01	-.97E-07	.73E-07	.39E-05	.50E-05
4	.1353030164665E+01	-.86E-10	.11E-08	.92E-07	.13E-06
5	.1353030164536E+01	.42E-10	.48E-10	.15E-08	.20E-08
6	.1353030164578E+01	.50E-13	.52E-12	.60E-10	.86E-10
7	.1353030164578E+01	-.28E-13	.75E-13	.69E-12	.84E-12

RESID.LT.1.0E-12

RELIN.LT.1.0E-12

Modifications of the program ITEIG

We discuss how PROGRAM ITEIG can be easily adapted to deal with a large number of different situations.

(i) Parameter values

By simply assigning different values to the parameters in the second line of the program, one can alter the maximum number L of the iterations, the size M of the grid which discretizes the integral operator T , the order N of the matrix A in the initial eigenvalue problem, and the serial number N1 of the selected eigenvalue of A with which we start the iteration process.

(ii) Various iteration schemes

Instead of the Rayleigh-Schrödinger scheme (11.18) used in PROGRAM ITEIG, we can use the fixed point scheme (11.19), the modified fixed point scheme (11.31), or the Ahués scheme (11.35). The algorithms 17.9, 17.10 and 17.11 indicate the required changes in the program ITEIG for implementing these schemes. There is no need for the vector PRIT in these schemes. Hence the DO loops 930 and 940 can be dropped altogether. In fact, there is no need for the double arrays ALPHA(N,0:L) and PH(M,0:L); instead, single arrays ALPHA(N), PH(M) and PRPH(M) (representing the current solution of the linear system, the current eigenvector iterate and the previous eigenvector iterate, respectively) will suffice.

Fixed point scheme: Change the DO loops 820 and 970 as follows:

```
      DO 820 K = 0,J-1
         SUM(I) = SUM(I)+LAM(J)*ALPHA(I,K)
820   CONTINUE
```

and

```
      DO 970 I = 1, M
         PH(I,J) = (AVAL(I)+(LAM(0)-LAM(J))*PH(I,J-1)+TMPH(I))/LAM(0)
970   CONTINUE
```

Modified fixed point scheme: Declare

```
     REAL        T2AH(N,M),T2M(M,M),T2AHPH(N),MU(L),T2MPH(M)
```

Add the following comments and statements after the nested DO loops 540 and 550 :

```
*     GENERATION OF T2AH
      DO 555 I = 1,N
         DO 565 J = 1,M
            T2AH(I,J) = 0.0
            DO 575 K = 1,M
               T2AH(I,J) = T2AH(I,J)+TAH(I,K)*TM(K,J)
575         CONTINUE
565      CONTINUE
555   CONTINUE

*     GENERATION OF T2M
      DO 585 I = 1,M
         DO 595 J = 1,M
            T2M(I,J) = 0.0
            DO 605 K = 1,M
               T2M(I,J) = T2M(I,J)+TM(I,K)*TM(K,J)
605         CONTINUE
595      CONTINUE
585   CONTINUE
```

Add the following lines after statement 740:

```
      DO 745 I = 1,N
         T2AHPH(I) = 0.0
         DO 755 K = 1,M
            T2AHPH(I) = T2AHPH(I)+T2AH(I,K)*PH(K,J-1)
755      CONTINUE
745   CONTINUE

      MU(J) = 0.0
      DO 765 I = 1,N
         MU(J) = MU(J)+T2AHPH(I)*V(I)
765   CONTINUE

      DO 775 I = 1,M
         T2MPH(I) = 0.0

         DO 785 K =1,M
            T2MPH(I) = T2MPH(I)+T2M(I,K)*PH(K,J-1)
785      CONTINUE
775   CONTINUE
```

Delete the DO loops 810 and 820.

Change the DO loops 830 and 970 as follows:

```
      DO 830 I = 1,N
         BETA(I+1) = (-T2AHPH(I)+MU(J)/LAM(J)*TAHPH(I))/LAM(J)
830   CONTINUE
```

and

```
      DO 970 I = 1,M
         PH(I,J) = (LAM(J)*AVAL(I)+(LAM(0)-MU(J)/LAM(J))*TMPH(I)
     1                   + T2MPH(I))/(LAM(0)*LAM(J))
970   CONTINUE
```

Ahués scheme: Same additions and deletions as in the case of the modified fixed point scheme. Also, change the DO loops 830 and 970 as follows:

```
          DO 830 I = 1,N
             BETA(I+1) = (-T2AHPH(I)+LAM(J)*TAHPH(I)
    1                   +LAM(0)*(MU(J)-LAM(J)*LAM(J))*U(I))/LAM(J)
 830      CONTINUE
```

and

```
          DO 970 I = 1,M
             PH(I,J) = (LAM(J)*AVAL(I)+(LAM(J)*LAM(J)-MU(J))*PH(I,0)
    1                 +(LAM(0)-LAM(J))*TMPH(I)
    1                 +T2MPH(I))/(LAM(0)*LAM(J))
 970      CONTINUE
```

(iii) Various methods

By altering, if necessary, the matrices A, AV and TAH appearing in the nested DO loops 310-320,410-430 and 520-530, respectively, one can employ any of the following methods: Projection, Sloan, Galerkin(1) and (2), Nyström, Fredholm(1). The required alterations can be quickly found from Table 19.1. For example, to employ the Nyström method we need only alter the matrix AV; for this purpose we replace the nested DO loops 410-430 by

```
          DO 410 I = 1,M
             DO 420 J = 1,N
                AV(I,J) = KV(I,J)
 420         CONTINUE
 410      CONTINUE
```

With these changes, the program will work provided the matrix A is real and symmetric. If it is not, further changes are necessary. They are outlined later.

(iv) Kernel, nodes and weights

By changing the definitions of KERNEL, NODE and WEIGHT in the function subprograms given at the end of PROGRAM ITEIG, we can vary the kernel of the integral operator T as well as the nodes and the weights used in the quadrature formula which discretizes T. With these changes, the program will work if the matrix A remains real and symmetric. Otherwise further changes are required, as detailed below.

(v) General complex matrix A

The matrix A appearing in the DO loops 310-320 is real and symmetric, and it remains so for the Fredholm and the Nyström methods as long as the kernel is real and symmetric and the weights are all real and equal. When A is not real and symmetric, make the following changes

1. COMPLEX (instead of REAL) declarations of appropriate arguments; use of the FORTRAN 77 intrinsic function CONJG which yields the conjugate of a complex number.

2. Instead of the routine EIGRS of the IMSL LIBRARY, Edition 9.2 (or its equivalent EVCSF of the IMSL MATH/LIBRARY, Edition 10.0), the following IMSL routines need to be called in appropriate cases.

A	Edition 9.2	Edition 10.0
Complex Hermitian	EIGCH	EVCHF
Real general	EIGRF	EVCRG
Complex general	EIGCC	EVCCG

A set of Library interface routines is available to link the routines in the old and the new editions. The routines in Edition 9.2 treat a complex matrix of order N as a real vector of length $2N^2$; an appropriate equivalence statement may be required when an array is of one type in the calling program but of another type in the subroutine.

For the routines in Edition 10.0, the eigenvalues appear in a complex N vector EVAL in increasing lexicographic order and the I-th column of a complex N by N matrix EVEC gives an eigenvector corresponding to EVAL(I); each eigenvector U is normalized such that

$$\max\{|\mathrm{Re}\ U(1)|+|\mathrm{Im}\ U(1)|,\ldots\ldots,|\mathrm{Re}\ U(N)|+|\mathrm{Im}\ U(N)|\} = 1 .$$

We then pick a simple nonzero eigenvalue LAM(0) of A and a corresponding eigenvector U according to our choice.

3. Let ACT denote the conjugate transpose of the matrix A . If A is normal (i.e., ACT commutes with A) , then U itself is an eigenvector of ACT corresponding to CONJG(LAM(0)). Hence in this case we simply need to replace LAM(0) by CONJG(LAM(0)) in the DO loop 360. If A is Hermitian, then ACT = A and CONJG(LAM(0)) = LAM(0) , and there is no change in the DO loop 360.

For a general (real or complex) matrix A , we generate ACT as follows:

```
*     GENERATION OF ACT
         DO 360 I = 1,N
            DO 370 J = 1,N
               ACT(I,J) = CONJG(A(J,I))
 370        CONTINUE
 360     CONTINUE
```

We can then solve the eigenvalue problem for ACT just as we do for A . Let CONJG(LAM(0)) be the N2-th entry of the vector D or EVAL, so that an eigenvector of ACT corresponding to CONJG(LAM(0)) appears in the N2-th column of the matrix Z or EVEC. To obtain an eigenvector V of ACT whose inner product with U is 1/LAM(0), we proceed as follows. The complex argument SP denotes 'scalar product'.

```
      COMPLEX  SP
        SP = 0.0
        DO 380 I = 1,N
           SP = SP+U(I)*CONJG(Z(I,N2))
 380    CONTINUE
        DO 390 I = 1,N
           V(I) = Z(I,N2)/CONJG(SP*LAM(0))
 390    CONTINUE
```

Alternatively, we can find V as the least squares solution of a linear system with its coefficient matrix CBAR and right hand side BETABAR, defined as follows:

```
      COMPLEX  CBAR(N+1,N), BETABAR(N+1)
*     GENERATION OF CBAR
        DO 360 J = 1,N
           CBAR(1,J) = CONJG(U(J))
 360    CONTINUE
        DO 370 I = 2, N+1
           DO 380 J = 1,N
              CBAR(I,J) = ACT(I-1,J)
 380       CONTINUE
 370    CONTINUE
        DO 390 I = 1,N
           CBAR(I+1,I) = ACT(I,I)-CONJG(LAM(0))
 390    CONTINUE
        BETABAR(1) = 1/CONJG(LAM(0))
        DO 400 I = 1,N
           BETABAR(I+1) = 0.0
 400    CONTINUE
```

If ACT is a real matrix, the IMSL subroutine LLBQF of Edition 9.2 or LSBRR of Edition 10.0 can be used for the solution of the above least squares problem. The LINPACK routines SQRDC and SQRSL also give the solution of a least squares problem with a real coefficient matrix. Their complex analogues CQRDC and CQRSL are available.

Since V is, in general, a complex array, and since the inner product is conjugate linear in the second variable, we change V(I) to CONJG(V(I)) in the DO loop 740 of PROGRAM ITEIG which gives LAM(J) and in the DO loop 765 of its modification which gives MU(J).

4. If the functions KERNEL and WEIGHT are real-valued, LAM(0) is real and the entries of U and V are real, then the coefficient matrix C and the right hand side vector BETA are real. We can then continue to use the IMSL routine LLBQF or LSBRR for obtaining the least squares solution ALPHA in the DO loop 840. Otherwise, LINPACK routines CQRDC and CQRSL can be employed to handle the complex case.

Unless A is normal and U has Euclidean norm 1, the scaling factor ZETA for the first row of C may be inappropriate (Cf. (18.15) and (18.17)). Hence the DO loop 470 may be dropped and the DO loop 480 be changed as follows:

```
      DO 480 J = 1,N
         C(1,J) = CONJG(V(J))
  480 CONTINUE
```

We describe an alternative method for obtaining the solution SOL in the DO loop 840. It is based on our discussion of (18.18) and (18.20).

Instead of generating the matrix C in the DO loops 470 to 510, we generate an N by N matrix B as follows.

```
      COMPLEX B(N,N)
*     GENERATION OF B
        DO 470 I = 1,N
           DO 480 J = 1,N
              B(I,J) = A(I,J)-LAM(0)*LAM(0)*U(I)*CONJG(V(J))
480        CONTINUE
        CONTINUE
        DO 490 I = 1,N
           B(I,I) = B(I,I)-LAM(0)
490     CONTINUE
```

Then SOL can be obtained as the solution of the linear system with coefficient matrix B and right hand side BETA(I+1),I=1,...,N . The following IMSL routines can be used to compute this solution.

B	Edition 9.2	Edition 10.0
Real symmetric	LEQ2S	LSASF
Complex Hermitian	-	LSAHF
Real general	LEQT2F	LSARG
Complex general	LEQ2C	LSACG