Algorithms in the Nashlib set in various programming languages – Part 5

John C Nash, retired professor, University of Ottawa Peter Olsen, retired ??

11/01/2021

Contents

Abstract

Algorithms 11 and 12 and 26 from the book Nash (1979) are implemented in a variety of programming languages including Fortran, BASIC, Pascal, Python and R. These concern the eigensolutions of a general complex square matrix and possible extensions.

Overview of this document

This section is repeated for each of the parts of Nashlib documentation.

A companion document **Overview of Nashlib and its Implementations** describes the process and computing environments for the implementation of Nashlib algorithms. This document gives comments and/or details relating to implementations of the algorithms themselves.

Note that some discussion of the reasoning behind certain choices in algorithms or implementations are given in the Overview document.

Algorithm 26 – Complex matrix eigensolutions

Fortran

Listing

```
C     MASTER COMDRIVE
      IMPLICIT REAL*8(A-H,O-Z)
      INTEGER UPP
      INTEGER CLKTC1, CLKTC2
      DIMENSION AR(10,10),AI(10,10),ASR(10,10),ASI(10,10),WR(10),WI(10)
      DIMENSION ZR(10,10),ZI(10,10),SCALE(10),INT(10)
      NIN = 5
      NOUT = 6
    1 READ (NIN,901) N
      IF (N.LE.0) STOP
      WRITE(NOUT, 900)N
 900  FORMAT(' COMPLEX MATRIX OF ORDER ',I4,' REAL FRANK, IMAG MOLER')
      DO 110 I=1,N
         DO 105 J=1,N
            AR(I,J) = MIN(I,J)
            AI(I,J) = MIN(I,J) - 2.0
  105    CONTINUE
         AR(I,I)=I
         AI(I,I)=I
  110  CONTINUE
C   START
      RGRD=0.0
      ISS=0
      JSS=0
      WRITE (NOUT,903) N
      WRITE (NOUT,904)
      do 103 i=1,n
      WRITE (NOUT,905) (AR(I,J),J=1,N)
 103  continue
      WRITE (NOUT,906)
      do 104 i=1,n
      WRITE (NOUT,905) (AI(I,J),J=1,N)
 104  continue
      DO 25 I=1,N
      DO 20 J=1,N
      ASR(I,J)=AR(I,J)
      ASI(I,J)=AI(I,J)
 20   CONTINUE
 25   CONTINUE
      RADX=16.0
c      CALL TIMER(CLKTC1)
c   20210128 -- SUPPRESS BALANCING FOR NOW
C      CALL CBAL(10,N,RADX,AR,AI,LOW,LUP,SCALE)
      CALL COMEIG(N,10,AR,AI,ZR,ZI,WR,WI)
C      CALL CBABK2(10,N,LOW,LUP,SCALE,N,ZR,ZI)
c      CALL TIMER(CLKTC2)
C      CLKTC2=CLKTC2-CLKTC1
      DO 6 J=1,N
```

```
      IT=1
      BIG=ZR(1,J)**2+ZI(1,J)**2
      IF (N.LT.2) GO TO 4
      DO 3 I=2,N
      U=ZR(I,J)**2+ZI(I,J)**2
      IF (U.LE.BIG) GO TO 3
      BIG=U
      IT=I
    3 CONTINUE
    4 U=ZR(IT,J)/BIG
      V=-ZI(IT,J)/BIG
      DO 5 I=1,N
      BIG=ZR(I,J)*U-ZI(I,J)*V
      ZI(I,J)=ZI(I,J)*U+ZR(I,J)*V
      ZR(I,J)=BIG
    5 CONTINUE
    6 CONTINUE
      DO 9 J=1,N
      WRITE (NOUT,907) J,WR(J),WI(J)
      WRITE (NOUT,908) (ZR(I,J),ZI(I,J),I=1,N)
      WRITE (NOUT,909)
      AL=WR(J)
      GA=WI(J)
      DO 8 I=1,N
      U=0.0
      V=0.0
      DO 7 K=1,N
      U=U+ASR(I,K)*ZR(K,J)-ASI(I,K)*ZI(K,J)
      V=V+ASR(I,K)*ZI(K,J)+ASI(I,K)*ZR(K,J)
    7 CONTINUE
      U=U-AL*ZR(I,J)+GA*ZI(I,J)
      V=V-GA*ZR(I,J)-AL*ZI(I,J)
      TEM=DSQRT(U**2+V**2)
      IF (TEM.LE.RGRD) GO TO 8
      RGRD=TEM
      ISS=I
      JSS=J
      WRITE (NOUT,908) U,V
    8 CONTINUE
    9 CONTINUE
      WRITE (NOUT,910) RGRD,JSS,ISS
C      WRITE (NOUT,912) CLKTC2
      GO TO 1
  901 FORMAT (I4)
  902 FORMAT (8F10.5)
  903 FORMAT (' ORDER OF MATRIX',I4)
  904 FORMAT (' REAL PART OF MATRIX')
  905 FORMAT (' ',1P5D16.8)
  906 FORMAT (' IMAGINARY PART OF MATRIX')
  907 FORMAT (' EIGENVALUE ',I4,' = ',1PD16.8,' + I*',1PD16.8)
  908 FORMAT (' ',2D16.8)
  909 FORMAT (' RESIDUALS,REAL AND IMAGINARY')
  910 FORMAT (' MAXIMUM RESIDUAL MAGNITUDE=',1PD16.8,' SOLUTION',I4,
```

```
     #' ELEMENT',I4)
C  912 FORMAT ('  TIME FOR EIGENSOLUTION =',I9,' * 0.01 SECS')
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C     include 'comeig.for'
      SUBROUTINE COMEIG(N,ND,A,Z,T,U,RR,RI)
      IMPLICIT REAL*8(A-H,O-Z)
      LOGICAL MARK
C     SOLVES COMPLEX EIGENPROBLEM FOR A+I*Z
C     VECTORS (RIGHT HAND) RETURNED IN T+I*U
C     EIGENVALUES IN DECREASING ORDER OF MAGNITUDE DOWN DIAGONALS OF A
C     AND Z ARE RESTORED IN ER AND EI ON OUTPUT
C     EPS IS MACHINE DEPENDENT TOLERANCE
C     ITERATION LIMIT IS 35
      DIMENSION A(ND,N),Z(ND,N),T(ND,N),U(ND,N),RR(N),RI(N)
      EQUIVALENCE (AKI,AIK,TIK),(AIM,AMI,TIM)
      EQUIVALENCE (ZKI,ZIK,UIK),(ZMI,ZIM,UIM)
      IF (N.LE.1) GO TO 24
C     SET TOLERANCES
      CALL ENVROD(IB,IT,IR)
      EPS=(1.0D0*IB)**(1-IT)
      MARK=.FALSE.
C     PUT IDENTITY MATRIX IN T AND ZERO IN U
      N1=N-1
      DO 2 I=1,N1
      T(I,I)=1.0D0
      U(I,I)=0.0D0
      I1=I+1
      DO 1 J=I1,N
      T(I,J)=0.0D0
      U(I,J)=0.0D0
      U(J,I)=0.0D0
      T(J,I)=0.0D0
    1 CONTINUE
    2 CONTINUE
      T(N,N)=1.0D0
      U(N,N)=0.0D0
C     SAFETY LOOP
      DO 23 IT=1,35
      IF (MARK) GO TO 24
C     CONVERGENCE CRITERIA
      TAU=0.0D0
      DO 4 K=1,N
      TEM=0.0D0
      DO 3 I=1,N
      IF (I.NE.K) TEM=DABS(A(I,K))+DABS(Z(I,K))+TEM
    3 CONTINUE
      TAU=TAU+TEM
      RR(K)=TEM+DABS(A(K,K))+DABS(Z(K,K))
    4 CONTINUE
      WRITE (NOUT,901) TAU,IT
C     INTERCHANGE COLUMNS AND ROWS
      DO 8 K=1,N1
```

```
      SMAX=RR(K)
      I=K
      K1=K+1
      DO 5 J=K1,N
      IF (SMAX.GE.RR(J)) GO TO 5
      SMAX=RR(J)
      I=J
    5 CONTINUE
      IF (I.EQ.K) GO TO 8
      RR(I)=RR(K)
      DO 6 J=1,N
      TEP=A(K,J)
      A(K,J)=A(I,J)
      A(I,J)=TEP
      TEP=Z(K,J)
      Z(K,J)=Z(I,J)
      Z(I,J)=TEP
    6 CONTINUE
      DO 7 J=1,N
      TEP=A(J,K)
      A(J,K)=A(J,I)
      A(J,I)=TEP
      TEP=Z(J,K)
      Z(J,K)=Z(J,I)
      Z(J,I)=TEP
      TEP=T(J,K)
      T(J,K)=T(J,I)
      T(J,I)=TEP
      TEP=U(J,K)
      U(J,K)=U(J,I)
      U(J,I)=TEP
    7 CONTINUE
    8 CONTINUE
      IF (TAU.LT.(100.0DO*EPS)) GO TO 24
C     BEGIN SWEEP
      MARK=.TRUE.
      DO 22 K=1,N1
      K1=K+1
      DO 21 M=K1,N
      HJ=0.0DO
      HR=0.0DO
      HI=0.0DO
      G=0.0DO
      DO 9 I=1,N
      IF (I.EQ.K.OR.I.EQ.M) GO TO 9
      HR=HR+A(K,I)*A(M,I)+Z(K,I)*Z(M,I)-A(I,K)*A(I,M)-Z(I,K)*Z(I,M)
      HI=HI+Z(K,I)*A(M,I)-A(K,I)*Z(M,I)-A(I,K)*Z(I,M)+Z(I,K)*A(I,M)
      TE=A(I,K)**2+Z(I,K)**2+A(M,I)**2+Z(M,I)**2
      TEE=A(I,M)**2+Z(I,M)**2+A(K,I)**2+Z(K,I)**2
      G=G+TE+TEE
      HJ=HJ-TE+TEE
    9 CONTINUE
      BR=A(K,M)+A(M,K)
```

```
      BI=Z(K,M)+Z(M,K)
      ER=A(K,M)-A(M,K)
      EI=Z(K,M)-Z(M,K)
      DR=A(K,K)-A(M,M)
      DI=Z(K,K)-Z(M,M)
      TE=BR**2+EI**2+DR**2
      TEE=BI**2+ER**2+DI**2
      IF (TE.LT.TEE) GO TO 10
      SISW=1.0D0
      C=BR
      S=EI
      D=DR
      DE=DI
      ROOT2=DSQRT(TE)
      GO TO 11
   10 SISW=-1.0D0
      C=BI
      S=-ER
      D=DI
      DE=DR
      ROOT2=DSQRT(TEE)
   11 ROOT1=DSQRT(S*S+C*C)
      SIG=DSIGN(1.0D0,D)
      SA=0.0D0
      CA=DSIGN(1.0D0,C)
      IF (ROOT1.GE.EPS) GO TO 14
      SX=0.0D0
      SA=0.0D0
      CX=1.0D0
      CA=1.0D0
      IF (SISW.GT.0.0D0) GO TO 12
      E=EI
      B=-BR
      GO TO 13
   12 E=ER
      B=BI
   13 SND=D**2+DE**2
      GO TO 16
   14 IF (DABS(S).LE.EPS) GO TO 15
      CA=C/ROOT1
      SA=S/ROOT1
   15 COT2X=D/ROOT1
      COTX=COT2X+(SIG*DSQRT(1.0D0+COT2X**2))
      SX=SIG/DSQRT(1.0D0+COTX**2)
      CX=SX*COTX
C     FIND ROTATED ELEMENTS
      ETA=(ER*BR+BI*EI)/ROOT1
      TSE=(BR*BI-ER*EI)/ROOT1
      TE=SIG*(-ROOT1*DE+TSE*D)/ROOT2
      TEE=(D*DE+ROOT1*TSE)/ROOT2
      SND=ROOT2**2+TEE**2
      TEE=HJ*CX*SX
      COS2A=CA**2-SA**2
```

```
      SIN2A=2.0DO*CA*SA
      TEM=HR*COS2A+HI*SIN2A
      TEP=HI*COS2A-HR*SIN2A
      HR=CX*CX*HR-SX*SX*TEM-CA*TEE
      HI=CX*CX*HI+SX*SX*TEP-SA*TEE
      B=SISW*TE*CA+ETA*SA
      E=CA*ETA-SISW*TE*SA
   16 S=HR-SIG*ROOT2*E
      C=HI-SIG*ROOT2*B
      ROOT=DSQRT(C*C+S*S)
      IF (ROOT.GE.EPS) GO TO 17
      CB=1.0DO
      CH=1.0DO
      SB=0.0DO
      SH=0.0DO
      GO TO 18
   17 CB=-C/ROOT
      SB=S/ROOT
      TEE=CB*B-E*SB
      SNC=TEE*TEE
      TANH=ROOT/(G+2.0DO*(SNC+SND))
      CH=1.0DO/DSQRT(1.0DO-TANH**2)
      SH=CH*TANH
C     PREPARE FOR TRANSFORMATION
   18 TEM=SX*SH*(SA*CB-SB*CA)
      C1R=CX*CH-TEM
      C2R=CX*CH+TEM
      C1I=-SX*SH*(CA*CB+SA*SB)
      C2I=C1I
      TEP=SX*CH*CA
      TEM=CX*SH*SB
      S1R=TEP-TEM
      S2R=-TEP-TEM
      TEP=SX*CH*SA
      TEM=CX*SH*CB
      S1I=TEP+TEM
      S2I=TEP-TEM
C     DECIDE WHETHER TO MAKE TRANSFORMATION
      TEM=DSQRT(S1R**2+S1I**2)
      TEP=DSQRT(S2R**2+S2I**2)
      IF (TEM.LE.EPS.AND.TEP.LE.EPS) GO TO 21
      MARK=.FALSE.
C     TRANSFORMATION ON LEFT
      DO 19 I=1,N
      AKI=A(K,I)
      AMI=A(M,I)
      ZKI=Z(K,I)
      ZMI=Z(M,I)
      A(K,I)=C1R*AKI-C1I*ZKI+S1R*AMI-S1I*ZMI
      Z(K,I)=C1R*ZKI+C1I*AKI+S1R*ZMI+S1I*AMI
      A(M,I)=S2R*AKI-S2I*ZKI+C2R*AMI-C2I*ZMI
      Z(M,I)=S2R*ZKI+S2I*AKI+C2R*ZMI+C2I*AMI
   19 CONTINUE
```

```
C     TRANSFORMATION ON RIGHT
      DO 20 I=1,N
      AKI=A(I,K)
      AMI=A(I,M)
      ZKI=Z(I,K)
      ZMI=Z(I,M)
      A(I,K)=C2R*AKI-C2I*ZKI-S2R*AMI+S2I*ZMI
      Z(I,K)=C2R*ZKI+C2I*AKI-S2R*ZMI-S2I*AMI
      A(I,M)=-S1R*AKI+S1I*ZKI+C1R*AMI-C1I*ZMI
      Z(I,M)=-S1R*ZKI-S1I*AKI+C1R*ZMI+C1I*AMI
      AKI=T(I,K)
      AMI=T(I,M)
      ZKI=U(I,K)
      ZMI=U(I,M)
      T(I,K)=C2R*AKI-C2I*ZKI-S2R*AMI+S2I*ZMI
      U(I,K)=C2R*ZKI+C2I*AKI-S2R*ZMI-S2I*AMI
      T(I,M)=-S1R*AKI+S1I*ZKI+C1R*AMI-C1I*ZMI
      U(I,M)=-S1R*ZKI-S1I*AKI+C1R*ZMI+C1I*AMI
   20 CONTINUE
   21 CONTINUE
   22 CONTINUE
   23 CONTINUE
   24 DO 25 I=1,N
      RR(I)=A(I,I)
      RI(I)=Z(I,I)
   25 CONTINUE
      RETURN
  901 FORMAT (' TAU=',D20.10,' AT ITERATION ',I4)
      END
      SUBROUTINE ENVROD(BETA,T,RND)
C     DOUBLE PRECISION MACHINE ENVIRONMENT
C     NO INPUT
C     PARAMETERS:
C        BETA  - MACHINE RADIX
C        T     - NUMBER OF RADIX DIGITS IN WORD (DOUBLE)
C        RND   - SET TO 1 IF MACHINE ROUNDS
C                SET TO 0 IF MACHINE CHOPS
C     ALL PARAMETERS ARE INTEGERS
C     MACHINE DOUBLE PRECISION I.E. SMALLEST POSITIVE NUMBER SUCH
C     THAT 1. + NUMBER .GT. 1. , IS DBLE(FLOAT(BETA))**(1-T)
      INTEGER BETA,T,RND
      DOUBLE PRECISION A,B,DBLE
      RND=1
      A=2.0D0
      B=2.0D0
    1 IF ((A+1.0D0)-A.NE.1.0D0) GO TO 2
      A=2.0D0*A
      GO TO 1
    2 IF (A+B.NE.A) GO TO 3
      B=2.0D0*B
      GO TO 2
    3 BETA=(A+B)-A
      IF (A+DBLE(FLOAT(BETA-1)).EQ.A) RND=0
```

```
      IF (A+DBLE(FLOAT(BETA-1)).EQ.A) RND=0
      T=0
      A=1.0D0
    4 T=T+1
      A=A*DBLE(FLOAT(BETA))
      IF ((A+1.0D0)-A.EQ.1.0D0) GO TO 4
      RETURN
      END
C      include 'cbal.for'
C
C     ------------------------------------------------------------------
C
      SUBROUTINE CBAL (NM,N,RADIX,AR,AI,LOW,UPP,SCALE)
      IMPLICIT REAL*8 (A-H,O-Z)
C
C
      REAL*8 AR(NM,N),AI(NM,N),SCALE(N)
      INTEGER UPP
      LOGICAL NOCONV
      COMPLEX*16 DCMPLX
C
C     THIS SUBROUTINE IS A TRANSLATION OF THE ALGOL PROCEDURE
C     CBALANCE, WHICH IS A COMPLEX VERSION OF  BALANC,
C     NUM. MATH. 13,293-304(1969) BY PARLETT AND REINSCH.
C
C     THIS SUBROUTINE BALANCES A COMPLEX MATRIX AND ISOLATES
C     EIGENVALUES WHENEVER POSSIBLE.
C
C     ON INPUT--
C
C        NM MUST BE SET TO THE ROW DIMENSION OF TWO-DIMENSIONAL
C          ARRAY PARAMETERS AS DECLARED IN THE CALLING PROGRAM
C          DIMENSION STATEMENT,
C
C        N IS THE ORDER OF THE MATRIX,
C
C        RADIX IS THE BASE OF THE MACHINE FLOATING POINT
C          REPRESENTATION.  FOR SYSTEM/360 THIS IS 16.0,
C
C        AR AND AI ARE ARRAYS CONTAINING THE REAL AND IMAGINARY
C          PARTS, RESPECTIVELY, OF THE ELEMENTS OF THE MATRIX
C          TO BE BALANCED.
C
C     ON OUTPUT--
C
C        AR AND AI CONTAIN THE REAL AND IMAGINARY PARTS,
C          RESPECTIVELY, OF THE ELEMENTS OF THE BALANCED MATRIX,
C
C        LOW, UPP ARE TWO INTEGERS SUCH THAT AR(I,J) AND AI(I,J)
C          ARE EQUAL TO ZERO IF
C           (1) I IS GREATER THAN J AND
C           (2) J=1,...,LOW-1 OR I=UPP+1,...,N,
C
```

```
C        SCALE IS AN ARRAY WHICH CONTAINS INFORMATION DETERMINING
C           THE PERMUTATIONS USED AND SCALING FACTORS.
C
C     SUPPOSE THAT THE PRINCIPAL SUBMATRIX IN ROWS LOW THROUGH UPP
C     HAS BEEN BALANCED, THAT P(J) DENOTES THE INDEX INTERCHANGED
C     WITH J DURING THE PERMUTATION STEP, AND THAT THE ELEMENTS
C     OF THE DIAGONAL MATRIX USED ARE DENOTED BY D(I,J).  THEN
C        SCALE(J) = P(J),    FOR J = 1,...,LOW-1
C                 = D(J,J)       J = LOW,...,UPP
C                 = P(J)         J = UPP+1,...,N.
C     THE ORDER IN WHICH THE INTERCHANGES ARE MADE IS N TO UPP+1,
C     THEN 1 TO LOW-1.
C
C     THE ALGOL PROCEDURE EXC CONTAINED IN CBALANCE APPEARS IN
C     CBAL  IN LINE.  (NOTE THAT THE ALGOL ROLES OF IDENTIFIERS
C     K,L HAVE BEEN REVERSED.)
C
C     ARITHMETIC IS REAL EXCEPT FOR THE USE OF THE SUBROUTINES
C     CABS AND CMPLX.
C
C     TRANSLATED BY V. KLEMA, ARGONNE NATIONAL LABORATORY, AUG., 1969.
C     MODIFIED BY B. GARBOW, JAN., 1971.
C
C     ------------------------------------------------------------
C
      B2 = RADIX * RADIX
      K = 1
      L = N
      GO TO 100
C     ********** IN-LINE PROCEDURE FOR ROW AND
C                COLUMN EXCHANGE **********
   20 SCALE(M) = J
      IF (J .EQ. M) GO TO 50
C
      DO 30 I = 1, L
         F = AR(I,J)
         AR(I,J) = AR(I,M)
         AR(I,M) = F
         F = AI(I,J)
         AI(I,J) = AI(I,M)
         AI(I,M) = F
   30 CONTINUE
C
      DO 40 I = K, N
         F = AR(J,I)
         AR(J,I) = AR(M,I)
         AR(M,I) = F
         F = AI(J,I)
         AI(J,I) = AI(M,I)
         AI(M,I) = F
   40 CONTINUE
C
   50 GO TO (80,130), IEXC
```

```
C     ********** SEARCH FOR ROWS ISOLATING AN EIGENVALUE
C                AND PUSH THEM DOWN **********
   80 L = L - 1
      IF (L .LT. 1) GO TO 280
  100 LP1 = L + 1
C     ********** FOR J=L STEP -1 UNTIL 1 DO -- **********
      DO 120 JJ = 1, L
         J = LP1 - JJ
         R = 0.0
C
         DO 110 I = 1, L
            IF (I .EQ. J) GO TO 110
            R = R + CDABS(DCMPLX(AR(J,I),AI(J,I)))
  110    CONTINUE
C
         IF (R .NE. 0.0) GO TO 120
         M = L
         IEXC = 1
         GO TO 20
  120 CONTINUE
C
      GO TO 140
C     ********** SEARCH FOR COLUMNS ISOLATING AN EIGENVALUE
C                AND PUSH THEM LEFT **********
  130 K = K + 1
C
  140 DO 170 J = K, L
         C = 0.0
C
         DO 150 I = K, L
            IF (I .EQ. J) GO TO 150
            C = C + CDABS(DCMPLX(AR(I,J),AI(I,J)))
  150    CONTINUE
C
         IF (C .NE. 0.0) GO TO 170
         M = K
         IEXC = 2
         GO TO 20
  170 CONTINUE
C     ********** NOW BALANCE THE SUBMATRIX IN ROWS K TO L **********
      DO 180 I = K, L
      SCALE(I) = 1.0
  180 CONTINUE
C     ********** ITERATIVE LOOP FOR NORM REDUCTION **********
  190 NOCONV = .FALSE.
C
      DO 270 I = K, L
         C = 0.0
         R = 0.0
C
         DO 200 J = K, L
            IF (J .EQ. I) GO TO 200
            C = C + CDABS(DCMPLX(AR(J,I),AI(J,I)))
```

```
            R = R + CDABS(DCMPLX(AR(I,J),AI(I,J)))
  200    CONTINUE
C
         G = R / RADIX
         F = 1.0
  210    IF (C .GE. G) GO TO 220
         F = F * RADIX
         C = C * B2
         GO TO 210
  220    G = R * RADIX
  230    IF (C .LT. G) GO TO 240
         F = F / RADIX
         C = C / B2
         GO TO 230
C     ********** NOW BALANCE **********
  240    IF (F .EQ. 1.0) GO TO 270
         G = 1.0 / F
         SCALE(I) = SCALE(I) * F
         NOCONV = .TRUE.
C
         DO 250 J = K, N
            AR(I,J) = AR(I,J) * G
            AI(I,J) = AI(I,J) * G
  250    CONTINUE
C
         DO 260 J = 1, L
            AR(J,I) = AR(J,I) * F
            AI(J,I) = AI(J,I) * F
  260    CONTINUE
C
  270 CONTINUE
C
      IF(NOCONV) GO TO 190
  280 LOW = K
      UPP = L
      RETURN
C     ********** LAST CARD OF CBAL **********
      END
C      include 'cbabk2.for'
C
C     ------------------------------------------------------------------
C
      SUBROUTINE CBABK2(NM,N,LOW,UPP,SCALE,M,ZR,ZI)
      IMPLICIT REAL*8 (A-H,O-Z)
C
C
      REAL*8 SCALE(N),ZR(NM,M),ZI(NM,M)
      INTEGER UPP
C
C     THIS SUBROUTINE IS A TRANSLATION OF THE ALGOL PROCEDURE
C     CBABK2, WHICH IS A COMPLEX VERSION OF BALBAK,
C     NUM. MATH. 13,293-304(1969) BY PARLETT AND REINSCH.
C
```

```
C     THIS SUBROUTINE FORMS THE EIGENVECTORS OF A COMPLEX
C     GENERAL MATRIX BY BACK TRANSFORMING THOSE OF THE
C     CORRESPONDING BALANCED MATRIX PRODUCED BY  CBAL.
C
C     ON INPUT--
C
C        NM MUST BE SET TO THE ROW DIMENSION OF TWO-DIMENSIONAL
C          ARRAY PARAMETERS AS DECLARED IN THE CALLING PROGRAM
C          DIMENSION STATEMENT,
C
C        N IS THE ORDER OF THE MATRIX,
C
C        LOW AND UPP ARE INTEGERS PRODUCED BY THE BALANCING
C          SUBROUTINE  CBAL,
C
C        SCALE IS AN ARRAY PRODUCED BY  CBAL  CONTAINING
C          INFORMATION ABOUT THE PERMUTATION AND SCALING
C          TRANSFORMATIONS USED IN BALANCING,
C
C        M IS AN INTEGER GIVING THE NUMBER OF COLUMNS OF
C          Z = (ZR,ZI) TO BE BACK TRANSFORMED,
C
C        ZR AND ZI ARE ARRAYS, THE FIRST M COLUMNS OF WHICH
C          CONTAIN THE REAL AND IMAGINARY PARTS, RESPECTIVELY,
C          OF THE EIGENVECTORS TO BE BACK TRANSFORMED.
C
C     ON OUTPUT--
C
C        ZR AND ZI CONTAIN THE REAL AND IMAGINARY PARTS,
C          RESPECTIVELY, OF THE BACK TRANSFORMED EIGENVECTORS
C          IN THEIR FIRST M COLUMNS.
C
C     TRANSLATED BY V. KLEMA, ARGONNE NATIONAL LABORATORY, AUG., 1969.
C     MODIFIED BY B. GARBOW, JAN., 1971.
C
C     ------------------------------------------------------------------
C
      IF (UPP .LT. LOW) GO TO 120
C
      DO 110 I = LOW, UPP
         S = SCALE(I)
C     ********** LEFT HAND EIGENVECTORS ARE BACK TRANSFORMED
C                IF THE FOREGOING STATEMENT IS REPLACED BY
C                S=1.0/SCALE(I) **********
         DO 100 J = 1, M
            ZR(I,J) = ZR(I,J) * S
            ZI(I,J) = ZI(I,J) * S
  100    CONTINUE
C
  110 CONTINUE
C     ********** FOR I=LOW-1 STEP -1 UNTIL 1,
C                UPP+1 STEP 1 UNTIL N DO -- **********
  120 DO 140 II = 1, N
```

```
         I = II
         IF (I .GE. LOW .AND. I .LE. UPP) GO TO 140
         IF (I .LT. LOW) I = LOW - II
         K = SCALE(I)
         IF (K .EQ. I) GO TO 140
C
         DO 130 J = 1, M
            S = ZR(I,J)
            ZR(I,J) = ZR(K,J)
            ZR(K,J) = S
            S = ZI(I,J)
            ZI(I,J) = ZI(K,J)
            ZI(K,J) = S
  130    CONTINUE
C
  140 CONTINUE
C
      RETURN
C     ********** LAST CARD OF CBABK2 **********
      END
```

Example output

?? explanation needed. Note use of Pascal input file

```
gfortran ../fortran/comeigd26.f
mv ./a.out ../fortran/comeigd26.run
../fortran/comeigd26.run < ../fortran/d26f.in
```

```
##  COMPLEX MATRIX OF ORDER    1 REAL FRANK, IMAG MOLER
##  ORDER OF MATRIX   1
##  REAL PART OF MATRIX
##    1.00000000D+00
##  IMAGINARY PART OF MATRIX
##    1.00000000D+00
##  EIGENVALUE    1 =   1.00000000D+00 + I*  1.00000000D+00
##          Infinity             NaN
##  RESIDUALS,REAL AND IMAGINARY
##               NaN             NaN
##  MAXIMUM RESIDUAL MAGNITUDE=             NaN SOLUTION   1 ELEMENT   1
##  COMPLEX MATRIX OF ORDER    5 REAL FRANK, IMAG MOLER
##  ORDER OF MATRIX   5
##  REAL PART OF MATRIX
##    1.00000000D+00  1.00000000D+00  1.00000000D+00  1.00000000D+00  1.00000000D+00
##    1.00000000D+00  2.00000000D+00  2.00000000D+00  2.00000000D+00  2.00000000D+00
##    1.00000000D+00  2.00000000D+00  3.00000000D+00  3.00000000D+00  3.00000000D+00
##    1.00000000D+00  2.00000000D+00  3.00000000D+00  4.00000000D+00  4.00000000D+00
##    1.00000000D+00  2.00000000D+00  3.00000000D+00  4.00000000D+00  5.00000000D+00
##  IMAGINARY PART OF MATRIX
##    1.00000000D+00 -1.00000000D+00 -1.00000000D+00 -1.00000000D+00 -1.00000000D+00
##   -1.00000000D+00  2.00000000D+00  0.00000000D+00  0.00000000D+00  0.00000000D+00
##   -1.00000000D+00  0.00000000D+00  3.00000000D+00  1.00000000D+00  1.00000000D+00
##   -1.00000000D+00  0.00000000D+00  1.00000000D+00  4.00000000D+00  2.00000000D+00
##   -1.00000000D+00  0.00000000D+00  1.00000000D+00  2.00000000D+00  5.00000000D+00
```

```
##  TAU=    0.5600000000D+02 AT ITERATION    1
##  TAU=    0.7210750156D+01 AT ITERATION    2
##  TAU=    0.1616763195D+01 AT ITERATION    3
##  TAU=    0.1413590986D+00 AT ITERATION    4
##  TAU=    0.4138739759D-03 AT ITERATION    5
##  TAU=    0.2340651578D-07 AT ITERATION    6
##  TAU=    0.3311244821D-13 AT ITERATION    7
##  TAU=    0.1320750645D-14 AT ITERATION    8
##  EIGENVALUE    1 =   1.14382331D+01 + I*  5.89078700D+00
##    0.72919929D-01 -0.36498261D+00
##    0.41137353D+00 -0.25750477D+00
##    0.69331293D+00 -0.14425542D+00
##    0.89498614D+00 -0.51704965D-01
##    0.10000000D+01  0.00000000D+00
##  RESIDUALS,REAL AND IMAGINARY
##   -0.22204460D-15 -0.39968029D-14
##    0.37747583D-14 -0.31086245D-14
##    0.56066263D-14 -0.42188475D-14
##  EIGENVALUE    2 =   1.95778803D+00 + I*  1.84957614D+00
##    0.10000000D+01  0.00000000D+00
##    0.53123189D+00  0.45320234D+00
##    0.61249199D-01  0.40348153D+00
##   -0.21682241D+00  0.13113052D+00
##   -0.32155273D+00 -0.84132303D-01
##  RESIDUALS,REAL AND IMAGINARY
##  EIGENVALUE    3 =   9.16755008D-01 + I*  2.61387519D+00
##   -0.71977203D+00 -0.44570949D+00
##    0.52009034D+00 -0.42594831D+00
##    0.10000000D+01 -0.69388939D-17
##    0.22249390D+00  0.17713194D+00
##   -0.79070621D+00  0.76175261D-01
##  RESIDUALS,REAL AND IMAGINARY
##   -0.88817842D-15  0.83821838D-14
##  EIGENVALUE    4 =   4.07280150D-01 + I*  2.37082457D+00
##   -0.48556655D+00 -0.27303323D+00
##    0.10000000D+01  0.13877788D-16
##   -0.79151794D-01  0.15287163D+00
##   -0.93693223D+00 -0.76818410D-01
##    0.59902399D+00  0.30923994D-02
##  RESIDUALS,REAL AND IMAGINARY
##  EIGENVALUE    5 =   2.79943756D-01 + I*  2.27493710D+00
##    0.25776077D+00  0.11786836D+00
##   -0.78526602D+00 -0.57560560D-01
##    0.10000000D+01  0.00000000D+00
##   -0.81881791D+00  0.25041148D-01
##    0.31426697D+00 -0.13540625D-01
##  RESIDUALS,REAL AND IMAGINARY
##  MAXIMUM RESIDUAL MAGNITUDE=  8.42910830D-15 SOLUTION   3 ELEMENT   1
## Note: The following floating-point exceptions are signalling: IEEE_INVALID_FLAG IEEE_DIVIDE_BY_ZERO
```

BASIC

Listing

```
10 DIM A(20,20),Z(20,20),T(20,20),U(20,20),R(20)
15 DIM X(20,20),Y(20,20)
20 LET E9=1e-7 : REM machine precision approximation
25 LET M9=0
30 PRINT "COMEIG AT JULY 31 1984"
40 READ N
45 PRINT "ORDER = ";N
50 IF N <= 0 THEN QUIT
60 LET N1=N-1
80 PRINT "Real part Frank, Imag part Moler"
90 FOR I=1 TO N
100 FOR J=1 TO N
110 IF (I < J) THEN 130
115 LET A(I,J)=J
120 LET Z(I,J)=J-2
125 GOTO 145
130 LET A(I,J)=I
140 LET Z(I,J)=I-2
145 LET Y(I,J)=Z(I,J)
150 LET X(I,J)=A(I,J)
155 LET T(I,J)=0
160 LET U(I,J)=0
170 IF I<>J THEN 190
180 LET T(I,I)=1
190 REM PRINT
200 NEXT J
210 NEXT I
230 LET I3=0
240 LET I3=I3+1
250 IF I3>35 THEN 2220
260 IF M9=1 THEN 2220
270 LET T9=0
280 FOR K=1 TO N
290 LET T8=0
300 FOR I=1 TO N
310 IF I=K THEN 330
320 LET T8=T8+ABS(A(I,K))+ABS(Z(I,K))
330 NEXT I
340 LET T9=T9+T8
350 LET R(K)=T8+ABS(A(K,K))+ABS(Z(K,K))
360 NEXT K
370 PRINT "TAU=";T9;"  AT ITN ";I3
380 FOR K=1 TO N1
390 LET S9=R(K)
400 LET I=K
410 LET K1=K+1
420 FOR J=K1 TO N
430 IF S9>=R(J) THEN 460
440 LET S9=R(J)
450 LET I=J
```

```
460 NEXT J
470 IF I=K THEN 710
480 LET R(I)=R(K)
490 FOR J=1 TO N
500 LET T7=A(K,J)
510 LET A(K,J)=A(I,J)
520 LET A(I,J)=T7
530 LET T7=Z(K,J)
540 LET Z(K,J)=Z(I,J)
550 LET Z(I,J)=T7
560 NEXT J
570 FOR J=1 TO N
580 LET T7=A(J,K)
590 LET A(J,K)=A(J,I)
600 LET A(J,I)=T7
610 LET T7=Z(J,K)
620 LET Z(J,K)=Z(J,I)
630 LET Z(J,I)=T7
640 LET T7=T(J,K)
650 LET T(J,K)=T(J,I)
660 LET T(J,I)=T7
670 LET T7=U(J,K)
680 LET U(J,K)=U(J,I)
690 LET U(J,I)=T7
700 NEXT J
710 NEXT K
720 IF T9<100*E9 THEN 2220
730 LET M9=1
740 FOR K=1 TO N1
750 LET K1=K+1
760 FOR M=K1 TO N
770 LET H1=0
780 LET H2=0
790 LET H3=0
800 LET G=0
810 FOR I=1 TO N
820 IF I=K THEN 920
830 IF I=M THEN 920
840 LET H3=H3+A(K,I)*A(M,I)+Z(K,I)*Z(M,I)
850 LET H3=H3-A(I,K)*A(I,M)-Z(I,K)*Z(I,M)
860 LET H2=H2+Z(K,I)*A(M,I)-A(K,I)*Z(M,I)
870 LET H2=H2-A(I,K)*Z(I,M)+Z(I,K)*A(I,M)
880 LET T6=A(I,K)*A(I,K)+Z(I,K)*Z(I,K)+A(M,I)*A(M,I)+Z(M,I)*Z(M,I)
890 LET T5=A(I,M)*A(I,M)+Z(I,M)*Z(I,M)+A(K,I)*A(K,I)+Z(K,I)*Z(K,I)
900 LET G=G+T6+T5
910 LET H1=H1-T6+T5
920 NEXT I
930 LET B1=A(K,M)+A(M,K)
940 LET B2=Z(K,M)+Z(M,K)
950 LET E1=A(K,M)-A(M,K)
960 LET E2=Z(K,M)-Z(M,K)
970 LET D1=A(K,K)-A(M,M)
980 LET D2=Z(K,K)-Z(M,M)
```

```
990 LET T6=B1*B1+E2*E2+D1*D1
1000 LET T5=B2*B2+E1*E1+D2*D2
1010 IF T6<T5 THEN 1090
1020 LET S8=1
1030 LET C1=B1
1040 LET S=E2
1050 LET C3=D1
1060 LET C4=D2
1070 LET R8=SQR(T6)
1080 GOTO 1150
1090 LET S8=-1
1100 LET C1=B2
1110 LET S=-E1
1120 LET C3=D2
1130 LET C4=D1
1140 LET R8=SQR(T5)
1150 LET R7=SQR(S*S+C1*C1)
1160 LET S6=-1
1170 IF C3<0 THEN 1190
1180 LET S6=1
1190 LET S7=0
1200 LET C7=-1
1210 IF C1<0 THEN 1230
1220 LET C7=1
1230 IF R7>E9 THEN 1360
1240 LET S5=0
1250 LET S7=0
1260 LET C5=1
1270 LET C7=1
1280 IF S8>0 THEN 1320
1290 LET E=E2
1300 LET B=-B1
1310 GOTO 1340
1320 LET E=E1
1330 LET B=B2
1340 LET S4=C3*C3+C4*C4
1350 GOTO 1570
1360 IF ABS(S)<=E9 THEN 1390
1370 LET C7=C1/R7
1380 LET S7=S/R7
1390 LET C9=C3/R7
1400 LET C0=C9+(S6*SQR(1+C9*C9))
1410 LET S5=S6/SQR(1+C0*C0)
1420 LET C5=S5*C0
1430 LET E3=(E1*B1+E2*B2)/R7
1440 LET T4=(B1*B2-E1*E2)/R7
1450 LET T6=S6*(T4*C3-C4*R7)/R8
1460 LET T5=(C3*C4+R7*T4)/R8
1470 LET S4=R8*R8+T5*T5
1480 LET T5=H1*C5*S5
1490 LET C2=C7*C7-S7*S7
1500 LET S2=2*C7*S7
1510 LET T8=H3*C2+H2*S2
```

```
1520 LET T7=H2*C2-H3*S2
1530 LET H3=H3*C5*C5-T8*S5*S5-C7*T5
1540 LET H2=H2*C5*C5+T7*S5*S5-S7*T5
1550 LET B=S8*T6*C7+E3*S7
1560 LET E=C7*E3-S8*T6*S7
1570 LET S=H3-S6*R8*E
1580 LET C1=H2-S6*R8*B
1590 LET R9=SQR(C1*C1+S*S)
1600 IF R9>=E9 THEN 1660
1610 LET C8=1
1620 LET C6=1
1630 LET S0=0
1640 LET S1=0
1650 GOTO 1730
1660 LET C8=-C1/R9
1670 LET S0=S/R9
1680 LET T5=C8*B-E*S0
1690 LET Q8=T5*T5
1700 LET Q9=R9/(G+2*(Q8+S4))
1710 LET C6=1/SQR(1-Q9*Q9)
1720 LET S1=C6*Q9
1730 LET T8=S5*S1*(S7*C8-S0*C7)
1740 LET Q7=C5*C6-T8
1750 LET Q6=C5*C6+T8
1760 LET Q5=-S5*S1*(C7*C8+S7*S0)
1770 LET Q4=Q5
1780 LET T7=S5*C6*C7
1790 LET T8=C5*S1*S0
1800 LET P1=T7-T8
1810 LET P2=-T7-T8
1820 LET T7=S5*C6*S7
1830 LET T8=C5*S1*C8
1840 LET P3=T7+T8
1850 LET P4=T7-T8
1860 LET T8=SQR(P1*P1+P3*P3)
1870 LET T7=SQR(P2*P2+P4*P4)
1880 IF T7>E9 THEN 1900
1890 IF T8<=E9 THEN 2190
1900 LET M9=0
1910 FOR I=1 TO N
1920 LET P5=A(K,I)
1930 LET P6=A(M,I)
1940 LET P7=Z(K,I)
1950 LET P8=Z(M,I)
1960 LET A(K,I)=Q7*P5-Q5*P7+P1*P6-P3*P8
1970 LET Z(K,I)=Q7*P7+Q5*P5+P1*P8+P3*P6
1980 LET A(M,I)=P2*P5-P4*P7+Q6*P6-Q4*P8
1990 LET Z(M,I)=P2*P7+P4*P5+Q6*P8+Q4*P6
2000 NEXT I
2010 FOR I=1 TO N
2020 LET P5=A(I,K)
2030 LET P6=A(I,M)
2040 LET P7=Z(I,K)
```

```
2050 LET P8=Z(I,M)
2060 LET A(I,K)=Q6*P5-Q4*P7-P2*P6+P4*P8
2070 LET Z(I,K)=Q6*P7+Q4*P5-P2*P8-P4*P6
2080 LET A(I,M)=-P1*P5+P3*P7+Q7*P6-Q5*P8
2090 LET Z(I,M)=-P1*P7-P3*P5+Q7*P8+Q5*P6
2100 LET P5=T(I,K)
2110 LET P6=T(I,M)
2120 LET P7=U(I,K)
2130 LET P8=U(I,M)
2140 LET T(I,K)=Q6*P5-Q4*P7-P2*P6+P4*P8
2150 LET U(I,K)=Q6*P7+Q4*P5-P2*P8-P4*P6
2160 LET T(I,M)=-P1*P5+P3*P7+Q7*P6-Q5*P8
2170 LET U(I,M)=-P1*P7-P3*P5+Q7*P8+Q5*P6
2180 NEXT I
2190 NEXT M
2200 NEXT K
2210 GOTO 240
2220 IF I3<=35 THEN 2240
2230 PRINT "FAILURE TO CONVERGE IN 35 ITERATIONS"
2240 PRINT "EIGENSOLUTIONS"
2250 FOR I=1 TO N
2260 LET G=T(1,I)*T(1,I)+U(1,I)*U(1,I)
2270 LET K=1
2280 IF N=1 THEN 2350
2290 FOR M=2 TO N
2300 LET B=T(M,I)*T(M,I)+U(M,I)*U(M,I)
2310 IF B<=G THEN 2340
2320 LET K=M
2330 LET G=B
2340 NEXT M
2350 LET E=T(K,I)/G
2360 LET S=-U(K,I)/G
2370 FOR K=1 TO N
2380 LET G=T(K,I)*E-U(K,I)*S
2390 LET U(K,I)=U(K,I)*E+T(K,I)*S
2400 LET T(K,I)=G
2410 NEXT K
2430 PRINT "EIGENVALUE";I;"=(";A(I,I);",";Z(I,I);")"
2440 PRINT "VECTOR"
2450 FOR K=1 TO N
2460 PRINT "(";T(K,I);",";U(K,I);")"
2470 NEXT K
2480 PRINT "RESIDUALS"
2490 FOR J=1 TO N
2500 LET S=-A(I,I)*T(J,I)+Z(I,I)*U(J,I)
2510 LET G=-Z(I,I)*T(J,I)-A(I,I)*U(J,I)
2520 FOR K=1 TO N
2530 LET S=S+X(J,K)*T(K,I)-Y(J,K)*U(K,I)
2540 LET G=G+X(J,K)*U(K,I)+Y(J,K)*T(K,I)
2550 NEXT K
2560 PRINT "(";S;",";G;")"
2570 NEXT J
2590 REM
```

```
2600 NEXT I
2610 GOTO 40
2620 DATA 1, 5, 0
2800 END
```

Example output

```
bwbasic ../BASIC/comeiga26.bas  >../BASIC/a26.out
# echo "done"
```

```
Bywater BASIC Interpreter/Shell, version 2.20 patch level 2

Copyright (c) 1993, Ted A. Campbell

Copyright (c) 1995-1997, Jon B. Volkoff


COMEIG AT JULY 31 1984
ORDER =  1
Real part Frank, Imag part Moler
TAU= 0  AT ITN  1
EIGENSOLUTIONS
EIGENVALUE 1=( 1, -1)
VECTOR
( 1, 0)
RESIDUALS
( 0, 0)
ORDER =  5
Real part Frank, Imag part Moler
TAU= 56  AT ITN  1
TAU= 7.2107502  AT ITN  2
TAU= 1.6167632  AT ITN  3
TAU= 0.141359  AT ITN  4
TAU= 0.0004139  AT ITN  5
TAU= 0  AT ITN  6
EIGENSOLUTIONS
EIGENVALUE 1=( 11.438233, 3.8907870)
VECTOR
( 0.0729199, -0.3649826)
( 0.4113735, -0.2575048)
( 0.6933129, -0.1442554)
( 0.8949861, -0.0517050)
( 1.0000000, 0)
RESIDUALS
( 0, 0)
( 0, 0)
( 0, 0)
( -0, 0)
( -0, -0)
EIGENVALUE 2=( 1.957788, -0.1504239)
VECTOR
( 1, -0)
( 0.5312319, 0.4532023)
( 0.0612492, 0.4034815)
```

```
( -0.2168224, 0.1311305)
( -0.3215527, -0.0841323)
RESIDUALS
( 0, -0)
( 0, 0)
( -0, 0)
( -0, -0)
( 0, -0)
EIGENVALUE 3=( 0.916755, 0.6138752)
VECTOR
( -0.719772, -0.4457095)
( 0.5200903, -0.4259483)
( 1, 0)
( 0.2224939, 0.1771319)
( -0.7907062, 0.0761753)
RESIDUALS
( 0, 0)
( -0, 0)
( -0, 0)
( -0, -0)
( -0, -0)
EIGENVALUE 4=( 0.4072801, 0.3708246)
VECTOR
( -0.4855665, -0.2730332)
( 1, 0)
( -0.0791518, 0.1528716)
( -0.9369322, -0.0768184)
( 0.5990240, 0.0030924)
RESIDUALS
( 0, 0)
( -0, 0)
( -0, 0)
( -0, -0)
( 0, -0)
EIGENVALUE 5=( 0.2799438, 0.2749371)
VECTOR
( 0.2577608, 0.1178684)
( -0.785266, -0.0575606)
( 1, 0)
( -0.8188179, 0.0250411)
( 0.3142670, -0.0135406)
RESIDUALS
( -0, -0)
( -0, -0)
( -0, -0)
( 0, -0)
( 0, -0)
ORDER =  0
```

Pascal

Listing

```
program dr26(input,output);

{dr26.pas == eigensolutions of a complex matrix by Eberlein's
          complex Jacobi procedure

          Copyright 1988 J.C.Nash
}
{constype.def ==
  This file contains various definitions and type statements which are
  used throughout the collection of "Compact Numerical Methods".  In many
  cases not all definitions are needed, and users with very tight memory
  constraints may wish to remove some of the lines of this file when
  compiling certain programs.

  Modified for Turbo Pascal 5.0

          Copyright 1988, 1990 J.C.Nash
}
uses Dos, Crt; {Turbo Pascal 5.0 Modules}
{ 1. Interrupt, Unit, Interface, Implementation, Uses are reserved words now.}
{ 2. System,Dos,Crt are standard unit names in Turbo 5.0.}

const
  big = 1.0E+35;    {a very large number}
  Maxconst = 25;    {Maximum number of constants in data record}
  Maxobs = 100;     {Maximum number of observations in data record}
  Maxparm  = 25;    {Maximum number of parameters to adjust}
  Maxvars = 10;     {Maximum number of variables in data record}
  acctol = 0.0001;  {acceptable point tolerance for minimisation codes}
  maxm = 20;        {Maximum number or rows in a matrix}
  maxn = 20;        {Maximum number of columns in a matrix}
  maxmn = 40;       {maxn+maxm, the number of rows in a working array}
  maxsym = 210;     {maximum number of elements of a symmetric matrix
              which need to be stored = maxm * (maxm + 1)/2 }
  reltest = 10.0;   {a relative size used to check equality of numbers.
              Numbers x and y are considered equal if the
              floating-point representation of reltest+x equals
              that of reltest+y.}
  stepredn = 0.2;   {factor to reduce stepsize in line search}
  yearwrit = 1990;  {year in which file was written}

type
  str2 = string[2];
  rmatrix = array[1..maxm, 1..maxn] of real; {a real matrix}
  wmatrix = array[1..maxmn, 1..maxn] of real; {a working array, formed
                  as one real matrix stacked on another}
  smatvec = array[1..maxsym] of real; {a vector to store a symmetric matrix
              as the row-wise expansion of its lower triangle}
  rvector = array[1..maxm] of real;  {a real vector. We will use vectors
              of m elements always. While this is NOT space efficient,
```

```
              it simplifies program codes.}
  cgmethodtype= (Fletcher_Reeves,Polak_Ribiere,Beale_Sorenson);
    {three possible forms of the conjugate gradients updating formulae}
  probdata = record
          m     : integer; {number of observations}
          nvar  : integer; {number of variables}
          nconst: integer; {number of constants}
          vconst: array[1..Maxconst] of real;
          Ydata : array[1..Maxobs, 1..Maxvars] of real;
          nlls  : boolean; {true if problem is nonlinear least squares}
        end;
{
  NOTE: Pascal does not let us define the work-space for the function
  within the user-defined code.  This is a weakness of Pascal for this
  type of work.

}
var {global definitions}
  banner     : string[80]; {program name and description}

function calceps:real;
{calceps.pas ==
  This function returns the machine EPSILON or floating point tolerance,
  the smallest positive real number such that 1.0 + EPSILON > 1.0.
  EPSILON is needed to set various tolerances for different algorithms.
  While it could be entered as a constant, I prefer to calculate it, since
  users tend to move software between machines without paying attention to
  the computing environment. Note that more complete routines exist.
}
var
  e,e0: real;
  i: integer;
begin {calculate machine epsilon}
  e0 := 1; i:=0;
  repeat
    e0 := e0/2; e := 1+e0;  i := i+1;
  until (e=1.0) or (i=50); {note safety check}
  e0 := e0*2;
{ Writeln('Machine EPSILON =',e0);}
  calceps:=e0;
end; {calceps}

Procedure matrixin(var m, n: integer; var A: rmatrix;
              var avector: smatvec; var sym :boolean);

{matrixin.pas --

  This procedure generates an m by n real matrix in both
  A or avector.

  A is of type rmatrix, an array[1..nmax, 1..nmax] of real
  where nmax >= n for all possible n to be provided.
```

```
  avector is of type rvector, an array[1..nmax*(nmax+1)/2]
  of real, with nmax as above.

  sym is set true if the resulting matrix is symmetric.
}

var
  temp : real;
  i,j,k: integer;
  inchar: char;
  mtype: integer;
  mn : integer;

begin
  if (m=0) or (n=0) then
  begin
    writeln;
    writeln('******* Matrix dimensions zero *********');
    halt;
  end;
  writeln('Matrixin.pas -- generate or input a real matrix ',m,' by ',n);
  writeln('Possible matrices to generate:');
  writeln('0) Keyboard or console file input');
  writeln('1) Hilbert segment');
  writeln('2) Ding Dong');
  writeln('3) Moler');
  writeln('4) Frank symmetric');
  writeln('5) Bordered symmetric');
  writeln('6) Diagonal');
  writeln('7) Wilkinson W+');
  writeln('8) Wilkinson W-');
  writeln('9) Constant');
  writeln('10) Unit');
{ Note: others could be added.}
  mn:=n;
  if m>mn then mn:=m; {mn is maximum of m and n}
  write('Enter type to generate ');
  readln(mtype);
  writeln(mtype);
  case mtype of
    0: begin
      sym:=false;
      if m=n then
      begin
        write('Is matrix symmetric? ');  readln(inchar);
        writeln(inchar);
        if (inchar='y') or (inchar='Y') then sym:=true else sym:=false;
      end; {ask if symmetric}
      if sym then
      begin
        for i:=1 to n do
        begin
          writeln('Row ',i,' lower triangle elements');
```

```
        for j:=1 to i do
        begin
          read(A[i,j]);
          write(A[i,j]:10:5,' ');
          A[j,i]:=A[i,j];
          if (7*(j div 7) = j) and (j<i) then writeln;
        end;
        writeln;
      end;
    end {symmetric matrix}
    else
    begin {not symmetric}
      for i:=1 to m do
      begin
        writeln('Row ',i);
        for j:=1 to n do
        begin
          read(A[i,j]);
          write(A[i,j]:10:5,' ');
        end; {loop on j}
        writeln;
      end; {loop on i}
    end; {else not symmetric}
  end; {case 0 -- input of matrix}
  1: begin {Hilbert}
    for i:=1 to mn do
      for j:=1 to mn do
        A[i,j]:=1.0/(i+j-1.0);
    if m=n then sym:=true;
  end;
  2: begin {Ding Dong}
    for i:=1 to mn do
      for j:=1 to mn do
        A[i,j]:=0.5/(1.5+n-i-j);
    if m=n then sym:=true;
  end;
  3: begin {Moler}
    for i:=1 to mn do
    begin
      for j:=1 to i do
      begin
        A[i,j]:=j-2.0;
        A[j,i]:=j-2.0;
      end;
      A[i,i]:=i;
      if m=n then sym:=true;
    end;
  end;
  4: begin {Frank symmetric}
    for i:=1 to mn do
      for j:=1 to i do
      begin
        A[i,j]:=j;
```

```
        A[j,i]:=j;
      end;
      if m=n then sym:=true;
  end;
  5: begin {Bordered}
    temp:=2.0;
    for i:=1 to (mn-1) do
    begin
      temp:=temp/2.0; {2^(1-i)}
      for j:=1 to mn do
        A[i,j]:=0.0;
      A[i,mn]:=temp;
      A[mn,i]:=temp;
      A[i,i]:=1.0;
    end;
    A[mn,mn]:=1.0;
    if m=n then sym:=true;
  end;
  6: begin {Diagonal}
    for i:=1 to mn do
    begin
      for j:=1 to mn do
        A[i,j]:=0.0;
      A[i,i]:=i;
    end;
    if m=n then sym:=true;
  end;
  7: begin {W+}
    k:=mn div 2; {[n/2]}
    for i:=1 to mn do
      for j:=1 to mn do
        A[i,j]:=0.0;
    if m=n then sym:=true;
    for i:=1 to k do
    begin
      A[i,i]:=k+1-i;
      A[mn-i+1,mn-i+1]:=k+1-i;
    end;
    for i:=1 to mn-1 do
    begin
      A[i,i+1]:=1.0;
      A[i+1,i]:=1.0;
    end;
  end;
  8: begin {W-}
    k:=mn div 2; {[n/2]}
    for i:=1 to mn do
      for j:=1 to mn do
        A[i,j]:=0.0;
    if m=n then sym:=true;
    for i:=1 to k do
    begin
      A[i,i]:=k+1-i;
```

```
        A[mn-i+1,mn-i+1]:=i-1-k;
      end;
      for i:=1 to mn-1 do
      begin
        A[i,i+1]:=1.0;
        A[i+1,i]:=1.0;
      end;
      if m=n then sym:=true;
    end;
    9: begin {Constant}
      write('Set all elements to a constant value = ');
      readln(temp);
      writeln(temp);
      for i:=1 to mn do
        for j:=1 to mn do
          A[i,j]:=temp;
      if m=n then sym:=true;
    end;
    10: begin {Unit}
      for i:=1 to mn do
      begin
        for j:=1 to mn do A[i,j]:=0.0;
        A[i,i]:=1.0;
      end;
      if m=n then sym:=true;
    end;
    else {case statement else} {!!!! Note missing close bracket here}
    begin
      writeln;
      writeln('*** ERROR *** unrecognized option');
      halt;
    end; {else of case statement}
  end; {case statement}
  if sym then
  begin {convert to vector form}
    k:=0; {index for vector element}
    for i:=1 to n do
    begin
      for j:=1 to i do
      begin
        k:=k+1;
        avector[k]:=A[i,j];
      end;
    end;
  end;
end; {matrixin}

procedure comeig( n : integer;
          var itcount: integer;
          var A, Z, T, U : rmatrix);

var
  Rvec : rvector;
```

```
  i, itlimit, j, k, k1, m, n1 : integer;
  aki, ami, bv, br, bi : real;
  c, c1i, c1r, c2i, c2r, ca, cb, ch, cos2a, cot2x, cotx, cx : real;
  d, de, di, diag, dr, e, ei, er, eps, eta, g, hi, hj, hr : real;
  isw, max, nc, nd, root1, root2, root : real;
  s, s1i, s1r, s2i, s2r, sa, sb, sh, sig, sin2a, sx : real;
  tanh, tau, te, tee, tem, tep ,tse, zki, zmi : real;
  mark : boolean;

begin

  writeln('alg26.pas -- comeig');
  eps := Calceps;
  mark := false; n1 := n-1;
  for i := 1 to n do
  begin
    for j := 1 to n do
    begin
      T[i,j] := 0.0; U[i,j] := 0.0; if i=j then T[i,i] := 1.0;
    end;
  end;
  itlimit := itcount;
  itcount := 0;
  while (itcount<=itlimit) and (not mark) do
  begin
    itcount := itcount+1;
    tau := 0.0;
    diag := 0.0;
    for k := 1 to n do
    begin
      tem := 0;
      for i := 1 to n do if i<>k then tem := tem+ABS(A[i,k])+ABS(Z[i,k]);
      tau := tau+tem; tep := abs(A[k,k])+abs(Z[k,k]);
      diag := diag+tep;
      Rvec[k] := tem+tep;
    end;
    writeln('TAU=',tau,'  AT ITN ',itcount);
    for k := 1 to n1 do
    begin
      max := Rvec[k]; i := k; k1 := k+1;
      for j := k1 to n do
      begin
        if max<Rvec[j] then
        begin
          max := Rvec[j]; i := j;
        end;
      end;
      if i<>k then
      begin
        Rvec[i] := Rvec[k];
        for j := 1 to n do
        begin
          tep := A[k,j]; A[k,j] := A[i,j]; A[i,j] := tep; tep := Z[k,j];
```

```
        Z[k,j] := Z[i,j]; Z[i,j] := tep;
      end;
      for j := 1 to n do
      begin
        tep := A[j,k]; A[j,k] := A[j,i]; A[j,i] := tep; tep := Z[j,k];
        Z[j,k] := Z[j,i]; Z[j,i] := tep; tep := T[j,k]; T[j,k] := T[j,i];
        T[j,i] := tep; tep := U[j,k]; U[j,k] := U[j,i]; U[j,i] := tep;
      end;
    end;
  end;
  if tau>=100.0*eps then
  begin
    mark := true;
    for k := 1 to n1 do
    begin
      k1 := k+1;
      for m := k1 to n do
      begin
        hj := 0.0; hr := 0.0; hi := 0.0; g := 0.0;
        for i := 1 to n do
        begin
          if (i<>k) and (i<>m) then
          begin
            hr := hr+A[k,i]*A[m,i]+Z[k,i]*Z[m,i];
            hr := hr-A[i,k]*A[i,m]-Z[i,k]*Z[i,m];
            hi := hi+Z[k,i]*A[m,i]-A[k,i]*Z[m,i];
            hi := hi-A[i,k]*Z[i,m]+Z[i,k]*A[i,m];
            te := A[i,k]*A[i,k]+Z[i,k]*Z[i,k]+A[m,i]*A[m,i]+Z[m,i]*Z[m,i];
            tee := A[i,m]*A[i,m]+Z[i,m]*Z[i,m]+A[k,i]*A[k,i]+Z[k,i]*Z[k,i];
            g := g+te+tee; hj := hj-te+tee;
          end;
        end;
        br := A[k,m]+A[m,k]; bi := Z[k,m]+Z[m,k]; er := A[k,m]-A[m,k];
        ei := Z[k,m]-Z[m,k]; dr := A[k,k]-A[m,m]; di := Z[k,k]-Z[m,m];
        te := br*br+ei*ei+dr*dr; tee := bi*bi+er*er+di*di;
        if te>=tee then
        begin
          isw := 1.0; c := br; s := ei; d := dr; de := di;
          root2 := sqrt(te);
        end
        else
        begin
          isw := -1.0; c := bi; s := -er; d := di; de := dr;
          root2 := sqrt(tee);
        end;
        root1 := sqrt(s*s+c*c); sig := -1.0; if d>=0.0 then sig := 1.0;
        sa := 0.0; ca := -1.0; if c>=0.0 then ca := 1.0;
        if root1<=eps then
        begin
          sx := 0.0; sa := 0.0; cx := 1.0; ca := 1.0;
          if isw<=0.0 then
          begin
            e := ei; bv := -br;
```

```
    end
    else
    begin
      e := er; bv := bi;
    end;
    nd := d*d+de*de;
  end
  else
  begin
    if abs(s)>eps then
    begin
      ca := c/root1; sa := s/root1;
    end;
    cot2x := d/root1; cotx := cot2x+(sig*sqrt(1.0+cot2x*cot2x));
    sx := sig/sqrt(1.0+cotx*cotx); cx := sx*cotx;

    eta := (er*br+ei*bi)/root1; tse := (br*bi-er*ei)/root1;
    te := sig*(tse*d-de*root1)/root2; tee := (d*de+root1*tse)/root2;
    nd := root2*root2+tee*tee; tee := hj*cx*sx; cos2a := ca*ca-sa*sa;
    sin2a := 2.0*ca*sa; tem := hr*cos2a+hi*sin2a;
    tep := hi*cos2a-hr*sin2a; hr := hr*cx*cx-tem*sx*sx-ca*tee;
    hi := hi*cx*cx+tep*sx*sx-sa*tee;
    bv := isw*te*ca+eta*sa; e := ca*eta-isw*te*sa;
  end;

  s := hr-sig*root2*e; c := hi-sig*root2*bv; root := sqrt(c*c+s*s);
  if root<eps then
  begin
    cb := 1.0; ch := 1.0; sb := 0.0; sh := 0.0;
  end
  else
  begin
    cb := -c/root; sb := s/root; tee := cb*bv-e*sb; nc := tee*tee;
    tanh := root/(g+2.0*(nc+nd)); ch := 1.0/sqrt(1.0-tanh*tanh);
    sh := ch*tanh;
  end;
  tem := sx*sh*(sa*cb-sb*ca); c1r := cx*ch-tem; c2r := cx*ch+tem;
  c1i := -sx*sh*(ca*cb+sa*sb); c2i := c1i; tep := sx*ch*ca;
  tem := cx*sh*sb; s1r := tep-tem; s2r := -tep-tem; tep := sx*ch*sa;
  tem := cx*sh*cb; s1i := tep+tem; s2i := tep-tem;
  tem := sqrt(s1r*s1r+s1i*s1i); tep := sqrt(s2r*s2r+s2i*s2i);
  if tep>eps then mark := false;
  if (tep>eps) and (tem>eps) then
  begin
    for i := 1 to n do
    begin
      aki := A[k,i]; ami := A[m,i]; zki := Z[k,i]; zmi := Z[m,i];
      A[k,i] := c1r*aki-c1i*zki+s1r*ami-s1i*zmi;
      Z[k,i] := c1r*zki+c1i*aki+s1r*zmi+s1i*ami;
      A[m,i] := s2r*aki-s2i*zki+c2r*ami-c2i*zmi;
      Z[m,i] := s2r*zki+s2i*aki+c2r*zmi+c2i*ami;
    end;
    for i := 1 to n do
```

```
            begin
              aki := A[i,k]; ami := A[i,m]; zki := Z[i,k]; zmi := Z[i,m];
              A[i,k] := c2r*aki-c2i*zki-s2r*ami+s2i*zmi;
              Z[i,k] := c2r*zki+c2i*aki-s2r*zmi-s2i*ami;
              A[i,m] := -s1r*aki+s1i*zki+c1r*ami-c1i*zmi;
              Z[i,m] := -s1r*zki-s1i*aki+c1r*zmi+c1i*ami;
              aki := T[i,k]; ami := T[i,m]; zki := U[i,k]; zmi := U[i,m];
              T[i,k] := c2r*aki-c2i*zki-s2r*ami+s2i*zmi;
              U[i,k] := c2r*zki+c2i*aki-s2r*zmi-s2i*ami;
              T[i,m] := -s1r*aki+s1i*zki+c1r*ami-c1i*zmi;
              U[i,m] := -s1r*zki-s1i*aki+c1r*zmi+c1i*ami;
            end;
          end;
        end;
      end;
    end
    else mark := true;
  end;
  if itcount>itlimit then itcount := -itcount;
end;

procedure stdceigv(n: integer;
                var T, U: rmatrix);

var
  i, k, m : integer;
  b, e, g, s : real;

begin
  writeln('alg11.pas -- standardized eigensolutions');
  for i := 1 to n do
  begin
    g := T[1,i]*T[1,i]+U[1,i]*U[1,i];
    k := 1;
    if n>1 then
    begin
      for m := 2 to n do
      begin
        b := T[m,i]*T[m,i]+U[m,i]*U[m,i];
        if b>g then
        begin
          k := m;
          g := b;
        end;
      end;
    end;
    e := T[k,i]/g;
    s := -U[k,i]/g;
    for k := 1 to n do
    begin
      g := T[k,i]*e-U[k,i]*s; U[k,i] := U[k,i]*e+T[k,i]*s; T[k,i] := g;
    end;
  end;
```

```
end;

procedure comres( i, n: integer;
                  A, Z, T, U, Acopy, Zcopy : rmatrix);
var
  j, k: integer;
  g, s, ss : real;

begin
  writeln('alg12.pas -- complex eigensolution residuals');
  ss := 0.0;
  for j := 1 to n do
  begin
    s := -A[i,i]*T[j,i]+Z[i,i]*U[j,i]; g := -Z[i,i]*T[j,i]-A[i,i]*U[j,i];
    for k := 1 to n do
    begin
      s := s+Acopy[j,k]*T[k,i]-Zcopy[j,k]*U[k,i];
      g := g+Acopy[j,k]*U[k,i]+Zcopy[j,k]*T[k,i];
    end;
    writeln('(',s,',',g,')');
    ss := ss+s*s+g*g;
  end;
  writeln('Sum of squares = ',ss);
end;

{Main program}
var
  A, Z, Acopy, Zcopy, T, U : rmatrix;
  i, it, j, k, n : integer;
  sym : boolean;
  avec : smatvec; {for compatibility of Matrixin only}

begin
  banner:='dr26.pas -- Eigensolutions of a general complex matrix';
  write(' Order of matrix = '); readln(n);
  writeln(n);
  writeln('Provide real part of matrix (A)');
  matrixin(n,n,A,avec,sym);
  writeln('Provide imaginary part of matrix (Z)');
  matrixin(n,n,Z,avec,sym);
  for i:=1 to n do
  begin
    for j:=1 to n do
    begin
    Acopy[i,j]:=A[i,j]; Zcopy[i,j]:=Z[i,j];
    write('(',A[i,j]:10:5,',',Z[i,j]:10:5,') ');
    if (3 * (j div 3) = j) and (j<n) then writeln;
    end; {copy loop j}
    writeln;
  end; {copy loop i}
  it:=50; {allow a maximum of 50 iterations}
  comeig( n, it, A, Z, T, U);
  if it>0 then writeln('Converged in ',it,' iterations')
```

```
    else writeln('Not converged after ',it,' iterations');
  stdceigv(n, T, U); {standardize the eigensolutions -- alg11.pas}
  for i:=1 to n do
  begin
    writeln('EIGENVALUE ',i,'=(',A[i,i],',',Z[i,i],')');
    writeln('VECTOR');
    for k:=1 to n do
    begin
    writeln('(',T[k,i],',',U[k,i],')');
    end; { loop on  k}
    comres( i, n, A, Z, T, U, Acopy, Zcopy); {residuals -- alg12.pas}
  end; {loop on i}
end. {dr26.pas == eigensolutions of a complex matrix}
```

Example output

We create an order 5 complex matrix where the real part is a Frank matrix and the imaginary part is a Moler matrix.

```
fpc ../Pascal2021/dr26.pas
# copy to run file
mv ../Pascal2021/dr26 ../Pascal2021/dr26.run
../Pascal2021/dr26.run <../Pascal2021/dr26p.in >../Pascal2021/dr26p.out
```

```
## Free Pascal Compiler version 3.0.4+dfsg-23 [2019/11/25] for x86_64
## Copyright (c) 1993-2017 by Florian Klaempfl and others
## Target OS: Linux for x86-64
## Compiling ../Pascal2021/dr26.pas
## Linking ../Pascal2021/dr26
## /usr/bin/ld.bfd: warning: link.res contains output sections; did you forget -T?
## 594 lines compiled, 0.1 sec
```

```
 Order of matrix = 5
Provide real part of matrix (A)
Matrixin.pas -- generate or input a real matrix 5 by 5
Possible matrices to generate:
0) Keyboard or console file input
1) Hilbert segment
2) Ding Dong
3) Moler
4) Frank symmetric
5) Bordered symmetric
6) Diagonal
7) Wilkinson W+
8) Wilkinson W-
9) Constant
10) Unit
Enter type to generate 4
Provide imaginary part of matrix (Z)
Matrixin.pas -- generate or input a real matrix 5 by 5
Possible matrices to generate:
0) Keyboard or console file input
1) Hilbert segment
2) Ding Dong
3) Moler
```

```
4) Frank symmetric
5) Bordered symmetric
6) Diagonal
7) Wilkinson W+
8) Wilkinson W-
9) Constant
10) Unit
Enter type to generate 3
(   1.00000,   1.00000) (   1.00000,  -1.00000) (   1.00000,  -1.00000)
(   1.00000,  -1.00000) (   1.00000,  -1.00000)
(   1.00000,  -1.00000) (   2.00000,   2.00000) (   2.00000,   0.00000)
(   2.00000,   0.00000) (   2.00000,   0.00000)
(   1.00000,  -1.00000) (   2.00000,   0.00000) (   3.00000,   3.00000)
(   3.00000,   1.00000) (   3.00000,   1.00000)
(   1.00000,  -1.00000) (   2.00000,   0.00000) (   3.00000,   1.00000)
(   4.00000,   4.00000) (   4.00000,   2.00000)
(   1.00000,  -1.00000) (   2.00000,   0.00000) (   3.00000,   1.00000)
(   4.00000,   2.00000) (   5.00000,   5.00000)
alg26.pas -- comeig
TAU= 5.6000000000000000E+001  AT ITN 1
TAU= 7.2107501562138063E+000  AT ITN 2
TAU= 1.6167631945882213E+000  AT ITN 3
TAU= 1.4135909863720170E-001  AT ITN 4
TAU= 4.1387397594418944E-004  AT ITN 5
TAU= 2.3406515779719136E-008  AT ITN 6
TAU= 3.4129840545673970E-014  AT ITN 7
Converged in 7 iterations
alg11.pas -- standardized eigensolutions
EIGENVALUE 1=( 1.1438233058170844E+001, 5.8907869982801460E+000)
VECTOR
( 7.2919928669437584E-002,-3.6498261444044539E-001)
( 4.1137352665876853E-001,-2.5750477494283192E-001)
( 6.9331292735519889E-001,-1.4425541881377915E-001)
( 8.9498613998517162E-001,-5.1704964902128843E-002)
( 9.9999999999999989E-001, 0.0000000000000000E+000)
alg12.pas -- complex eigensolution residuals
(-9.8809849191638932E-015, 1.1102230246251565E-016)
(-6.4392935428259079E-015,-7.0637939941775585E-015)
( 1.3322676295501878E-015,-5.8841820305133297E-015)
( 6.6613381477509392E-015,-6.8833827526759706E-015)
( 6.2172489379008766E-015,-4.4408920985006262E-015)
Sum of squares =  3.7553650207708380E-028
EIGENVALUE 2=( 1.9577880276110047E+000, 1.8495761391986802E+000)
VECTOR
( 1.0000000000000000E+000,-2.7755575615628914E-017)
( 5.3123189422526218E-001, 4.5320234203420851E-001)
( 6.1249198876340880E-002, 4.0348152506126495E-001)
(-2.1682241447409578E-001, 1.3113051937238548E-001)
(-3.2155273442134791E-001,-8.4132303063020955E-002)
alg12.pas -- complex eigensolution residuals
(-7.2858385991025898E-015,-1.6098233857064770E-015)
(-1.3322676295501878E-015,-2.9698465908722937E-015)
( 5.1347814888913490E-016,-4.4408920985006262E-015)
```

```
(-2.7478019859472624E-015,-3.1086244689504383E-015)
(-6.6613381477509392E-015,-2.6645352591003757E-015)
Sum of squares =  1.5494221958391976E-028
EIGENVALUE 3=( 9.1675500804230603E-001, 2.6138751925800818E+000)
VECTOR
(-7.1977202607833091E-001,-4.4570949348429373E-001)
( 5.2009033993569398E-001,-4.2594831161408075E-001)
( 9.9999999999999989E-001,-6.9388939039072284E-018)
( 2.2249389617958162E-001, 1.7713193887438106E-001)
(-7.9070621353809867E-001, 7.6175261316525300E-002)
alg12.pas -- complex eigensolution residuals
( 3.4416913763379853E-015, 2.9976021664879227E-015)
( 1.3322676295501878E-015,-2.8865798640254070E-015)
(-3.3306690738754696E-016,-4.2188474935755949E-015)
( 2.2204460492503131E-016, 8.8817841970012523E-016)
(-7.7715611723760958E-016, 5.7731597280508140E-015)
Sum of squares =  8.3619255953427251E-029
EIGENVALUE 4=( 4.0728014995617290E-001, 2.3708245672136488E+000)
VECTOR
(-4.8556654637005797E-001,-2.7303322645238509E-001)
( 9.9999999999999989E-001, 0.0000000000000000E+000)
(-7.9151794360700342E-002, 1.5287163429309791E-001)
(-9.3693223105011536E-001,-7.6818410030158116E-002)
( 5.9902398680012370E-001, 3.0923993850483986E-003)
alg12.pas -- complex eigensolution residuals
(-5.0653925498522767E-016, 9.9920072216264089E-016)
( 0.0000000000000000E+000, 7.3552275381416621E-016)
( 2.8449465006019636E-016,-2.2204460492503131E-015)
(-7.6327832942979512E-016,-6.6613381477509392E-016)
(-7.9797279894933126E-016,-8.8817841970012523E-016)
Sum of squares =  9.2592452453819041E-030
EIGENVALUE 5=( 2.7994375621967177E-001, 2.2749371027274541E+000)
VECTOR
( 2.5776076693001215E-001, 1.1786835850188734E-001)
(-7.8526601960503983E-001,-5.7560559889801660E-002)
( 9.9999999999999989E-001, 0.0000000000000000E+000)
(-8.1881790998991266E-001, 2.5041147656711178E-002)
( 3.1426697408278631E-001,-1.3540625375941799E-002)
alg12.pas -- complex eigensolution residuals
(-2.7061686225238191E-016,-1.6653345369377348E-016)
( 1.1102230246251565E-016,-1.0824674490095276E-015)
(-6.7307270867900115E-016,-4.9960036108132044E-016)
(-6.8001160258290838E-016, 1.2212453270876722E-015)
(-1.3600232051658168E-015, 6.6613381477509392E-016)
Sum of squares =  6.2349093054620180E-030
```

Algorithms 11 and 12 – standardization and residuals for a complex eigensolution

These algorithms are probably among the least used of those included in Nashlib. Their intent was to allow proposed eigensolutions of complex matrices to be standardized and tested. This seemed a potentially important task in the 1970s. ?? include COMEIG (ref to Eberlein's work, others??)

The purpose of standardization is to facilitate comparisons between eigenvectors that are supposedly equivalent. Any (preferably) unit-length multiple of an eigenvector is also an eigenvector, so it is difficult to compare two proposed solutions for the same eigenvalue. Therefore we choose a multiplier so that the largest magnitude component of the eigenvector is set to $1 + 0i$ where

$$i = \sqrt{-1}$$

Fortran

The Algorithm 11 code:

```
      SUBROUTINE A11VS(N,X,Y,VNORM)
C  ALGORITHM 11  -  COMPLEX VECTOR STANDARDISATION
C  J.C. NASH   JULY 1978, FEBRUARY 1980, APRIL 1989
C  STANDARDISES COMPEX VECTOR (N ELEMENTS)  X+SQRT(-1)*Y
C      TO  1.0+SQRT(-1)*0.0
C  VNORM = NORM OF VECTOR (LARGEST ELEMENT)
C  STEP 0
      INTEGER N,I,K
      REAL X(N),Y(N),VNORM,G,B,E,S
C  STEP 1
      G=0.0
C  STEP 2
      DO 60 I=1,N
C  STEP 3
        B=X(I)**2+Y(I)**2
C  STEP 4
        IF(B.LE.G)GOTO 60
C  STEP 5
        K=I
        G=B
C  STEP 6
  60  CONTINUE
C  SAVE NORM
      VNORM=G
C  SAFETY CHECK
      IF(G.EQ.0.0)RETURN
C  STEP 7
      E=X(K)/G
      S=-Y(K)/G
C  STEP 8
      DO 85 I=1,N
        G=X(I)*E-Y(I)*S
        Y(I)=Y(I)*E+X(I)*S
        X(I)=G
  85  CONTINUE
C  END
      RETURN
```

```
      END
```

The Algorithm 11 code:

```
      SUBROUTINE A12CVR(N,X,Y,A,NA,Z,NZ,E,F,U,V)
C  ALGORITHM 12 RESIDUALS OF A COMPLEX EIGENSOLUTION
C  J.C. NASH   JULY 1978, FEBRUARY 1980, APRIL 1989
C  N    =  ORDER OF PROBLEM
C  U + I*V =  ( A + I*Z - E - I*F)*(X + I*Y)  WHERE I=SQRT(-1)
C  NA,NZ = FIRST DIMENSIONS OF A & Z RESPECTIVELY
C  STEP 0
      INTEGER N,NA,NZ,J,K
      REAL A(NA,N),Z(NZ,N),X(N),Y(N),E,F,U(N),V(N),S,G
C  STEP 1
      DO 50 J=1,N
C  STEP 2
        S=-E*X(J)+F*Y(J)
        G=-F*X(J)-E*Y(J)
C  STEP 3
        DO 35 K=1,N
          S=S+A(J,K)*X(K)-Z(J,K)*Y(K)
          G=G+A(J,K)*Y(K)+Z(J,K)*X(K)
  35    CONTINUE
C  STEP 4  NOTE SAVE IN U & V
        U(J)=S
        V(J)=G
C  STEP 5
  50  CONTINUE
      RETURN
      END
```

Example output

We illustrate by finding a single eigensolution of the Hilbert segments of order 5 and 10. ?? Do we want to swap in the Frank matrix (the computations are generally easier)?

```
## #!/bin/bash
gfortran ../fortran/a1112.f
mv ./a.out ../fortran/a1112.run
../fortran/a1112.run <../fortran/a1112.in
```

```
##  ORDER=   2
##  MATRIX A
##  ROW  1
##    1.00000000E+00  0.00000000E+00
##  ROW  2
##    0.00000000E+00  1.00000000E+00
##  MATRIX Z
##  ROW  1
##    0.00000000E+00 -1.00000000E+00
##  ROW  2
##    1.00000000E+00  0.00000000E+00
##  SOLUTION   1  EV=  2.00000000E+00 + SQRT(-1)*  0.00000000E+00
##  REAL PART
##    0.50000000E+00  0.00000000E+00
##  IMAGINARY PART
```

```
##    0.00000000E+00  0.50000000E+00
##  STANDARDIZED VECTOR - NORM=  2.50000000E-01
##  REAL PART
##    0.10000000E+01  0.00000000E+00
##  IMAGINARY PART
##    0.00000000E+00  0.10000000E+01
##  RESIDUALS
##  REAL PART
##    0.00000000E+00  0.00000000E+00
##  IMAGINARY PART
##    0.00000000E+00  0.00000000E+00
##  SOLUTION   2  EV=  0.00000000E+00 + SQRT(-1)*  0.00000000E+00
##  REAL PART
##   -0.10000000E+01  0.00000000E+00
##  IMAGINARY PART
##    0.00000000E+00  0.10000000E+01
##  STANDARDIZED VECTOR - NORM=  1.00000000E+00
##  REAL PART
##    0.10000000E+01  0.00000000E+00
##  IMAGINARY PART
##    0.00000000E+00 -0.10000000E+01
##  RESIDUALS
##  REAL PART
##    0.00000000E+00  0.00000000E+00
##  IMAGINARY PART
##    0.00000000E+00  0.00000000E+00
##  SOLUTION  -1  EV=  0.00000000E+00 + SQRT(-1)*  0.00000000E+00
##  ORDER=   2
##  MATRIX A
##  ROW  1
##    1.00000000E+00  1.00000000E+00
##  ROW  2
##    1.00000000E+00  1.00000000E+00
##  MATRIX Z
##  ROW  1
##    0.00000000E+00 -1.00000000E+00
##  ROW  2
##    1.00000000E+00  0.00000000E+00
##  SOLUTION   1  EV=  2.41420007E+00 + SQRT(-1)*  0.00000000E+00
##  REAL PART
##    0.14141999E+01  0.10000000E+01
##  IMAGINARY PART
##    0.00000000E+00  0.10000000E+01
##  STANDARDIZED VECTOR - NORM=  2.00000000E+00
##  REAL PART
##    0.70709997E+00  0.10000000E+01
##  IMAGINARY PART
##   -0.70709997E+00  0.00000000E+00
##  RESIDUALS
##  REAL PART
##    0.19133091E-04 -0.23841858E-06
##  IMAGINARY PART
##   -0.19133091E-04  0.00000000E+00
##  SOLUTION   2  EV= -5.85799992E-01 + SQRT(-1)*  0.00000000E+00
```

```
##  REAL PART
##   -0.14141999E+01  0.10000000E+01
##  IMAGINARY PART
##    0.00000000E+00  0.10000000E+01
##  STANDARDIZED VECTOR - NORM=  2.00000000E+00
##  REAL PART
##   -0.70709997E+00  0.10000000E+01
##  IMAGINARY PART
##    0.70709997E+00  0.00000000E+00
##  RESIDUALS
##  REAL PART
##   -0.12131917E+00  0.17160004E+00
##  IMAGINARY PART
##    0.12131917E+00  0.00000000E+00
##  SOLUTION   0  EV=  0.00000000E+00 + SQRT(-1)*  0.00000000E+00
##  ORDER=   3
##  MATRIX A
##  ROW  1
##    1.00000000E+00  2.00000000E+00  3.00000000E+00
##  ROW  2
##    4.00000000E+00  5.00000000E+00  6.00000000E+00
##  ROW  3
##    7.00000000E+00  8.00000000E+00  9.00000000E+00
##  MATRIX Z
##  ROW  1
##    9.00000000E+00  8.00000000E+00  7.00000000E+00
##  ROW  2
##    6.00000000E+00  5.00000000E+00  4.00000000E+00
##  ROW  3
##    3.00000000E+00  2.00000000E+00  1.00000000E+00
##  SOLUTION   1  EV=  1.36844997E+01 + SQRT(-1)*  1.36844997E+01
##  REAL PART
##    0.49906299E+00  0.56529301E+00  0.63152498E+00
##  IMAGINARY PART
##    0.47247899E+00  0.10070200E+00 -0.27107400E+00
##  STANDARDIZED VECTOR - NORM=  4.72304910E-01
##  REAL PART
##    0.39612967E+00  0.69806379E+00  0.10000000E+01
##  IMAGINARY PART
##    0.91818923E+00  0.45909339E+00  0.00000000E+00
##  RESIDUALS
##  REAL PART
##   -0.69141388E-04  0.45776367E-04  0.16307831E-03
##  IMAGINARY PART
##    0.25606155E-03  0.20432472E-03  0.89049339E-04
##  SOLUTION   2  EV=  1.31529999E+00 + SQRT(-1)*  1.31531000E+00
##  REAL PART
##    0.74067497E+00 -0.24551900E-01 -0.78978002E+00
##  IMAGINARY PART
##   -0.27984101E+00 -0.11183600E+00  0.56165900E-01
##  STANDARDIZED VECTOR - NORM=  6.26910388E-01
##  REAL PART
##    0.10000000E+01  0.20914188E-01 -0.95817167E+00
##  IMAGINARY PART
```

```
##    0.00000000E+00 -0.14309023E+00 -0.28618470E+00
##  RESIDUALS
##  REAL PART
##    0.27894974E-04  0.14424324E-04 -0.43809414E-05
##  IMAGINARY PART
##    0.67234039E-04  0.22649765E-04 -0.16033649E-04
##  SOLUTION   3  EV=  0.00000000E+00 + SQRT(-1)*  0.00000000E+00
##  REAL PART
##   -0.40955499E+00  0.81911302E+00 -0.40955701E+00
##  IMAGINARY PART
##   -0.16320599E-01  0.32640599E-01 -0.16320400E-01
##  STANDARDIZED VECTOR - NORM=  6.72011554E-01
##  REAL PART
##   -0.49999815E+00  0.10000000E+01 -0.50000060E+00
##  IMAGINARY PART
##   -0.43958426E-06 -0.37252903E-08 -0.96857548E-07
##  RESIDUALS
##  REAL PART
##    0.47311187E-05  0.68247318E-05  0.91567636E-05
##  IMAGINARY PART
##    0.11682510E-04  0.64373016E-05  0.89406967E-06
##  SOLUTION  -1  EV=  0.00000000E+00 + SQRT(-1)*  0.00000000E+00
##  ORDER=   0
```

Pascal

Currently we do not seem to have a separate example driver for these two codes. However, they are used in conjunction with the example(s) for Algorithm 26 (Complex matrix eigensolutions).

Listing – Algorithm 11

```
procedure stdceigv(n: integer;
                var T, U: rmatrix);

var
  i, k, m : integer;
  b, e, g, s : real;

begin
  writeln('alg11.pas -- standardized eigensolutions');
  writeln(confile,'alg11.pas -- standardized eigensolutions');
  for i := 1 to n do
  begin
    g := T[1,i]*T[1,i]+U[1,i]*U[1,i];

    k := 1;
    if n>1 then
    begin
      for m := 2 to n do
      begin
        b := T[m,i]*T[m,i]+U[m,i]*U[m,i];
        if b>g then
        begin
          k := m;
```

```
          g := b;
        end;
      end;
    end;
    e := T[k,i]/g;
    s := -U[k,i]/g;
    for k := 1 to n do
    begin
      g := T[k,i]*e-U[k,i]*s; U[k,i] := U[k,i]*e+T[k,i]*s; T[k,i] := g;
    end;
  end;
end;
```

Listing – Algorithm 12

```
procedure comres( i, n: integer;
                  A, Z, T, U, Acopy, Zcopy : rmatrix);

var
  j, k: integer;
  g, s, ss : real;

begin
  writeln('alg12.pas -- complex eigensolution residuals');
  writeln(confile,'alg12.pas -- complex eigensolution residuals');
  ss := 0.0;
  for j := 1 to n do
  begin
    s := -A[i,i]*T[j,i]+Z[i,i]*U[j,i]; g := -Z[i,i]*T[j,i]-A[i,i]*U[j,i];

    for k := 1 to n do
    begin
      s := s+Acopy[j,k]*T[k,i]-Zcopy[j,k]*U[k,i];
      g := g+Acopy[j,k]*U[k,i]+Zcopy[j,k]*T[k,i];
    end;
    writeln('(',s,',',g,')'); writeln(confile,'(',s,',',g,')');
    ss := ss+s*s+g*g;
  end;
  writeln('Sum of squares = ',ss); writeln(confile,'Sum of squares = ',ss);
  writeln; writeln(confile);
end;
```

Cleanup of working files

The following script is included to remove files created during compilation or execution of the examples.

```
## remove object and run files
cd ../fortran/
echo `pwd`
rm  *.o
rm  *.run
rm *.out
cd ../Pascal2021/
```

```
echo `pwd`
rm  *.o
rm  *.run
rm *.out
cd ../BASIC
echo `pwd`
rm *.out
cd ../Documentation
## ?? others
```

```
## /j19z/j19store/versioned/Nash-Compact-Numerical-Methods/fortran
## rm: cannot remove '*.o': No such file or directory
## rm: cannot remove '*.out': No such file or directory
## /j19z/j19store/versioned/Nash-Compact-Numerical-Methods/Pascal2021
## /j19z/j19store/versioned/Nash-Compact-Numerical-Methods/BASIC
```

References

Nash, John C. 1979. *Compact Numerical Methods for Computers : Linear Algebra and Function Minimisation.* Book. Hilger: Bristol.