NAG Library Routine Document

F02JQF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

F02JQF solves the quadratic eigenvalue problem

$$\left(\lambda^2 A + \lambda B + C\right)x = 0,$$

where A, B and C are complex n by n matrices.

The routine returns the $2n$ eigenvalues, λ_j, for $j = 1, 2, \ldots, 2n$, and can optionally return the corresponding right eigenvectors, x_j and/or left eigenvectors, y_j as well as estimates of the condition numbers of the computed eigenvalues and backward errors of the computed right and left eigenvectors. A left eigenvector satisfies the equation

$$y^{\mathrm{H}}\left(\lambda^2 A + \lambda B + C\right) = 0,$$

where y^{H} is the complex conjugate transpose of y.

λ is represented as the pair (α, β), such that $\lambda = \alpha/\beta$. Note that the computation of α/β may overflow and indeed β may be zero.

2 Specification

```
SUBROUTINE F02JQF (SCAL, JOBVL, JOBVR, SENSE, TOL, N, A, LDA, B, LDB, C,    &
                   LDC, ALPHA, BETA, VL, LDVL, VR, LDVR, S, BEVL, BEVR,     &
                   IWARN, IFAIL)
INTEGER                 SCAL, SENSE, N, LDA, LDB, LDC, LDVL, LDVR, IWARN,   &
                        IFAIL
REAL (KIND=nag_wp)      TOL, S(*), BEVL(*), BEVR(*)
COMPLEX (KIND=nag_wp) A(LDA,*), B(LDB,*), C(LDC,*), ALPHA(2*N),            &
                        BETA(2*N), VL(LDVL,*), VR(LDVR,*)
CHARACTER(1)            JOBVL, JOBVR
```

3 Description

The quadratic eigenvalue problem is solved by linearizing the problem and solving the resulting $2n$ by $2n$ generalized eigenvalue problem. The linearization is chosen to have favourable conditioning and backward stability properties. An initial preprocessing step is performed that reveals and deflates the zero and infinite eigenvalues contributed by singular leading and trailing matrices.

The algorithm is backward stable for problems that are not too heavily damped, that is $\|B\| \leq 10\sqrt{\|A\| \cdot \|C\|}$.

Further details on the algorithm are given in Hammarling *et al.* (2013).

4 References

Fan H -Y, Lin W.-W and Van Dooren P. (2004) Normwise scaling of second order polynomial matrices. SIAM J. Matrix Anal. Appl. **26, 1** 252–256

Gaubert S and Sharify M (2009) Tropical scaling of polynomial matrices *Lecture Notes in Control and Information Sciences Series* **389** 291–303 Springer–Verlag

Hammarling S, Munro C J and Tisseur F (2013) An algorithm for the complete solution of quadratic eigenvalue problems. *ACM Trans. Math. Software.* **39(3):18:1–18:119** http://eprints.ma.man.ac.uk/1815/

5 Arguments

1: SCAL – INTEGER *Input*

On entry: determines the form of scaling to be performed on A, B and C.

SCAL $= 0$
No scaling.

SCAL $= 1$ (the recommended value)
Fan, Lin and Van Dooren scaling if $\frac{\|B\|}{\sqrt{\|A\| \times \|C\|}} < 10$ and no scaling otherwise where $\|Z\|$ is the Frobenius norm of Z.

SCAL $= 2$
Fan, Lin and Van Dooren scaling.

SCAL $= 3$
Tropical scaling with largest root.

SCAL $= 4$
Tropical scaling with smallest root.

Constraint: SCAL $= 0$, 1, 2, 3 or 4.

2: JOBVL – CHARACTER(1) *Input*

On entry: if JOBVL = 'N', do not compute left eigenvectors.

If JOBVL = 'V', compute the left eigenvectors.

If SENSE $= 1$, 2, 4, 5, 6 or 7, JOBVL must be set to `V'.

Constraint: JOBVL = 'N' or 'V'.

3: JOBVR – CHARACTER(1) *Input*

On entry: if JOBVR = 'N', do not compute right eigenvectors.

If JOBVR = 'V', compute the right eigenvectors.

If SENSE $= 1$, 3, 4, 5, 6 or 7, JOBVR must be set to `V'.

Constraint: JOBVR = 'N' or 'V'.

4: SENSE – INTEGER *Input*

On entry: determines whether, or not, condition numbers and backward errors are computed.

SENSE $= 0$
Do not compute condition numbers, or backward errors.

SENSE $= 1$
Just compute condition numbers for the eigenvalues.

SENSE $= 2$
Just compute backward errors for the left eigenpairs.

SENSE $= 3$
Just compute backward errors for the right eigenpairs.

SENSE $= 4$
Compute backward errors for the left and right eigenpairs.

SENSE $= 5$
Compute condition numbers for the eigenvalues and backward errors for the left eigenpairs.

SENSE $= 6$
: Compute condition numbers for the eigenvalues and backward errors for the right eigenpairs.

SENSE $= 7$
: Compute condition numbers for the eigenvalues and backward errors for the left and right eigenpairs.

Constraint: SENSE $= 0$, 1, 2, 3, 4, 5, 6 or 7.

5: TOL – REAL (KIND=nag_wp) *Input*

On entry: TOL is used as the tolerance for making decisions on rank in the deflation procedure. If TOL is zero on entry then $n \times$ ***machine precision*** is used in place of TOL, where ***machine precision*** is as returned by routine X02AJF. A diagonal element of a triangular matrix, R, is regarded as zero if $|r_{jj}| \leq \text{TOL} \times \text{size}(X)$, or $n \times$ ***machine precision*** $\times\, \text{size}(X)$ when TOL is zero, where $\text{size}(X)$ is based on the size of the absolute values of the elements of the matrix X containing the matrix R. See Hammarling *et al.* (2013) for the motivation. If TOL is -1.0 on entry then no deflation is attempted. The recommended value for TOL is zero.

6: N – INTEGER *Input*

On entry: the order of the matrices A, B and C.

Constraint: $\text{N} \geq 0$.

7: A(LDA, $*$) – COMPLEX (KIND=nag_wp) array *Input/Output*

Note: the second dimension of the array A must be at least N.

On entry: the n by n matrix A.

On exit: A is used as internal workspace, but if JOBVL = 'V' or JOBVR = 'V', then A is restored on exit.

8: LDA – INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F02JQF is called.

Constraint: $\text{LDA} \geq \text{N}$.

9: B(LDB, $*$) – COMPLEX (KIND=nag_wp) array *Input/Output*

Note: the second dimension of the array B must be at least N.

On entry: the n by n matrix B.

On exit: B is used as internal workspace, but is restored on exit.

10: LDB – INTEGER *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F02JQF is called.

Constraint: $\text{LDB} \geq \text{N}$.

11: C(LDC, $*$) – COMPLEX (KIND=nag_wp) array *Input/Output*

Note: the second dimension of the array C must be at least N.

On entry: the n by n matrix C.

On exit: C is used as internal workspace, but if JOBVL = 'V' or JOBVR = 'V', C is restored on exit.

12: LDC – INTEGER *Input*

On entry: the first dimension of the array C as declared in the (sub)program from which F02JQF is called.

Constraint: LDC $\geq$ N.

13: ALPHA($2 \times$ N) – COMPLEX (KIND=nag_wp) array *Output*

On exit: ALPHA(j), for $j = 1, 2, \ldots, 2n$, contains the first part of the the jth eigenvalue pair (α_j, β_j) of the quadratic eigenvalue problem.

14: BETA($2 \times$ N) – COMPLEX (KIND=nag_wp) array *Output*

On exit: BETA(j), for $j = 1, 2, \ldots, 2n$, contains the second part of the jth eigenvalue pair (α_j, β_j) of the quadratic eigenvalue problem. Although BETA is declared complex, it is actually real and non-negative. Infinite eigenvalues have β_j set to zero.

15: VL(LDVL, $*$) – COMPLEX (KIND=nag_wp) array *Output*

Note: the second dimension of the array VL must be at least $2 \times$ N if JOBVL = 'V'.

On exit: if JOBVL = 'V', the left eigenvectors y_j are stored one after another in the columns of VL, in the same order as the corresponding eigenvalues. Each eigenvector will be normalized with length unity and with the element of largest modulus real and positive.

If JOBVL = 'N', VL is not referenced.

16: LDVL – INTEGER *Input*

On entry: the first dimension of the array VL as declared in the (sub)program from which F02JQF is called.

Constraint: LDVL $\geq$ N.

17: VR(LDVR, $*$) – COMPLEX (KIND=nag_wp) array *Output*

Note: the second dimension of the array VR must be at least $2 \times$ N if JOBVR = 'V'.

On exit: if JOBVR = 'V', the right eigenvectors x_j are stored one after another in the columns of VR, in the same order as the corresponding eigenvalues. Each eigenvector will be normalized with length unity and with the element of largest modulus real and positive.

If JOBVR = 'N', VR is not referenced.

18: LDVR – INTEGER *Input*

On entry: the first dimension of the array VR as declared in the (sub)program from which F02JQF is called.

Constraint: LDVR $\geq$ N.

19: S($*$) – REAL (KIND=nag_wp) array *Output*

Note: the dimension of the array S must be at least $2 \times$ N if SENSE = 1, 5, 6 or 7.

Note: also: computing the condition numbers of the eigenvalues requires that both the left and right eigenvectors be computed.

On exit: if SENSE = 1, 5, 6 or 7, S(j) contains the condition number estimate for the jth eigenvalue (large condition numbers imply that the problem is near one with multiple eigenvalues). Infinite condition numbers are returned as the largest model real number (X02ALF).

If SENSE = 0, 2, 3 or 4, S is not referenced.

20: BEVL($*$) – REAL (KIND=nag_wp) array *Output*

Note: the dimension of the array BEVL must be at least $2 \times \mathrm{N}$ if SENSE $= 2$, 4, 5 or 7.

On exit: if SENSE $= 2$, 4, 5 or 7, BEVL(j) contains the backward error estimate for the computed left eigenpair (λ_j, y_j).

If SENSE $= 0$, 1, 3 or 6, BEVL is not referenced.

21: BEVR($*$) – REAL (KIND=nag_wp) array *Output*

Note: the dimension of the array BEVR must be at least $2 \times \mathrm{N}$ if SENSE $= 3$, 4, 6 or 7.

On exit: if SENSE $= 3$, 4, 6 or 7, BEVR(j) contains the backward error estimate for the computed right eigenpair (λ_j, x_j).

If SENSE $= 0$, 1, 2 or 5, BEVR is not referenced.

22: IWARN – INTEGER *Output*

On exit: IWARN will be positive if there are warnings, otherwise IWARN will be 0.

If IFAIL $= 0$ then:

if IWARN $= 1$ then one, or both, of the matrices A and C is zero. In this case no scaling is performed, even if SCAL > 0;

if IWARN $= 2$ then the matrices A and C are singular, or nearly singular, so the problem is potentially ill-posed;

if IWARN $= 3$ then both the conditions for IWARN $= 1$ and IWARN $= 2$ above, apply. If IWARN $= 4$, $\|\mathrm{B}\| \geq 10\sqrt{\|A\| \cdot \|C\|}$ and backward stability cannot be guaranteed.

If IFAIL $= 2$, IWARN returns the value of INFO from F08XNF (ZGGES).

If IFAIL $= 3$, IWARN returns the value of INFO from F08WNF (ZGGEV).

23: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, if you are not familiar with this argument, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: IFAIL $= 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL $= 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

IFAIL $= 1$

The quadratic matrix polynomial is nonregular (singular).

IFAIL $= 2$

The QZ iteration failed in F08XNF (ZGGES).
IWARN returns the value of INFO returned by F08XNF (ZGGES). This failure is unlikely to happen, but if it does, please contact NAG.

IFAIL $= 3$

The QZ iteration failed in F08WNF (ZGGEV).
IWARN returns the value of INFO returned by F08WNF (ZGGEV). This failure is unlikely to happen, but if it does, please contact NAG.

IFAIL $= -1$

On entry, SCAL $= \langle value \rangle$.
Constraint: SCAL $= 0$, 1, 2, 3 or 4.

IFAIL $= -2$

On entry, JOBVL $= \langle value \rangle$.
Constraint: JOBVL = 'N' or 'V'.

On entry, SENSE $= \langle value \rangle$ and JOBVL $= \langle value \rangle$.
Constraint: when JOBVL = 'N', SENSE $= 0$ or 3,
when JOBVL = 'V', SENSE $= 1$, 2, 4, 5, 6 or 7.

IFAIL $= -3$

On entry, JOBVR $= \langle value \rangle$.
Constraint: JOBVR = 'N' or 'V'.

On entry, SENSE $= \langle value \rangle$ and JOBVR $= \langle value \rangle$.
Constraint: when JOBVR = 'N', SENSE $= 0$ or 2,
when JOBVR = 'V', SENSE $= 1$, 3, 4, 5, 6 or 7.

IFAIL $= -4$

On entry, SENSE $= \langle value \rangle$.
Constraint: SENSE $= 0$, 1, 2, 3, 4, 5, 6 or 7.

IFAIL $= -6$

On entry, N $= \langle value \rangle$.
Constraint: N ≥ 0.

IFAIL $= -8$

On entry, LDA $= \langle value \rangle$ and N $= \langle value \rangle$.
Constraint: LDA $\geq$ N.

IFAIL $= -10$

On entry, LDB $= \langle value \rangle$ and N $= \langle value \rangle$.
Constraint: LDB $\geq$ N.

IFAIL $= -12$

On entry, LDC $= \langle value \rangle$ and N $= \langle value \rangle$.
Constraint: LDC $\geq$ N.

IFAIL $= -16$

On entry, LDVL $= \langle value \rangle$, N $= \langle value \rangle$ and JOBVL $= \langle value \rangle$.
Constraint: when JOBVL = 'V', LDVL $\geq$ N.

IFAIL $= -18$

On entry, LDVR $= \langle value \rangle$, N $= \langle value \rangle$ and JOBVR $= \langle value \rangle$.
Constraint: when JOBVR = 'V', LDVR $\geq$ N.

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

The algorithm is backward stable for problems that are not too heavily damped, that is $\|B\| \leq \sqrt{\|A\| \cdot \|C\|}$.

8 Parallelism and Performance

F02JQF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

F02JQF makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

None.

10 Example

To solve the quadratic eigenvalue problem

$$\left(\lambda^2 A + \lambda B + C\right)x = 0$$

where

$$A = \begin{pmatrix} 2i & 4i & 4i \\ 6i & 2i & 2i \\ 6i & 4i & 2i \end{pmatrix}, \quad B = \begin{pmatrix} 3+3i & 2+2i & 1+i \\ 2+2i & 1+i & 3+3i \\ 1+i & 3+3i & 2+2i \end{pmatrix} \quad \text{and} \quad C = \begin{pmatrix} 1 & 1 & 2 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{pmatrix}.$$

The example also returns the left eigenvectors, condition numbers for the computed eigenvalues and the maximum backward errors of the computed right and left eigenpairs.

10.1 Program Text

```
    Program f02jqfe

!     F02JQF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
```

```
      Use nag_library, Only: f02jqf, m01def, m01edf, nag_wp, x04caf, x04daf
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Real (Kind=nag_wp), Parameter    :: tol = 0.0E0_nag_wp
      Real (Kind=nag_wp), Parameter    :: zero = 0.0E+0_nag_wp
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: t0, t1
      Integer                          :: i, ifail, iwarn, j, lda, ldb, ldc,   &
                                          ldvl, ldvr, n, scal, sense, tdvl,    &
                                          tdvr
      Character (1)                    :: jobvl, jobvr
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: a(:,:), alpha(:), b(:,:), beta(:), &
                                          c(:,:), cvr(:), e(:), vl(:,:),       &
                                          vr(:,:)
      Real (Kind=nag_wp), Allocatable  :: bevl(:), bevr(:), ea(:,:), s(:)
      Integer, Allocatable             :: irank(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: abs, all, maxval, real
!     .. Executable Statements ..
      Write (nout,*) 'F02JQF Example Program Results'
!     Skip heading in data file and read in n, scal, sense, jobVL and jobVR
      Read (nin,*)
      Read (nin,*) n, scal, sense
      Read (nin,*) jobvl, jobvr

      lda = n
      ldb = n
      ldc = n
      ldvl = n
      ldvr = n
      tdvl = 2*n
      tdvr = 2*n
      Allocate (a(lda,n),b(ldb,n),c(ldc,n),alpha(2*n),beta(2*n),e(2*n),        &
        vl(ldvl,tdvl),vr(ldvr,tdvr),s(2*n),bevr(2*n),bevl(2*n),cvr(n),         &
        ea(2*n,2),irank(2*n))

!     Read in the matrices A, B and C

      Read (nin,*)(a(i,1:n),i=1,n)
      Read (nin,*)(b(i,1:n),i=1,n)
      Read (nin,*)(c(i,1:n),i=1,n)

!     Solve the quadratic eigenvalue problem
      ifail = -1
      Call f02jqf(scal,jobvl,jobvr,sense,tol,n,a,lda,b,ldb,c,ldc,alpha,beta,   &
        vl,ldvl,vr,ldvr,s,bevl,bevr,iwarn,ifail)
      If (iwarn/=0) Then
        Write (nout,*)
        Write (nout,99999) 'Warning from f02jqf. IWARN =', iwarn
      End If

      Write (nout,*)
      If (ifail/=0) Then
        Write (nout,99999) 'Failure in f02jqf. IFAIL =', ifail
        Go To 100
      End If

      If (all(real(beta(1:2*n))>zero)) Then
        e(1:2*n) = alpha(1:2*n)/beta(1:2*n)
!       Sort eigenvalues by absolute value and then by real part.
!       Add 1000.0 to tie differences of small orders of epsilon.
        ea(1:2*n,1) = 1000.0_nag_wp + abs(e(1:2*n))
        ea(1:2*n,2) = real(e(1:2*n))
        ifail = 0
        Call m01def(ea,2*n,1,2*n,1,2,'Descending',irank,ifail)
        Call m01edf(e,1,2*n,irank,ifail)

!       Print Eigenvalues
```

```
        ifail = 0
        Call x04daf('General',' ',1,2*n,e,1,'Eigenvalues:',ifail)

        If (jobvr=='V' .Or. jobvr=='v') Then
!         Sort right eigenvectors using irank
          Do j = 1, n
            e(1:2*n) = vr(j,1:2*n)
            Call m01edf(e,1,2*n,irank,ifail)
            vr(j,1:2*n) = e(1:2*n)
          End Do
        End If
        If (jobvl=='V' .Or. jobvl=='v') Then
!         Sort left eigenvectors using irank
          Do j = 1, n
            e(1:2*n) = vl(j,1:2*n)
            Call m01edf(e,1,2*n,irank,ifail)
            vl(j,1:2*n) = e(1:2*n)
          End Do
        End If
      Else
!       Some eigenvalues are infinite
!       Print alpha and beta
        ifail = 0
        Call x04daf('General',' ',1,2*n,alpha,1,'Alpha:',ifail)
        ifail = 0
        Call x04daf('General',' ',1,2*n,beta,1,'Beta:',ifail)

      End If
      If (jobvr=='V' .Or. jobvr=='v') Then
!       Print Right Eigenvectors
        Write (nout,*)
        ifail = 0
        Call x04daf('G',' ',n,2*n,vr,n,'Right Eigenvectors (columns):',ifail)
      End If
      If (jobvl=='V' .Or. jobvl=='v') Then
!       Print Left Eigenvectors
        Write (nout,*)
        ifail = 0
        Call x04daf('G',' ',n,2*n,vl,n,'Left Eigenvectors (columns):',ifail)
      End If

      If (sense==1 .Or. sense>4) Then
        Write (nout,*)
!       Print Eigenvalues
        ifail = 0
        Call x04caf('G',' ',1,2*n,s,1,'Eigenvalue Condition numbers:',ifail)
      End If

      If (sense==3 .Or. sense==4 .Or. sense>5) Then
        t0 = maxval(bevr)
        Write (nout,*)
        Write (nout,99998)                                                   &
          'Max backward error for eigenvalues and right eigenvectors', t0
      End If

      If (sense==2 .Or. sense==4 .Or. sense==5 .Or. sense==7) Then
        t1 = maxval(bevl)
        Write (nout,*)
        Write (nout,99998)                                                   &
          'Max backward error for eigenvalues and left eigenvectors ', t1
      End If

100   Continue

99999 Format (1X,3(A,I4))
99998 Format (1X,A,1P,E11.1)
    End Program f02jqfe
```

10.2 Program Data

```
F02JQF Example Program Data
 3    1    7                            : n, scal, sense
 'V'  'V'                               : jobVL, jobVR

 (0.0, 2.0)  (0.0, 4.0)  (0.0, 4.0)
 (0.0, 6.0)  (0.0, 2.0)  (0.0, 2.0)
 (0.0, 6.0)  (0.0, 4.0)  (0.0, 2.0)  : A

 (3.0, 3.0)  (2.0, 2.0)  (1.0, 1.0)
 (2.0, 2.0)  (1.0, 1.0)  (3.0, 3.0)
 (1.0, 1.0)  (3.0, 3.0)  (2.0, 2.0)  : B

 (1.0, 0.0)  (1.0, 0.0)  (2.0, 0.0)
 (2.0, 0.0)  (3.0, 0.0)  (1.0, 0.0)
 (3.0, 0.0)  (1.0, 0.0)  (2.0, 0.0)  : C
```

10.3 Program Results

```
 F02JQF Example Program Results

 Eigenvalues:
            1          2          3          4          5          6
 1     -1.9256     0.1053    -0.6975     0.5729    -0.0496     0.3945
        1.9256     0.6975    -0.1053     0.0496    -0.5729    -0.3945

 Right Eigenvectors (columns):
         1       2       3       4       5       6
 1  -0.2108  0.3751  0.3751 -0.6593 -0.6593 -0.3478
     0.0000 -0.1877  0.1877  0.0424 -0.0424  0.0000

 2   0.7695  0.5020  0.5020  0.0302  0.0302  0.8277
     0.0000 -0.2433  0.2433  0.0197 -0.0197  0.0000

 3  -0.6028  0.7162  0.7162  0.7498  0.7498 -0.4405
    -0.0000  0.0000  0.0000  0.0000  0.0000 -0.0000

 Left Eigenvectors (columns):
         1       2       3       4       5       6
 1   0.1052  0.7816  0.7816  0.8079  0.8079  0.0358
    -0.0000  0.0000  0.0000  0.0000  0.0000  0.0000

 2   0.7381  0.5075  0.5075 -0.1124 -0.1124  0.7072
     0.0000 -0.1352  0.1352 -0.0314  0.0314  0.0000

 3  -0.6664  0.3202  0.3202 -0.5704 -0.5704 -0.7061
     0.0000 -0.1038  0.1038  0.0913 -0.0913 -0.0000

 Eigenvalue Condition numbers:
            1          2          3          4          5          6
 1      3.0717     0.6620     0.6620     2.3848     2.3848     1.7625

 Max backward error for eigenvalues and right eigenvectors    5.4E-16

 Max backward error for eigenvalues and left eigenvectors     5.5E-16
```