NAG Library Routine Document

F08WPF (ZGGEVX)

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

F08WPF (ZGGEVX) computes for a pair of n by n complex nonsymmetric matrices (A, B) the generalized eigenvalues and, optionally, the left and/or right generalized eigenvectors using the QZ algorithm.

Optionally it also computes a balancing transformation to improve the conditioning of the eigenvalues and eigenvectors, reciprocal condition numbers for the eigenvalues, and reciprocal condition numbers for the right eigenvectors.

2 Specification

```
SUBROUTINE F08WPF (BALANC, JOBVL, JOBVR, SENSE, N, A, LDA, B, LDB, ALPHA,     &
                   BETA, VL, LDVL, VR, LDVR, ILO, IHI, LSCALE, RSCALE,        &
                   ABNRM, BBNRM, RCONDE, RCONDV, WORK, LWORK, RWORK, IWORK,   &
                   BWORK, INFO)

INTEGER                 N, LDA, LDB, LDVL, LDVR, ILO, IHI, LWORK, IWORK(*),   &
                        INFO
REAL (KIND=nag_wp)      LSCALE(N), RSCALE(N), ABNRM, BBNRM, RCONDE(*),        &
                        RCONDV(*), RWORK(6*N)
COMPLEX (KIND=nag_wp) A(LDA,*), B(LDB,*), ALPHA(N), BETA(N), VL(LDVL,*),      &
                        VR(LDVR,*), WORK(max(1,LWORK))
LOGICAL                 BWORK(*)
CHARACTER(1)            BALANC, JOBVL, JOBVR, SENSE
```

The routine may be called by its LAPACK name ***zggevx***.

3 Description

A generalized eigenvalue for a pair of matrices (A, B) is a scalar λ or a ratio $\alpha/\beta = \lambda$, such that $A - \lambda B$ is singular. It is usually represented as the pair (α, β), as there is a reasonable interpretation for $\beta = 0$, and even for both being zero.

The right generalized eigenvector v_j corresponding to the generalized eigenvalue λ_j of (A, B) satisfies

$$Av_j = \lambda_j B v_j.$$

The left generalized eigenvector u_j corresponding to the generalized eigenvalue λ_j of (A, B) satisfies

$$u_j^{\mathrm{H}} A = \lambda_j u_j^{\mathrm{H}} B,$$

where u_j^{H} is the conjugate-transpose of u_j.

All the eigenvalues and, if required, all the eigenvectors of the complex generalized eigenproblem $Ax = \lambda Bx$, where A and B are complex, square matrices, are determined using the QZ algorithm. The complex QZ algorithm consists of three stages:

1. A is reduced to upper Hessenberg form (with real, non-negative subdiagonal elements) and at the same time B is reduced to upper triangular form.
2. A is further reduced to triangular form while the triangular form of B is maintained and the diagonal elements of B are made real and non-negative. This is the generalized Schur form of the pair (A, B).

This routine does not actually produce the eigenvalues λ_j, but instead returns α_j and β_j such that

$$\lambda_j = \alpha_j / \beta_j, \qquad j = 1, 2, \ldots, n.$$

The division by β_j becomes your responsibility, since β_j may be zero, indicating an infinite eigenvalue.

3. If the eigenvectors are required they are obtained from the triangular matrices and then transformed back into the original coordinate system.

For details of the balancing option, see Section 3 in F08WVF (ZGGBAL).

4 References

Anderson E, Bai Z, Bischof C, Blackford S, Demmel J, Dongarra J J, Du Croz J J, Greenbaum A, Hammarling S, McKenney A and Sorensen D (1999) *LAPACK Users' Guide* (3rd Edition) SIAM, Philadelphia http://www.netlib.org/lapack/lug

Golub G H and Van Loan C F (1996) *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

Wilkinson J H (1979) Kronecker's canonical form and the QZ algorithm *Linear Algebra Appl.* **28** 285–303

5 Parameters

1: BALANC – CHARACTER(1) *Input*

On entry: specifies the balance option to be performed.

BALANC = 'N'
Do not diagonally scale or permute.

BALANC = 'P'
Permute only.

BALANC = 'S'
Scale only.

BALANC = 'B'
Both permute and scale.

Computed reciprocal condition numbers will be for the matrices after permuting and/or balancing. Permuting does not change condition numbers (in exact arithmetic), but balancing does. In the absence of other information, BALANC = 'B' is recommended.

Constraint: BALANC = 'N', 'P', 'S' or 'B'.

2: JOBVL – CHARACTER(1) *Input*

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

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

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

3: JOBVR – CHARACTER(1) *Input*

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

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

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

4: SENSE – CHARACTER(1) *Input*

On entry: determines which reciprocal condition numbers are computed.

SENSE = 'N'
: None are computed.

SENSE = 'E'
: Computed for eigenvalues only.

SENSE = 'V'
: Computed for eigenvectors only.

SENSE = 'B'
: Computed for eigenvalues and eigenvectors.

Constraint: SENSE = 'N', 'E', 'V' or 'B'.

5: N – INTEGER *Input*

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

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

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

Note: the second dimension of the array A must be at least $\max(1, \mathrm{N})$.

On entry: the matrix A in the pair (A, B).

On exit: A has been overwritten. If JOBVL = 'V' or JOBVR = 'V' or both, then A contains the first part of the Schur form of the 'balanced' versions of the input A and B.

7: LDA – INTEGER *Input*

On entry: the first dimension of the array A as declared in the (sub)program from which F08WPF (ZGGEVX) is called.

Constraint: $\mathrm{LDA} \geq \max(1, \mathrm{N})$.

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

Note: the second dimension of the array B must be at least $\max(1, \mathrm{N})$.

On entry: the matrix B in the pair (A, B).

On exit: B has been overwritten.

9: LDB – INTEGER *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F08WPF (ZGGEVX) is called.

Constraint: $\mathrm{LDB} \geq \max(1, \mathrm{N})$.

10: ALPHA(N) – COMPLEX (KIND=nag_wp) array *Output*

On exit: see the description of BETA.

11: BETA(N) – COMPLEX (KIND=nag_wp) array *Output*

On exit: $\mathrm{ALPHA}(j)/\mathrm{BETA}(j)$, for $j = 1, 2, \ldots, \mathrm{N}$, will be the generalized eigenvalues.

Note: the quotients $\mathrm{ALPHA}(j)/\mathrm{BETA}(j)$ may easily overflow or underflow, and $\mathrm{BETA}(j)$ may even be zero. Thus, you should avoid naively computing the ratio α_j/β_j. However, $\max|\alpha_j|$ will always be less than and usually comparable with $\|\mathrm{A}\|_2$ in magnitude, and $\max|\beta_j|$ will always be less than and usually comparable with $\|\mathrm{B}\|_2$.

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

Note: the second dimension of the array VL must be at least $\max(1, \mathrm{N})$ if JOBVL = 'V', and at least 1 otherwise.

On exit: if JOBVL = 'V', the left generalized eigenvectors u_j are stored one after another in the columns of VL, in the same order as the corresponding eigenvalues. Each eigenvector will be scaled so the largest component will have $|\text{real part}| + |\text{imag. part}| = 1$.

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

13: LDVL – INTEGER *Input*

On entry: the first dimension of the array VL as declared in the (sub)program from which F08WPF (ZGGEVX) is called.

Constraints:

if JOBVL = 'V', LDVL $\geq \max(1, \mathrm{N})$;
otherwise LDVL ≥ 1.

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

Note: the second dimension of the array VR must be at least $\max(1, \mathrm{N})$ if JOBVR = 'V', and at least 1 otherwise.

On exit: if JOBVR = 'V', the right generalized eigenvectors v_j are stored one after another in the columns of VR, in the same order as the corresponding eigenvalues. Each eigenvector will be scaled so the largest component will have $|\text{real part}| + |\text{imag. part}| = 1$.

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

15: LDVR – INTEGER *Input*

On entry: the first dimension of the array VR as declared in the (sub)program from which F08WPF (ZGGEVX) is called.

Constraints:

if JOBVR = 'V', LDVR $\geq \max(1, \mathrm{N})$;
otherwise LDVR ≥ 1.

16: ILO – INTEGER *Output*
17: IHI – INTEGER *Output*

On exit: ILO and IHI are integer values such that $\mathrm{A}(i,j) = 0$ and $\mathrm{B}(i,j) = 0$ if $i > j$ and $j = 1, 2, \ldots, \mathrm{ILO} - 1$ or $i = \mathrm{IHI} + 1, \ldots, \mathrm{N}$.

If BALANC = 'N' or 'S', ILO = 1 and IHI = N.

18: LSCALE(N) – REAL (KIND=nag_wp) array *Output*

On exit: details of the permutations and scaling factors applied to the left side of A and B.

If pl_j is the index of the row interchanged with row j, and dl_j is the scaling factor applied to row j, then:

$\mathrm{LSCALE}(j) = pl_j$, for $j = 1, 2, \ldots, \mathrm{ILO} - 1$;

$\mathrm{LSCALE} = dl_j$, for $j = \mathrm{ILO}, \ldots, \mathrm{IHI}$;

$\mathrm{LSCALE} = pl_j$, for $j = \mathrm{IHI} + 1, \ldots, \mathrm{N}$.

The order in which the interchanges are made is N to $\mathrm{IHI} + 1$, then 1 to $\mathrm{ILO} - 1$.

19: RSCALE(N) – REAL (KIND=nag_wp) array *Output*

On exit: details of the permutations and scaling factors applied to the right side of A and B.

If pr_j is the index of the column interchanged with column j, and dr_j is the scaling factor applied to column j, then:

RSCALE$(j) = pr_j$, for $j = 1, 2, \ldots, \mathrm{ILO} - 1$;

if RSCALE $= dr_j$, for $j = \mathrm{ILO}, \ldots, \mathrm{IHI}$;

if RSCALE $= pr_j$, for $j = \mathrm{IHI} + 1, \ldots, \mathrm{N}$.

The order in which the interchanges are made is N to IHI $+ 1$, then 1 to ILO $- 1$.

20: ABNRM – REAL (KIND=nag_wp) *Output*

On exit: the 1-norm of the balanced matrix A.

21: BBNRM – REAL (KIND=nag_wp) *Output*

On exit: the 1-norm of the balanced matrix B.

22: RCONDE$(*)$ – REAL (KIND=nag_wp) array *Output*

Note: the dimension of the array RCONDE must be at least $\max(1, \mathrm{N})$.

On exit: if SENSE = 'E' or 'B', the reciprocal condition numbers of the eigenvalues, stored in consecutive elements of the array.

If SENSE = 'N' or 'V', RCONDE is not referenced.

23: RCONDV$(*)$ – REAL (KIND=nag_wp) array *Output*

Note: the dimension of the array RCONDV must be at least $\max(1, \mathrm{N})$.

On exit: if SENSE = 'V' or 'B', the estimated reciprocal condition numbers of the selected eigenvectors, stored in consecutive elements of the array.

If SENSE = 'N' or 'E', RCONDV is not referenced.

24: WORK$(\max(1, \mathrm{LWORK}))$ – COMPLEX (KIND=nag_wp) array *Workspace*

On exit: if INFO $= 0$, the real part of WORK(1) contains the minimum value of LWORK required for optimal performance.

25: LWORK – INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F08WPF (ZGGEVX) is called.

If LWORK $= -1$, a workspace query is assumed; the routine only calculates the optimal size of the WORK array, returns this value as the first entry of the WORK array, and no error message related to LWORK is issued.

Suggested value: for optimal performance, LWORK must generally be larger than the minimum; increase workspace by, say, $nb \times \mathrm{N}$, where nb is the optimal ***block size***.

Constraints:

if SENSE = 'N', LWORK $\geq \max(1, 2 \times \mathrm{N})$;
if SENSE = 'E', LWORK $\geq \max(1, 4 \times \mathrm{N})$;
if SENSE = 'B' or 'V', LWORK $\geq \max(1, 2 \times \mathrm{N} \times \mathrm{N} + 2 \times \mathrm{N})$.

26: RWORK$(6 \times \mathrm{N})$ – REAL (KIND=nag_wp) array *Workspace*

Real workspace.

27: IWORK$(*)$ – INTEGER array *Workspace*

Note: the dimension of the array IWORK must be at least $\max(1, \mathrm{N} + 2)$.

If SENSE = 'E', IWORK is not referenced.

28: BWORK($*$) – LOGICAL array *Workspace*

Note: the dimension of the array BWORK must be at least $\max(1, \mathrm{N})$.

If SENSE = 'N', BWORK is not referenced.

29: INFO – INTEGER *Output*

On exit: INFO = 0 unless the routine detects an error (see Section 6).

6 Error Indicators and Warnings

Errors or warnings detected by the routine:

INFO < 0

If INFO $= -i$, argument i had an illegal value. An explanatory message is output, and execution of the program is terminated.

INFO = 1 to N

The QZ iteration failed. No eigenvectors have been calculated, but ALPHA(j) and BETA(j) should be correct for $j = \mathrm{INFO} + 1, \ldots, \mathrm{N}$.

INFO = N + 1

Unexpected error returned from F08XSF (ZHGEQZ).

INFO = N + 2

Error returned from F08YXF (ZTGEVC).

7 Accuracy

The computed eigenvalues and eigenvectors are exact for a nearby matrices $(A + E)$ and $(B + F)$, where

$$\|(E, F)\|_F = O(\epsilon)\|(A, B)\|_F,$$

and ϵ is the ***machine precision***.

An approximate error bound on the chordal distance between the ith computed generalized eigenvalue w and the corresponding exact eigenvalue λ is

$$\epsilon \times \|\mathrm{ABNRM}, \mathrm{BBNRM}\|_2/\mathrm{RCONDE}(i).$$

An approximate error bound for the angle between the ith computed eigenvector VL(i) or VR(i) is given by

$$\epsilon \times \|\mathrm{ABNRM}, \mathrm{BBNRM}\|_2/\mathrm{RCONDV}(i).$$

For further explanation of the reciprocal condition numbers RCONDE and RCONDV, see Section 4.11 of Anderson *et al.* (1999).

Note: interpretation of results obtained with the QZ algorithm often requires a clear understanding of the effects of small changes in the original data. These effects are reviewed in Wilkinson (1979), in relation to the significance of small values of α_j and β_j. It should be noted that if α_j and β_j are **both** small for any j, it may be that no reliance can be placed on **any** of the computed eigenvalues $\lambda_i = \alpha_i/\beta_i$. You are recommended to study Wilkinson (1979) and, if in difficulty, to seek expert advice on determining the sensitivity of the eigenvalues to perturbations in the data.

8 Further Comments

The total number of floating point operations is proportional to n^3.

The real analogue of this routine is F08WBF (DGGEVX).

9 Example

This example finds all the eigenvalues and right eigenvectors of the matrix pair (A, B), where

$$A = \begin{pmatrix} -21.10 - 22.50i & 53.50 - 50.50i & -34.50 + 127.50i & 7.50 + \ 0.50i \\ -0.46 - \ 7.78i & -3.50 - 37.50i & -15.50 + \ 58.50i & -10.50 - \ 1.50i \\ 4.30 - \ 5.50i & 39.70 - 17.10i & -68.50 + \ 12.50i & -7.50 - \ 3.50i \\ 5.50 + \ 4.40i & 14.40 + 43.30i & -32.50 - \ 46.00i & -19.00 - 32.50i \end{pmatrix}$$

and

$$B = \begin{pmatrix} 1.00 - 5.00i & 1.60 + 1.20i & -3.00 + 0.00i & 0.00 - 1.00i \\ 0.80 - 0.60i & 3.00 - 5.00i & -4.00 + 3.00i & -2.40 - 3.20i \\ 1.00 + 0.00i & 2.40 + 1.80i & -4.00 - 5.00i & 0.00 - 3.00i \\ 0.00 + 1.00i & -1.80 + 2.40i & 0.00 - 4.00i & 4.00 - 5.00i \end{pmatrix},$$

together with estimates of the condition number and forward error bounds for each eigenvalue and eigenvector. The option to balance the matrix pair is used.

Note that the block size (NB) of 64 assumed in this example is not realistic for such a small problem, but should be suitable for large problems.

9.1 Program Text

```
    Program f08wpfe

!     F08WPF Example Program Text

!     Mark 24 Release. NAG Copyright 2012.

!     .. Use Statements ..
      Use nag_library, Only: f06bnf, nag_wp, x02ajf, x02amf, zggevx
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter                 :: nb = 64, nin = 5, nout = 6
!     .. Local Scalars ..
      Complex (Kind=nag_wp)              :: eig
      Real (Kind=nag_wp)                 :: abnorm, abnrm, bbnrm, eps, small, tol
      Integer                            :: i, ihi, ilo, info, j, lda, ldb,      &
                                            ldvr, lwork, n
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: a(:,:), alpha(:), b(:,:), beta(:), &
                                              vr(:,:), work(:)
      Complex (Kind=nag_wp)              :: dummy(1,1)
      Real (Kind=nag_wp), Allocatable    :: lscale(:), rconde(:), rcondv(:),     &
                                            rscale(:), rwork(:)
      Integer, Allocatable               :: iwork(:)
      Logical, Allocatable               :: bwork(:)
!     .. Intrinsic Procedures ..
      Intrinsic                          :: abs, max, nint, real
!     .. Executable Statements ..
      Write (nout,*) 'F08WPF Example Program Results'
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) n
      lda = n
      ldb = n
      ldvr = n
      Allocate (a(lda,n),alpha(n),b(ldb,n),beta(n),vr(ldvr,n),lscale(n), &
        rconde(n),rcondv(n),rscale(n),rwork(6*n),iwork(n+2),bwork(n))

!     Use routine workspace query to get optimal workspace.
      lwork = -1
!     The NAG name equivalent of zggevx is f08wpf
      Call zggevx('Balance','No vectors (left)','Vectors (right)', &
        'Both reciprocal condition numbers',n,a,lda,b,ldb,alpha,beta,dummy,1, &
        vr,ldvr,ilo,ihi,lscale,rscale,abnrm,bbnrm,rconde,rcondv,dummy,lwork, &
```

```
        rwork,iwork,bwork,info)

!     Make sure that there is enough workspace for blocksize nb.
      lwork = max((nb+2*n)*n,nint(real(dummy(1,1))))
      Allocate (work(lwork))

!     Read in the matrices A and B

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

!     Solve the generalized eigenvalue problem

!     The NAG name equivalent of zggevx is f08wpf
      Call zggevx('Balance','No vectors (left)','Vectors (right)', &
        'Both reciprocal condition numbers',n,a,lda,b,ldb,alpha,beta,dummy,1, &
        vr,ldvr,ilo,ihi,lscale,rscale,abnrm,bbnrm,rconde,rcondv,work,lwork, &
        rwork,iwork,bwork,info)

      If (info>0) Then
        Write (nout,*)
        Write (nout,99999) 'Failure in ZGGEVX. INFO =', info
      Else

!       Compute the machine precision, the safe range parameter
!       SMALL and sqrt(ABNRM**2+BBNRM**2)

        eps = x02ajf()
        small = x02amf()
        abnorm = f06bnf(abnrm,bbnrm)
        tol = eps*abnorm

!       Print out eigenvalues and vectors and associated condition
!       number and bounds

        Write (nout,*)
        Write (nout,*) 'Eigenvalues'
        Write (nout,*)
        Write (nout,*) '         Eigenvalue           rcond    error'

        Do j = 1, n

!         Print out information on the jth eigenvalue

          If ((abs(alpha(j)))*small>=abs(beta(j))) Then
            If (rconde(j)>0.0_nag_wp) Then
              If (tol/rconde(j)<100.0_nag_wp*eps) Then
                Write (nout,99995) j, rconde(j), '-'
              Else
                Write (nout,99994) j, rconde(j), tol/rconde(j)
              End If
            Else
              Write (nout,99995) j, rconde(j), 'Inf'
            End If
          Else
            eig = alpha(j)/beta(j)
            If (rconde(j)>0.0_nag_wp) Then
              If (tol/rconde(j)<100.0_nag_wp*eps) Then
                Write (nout,99998) j, eig, rconde(j), '-'
              Else
                Write (nout,99997) j, eig, rconde(j), tol/rconde(j)
              End If
            Else
              Write (nout,99998) j, eig, rconde(j), 'Inf'
            End If
          End If

        End Do


        Write (nout,*)
```

```
        Write (nout,*) 'Eigenvectors'
        Write (nout,*)
        Write (nout,*) '        Eigenvector          rcond    error'

        Do j = 1, n

!         Print information on jth eigenvector
          Write (nout,*)

!         Make first real part component be positive
          If (real(vr(1,j))<0.0_nag_wp) Then
            vr(1:n,j) = -vr(1:n,j)
          End If

          If (rcondv(j)>0.0_nag_wp) Then
            If (tol/rcondv(j)<100.0_nag_wp*eps) Then
              Write (nout,99998) j, vr(1,j), rcondv(j), '-'
            Else
              Write (nout,99997) j, vr(1,j), rcondv(j), tol/rcondv(j)
            End If
          Else
            Write (nout,99998) j, vr(1,j), rcondv(j), 'Inf'
          End If

          Write (nout,99996) vr(2:n,j)

        End Do

        Write (nout,*)
        Write (nout,*) 'Errors below 100*machine precision are not displayed'
      End If

99999 Format (1X,A,I4)
99998 Format (1X,I2,1X,'(',1P,E11.4,',',E11.4,')',1X,0P,F7.4,4X,A)
99997 Format (1X,I2,1X,'(',1P,E11.4,',',E11.4,')',1X,0P,F7.4,1X,1P,E8.1)
99996 Format (1X,3X,'(',1P,E11.4,',',E11.4,')')
99995 Format (1X,I2,1X,'  Infinite or undetermined',1X,0P,F7.4,4X,A)
99994 Format (1X,I2,1X,'  Infinite or undetermined',1X,0P,F7.4,1X,1P,E8.1)

    End Program f08wpfe
```

9.2 Program Data

```
F08WPF Example Program Data
  4                                                         : Value of N
  (-21.10,-22.50) ( 53.50,-50.50) (-34.50,127.50) (  7.50,  0.50)
  ( -0.46, -7.78) ( -3.50,-37.50) (-15.50, 58.50) (-10.50, -1.50)
  (  4.30, -5.50) ( 39.70,-17.10) (-68.50, 12.50) ( -7.50, -3.50)
  (  5.50,  4.40) ( 14.40, 43.30) (-32.50,-46.00) (-19.00,-32.50) : End of A
  (  1.00, -5.00) (  1.60,  1.20) ( -3.00,  0.00) (  0.00, -1.00)
  (  0.80, -0.60) (  3.00, -5.00) ( -4.00,  3.00) ( -2.40, -3.20)
  (  1.00,  0.00) (  2.40,  1.80) ( -4.00, -5.00) (  0.00, -3.00)
  (  0.00,  1.00) ( -1.80,  2.40) (  0.00, -4.00) (  4.00, -5.00) : End of B
```

9.3 Program Results

```
 F08WPF Example Program Results

 Eigenvalues

         Eigenvalue           rcond    error
  1 ( 3.0000E+00,-9.0000E+00)  0.5108    -
  2 ( 2.0000E+00,-5.0000E+00)  0.3756    -
  3 ( 3.0000E+00,-1.0000E+00)  0.1340  1.2E-14
  4 ( 4.0000E+00,-5.0000E+00)  0.6195    -

 Eigenvectors

         Eigenvector          rcond    error
```

```
  1 ( 7.3959E-01, 2.6041E-01)  0.0471  3.4E-14
    ( 1.4958E-01,-4.7086E-02)
    ( 4.7086E-02, 1.4958E-01)
    (-1.4958E-01, 4.7086E-02)

  2 ( 6.2369E-01, 3.7631E-01)  0.0662  2.4E-14
    ( 4.1414E-03,-4.1806E-04)
    ( 3.9203E-02, 2.3654E-02)
    (-2.3654E-02, 3.9203E-02)

  3 ( 4.8804E-01, 5.1196E-01)  0.1723    -
    ( 1.3952E-01, 2.3350E-02)
    ( 1.4048E-01,-1.6650E-02)
    ( 1.6650E-02, 1.4048E-01)

  4 ( 3.6600E-01,-6.3400E-01)  0.0346  4.6E-14
    (-9.7340E-04,-8.0756E-03)
    (-1.2200E-02, 2.1133E-02)
    ( 9.8623E-02, 5.6933E-02)

 Errors below 100*machine precision are not displayed
```