NAG Library Routine Document

F08WBF (DGGEVX)

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

F08WBF (DGGEVX) computes for a pair of n by n real 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 F08WBF (BALANC, JOBVL, JOBVR, SENSE, N, A, LDA, B, LDB,        &
                   ALPHAR, ALPHAI, BETA, VL, LDVL, VR, LDVR, ILO, IHI,    &
                   LSCALE, RSCALE, ABNRM, BBNRM, RCONDE, RCONDV, WORK,    &
                   LWORK, IWORK, BWORK, INFO)
INTEGER            N, LDA, LDB, LDVL, LDVR, ILO, IHI, LWORK, IWORK(*),    &
                   INFO
REAL (KIND=nag_wp) A(LDA,*), B(LDB,*), ALPHAR(N), ALPHAI(N), BETA(N),     &
                   VL(LDVL,*), VR(LDVR,*), LSCALE(N), RSCALE(N),          &
                   ABNRM, BBNRM, RCONDE(*), RCONDV(*),                    &
                   WORK(max(1,LWORK))
LOGICAL            BWORK(*)
CHARACTER(1)       BALANC, JOBVL, JOBVR, SENSE
```

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

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

$$Av_j = \lambda_j Bv_j.$$

The left eigenvector u_j corresponding to the 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 generalized eigenproblem $Ax = \lambda Bx$, where A and B are real, square matrices, are determined using the QZ algorithm. The QZ algorithm consists of four stages:

1. A is reduced to upper Hessenberg form and at the same time B is reduced to upper triangular form.
2. A is further reduced to quasi-triangular form while the triangular form of B is maintained. This is the real generalized Schur form of the pair (A, B).
3. The quasi-triangular form of A is reduced to triangular form and the eigenvalues extracted. This routine does not actually produce the eigenvalues λ_j, but instead returns α_j and β_j such that

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

The division by β_j becomes your responsibility, since β_j may be zero, indicating an infinite eigenvalue. Pairs of complex eigenvalues occur with α_j/β_j and α_{j+1}/β_{j+1} complex conjugates, even though α_j and α_{j+1} are not conjugate.

4. 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 F08WHF (DGGBAL).

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 (2012) *Matrix Computations* (4th 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 Arguments

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: $\mathrm{A}(\mathrm{LDA},*)$ – REAL (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 real 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 F08WBF (DGGEVX) is called.

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

8: $\mathrm{B}(\mathrm{LDB},*)$ – REAL (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 F08WBF (DGGEVX) is called.

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

10: ALPHAR(N) – REAL (KIND=nag_wp) array *Output*

On exit: the element $\mathrm{ALPHAR}(j)$ contains the real part of α_j.

11: ALPHAI(N) – REAL (KIND=nag_wp) array *Output*

On exit: the element $\mathrm{ALPHAI}(j)$ contains the imaginary part of α_j.

12: BETA(N) – REAL (KIND=nag_wp) array *Output*

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

If $\mathrm{ALPHAI}(j)$ is zero, then the jth eigenvalue is real; if positive, then the jth and $(j+1)$st eigenvalues are a complex conjugate pair, with $\mathrm{ALPHAI}(j+1)$ negative.

Note: the quotients $\mathrm{ALPHAR}(j)/\mathrm{BETA}(j)$ and $\mathrm{ALPHAI}(j)/\mathrm{BETA}(j)$ 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 $\|A\|_2$ in magnitude, and $\max|\beta_j|$ will always be less than and usually comparable with $\|B\|_2$.

13: VL(LDVL, $*$) – REAL (KIND=nag_wp) array *Output*

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

14: LDVL – INTEGER *Input*

On entry: the first dimension of the array VL as declared in the (sub)program from which F08WBF (DGGEVX) is called.

Constraints:

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

15: VR(LDVR, $*$) – REAL (KIND=nag_wp) array *Output*

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

16: LDVR – INTEGER *Input*

On entry: the first dimension of the array VR as declared in the (sub)program from which F08WBF (DGGEVX) is called.

Constraints:

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

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

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

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

19: 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:

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

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

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

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

20: 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:

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

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

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

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

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

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

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

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

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

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

On exit: if SENSE = 'E' or 'B', the reciprocal condition numbers of the eigenvalues, stored in consecutive elements of the array. For a complex conjugate pair of eigenvalues two consecutive elements of RCONDE are set to the same value. Thus RCONDE(j), RCONDV(j), and the jth columns of VL and VR all correspond to the jth eigenpair.

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

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

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

On exit: if SENSE = 'V' or 'B', the estimated reciprocal condition numbers of the eigenvectors, stored in consecutive elements of the array. For a complex eigenvector two consecutive elements of RCONDV are set to the same value.

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

25: WORK($\max(1, \text{LWORK})$) – REAL (KIND=nag_wp) array *Workspace*

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

26: LWORK – INTEGER *Input*

On entry: the dimension of the array WORK as declared in the (sub)program from which F08WBF (DGGEVX) 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, $nb \times \text{N}$, where nb is the optimal ***block size***.

Constraints:

if SENSE = 'N',

if BALANC = 'N' or 'P' and JOBVL = 'N' and JOBVR = 'N', $\text{LWORK} \geq \max(1, 2 \times \text{N})$;
otherwise $\text{LWORK} \geq \max(1, 6 \times \text{N})$.;

if SENSE = 'E', $\text{LWORK} \geq \max(1, 10 \times \text{N})$;

if SENSE = 'B' or SENSE = 'V', $\text{LWORK} \geq \max(10 \times \text{N}, 2 \times \text{N} \times (\text{N} + 4) + 16)$.

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

Note: the dimension of the array IWORK must be at least $\mathrm{N}+6$.

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

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 ALPHAR(j), ALPHAI(j), and BETA(j) should be correct for $j=\mathrm{INFO}+1,\dots,\mathrm{N}$.

INFO $=\mathrm{N}+1$

Unexpected error returned from F08XEF (DHGEQZ).

INFO $=\mathrm{N}+2$

Error returned from F08YKF (DTGEVC).

7 Accuracy

The computed eigenvalues and eigenvectors are exact for 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 u_j or v_j 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 Parallelism and Performance

F08WBF (DGGEVX) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

F08WBF (DGGEVX) 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

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

The complex analogue of this routine is F08WPF (ZGGEVX).

10 Example

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

$$A = \begin{pmatrix} 3.9 & 12.5 & -34.5 & -0.5 \\ 4.3 & 21.5 & -47.5 & 7.5 \\ 4.3 & 21.5 & -43.5 & 3.5 \\ 4.4 & 26.0 & -46.0 & 6.0 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 1.0 & 2.0 & -3.0 & 1.0 \\ 1.0 & 3.0 & -5.0 & 4.0 \\ 1.0 & 3.0 & -4.0 & 3.0 \\ 1.0 & 3.0 & -4.0 & 4.0 \end{pmatrix},$$

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.

10.1 Program Text

```
    Program f08wbfe

!     F08WBF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: dggevx, dnrm2, m01daf, m01eaf, nag_wp, x02ajf,    &
                             x02amf
!     .. 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)               :: abnrm, bbnrm, eps, jswap, rcnd,      &
                                          scal_i, scal_r, small
      Integer                          :: i, ifail, ihi, ilo, info, j, k, lda, &
                                          ldb, ldvr, lwork, n
      Logical                          :: pair
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: a(:,:), alphai(:), alphar(:),        &
                                          b(:,:), beta(:), lscale(:),          &
                                          rconde(:), rcondv(:), rscale(:),     &
                                          vr(:,:), vr_row(:), work(:)
      Real (Kind=nag_wp)               :: dummy(1,1)
      Integer, Allocatable             :: iwork(:)
      Logical, Allocatable             :: bwork(:)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: abs, all, cmplx, max, maxloc, nint,  &
```

```
                                                 sqrt, sum
!     .. Executable Statements ..
      Write (nout,*) 'F08WBF Example Program Results'
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) n
      lda = n
      ldb = n
      ldvr = n
      Allocate (a(lda,n),alphai(n),alphar(n),b(ldb,n),beta(n),lscale(n),       &
        rconde(n),rcondv(n),rscale(n),vr(ldvr,n),iwork(n+6),bwork(n))

!     Use routine workspace query to get optimal workspace.
      lwork = -1
!     The NAG name equivalent of dggevx is f08wbf
      Call dggevx('Balance','No vectors (left)','Vectors (right)',             &
        'Both reciprocal condition numbers',n,a,lda,b,ldb,alphar,alphai,beta,  &
        dummy,1,vr,ldvr,ilo,ihi,lscale,rscale,abnrm,bbnrm,rconde,rcondv,dummy, &
        lwork,iwork,bwork,info)

!     Make sure that there is enough workspace for block size nb.
      lwork = max((nb+2*n)*n,nint(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 dggevx is f08wbf
      Call dggevx('Balance','No vectors (left)','Vectors (right)',             &
        'Both reciprocal condition numbers',n,a,lda,b,ldb,alphar,alphai,beta,  &
        dummy,1,vr,ldvr,ilo,ihi,lscale,rscale,abnrm,bbnrm,rconde,rcondv,work,  &
        lwork,iwork,bwork,info)

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

!       Compute the machine precision and the safe range parameter
!       small

        eps = x02ajf()
        small = x02amf()

!       If beta(:) > eps, Order eigenvalues by ascending real parts
        If (all(abs(beta(1:n))>eps)) Then
          work(1:n) = alphar(1:n)/beta(1:n)
          ifail = 0
          Call m01daf(work,1,n,'Ascending',iwork,ifail)
          Call m01eaf(alphar,1,n,iwork,ifail)
          Call m01eaf(alphai,1,n,iwork,ifail)
          Call m01eaf(beta,1,n,iwork,ifail)
!         Order the eigenvectors in the same way
          Allocate (vr_row(n))
          Do j = 1, n
            vr_row(1:n) = vr(j,1:n)
            Call m01eaf(vr_row,1,n,iwork,ifail)
            vr(j,1:n) = vr_row(1:n)
          End Do
          Deallocate (vr_row)
        End If

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

        pair = .False.
        Do j = 1, n
```

```
!         Print out information on the j-th eigenvalue

          Write (nout,*)
          If ((abs(alphar(j))+abs(alphai(j)))*small>=abs(beta(j))) Then
            Write (nout,99998) 'Eigenvalue(', j, ')',                        &
              ' is numerically infinite or undetermined', 'ALPHAR(', j,      &
              ') = ', alphar(j), ', ALPHAI(', j, ') = ', alphai(j), ', BETA(', &
              j, ') = ', beta(j)
          Else
            If (.Not. pair) Then
              jswap = 1.0_nag_wp
              If (alphai(j)>0.0_nag_wp) Then
                jswap = -jswap
              End If
            End If
            If (alphai(j)==0.0E0_nag_wp) Then
              Write (nout,99997) 'Eigenvalue(', j, ') = ', alphar(j)/beta(j)
            Else
              eig = cmplx(alphar(j),jswap*alphai(j),kind=nag_wp)/              &
                cmplx(beta(j),kind=nag_wp)
              Write (nout,99996) 'Eigenvalue(', j, ') = ', eig
            End If
          End If
          rcnd = rconde(j)
          Write (nout,*)
          Write (nout,99995) '  Reciprocal condition number = ', rcnd

!         Print out information on the j-th eigenvector

          Write (nout,*)
          Write (nout,99994) 'Eigenvector(', j, ')'
          If (alphai(j)==0.0E0_nag_wp) Then
!           Let largest element be positive
            work(1:n) = abs(vr(1:n,j))
            k = maxloc(work(1:n),1)
            If (vr(k,j)<0.0_nag_wp) Then
              vr(1:n,j) = -vr(1:n,j)/dnrm2(n,vr(1,j),1)
            End If
            Write (nout,99993)(vr(i,j),i=1,n)
          Else
            If (pair) Then
              Write (nout,99992)(vr(i,j-1),-jswap*vr(i,j),i=1,n)
            Else
!             Let largest element be real (and positive).
              work(1:n) = vr(1:n,j)**2 + vr(1:n,j+1)**2
              k = maxloc(work(1:n),1)
              scal_r = vr(k,j)/sqrt(work(k))/sqrt(sum(work(1:n)))
              scal_i = -vr(k,j+1)/sqrt(work(k))/sqrt(sum(work(1:n)))
              work(1:n) = vr(1:n,j)
              vr(1:n,j) = scal_r*work(1:n) - scal_i*vr(1:n,j+1)
              vr(1:n,j+1) = scal_r*vr(1:n,j+1) + scal_i*work(1:n)
              vr(k,j+1) = 0.0_nag_wp
              Write (nout,99992)(vr(i,j),jswap*vr(i,j+1),i=1,n)
            End If
            pair = .Not. pair
          End If
          rcnd = rcondv(j)
          Write (nout,*)
          Write (nout,99995) '  Reciprocal condition number = ', rcnd
        End Do

      End If

99999 Format (1X,A,I4)
99998 Format (/,1X,A,I2,2A,/,1X,2(A,I2,A,F11.5),A,I2,A,F11.5)
99997 Format (/,1X,A,I2,A,F11.5)
99996 Format (/,1X,A,I2,A,'(',F11.5,',',F11.5,')')
```

```
99995 Format (1X,A,1P,E8.1)
99994 Format (1X,A,I2,A)
99993 Format (1X,F11.5)
99992 Format (1X,'(',F11.5,',',F11.5,')')
    End Program f08wbfe
```

10.2 Program Data

```
F08WBF Example Program Data
   4                     :Value of N
   3.9  12.5 -34.5  -0.5
   4.3  21.5 -47.5   7.5
   4.3  21.5 -43.5   3.5
   4.4  26.0 -46.0   6.0 :End of matrix A
   1.0   2.0  -3.0   1.0
   1.0   3.0  -5.0   4.0
   1.0   3.0  -4.0   3.0
   1.0   3.0  -4.0   4.0 :End of matrix B
```

10.3 Program Results

```
 F08WBF Example Program Results


 Eigenvalue( 1) =     2.00000

   Reciprocal condition number =  9.5E-02

 Eigenvector( 1)
     0.99606
     0.00569
     0.06261
     0.06261

   Reciprocal condition number =  1.3E-01


 Eigenvalue( 2) = (    3.00000,   -4.00000)

   Reciprocal condition number =  1.7E-01

 Eigenvector( 2)
 (    0.94491,   -0.00000)
 (    0.18898,    0.00000)
 (    0.11339,    0.15119)
 (    0.11339,    0.15119)

   Reciprocal condition number =  3.8E-02


 Eigenvalue( 3) = (    3.00000,    4.00000)

   Reciprocal condition number =  1.7E-01

 Eigenvector( 3)
 (    0.94491,    0.00000)
 (    0.18898,   -0.00000)
 (    0.11339,   -0.15119)
 (    0.11339,   -0.15119)

   Reciprocal condition number =  3.8E-02


 Eigenvalue( 4) =     4.00000

   Reciprocal condition number =  5.1E-01

 Eigenvector( 4)
     0.98752
```

```
  0.01097
 -0.03292
  0.15361

Reciprocal condition number =  7.1E-02
```