

# 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  $\lambda$  or a ratio  $\alpha/\beta = \lambda$ , such that  $A - \lambda B$  is singular. It is usually represented as the pair  $(\alpha, \beta)$ , 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  $\lambda_j$  of  $(A, B)$  satisfies

$$Av_j = \lambda_j Bv_j.$$

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

$$u_j^H A = \lambda_j u_j^H B,$$

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  $\lambda_j$ , but instead returns  $\alpha_j$  and  $\beta_j$  such that

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

The division by  $\beta_j$  becomes your responsibility, since  $\beta_j$  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 (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:*  $N \geq 0$ .
- 6: A(LDA,\*) – COMPLEX (KIND=nag\_wp) array *Input/Output*  
**Note:** the second dimension of the array  $A$  must be at least  $\max(1, N)$ .  
*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:*  $LDA \geq \max(1, N)$ .
- 8: B(LDB,\*) – COMPLEX (KIND=nag\_wp) array *Input/Output*  
**Note:** the second dimension of the array  $B$  must be at least  $\max(1, N)$ .  
*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:*  $LDB \geq \max(1, N)$ .
- 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:*  $\text{ALPHA}(j)/\text{BETA}(j)$ , for  $j = 1, 2, \dots, N$ , will be the generalized eigenvalues.  
**Note:** the quotients  $\text{ALPHA}(j)/\text{BETA}(j)$  may easily overflow or underflow, and  $\text{BETA}(j)$  may even be zero. Thus, you should avoid naively computing the ratio  $\alpha_j/\beta_j$ . 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  $\|B\|_2$ .

- 12: VL(LDVL,\*) – COMPLEX (KIND=nag\_wp) array Output  
**Note:** the second dimension of the array VL must be at least  $\max(1, N)$  if  $\text{JOBVL} = 'V'$ , and at least 1 otherwise.  
*On exit:* if  $\text{JOBVL} = 'V'$ , 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  $\text{JOBVL} = 'N'$ , 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  $\text{JOBVL} = 'V'$ ,  $\text{LDVL} \geq \max(1, N)$ ;  
     otherwise  $\text{LDVL} \geq 1$ .
- 14: VR(LDVR,\*) – COMPLEX (KIND=nag\_wp) array Output  
**Note:** the second dimension of the array VR must be at least  $\max(1, N)$  if  $\text{JOBVR} = 'V'$ , and at least 1 otherwise.  
*On exit:* if  $\text{JOBVR} = 'V'$ , 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  $\text{JOBVR} = 'N'$ , 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  $\text{JOBVR} = 'V'$ ,  $\text{LDVR} \geq \max(1, N)$ ;  
     otherwise  $\text{LDVR} \geq 1$ .
- 16: ILO – INTEGER Output  
 17: 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  $j = 1, 2, \dots, \text{ILO} - 1$  or  $i = \text{IHI} + 1, \dots, N$ .  
 If  $\text{BALANC} = 'N'$  or  $'S'$ ,  $\text{ILO} = 1$  and  $\text{IHI} = N$ .
- 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:  
      $\text{LSCALE}(j) = pl_j$ , for  $j = 1, 2, \dots, \text{ILO} - 1$ ;  
      $\text{LSCALE} = dl_j$ , for  $j = \text{ILO}, \dots, \text{IHI}$ ;  
      $\text{LSCALE} = pl_j$ , for  $j = \text{IHI} + 1, \dots, N$ .  
 The order in which the interchanges are made is  $N$  to  $\text{IHI} + 1$ , then 1 to  $\text{ILO} - 1$ .
- 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:

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

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

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

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  $\max(1, N)$ .

*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  $\max(1, N)$ .

*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, 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,  $nb \times N$ , where  $nb$  is the optimal **block size**.

*Constraints:*

$$\text{if SENSE} = \text{'N'}, \text{LWORK} \geq \max(1, 2 \times N);$$

$$\text{if SENSE} = \text{'E'}, \text{LWORK} \geq \max(1, 4 \times N);$$

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

26: RWORK(6 × 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  $\max(1, N + 2)$ .  
 If SENSE = 'E', IWORK is not referenced.
- 28: BWORK(\*) – LOGICAL array *Workspace*  
**Note:** the dimension of the array BWORK must be at least  $\max(1, N)$ .  
 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 ALPHA( $j$ ) and BETA( $j$ ) should be correct for  $j = \text{INFO} + 1, \dots, N$ .

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 nearby matrices  $(A + E)$  and  $(B + F)$ , where

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

and  $\epsilon$  is the *machine precision*.

An approximate error bound on the chordal distance between the  $i$ th computed generalized eigenvalue  $w$  and the corresponding exact eigenvalue  $\lambda$  is

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

An approximate error bound for the angle between the  $i$ th computed eigenvector  $u_j$  or  $v_j$  is given by

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

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  $\alpha_j$  and  $\beta_j$ . It should be noted that if  $\alpha_j$  and  $\beta_j$  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

F08WPF (ZGGEVX) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

F08WPF (ZGGEVX) 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 real analogue of this routine is F08WBF (DGGEVX).

## 10 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.

### 10.1 Program Text

```

Program f08wpfe

!      F08WPF Example Program Text

!      Mark 26 Release. NAG Copyright 2016.

!      .. Use Statements ..
Use nag_library, Only: f06bnf, nag_wp, x02ajf, x02amf, zggev
!      .. Implicit None Statement ..
Implicit None
!      .. Parameters ..
Integer, Parameter          :: nb = 64, nin = 5, nout = 6
!      .. Local Scalars ..
Complex (Kind=nag_wp)      :: eig, scal
Real (Kind=nag_wp)        :: abnorm, abnorm, bbnrm, eps, small, &
                             tol
Integer                    :: i, ihi, ilo, info, j, k, 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, maxloc, 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 zggevz is f08wpf
Call zggevz('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,abnorm,bbnorm,rconde,rcondv,dummy,lwork, &
           rwork,iwork,bwork,info)
! Make sure that there is enough workspace for block size 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 zggevz is f08wpf
Call zggevz('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,abnorm,bbnorm,rconde,rcondv,work,lwork, &
           rwork,iwork,bwork,info)
If (info>0) Then
  Write (nout,*)
  Write (nout,99999) 'Failure in ZGGEVZ. INFO =', info
Else
! Compute the machine precision, the safe range parameter
! SMALL and sqrt(ABNRM**2+BBNRM**2)
eps = x02ajf()
small = x02amf()
abnorm = f06bnf(abnorm,bbnorm)
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 j-th 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 j-th eigenvector
    Write (nout,*)

!      Re-normalize eigenvector, largest absolute element real (=1)
    rwork(1:n) = abs(vr(1:n,j))
    k = maxloc(rwork(1:n),1)
    scal = (1.0_nag_wp,0.0_nag_wp)/vr(k,j)
    vr(1:n,j) = vr(1:n,j)*scal

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

End Program f08wpfe

```

## 10.2 Program Data

F08WPF Example Program Data

```

4
(-21.10,-22.50) ( 53.50,-50.50) (-34.50,127.50) ( 7.50, 0.50) : Value of N
( -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

```

## 10.3 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	( 1.0000E+00, 0.0000E+00) ( 1.6000E-01,-1.2000E-01) ( 1.2000E-01, 1.6000E-01) (-1.6000E-01, 1.2000E-01)	0.0471	3.4E-14
2	( 1.0000E+00, 5.5511E-17) ( 4.5714E-03,-3.4286E-03) ( 6.2857E-02,-2.0123E-16) ( 1.8041E-16, 6.2857E-02)	0.0662	2.4E-14
3	( 1.0000E+00, 0.0000E+00) ( 1.6000E-01,-1.2000E-01) ( 1.2000E-01,-1.6000E-01) ( 1.6000E-01, 1.2000E-01)	0.1723	-
4	( 1.0000E+00, 0.0000E+00) ( 8.8889E-03,-6.6667E-03) (-3.3333E-02,-2.0123E-16) ( 3.3307E-16, 1.5556E-01)	0.0346	4.6E-14

Errors below 100\*machine precision are not displayed

---