

# NAG Library Routine Document

## F08WAF (DGGEV)

**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

F08WAF (DGGEV) 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. F08WAF (DGGEV) is marked as *deprecated* by LAPACK; the replacement routine is F08WCF (DGGEV3) which makes better use of level 3 BLAS.

### 2 Specification

```
SUBROUTINE F08WAF (JOBVL, JOBVR, N, A, LDA, B, LDB, ALPHAR, ALPHAI,      &
                  BETA, VL, LDVL, VR, LDVR, WORK, LWORK, INFO)
INTEGER           N, LDA, LDB, LDVL, LDVR, LWORK, INFO
REAL (KIND=nag_wp) A(LDA,*), B(LDB,*), ALPHAR(N), ALPHAI(N), BETA(N),  &
                  VL(LDVL,*), VR(LDVR,*), WORK(max(1,LWORK))
CHARACTER(1)     JOBVL, JOBVR
```

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

### 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 eigenvector  $v_j$  corresponding to the eigenvalue  $\lambda_j$  of  $(A, B)$  satisfies

$$Av_j = \lambda_j Bv_j.$$

The left eigenvector  $u_j$  corresponding to the 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 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  $\lambda_j$ , 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. Pairs of complex eigenvalues occur with  $\alpha_j/\beta_j$  and  $\alpha_{j+1}/\beta_{j+1}$  complex conjugates, even though  $\alpha_j$  and  $\alpha_{j+1}$  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.

## 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:   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'.
  
- 2:   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'.
  
- 3:   N – INTEGER *Input*  
*On entry:*  $n$ , the order of the matrices  $A$  and  $B$ .  
*Constraint:*  $N \geq 0$ .
  
- 4:   A(LDA,\*) – REAL (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.
  
- 5:   LDA – INTEGER *Input*  
*On entry:* the first dimension of the array  $A$  as declared in the (sub)program from which F08WAF (DGGEV) is called.  
*Constraint:*  $LDA \geq \max(1, N)$ .
  
- 6:   B(LDB,\*) – REAL (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.
  
- 7:   LDB – INTEGER *Input*  
*On entry:* the first dimension of the array  $B$  as declared in the (sub)program from which F08WAF (DGGEV) is called.  
*Constraint:*  $LDB \geq \max(1, N)$ .

- 8: ALPHAR(N) – REAL (KIND=nag\_wp) array Output  
*On exit:* the element ALPHAR( $j$ ) contains the real part of  $\alpha_j$ .
- 9: ALPHAI(N) – REAL (KIND=nag\_wp) array Output  
*On exit:* the element ALPHAI( $j$ ) contains the imaginary part of  $\alpha_j$ .
- 10: BETA(N) – REAL (KIND=nag\_wp) array Output  
*On exit:* (ALPHAR( $j$ ) + ALPHAI( $j$ )  $\times$   $i$ )/BETA( $j$ ), for  $j = 1, 2, \dots, N$ , will be the generalized eigenvalues.  
 If ALPHAI( $j$ ) is zero, then the  $j$ th eigenvalue is real; if positive, then the  $j$ th and ( $j + 1$ )st eigenvalues are a complex conjugate pair, with ALPHAI( $j + 1$ ) negative.  
**Note:** the quotients ALPHAR( $j$ )/BETA( $j$ ) and ALPHAI( $j$ )/BETA( $j$ ) may easily overflow or underflow, and 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  $\|B\|_2$ .
- 11: 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 eigenvectors  $u_j$  are stored one after another in the columns of VL, in the same order as the corresponding eigenvalues.  
 If the  $j$ th eigenvalue is real, then  $u_j = \text{VL}(:, j)$ , the  $j$ th column of VL.  
 If the  $j$ th and ( $j + 1$ )th eigenvalues form a complex conjugate pair, then  $u_j = \text{VL}(:, j) + i \times \text{VL}(:, j + 1)$  and  $u_{j+1} = \text{VL}(:, j) - i \times \text{VL}(:, j + 1)$ . Each eigenvector will be scaled so the largest component has  $|\text{real part}| + |\text{imag. part}| = 1$ .  
 If JOBVL = 'N', VL is not referenced.
- 12: LDVL – INTEGER Input  
*On entry:* the first dimension of the array VL as declared in the (sub)program from which F08WAF (DGGEV) is called.  
**Constraints:**  
     if JOBVL = 'V', LDVL  $\geq \max(1, N)$ ;  
     otherwise LDVL  $\geq 1$ .
- 13: 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 eigenvectors  $v_j$  are stored one after another in the columns of VR, in the same order as the corresponding eigenvalues.  
 If the  $j$ th eigenvalue is real, then  $v_j = \text{VR}(:, j)$ , the  $j$ th column of VR.  
 If the  $j$ th and ( $j + 1$ )th eigenvalues form a complex conjugate pair, then  $v_j = \text{VR}(:, j) + i \times \text{VR}(:, j + 1)$  and  $v_{j+1} = \text{VR}(:, j) - i \times \text{VR}(:, j + 1)$ . Each eigenvector will be scaled so the largest component has  $|\text{real part}| + |\text{imag. part}| = 1$ .  
 If JOBVR = 'N', VR is not referenced.
- 14: LDVR – INTEGER Input  
*On entry:* the first dimension of the array VR as declared in the (sub)program from which F08WAF (DGGEV) is called.

*Constraints:*

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

15: WORK(max(1, LWORK)) – REAL (KIND=nag\_wp) array *Workspace*

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

16: LWORK – INTEGER *Input*

*On entry:* the dimension of the array WORK as declared in the (sub)program from which F08WAF (DGGEV) 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**.

*Constraint:* LWORK  $\geq$  max(1,  $8 \times N$ ).

17: 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  $j = \text{INFO} + 1, \dots, N$ .

INFO = N + 1

Unexpected error returned from F08XEF (DHGEQZ).

INFO = 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  $\epsilon$  is the **machine precision**. See Section 4.11 of Anderson *et al.* (1999) for further details.

**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

F08WAF (DGGEV) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

F08WAF (DGGEV) 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 F08WNF (ZGGEV).

## 10 Example

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

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

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 f08wafe

!      F08WAF Example Program Text

!      Mark 26 Release. NAG Copyright 2016.

!      .. Use Statements ..
      Use nag_library, Only: dggev, m0ldef, m0leaf, nag_wp, x02ajf, x02amf
!      .. Implicit None Statement ..
      Implicit None
!      .. Parameters ..
      Real (Kind=nag_wp), Parameter      :: zero = 0.0_nag_wp
      Integer, Parameter                 :: nb = 64, nin = 5, nout = 6
!      .. Local Scalars ..
      Complex (Kind=nag_wp)              :: eig
      Real (Kind=nag_wp)                  :: scal_i, scal_r, small
      Integer                             :: i, ifail, info, j, k, lda, ldb,      &
      ldvr, lwork, n
      Logical                             :: pair
!      .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable     :: a(:, :), alphai(:), alphas(:),      &
      b(:, :), beta(:), vr(:, :), vr_row(:), &
      work(:)
      Real (Kind=nag_wp)                  :: dummy(1,1)
      Integer, Allocatable                :: irank(:)
!      .. Intrinsic Procedures ..
      Intrinsic                           :: abs, all, cmplx, max, maxloc, nint, &
      sqrt
!      .. Executable Statements ..
      Write (nout,*) 'F08WAF Example Program Results'
!      Skip heading in data file
      Read (nin,*)
      Read (nin,*) n
      lda = n

```

```

    ldb = n
    ldvr = n
    Allocate (a(lda,n),alphi(n),alphiar(n),b(ldb,n),beta(n),vr(ldvr,n),      &
             irank(n))

!   Use routine workspace query to get optimal workspace.
    lwork = -1
!   The NAG name equivalent of dggev is f08waf
    Call dggev('No left vectors','Vectors (right)',n,a,lda,b,ldb,alphiar,      &
             alphi,beta,dummy,1,vr,ldvr,dummy,lwork,info)

!   Make sure that there is enough workspace for block size nb.
    lwork = max((nb+7)*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 dggev is f08waf
    Call dggev('No left vectors','Vectors (right)',n,a,lda,b,ldb,alphiar,      &
             alphi,beta,dummy,1,vr,ldvr,work,lwork,info)

    If (info>0) Then
        Write (nout,*)
        Write (nout,99999) 'Failure in DGGEV. INFO =', info
    Else
!   If beta(:) > eps, Order eigenvalues by ascending real parts
!   and then by ascending imaginary parts
        If (all(abs(beta(1:n))>x02ajf())) Then
            work(1:n) = alphiar(1:n)/beta(1:n)
            work(n+1:2*n) = alphi(1:n)/beta(1:n)
            ifail = 0
            Call m0ldef(work,n,1,n,1,2,'Ascending',irank,ifail)
            Call m0leaf(alphiar,1,n,irank,ifail)
            Call m0leaf(alphi,1,n,irank,ifail)
            Call m0leaf(beta,1,n,irank,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 m0leaf(vr_row,1,n,irank,ifail)
                vr(j,1:n) = vr_row(1:n)
            End Do
            Deallocate (vr_row)
        End If
        small = x02amf()
        pair = .False.
        Do j = 1, n
            Write (nout,*)
            If ((abs(alphiar(j))+abs(alphi(j)))*small>=abs(beta(j))) Then
                Write (nout,99998) 'Eigenvalue(', j, ')',      &
                    ' is numerically infinite or undetermined', 'ALPHAR(', j,      &
                    ') = ', alphiar(j), ', ALPHAI(', j, ') = ', alphi(j), ', BETA(', &
                    j, ') = ', beta(j)
            Else
                If (alphi(j)==zero) Then
                    Write (nout,99997) 'Eigenvalue(', j, ') = ', alphiar(j)/beta(j)
                Else
                    eig = cmplx(alphiar(j),alphi(j),kind=nag_wp)/      &
                        cmplx(beta(j),kind=nag_wp)
                    Write (nout,99996) 'Eigenvalue(', j, ') = ', eig
                End If
            End If
            Write (nout,*)
            Write (nout,99995) 'Eigenvector(', j, ')'
            If (alphi(j)==zero) Then
!   Let largest element be positive

```

```

        work(1:n) = abs(vr(1:n,j))
        k = maxloc(work(1:n),1)
        If (vr(k,j)<zero) Then
            vr(1:n,j) = -vr(1:n,j)
        End If
        Write (nout,99994)(vr(i,j),i=1,n)
    Else
        If (pair) Then
            Write (nout,99993)(vr(i,j-1),-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))
            scal_i = -vr(k,j+1)/sqrt(work(k))
            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)
            Write (nout,99993)(vr(i,j),vr(i,j+1),i=1,n)
        End If
        pair = .Not. pair
    End If
End Do

End If

99999 Format (1X,A,I4)
99998 Format (1X,A,I2,2A,/,1X,2(A,I2,A,1P,F11.3,3X),A,I2,A,1P,F11.3)
99997 Format (1X,A,I2,A,1P,F11.3)
99996 Format (1X,A,I2,A,'(',1P,F11.3,',',1P,F11.3,')')
99995 Format (1X,A,I2,A)
99994 Format (1X,1P,F11.5)
99993 Format (1X,'(',1P,F11.5,',',1P,F11.5,')')
End Program f08wafe

```

## 10.2 Program Data

F08WAF 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

F08WAF Example Program Results

Eigenvalue( 1) = 20.000

Eigenvector( 1)

```

10.00000
 0.05714
 0.62857
 0.62857

```

Eigenvalue( 2) = ( 30.000, -40.000)

Eigenvector( 2)

```

( 7.12215, 0.00000)
( 1.42443, -0.00000)
( 0.85466, 1.13954)
( 0.85466, 1.13954)

```

Eigenvalue( 3) = ( 30.000, 40.000)

Eigenvector( 3)

( 7.12215, 0.00000)

( 1.42443, 0.00000)

( 0.85466, -1.13954)

( 0.85466, -1.13954)

Eigenvalue( 4) = 40.000

Eigenvector( 4)

10.00000

0.11111

-0.33333

1.55556

---