NAG Fortran Library Routine Document

F02GJF

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

F02GJF calculates 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, using the QZ algorithm.

2 Specification

```
 SUBROUTINE F02GJF(N, AR, IAR, AI, IAI, BR, IBR, BI, IBI, EPS1, ALFR,
1                  ALFI, BETA, MATV, VR, IVR, VI, IVI, ITER, IFAIL)
 INTEGER           N, IAR, IAI, IBR, IBI, IVR, IVI, ITER(N), IFAIL
 real              AR(IAR,N), AI(IAI,N), BR(IBR,N), BI(IBI,N), EPS1,
1                  ALFR(N), ALFI(N), BETA(N), VR(IVR,N), VI(IVI,N)
 LOGICAL           MATV
```

3 Description

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

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

   The division by β_j becomes the responsibility of the user's program, since β_j may be zero, indicating an infinite eigenvalue.
3. If the eigenvectors are required (MATV = .TRUE.), they are obtained from the triangular matrices and then transferred back into the original co-ordinate system.

4 References

Moler C B and Stewart G W (1973) An algorithm for generalized matrix eigenproblems *SIAM J. Numer. Anal.* **10** 241–256

Ward R C (1975) The combination shift QZ algorithm *SIAM J. Numer. Anal.* **12** 835–853

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

5 Parameters

1: N – INTEGER *Input*

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

Constraint: N $\geq$ 1.

2: AR(IAR,N) – ***real*** array *Input/Output*

On entry: the real parts of the elements of the n by n complex matrix A.

On exit: the array is overwritten.

3: IAR – INTEGER *Input*

On entry: the first dimension of the array AR as declared in the (sub)program from which F02GJF is called.

Constraint: IAR $\geq$ N.

4: AI(IAI,N) – ***real*** array *Input/Output*

On entry: the imaginary parts of the elements of the n by n complex matrix A.

On exit: the array is overwritten.

5: IAI – INTEGER *Input*

On entry: the first dimension of the array AI as declared in the (sub)program from which F02GJF is called.

Constraint: IAI $\geq$ N.

6: BR(IBR,N) – ***real*** array *Input/Output*

On entry: the real parts of the elements of the n by n complex matrix B.

On exit: the array is overwritten.

7: IBR – INTEGER *Input*

On entry: the first dimension of the array BR as declared in the (sub)program from which F02GJF is called.

Constraint: IBR $\geq$ N.

8: BI(IBI,N) – ***real*** array *Input/Output*

On entry: the imaginary parts of the elements of the n by n complex matrix B.

On exit: the array is overwritten.

9: IBI – INTEGER *Input*

On entry: the first dimension of the array BI as declared in the (sub)program from which F02GJF is called.

Constraint: IBI $\geq$ N.

10: EPS1 – ***real*** *Input*

On entry: a tolerance used to determine negligible elements. If EPS1 $>$ 0.0, an element will be considered negligible if it is less than EPS1 times the norm of its matrix. If EPS1 $\leq$ 0.0, ***machine precision*** is used for EPS1. A positive value of EPS1 may result in faster execution but less accurate results.

11: ALFR(N) – ***real*** array *Output*
12: ALFI(N) – ***real*** array *Output*

On exit: the real and imaginary parts of α_j, for $j = 1, 2, \ldots, n$.

13: BETA(N) – ***real*** array *Output*

On exit: β_j, for $j = 1, 2, \ldots, n$.

14: MATV – LOGICAL *Input*

On entry: MATV must be set .TRUE. if the eigenvectors are required, otherwise .FALSE..

15: VR(IVR,N) – ***real*** array *Output*

On exit: if MATV = .TRUE., the jth column of VR contains the real parts of the eigenvector corresponding to the jth eigenvalue. The eigenvectors are normalised so that the sum of squares of the moduli of the components is equal to 1.0 and the component of largest modulus is real.

If MATV = .FALSE., VR is not used.

16: IVR – INTEGER *Input*

On entry: the first dimension of the array VR as declared in the (sub)program from which F02GJF is called.

Constraint: IVR $\geq$ N.

17: VI(IVI,N) – ***real*** array *Output*

On exit: if MATV = .TRUE., the jth column of VI contains the imaginary parts of the eigenvector corresponding to the jth eigenvalue.

If MATV = .FALSE., VI is not used.

18: IVI – INTEGER *Input*

On entry: the first dimension of the array VI as declared in the (sub)program from which F02GJF is called.

Constraint: IVI $\geq$ N.

19: ITER(N) – INTEGER array *Output*

On exit: ITER(j) contains the number of iterations needed to obtain the jth eigenvalue. Note that the eigenvalues are obtained in reverse order, starting with the nth.

20: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. Users who are unfamiliar with this parameter should refer to Chapter P01 for details.

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

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, for users not familiar with this parameter the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

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 = i

More than $30 \times$ N iterations have been performed altogether in the second step of the QZ algorithm; IFAIL is set to the index i of the eigenvalue at which the failure occurs. On soft failure, α_j and β_j are correct for $j = i+1, i+2, \ldots, n$, but the arrays VR and VI do not contain any correct eigenvectors.

7 Accuracy

The computed eigenvalues are always exact for a problem $(A+E)x = \lambda(B+F)x$ where $\|E\|/\|A\|$ and $\|F\|/\|B\|$ are both of the order of $\max(\mathrm{EPS1}, \epsilon)$, EPS1 being defined as in Section 5 and ϵ being the ***machine precision***.

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 $\lambda_i = \alpha_i/\beta_i$. The user is 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 time taken by the routine is approximately proportional to n^3 and also depends on the value chosen for parameter EPS1.

9 Example

To find all the eigenvalues and eigenvectors of $Ax = \lambda Bx$ where

$$A = \begin{pmatrix} -21.10 - 22.50i & 53.5 - 50.5i & -34.5 + 127.5i & 7.5 + 0.5i \\ -0.46 - 7.78i & -3.5 - 37.5i & -15.5 + 58.5i & -10.5 - 1.5i \\ 4.30 - 5.50i & 39.7 - 17.1i & -68.5 + 12.5i & -7.5 - 3.5i \\ 5.50 + 4.40i & 14.4 + 43.3i & -32.5 - 46.0i & -19.0 - 32.5i \end{pmatrix}$$

and

$$B = \begin{pmatrix} 1.0 - 5.0i & 1.6 + 1.2i & -3.0 & -1.0i \\ 0.8 - 0.6i & 3.0 - 5.0i & -4.0 + 3.0i & -2.4 - 3.2i \\ 1.0 & 2.4 + 1.8i & -4.0 - 5.0i & -3.0i \\ 1.0i & -1.8 + 2.4i & -4.0 - 4.0i & 4.0 - 5.0i \end{pmatrix}.$$

9.1 Program Text

Note: the listing of the example program presented below uses ***bold italicised*** terms to denote precision-dependent details. Please read the Users' Note for your implementation to check the interpretation of these terms. As explained in the Essential Introduction to this manual, the results produced may not be identical for all implementations.

```
*     F02GJF Example Program Text
*     Mark 14 Revised.  NAG Copyright 1989.
*     .. Parameters ..
      INTEGER          NMAX, IAR, IAI, IBR, IBI, IVR, IVI
      PARAMETER        (NMAX=4,IAR=NMAX,IAI=NMAX,IBR=NMAX,IBI=NMAX,
     +                 IVR=NMAX,IVI=NMAX)
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
*     .. Local Scalars ..
      real             EPS1
      INTEGER          I, IFAIL, J, N
      LOGICAL          MATV
*     .. Local Arrays ..
      real             AI(IAI,NMAX), ALFI(NMAX), ALFR(NMAX),
     +                 AR(IAR,NMAX), BETA(NMAX), BI(IBI,NMAX),
     +                 BR(IBR,NMAX), VI(IVI,NMAX), VR(IVR,NMAX)
      INTEGER          ITER(NMAX)
*     .. External Functions ..
      real             X02AJF
      EXTERNAL         X02AJF
*     .. External Subroutines ..
      EXTERNAL         F02GJF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F02GJF Example Program Results'
*     Skip heading in data file
```

```
      READ (NIN,*)
      READ (NIN,*) N
      IF (N.GT.0 .AND. N.LE.NMAX) THEN
         READ (NIN,*) ((AR(I,J),AI(I,J),J=1,N),I=1,N)
         READ (NIN,*) ((BR(I,J),BI(I,J),J=1,N),I=1,N)
         EPS1 = X02AJF()
         MATV = .TRUE.
         IFAIL = 1
*
         CALL F02GJF(N,AR,IAR,AI,IAI,BR,IBR,BI,IBI,EPS1,ALFR,ALFI,BETA,
     +               MATV,VR,IVR,VI,IVI,ITER,IFAIL)
*
         WRITE (NOUT,*)
         IF (IFAIL.NE.0) THEN
            WRITE (NOUT,99999) 'Error in F02GJF. IFAIL =', IFAIL
         ELSE
            DO 20 I = 1, N
               ALFR(I) = ALFR(I)/BETA(I)
               ALFI(I) = ALFI(I)/BETA(I)
   20       CONTINUE
            WRITE (NOUT,*) 'Eigenvalues'
            WRITE (NOUT,99998) (' (',ALFR(I),',',ALFI(I),')',I=1,N)
            WRITE (NOUT,*)
            WRITE (NOUT,*) 'Eigenvectors'
            DO 40 I = 1, N
               WRITE (NOUT,99998) (' (',VR(I,J),',',VI(I,J),')',J=1,N)
   40       CONTINUE
         END IF
      ELSE
         WRITE (NOUT,99999) 'N is out of range: N = ', N
      END IF
      STOP
*
99999 FORMAT (1X,A,I5)
99998 FORMAT (1X,4(A,F7.3,A,F7.3,A))
      END
```

9.2 Program Data

```
F02GJF Example Program Data
  4
  -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
    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
```

9.3 Program Results

```
 F02GJF Example Program Results

 Eigenvalues
  (  3.000, -9.000) (  2.000, -5.000) (  3.000, -1.000) (  4.000, -5.000)

 Eigenvectors
  (  0.945,  0.000) (  0.996,  0.000) (  0.945,  0.000) (  0.988,  0.000)
  (  0.151, -0.113) (  0.005, -0.003) (  0.151, -0.113) (  0.009, -0.007)
  (  0.113,  0.151) (  0.063,  0.000) (  0.113, -0.151) ( -0.033,  0.000)
  ( -0.151,  0.113) ( -0.000,  0.063) (  0.151,  0.113) ( -0.000,  0.154)
```