F07BVF (CGBRFS/ZGBRFS) – NAG Fortran Library Routine Document

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

F07BVF (CGBRFS/ZGBRFS) returns error bounds for the solution of a complex band system of linear equations with multiple right-hand sides,$AX = B$, $A^T X = B$ or $A^H X = B$. It improves the solution by iterative refinement, in order to reduce the backward error as much as possible.

2 Specification

```
      SUBROUTINE F07BVF(TRANS, N, KL, KU, NRHS, AB, LDAB, AFB, LDAFB,
     1                  IPIV, B, LDB, X, LDX, FERR, BERR, WORK, RWORK,
     2                  INFO)
      ENTRY      cgbrfs(TRANS, N, KL, KU, NRHS, AB, LDAB, AFB, LDAFB,
     1                  IPIV, B, LDB, X, LDX, FERR, BERR, WORK, RWORK,
     2                  INFO)
      INTEGER           N, KL, KU, NRHS, LDAB, LDAFB, IPIV(*), LDB, LDX,
     1                  INFO
      real              FERR(*), BERR(*), RWORK(*)
      complex           AB(LDAB,*), AFB(LDAFB,*), B(LDB,*), X(LDX,*),
     1                  WORK(*)
      CHARACTER*1       TRANS
```

The ENTRY statement enables the routine to be called by its LAPACK name.

3 Description

This routine returns the backward errors and estimated bounds on the forward errors for the solution of a complex band system of linear equations with multiple right-hand sides $AX = B$, $A^T X = B$ or $A^H X = B$. The routine handles each right-hand side vector (stored as a column of the matrix B) independently, so we describe the function of the routine in terms of a single right-hand side b and solution x.

Given a computed solution x, the routine computes the *component-wise backward error* β. This is the size of the smallest relative perturbation in each element of A and b such that x is the exact solution of a perturbed system

$$(A + \delta A)x = b + \delta b$$
$$|\delta a_{ij}| \leq \beta|a_{ij}| \text{ and } |\delta b_i| \leq \beta|b_i|.$$

Then the routine estimates a bound for the *component-wise forward error* in the computed solution, defined by:

$$\max_i |x_i - \hat{x}_i| / \max_i |x_i|$$

where $\hat{x}$ is the true solution.

For details of the method, see the the Chapter Introduction.

4 References

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

5 Parameters

1: TRANS — CHARACTER*1 *Input*

On entry: indicates the form of the linear equations for which X is the computed solution as follows:

if TRANS = 'N', then the linear equations are of the form $AX = B$;
if TRANS = 'T', then the linear equations are of the form $A^T X = B$;
if TRANS = 'C', then the linear equations are of the form $A^H X = B$.

Constraint: TRANS = 'N', 'T' or 'C'.

2: N — INTEGER *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

3: KL — INTEGER *Input*

On entry: k_l, the number of sub-diagonals within the band of A.

Constraint: KL $\geq$ 0.

4: KU — INTEGER *Input*

On entry: k_u, the number of super-diagonals within the band of A.

Constraint: KU $\geq$ 0.

5: NRHS — INTEGER *Input*

On entry: r, the number of right-hand sides.

Constraint: NRHS $\geq$ 0.

6: AB(LDAB,$*$) — ***complex*** array *Input*

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

On entry: the n by n original band matrix A as supplied to F07BRF (CGBTRF/ZGBTRF), but stored in rows 1 to $(k_l + k_u + 1)$ of the array rather than in rows $(k_l + 1)$ to $(2k_l + k_u + 1)$.

7: LDAB — INTEGER *Input*

On entry: n, the order of the matrix A.

Constraint: N $\geq$ 0.

8: AFB(LDAFB,$*$) — ***complex*** array *Input*

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

On entry: the *LU* factorization of A, as returned by F07BRF (CGBTRF/ZGBTRF).

9: LDAFB — INTEGER *Input*

On entry: the first dimension of the array AFB as declared in the (sub)program from which F07BVF (CGBRFS/ZGBRFS) is called.

Constraint: LDAFB $\geq$ 2 $\times$ KL + KU + 1.

10: IPIV($*$) — INTEGER array *Input*

Note: the dimension of the array IPIV must be at least max(1,N).

On entry: the pivot indices, as returned by F07BRF (CGBTRF/ZGBTRF).

11: B(LDB,$*$) — ***complex*** array *Input*

Note: the second dimension of the array B must be at least max(1,NRHS).

On entry: the n by r right-hand side matrix B.

12: LDB — INTEGER *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07BVF (CGBRFS/ZGBRFS) is called.

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

13: X(LDX,∗) — ***complex*** array *Input/Output*

Note: the second dimension of the array X must be at least max(1,NRHS).

On entry: the n by r solution matrix X, as returned by F07BSF (CGBTRS/ZGBTRS).

On exit: the improved solution matrix X.

14: LDX — INTEGER *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which F07BVF (CGBRFS/ZGBRFS) is called.

Constraint: LDX $\geq$ max(1,N).

15: FERR(∗) — ***real*** array *Output*

Note: the dimension of the array FERR must be at least max(1,NRHS).

On exit: FERR(j) contains an estimated error bound for the jth solution vector, that is, the jth column of X, for $j = 1, 2, \ldots, r$.

16: BERR(∗) — ***real*** array *Output*

Note: the dimension of the array BERR must be at least max(1,NRHS).

On exit: BERR(j) contains the component-wise backward error bound β for the jth solution vector, that is, the jth column of X, for $j = 1, 2, \ldots, r$.

17: WORK(∗) — ***complex*** array *Workspace*

Note: the dimension of the array WORK must be at least max(1,2∗N).

18: RWORK(∗) — ***real*** array *Workspace*

Note: the dimension of the array RWORK must be at least max(1,N).

19: 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 $= -i$, the ith parameter had an illegal value. An explanatory message is output, and execution of the program is terminated.

7 Accuracy

The bounds returned in FERR are not rigorous, because they are estimated, not computed exactly; but in practice they almost always overestimate the actual error.

8 Further Comments

For each right-hand side, computation of the backward error involves a minimum of $16n(k_l + k_u)$ real floating-point operations. Each step of iterative refinement involves an additional $8n(4k_l + 3k_u)$ real operations. This assumes $n \gg k_l$ and $n \gg k_u$. At most 5 steps of iterative refinement are performed, but usually only 1 or 2 steps are required.

Estimating the forward error involves solving a number of systems of linear equations of the form $Ax = b$ or $A^H x = b$; the number is usually 5 and never more than 11. Each solution involves approximately $8n(2k_l + k_u)$ real operations.

The real analogue of this routine is F07BHF (SGBRFS/DGBRFS).

9 Example

To solve the system of equations $AX = B$ using iterative refinement and to compute the forward and backward error bounds, where

$$A = \begin{pmatrix} -1.65+2.26i & -2.05-0.85i & 0.97-2.84i & 0.00+0.00i \\ 0.00+6.30i & -1.48-1.75i & -3.99+4.01i & 0.59-0.48i \\ 0.00+0.00i & -0.77+2.83i & -1.06+1.94i & 3.33-1.04i \\ 0.00+0.00i & 0.00+0.00i & 4.48-1.09i & -0.46-1.72i \end{pmatrix} \text{ and}$$

$$B = \begin{pmatrix} -1.06+21.50i & 12.85+\ 2.84i \\ -22.72-53.90i & -70.22+21.57i \\ 28.24-38.60i & -20.73-\ 1.23i \\ -34.56+16.73i & 26.01+31.97i \end{pmatrix}.$$

Here A is nonsymmetric and is treated as a band matrix, which must first be factorized by F07BRF (CGBTRF/ZGBTRF).

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.

```
*     F07BVF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      complex          ZERO
      PARAMETER        (ZERO=(0.0e0,0.0e0))
      INTEGER          NMAX, NRHMAX, KLMAX, KUMAX, LDAB, LDAFB, LDB, LDX
      PARAMETER        (NMAX=8,NRHMAX=NMAX,KLMAX=8,KUMAX=8,
     +                 LDAB=KLMAX+KUMAX+1,LDAFB=2*KLMAX+KUMAX+1,
     +                 LDB=NMAX,LDX=NMAX)
      CHARACTER        TRANS
      PARAMETER        (TRANS='N')
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, K, KL, KU, N, NRHS
*     .. Local Arrays ..
      complex          AB(LDAB,NMAX), AFB(LDAFB,NMAX), B(LDB,NRHMAX),
     +                 WORK(2*NMAX), X(LDX,NMAX)
      real             BERR(NRHMAX), FERR(NRHMAX), RWORK(NMAX)
      INTEGER          IPIV(NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         cgbrfs, cgbtrf, cgbtrs, F06TFF, F06THF, X04DBF
```

```
*     .. Intrinsic Functions ..
      INTRINSIC        MAX, MIN
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07BVF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS, KL, KU
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX .AND. KL.LE.KLMAX .AND. KU.LE.
     +    KUMAX) THEN
*
*        Set A to zero to avoid referencing uninitialized elements
*
         CALL F06THF('General',KL+KU+1,N,ZERO,ZERO,AB,LDAB)
*
*        Read A and B from data file, and copy A to AFB and B to X
*
         K = KU + 1
         READ (NIN,*) ((AB(K+I-J,J),J=MAX(I-KL,1),MIN(I+KU,N)),I=1,N)
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
         CALL F06TFF('General',KL+KU+1,N,AB,LDAB,AFB(KL+1,1),LDAFB)
         CALL F06TFF('General',N,NRHS,B,LDB,X,LDX)
*
*        Factorize A in the array AFB
*
         CALL cgbtrf(N,N,KL,KU,AFB,LDAFB,IPIV,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute solution in the array X
*
            CALL cgbtrs(TRANS,N,KL,KU,NRHS,AFB,LDAFB,IPIV,X,LDX,INFO)
*
*           Improve solution, and compute backward errors and
*           estimated bounds on the forward errors
*
            CALL cgbrfs(TRANS,N,KL,KU,NRHS,AB,LDAB,AFB,LDAFB,IPIV,B,LDB,
     +                  X,LDX,FERR,BERR,WORK,RWORK,INFO)
*
*           Print solution
*
            IFAIL = 0
            CALL X04DBF('General',' ',N,NRHS,X,LDX,'Bracketed','F7.4',
     +                  'Solution(s)','Integer',RLABS,'Integer',CLABS,
     +                  80,0,IFAIL)
            WRITE (NOUT,*)
            WRITE (NOUT,*) 'Backward errors (machine-dependent)'
            WRITE (NOUT,99999) (BERR(J),J=1,NRHS)
            WRITE (NOUT,*)
     +        'Estimated forward error bounds (machine-dependent)'
            WRITE (NOUT,99999) (FERR(J),J=1,NRHS)
         ELSE
            WRITE (NOUT,*) 'The factor U is singular'
         END IF
      END IF
      STOP
*
```

```
99999 FORMAT ((5X,1P,4(e11.1,7X)))
      END
```

9.2 Program Data

```
F07BVF Example Program Data
  4  2  1  2                                           :Values of N, NRHS, KL and KU
 (-1.65, 2.26) (-2.05,-0.85) ( 0.97,-2.84)
 ( 0.00, 6.30) (-1.48,-1.75) (-3.99, 4.01) ( 0.59,-0.48)
               (-0.77, 2.83) (-1.06, 1.94) ( 3.33,-1.04)
                             ( 4.48,-1.09) (-0.46,-1.72)  :End of matrix A
 ( -1.06, 21.50) ( 12.85,  2.84)
 (-22.72,-53.90) (-70.22, 21.57)
 ( 28.24,-38.60) (-20.73, -1.23)
 (-34.56, 16.73) ( 26.01, 31.97)                          :End of matrix B
```

9.3 Program Results

```
 F07BVF Example Program Results

 Solution(s)
                    1                  2
 1  (-3.0000, 2.0000) ( 1.0000, 6.0000)
 2  ( 1.0000,-7.0000) (-7.0000,-4.0000)
 3  (-5.0000, 4.0000) ( 3.0000, 5.0000)
 4  ( 6.0000,-8.0000) (-8.0000, 2.0000)

 Backward errors (machine-dependent)
        2.7E-17            6.7E-17
 Estimated forward error bounds (machine-dependent)
        3.5E-14            4.3E-14
```