NAG Library Routine Document

F07BVF (ZGBRFS)

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 (ZGBRFS) returns error bounds for the solution of a complex band system of linear equations with multiple right-hand sides, $AX = B$, $A^{\mathrm{T}}X = B$ or $A^{\mathrm{H}}X = B$. 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, IPIV, B,       &
                   LDB, X, LDX, FERR, BERR, WORK, RWORK, INFO)
INTEGER                 N, KL, KU, NRHS, LDAB, LDAFB, IPIV(*), LDB, LDX,        &
                        INFO
REAL (KIND=nag_wp)      FERR(NRHS), BERR(NRHS), RWORK(N)
COMPLEX (KIND=nag_wp)   AB(LDAB,*), AFB(LDAFB,*), B(LDB,*), X(LDX,*),           &
                        WORK(2*N)
CHARACTER(1)            TRANS
```

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

3 Description

F07BVF (ZGBRFS) 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 $AX = B$, $A^{\mathrm{T}}X = B$ or $A^{\mathrm{H}}X = B$. The routine handles each right-hand side vector (stored as a column of the matrix B) independently, so we describe the function of F07BVF (ZGBRFS) 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

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

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

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

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

For details of the method, see the F07 Chapter Introduction.

4 References

Golub G H and Van Loan C F (1996) *Matrix Computations* (3rd Edition) Johns Hopkins University Press, 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:

TRANS = 'N'
: The linear equations are of the form $AX = B$.

TRANS = 'T'
: The linear equations are of the form $A^{\mathrm{T}}X = B$.

TRANS = 'C'
: The linear equations are of the form $A^{\mathrm{H}}X = B$.

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

2: N – INTEGER *Input*

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

Constraint: $\mathrm{N} \geq 0$.

3: KL – INTEGER *Input*

On entry: k_l, the number of subdiagonals within the band of the matrix A.

Constraint: $\mathrm{KL} \geq 0$.

4: KU – INTEGER *Input*

On entry: k_u, the number of superdiagonals within the band of the matrix A.

Constraint: $\mathrm{KU} \geq 0$.

5: NRHS – INTEGER *Input*

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

Constraint: $\mathrm{NRHS} \geq 0$.

6: AB(LDAB,$*$) – COMPLEX (KIND=nag_wp) array *Input*

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

On entry: the original n by n band matrix A as supplied to F07BRF (ZGBTRF).

The matrix is stored in rows 1 to $k_l + k_u + 1$, more precisely, the element A_{ij} must be stored in

$$\mathrm{AB}(k_u + 1 + i - j, j) \qquad \text{for } \max(1, j - k_u) \leq i \leq \min(n, j + k_l).$$

See Section 8 in F07BNF (ZGBSV) for further details.

7: LDAB – INTEGER *Input*

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

Constraint: $\mathrm{LDAB} \geq \mathrm{KL} + \mathrm{KU} + 1$.

8: AFB(LDAFB,$*$) – COMPLEX (KIND=nag_wp) array *Input*

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

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

9: LDAFB – INTEGER *Input*

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

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

10: IPIV($*$) – INTEGER array *Input*

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

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

11: B(LDB,$*$) – COMPLEX (KIND=nag_wp) array *Input*

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

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 (ZGBRFS) is called.

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

13: X(LDX,$*$) – COMPLEX (KIND=nag_wp) array *Input/Output*

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

On entry: the n by r solution matrix X, as returned by F07BSF (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 (ZGBRFS) is called.

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

15: FERR(NRHS) – REAL (KIND=nag_wp) array *Output*

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(NRHS) – REAL (KIND=nag_wp) array *Output*

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($2 \times \text{N}$) – COMPLEX (KIND=nag_wp) array *Workspace*

18: RWORK(N) – REAL (KIND=nag_wp) array *Workspace*

19: INFO – INTEGER *Output*

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

6 Error Indicators and Warnings

Errors or warnings detected by the routine:

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 five steps of iterative refinement are performed, but usually only one or two steps are required.

Estimating the forward error involves solving a number of systems of linear equations of the form $Ax = b$ or $A^{\mathrm{H}}x = b$; 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 (DGBRFS).

9 Example

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

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

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 (ZGBTRF).

9.1 Program Text

```
    Program f07bvfe

!     F07BVF Example Program Text

!     Mark 24 Release. NAG Copyright 2012.

!     .. Use Statements ..
      Use nag_library, Only: nag_wp, x04dbf, zgbrfs, zgbtrf, zgbtrs
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Complex (Kind=nag_wp), Parameter :: zero = (0.0_nag_wp,0.0_nag_wp)
      Integer, Parameter               :: nin = 5, nout = 6
      Character (1), Parameter         :: trans = 'N'
!     .. Local Scalars ..
      Integer                          :: i, ifail, info, j, k, kl, ku, ldab,  &
                                          ldafb, ldb, ldx, n, nrhs
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: ab(:,:), afb(:,:), b(:,:),         &
                                            work(:), x(:,:)
      Real (Kind=nag_wp), Allocatable  :: berr(:), ferr(:), rwork(:)
      Integer, Allocatable             :: ipiv(:)
      Character (1)                    :: clabs(1), rlabs(1)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: max, min
```

```
!     .. Executable Statements ..
      Write (nout,*) 'F07BVF Example Program Results'
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) n, nrhs, kl, ku
      ldb = n
      ldx = n
      ldab = kl + ku + 1
      ldafb = 2*kl + ku + 1
      Allocate (ab(ldab,n),afb(ldafb,n),b(ldb,nrhs),work(2*n),x(ldx,n), &
        berr(nrhs),ferr(nrhs),rwork(n),ipiv(n))

!     Set A to zero to avoid referencing uninitialized elements

      ab(1:kl+ku+1,1:n) = zero

!     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,1:nrhs),i=1,n)

      afb(kl+1:2*kl+ku+1,1:n) = ab(1:kl+ku+1,1:n)
      x(1:n,1:nrhs) = b(1:n,1:nrhs)

!     Factorize A in the array AFB
!     The NAG name equivalent of zgbtrf is f07brf
      Call zgbtrf(n,n,kl,ku,afb,ldafb,ipiv,info)

      Write (nout,*)
      Flush (nout)
      If (info==0) Then

!       Compute solution in the array X
!       The NAG name equivalent of zgbtrs is f07bsf
        Call zgbtrs(trans,n,kl,ku,nrhs,afb,ldafb,ipiv,x,ldx,info)

!       Improve solution, and compute backward errors and
!       estimated bounds on the forward errors
!       The NAG name equivalent of zgbrfs is f07bvf
        Call zgbrfs(trans,n,kl,ku,nrhs,ab,ldab,afb,ldafb,ipiv,b,ldb,x,ldx, &
          ferr,berr,work,rwork,info)

!       Print solution

!       ifail: behaviour on error exit
!              =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
        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(1:nrhs)
        Write (nout,*) 'Estimated forward error bounds (machine-dependent)'
        Write (nout,99999) ferr(1:nrhs)
      Else
        Write (nout,*) 'The factor U is singular'
      End If

99999 Format ((5X,1P,4(E11.1,7X)))
    End Program f07bvfe
```

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)
        1.8E-17          6.7E-17
Estimated forward error bounds (machine-dependent)
        3.5E-14          4.3E-14
```