NAG Library Routine Document

F07HVF (ZPBRFS)

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

F07HVF (ZPBRFS) returns error bounds for the solution of a complex Hermitian positive definite band system of linear equations with multiple right-hand sides, $AX = B$. It improves the solution by iterative refinement, in order to reduce the backward error as much as possible.

2 Specification

```
SUBROUTINE F07HVF (UPLO, N, KD, NRHS, AB, LDAB, AFB, LDAFB, B, LDB, X, LDX,     &
                   FERR, BERR, WORK, RWORK, INFO)

INTEGER                 N, KD, NRHS, LDAB, LDAFB, 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)            UPLO
```

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

3 Description

F07HVF (ZPBRFS) returns the backward errors and estimated bounds on the forward errors for the solution of a complex Hermitian positive definite band system of linear equations with multiple right-hand sides $AX = B$. The routine handles each right-hand side vector (stored as a column of the matrix B) independently, so we describe the function of F07HVF (ZPBRFS) 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

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

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 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: UPLO – CHARACTER(1) *Input*

On entry: specifies whether the upper or lower triangular part of A is stored and how A is to be factorized.

UPLO = 'U'

The upper triangular part of A is stored and A is factorized as $U^{\mathrm{H}}U$, where U is upper triangular.

UPLO = 'L'

The lower triangular part of A is stored and A is factorized as LL^{H}, where L is lower triangular.

Constraint: UPLO = 'U' or 'L'.

2: N – INTEGER *Input*

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

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

3: KD – INTEGER *Input*

On entry: k_d, the number of superdiagonals or subdiagonals of the matrix A.

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

4: NRHS – INTEGER *Input*

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

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

5: 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 n by n original Hermitian positive definite band matrix A as supplied to F07HRF (ZPBTRF).

6: LDAB – INTEGER *Input*

On entry: the first dimension of the array AB as declared in the (sub)program from which F07HVF (ZPBRFS) is called.

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

7: 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 Cholesky factor of A, as returned by F07HRF (ZPBTRF).

8: LDAFB – INTEGER *Input*

On entry: the first dimension of the array AFB as declared in the (sub)program from which F07HVF (ZPBRFS) is called.

Constraint: $\mathrm{LDAFB} \geq \mathrm{KD} + 1$.

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

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

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

10: LDB – INTEGER *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07HVF (ZPBRFS) is called.

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

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

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

On entry: the n by r solution matrix X, as returned by F07HSF (ZPBTRS).

On exit: the improved solution matrix X.

12: LDX – INTEGER *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which F07HVF (ZPBRFS) is called.

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

13: 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 $j = 1, 2, \ldots, r$.

14: 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 $j = 1, 2, \ldots, r$.

15: WORK($2 \times$ N) – COMPLEX (KIND=nag_wp) array *Workspace*

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

17: 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 $32nk$ real floating point operations. Each step of iterative refinement involves an additional $48nk$ real operations. This assumes $n \gg k$. 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 $Ax = b$; the number is usually 5 and never more than 11. Each solution involves approximately $16nk$ real operations.

The real analogue of this routine is F07HHF (DPBRFS).

9 Example

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

$$A = \begin{pmatrix} 9.39 + 0.00i & 1.08 - 1.73i & 0.00 + 0.00i & 0.00 + 0.00i \\ 1.08 + 1.73i & 1.69 + 0.00i & -0.04 + 0.29i & 0.00 + 0.00i \\ 0.00 + 0.00i & -0.04 - 0.29i & 2.65 + 0.00i & -0.33 + 2.24i \\ 0.00 + 0.00i & 0.00 + 0.00i & -0.33 - 2.24i & 2.17 + 0.00i \end{pmatrix}$$

and

$$B = \begin{pmatrix} -12.42 + 68.42i & 54.30 - 56.56i \\ -9.93 + 0.88i & 18.32 + 4.76i \\ -27.30 - 0.01i & -4.40 + 9.97i \\ 5.31 + 23.63i & 9.43 + 1.41i \end{pmatrix}.$$

Here A is Hermitian positive definite, and is treated as a band matrix, which must first be factorized by F07HRF (ZPBTRF).

9.1 Program Text

```
    Program f07hvfe

!     F07HVF Example Program Text

!     Mark 24 Release. NAG Copyright 2012.

!     .. Use Statements ..
      Use nag_library, Only: nag_wp, x04dbf, zpbrfs, zpbtrf, zpbtrs
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Complex (Kind=nag_wp), Parameter :: zero = (0.0E0_nag_wp,0.0E0_nag_wp)
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: i, ifail, info, j, kd, ldab, ldafb,  &
                                          ldb, ldx, n, nrhs
      Character (1)                    :: uplo
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: ab(:,:), afb(:,:), b(:,:),         &
                                            work(:), x(:,:)
      Real (Kind=nag_wp), Allocatable  :: berr(:), ferr(:), rwork(:)
      Character (1)                    :: clabs(1), rlabs(1)
!     .. Intrinsic Procedures ..
      Intrinsic                        :: max, min
!     .. Executable Statements ..
      Write (nout,*) 'F07HVF Example Program Results'
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) n, kd, nrhs
      ldab = kd + 1
      ldafb = kd + 1
      ldb = n
      ldx = n
      Allocate (ab(ldab,n),afb(ldafb,n),b(ldb,nrhs),work(2*n),x(ldx,n), &
        berr(nrhs),ferr(nrhs),rwork(n))

!     Set A to zero to avoid referencing uninitialized elements

      ab(1:kd+1,1:n) = zero

!     Read A and B from data file, and copy A to AFB and B to X

      Read (nin,*) uplo
```

```
      If (uplo=='U') Then
        Do i = 1, n
          Read (nin,*)(ab(kd+1+i-j,j),j=i,min(n,i+kd))
        End Do
      Else If (uplo=='L') Then
        Do i = 1, n
          Read (nin,*)(ab(1+i-j,j),j=max(1,i-kd),i)
        End Do
      End If
      Read (nin,*)(b(i,1:nrhs),i=1,n)

      afb(1:kd+1,1:n) = ab(1:kd+1,1:n)
      x(1:n,1:nrhs) = b(1:n,1:nrhs)

!     Factorize A in the array AFB

      Call zpbtrf(uplo,n,kd,afb,ldafb,info)

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

!       Compute solution in the array X

        Call zpbtrs(uplo,n,kd,nrhs,afb,ldafb,x,ldx,info)

!       Improve solution, and compute backward errors and
!       estimated bounds on the forward errors

        Call zpbrfs(uplo,n,kd,nrhs,ab,ldab,afb,ldafb,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,*) 'A is not positive definite'
      End If

99999 Format ((5X,1P,4(E11.1,7X)))
    End Program f07hvfe
```

9.2 Program Data

```
F07HVF Example Program Data
  4  1  2                                                :Values of N, KD and NRHS
  'L'                                                       :Value of UPLO
 ( 9.39, 0.00)
 ( 1.08, 1.73) ( 1.69, 0.00)
               (-0.04,-0.29) ( 2.65, 0.00)
                             (-0.33,-2.24) ( 2.17, 0.00)  :End of matrix A
 (-12.42,68.42) (54.30,-56.56)
 ( -9.93, 0.88) (18.32,  4.76)
 (-27.30,-0.01) (-4.40,  9.97)
 (  5.31,23.63) ( 9.43,  1.41)                            :End of matrix B
```

9.3 Program Results

```
 F07HVF Example Program Results

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

 Backward errors (machine-dependent)
         8.2E-17           8.3E-17
 Estimated forward error bounds (machine-dependent)
         3.5E-14           3.2E-14
```