F07GVF (CPPRFS/ZPPRFS) – 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

F07GVF (CPPRFS/ZPPRFS) returns error bounds for the solution of a complex Hermitian positive-definite system of linear equations with multiple right-hand sides, $AX = B$, using packed storage. It improves the solution by iterative refinement, in order to reduce the backward error as much as possible.

2 Specification

```
 SUBROUTINE F07GVF(UPLO, N, NRHS, AP, AFP, B, LDB, X, LDX, FERR,
1                  BERR, WORK, RWORK, INFO)
 ENTRY      cpprfs(UPLO, N, NRHS, AP, AFP, B, LDB, X, LDX, FERR,
1                  BERR, WORK, RWORK, INFO)
 INTEGER           N, NRHS, LDB, LDX, INFO
 real              FERR(*), BERR(*), RWORK(*)
 complex           AP(*), AFP(*), B(LDB,*), X(LDX,*), WORK(*)
 CHARACTER*1       UPLO
```

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 Hermitian positive-definite system of linear equations with multiple right-hand sides $AX = B$, using packed storage. 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 backwarderror* 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

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

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: UPLO — CHARACTER*1 *Input*

On entry: indicates whether the upper or lower triangular part of A is stored and how A has been factorized, as follows:

if UPLO = 'U', then the upper triangular part of A is stored and A is factorized as $U^H U$, where U is upper triangular;

if UPLO = 'L', then 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: N $\geq$ 0.

3: NRHS — INTEGER *Input*

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

Constraint: NRHS $\geq$ 0.

4: AP(∗) — ***complex*** array *Input*

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

On entry: the n by n original Hermitian positive-definite matrix A as supplied to F07GRF (CPPTRF/ZPPTRF).

5: AFP(∗) — ***complex*** array *Input*

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

On entry: the Cholesky factor of A stored in packed form, as returned by F07GRF (CPPTRF/ZPPTRF).

6: 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.

7: LDB — INTEGER *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07GVF (CPPRFS/ZPPRFS) is called.

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

8: 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 F07GSF (CPPTRS/ZPPTRS).

On exit: the improved solution matrix X.

9: LDX — INTEGER *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which F07GVF (CPPRFS/ZPPRFS) is called.

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

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

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

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

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

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

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

14: 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 $16n^2$ real floating-point operations. Each step of iterative refinement involves an additional $24n^2$ real operations. 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 $Ax = b$; the number is usually 5 and never more than 11. Each solution involves approximately $8n^2$ real operations.

The real analogue of this routine is F07GHF (SPPRFS/DPPRFS).

9 Example

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

$$A = \begin{pmatrix} 3.23+0.00i & 1.51-1.92i & 1.90+0.84i & 0.42+2.50i \\ 1.51+1.92i & 3.58+0.00i & -0.23+1.11i & -1.18+1.37i \\ 1.90-0.84i & -0.23-1.11i & 4.09+0.00i & 2.33-0.14i \\ 0.42-2.50i & -1.18-1.37i & 2.33+0.14i & 4.29+0.00i \end{pmatrix}$$

and

$$B = \begin{pmatrix} 3.93-\ 6.14i & 1.48+\ 6.58i \\ 6.17+\ 9.42i & 4.65-\ 4.75i \\ -7.17-21.83i & -4.91+\ 2.29i \\ 1.99-14.38i & 7.64-10.79i \end{pmatrix}.$$

Here A is Hermitian positive-definite, stored in packed form, and must first be factorized by F07GRF (CPPTRF/ZPPTRF).

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.

```
*     F07GVF Example Program Text
*     Mark 15 Release. NAG Copyright 1991.
*     .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, NRHMAX, LDB, LDX
      PARAMETER        (NMAX=8,NRHMAX=NMAX,LDB=NMAX,LDX=NMAX)
*     .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N, NRHS
      CHARACTER        UPLO
*     .. Local Arrays ..
      complex          AFP(NMAX*(NMAX+1)/2), AP(NMAX*(NMAX+1)/2),
     +                 B(LDB,NRHMAX), WORK(2*NMAX), X(LDX,NMAX)
      real             BERR(NRHMAX), FERR(NRHMAX), RWORK(NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*     .. External Subroutines ..
      EXTERNAL         cpprfs, cpptrf, cpptrs, F06TFF, X04DBF
*     .. Executable Statements ..
      WRITE (NOUT,*) 'F07GVF Example Program Results'
*     Skip heading in data file
      READ (NIN,*)
      READ (NIN,*) N, NRHS
      IF (N.LE.NMAX .AND. NRHS.LE.NRHMAX) THEN
*
*        Read A and B from data file, and copy A to AFP and B to X
*
         READ (NIN,*) UPLO
         IF (UPLO.EQ.'U') THEN
            READ (NIN,*) ((AP(I+J*(J-1)/2),J=I,N),I=1,N)
         ELSE IF (UPLO.EQ.'L') THEN
            READ (NIN,*) ((AP(I+(2*N-J)*(J-1)/2),J=1,I),I=1,N)
         END IF
         READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
         DO 20 I = 1, N*(N+1)/2
            AFP(I) = AP(I)
   20    CONTINUE
         CALL F06TFF('General',N,NRHS,B,LDB,X,LDX)
*
*        Factorize A in the array AFP
*
         CALL cpptrf(UPLO,N,AFP,INFO)
*
         WRITE (NOUT,*)
         IF (INFO.EQ.0) THEN
*
*           Compute solution in the array X
*
            CALL cpptrs(UPLO,N,NRHS,AFP,X,LDX,INFO)
*
*           Improve solution, and compute backward errors and
*           estimated bounds on the forward errors
*
            CALL cpprfs(UPLO,N,NRHS,AP,AFP,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,*) 'A is not positive-definite'
         END IF
      END IF
      STOP
*
99999 FORMAT ((5X,1P,4(e11.1,7X)))
      END
```

9.2 Program Data

```
F07GVF Example Program Data
  4  2                                                      :Values of N and NRHS
  'L'                                                       :Value of UPLO
 (3.23, 0.00)
 (1.51, 1.92) ( 3.58, 0.00)
 (1.90,-0.84) (-0.23,-1.11) ( 4.09, 0.00)
 (0.42,-2.50) (-1.18,-1.37) ( 2.33, 0.14) ( 4.29, 0.00)  :End of matrix A
 ( 3.93, -6.14) ( 1.48,  6.58)
 ( 6.17,  9.42) ( 4.65, -4.75)
 (-7.17,-21.83) (-4.91,  2.29)
 ( 1.99,-14.38) ( 7.64,-10.79)                              :End of matrix B
```

9.3 Program Results

```
 F07GVF Example Program Results

 Solution(s)
                    1                 2
 1  ( 1.0000,-1.0000) (-1.0000, 2.0000)
 2  ( 0.0000, 3.0000) ( 3.0000,-4.0000)
 3  (-4.0000,-5.0000) (-2.0000, 3.0000)
 4  ( 2.0000, 1.0000) ( 4.0000,-5.0000)

 Backward errors (machine-dependent)
        5.5E-17           7.9E-17
 Estimated forward error bounds (machine-dependent)
        6.0E-14           7.2E-14
```