

NAG Library Routine Document

F07GVF (ZPPRFS)

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 (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, BERR,      &
                  WORK, RWORK, INFO)
INTEGER                N, NRHS, LDB, LDX, INFO
REAL (KIND=nag_wp)    FERR(NRHS), BERR(NRHS), RWORK(N)
COMPLEX (KIND=nag_wp) AP(*), AFP(*), B(LDB,*), X(LDX,*), WORK(2*N)
CHARACTER(1)          UPLO
```

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

3 Description

F07GVF (ZPPRFS) 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 F07GVF (ZPPRFS) 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 $U^H U$, 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: $N \geq 0$.
- 3: NRHS – INTEGER *Input*
On entry: r , the number of right-hand sides.
Constraint: NRHS ≥ 0 .
- 4: AP(*) – COMPLEX (KIND=nag_wp) array *Input*
Note: the dimension of the array AP must be at least $\max(1, N \times (N + 1)/2)$.
On entry: the n by n original Hermitian positive definite matrix A as supplied to F07GRF (ZPPTRF).
- 5: AFP(*) – COMPLEX (KIND=nag_wp) array *Input*
Note: the dimension of the array AFP must be at least $\max(1, N \times (N + 1)/2)$.
On entry: the Cholesky factor of A stored in packed form, as returned by F07GRF (ZPPTRF).
- 6: 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 .
- 7: LDB – INTEGER *Input*
On entry: the first dimension of the array B as declared in the (sub)program from which F07GVF (ZPPRFS) is called.
Constraint: LDB $\geq \max(1, N)$.
- 8: 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 F07GSF (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 (ZPPRFS) is called.
Constraint: LDX $\geq \max(1, N)$.

- 10: FERR(NRHS) – REAL (KIND=nag_wp) array Output
On exit: FERR(j) contains an estimated error bound for the j th solution vector, that is, the j th column of X , for $j = 1, 2, \dots, r$.
- 11: BERR(NRHS) – REAL (KIND=nag_wp) array Output
On exit: BERR(j) contains the component-wise backward error bound β for the j th solution vector, that is, the j th column of X , for $j = 1, 2, \dots, r$.
- 12: WORK($2 \times N$) – COMPLEX (KIND=nag_wp) array Workspace
- 13: RWORK(N) – REAL (KIND=nag_wp) array Workspace
- 14: 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 i th 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 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 $8n^2$ real operations.

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

9 Example

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

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

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 (ZPPTRF).

9.1 Program Text

```

Program f07gvfe

!      F07GVF Example Program Text

!      Mark 24 Release. NAG Copyright 2012.

!      .. Use Statements ..
Use nag_library, Only: nag_wp, x04dbf, zpprfs, zpptrf, zpptrs
!      .. Implicit None Statement ..
Implicit None
!      .. Parameters ..
Integer, Parameter          :: nin = 5, nout = 6
!      .. Local Scalars ..
Integer                     :: aplen, i, ifail, info, j, ldb, ldx, &
                             n, nrhs
Character (1)               :: uplo
!      .. Local Arrays ..
Complex (Kind=nag_wp), Allocatable :: afp(:), ap(:), b(:,,:), work(:), &
                             x(:,,:)
Real (Kind=nag_wp), Allocatable  :: berr(:), ferr(:), rwork(:)
Character (1)                   :: clabs(1), rlabs(1)
!      .. Executable Statements ..
Write (nout,*) 'F07GVF Example Program Results'
!      Skip heading in data file
Read (nin,*)
Read (nin,*) n, nrhs
ldb = n
ldx = n
aplen = n*(n+1)/2
Allocate (afp(aplen),ap(aplen),b(ldb,nrhs),work(2*n),x(ldx,n), &
         berr(nrhs),ferr(nrhs),rwork(n))

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

Read (nin,*) uplo
If (uplo=='U') Then
  Read (nin,*)((ap(i+j*(j-1)/2),j=i,n),i=1,n)
Else If (uplo=='L') Then
  Read (nin,*)((ap(i+(2*n-j)*(j-1)/2),j=1,i),i=1,n)
End If
Read (nin,*)(b(i,1:nrhs),i=1,n)

afp(1:aplen) = ap(1:aplen)
x(1:n,1:nrhs) = b(1:n,1:nrhs)

!      Factorize A in the array AFP
!      The NAG name equivalent of zpptrf is f07grf
Call zpptrf(uplo,n,afp,info)

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

!      Compute solution in the array X
!      The NAG name equivalent of zpptrs is f07gsf
Call zpptrs(uplo,n,nrhs,afp,x,ldx,info)

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

!      The NAG name equivalent of zpprfs is f07gvf
Call zpprfs(uplo,n,nrhs,ap,afp,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 f07gvfe

```

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

```
