NAG Library Routine Document

F07QVF (ZSPRFS)

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

F07QVF (ZSPRFS) returns error bounds for the solution of a complex symmetric 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 F07QVF (UPLO, N, NRHS, AP, AFP, IPIV, B, LDB, X, LDX, FERR,          &
                   BERR, WORK, RWORK, INFO)

INTEGER                 N, NRHS, IPIV(*), 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 ***zsprfs***.

3 Description

F07QVF (ZSPRFS) returns the backward errors and estimated bounds on the forward errors for the solution of a complex symmetric 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 F07QVF (ZSPRFS) 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: 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 $PUDU^{\mathrm{T}}P^{\mathrm{T}}$, where U is upper triangular.

UPLO = 'L'
: The lower triangular part of A is stored and A is factorized as $PLDL^{\mathrm{T}}P^{\mathrm{T}}$, 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: NRHS – INTEGER *Input*

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

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

4: AP($*$) – COMPLEX (KIND=nag_wp) array *Input*

Note: the dimension of the array AP must be at least $\max(1, \mathrm{N} \times (\mathrm{N}+1)/2)$.

On entry: the n by n original symmetric matrix A as supplied to F07QRF (ZSPTRF).

5: AFP($*$) – COMPLEX (KIND=nag_wp) array *Input*

Note: the dimension of the array AFP must be at least $\max(1, \mathrm{N} \times (\mathrm{N}+1)/2)$.

On entry: the factorization of A stored in packed form, as returned by F07QRF (ZSPTRF).

6: IPIV($*$) – INTEGER array *Input*

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

On entry: details of the interchanges and the block structure of D, as returned by F07QRF (ZSPTRF).

7: 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.

8: LDB – INTEGER *Input*

On entry: the first dimension of the array B as declared in the (sub)program from which F07QVF (ZSPRFS) is called.

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

9: 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 F07QSF (ZSPTRS).

On exit: the improved solution matrix X.

10: LDX – INTEGER *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which F07QVF (ZSPRFS) is called.

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

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

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

12: BERR(NRHS) – REAL (KIND=nag_wp) array *Output*

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

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

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

15: INFO – INTEGER *Output*

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

6 Error Indicators and Warnings

Errors or warnings detected by the routine:

$\text{INFO} < 0$

If $\text{INFO} = -i$, 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 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 F07PHF (DSPRFS).

9 Example

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

$$A = \begin{pmatrix} -0.39 - 0.71i & 5.14 - 0.64i & -7.86 - 2.96i & 3.80 + 0.92i \\ 5.14 - 0.64i & 8.86 + 1.81i & -3.52 + 0.58i & 5.32 - 1.59i \\ -7.86 - 2.96i & -3.52 + 0.58i & -2.83 - 0.03i & -1.54 - 2.86i \\ 3.80 + 0.92i & 5.32 - 1.59i & -1.54 - 2.86i & -0.56 + 0.12i \end{pmatrix}$$

and

$$B = \begin{pmatrix} -55.64 + 41.22i & -19.09 - 35.97i \\ -48.18 + 66.00i & -12.08 - 27.02i \\ -0.49 - \ \ 1.47i & 6.95 + 20.49i \\ -6.43 + 19.24i & -4.59 - 35.53i \end{pmatrix}.$$

Here A is symmetric, stored in packed form, and must first be factorized by F07QRF (ZSPTRF).

9.1 Program Text

```
    Program f07qvfe

!     F07QVF Example Program Text

!     Mark 24 Release. NAG Copyright 2012.

!     .. Use Statements ..
      Use nag_library, Only: nag_wp, x04dbf, zsprfs, zsptrf, zsptrs
!     .. 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(:)
      Integer, Allocatable              :: ipiv(:)
      Character (1)                     :: clabs(1), rlabs(1)
!     .. Executable Statements ..
      Write (nout,*) 'F07QVF 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),ipiv(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 zsptrf is f07qrf
      Call zsptrf(uplo,n,afp,ipiv,info)

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

!       Compute solution in the array X
!       The NAG name equivalent of zsptrs is f07qsf
        Call zsptrs(uplo,n,nrhs,afp,ipiv,x,ldx,info)

!       Improve solution, and compute backward errors and
```

```
!       estimated bounds on the forward errors

!       The NAG name equivalent of zsprfs is f07qvf
        Call zsprfs(uplo,n,nrhs,ap,afp,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 D is singular'
      End If

99999 Format ((5X,1P,4(E11.1,7X)))
    End Program f07qvfe
```

9.2 Program Data

```
F07QVF Example Program Data
  4  2                                                       :Values of N and NRHS
  'L'                                                        :Value of UPLO
 (-0.39,-0.71)
 ( 5.14,-0.64) ( 8.86, 1.81)
 (-7.86,-2.96) (-3.52, 0.58) (-2.83,-0.03)
 ( 3.80, 0.92) ( 5.32,-1.59) (-1.54,-2.86) (-0.56, 0.12)  :End of matrix A
 (-55.64, 41.22) (-19.09,-35.97)
 (-48.18, 66.00) (-12.08,-27.02)
 ( -0.49, -1.47) (  6.95, 20.49)
 ( -6.43, 19.24) ( -4.59,-35.53)                             :End of matrix B
```

9.3 Program Results

```
 F07QVF Example Program Results

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

 Backward errors (machine-dependent)
        8.9E-17           7.3E-17
 Estimated forward error bounds (machine-dependent)
        1.2E-14           1.2E-14
```