

# NAG Fortran Library Routine Document

## F07NVF (CSYRFS/ZSYRFS)

**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

F07NVF (CSYRFS/ZSYRFS) returns error bounds for the solution of a complex symmetric 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 F07NVF(UPLO, N, NRHS, A, LDA, AF, LDAF, IPIV, B, LDB, X, LDX,
1              FERR, BERR, WORK, RWORK, INFO)
ENTRY          csyrfs (UPLO, N, NRHS, A, LDA, AF, LDAF, IPIV, B, LDB, X, LDX,
1              FERR, BERR, WORK, RWORK, INFO)
INTEGER        N, NRHS, LDA, LDAF, IPIV(*), LDB, LDX, INFO
real          FERR(*), BERR(*), RWORK(*)
complex      A(LDA,*), AF(LDAF,*), 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 symmetric 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 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 backward error*  $\beta$ . 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:* indicates whether the upper or lower triangular part of  $A$  is stored and how  $A$  is to be factorized, as follows:

if UPLO = 'U', the upper triangular part of  $A$  is stored and  $A$  is factorized as  $PUDU^T P^T$ , where  $U$  is upper triangular;

if UPLO = 'L', the lower triangular part of  $A$  is stored and  $A$  is factorized as  $PLDL^T P^T$ , where  $L$  is lower triangular.

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

- 2:    N – INTEGER *Input*  
*On entry:*  $n$ , the order of the matrix  $A$ .  
*Constraint:*  $N \geq 0$ .
  
- 3:    NRHS – INTEGER *Input*  
*On entry:*  $r$ , the number of right-hand sides.  
*Constraint:*  $NRHS \geq 0$ .
  
- 4:    A(LDA,\*) – **complex** array *Input*  
**Note:** the second dimension of the array  $A$  must be at least  $\max(1, N)$ .  
*On entry:* the  $n$  by  $n$  original symmetric matrix  $A$  as supplied to F07NRF (CSYTRF/ZSYTRF).
  
- 5:    LDA – INTEGER *Input*  
*On entry:* the first dimension of the array  $A$  as declared in the (sub)program from which F07NVF (CSYRFS/ZSYRFS) is called.  
*Constraint:*  $LDA \geq \max(1, N)$ .
  
- 6:    AF(LDAF,\*) – **complex** array *Input*  
**Note:** the second dimension of the array  $AF$  must be at least  $\max(1, N)$ .  
*On entry:* details of the factorization of  $A$ , as returned by F07NRF (CSYTRF/ZSYTRF).
  
- 7:    LDAF – INTEGER *Input*  
*On entry:* the first dimension of the array  $AF$  as declared in the (sub)program from which F07NVF (CSYRFS/ZSYRFS) is called.  
*Constraint:*  $LDAF \geq \max(1, N)$ .
  
- 8:    IPIV(\*) – INTEGER array *Input*  
**Note:** the dimension of the array IPIV must be at least  $\max(1, N)$ .  
*On entry:* details of the interchanges and the block structure of  $D$ , as returned by F07NRF (CSYTRF/ZSYTRF).
  
- 9:    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  $B$ .
  
- 10:    LDB – INTEGER *Input*  
*On entry:* the first dimension of the array  $B$  as declared in the (sub)program from which F07NVF (CSYRFS/ZSYRFS) is called.  
*Constraint:*  $LDB \geq \max(1, N)$ .

- 11:  $X(LDX,*)$  – **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 F07NSF (CSYTRS/ZSYTRS).  
*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 F07NVF (CSYRFS/ZSYRFS) is called.  
*Constraint:*  $LDX \geq \max(1, N)$ .
- 13:  $FERR(*)$  – **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  $j$ th solution vector, that is, the  $j$ th column of  $X$ , for  $j = 1, 2, \dots, r$ .
- 14:  $BERR(*)$  – **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  $\beta$  for the  $j$ th solution vector, that is, the  $j$ th column of  $X$ , for  $j = 1, 2, \dots, r$ .
- 15:  $WORK(*)$  – **complex** array *Workspace*  
**Note:** the dimension of the array  $WORK$  must be at least  $\max(1, 2 * N)$ .
- 16:  $RWORK(*)$  – **real** array *Workspace*  
**Note:** the dimension of the array  $RWORK$  must be at least  $\max(1, N)$ .
- 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  $i$ 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 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 F07MHF (SSYRFS/DSYRFS).

## 9 Example

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

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

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 and must first be factorized by F07NRF (CSYTRF/ZSYTRF).

### 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.

```
*      F07NVF Example Program Text
*      Mark 15 Release. NAG Copyright 1991.
*      .. Parameters ..
      INTEGER          NIN, NOUT
      PARAMETER        (NIN=5,NOUT=6)
      INTEGER          NMAX, NRHMAX, LDA, LWORK, LDAF, LDB, LDX
      PARAMETER        (NMAX=8,NRHMAX=NMAX,LDA=NMAX,LWORK=64*NMAX,
+                      LDAF=NMAX,LDB=NMAX,LDX=NMAX)
*      .. Local Scalars ..
      INTEGER          I, IFAIL, INFO, J, N, NRHS
      CHARACTER        UPLO
*      .. Local Arrays ..
      complex          A(LDA,NMAX), AF(LDAF,NMAX), B(LDB,NRHMAX),
+                      WORK(LWORK), X(LDX,NMAX)
      real             BERR(NRHMAX), FERR(NRHMAX), RWORK(NMAX)
      INTEGER          IPIV(NMAX)
      CHARACTER        CLABS(1), RLABS(1)
*      .. External Subroutines ..
      EXTERNAL         csyrfs, csytrf, csytrs, F06TFF, X04DBF
*      .. Executable Statements ..
      WRITE (NOUT,*) 'F07NVF 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 AF and B to X
*
      READ (NIN,*) UPLO
      IF (UPLO.EQ.'U') THEN
        READ (NIN,*) ((A(I,J),J=I,N),I=1,N)
      ELSE IF (UPLO.EQ.'L') THEN
        READ (NIN,*) ((A(I,J),J=1,I),I=1,N)
      END IF
      READ (NIN,*) ((B(I,J),J=1,NRHS),I=1,N)
      CALL F06TFF(UPLO,N,N,A,LDA,AF,LDAF)
```

```

      CALL F06TFF('General',N,NRHS,B,LDB,X,LDX)
*
*      Factorize A in the array AF
*
      CALL csytrf(UPLO,N,AF,LDAF,IPIV,WORK,LWORK,INFO)
*
      WRITE (NOUT,*)
      IF (INFO.EQ.0) THEN
*
*          Compute solution in the array X
*
          CALL csytrs(UPLO,N,NRHS,AF,LDAF,IPIV,X,LDX,INFO)
*
*          Improve solution, and compute backward errors and
*          estimated bounds on the forward errors
*
          CALL csyrfs(UPLO,N,NRHS,A,LDA,AF,LDAF,IPIV,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,*) 'The factor D is singular'
        END IF
      END IF
      STOP
*
99999 FORMAT ((5X,1P,4(E11.1,7X)))
END

```

## 9.2 Program Data

F07NVF 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

F07NVF 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)

1.1E-16	7.3E-17
---------	---------

Estimated forward error bounds (machine-dependent)

1.3E-14	1.3E-14
---------	---------