# NAG Library Routine Document

# F07NVF (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 (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, FERR, BERR, WORK, RWORK, INFO)
INTEGER                 N, NRHS, LDA, LDAF, IPIV(*), LDB, LDX, INFO
REAL (KIND=nag_wp)      FERR(NRHS), BERR(NRHS), RWORK(N)
COMPLEX (KIND=nag_wp) A(LDA,*), AF(LDAF,*), B(LDB,*), X(LDX,*),             &
                        WORK(2*N)
CHARACTER(1)            UPLO
```

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

## 3 Description

F07NVF (ZSYRFS) 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 F07NVF (ZSYRFS) 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

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

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 $PUDU^{T}P^{T}$, where $U$ is upper triangular.

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

3: NRHS – INTEGER *Input*

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

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

4: $\mathrm{A}(\mathrm{LDA},*)$ – COMPLEX (KIND=nag_wp) array *Input*

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

*On entry*: the $n$ by $n$ original symmetric matrix $A$ as supplied to F07NRF (ZSYTRF).

5: LDA – INTEGER *Input*

*On entry*: the first dimension of the array A as declared in the (sub)program from which F07NVF (ZSYRFS) is called.

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

6: $\mathrm{AF}(\mathrm{LDAF},*)$ – COMPLEX (KIND=nag_wp) array *Input*

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

*On entry*: details of the factorization of $A$, as returned by F07NRF (ZSYTRF).

7: LDAF – INTEGER *Input*

*On entry*: the first dimension of the array AF as declared in the (sub)program from which F07NVF (ZSYRFS) is called.

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

8: $\mathrm{IPIV}(*)$ – 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 F07NRF (ZSYTRF).

9: $\mathrm{B}(\mathrm{LDB},*)$ – 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 F07NVF (ZSYRFS) is called.

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

11: $\text{X}(\text{LDX}, *)$ – 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 F07NSF (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 (ZSYRFS) is called.

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

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

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

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

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

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

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

17: INFO – INTEGER *Output*

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

## 6 Error Indicators and Warnings

$\text{INFO} < 0$

If $\text{INFO} = -i$, argument $i$ 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 Parallelism and Performance

F07NVF (ZSYRFS) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

F07NVF (ZSYRFS) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

## 9 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 F07MHF (DSYRFS).

## 10 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 and must first be factorized by F07NRF (ZSYTRF).

### 10.1 Program Text

```
    Program f07nvfe

!     F07NVF Example Program Text

!     Mark 25 Release. NAG Copyright 2014.

!     .. Use Statements ..
      Use nag_library, Only: f06tff, nag_wp, x04dbf, zsyrfs, zsytrf, zsytrs
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: i, ifail, info, lda, ldaf, ldb, ldx, &
                                          lwork, n, nrhs
      Character (1)                    :: uplo
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: a(:,:), af(:,:), b(:,:), work(:),  &
                                            x(:,:)
      Real (Kind=nag_wp), Allocatable  :: berr(:), ferr(:), rwork(:)
      Integer, Allocatable             :: ipiv(:)
      Character (1)                    :: clabs(1), rlabs(1)
!     .. Executable Statements ..
      Write (nout,*) 'F07NVF Example Program Results'
!     Skip heading in data file
      Read (nin,*)
      Read (nin,*) n, nrhs
      lda = n
      ldaf = n
```

```
      ldb = n
      ldx = n
      lwork = 64*n
      Allocate (a(lda,n),af(ldaf,n),b(ldb,nrhs),work(lwork),x(ldx,n), &
        berr(nrhs),ferr(nrhs),rwork(n),ipiv(n))

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

      Read (nin,*) uplo
      If (uplo=='U') Then
        Read (nin,*)(a(i,i:n),i=1,n)
      Else If (uplo=='L') Then
        Read (nin,*)(a(i,1:i),i=1,n)
      End If
      Read (nin,*)(b(i,1:nrhs),i=1,n)
      Call f06tff(uplo,n,n,a,lda,af,ldaf)
      x(1:n,1:nrhs) = b(1:n,1:nrhs)

!     Factorize A in the array AF
!     The NAG name equivalent of zsytrf is f07nrf
      Call zsytrf(uplo,n,af,ldaf,ipiv,work,lwork,info)

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

!       Compute solution in the array X
!       The NAG name equivalent of zsytrs is f07nsf
        Call zsytrs(uplo,n,nrhs,af,ldaf,ipiv,x,ldx,info)

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

!       The NAG name equivalent of zsyrfs is f07nvf
        Call zsyrfs(uplo,n,nrhs,a,lda,af,ldaf,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 f07nvfe
```

## 10.2 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
```

### 10.3 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)
        8.9E-17           7.3E-17
Estimated forward error bounds (machine-dependent)
        1.2E-14           1.2E-14
```