NAG Toolbox

nag_lapack_zsyrfs (f07nv)

1 Purpose

nag_lapack_zsyrfs (f07nv) 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 Syntax

```
[x, ferr, berr, info] = nag_lapack_zsyrfs(uplo, a, af, ipiv, b, x, 'n', n,
'nrhs_p', nrhs_p)
```

```
[x, ferr, berr, info] = f07nv(uplo, a, af, ipiv, b, x, 'n', n, 'nrhs_p', nrhs_p)
```

3 Description

nag_lapack_zsyrfs (f07nv) 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 function handles each right-hand side vector (stored as a column of the matrix B) independently, so we describe the function of nag_lapack_zsyrfs (f07nv) in terms of a single right-hand side b and solution x.

Given a computed solution x, the function 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 function 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

5.1 Compulsory Input Parameters

1: **uplo** – CHARACTER(1)

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 $PLDL^{T}P^{T}$, where L is lower triangular.

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

2: **a**$(lda,:)$ – COMPLEX (KIND=nag_wp) array

The first dimension of the array **a** must be at least $\max(1,\mathbf{n})$.

The second dimension of the array **a** must be at least $\max(1,\mathbf{n})$.

The n by n original symmetric matrix A as supplied to nag_lapack_zsytrf (f07nr).

3: **af**$(ldaf,:)$ – COMPLEX (KIND=nag_wp) array

The first dimension of the array **af** must be at least $\max(1,\mathbf{n})$.

The second dimension of the array **af** must be at least $\max(1,\mathbf{n})$.

Details of the factorization of A, as returned by nag_lapack_zsytrf (f07nr).

4: **ipiv**$(:)$ – INTEGER array

The dimension of the array **ipiv** must be at least $\max(1,\mathbf{n})$

Details of the interchanges and the block structure of D, as returned by nag_lapack_zsytrf (f07nr).

5: **b**$(ldb,:)$ – COMPLEX (KIND=nag_wp) array

The first dimension of the array **b** must be at least $\max(1,\mathbf{n})$.

The second dimension of the array **b** must be at least $\max(1,\mathbf{nrhs\_p})$.

The n by r right-hand side matrix B.

6: **x**$(ldx,:)$ – COMPLEX (KIND=nag_wp) array

The first dimension of the array **x** must be at least $\max(1,\mathbf{n})$.

The second dimension of the array **x** must be at least $\max(1,\mathbf{nrhs\_p})$.

The n by r solution matrix X, as returned by nag_lapack_zsytrs (f07ns).

5.2 Optional Input Parameters

1: **n** – INTEGER

Default: the first dimension of the arrays **a**, **af**, **b**, **x** and the second dimension of the arrays **a**, **af**, **ipiv**.

n, the order of the matrix A.

Constraint: $\mathbf{n} \geq 0$.

2: **nrhs_p** – INTEGER

Default: the second dimension of the arrays **b**, **x**. (An error is raised if these dimensions are not equal.)

r, the number of right-hand sides.

Constraint: $\mathbf{nrhs\_p} \geq 0$.

5.3 Output Parameters

1: **x**$(ldx,:)$ – COMPLEX (KIND=nag_wp) array

The first dimension of the array **x** will be $\max(1, \mathbf{n})$.

The second dimension of the array **x** will be $\max(1, \mathbf{nrhs\_p})$.

The improved solution matrix X.

2: **ferr**(**nrhs_p**) – REAL (KIND=nag_wp) array

ferr(j) contains an estimated error bound for the jth solution vector, that is, the jth column of X, for $j = 1, 2, \ldots, r$.

3: **berr**(**nrhs_p**) – REAL (KIND=nag_wp) array

berr(j) contains the component-wise backward error bound β for the jth solution vector, that is, the jth column of X, for $j = 1, 2, \ldots, r$.

4: **info** – INTEGER

info $= 0$ unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

info < 0

If **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 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 function is nag_lapack_dsyrfs (f07mh).

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 and must first be factorized by nag_lapack_zsytrf (f07nr).

9.1 Program Text

```
function f07nv_example

fprintf('f07nv example results\n\n');

% Complex symmetrix matrix A, lower triangle stored.
uplo = 'L';
a = [-0.39 - 0.71i,  0    + 0i,     0    + 0i,     0    + 0i;
      5.14 - 0.64i,  8.86 + 1.81i,  0    + 0i,     0    + 0i;
     -7.86 - 2.96i, -3.52 + 0.58i, -2.83 - 0.03i,  0    + 0i;
      3.80 + 0.92i,  5.32 - 1.59i, -1.54 - 2.86i, -0.56 + 0.12i];

% Factorize A
[af, ipiv, info] = f07nr( ...
                          uplo, a);

% RHS
b = [ -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];

% Solve Ax=b
[x, info] = f07ns( ...
                   uplo, af, ipiv, b);

% Refine
[x, ferr, berr, info] = f07nv( ...
                               uplo, a, af, ipiv, b, x);

disp('Solution(s)');
disp(x);
fprintf('Backward errors (machine-dependent)\n   ')
fprintf('%11.1e', berr);
fprintf('\nEstimated forward error bounds (machine-dependent)\n   ')
fprintf('%11.1e', ferr);
fprintf('\n');
```

9.2 Program Results

```
     f07nv example results

Solution(s)
   1.0000 - 1.0000i  -2.0000 - 1.0000i
  -2.0000 + 5.0000i   1.0000 - 3.0000i
   3.0000 - 2.0000i   3.0000 + 2.0000i
  -4.0000 + 3.0000i  -1.0000 + 1.0000i

Backward errors (machine-dependent)
       5.5e-17    7.3e-17
Estimated forward error bounds (machine-dependent)
       1.2e-14    1.2e-14
```