NAG Toolbox

nag_lapack_zhprfs (f07pv)

1 Purpose

nag_lapack_zhprfs (f07pv) returns error bounds for the solution of a complex Hermitian indefinite 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 Syntax

```
[x, ferr, berr, info] = nag_lapack_zhprfs(uplo, ap, afp, ipiv, b, x, 'n', n, 'nrhs_p', nrhs_p)
[x, ferr, berr, info] = f07pv(uplo, ap, afp, ipiv, b, x, 'n', n, 'nrhs_p', nrhs_p)
```

3 Description

nag_lapack_zhprfs (f07pv) returns the backward errors and estimated bounds on the forward errors for the solution of a complex Hermitian indefinite system of linear equations with multiple right-hand sides $AX = B$, using packed storage. 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_zhprfs (f07pv) 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

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

Then the function 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

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

uplo = 'L'
The lower triangular part of A is stored and A is factorized as $PLDL^{\mathrm{H}}P^{\mathrm{T}}$, where L is lower triangular.

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

2: **ap**(:) – COMPLEX (KIND=nag_wp) array

The dimension of the array **ap** must be at least $\max(1, \mathbf{n} \times (\mathbf{n}+1)/2)$

The n by n original Hermitian matrix A as supplied to nag_lapack_zhptrf (f07pr).

3: **afp**(:) – COMPLEX (KIND=nag_wp) array

The dimension of the array **afp** must be at least $\max(1, \mathbf{n} \times (\mathbf{n}+1)/2)$

The factorization of A stored in packed form, as returned by nag_lapack_zhptrf (f07pr).

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_zhptrf (f07pr).

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_zhptrs (f07ps).

5.2 Optional Input Parameters

1: **n** – INTEGER

Default: the first dimension of the arrays **b**, **x** and the dimension of the array **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 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 function is nag_lapack_dsprfs (f07ph).

9 Example

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

$$A = \begin{pmatrix} -1.36 + 0.00i & 1.58 + 0.90i & 2.21 - 0.21i & 3.91 + 1.50i \\ 1.58 - 0.90i & -8.87 + 0.00i & -1.84 - 0.03i & -1.78 + 1.18i \\ 2.21 + 0.21i & -1.84 + 0.03i & -4.63 + 0.00i & 0.11 + 0.11i \\ 3.91 - 1.50i & -1.78 - 1.18i & 0.11 - 0.11i & -1.84 + 0.00i \end{pmatrix}$$

and

$$B = \begin{pmatrix} 7.79 + 5.48i & -35.39 + 18.01i \\ -0.77 - 16.05i & 4.23 - 70.02i \\ -9.58 + 3.88i & -24.79 - 8.40i \\ 2.98 - 10.18i & 28.68 - 39.89i \end{pmatrix}.$$

Here A is Hermitian indefinite, stored in packed form, and must first be factorized by nag_lapack_zhptrf (f07pr).

9.1 Program Text

```
     function f07pv_example

fprintf('f07pv example results\n\n');

% Hermitian indefinite matrix A (Lower triangular part stored in packed form)
uplo = 'L';
n = nag_int(4);
ap = [-1.36 + 0i;  1.58 - 0.9i;   2.21 + 0.21i;  3.91 - 1.5i;
                 -8.87 + 0i;    -1.84 + 0.03i; -1.78 - 1.18i;
                                -4.63 + 0i;     0.11 - 0.11i;
                                               -1.84 + 0i];

% Factorize
[apf, ipiv, info] = f07pr( ...
                           uplo, n, ap);

% RHS
b = [ 7.79 +  5.48i, -35.39 + 18.01i;
     -0.77 - 16.05i,   4.23 - 70.02i;
     -9.58 +  3.88i, -24.79 -  8.40i;
      2.98 - 10.18i,  28.68 - 39.89i];

% Solve
[x, info] = f07ps( ...
                   uplo, apf, ipiv, b);

% Refine
[x, ferr, berr, info] = f07pv( ...
                               uplo, ap, apf, 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

```
     f07pv example results

Solution(s)
   1.0000 - 1.0000i   3.0000 - 4.0000i
  -1.0000 + 2.0000i  -1.0000 + 5.0000i
   3.0000 - 2.0000i   7.0000 - 2.0000i
   2.0000 + 1.0000i  -8.0000 + 6.0000i

Backward errors (machine-dependent)
       5.6e-17    8.1e-17
Estimated forward error bounds (machine-dependent)
       2.5e-15    3.0e-15
```