NAG Toolbox

nag_lapack_zpbrfs (f07hv)

1 Purpose

nag_lapack_zpbrfs (f07hv) returns error bounds for the solution of a complex Hermitian positive definite band 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_zpbrfs(uplo, kd, ab, afb, b, x, 'n', n,
'nrhs_p', nrhs_p)
```

```
[x, ferr, berr, info] = f07hv(uplo, kd, ab, afb, b, x, 'n', n, 'nrhs_p', nrhs_p)
```

3 Description

nag_lapack_zpbrfs (f07hv) returns the backward errors and estimated bounds on the forward errors for the solution of a complex Hermitian positive definite band 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_zpbrfs (f07hv) 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

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

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

uplo = 'L'

The lower triangular part of A is stored and A is factorized as LL^{H}, where L is lower triangular.

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

2: **kd** – INTEGER

k_d, the number of superdiagonals or subdiagonals of the matrix A.

Constraint: **kd** ≥ 0.

3: **ab**$(ldab,:)$ – COMPLEX (KIND=nag_wp) array

The first dimension of the array **ab** must be at least **kd** $+ 1$.

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

The n by n original Hermitian positive definite band matrix A as supplied to nag_lapack_zpbtrf (f07hr).

4: **afb**$(ldafb,:)$ – COMPLEX (KIND=nag_wp) array

The first dimension of the array **afb** must be at least **kd** $+ 1$.

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

The Cholesky factor of A, as returned by nag_lapack_zpbtrf (f07hr).

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_zpbtrs (f07hs).

5.2 Optional Input Parameters

1: **n** – INTEGER

Default: the second dimension of the array **ab**.

n, the order of the matrix A.

Constraint: **n** ≥ 0.

2: **nrhs_p** – INTEGER

Default: the second dimension of the arrays **b**, **x**.

r, the number of right-hand sides.

Constraint: **nrhs_p** ≥ 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 $\max(1, \textbf{nrhs\_p})$.

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 $32nk$ real floating-point operations. Each step of iterative refinement involves an additional $48nk$ real operations. This assumes $n \gg k$. 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 $16nk$ real operations.

The real analogue of this function is nag_lapack_dpbrfs (f07hh).

9 Example

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

$$A = \begin{pmatrix} 9.39 + 0.00i & 1.08 - 1.73i & 0.00 + 0.00i & 0.00 + 0.00i \\ 1.08 + 1.73i & 1.69 + 0.00i & -0.04 + 0.29i & 0.00 + 0.00i \\ 0.00 + 0.00i & -0.04 - 0.29i & 2.65 + 0.00i & -0.33 + 2.24i \\ 0.00 + 0.00i & 0.00 + 0.00i & -0.33 - 2.24i & 2.17 + 0.00i \end{pmatrix}$$

and

$$B = \begin{pmatrix} -12.42 + 68.42i & 54.30 - 56.56i \\ -9.93 + 0.88i & 18.32 + 4.76i \\ -27.30 - 0.01i & -4.40 + 9.97i \\ 5.31 + 23.63i & 9.43 + 1.41i \end{pmatrix}.$$

Here A is Hermitian positive definite, and is treated as a band matrix, which must first be factorized by nag_lapack_zpbtrf (f07hr).

9.1 Program Text

```
     function f07hv_example

fprintf('f07hv example results\n\n');

% A in Hermitian banded format
uplo = 'L';
kd = nag_int(1);
n  = nag_int(4);
ab = [ 9.39 + 0i,     1.69 + 0i,      2.65 + 0i,      2.17 + 0i;
       1.08 + 1.73i, -0.04 - 0.29i,  -0.33 - 2.24i    0    + 0i];

% RHS
b = [-12.42 + 68.42i,  54.30 - 56.56i;
      -9.93 +  0.88i,  18.32 +  4.76i;
     -27.30 -  0.01i,  -4.40 +  9.97i;
       5.31 + 23.63i,   9.43 +  1.41i];

% Factorize
[abf, info] = f07hr( ...
                     uplo, kd, ab);

% Solve
[x, info] = f07hs( ...
                   uplo, kd, abf, b);

% Iterative refinement
[x, ferr, berr, info] = f07hv( ...
                               uplo, kd, ab, abf, b, x);
disp('Solution(s)');
disp(x);
fprintf('Forward  error bounds = %10.1e  %10.1e\n',ferr);
fprintf('Backward error bounds = %10.1e  %10.1e\n',berr);
```

9.2 Program Results

```
     f07hv example results

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

Forward  error bounds =    3.7e-14     3.1e-14
Backward error bounds =    1.0e-16     6.7e-17
```