NAG Toolbox

nag_lapack_dpbrfs (f07hh)

1 Purpose

nag_lapack_dpbrfs (f07hh) returns error bounds for the solution of a real symmetric 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_dpbrfs(uplo, kd, ab, afb, b, x, 'n', n,
'nrhs_p', nrhs_p)
```

```
[x, ferr, berr, info] = f07hh(uplo, kd, ab, afb, b, x, 'n', n, 'nrhs_p', nrhs_p)
```

3 Description

nag_lapack_dpbrfs (f07hh) returns the backward errors and estimated bounds on the forward errors for the solution of a real symmetric 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_dpbrfs (f07hh) 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 $U^{\mathrm{T}}U$, where U is upper triangular.

uplo $=$ 'L'

The lower triangular part of A is stored and A is factorized as LL^{T}, 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**$(ldab, :)$ – REAL (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 symmetric positive definite band matrix A as supplied to nag_lapack_dpbtrf (f07hd).

4: **afb**$(ldafb, :)$ – REAL (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_dpbtrf (f07hd).

5: **b**$(ldb, :)$ – REAL (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**$(ldx, :)$ – REAL (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_dpbtrs (f07he).

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**$(ldx, :)$ – REAL (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 $8nk$ floating-point operations. Each step of iterative refinement involves an additional $12nk$ 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 4 or 5 and never more than 11. Each solution involves approximately $4nk$ operations.

The complex analogue of this function is nag_lapack_zpbrfs (f07hv).

9 Example

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

$$A = \begin{pmatrix} 5.49 & 2.68 & 0.00 & 0.00 \\ 2.68 & 5.63 & -2.39 & 0.00 \\ 0.00 & -2.39 & 2.60 & -2.22 \\ 0.00 & 0.00 & -2.22 & 5.17 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 22.09 & 5.10 \\ 9.31 & 30.81 \\ -5.24 & -25.82 \\ 11.83 & 22.90 \end{pmatrix}.$$

Here A is symmetric and positive definite, and is treated as a band matrix, which must first be factorized by nag_lapack_dpbtrf (f07hd).

9.1 Program Text

```
     function f07hh_example

fprintf('f07hh example results\n\n');

% Symmetric banded matrix A in ab.
uplo = 'L';
kd = nag_int(1);
ab = [5.49,  5.63,  2.6,  5.17;
      2.68, -2.39, -2.22, 0   ];

% Factorize A
[abf, info] = f07hd( ...
                     uplo, kd, ab);

% RHS
b = [22.09,   5.10;
      9.31,  30.81;
     -5.24, -25.82;
     11.83,  22.90];
% Solve Ax = B
[x, info] = f07he( ...
                   uplo, kd, abf, b);

% Iterative refinement
[x, ferr, berr, info] = f07hh(...
                            uplo, kd, ab, abf, b, x);

disp('Solution');
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

```
     f07hh example results

Solution
    5.0000   -2.0000
   -2.0000    6.0000
   -3.0000   -1.0000
    1.0000    4.0000

Forward  error bounds =    2.1e-14     2.8e-14
Backward error bounds =    8.6e-17     1.1e-16
```