NAG Toolbox for MATLAB

f07bh

1 Purpose

f07bh returns error bounds for the solution of a real band system of linear equations with multiple right-hand sides, $AX = B$ or $A^{T}X = B$. It improves the solution by iterative refinement, in order to reduce the backward error as much as possible.

2 Syntax

```
[x, ferr, berr, info] = f07bh(trans, kl, ku, ab, afb, ipiv, b, x, 'n',
n, 'nrhs_p', nrhs_p)
```

3 Description

f07bh returns the backward errors and estimated bounds on the forward errors for the solution of a real band system of linear equations with multiple right-hand sides $AX = B$ or $A^{T}X = B$. The function handles each right-hand side vector (stored as a column of the matrix B) independently, so we describe the function of f07bh 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:

$$\max_{i}|x_{i} - \hat{x}_{i}| / \max_{i}|x_{i}|$$

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: **trans – string**

Indicates the form of the linear equations for which X is the computed solution.

trans = 'N'

The linear equations are of the form $AX = B$.

trans = 'T' or 'C'

The linear equations are of the form $A^{T}X = B$.

Constraint: **trans** = 'N', 'T' or 'C'.

2: **kl – int32 scalar**

k_l, the number of subdiagonals within the band of the matrix A.

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

3: **ku – int32 scalar**

k_u, the number of superdiagonals within the band of the matrix A.

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

4: **ab(ldab,*) – double array**

The first dimension of the array **ab** must be at least $\mathbf{kl} + \mathbf{ku} + 1$

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

The original n by n band matrix A as supplied to f07bd.

The matrix is stored in rows 1 to $k_l + k_u + 1$, more precisely, the element A_{ij} must be stored in

$$\mathbf{ab}(k_u + 1 + i - j, j) \qquad \text{for } \max(1j - k_u) \leq i \leq \min(nj + k_l).$$

5: **afb(ldafb,*) – double array**

The first dimension of the array **afb** must be at least $2 \times \mathbf{kl} + \mathbf{ku} + 1$

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

The LU factorization of A, as returned by f07bd.

6: **ipiv(*) – int32 array**

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

The pivot indices, as returned by f07bd.

7: **b(ldb,*) – double array**

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

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

The n by r right-hand side matrix B.

8: **x(ldx,*) – double array**

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

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

The n by r solution matrix X, as returned by f07be.

5.2 Optional Input Parameters

1: **n – int32 scalar**

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

n, the order of the matrix A.

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

2: **nrhs_p – int32 scalar**

Default: The second dimension of the array **b** The second dimension of the array **x**.

r, the number of right-hand sides.

Constraint: **nrhs_p** ≥ 0.

5.3 Input Parameters Omitted from the MATLAB Interface

ldab, ldafb, ldb, ldx, work, iwork

5.4 Output Parameters

1: **x(ldx,$*$) – double array**

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

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

The improved solution matrix X.

2: **ferr($*$) – double array**

Note: the dimension of the array **ferr** must be at least $\max(1, \mathbf{nrhs\_p})$.

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($*$) – double array**

Note: the dimension of the array **berr** must be at least $\max(1, \mathbf{nrhs\_p})$.

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 – int32 scalar**

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

6 Error Indicators and Warnings

Errors or warnings detected by the function:

info $= -i$

If **info** $= -i$, parameter i had an illegal value on entry. The parameters are numbered as follows:

1: **trans**, 2: **n**, 3: **kl**, 4: **ku**, 5: **nrhs_p**, 6: **ab**, 7: **ldab**, 8: **afb**, 9: **ldafb**, 10: **ipiv**, 11: **b**, 12: **ldb**, 13: **x**, 14: **ldx**, 15: **ferr**, 16: **berr**, 17: **work**, 18: **iwork**, 19: **info**.

It is possible that **info** refers to a parameter that is omitted from the MATLAB interface. This usually indicates that an error in one of the other input parameters has caused an incorrect value to be inferred.

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 $4n(k_l + k_u)$ floating-point operations. Each step of iterative refinement involves an additional $2n(4k_l + 3k_u)$ operations. This assumes $n \gg k_l$ and $n \gg k_u$. 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 $Ax = b$ or $A^{\mathrm{T}}x = b$; the number is usually 4 or 5 and never more than 11. Each solution involves approximately $2n(2k_l + k_u)$ operations.

The complex analogue of this function is f07bv.

9 Example

```
trans = 'N';
kl = int32(1);
ku = int32(2);
ab = [0, 0, -3.66, -2.13;
     0, 2.54, -2.73, 4.07;
     -0.23, 2.46, 2.46, -3.82;
     -6.98, 2.56, -4.78, 0];
afb = [0, -0.01423481108172949, -0.01423851186822769, -2.13;
     0, 0, -2.73, 4.07;
     0, 2.46, 2.46, -3.839143870881089;
     -6.98, 2.56, -5.932930470988539, -0.7269066639923109;
     0.0329512893982808, 0.9605233703438396, 0.8056726812110376, 0];
ipiv = [int32(2);
     int32(3);
     int32(3);
     int32(4)];
b = [4.42, -36.01;
     27.13, -31.67;
     -6.14, -1.16;
     10.5, -25.82];
x = [-1.999999999999998, 1.000000000000002;
     3.000000000000005, -3.999999999999995;
     1.000000000000003, 7.000000000000005;
     -4.000000000000004, -2.000000000000006];
[xOut, ferr, berr, info] = f07bh(trans, kl, ku, ab, afb, ipiv, b, x)
```

```
xOut =
   -2.0000    1.0000
    3.0000   -4.0000
    1.0000    7.0000
   -4.0000   -2.0000
ferr =
   1.0e-13 *
    0.1435
    0.1921
berr =
   1.0e-16 *
    0.5813
    0.9866
info =
           0
```