NAG Toolbox for MATLAB

f07bv

1 Purpose

f07bv returns error bounds for the solution of a complex band system of linear equations with multiple right-hand sides, $AX = B$, $A^{\mathrm{T}}X = B$ or $A^{\mathrm{H}}X = B$. It improves the solution by iterative refinement, in order to reduce the backward error as much as possible.

2 Syntax

```
[x, ferr, berr, info] = f07bv(trans, kl, ku, ab, afb, ipiv, b, x, 'n', n, 'nrhs_p', nrhs_p)
```

3 Description

f07bv returns the backward errors and estimated bounds on the forward errors for the solution of a complex band system of linear equations with multiple right-hand sides $AX = B$, $A^{\mathrm{T}}X = B$ or $A^{\mathrm{H}}X = B$. The function handles each right-hand side vector (stored as a column of the matrix B) independently, so we describe the function of f07bv 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: **trans – string**

Indicates the form of the linear equations for which X is the computed solution as follows:

trans = 'N'

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

trans = 'T'

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

trans = 'C'

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

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

2: **kl – int32 scalar**

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

Constraint: **kl** ≥ 0.

3: **ku – int32 scalar**

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

Constraint: **ku** ≥ 0.

4: **ab**(**ldab,**$*$) **– complex array**

The first dimension of the array **ab** must be at least **kl** + **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 f07br.

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,**$*$) **– complex array**

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

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

The LU factorization of A, as returned by f07br.

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 f07br.

7: **b**(**ldb,**$*$) **– complex 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,**$*$) **– complex 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 f07bs.

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

5.3 Input Parameters Omitted from the MATLAB Interface

ldab, ldafb, ldb, ldx, work, rwork

5.4 Output Parameters

1: **x(ldx,∗) – complex 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: **rwork**, 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 $16n(k_l + k_u)$ real floating-point operations. Each step of iterative refinement involves an additional $8n(4k_l + 3k_u)$ real 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 $A^{\mathrm{H}}x = b$; the number is usually 5 and never more than 11. Each solution involves approximately $8n(2k_l + k_u)$ real operations.

The real analogue of this function is f07bh.

9 Example

```
trans = 'N';
kl = int32(1);
ku = int32(2);
ab = [complex(0, +0), complex(0, +0), complex(0.97, -2.84), complex(0.59,
-0.48);
         complex(0,  +0),  complex(-2.05,  -0.85),  complex(-3.99,  +4.01),
complex(3.33, -1.04);
     complex(-1.65, +2.26), complex(-1.48, -1.75), complex(-1.06, +1.94),
complex(-0.46, -1.72);
        complex(0,  +6.3),  complex(-0.77,  +2.83),  complex(4.48,  -1.09),
complex(0, +0)];
afb = [complex(0, +0), complex(0, +0), complex(0, +0), complex(0.59, -
0.48);
     complex(0, +0), complex(0, +0), complex(-3.99, +4.01), complex(3.33,
-1.04);
         complex(0,  +0),  complex(-1.48,  -1.75),  complex(-1.06,  +1.94),
complex(-1.769209381609681, -1.858747281945787);
      complex(0, +6.3), complex(-0.77, +2.83), complex(4.930266941175471,
...
                      -3.008563740627192),    complex(0.4337749265901603,
+0.123252818156083);
                     complex(0.3587301587301587,    +0.2619047619047619),
complex(0.2314260728743743, ...
                    +0.6357648842047455),     complex(0.7604226619635511,
+0.2429442589267133), complex(0, +0)];
ipiv = [int32(2);
     int32(3);
     int32(3);
     int32(4)];
b = [complex(-1.06, +21.5), complex(12.85, +2.84);
     complex(-22.72, -53.9), complex(-70.22, +21.57);
     complex(28.24, -38.6), complex(-20.73, -1.23);
     complex(-34.56, +16.73), complex(26.01, +31.97)];
x    =    [complex(-2.999999999999998,    +1.999999999999998),    complex(1,
+5.999999999999997);
              complex(1.000000000000005,   -6.999999999999999),   complex(-
6.999999999999996, -4.000000000000003);
                      complex(-4.999999999999999,    +4.000000000000001),
complex(3.000000000000002, +5);
            complex(6.000000000000003,   -8.000000000000004),   complex(-8,
+1.999999999999996)];
[xOut, ferr, berr, info] = f07bv(trans, kl, ku, ab, afb, ipiv, b, x)
```

```
xOut =
  -3.0000 + 2.0000i   1.0000 + 6.0000i
```

```
   1.0000 - 7.0000i  -7.0000 - 4.0000i
  -5.0000 + 4.0000i   3.0000 + 5.0000i
   6.0000 - 8.0000i  -8.0000 + 2.0000i
ferr =
   1.0e-13 *
    0.3664
    0.4457
berr =
   1.0e-16 *
    0.8026
    0.8424
info =
           0
```