NAG Toolbox for MATLAB

f07av

1 Purpose

f07av returns error bounds for the solution of a complex 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] = f07av(trans, a, af, ipiv, b, x, 'n', n, 'nrhs_p', nrhs_p)
```

3 Description

f07av returns the backward errors and estimated bounds on the forward errors for the solution of a complex 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 f07av 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: **a**(**lda**,$*$) – **complex array**

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

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

The n by n original matrix A as supplied to f07ar.

3: **af**(**ldaf**,$*$) – **complex array**

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

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

The LU factorization of A, as returned by f07ar.

4: **ipiv**($*$) – **int32 array**

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

The pivot indices, as returned by f07ar.

5: **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.

6: **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 f07as.

5.2 Optional Input Parameters

1: **n** – **int32 scalar**

Default: The second dimension of the array **a** The second dimension of the array **af** The dimension of the array **ipiv**.

n, the order of the matrix A.

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

2: **nrhs_p** – **int32 scalar**

Default: 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

lda, ldaf, 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: **nrhs_p**, 4: **a**, 5: **lda**, 6: **af**, 7: **ldaf**, 8: **ipiv**, 9: **b**, 10: **ldb**, 11: **x**, 12: **ldx**, 13: **ferr**, 14: **berr**, 15: **work**, 16: **rwork**, 17: **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 $16n^2$ 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 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 $8n^2$ real operations.

The real analogue of this function is f07ah.

9 Example

```
trans = 'N';
a = [complex(-1.34, +2.55), complex(0.28, +3.17), complex(-6.39, -2.2),
complex(0.72, -0.92);
     complex(-0.17, -1.41), complex(3.31, -0.15), complex(-0.15, +1.34),
complex(1.29, +1.38);
     complex(-3.29, -2.39), complex(-1.91, +4.42), complex(-0.14, -1.35),
complex(1.72, +1.35);
     complex(2.41, +0.39), complex(-0.5600000000000001, +1.47), complex(-
0.83, ...
     -0.6899999999999999), complex(-1.96, +0.67)];
af = [complex(-3.29, -2.39), complex(-1.91, +4.42), complex(-0.14, -
1.35), complex(1.72, +1.35);
                    complex(0.2376120269469406,    +0.2559596521570857),
complex(4.895180634002975, ...
    -0.711362223485444), complex(-0.462279846639494, +1.696610587680362),
...
     complex(1.226852844063328, +0.6189731619114427);
           complex(-0.1019520808892006,  -0.7010135339437114),  complex(-
0.6691496431943212, ...
                +0.3688698547971655),    complex(-5.141410913810232,    -
1.129969734660953), ...
     complex(0.9982579915199449, +0.3850152275763778);
           complex(-0.5358546703595748,  +0.2707272529359829),  complex(-
0.2040217794587079, ...
                  +0.8601180679984047),    complex(0.008233047063940025,
+0.1210636819910635), ...
     complex(0.1482391810748419, -0.1252239986075848)];
ipiv = [int32(3);
     int32(2);
     int32(3);
     int32(4)];
b = [complex(26.26, +51.78), complex(31.32, -6.7);
     complex(6.43, -8.68), complex(15.86, -1.42);
     complex(-5.75, +25.31), complex(-2.15, +30.19);
     complex(1.16, +2.57), complex(-2.56, +7.55)];
x   =   [complex(0.9999999999999976,   +0.9999999999999998),   complex(-
0.9999999999999936, -2.000000000000014);
                      complex(1.999999999999999,    -2.999999999999999),
complex(4.999999999999995, +0.9999999999999998);
            complex(-4.000000000000002,  -5.000000000000001),  complex(-
2.999999999999997, +3.999999999999998);
                   complex(-1.748236337273752e-15,    +5.999999999999998),
complex(2.000000000000015, -3.000000000000014)];
[xOut, ferr, berr, info] = f07av(trans, a, af, ipiv, b, x)
```

```
xOut =
   1.0000 + 1.0000i  -1.0000 - 2.0000i
   2.0000 - 3.0000i   5.0000 + 1.0000i
  -4.0000 - 5.0000i  -3.0000 + 4.0000i
  -0.0000 + 6.0000i   2.0000 - 3.0000i
ferr =
   1.0e-13 *
    0.5707
    0.7613
berr =
   1.0e-16 *
    0.3355
    0.4891
info =
           0
```