NAG Toolbox for MATLAB

f07mv

1 Purpose

f07mv returns error bounds for the solution of a complex Hermitian indefinite 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] = f07mv(uplo, a, af, ipiv, b, x, 'n', n, 'nrhs_p',
nrhs_p)
```

3 Description

f07mv returns the backward errors and estimated bounds on the forward errors for the solution of a complex Hermitian indefinite 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 f07mv 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

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

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: **uplo – string**

Indicates 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 $PUDU^{H}P^{T}$, where U is upper triangular.

uplo = 'L'

The lower triangular part of A is stored and A is factorized as $PLDL^{H}P^{T}$, where L is lower triangular.

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

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 Hermitian matrix A as supplied to f07mr.

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 $\max(1, \mathbf{n})$

Details of the factorization of A, as returned by f07mr.

4: **ipiv($*$) – int32 array**

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

Details of the interchanges and the block structure of D, as returned by f07mr.

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 f07ms.

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 arrays **b**, **x**. (An error is raised if these dimensions are not equal.)

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 $\max(1, \textbf{nrhs\_p})$.

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 $\max(1, \textbf{nrhs\_p})$.

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: **uplo**, 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 $Ax = b$; 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 f07mh.

9 Example

```
uplo = 'L';
a = [complex(-1.36, +0), complex(0, 0), complex(0, 0), complex(0, 0);
      complex(1.58, -0.9), complex(-8.87, 0), complex(0, 0), complex(0,
0);
       complex(2.21, +0.21), complex(-1.84, +0.03), complex(-4.63, +0),
complex(0, 0);
      complex(3.91, -1.5), complex(-1.78, -1.18), complex(0.11, -0.11),
```

```
complex(-1.84, +0)];
b = [complex(7.79, +5.48), complex(-35.39, +18.01);
     complex(-0.77, -16.05), complex(4.23, -70.02);
     complex(-9.58, +3.88), complex(-24.79, -8.4);
     complex(2.98, -10.18), complex(28.68, -39.89)];
% factorise a in the array af
[af, ipiv, info] = f07mr(uplo, a);

% compute solution in the array x
[x, info] = f07ms(uplo, a, ipiv, b);

% improve solution, and compute backwards errors and estimated bounds
% on the forward errors
[xOut, ferr, berr, info] = f07mv(uplo, a, af, ipiv, b, x)
```

```
xOut =
   1.0000 - 1.0000i   3.0000 - 4.0000i
  -1.0000 + 2.0000i  -1.0000 + 5.0000i
   3.0000 - 2.0000i   7.0000 - 2.0000i
   2.0000 + 1.0000i  -8.0000 + 6.0000i
ferr =
   1.0e-14 *
    0.2483
    0.2958
berr =
   1.0e-16 *
    0.5624
    0.3558
info =
           0
```