NAG Toolbox for MATLAB

f07cv

1 Purpose

f07cv computes error bounds and refines the solution to a complex system of linear equations $AX = B$ or $A^{\mathrm{T}}X = B$ or $A^{\mathrm{H}}X = B$, where A is an n by n tridiagonal matrix and X and B are n by r matrices, using the LU factorization returned by f07cr and an initial solution returned by f07cs. Iterative refinement is used to reduce the backward error as much as possible.

2 Syntax

```
[x, ferr, berr, info] = f07cv(trans, dl, d, du, dlf, df, duf, du2, ipiv,
b, x, 'n', n, 'nrhs_p', nrhs_p)
```

3 Description

f07cv should normally be preceded by calls to f07cr and f07cs. f07cr uses Gaussian elimination with partial pivoting and row interchanges to factorize the matrix A as

$$A = PLU,$$

where P is a permutation matrix, L is unit lower triangular with at most one nonzero subdiagonal element in each column, and U is an upper triangular band matrix, with two superdiagonals. f07cs then utilizes the factorization to compute a solution, $\hat{X}$, to the required equations. Letting $\hat{x}$ denote a column of $\hat{X}$, f07cv computes a *component-wise backward error*, β, the smallest relative perturbation in each element of A and b such that $\hat{x}$ is the exact solution of a perturbed system

$$(A + E)\hat{x} = b + f, \quad \text{with} \quad |e_{ij}| \leq \beta|a_{ij}|, \quad \text{and} \quad |f_j| \leq \beta|b_j|.$$

The function also estimates a bound for the *component-wise forward error* in the computed solution defined by $\max|x_i - \hat{x}_i|/\max|\hat{x}_i|$, where x is the corresponding column of the exact solution, X.

4 References

Anderson E, Bai Z, Bischof C, Blackford S, Demmel J, Dongarra J J, Du Croz J J, Greenbaum A, Hammarling S, McKenney A and Sorensen D 1999 *LAPACK Users' Guide* (3rd Edition) SIAM, Philadelphia URL: http://www.netlib.org/lapack/lug

5 Parameters

5.1 Compulsory Input Parameters

1: **trans – string**

Specifies the equations to be solved as follows:

trans = 'N'

Solve $AX = B$ for X.

trans = 'T'

Solve $A^{\mathrm{T}}X = B$ for X.

trans = 'C'

Solve $A^{\mathrm{H}}X = B$ for X.

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

2: **dl**$(*)$ **– complex array**

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

Must contain the $(n-1)$ subdiagonal elements of the matrix A.

3: **d**$(*)$ **– complex array**

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

Must contain the n diagonal elements of the matrix A.

4: **du**$(*)$ **– complex array**

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

Must contain the $(n-1)$ superdiagonal elements of the matrix A.

5: **dlf**$(*)$ **– complex array**

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

Must contain the $(n-1)$ multipliers that define the matrix L of the LU factorization of A.

6: **df**$(*)$ **– complex array**

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

Must contain the n diagonal elements of the upper triangular matrix U from the LU factorization of A.

7: **duf**$(*)$ **– complex array**

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

Must contain the $(n-1)$ elements of the first superdiagonal of U.

8: **du2**$(*)$ **– complex array**

Note: the dimension of the array **du2** must be at least $\max(1, \mathbf{n} - 2)$.

Must contain the $(n-2)$ elements of the second superdiagonal of U.

9: **ipiv**$(*)$ **– int32 array**

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

Must contain the n pivot indices that define the permutation matrix P. At the ith step, row i of the matrix was interchanged with row **ipiv**(i), and **ipiv**(i) must always be either i or $(i+1)$, **ipiv**$(i) = i$ indicating that a row interchange was not performed.

10: **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 matrix of right-hand sides B.

11: **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 initial solution matrix X.

5.2 Optional Input Parameters

1: **n – int32 scalar**

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

r, the number of right-hand sides, i.e., the number of columns of the matrix B.

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

5.3 Input Parameters Omitted from the MATLAB Interface

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 n by r refined solution matrix X.

2: **ferr($*$) – double array**

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

Estimate of the forward error bound for each computed solution vector, such that $\|\hat{x}_j - x_j\|_\infty / \|x_j\|_\infty \leq \mathbf{ferr}(j)$, where $\hat{x}_j$ is the jth column of the computed solution returned in the array **x** and x_j is the corresponding column of the exact solution X. The estimate is almost always a slight overestimate of the true error.

3: **berr($*$) – double array**

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

Estimate of the component-wise relative backward error of each computed solution vector $\hat{x}_j$ (i.e., the smallest relative change in any element of A or B that makes $\hat{x}_j$ an exact solution).

4: **info – int32 scalar**

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

6 Error Indicators and Warnings

Errors or warnings detected by the function:

$\mathbf{info} = -i$

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

1: **trans**, 2: **n**, 3: **nrhs_p**, 4: **dl**, 5: **d**, 6: **du**, 7: **dlf**, 8: **df**, 9: **duf**, 10: **du2**, 11: **ipiv**, 12: **b**, 13: **ldb**, 14: **x**, 15: **ldx**, 16: **ferr**, 17: **berr**, 18: **work**, 19: **rwork**, 20: **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 computed solution for a single right-hand side, $\hat{x}$, satisfies an equation of the form

$$(A+E)\hat{x}=b,$$

where

$$\|E\|_\infty = O(\epsilon)\|A\|_\infty$$

and ϵ is the ***machine precision***. An approximate error bound for the computed solution is given by

$$\frac{\|\hat{x}-x\|_\infty}{\|x\|_\infty} \leq \kappa(A)\frac{\|E\|_\infty}{\|A\|_\infty},$$

where $\kappa(A)=\left\|A^{-1}\right\|_\infty\|A\|_\infty$, the condition number of A with respect to the solution of the linear equations. See Section 4.4 of Anderson *et al.* 1999 for further details.

Function f07cu can be used to estimate the condition number of A.

8 Further Comments

The total number of floating-point operations required to solve the equations $AX=B$ or $A^{\mathrm{T}}X=B$ or $A^{\mathrm{H}}X=B$ is proportional to nr. At most five steps of iterative refinement are performed, but usually only one or two steps are required.

The real analogue of this function is f07ch.

9 Example

```
trans = 'No transpose';
dl = [complex(1, -2);
     complex(1, +1);
     complex(2, -3);
     complex(1, +1)];
d = [complex(-1.3, +1.3);
     complex(-1.3, +1.3);
     complex(-1.3, +3.3);
     complex(-0.3, +4.3);
     complex(-3.3, +1.3)];
du = [complex(2, -1);
     complex(2, +1);
     complex(-1, +1);
     complex(1, -1)];
dlf = [complex(-0.78, -0.26);
     complex(0.162, -0.4859999999999999);
     complex(-0.04516923076923077, -0.0009538461538460446);
     complex(-0.3978553846153843, -0.05620307692307711)];
df = [complex(1, -2);
     complex(1, +1);
     complex(2, -3);
     complex(1, +1);
     complex(-1.339863692307691, +0.2875264615384604)];
duf = [complex(-1.3, +1.3);
     complex(-1.3, +3.3);
     complex(-0.3, +4.3);
     complex(-3.3, +1.3)];
du2 = [complex(2, +1);
     complex(-1, +1);
     complex(1, -1)];
```

```
ipiv = [int32(2);
     int32(3);
     int32(4);
     int32(5);
     int32(5)];
b = [complex(2.4, -5), complex(2.7, +6.9);
     complex(3.4, +18.2), complex(-6.9, -5.3);
     complex(-14.7, +9.699999999999999), complex(-6, -0.6);
     complex(31.9, -7.7), complex(-3.9, +9.300000000000001);
     complex(-1, +1.6), complex(-3, +12.2)];
x  =  [complex(1,  +0.9999999999999971),  complex(2.000000000000001,  -
0.9999999999999977);
            complex(3,  -1.000000000000002),  complex(1.000000000000002,
+2.000000000000002);
     complex(4.000000000000001, +5), complex(-1.000000000000001, +1);
     complex(-0.9999999999999996, -2), complex(2, +1);
     complex(1, -0.9999999999999998), complex(2, -2)];
[xOut, ferr, berr, info] = f07cv(trans, dl, d, du, dlf, df, duf, du2,
ipiv, b, x)
```

```
xOut =
   1.0000 + 1.0000i   2.0000 - 1.0000i
   3.0000 - 1.0000i   1.0000 + 2.0000i
   4.0000 + 5.0000i  -1.0000 + 1.0000i
  -1.0000 - 2.0000i   2.0000 + 1.0000i
   1.0000 - 1.0000i   2.0000 - 2.0000i
ferr =
   1.0e-13 *
    0.5638
    0.7460
berr =
   1.0e-16 *
    0.7485
    0.6729
info =
           0
```