NAG Toolbox for MATLAB

f07ap

1 Purpose

f07ap uses the LU factorization to compute the solution to a complex system of linear equations

$$AX = B \quad \text{or} \quad A^{\mathrm{T}}X = B \quad \text{or} \quad A^{\mathrm{H}}X = B,$$

where A is an n by n matrix and X and B are n by r matrices. Error bounds on the solution and a condition estimate are also provided.

2 Syntax

```
[a, af, ipiv, equed, r, c, b, x, rcond, ferr, berr, rwork, info] =
f07ap(fact, trans, a, af, ipiv, equed, r, c, b, 'n', n, 'nrhs_p',
nrhs_p)
```

3 Description

f07ap performs the following steps:

1. **Equilibration**

   The linear system to be solved may be badly scaled. However, the system can be equilibrated as a first stage by setting **fact** = 'E'. In this case, real scaling factors are computed and these factors then determine whether the system is to be equilibrated. Equilibrated forms of the systems $AX = B$ and $A^{\mathrm{T}}X = B$ are

   $$(D_R A D_C)\left(D_C^{-1}X\right) = D_R B$$

   and

   $$(D_R A D_C)^{\mathrm{T}}\left(D_R^{-1}X\right) = D_C B,$$

   respectively, where D_R and D_C are diagonal matrices, with positive diagonal elements, formed from the computed scaling factors.

   When equilibration is used, A will be overwritten by $D_R A D_C$ and B will be overwritten by $D_R B$ (or $D_C B$ when the solution of $A^{\mathrm{T}}X = B$ or $A^{\mathrm{H}}X = B$ is sought).

2. **Factorization**

   The matrix A, or its scaled form, is copied and factored using the LU decomposition

   $$A = PLU,$$

   where P is a permutation matrix, L is a unit lower triangular matrix, and U is upper triangular.

   This stage can be by-passed when a factored matrix (with scaled matrices and scaling factors) are supplied; for example, as provided by a previous call to f07ap with the same matrix A.

3. **Condition Number Estimation**

   The LU factorization of A determines whether a solution to the linear system exists. If some diagonal element of U is zero, then U is exactly singular, no solution exists and the function returns with a failure. Otherwise the factorized form of A is used to estimate the condition number of the matrix A. If the reciprocal of the condition number is less than ***machine precision*** then a warning code is returned on final exit.

4. **Solution**

   The (equilibrated) system is solved for X ($D_C^{-1}X$ or $D_R^{-1}X$) using the factored form of A ($D_R A D_C$).

5. **Iterative Refinement**

Iterative refinement is applied to improve the computed solution matrix and to calculate error bounds and backward error estimates for the computed solution.

6. **Construct Solution Matrix X**

If equilibration was used, the matrix X is premultiplied by D_C (if **trans** $=$ 'N') or D_R (if **trans** $=$ 'T' or 'C') so that it solves the original system before equilibration.

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

Golub G H and Van Loan C F 1996 *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

Higham N J 2002 *Accuracy and Stability of Numerical Algorithms* (2nd Edition) SIAM, Philadelphia

5 Parameters

5.1 Compulsory Input Parameters

1: **fact – string**

Specifies whether or not the factorized form of the matrix A is supplied on entry, and if not, whether the matrix A should be equilibrated before it is factorized.

fact $=$ 'F'

af and **ipiv** contain the factorized form of A. If **equed** $\neq$ 'N', the matrix A has been equilibrated with scaling factors given by **r** and **c**. **a**, **af** and **ipiv** are not modified.

fact $=$ 'N'

The matrix A will be copied to **af** and factorized.

fact $=$ 'E'

The matrix A will be equilibrated if necessary, then copied to **af** and factorized.

Constraint: **fact** $=$ 'F', 'N' or 'E'.

2: **trans – string**

Specifies the form of the system of equations.

trans $=$ 'N'

$AX = B$ (No transpose).

trans $=$ 'T'

$A^{\mathrm{T}}X = B$ (Transpose).

trans $=$ 'C'

$A^{\mathrm{H}}X = B$ (Conjugate transpose).

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

3: **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 matrix A.

If **fact** $=$ 'F' and **equed** $\neq$ 'N', **a** must have been equilibrated by the scaling factors in **r** and/or **c**.

4: **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 $\max(1, \mathbf{n})$

If **fact** = 'F', **af** contains the factors L and U from the factorization $A = PLU$ as computed by f07ar. If **equed** $\neq$ 'N', **af** is the factorized form of the equilibrated matrix A.

If **fact** = 'N' or 'E', **af** need not be set.

5: **ipiv($*$) – int32 array**

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

If **fact** = 'F', **ipiv** contains the pivot indices from the factorization $A = PLU$ as computed by f07ar; at the ith step row i of the matrix was interchanged with row **ipiv**(i).

If **fact** = 'N' or 'E', **ipiv** need not be set. **ipiv**$(i) = i$ indicates a row interchange was not required.

6: **equed – string**

If **fact** = 'N' or 'E', **equed** need not be set.

If **fact** = 'F', **equed** must specify the form of the equilibration that was performed as follows:

if **equed** = 'N', no equilibration;
if **equed** = 'R', row equilibration, i.e., A has been premultiplied by D_R;
if **equed** = 'C', column equilibration, i.e., A has been postmultiplied by D_C;
if **equed** = 'B', both row and column equilibration, i.e., A has been replaced by $D_R A D_C$.

Constraint: if **fact** = 'F', **equed** = 'N', 'R', 'C' or 'B'.

7: **r($*$) – double array**

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

If **fact** = 'N' or 'E', **r** need not be set.

If **fact** = 'F' and **equed** = 'R' or 'B', **r** must contain the row scale factors for A, D_R; each element of **r** must be positive.

8: **c($*$) – double array**

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

If **fact** = 'N' or 'E', **c** need not be set.

If **fact** = 'F' or **equed** = 'C' or 'B', **c** must contain the column scale factors for A, D_C; each element of **c** must be positive.

9: **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.

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** The dimension of the array **r** The dimension of the array **c**.

n, the number of linear equations, i.e., the order of the matrix A.

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

2: **nrhs_p – int32 scalar**

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

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

Constraint: **nrhs_p** ≥ 0.

5.3 Input Parameters Omitted from the MATLAB Interface

lda, ldaf, ldb, ldx, work

5.4 Output Parameters

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

If **fact** = 'F' or 'N', or if **fact** = 'E' and **equed** = 'N', **a** is not modified.

If **fact** = 'E' or **equed** $\neq$ 'N', A is scaled as follows:

if **equed** = 'R', $A = D_R A$;
if **equed** = 'C', $A = A D_C$;
if **equed** = 'B', $A = D_R A D_C$.

2: **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 $\max(1, \mathbf{n})$

If **fact** = 'N', **af** returns the factors L and U from the factorization $A = PLU$ of the original matrix A.

If **fact** = 'E', **af** returns the factors L and U from the factorization $A = PLU$ of the equilibrated matrix A (see the description of **a** for the form of the equilibrated matrix).

If **fact** = 'F', **af** is unchanged from entry.

3: **ipiv($*$) – int32 array**

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

If **fact** = 'N', **ipiv** contains the pivot indices from the factorization $A = PLU$ of the original matrix A.

If **fact** = 'E', **ipiv** contains the pivot indices from the factorization $A = PLU$ of the equilibrated matrix A.

If **fact** = 'F', **ipiv** is unchanged from entry.

4: **equed – string**

If **fact** = 'F', **equed** is unchanged from entry.

Otherwise, if **info** ≥ 0, **equed** specifies the form of equilibration that was performed as specified above.

5: **r($*$) – double array**

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

If **fact** = 'F', **r** is unchanged from entry.

Otherwise, if **info** ≥ 0 and **equed** = 'R' or 'B', **r** contains the row scale factors for A, D_R, such that A is multiplied on the left by D_R; each element of **r** is positive.

6: **c**$(*)$ **– double array**

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

If **fact** = 'F', **c** is unchanged from entry.

Otherwise, if **info** ≥ 0 and **equed** = 'C' or 'B', **c** contains the row scale factors for A, D_C; each element of **c** is positive.

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 $\max(1, \mathbf{nrhs\_p})$

If **equed** = 'N', **b** is not modified.

If **trans** = 'N' and **equed** = 'R' or 'B', **b** contains $D_R B$.

If **trans** = 'T' or 'C' and **equed** = 'C' or 'B', **b** contains $D_C B$.

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 $\max(1, \mathbf{nrhs\_p})$

If **info** $= 0$ or **info** $\geq N + 1$, the n by r solution matrix X to the original system of equations. Note that the arrays A and B are modified on exit if **equed** $\neq$ 'N', and the solution to the equilibrated system is $D_C^{-1} X$ if **trans** = 'N' and **equed** = 'C' or 'B', or $D_R^{-1} X$ if **trans** = 'T' or 'C' and **equed** = 'R' or 'B'.

9: **rcond – double scalar**

If **info** ≥ 0, an estimate of the reciprocal condition number of the matrix A (after equilibration if that is performed), computed as **rcond** $= 1/\left(\|A\|_1 \|A^{-1}\|_1\right)$.

10: **ferr**$(*)$ **– double array**

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

If **info** $= 0$ or **info** $\geq N + 1$, an estimate of the forward error bound for each computed solution vector, such that $\|\hat{x}_j - x_j\|_\infty / \|x_j\|_\infty \leq$ **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 as reliable as the estimate for **rcond**, and is almost always a slight overestimate of the true error.

11: **berr**$(*)$ **– double array**

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

If **info** $= 0$ or **info** $\geq N + 1$, an 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).

12: **rwork**$(*)$ **– double array**

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

rwork(1) contains the reciprocal pivot growth factor $\|A\|/\|U\|$. The 'max absolute element' norm is used. If **rwork**(1) is much less than 1, then the stability of the LU factorization of the (equilibrated) matrix A could be poor. This also means that the solution **x**, condition estimator **rcond**, and forward error bound **ferr** could be unreliable. If factorization fails with **info** > 0leqN, then **rwork**(1) contains the reciprocal pivot growth factor for the leading **info** columns of A.

13: **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: **fact**, 2: **trans**, 3: **n**, 4: **nrhs_p**, 5: **a**, 6: **lda**, 7: **af**, 8: **ldaf**, 9: **ipiv**, 10: **equed**, 11: **r**, 12: **c**, 13: **b**, 14: **ldb**, 15: **x**, 16: **ldx**, 17: **rcond**, 18: **ferr**, 19: **berr**, 20: **work**, 21: **rwork**, 22: **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.

info > 0 and **info** $\leq N$

If **info** $= i$, u_{ii} is exactly zero. The factorization has been completed, but the factor U is exactly singular, so the solution and error bounds could not be computed. **rcond** $= 0$ is returned.

info $= N + 1$

U is nonsingular, but **rcond** is less than ***machine precision***, meaning that the matrix is singular to working precision. Nevertheless, the solution and error bounds are computed because there are a number of situations where the computed solution can be more accurate than the value of **rcond** would suggest.

7 Accuracy

For each right-hand side vector b, the computed solution $\hat{x}$ is the exact solution of a perturbed system of equations $(A + E)\hat{x} = b$, where

$$|E| \leq c(n)\epsilon P|L||U|,$$

$c(n)$ is a modest linear function of n, and ϵ is the ***machine precision***. See Section 9.3 of Higham 2002 for further details.

If x is the true solution, then the computed solution $\hat{x}$ satisfies a forward error bound of the form

$$\frac{\|x - \hat{x}\|_\infty}{\|\hat{x}\|_\infty} \leq w_c \operatorname{cond}(A, \hat{x}, b)$$

where $\operatorname{cond}(A, \hat{x}, b) = \left\| |A^{-1}|(|A||\hat{x}| + |b|) \right\|_\infty / \|\hat{x}\|_\infty \leq \operatorname{cond}(A) = \left\| |A^{-1}||A| \right\|_\infty \leq \kappa_\infty(A)$. If $\hat{x}$ is the jth column of X, then w_c is returned in **berr**(j) and a bound on $\|x - \hat{x}\|_\infty / \|\hat{x}\|_\infty$ is returned in **ferr**(j). See Section 4.4 of Anderson *et al.* 1999 for further details.

8 Further Comments

The factorization of A requires approximately $\frac{8}{3}n^3$ floating-point operations.

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 $8n^2$ operations.

In practice the condition number estimator is very reliable, but it can underestimate the true condition number; see Section 15.3 of Higham 2002 for further details.

The real analogue of this function is f07ab.

9 Example

```
fact = 'Equilibrate';
trans = 'No transpose';
a = [complex(-1.34, +2.55), complex(0.28, +3.17), complex(-6.39, -2.2),
complex(0.72, -0.92);
       complex(-1.7, -14.1), complex(33.1, -1.5), complex(-1.5, +13.4),
complex(12.9, +13.8);
     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(zeros(4, 4));
ipiv = [int32(8185080);
     int32(0);
     int32(0);
     int32(0)];
equed = ' ';
r = [0;
     0;
     0;
     0];
c = [0;
     0;
     0;
     0];
b = [complex(26.26, +51.78), complex(31.32, -6.7);
     complex(64.3, -86.8), complex(158.6, -14.2);
     complex(-5.75, +25.31), complex(-2.15, +30.19);
     complex(1.16, +2.57), complex(-2.56, +7.55)];
[aOut, afOut, ipivOut, equedOut, rOut, cOut, bOut, x, rcond, ferr, berr,
rwork, info] = ...
    f07ap(fact, trans, a, af, ipiv, equed, r, c, b)
```

```
aOut =
   -0.1560 + 0.2969i    0.0326 + 0.3690i   -0.7439 - 0.2561i    0.0838 -
0.1071i
   -0.0491 - 0.4075i    0.9566 - 0.0434i   -0.0434 + 0.3873i    0.3728 +
0.3988i
   -0.5197 - 0.3776i   -0.3017 + 0.6983i   -0.0221 - 0.2133i    0.2717 +
0.2133i
    0.8607 + 0.1393i   -0.2000 + 0.5250i   -0.2964 - 0.2464i   -0.7000 +
0.2393i
afOut =
    0.8607 + 0.1393i   -0.2000 + 0.5250i   -0.2964 - 0.2464i   -0.7000 +
0.2393i
   -0.6576 - 0.3322i   -0.6077 + 0.9771i   -0.1352 - 0.4738i   -0.2681 +
0.1381i
   -0.1222 + 0.3647i    0.2819 - 0.3796i   -0.6520 - 0.0959i    0.1087 +
0.0367i
   -0.1303 - 0.4524i   -0.3664 - 0.4815i   -0.3085 + 0.0724i    0.0449 +
0.0383i
ipivOut =
           4
           3
           4
           4
equedOut =
R
rOut =
    0.1164
    0.0289
    0.1580
    0.3571
cOut =
    1.0000
    1.0000
```

```
    1.0000
    1.0646
bOut =
   3.0570 + 6.0279i   3.6461 - 0.7800i
   1.8584 - 2.5087i   4.5838 - 0.4104i
  -0.9084 + 3.9984i  -0.3397 + 4.7694i
   0.4143 + 0.9179i  -0.9143 + 2.6964i
x =
   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
rcond =
    0.0104
ferr =
   1.0e-13 *
    0.5893
    0.7668
berr =
   1.0e-15 *
    0.1003
    0.0559
rwork =
    0.8323
    0.0000
    0.0000
    0.0000
         0
    0.7250
    1.1519
    1.4909
info =
           0
```