NAG Toolbox

nag_lapack_dgesvx (f07ab)

1 Purpose

nag_lapack_dgesvx (f07ab) uses the LU factorization to compute the solution to a real system of linear equations

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

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, work, info] =
nag_lapack_dgesvx(fact, trans, a, af, ipiv, equed, r, c, b, 'n', n, 'nrhs_p',
nrhs_p)
```

```
[a, af, ipiv, equed, r, c, b, x, rcond, ferr, berr, work, info] = f07ab(fact,
trans, a, af, ipiv, equed, r, c, b, 'n', n, 'nrhs_p', nrhs_p)
```

3 Description

nag_lapack_dgesvx (f07ab) 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 $A^{\mathrm{T}}X = B$ 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 nag_lapack_dgesvx (f07ab) 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_RAD_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 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** – CHARACTER(1)

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** – CHARACTER(1)

Specifies the form of the system of equations.

trans = 'N'
: $AX = B$ (No transpose).

trans = 'T' or 'C'
: $A^{\mathrm{T}}X = B$ (Transpose).

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

3: **a**(lda, :) – REAL (KIND=nag_wp) array

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

The second dimension of the array **a** 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**$(ldaf,:)$ – REAL (KIND=nag_wp) array

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

The second dimension of the array **af** 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 nag_lapack_dgetrf (f07ad). 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**$(:)$ – INTEGER array

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 nag_lapack_dgetrf (f07ad); at the ith step row i of the matrix was interchanged with row **ipiv**(i). **ipiv**$(i) = i$ indicates a row interchange was not required.

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

6: **equed** – CHARACTER(1)

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_RAD_C.

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

7: **r**$(:)$ – REAL (KIND=nag_wp) array

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**$(:)$ – REAL (KIND=nag_wp) array

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**$(ldb,:)$ – REAL (KIND=nag_wp) array

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

The second dimension of the array **b** 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** – INTEGER

Default: the first dimension of the arrays **a**, **af**, **b** and the second dimension of the arrays **a**, **af**, **ipiv**, **r**, **c**.

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

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

2: **nrhs_p** – INTEGER

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

5.3 Output Parameters

1: $\mathbf{a}(lda,:)$ – REAL (KIND=nag_wp) array

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

The second dimension of the array **a** will be $\max(1,\mathbf{n})$.

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', $A = AD_C$;

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

2: $\mathbf{af}(ldaf,:)$ – REAL (KIND=nag_wp) array

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

The second dimension of the array **af** will be $\max(1,\mathbf{n})$.

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**(:) – INTEGER array

The dimension of the array **ipiv** will be $\max(1,\mathbf{n})$

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** – CHARACTER(1)

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

Otherwise, if no constraints are violated, **equed** specifies the form of equilibration that was performed as specified above.

5: **r**(:) – REAL (KIND=nag_wp) array

The dimension of the array **r** will be $\max(1, \mathbf{n})$

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

Otherwise, if no constraints are violated 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**(:) – REAL (KIND=nag_wp) array

The dimension of the array **c** will be $\max(1, \mathbf{n})$

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

Otherwise, if no constraints are violated and **equed** = 'C' or 'B', **c** contains the row scale factors for A, D_C; each element of **c** is positive.

7: **b**(ldb, :) – REAL (KIND=nag_wp) array

The first dimension of the array **b** will be $\max(1, \mathbf{n})$.

The second dimension of the array **b** will be $\max(1, \mathbf{nrhs\_p})$.

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

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

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

8: **x**(ldx, :) – REAL (KIND=nag_wp) array

The first dimension of the array **x** will be $\max(1, \mathbf{n})$.

The second dimension of the array **x** will be $\max(1, \mathbf{nrhs\_p})$.

If **info** = 0 or $\mathbf{n} + 1$, 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** – REAL (KIND=nag_wp)

If no constraints are violated, an estimate of the reciprocal condition number of the matrix A (after equilibration if that is performed), computed as $\mathbf{rcond} = 1.0 / \left(\|A\|_1 \|A^{-1}\|_1 \right)$.

10: **ferr**(**nrhs_p**) – REAL (KIND=nag_wp) array

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

11: **berr**(**nrhs_p**) – REAL (KIND=nag_wp) array

If **info** = 0 or $\mathbf{n} + 1$, 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: **work**($\mathbf{max}(\mathbf{1}, \mathbf{4} \times \mathbf{n})$) – REAL (KIND=nag_wp) array

work(1) contains the reciprocal pivot growth factor $\|A\| / \|U\|$. The 'max absolute element' norm is used. If **work**(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 estimate **rcond**, and forward error bound **ferr** could be unreliable. If the factorization fails with

info > 0 and **info** $\leq$ **n**, then **work**(1) contains the reciprocal pivot growth factor for the leading **info** columns of A.

13: **info** – INTEGER

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

6 Error Indicators and Warnings

info < 0

If **info** $= -i$, argument i had an illegal value. An explanatory message is output, and execution of the program is terminated.

info > 0 and **info** $\leq$ **n** (*warning*)

Element $\langle value \rangle$ of the diagonal 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** $= 0.0$ is returned.

info $=$ **n** $+ 1$ (*warning*)

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 $\frac{2}{3}n^3$ 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 $2n^2$ 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 complex analogue of this function is nag_lapack_zgesvx (f07ap).

9 Example

This example solves the equations

$$AX = B,$$

where A is the general matrix

$$A = \begin{pmatrix} 1.80 & 2.88 & 2.05 & -0.89 \\ 525.00 & -295.00 & -95.00 & -380.00 \\ 1.58 & -2.69 & -2.90 & -1.04 \\ -1.11 & -0.66 & -0.59 & -0.80 \end{pmatrix}$$

and

$$B = \begin{pmatrix} 9.52 & 18.47 \\ 2435.00 & 225.00 \\ 0.77 & -13.28 \\ -6.22 & -6.21 \end{pmatrix}.$$

Error estimates for the solutions, information on scaling, an estimate of the reciprocal of the condition number of the scaled matrix A and an estimate of the reciprocal of the pivot growth factor for the factorization of A are also output.

9.1 Program Text

```
     function f07ab_example

fprintf('f07ab example results\n\n');

% Linear Problem
n    = 4;
nrhs = 2;
a = [  1.80,    2.88,   2.05,   -0.89;
     525.00, -295.00, -95.00, -380.00;
       1.58,   -2.69,  -2.90,   -1.04;
      -1.11,   -0.66,  -0.59,    0.80];
b = [   9.52,  18.47;
     2435.00, 225.00;
        0.77, -13.28;
       -6.22,  -6.21];

% Input parameter initialization
fact  = 'Equilibration';
trans = 'No transpose';
equed = 'N';
af   = zeros(n, n);
ipiv = zeros(n, 1, nag_int_name);
r    = zeros(n, 1);
c    = zeros(n, 1);

% Solve
[a, af, ipiv, equed, r, c, b, x, rcond, ferr, berr, work, info] = ...
  f07ab(...
        fact, trans, a, af, ipiv, equed, r, c, b);

fprintf('Solution is x:\n');
disp(x);
fprintf('\nApproximate condition number = %8.3f\n',1/rcond);
fprintf('Approximate forward  errors  :\n');
fprintf('                              %11.3e\n',ferr);
fprintf('Approximate backward errors  :\n')
fprintf('                              %11.3e\n',berr);
```

9.2 Program Results

```
     f07ab example results

Solution is x:
    1.0000    3.0000
   -1.0000    2.0000
    3.0000    4.0000
   -5.0000    1.0000

Approximate condition number =   54.967
Approximate forward  errors  :
                                 2.384e-14
                                 3.301e-14
Approximate backward errors  :
                                 6.800e-17
                                 8.040e-17
```