NAG Toolbox

nag_lapack_zgtsvx (f07cp)

1 Purpose

nag_lapack_zgtsvx (f07cp) uses the LU factorization to compute the solution to a complex system of linear equations

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

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

2 Syntax

```
[dlf, df, duf, du2, ipiv, x, rcond, ferr, berr, info] = nag_lapack_zgtsvx(fact, trans, dl, d, du, dlf, df, duf, du2, ipiv, b, 'n', n, 'nrhs_p', nrhs_p)
[dlf, df, duf, du2, ipiv, x, rcond, ferr, berr, info] = f07cp(fact, trans, dl, d, du, dlf, df, duf, du2, ipiv, b, 'n', n, 'nrhs_p', nrhs_p)
```

3 Description

nag_lapack_zgtsvx (f07cp) performs the following steps:

1. If **fact** = 'N', the LU decomposition is used to factor the matrix A as $A = LU$, where L is a product of permutation and unit lower bidiagonal matrices and U is upper triangular with nonzeros in only the main diagonal and first two superdiagonals.
2. If some $u_{ii} = 0$, so that U is exactly singular, then the function returns with **info** $= i$. Otherwise, the factored 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***, **info** $\geq$ **n** $+ 1$ is returned as a warning, but the function still goes on to solve for X and compute error bounds as described below.
3. The system of equations is solved for X using the factored form of A.
4. Iterative refinement is applied to improve the computed solution matrix and to calculate error bounds and backward error estimates for it.

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 has been supplied.

fact = 'F'
dlf, **df**, **duf**, **du2** and **ipiv** contain the factorized form of the matrix A. **dlf**, **df**, **duf**, **du2** and **ipiv** will not be modified.

fact = 'N'
The matrix A will be copied to **dlf**, **df** and **duf** and factorized.

Constraint: **fact** = 'F' or 'N'.

2: **trans** – CHARACTER(1)

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: **dl**(:) – COMPLEX (KIND=nag_wp) array

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

The $(n - 1)$ subdiagonal elements of A.

4: **d**(:) – COMPLEX (KIND=nag_wp) array

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

The n diagonal elements of A.

5: **du**(:) – COMPLEX (KIND=nag_wp) array

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

The $(n - 1)$ superdiagonal elements of A.

6: **dlf**(:) – COMPLEX (KIND=nag_wp) array

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

If **fact** = 'F', **dlf** contains the $(n - 1)$ multipliers that define the matrix L from the LU factorization of A.

7: **df**(:) – COMPLEX (KIND=nag_wp) array

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

If **fact** = 'F', **df** contains the n diagonal elements of the upper triangular matrix U from the LU factorization of A.

8: **duf**(:) – COMPLEX (KIND=nag_wp) array

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

If **fact** = 'F', **duf** contains the $(n - 1)$ elements of the first superdiagonal of U.

9: **du2**(:) – COMPLEX (KIND=nag_wp) array

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

If **fact** = 'F', **du2** contains the $(n - 2)$ elements of the second superdiagonal of U.

10: **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 LU factorization of A.

11: **b**(ldb, :) – COMPLEX (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 array **b** and the dimension of the arrays **d**, **df**, **ipiv**.

n, 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: **dlf**(:) – COMPLEX (KIND=nag_wp) array

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

If **fact** = 'N', **dlf** contains the $(n - 1)$ multipliers that define the matrix L from the LU factorization of A.

2: **df**(:) – COMPLEX (KIND=nag_wp) array

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

If **fact** = 'N', **df** contains the n diagonal elements of the upper triangular matrix U from the LU factorization of A.

3: **duf**(:) – COMPLEX (KIND=nag_wp) array

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

If **fact** = 'N', **duf** contains the $(n - 1)$ elements of the first superdiagonal of U.

4: **du2**(:) – COMPLEX (KIND=nag_wp) array

The dimension of the array **du2** will be $\max(1, \mathbf{n} - 2)$

If **fact** = 'N', **du2** contains the $(n - 2)$ elements of the second superdiagonal of U.

5: **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 LU factorization of A; row i of the matrix was interchanged with row **ipiv**(i). **ipiv**(i) will always be either i or $i+1$; **ipiv**$(i) = i$ indicates a row interchange was not required.

6: **x**$(ldx, :)$ – COMPLEX (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.

7: **rcond** – REAL (KIND=nag_wp)

The estimate of the reciprocal condition number of the matrix A. If **rcond** $= 0.0$, the matrix may be exactly singular. This condition is indicated by **info** > 0 and **info** $\leq$ **n**. Otherwise, if **rcond** is less than the ***machine precision***, the matrix is singular to working precision. This condition is indicated by **info** $\geq \mathbf{n}+1$.

8: **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 $\left\|\hat{x}_j - x_j\right\|_\infty / \left\|x_j\right\|_\infty \leq$ **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.

9: **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).

10: **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** $<$ **n** (*warning*)

Element $\langle value \rangle$ of the diagonal is exactly zero. The factorization has not 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 > 0 and **info** $=$ **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 $= \mathbf{n} + 1$ (*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

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

$c(n)$ 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

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

where $\mathrm{cond}(A, \hat{x}, b) = \left\| |A^{-1}|(|A||\hat{x}| + |b|) \right\|_\infty / \|\hat{x}\|_\infty \leq \mathrm{cond}(A) = \left\| |A^{-1}||A| \right\|_\infty \leq \kappa_\infty(A)$. 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 total number of floating-point operations required to solve the equations $AX = B$ is proportional to nr.

The condition number estimation typically requires between four and five solves and never more than eleven solves, following the factorization. The solution is then refined, and the errors estimated, using iterative refinement.

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 nag_lapack_dgtsvx (f07cb).

9 Example

This example solves the equations

$$AX = B,$$

where A is the tridiagonal matrix

$$A = \begin{pmatrix} -1.3 + 1.3i & 2.0 - 1.0i & 0 & 0 & 0 \\ 1.0 - 2.0i & -1.3 + 1.3i & 2.0 + 1.0i & 0 & 0 \\ 0 & 1.0 + 1.0i & -1.3 + 3.3i & -1.0 + 1.0i & 0 \\ 0 & 0 & 2.0 - 3.0i & -0.3 + 4.3i & 1.0 - 1.0i \\ 0 & 0 & 0 & 1.0 + 1.0i & -3.3 + 1.3i \end{pmatrix}$$

and

$$B = \begin{pmatrix} 2.4 - 5.0i & 2.7 + 6.9i \\ 3.4 + 18.2i & -6.9 - 5.3i \\ -14.7 + 9.7i & -6.0 - 0.6i \\ 31.9 - 7.7i & -3.9 + 9.3i \\ -1.0 + 1.6i & -3.0 + 12.2i \end{pmatrix}.$$

Estimates for the backward errors, forward errors and condition number are also output.

9.1 Program Text

```
function f07cp_example

fprintf('f07cp example results\n\n');

% Tridiagonal matrix stored by diagonals
du = [              2   - 1i     2   + 1i    -1   + 1i     1   - 1i  ];
d  = [-1.3 + 1.3i  -1.3 + 1.3i  -1.3 + 3.3i  -0.3 + 4.3i  -3.3 + 1.3i];
dl = [ 1   - 2i     1   + 1i     2   - 3i     1   + 1i               ];
n = numel(d);

% Rhs B
b = [  2.4 -  5.0i   2.7 +  6.9i;
       3.4 + 18.2i  -6.9 -  5.3i;
     -14.7 +  9.7i  -6.0 -  0.6i;
      31.9 -  7.7i  -3.9 +  9.3i;
      -1   +  1.6i  -3.0 + 12.2i];

% Input parameters
fact = 'No factors';
trans = 'No transpose';
dlf = dl;
df  = d;
duf = du;
du2 = complex(zeros(n-2,1));
ipiv = zeros(n,1,nag_int_name);

% Solve
[dlf, df, duf, du2, ipiv, x, rcond, ferr, berr, info] = ...
  f07cp( ...
         fact, trans, dl, d, du, dlf, df, duf, du2, ipiv, b);

disp('Solution(s)');
disp(x);
disp('Backward errors (machine-dependent)');
fprintf('%10.1e',berr);
fprintf('\n');
disp('Estimated forward error bounds (machine-dependent)');
fprintf('%10.1e',ferr);
fprintf('\n\n');
disp('Estimate of reciprocal condition number');
fprintf('%10.1e\n',rcond);
```

9.2 Program Results

```
f07cp example results

Solution(s)
   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

Backward errors (machine-dependent)
   3.6e-17   1.0e-16
Estimated forward error bounds (machine-dependent)
   5.5e-14   7.7e-14

Estimate of reciprocal condition number
   5.4e-03
```