NAG Library Function Document

nag_zgtsv (f07cnc)

1 Purpose

nag_zgtsv (f07cnc) computes the solution to a complex system of linear equations

$$AX = B,$$

where A is an n by n tridiagonal matrix and X and B are n by r matrices.

2 Specification

```
#include <nag.h>
#include <nagf07.h>
void nag_zgtsv (Nag_OrderType order, Integer n, Integer nrhs, Complex dl[],
     Complex d[], Complex du[], Complex b[], Integer pdb, NagError *fail)
```

3 Description

nag_zgtsv (f07cnc) uses Gaussian elimination with partial pivoting and row interchanges to solve the equations $AX = B$. The matrix A is factorized as $A = PLU$, where P is a permutation matrix, L is unit lower triangular with at most one nonzero subdiagonal element per column, and U is an upper triangular band matrix, with two superdiagonals.

Note that the equations $A^{\mathrm{T}}X = B$ may be solved by interchanging the order of the arguments **du** and **dl**.

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

5 Arguments

1: **order** – Nag_OrderType *Input*

On entry: the **order** argument specifies the two-dimensional storage scheme being used, i.e., row-major ordering or column-major ordering. C language defined storage is specified by **order** = Nag_RowMajor. See Section 2.3.1.3 in How to Use the NAG Library and its Documentation for a more detailed explanation of the use of this argument.

Constraint: **order** = Nag_RowMajor or Nag_ColMajor.

2: **n** – Integer *Input*

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

Constraint: **n** ≥ 0.

3: **nrhs** – Integer *Input*

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

Constraint: **nrhs** ≥ 0.

4: **dl**[dim] – Complex *Input/Output*

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

On entry: must contain the $(n - 1)$ subdiagonal elements of the matrix A.

On exit: if no constraints are violated, **dl** is overwritten by the $(n - 2)$ elements of the second superdiagonal of the upper triangular matrix U from the LU factorization of A, in $\mathbf{dl}[0], \mathbf{dl}[1], \ldots, \mathbf{dl}[n - 3]$.

5: **d**[dim] – Complex *Input/Output*

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

On entry: must contain the n diagonal elements of the matrix A.

On exit: if no constraints are violated, **d** is overwritten by the n diagonal elements of the upper triangular matrix U from the LU factorization of A.

6: **du**[dim] – Complex *Input/Output*

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

On entry: must contain the $(n - 1)$ superdiagonal elements of the matrix A.

On exit: if no constraints are violated, **du** is overwritten by the $(n - 1)$ elements of the first superdiagonal of U.

7: **b**[dim] – Complex *Input/Output*

Note: the dimension, *dim*, of the array **b** must be at least

$\max(1, \mathbf{pdb} \times \mathbf{nrhs})$ when **order** = Nag_ColMajor;
$\max(1, \mathbf{n} \times \mathbf{pdb})$ when **order** = Nag_RowMajor.

The (i, j)th element of the matrix B is stored in

$\mathbf{b}[(j - 1) \times \mathbf{pdb} + i - 1]$ when **order** = Nag_ColMajor;
$\mathbf{b}[(i - 1) \times \mathbf{pdb} + j - 1]$ when **order** = Nag_RowMajor.

On entry: the n by r right-hand side matrix B.

On exit: if **fail.code** = NE_NOERROR, the n by r solution matrix X.

8: **pdb** – Integer *Input*

On entry: the stride separating row or column elements (depending on the value of **order**) in the array **b**.

Constraints:

if **order** = Nag_ColMajor, **pdb** $\geq \max(1, \mathbf{n})$;
if **order** = Nag_RowMajor, **pdb** $\geq \max(1, \mathbf{nrhs})$.

9: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.
See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_INT

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 0.

On entry, **nrhs** = ⟨*value*⟩.
Constraint: **nrhs** ≥ 0.

On entry, **pdb** = ⟨*value*⟩.
Constraint: **pdb** > 0.

NE_INT_2

On entry, **pdb** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
Constraint: **pdb** ≥ max(1, **n**).

On entry, **pdb** = ⟨*value*⟩ and **nrhs** = ⟨*value*⟩.
Constraint: **pdb** ≥ max(1, **nrhs**).

NE_INTERNAL_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please contact NAG for assistance.

An unexpected error has been triggered by this function. Please contact NAG.
See Section 2.7.6 in How to Use the NAG Library and its Documentation for further information.

NE_NO_LICENCE

Your licence key may have expired or may not have been installed correctly.
See Section 2.7.5 in How to Use the NAG Library and its Documentation for further information.

NE_SINGULAR

Element ⟨*value*⟩ of the diagonal is exactly zero, and the solution has not been computed. The factorization has not been completed unless **n** = ⟨*value*⟩.

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

$$\|E\|_1 = O(\epsilon)\|A\|_1$$

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

$$\frac{\|\hat{x} - x\|_1}{\|x\|_1} \leq \kappa(A)\frac{\|E\|_1}{\|A\|_1},$$

where $\kappa(A) = \left\|A^{-1}\right\|_1 \|A\|_1$, 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.

Alternatives to nag_zgtsv (f07cnc), which return condition and error estimates are nag_complex_tridiag_lin_solve (f04ccc) and nag_zgtsvx (f07cpc).

8 Parallelism and Performance

nag_zgtsv (f07cnc) is not threaded in any implementation.

9 Further Comments

The total number of floating-point operations required to solve the equations $AX = B$ is proportional to nr.

The real analogue of this function is nag_dgtsv (f07cac).

10 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

$$b = \begin{pmatrix} 2.4 - 5.0i \\ 3.4 + 18.2i \\ -14.7 + 9.7i \\ 31.9 - 7.7i \\ -1.0 + 1.6i \end{pmatrix}.$$

10.1 Program Text

```
/* nag_zgtsv (f07cnc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf07.h>

int main(void)
{

  /* Scalars */
  Integer exit_status = 0, i, j, n, nrhs, pdb;

  /* Arrays */
  Complex *b = 0, *d = 0, *dl = 0, *du = 0;

  /* Nag Types */
  NagError fail;
  Nag_OrderType order;

#ifdef NAG_COLUMN_MAJOR
#define B(I, J) b[(J-1)*pdb + I - 1]
  order = Nag_ColMajor;
#else
#define B(I, J) b[(I-1)*pdb + J - 1]
  order = Nag_RowMajor;
#endif

  INIT_FAIL(fail);

  printf("nag_zgtsv (f07cnc) Example Program Results\n\n");
```

```
  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif

#ifdef _WIN32
  scanf_s("%" NAG_IFMT "%" NAG_IFMT "%*[^\n]", &n, &nrhs);
#else
  scanf("%" NAG_IFMT "%" NAG_IFMT "%*[^\n]", &n, &nrhs);
#endif
  if (n < 0 || nrhs < 0) {
    printf("Invalid n or nrhs\n");
    exit_status = 1;
    goto END;
  }
  /* Allocate memory */
  if (!(b = NAG_ALLOC(n * nrhs, Complex)) ||
      !(d = NAG_ALLOC(n, Complex)) ||
      !(dl = NAG_ALLOC(n - 1, Complex)) || !(du = NAG_ALLOC(n - 1, Complex)))
  {
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
  }

#ifdef NAG_COLUMN_MAJOR
  pdb = n;
#else
  pdb = nrhs;
#endif

  /* Read the tridiagonal matrix A and the right hand side B from data file */
#ifdef _WIN32
  for (i = 0; i < n - 1; ++i)
    scanf_s(" ( %lf , %lf )", &du[i].re, &du[i].im);
#else
  for (i = 0; i < n - 1; ++i)
    scanf(" ( %lf , %lf )", &du[i].re, &du[i].im);
#endif
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif
#ifdef _WIN32
  for (i = 0; i < n; ++i)
    scanf_s(" ( %lf , %lf )", &d[i].re, &d[i].im);
#else
  for (i = 0; i < n; ++i)
    scanf(" ( %lf , %lf )", &d[i].re, &d[i].im);
#endif
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif
#ifdef _WIN32
  for (i = 0; i < n - 1; ++i)
    scanf_s(" ( %lf , %lf )", &dl[i].re, &dl[i].im);
#else
  for (i = 0; i < n - 1; ++i)
    scanf(" ( %lf , %lf )", &dl[i].re, &dl[i].im);
#endif
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif
```

```
  for (i = 1; i <= n; ++i)
    for (j = 1; j <= nrhs; ++j)
#ifdef _WIN32
      scanf_s(" ( %lf , %lf )", &B(i, j).re, &B(i, j).im);
#else
      scanf(" ( %lf , %lf )", &B(i, j).re, &B(i, j).im);
#endif
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif

  /* Solve the equations Ax = b for x using nag_zgtsv (f07cnc). */
  nag_zgtsv(order, n, nrhs, dl, d, du, b, pdb, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_zgtsv (f07cnc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }
  /* Print solution */
  printf("Solution\n");
  for (i = 1; i <= n; ++i) {
    for (j = 1; j <= nrhs; ++j)
      printf("(%8.4f, %8.4f)%s", B(i, j).re, B(i, j).im,
             j % 4 == 0 ? "\n" : " ");
    printf("\n");
  }

END:
  NAG_FREE(b);
  NAG_FREE(d);
  NAG_FREE(dl);
  NAG_FREE(du);

  return exit_status;
}
```

10.2 Program Data

```
nag_zgtsv (f07cnc) Example Program Data
    5           1                                                    : n, nrhs
              (  2.0, -1.0) (  2.0,  1.0) ( -1.0,  1.0) (  1.0, -1.0) : du
 ( -1.3,  1.3) ( -1.3,  1.3) ( -1.3,  3.3) ( -0.3,  4.3) ( -3.3,  1.3) : d
 (  1.0, -2.0) (  1.0 , 1.0) (  2.0, -3.0) (  1.0,  1.0)               : dl
 (  2.4, -5.0) (  3.4, 18.2) (-14.7,  9.7) ( 31.9, -7.7) ( -1.0,  1.6) : B
```

10.3 Program Results

```
nag_zgtsv (f07cnc) Example Program Results

Solution
(  1.0000,   1.0000)
(  3.0000,  -1.0000)
(  4.0000,   5.0000)
( -1.0000,  -2.0000)
(  1.0000,  -1.0000)
```