# NAG Library Function Document

# nag_zgttrs (f07csc)

## 1 Purpose

nag_zgttrs (f07csc) computes 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 nag_zgttrf (f07crc).

## 2 Specification

```
#include <nag.h>
#include <nagf07.h>
```

```
void nag_zgttrs (Nag_OrderType order, Nag_TransType trans, Integer n,
     Integer nrhs, const Complex dl[], const Complex d[], const Complex du[],
     const Complex du2[], const Integer ipiv[], Complex b[], Integer pdb,
     NagError *fail)
```

## 3 Description

nag_zgttrs (f07csc) should be preceded by a call to nag_zgttrf (f07crc), which 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. nag_zgttrs (f07csc) then utilizes the factorization to solve the required equations.

## 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

## 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 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

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

2: **trans** – Nag_TransType *Input*

*On entry*: specifies the equations to be solved as follows:

**trans** = Nag_NoTrans
Solve $AX = B$ for $X$.

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

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

*Constraint*: **trans** = Nag_NoTrans, Nag_Trans or Nag_ConjTrans.

3: **n** – Integer *Input*

*On entry*: $n$, the order of the matrix $A$.

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

4: **nrhs** – Integer *Input*

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

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

5: **dl**[$dim$] – const Complex *Input*

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

*On entry*: must contain the $(n - 1)$ multipliers that define the matrix $L$ of the $LU$ factorization of $A$.

6: **d**[$dim$] – const Complex *Input*

**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 upper triangular matrix $U$ from the $LU$ factorization of $A$.

7: **du**[$dim$] – const Complex *Input*

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

*On entry*: must contain the $(n - 1)$ elements of the first superdiagonal of $U$.

8: **du2**[$dim$] – const Complex *Input*

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

*On entry*: must contain the $(n - 2)$ elements of the second superdiagonal of $U$.

9: **ipiv**[$dim$] – const Integer *Input*

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

*On entry*: must contain the $n$ pivot indices that define the permutation matrix $P$. At the $i$th step, row $i$ of the matrix was interchanged with row $\mathbf{ipiv}[i - 1]$, and $\mathbf{ipiv}[i - 1]$ must always be either $i$ or $(i + 1)$, $\mathbf{ipiv}[i - 1] = i$ indicating that a row interchange was not performed.

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

*On exit*: the $n$ by $r$ solution matrix $X$.

11: **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** $\geq \max(1, \mathbf{nrhs})$.

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

The NAG error argument (see Section 3.6 in the Essential Introduction).

# 6 Error Indicators and Warnings

**NE_ALLOC_FAIL**

Dynamic memory allocation failed.

**NE_BAD_PARAM**

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

**NE_INT**

On entry, **n** = ⟨*value*⟩.
Constraint: **n** $\geq 0$.

On entry, **nrhs** = ⟨*value*⟩.
Constraint: **nrhs** $\geq 0$.

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

**NE_INT_2**

On entry, **pdb** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
Constraint: **pdb** $\geq \max(1, \mathbf{n})$.

On entry, **pdb** = ⟨*value*⟩ and **nrhs** = ⟨*value*⟩.
Constraint: **pdb** $\geq \max(1, \mathbf{nrhs})$.

**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.

# 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 $\epsilon$ 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.

Following the use of this function nag_zgtcon (f07cuc) can be used to estimate the condition number of $A$ and nag_zgtrfs (f07cvc) can be used to obtain approximate error bounds.

## 8 Parallelism and Performance

Not applicable.

## 9 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 $nr$.

The real analogue of this function is nag_dgttrs (f07cec).

## 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

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

### 10.1 Program Text

```
/* nag_zgttrs (f07csc) Example Program.
 *
 * Copyright 2008 Numerical Algorithms Group.
 *
 * Mark 23, 2011
 */
#include <stdio.h>
#include <nag.h>
#include <nagx04.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, *du2 = 0;
  Integer       *ipiv = 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_zgttrs (f07csc) Example Program Results\n\n");
  /* Skip heading in data file */
  scanf("%*[^\n]");
  scanf("%ld%ld%*[^\n]", &n, &nrhs);
  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)) ||
      !(du2  = NAG_ALLOC(n-2, Complex)) ||
      !(ipiv = NAG_ALLOC(n, Integer)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }
#ifdef NAG_COLUMN_MAJOR
  pdb = n;
#else
  pdb = nrhs;
#endif

  /* Read the tridiagonal matrix A from data file */
  for (i = 0; i < n - 1; ++i) scanf(" ( %lf , %lf )", &du[i].re, &du[i].im);
  scanf("%*[^\n]");
  for (i = 0; i < n; ++i) scanf(" ( %lf , %lf )", &d[i].re, &d[i].im);
  scanf("%*[^\n]");
  for (i = 0; i < n - 1; ++i) scanf(" ( %lf , %lf )", &dl[i].re, &dl[i].im);
  scanf("%*[^\n]");

  /* Read the right hand matrix B */
  for (i = 1; i <= n; ++i)
    for (j = 1; j <= nrhs; ++j)
      scanf(" ( %lf , %lf )", &B(i, j).re, &B(i, j).im);
  scanf("%*[^\n]");

  /* Factorize the tridiagonal matrix A using nag_zgttrf (f07crc). */
  nag_zgttrf(n, dl, d, du, du2, ipiv, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_zgttrf (f07crc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  /* Solve the equations AX = B using nag_zgttrs (f07csc). */
  nag_zgttrs(order, Nag_NoTrans, n, nrhs, dl, d, du, du2, ipiv, b, pdb, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_zgttrs (f07csc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

  /* Print the solution using nag_gen_complx_mat_print_comp (x04dbc). */
  fflush(stdout);
  nag_gen_complx_mat_print_comp(order, Nag_GeneralMatrix, Nag_NonUnitDiag,
                                n, nrhs, b, pdb, Nag_BracketForm, "%7.4f",
```

```
                                "Solution(s)", Nag_IntegerLabels, 0,
                                Nag_IntegerLabels, 0, 80, 0, 0, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_gen_complx_mat_print_comp (x04dbc).\n%s\n",
             fail.message);
      exit_status = 1;
      goto END;
    }
 END:
  NAG_FREE(b);
  NAG_FREE(d);
  NAG_FREE(dl);
  NAG_FREE(du);
  NAG_FREE(du2);
  NAG_FREE(ipiv);

  return exit_status;
}
```

## 10.2 Program Data

```
nag_zgttrs (f07csc) Example Program Data
    5          2                                                    : 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) (  2.7,  6.9)
 (  3.4, 18.2) ( -6.9, -5.3)
 (-14.7,  9.7) ( -6.0, -0.6)
 ( 31.9, -7.7) ( -3.9,  9.3)
 ( -1.0,  1.6) ( -3.0, 12.2)                                           : B
```

## 10.3 Program Results

```
nag_zgttrs (f07csc) Example Program Results

 Solution(s)
                   1                  2
 1  ( 1.0000, 1.0000)  ( 2.0000,-1.0000)
 2  ( 3.0000,-1.0000)  ( 1.0000, 2.0000)
 3  ( 4.0000, 5.0000)  (-1.0000, 1.0000)
 4  (-1.0000,-2.0000)  ( 2.0000, 1.0000)
 5  ( 1.0000,-1.0000)  ( 2.0000,-2.0000)
```