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 \( A^T X = B \) 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,
Philadelphia http://www.netlib.org/lapack/lug

5 Arguments

1: \hspace{1em} order – Nag_OrderType \hspace{1em} \textit{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
   \textit{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: \textit{order} = Nag_RowMajor or Nag_ColMajor.

2: \hspace{1em} n – Integer \hspace{1em} \textit{Input}
   
   On entry: \( n \), the number of linear equations, i.e., the order of the matrix \( A \).
   
   Constraint: \( n \geq 0 \).

3: \hspace{1em} nrhs – Integer \hspace{1em} \textit{Input}
   
   On entry: \( r \), the number of right-hand sides, i.e., the number of columns of the matrix \( B \).
   
   Constraint: \( nrhs \geq 0 \).
4: \( \text{dl}[\text{dim}] \) – Complex  
**Input/Output**  
*Note:* the dimension, \( \text{dim} \), of the array \( \text{dl} \) must be at least \( \text{max}(1, n - 1) \).  
*On entry:* must contain the \((n - 1)\) subdiagonal elements of the matrix \( A \).  
*On exit:* if no constraints are violated, \( \text{dl} \) is overwritten by the \((n - 2)\) elements of the second superdiagonal of the upper triangular matrix \( U \) from the \( LU \) factorization of \( A \), in \( \text{dl}[0], \text{dl}[1], \ldots, \text{dl}[n - 3] \).

5: \( \text{d}[\text{dim}] \) – Complex  
**Input/Output**  
*Note:* the dimension, \( \text{dim} \), of the array \( \text{d} \) must be at least \( \text{max}(1, n) \).  
*On entry:* must contain the \( n \) diagonal elements of the matrix \( A \).  
*On exit:* if no constraints are violated, \( \text{d} \) is overwritten by the \( n \) diagonal elements of the upper triangular matrix \( U \) from the \( LU \) factorization of \( A \).

6: \( \text{du}[\text{dim}] \) – Complex  
**Input/Output**  
*Note:* the dimension, \( \text{dim} \), of the array \( \text{du} \) must be at least \( \text{max}(1, n - 1) \).  
*On entry:* must contain the \((n - 1)\) superdiagonal elements of the matrix \( A \).  
*On exit:* if no constraints are violated, \( \text{du} \) is overwritten by the \((n - 1)\) elements of the first superdiagonal of \( U \).

7: \( \text{b}[\text{dim}] \) – Complex  
**Input/Output**  
*Note:* the dimension, \( \text{dim} \), of the array \( \text{b} \) must be at least \( \text{max}(1, \text{pdb} \times \text{nrhs}) \) when \( \text{order} = \text{Nag\_ColMajor} \); \( \text{max}(1, n \times \text{pdb}) \) when \( \text{order} = \text{Nag\_RowMajor} \).  
The \((i,j)\)th element of the matrix \( B \) is stored in  
\[
\text{b}[(j - 1) \times \text{pdb} + i - 1] \text{ when } \text{order} = \text{Nag\_ColMajor};  
\text{b}[(i - 1) \times \text{pdb} + j - 1] \text{ when } \text{order} = \text{Nag\_RowMajor}.
\]  
*On entry:* the \( n \) by \( r \) right-hand side matrix \( B \).  
*On exit:* if \( \text{fail\_code} = \text{NE\_NOERROR} \), the \( n \) by \( r \) solution matrix \( X \).

8: \( \text{pdb} \) – Integer  
**Input**  
*On entry:* the stride separating row or column elements (depending on the value of \( \text{order} \)) in the array \( \text{b} \).  
*Constraints:*  
\[
\text{if } \text{order} = \text{Nag\_ColMajor}, \text{pdb} \geq \text{max}(1, n);  
\text{if } \text{order} = \text{Nag\_RowMajor}, \text{pdb} \geq \text{max}(1, \text{nrhs}).
\]

9: \( \text{fail} \) – 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.  
See Section 3.2.1.2 in the Essential Introduction for further information.

**NE\_BAD\_PARAM**

On entry, argument \( \langle \text{value} \rangle \) had an illegal value.
NE_INT
On entry, \( n = \langle\text{value}\rangle\).
Constraint: \( n \geq 0 \).

On entry, \( \text{nrhs} = \langle\text{value}\rangle\).
Constraint: \( \text{nrhs} \geq 0 \).

On entry, \( \text{pdbh} = \langle\text{value}\rangle\).
Constraint: \( \text{pdbh} > 0 \).

NE_INT_2
On entry, \( \text{pdbh} = \langle\text{value}\rangle\) and \( n = \langle\text{value}\rangle\).
Constraint: \( \text{pdbh} \geq \max(1, n) \).

On entry, \( \text{pdbh} = \langle\text{value}\rangle\) and \( \text{nrhs} = \langle\text{value}\rangle\).
Constraint: \( \text{pdbh} \geq \max(1, \text{nrhs}) \).

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 3.6.6 in the Essential Introduction for further information.

NE_NO_LICENCE
Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in the Essential Introduction for further information.

NE_SINGULAR
Element \( \langle\text{value}\rangle\) of the diagonal is exactly zero, and the solution has not been computed. The factorization has not been completed unless \( n = \langle\text{value}\rangle\).

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 \( \kappa(A) = \|A^{-1}\|_1\|A\|_1 \), 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
Not applicable.
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. */
* * Copyright 2014 Numerical Algorithms Group. * * Mark 23, 2011. */
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf07.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
...
```c
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;
}
#endif NAG_COLUMN_MAJOR
pdb = n;
#else
    pdb = nrhs;
#endif
#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
#define _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
#define _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
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
```

Mark 25

f07cnc

f07cnc.5
/* 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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1</td>
<td>n</td>
<td>nrhs</td>
<td></td>
</tr>
<tr>
<td>(2.0, -1.0)</td>
<td>(2.0, 1.0)</td>
<td>(-1.0, 1.0)</td>
<td>(1.0, -1.0)</td>
<td>du</td>
</tr>
<tr>
<td>(-1.3, 1.3)</td>
<td>(-1.3, 1.3)</td>
<td>(-1.3, 3.3)</td>
<td>(-0.3, 4.3)</td>
<td>(-3.3, 1.3)</td>
</tr>
<tr>
<td>(1.0, -2.0)</td>
<td>(1.0, 1.0)</td>
<td>(2.0, -3.0)</td>
<td>(1.0, 1.0)</td>
<td>dl</td>
</tr>
<tr>
<td>(2.4, -5.0)</td>
<td>(3.4, 18.2)</td>
<td>(-14.7, 9.7)</td>
<td>(31.9, -7.7)</td>
<td>(-1.0, 1.6)</td>
</tr>
</tbody>
</table>

10.3 Program Results

nag_zgtsv (f07cnc) Example Program Results

Solution

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(1.0000, 1.0000)</td>
<td></td>
</tr>
<tr>
<td>(3.0000, -1.0000)</td>
<td></td>
</tr>
<tr>
<td>(4.0000, 5.0000)</td>
<td></td>
</tr>
<tr>
<td>(-1.0000, -2.0000)</td>
<td></td>
</tr>
<tr>
<td>(1.0000, -1.0000)</td>
<td></td>
</tr>
</tbody>
</table>