NAG Library Function Document

nag_zgttrf (f07crc)

1 Purpose

nag_zgttrf (f07crc) computes the LU factorization of a complex n by n tridiagonal matrix A.

2 Specification

```
#include <nag.h>
#include <nagf07.h>
void nag_zgttrf (Integer n, Complex dl[], Complex d[], Complex du[],
     Complex du2[], Integer ipiv[], NagError *fail)
```

3 Description

nag_zgttrf (f07crc) 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.

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: **n** – Integer *Input*

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

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

2: **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: is overwritten by the $(n - 1)$ multipliers that define the matrix L of the LU factorization of A.

3: **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: is overwritten by the n diagonal elements of the upper triangular matrix U from the LU factorization of A.

4: **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: is overwritten by the $(n-1)$ elements of the first superdiagonal of U.

5: **du2**[**n** − **2**] – Complex *Output*

On exit: contains the $(n-2)$ elements of the second superdiagonal of U.

6: **ipiv**[**n**] – Integer *Output*

On exit: contains the n pivot indices that define the permutation matrix P. At the ith step, row i of the matrix was interchanged with row **ipiv**$[i-1]$. **ipiv**$[i-1]$ will always be either i or $(i+1)$, **ipiv**$[i-1] = i$ indicating that a row interchange was not performed.

7: **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.

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. The factorization has been completed, but the factor U is exactly singular, and division by zero will occur if it is used to solve a system of equations.

7 Accuracy

The computed factorization satisfies an equation of the form

$$A + E = PLU,$$

where

$$\|E\|_\infty = O(\epsilon)\|A\|_\infty$$

and ϵ is the ***machine precision***.

Following the use of this function, nag_zgttrs (f07csc) can be used to solve systems of equations $AX = B$ or $A^{\mathrm{T}}X = B$ or $A^{\mathrm{H}}X = B$, and nag_zgtcon (f07cuc) can be used to estimate the condition number of A.

8 Parallelism and Performance

nag_zgttrf (f07crc) is not threaded in any implementation.

9 Further Comments

The total number of floating-point operations required to factorize the matrix A is proportional to n.

The real analogue of this function is nag_dgttrf (f07cdc).

10 Example

This example factorizes the tridiagonal matrix A given by

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

10.1 Program Text

```
/* nag_zgttrf (f07crc) 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, n;

  /* Arrays */
  Complex *d = 0, *dl = 0, *du = 0, *du2 = 0;
  Integer *ipiv = 0;

  /* Nag Types */
  NagError fail;

  INIT_FAIL(fail);

  printf("nag_zgttrf (f07crc) 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 "%*[^\n]", &n);
#else
  scanf("%" NAG_IFMT "%*[^\n]", &n);
#endif
  if (n < 0) {
    printf("Invalid n\n");
    exit_status = 1;
    goto END;
  }

  /* Allocate memory */
  if (!(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;
  }

  /* Read the tridiagonal matrix A 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

  /* 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;
  }
```

```
  /* Print details of the factorization */
  printf("Details of factorization (U)\n\n");

  printf("%17s%24s%22s\n", "Main diagonal", "First superdiagonal",
         "Second superdiagonal");
  for (i = 0; i < n; ++i) {
    printf("(%8.4f, %8.4f)", d[i].re, d[i].im);
    if (i < n - 1)
      printf(" (%8.4f, %8.4f)", du[i].re, du[i].im);
    if (i < n - 2)
      printf(" (%8.4f, %8.4f)", du2[i].re, du2[i].im);
    printf("\n");
  }

  printf("\n Multipliers\n");
  for (i = 0; i < n - 1; ++i)
    printf("(%8.4f, %8.4f)\n", dl[i].re, dl[i].im);

  printf("\n Vector of interchanges\n");
  for (i = 0; i < n; ++i)
    printf("%7" NAG_IFMT "%s", ipiv[i], i % 5 == 4 ? "\n" : " ");
  printf("\n");
END:
  NAG_FREE(d);
  NAG_FREE(dl);
  NAG_FREE(du);
  NAG_FREE(du2);
  NAG_FREE(ipiv);

  return exit_status;
}
```

10.2 Program Data

```
nag_zgttrf (f07crc) Example Program Data
   5                                                       : n
            ( 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
```

10.3 Program Results

```
nag_zgttrf (f07crc) Example Program Results

Details of factorization (U)

    Main diagonal     First superdiagonal  Second superdiagonal
(  1.0000,  -2.0000) ( -1.3000,   1.3000) (  2.0000,   1.0000)
(  1.0000,   1.0000) ( -1.3000,   3.3000) ( -1.0000,   1.0000)
(  2.0000,  -3.0000) ( -0.3000,   4.3000) (  1.0000,  -1.0000)
(  1.0000,   1.0000) ( -3.3000,   1.3000)
( -1.3399,   0.2875)

 Multipliers
( -0.7800,  -0.2600)
(  0.1620,  -0.4860)
( -0.0452,  -0.0010)
( -0.3979,  -0.0562)

 Vector of interchanges
      2       3       4       5       5
```