NAG Library Function Document

nag_zgbtrf (f07brc)

1 Purpose

nag_zgbtrf (f07brc) computes the LU factorization of a complex m by n band matrix.

2 Specification

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

```
void nag_zgbtrf (Nag_OrderType order, Integer m, Integer n, Integer kl,
     Integer ku, Complex ab[], Integer pdab, Integer ipiv[], NagError *fail)
```

3 Description

nag_zgbtrf (f07brc) forms the LU factorization of a complex m by n band matrix A using partial pivoting, with row interchanges. Usually $m = n$, and then, if A has k_l nonzero subdiagonals and k_u nonzero superdiagonals, the factorization has the form $A = PLU$, where P is a permutation matrix, L is a lower triangular matrix with unit diagonal elements and at most k_l nonzero elements in each column, and U is an upper triangular band matrix with $k_l + k_u$ superdiagonals.

Note that L is not a band matrix, but the nonzero elements of L can be stored in the same space as the subdiagonal elements of A. U is a band matrix but with k_l additional superdiagonals compared with A. These additional superdiagonals are created by the row interchanges.

4 References

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: **m** – Integer *Input*

On entry: m, the number of rows of the matrix A.

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

3: **n** – Integer *Input*

On entry: n, the number of columns of the matrix A.

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

4: **kl** – Integer *Input*

On entry: k_l, the number of subdiagonals within the band of the matrix A.

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

5: **ku** – Integer *Input*

On entry: k_u, the number of superdiagonals within the band of the matrix A.

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

6: **ab**$[dim]$ – Complex *Input/Output*

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

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

On entry: the m by n matrix A.

This is stored as a notional two-dimensional array with row elements or column elements stored contiguously. The storage of elements A_{ij}, for row $i = 1, \ldots, m$ and column $j = \max(1, i - k_l), \ldots, \min(n, i + k_u)$, depends on the **order** argument as follows:

if **order** = 'Nag_ColMajor', A_{ij} is stored as $\mathbf{ab}[(j-1) \times \mathbf{pdab} + \mathbf{kl} + \mathbf{ku} + i - j]$;

if **order** = 'Nag_RowMajor', A_{ij} is stored as $\mathbf{ab}[(i-1) \times \mathbf{pdab} + \mathbf{kl} + j - i]$.

See Section 9 in nag_zgbsv (f07bnc) for further details.

On exit: **ab** is overwritten by details of the factorization.

The elements, u_{ij}, of the upper triangular band factor U with $k_l + k_u$ super-diagonals, and the multipliers, l_{ij}, used to form the lower triangular factor L are stored. The elements u_{ij}, for $i = 1, \ldots, m$ and $j = i, \ldots, \min(n, i + k_l + k_u)$, and l_{ij}, for $i = 1, \ldots, m$ and $j = \max(1, i - k_l), \ldots, i$, are stored where A_{ij} is stored on entry.

7: **pdab** – Integer *Input*

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

Constraint: $\mathbf{pdab} \geq 2 \times \mathbf{kl} + \mathbf{ku} + 1$.

8: **ipiv**$[\min(\mathbf{m}, \mathbf{n})]$ – Integer *Output*

On exit: the pivot indices that define the permutation matrix. At the ith step, if $\mathbf{ipiv}[i-1] > i$ then row i of the matrix A was interchanged with row $\mathbf{ipiv}[i-1]$, for $i = 1, 2, \ldots, \min(m, n)$. $\mathbf{ipiv}[i-1] \leq i$ indicates that, at the ith step, a row interchange was not required.

9: **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 $\langle value \rangle$ had an illegal value.

NE_INT

On entry, **kl** $= \langle value \rangle$.
Constraint: **kl** ≥ 0.

On entry, **ku** $= \langle value \rangle$.
Constraint: **ku** ≥ 0.

On entry, **m** $= \langle value \rangle$.
Constraint: **m** ≥ 0.

On entry, **n** $= \langle value \rangle$.
Constraint: **n** ≥ 0.

On entry, **pdab** $= \langle value \rangle$.
Constraint: **pdab** > 0.

NE_INT_3

On entry, **pdab** $= \langle value \rangle$, **kl** $= \langle value \rangle$ and **ku** $= \langle value \rangle$.
Constraint: **pdab** $\geq 2 \times$ **kl** $+$ **ku** $+ 1$.

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.

NE_SINGULAR

$U(\langle value \rangle, \langle value \rangle)$ 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 factors L and U are the exact factors of a perturbed matrix $A + E$, where

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

$c(k)$ is a modest linear function of $k = k_l + k_u + 1$, and ϵ is the ***machine precision***. This assumes $k \ll \min(m, n)$.

8 Parallelism and Performance

nag_zgbtrf (f07brc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_zgbtrf (f07brc) makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

The total number of real floating-point operations varies between approximately $8nk_l(k_u + 1)$ and $8nk_l(k_l + k_u + 1)$, depending on the interchanges, assuming $m = n \gg k_l$ and $n \gg k_u$.

A call to nag_zgbtrf (f07brc) may be followed by calls to the functions:

nag_zgbtrs (f07bsc) to solve $AX = B$, $A^{\mathrm{T}}X = B$ or $A^{\mathrm{H}}X = B$;

nag_zgbcon (f07buc) to estimate the condition number of A.

The real analogue of this function is nag_dgbtrf (f07bdc).

10 Example

This example computes the LU factorization of the matrix A, where

$$A = \begin{pmatrix} -1.65 + 2.26i & -2.05 - 0.85i & 0.97 - 2.84i & 0.00 + 0.00i \\ 0.00 + 6.30i & -1.48 - 1.75i & -3.99 + 4.01i & 0.59 - 0.48i \\ 0.00 + 0.00i & -0.77 + 2.83i & -1.06 + 1.94i & 3.33 - 1.04i \\ 0.00 + 0.00i & 0.00 + 0.00i & 4.48 - 1.09i & -0.46 - 1.72i \end{pmatrix}.$$

Here A is treated as a band matrix with one subdiagonal and two superdiagonals.

10.1 Program Text

```
/* nag_zgbtrf (f07brc) Example Program.
 *
 * Copyright 2001 Numerical Algorithms Group.
 *
 * Mark 7, 2001.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf07.h>
#include <nagx04.h>

int main(void)
{
  /* Scalars */
  Integer       i, ipiv_len, j, kl, ku, m, n, pdab;
  Integer       exit_status = 0;
  NagError      fail;
  Nag_OrderType order;

  /* Arrays */
  Complex       *ab = 0;
  Integer       *ipiv = 0;

#ifdef NAG_COLUMN_MAJOR
#define AB(I, J) ab[(J-1)*pdab + kl + ku + I - J]
  order = Nag_ColMajor;
#else
#define AB(I, J) ab[(I-1)*pdab + kl + J - I]
  order = Nag_RowMajor;
#endif

  INIT_FAIL(fail);

  printf("nag_zgbtrf (f07brc) Example Program Results\n\n");

  /* Skip heading in data file */
  scanf("%*[^\n] ");
  scanf("%ld%ld%ld%ld%*[^\n] ", &m, &n,
         &kl, &ku);
  ipiv_len = MIN(m, n);
  pdab = 2*kl + ku + 1;

  /* Allocate memory */
  if (!(ab = NAG_ALLOC((2*kl+ku+1) * n, Complex)) ||
      !(ipiv = NAG_ALLOC(ipiv_len, Integer)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  /* Read A from data file */
  for (i = 1; i <= m; ++i)
    {
      for (j = MAX(i-kl, 1); j <= MIN(i+ku, n); ++j)
        scanf(" ( %lf , %lf )", &AB(i, j).re, &AB(i, j).im);
    }
  scanf("%*[^\n] ");

  /* Factorize A */
  /* nag_zgbtrf (f07brc).
   * LU factorization of complex m by n band matrix
   */
  nag_zgbtrf(order, m, n, kl, ku, ab, pdab, ipiv, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_zgbtrf (f07brc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  /* Print details of factorization */
  /* nag_band_complx_mat_print_comp (x04dfc).
   * Print complex packed banded matrix (comprehensive)
   */
  fflush(stdout);
  nag_band_complx_mat_print_comp(order, m, n, kl, kl+ku, ab, pdab,
                                 Nag_BracketForm, "%7.4f",
                                 "Details of factorization", Nag_IntegerLabels,
                                 0, Nag_IntegerLabels, 0, 80, 0, 0,
                                 &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_band_complx_mat_print_comp (x04dfc).\n%s\n",
             fail.message);
      exit_status = 1;
      goto END;
    }
  /* Print pivot indices */
  printf("\nipiv\n");
  for (i = 1; i <= MIN(m, n); ++i)
    printf("%3ld%s", ipiv[i-1], i%7 == 0?"\n":"               ");
  printf("\n");
 END:
  NAG_FREE(ab);
  NAG_FREE(ipiv);
  return exit_status;
}
```

10.2 Program Data

```
nag_zgbtrf (f07brc) Example Program Data
  4  4  1  2                                          :Values of M, N, KL and KU
 (-1.65, 2.26) (-2.05,-0.85) ( 0.97,-2.84)
 ( 0.00, 6.30) (-1.48,-1.75) (-3.99, 4.01) ( 0.59,-0.48)
               (-0.77, 2.83) (-1.06, 1.94) ( 3.33,-1.04)
                             ( 4.48,-1.09) (-0.46,-1.72)  :End of matrix A
```

10.3 Program Results

```
nag_zgbtrf (f07brc) Example Program Results

 Details of factorization
                   1                  2                  3                  4
 1  ( 0.0000, 6.3000)  (-1.4800,-1.7500)  (-3.9900, 4.0100)  ( 0.5900,-0.4800)
 2  ( 0.3587, 0.2619)  (-0.7700, 2.8300)  (-1.0600, 1.9400)  ( 3.3300,-1.0400)
 3                     ( 0.2314, 0.6358)  ( 4.9303,-3.0086)  (-1.7692,-1.8587)
 4                                        ( 0.7604, 0.2429)  ( 0.4338, 0.1233)

ipiv
  2                 3                 3                 4
```