NAG C 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

```
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 non-zero sub-diagonals and k_u non-zero super-diagonals, 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 non-zero elements in each column, and U is an upper triangular band matrix with $k_l + k_u$ super-diagonals.

Note that L is not a band matrix, but the non-zero elements of L can be stored in the same space as the sub-diagonal elements of A. U is a band matrix but with k_l additional super-diagonals compared with A. These additional super-diagonals 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 Parameters

1: **order** – Nag_OrderType *Input*

On entry: the **order** parameter 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 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.

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

2: **m** – Integer *Input*

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

Constraint: **m** ≥ 0.

3: **n** – Integer *Input*

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

Constraint: **n** ≥ 0.

4: **kl** – Integer *Input*

On entry: k_l, the number of sub-diagonals within the band of A.

Constraint: **kl** ≥ 0.

5: **ku** – Integer *Input*

On entry: k_u, the number of super-diagonals within the band of A.

Constraint: **ku** ≥ 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** and at least $\max(1, \mathbf{pdab} \times \mathbf{m})$ 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 $i = 1, \ldots, m$ and $j = \max(1, i - k_l), \ldots, \min(n, i + k_u)$, depends on the **order** parameter 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 $\mathbf{ab}[(i-1) \times \mathbf{pdab} + \mathbf{kl} + j - i]$.

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 using the same storage scheme as as described for a_{ij} 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: **pdab** $\geq 2 \times$ **kl** + **ku** + 1.

8: **ipiv**$[dim]$ – Integer *Output*

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

On exit: the pivot indices. Row i of the matrix A was interchanged with row $\mathbf{ipiv}[i-1]$, for $i = 1, 2, \ldots, \min(m, n)$.

9: **fail** – NagError * *Output*

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

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

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

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

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

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

NE_INT_3

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

NE_SINGULAR

The factor U is exactly singular.

NE_ALLOC_FAIL

Memory allocation failed.

NE_BAD_PARAM

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

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 consult NAG for assistance.

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 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 this function may be followed by calls to the functions:

nag_zgbtrs (f07bsc) to solve $AX = B$, $A^TX = B$ or $A^HX = B$;

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

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

9 Example

To compute 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 1 sub-diagonal and 2 super-diagonals.

9.1 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);
  Vprintf("f07brc Example Program Results\n\n");

  /* Skip heading in data file */
  Vscanf("%*[^\n] ");
  Vscanf("%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)) )
    {
      Vprintf("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)
        Vscanf(" ( %lf , %lf )", &AB(i,j).re, &AB(i,j).im);
    }
  Vscanf("%*[^\n] ");

  /* Factorize A */
  f07brc(order, m, n, kl, ku, ab, pdab, ipiv, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from f07brc.\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  /* Print details of factorization */
  x04dfc(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)
    {
      Vprintf("Error from x04dfc.\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  /* Print pivot indices */
  Vprintf("\nIPIV\n");
  for (i = 1; i <= MIN(m,n); ++i)
    Vprintf("%3ld%s", ipiv[i-1], i%7==0 ?"\n":"               ");
  Vprintf("\n");
 END:
  if (ab) NAG_FREE(ab);
  if (ipiv) NAG_FREE(ipiv);
```

```
  return exit_status;
}
```

9.2 Program Data

```
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
```

9.3 Program Results

```
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
```