NAG Library Function Document

nag_zpbtrf (f07hrc)

1 Purpose

nag_zpbtrf (f07hrc) computes the Cholesky factorization of a complex Hermitian positive definite band matrix.

2 Specification

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

```
void nag_zpbtrf (Nag_OrderType order, Nag_UploType uplo, Integer n,
     Integer kd, Complex ab[], Integer pdab, NagError *fail)
```

3 Description

nag_zpbtrf (f07hrc) forms the Cholesky factorization of a complex Hermitian positive definite band matrix A either as $A = U^{\mathrm{H}}U$ if **uplo** $=$ Nag_Upper or $A = LL^{\mathrm{H}}$ if **uplo** $=$ Nag_Lower, where U (or L) is an upper (or lower) triangular band matrix with the same number of superdiagonals (or subdiagonals) as A.

4 References

Demmel J W (1989) On floating-point errors in Cholesky *LAPACK Working Note No. 14* University of Tennessee, Knoxville http://www.netlib.org/lapack/lawnspdf/lawn14.pdf

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: **uplo** – Nag_UploType *Input*

On entry: specifies whether the upper or lower triangular part of A is stored and how A is to be factorized.

uplo $=$ Nag_Upper
The upper triangular part of A is stored and A is factorized as $U^{\mathrm{H}}U$, where U is upper triangular.

uplo $=$ Nag_Lower
The lower triangular part of A is stored and A is factorized as LL^{H}, where L is lower triangular.

Constraint: **uplo** $=$ Nag_Upper or Nag_Lower.

3: **n** – Integer *Input*

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

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

4: **kd** – Integer *Input*

On entry: k_d, the number of superdiagonals or subdiagonals of the matrix A.

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

5: **ab**[*dim*] – Complex *Input/Output*

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

On entry: the n by n Hermitian positive definite band matrix A.

This is stored as a notional two-dimensional array with row elements or column elements stored contiguously. The storage of elements of A_{ij}, depends on the **order** and **uplo** arguments as follows:

if **order** = Nag_ColMajor and **uplo** = Nag_Upper,
A_{ij} is stored in $\mathbf{ab}[k_d + i - j + (j-1) \times \mathbf{pdab}]$, for $j = 1, \ldots, n$ and $i = \max(1, j - k_d), \ldots, j$;
if **order** = Nag_ColMajor and **uplo** = Nag_Lower,
A_{ij} is stored in $\mathbf{ab}[i - j + (j-1) \times \mathbf{pdab}]$, for $j = 1, \ldots, n$ and $i = j, \ldots, \min(n, j + k_d)$;
if **order** = Nag_RowMajor and **uplo** = Nag_Upper,
A_{ij} is stored in $\mathbf{ab}[j - i + (i-1) \times \mathbf{pdab}]$, for $i = 1, \ldots, n$ and $j = i, \ldots, \min(n, i + k_d)$;
if **order** = Nag_RowMajor and **uplo** = Nag_Lower,
A_{ij} is stored in $\mathbf{ab}[k_d + j - i + (i-1) \times \mathbf{pdab}]$, for $i = 1, \ldots, n$ and $j = \max(1, i - k_d), \ldots, i$.

On exit: the upper or lower triangle of A is overwritten by the Cholesky factor U or L as specified by **uplo**, using the same storage format as described above.

6: **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: $\mathbf{pdab} \geq \mathbf{kd} + 1$.

7: **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 ⟨*value*⟩ had an illegal value.

NE_INT

On entry, **kd** = ⟨*value*⟩.
Constraint: $\mathbf{kd} \geq 0$.

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

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

NE_INT_2

On entry, **pdab** $= \langle value \rangle$ and **kd** $= \langle value \rangle$.
Constraint: **pdab** $\geq$ **kd** $+ 1$.

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.

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_POS_DEF

The leading minor of order $\langle value \rangle$ is not positive definite and the factorization could not be completed. Hence A itself is not positive definite. This may indicate an error in forming the matrix A. There is no function specifically designed to factorize a Hermitian band matrix which is not positive definite; the matrix must be treated either as a nonsymmetric band matrix, by calling nag_zgbtrf (f07brc) or as a full Hermitian matrix, by calling nag_zhetrf (f07mrc).

7 Accuracy

If **uplo** $=$ Nag_Upper, the computed factor U is the exact factor of a perturbed matrix $A + E$, where

$$|E| \leq c(k+1)\epsilon|U^{\mathrm{H}}||U|,$$

$c(k+1)$ is a modest linear function of $k+1$, and ϵ is the ***machine precision***.

If **uplo** $=$ Nag_Lower, a similar statement holds for the computed factor L. It follows that $|e_{ij}| \leq c(k+1)\epsilon\sqrt{a_{ii}a_{jj}}$.

8 Parallelism and Performance

nag_zpbtrf (f07hrc) is not threaded by NAG in any implementation.

nag_zpbtrf (f07hrc) 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 X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this function. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

The total number of real floating-point operations is approximately $4n(k+1)^2$, assuming $n \gg k$.

A call to nag_zpbtrf (f07hrc) may be followed by calls to the functions:

nag_zpbtrs (f07hsc) to solve $AX = B$;

nag_zpbcon (f07huc) to estimate the condition number of A.

The real analogue of this function is nag_dpbtrf (f07hdc).

10 Example

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

$$A = \begin{pmatrix} 9.39 + 0.00i & 1.08 - 1.73i & 0.00 + 0.00i & 0.00 + 0.00i \\ 1.08 + 1.73i & 1.69 + 0.00i & -0.04 + 0.29i & 0.00 + 0.00i \\ 0.00 + 0.00i & -0.04 - 0.29i & 2.65 + 0.00i & -0.33 + 2.24i \\ 0.00 + 0.00i & 0.00 + 0.00i & -0.33 - 2.24i & 2.17 + 0.00i \end{pmatrix}.$$

10.1 Program Text

```
/* nag_zpbtrf (f07hrc) Example Program.
 *
 * Copyright 2014 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, j, k, kd, n, pdab;
  Integer       exit_status = 0;
  Nag_UploType  uplo;
  NagError      fail;
  Nag_OrderType order;

  /* Arrays */
  char          nag_enum_arg[40];
  Complex       *ab = 0;
#ifdef NAG_LOAD_FP
  /* The following line is needed to force the Microsoft linker
     to load floating point support */
  float         force_loading_of_ms_float_support = 0;
#endif /* NAG_LOAD_FP */

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

  INIT_FAIL(fail);

  printf("nag_zpbtrf (f07hrc) 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"%"NAG_IFMT"%*[^\n] ", &n, &kd);
#else
  scanf("%"NAG_IFMT"%"NAG_IFMT"%*[^\n] ", &n, &kd);
```

```
#endif
  pdab = kd + 1;

  /* Allocate memory */
  if (!(ab = NAG_ALLOC((kd+1) * n, Complex)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  /* Read A from data file */
#ifdef _WIN32
  scanf_s(" %39s%*[^\n] ", nag_enum_arg, _countof(nag_enum_arg));
#else
  scanf(" %39s%*[^\n] ", nag_enum_arg);
#endif
  /* nag_enum_name_to_value (x04nac).
   * Converts NAG enum member name to value
   */
  uplo = (Nag_UploType) nag_enum_name_to_value(nag_enum_arg);

  k = kd + 1;
  if (uplo == Nag_Upper)
    {
      for (i = 1; i <= n; ++i)
        {
          for (j = i; j <= MIN(i+kd, n); ++j)
#ifdef _WIN32
            scanf_s(" ( %lf , %lf )", &AB_UPPER(i, j).re,
                   &AB_UPPER(i, j).im);
#else
            scanf(" ( %lf , %lf )", &AB_UPPER(i, j).re,
                   &AB_UPPER(i, j).im);
#endif
        }
#ifdef _WIN32
      scanf_s("%*[^\n] ");
#else
      scanf("%*[^\n] ");
#endif
    }
  else
    {
      for (i = 1; i <= n; ++i)
        {
          for (j = MAX(1, i-kd); j <= i; ++j)
#ifdef _WIN32
            scanf_s(" ( %lf , %lf )", &AB_LOWER(i, j).re,
                   &AB_LOWER(i, j).im);
#else
            scanf(" ( %lf , %lf )", &AB_LOWER(i, j).re,
                   &AB_LOWER(i, j).im);
#endif
        }
#ifdef _WIN32
      scanf_s("%*[^\n] ");
#else
      scanf("%*[^\n] ");
#endif
    }
  /* Factorize A */
  /* nag_zpbtrf (f07hrc).
   * Cholesky factorization of complex Hermitian
   * positive-definite band matrix
   */
  nag_zpbtrf(order, uplo, n, kd, ab, pdab, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_zpbtrf (f07hrc).\n%s\n", fail.message);
      exit_status = 1;
```

```
      goto END;
    }
  /* Print details of factorization */
  if (uplo == Nag_Upper)
    {
      /* nag_band_complx_mat_print_comp (x04dfc).
       * Print complex packed banded matrix (comprehensive)
       */
      fflush(stdout);
      nag_band_complx_mat_print_comp(order, n, n, 0, kd, ab, pdab,
                                     Nag_BracketForm, "%7.4f", "Factor",
                                     Nag_IntegerLabels, 0, Nag_IntegerLabels,
                                     0, 80, 0, 0, &fail);
    }
  else
    {
      /* nag_band_complx_mat_print_comp (x04dfc), see above. */
      fflush(stdout);
      nag_band_complx_mat_print_comp(order, n, n, kd, 0, ab, pdab,
                                     Nag_BracketForm, "%7.4f", "Factor",
                                     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;
    }

 END:
  NAG_FREE(ab);
  return exit_status;
}
```

10.2 Program Data

```
nag_zpbtrf (f07hrc) Example Program Data
  4  1                                                        :Values of n and kd
  Nag_Lower                                                   :Value of uplo
 ( 9.39, 0.00)
 ( 1.08, 1.73) ( 1.69, 0.00)
               (-0.04,-0.29) ( 2.65, 0.00)
                             (-0.33,-2.24) ( 2.17, 0.00)  :End of matrix A
```

10.3 Program Results

```
nag_zpbtrf (f07hrc) Example Program Results

 Factor
                   1                  2                  3                  4
 1  ( 3.0643, 0.0000)
 2  ( 0.3524, 0.5646)  ( 1.1167, 0.0000)
 3                     (-0.0358,-0.2597)  ( 1.6066, 0.0000)
 4                                        (-0.2054,-1.3942)  ( 0.4289, 0.0000)
```