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 $A = U^H U$ if uplo = Nag_Upper or $A = LL^H$ 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


5 Arguments

1: order – Nag_OrderType

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

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 $U^H U$, where $U$ is upper triangular.

uplo = Nag_Lower

The lower triangular part of $A$ is stored and $A$ is factorized as $L L^H$, where $L$ is lower triangular.

Constraint: uplo = Nag_Upper or Nag_Lower.
3: \textbf{n} – Integer \hspace{1cm} \textit{Input}

\textit{On entry:} \textit{n}, the order of the matrix \textit{A}.
\textit{Constraint:} \textit{n} \geq 0.

4: \textbf{kd} – Integer \hspace{1cm} \textit{Input}

\textit{On entry:} \textit{kd}, the number of superdiagonals or subdiagonals of the matrix \textit{A}.
\textit{Constraint:} \textit{kd} \geq 0.

5: \textbf{ab[dim]} – Complex \hspace{1cm} \textit{Input/Output}

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

\textit{On entry:} the \textit{n} by \textit{n} Hermitian positive definite band matrix \textit{A}.
This is stored as a notional two-dimensional array with row elements or column elements stored contiguously. The storage of elements of \textit{A}_{ij}, depends on the \textit{order} and \textit{uplo} arguments as follows:

- \textit{if order = Nag_ColMajor and uplo = Nag_Upper,} 
  \textit{A}_{ij} is stored in \textit{ab}[kd + i - j + (j - 1) \times pdab], for \textit{j} = 1, \ldots, \textit{n} and 
  \textit{i} = \max(1, j - \textit{kd}), \ldots, \textit{j};

- \textit{if order = Nag_ColMajor and uplo = Nag_Lower,} 
  \textit{A}_{ij} is stored in \textit{ab}[i - j + (j - 1) \times pdab], for \textit{j} = 1, \ldots, \textit{n} and 
  \textit{i} = \textit{j}, \ldots, \min(\textit{n}, \textit{j} + \textit{kd});

- \textit{if order = Nag_RowMajor and uplo = Nag_Upper,} 
  \textit{A}_{ij} is stored in \textit{ab}[j - i + (i - 1) \times pdab], for \textit{i} = 1, \ldots, \textit{n} and 
  \textit{j} = \textit{i}, \ldots, \min(\textit{n}, \textit{i} + \textit{kd});

- \textit{if order = Nag_RowMajor and uplo = Nag_Lower,} 
  \textit{A}_{ij} is stored in \textit{ab}[kd + j - i + (i - 1) \times pdab], for \textit{i} = 1, \ldots, \textit{n} and 
  \textit{j} = \max(1, \textit{i} - \textit{kd}), \ldots, \textit{i}.

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

6: \textbf{pdab} – Integer \hspace{1cm} \textit{Input}

\textit{On entry:} the stride separating row or column elements (depending on the value of \textit{order}) of the matrix \textit{A} in the array \textit{ab}.
\textit{Constraint:} \textit{pdab} \geq \textit{kd} + 1.

7: \textbf{fail} – NagError * \hspace{1cm} \textit{Input/Output}

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 \hspace{1cm} \textbf{Error Indicators and Warnings}

\textbf{NE_ALLOC_FAIL} 
Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

\textbf{NE_BAD_PARAM} 
On entry, argument \textit{(value)} had an illegal value.

\textbf{NE_INT} 
On entry, \textit{kd} = \textit{(value)}.
\textit{Constraint:} \textit{kd} \geq 0.
On entry, \( n = \langle \text{value} \rangle \).
Constraint: \( n \geq 0 \).

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

**NE_INT_2**

On entry, \( \text{pdab} = \langle \text{value} \rangle \) and \( \text{kd} = \langle \text{value} \rangle \).
Constraint: \( \text{pdab} \geq \text{kd} + 1 \).

**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 \( \langle \text{value} \rangle \) 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 \( \text{nag_zgbtrf (f07brc)} \) or as a full Hermitian matrix, by calling \( \text{nag_zhetrf (f07mrc)} \).

7 **Accuracy**

If \( \text{uplo} = \text{Nag_Upper} \), the computed factor \( U \) is the exact factor of a perturbed matrix \( A + E \), where

\[
|E| \leq c(k + 1)\epsilon |U^H||U|,
\]

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

If \( \text{uplo} = \text{Nag_Lower} \), 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**

\( \text{nag_zpbtrf (f07hrc)} \) is not threaded by NAG in any implementation.

\( \text{nag_zpbtrf (f07hrc)} \) 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 \( \text{nag_zpbtrf (f07hrc)} \) may be followed by calls to the functions:

- \( \text{nag_zpbtrs (f07hsc)} \) to solve \( AX = B \);
- \( \text{nag_zpbccon (f07huc)} \) 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

```c
/* 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]
    #else
    #define AB_UPPER(I, J) ab[(I-1)*pdab + J-I]
    #define ABLOWER(I, J) ab[(I-1)*pdab + k + J - I - 1]
    #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);
    ```

f07hrc.4 Mark 25

NAG Library Manual
```c
#define _WIN32
#endif

int main(void)
{
    Nag_Boolean 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%[
", nag_enum_arg, _countof(nag_enum_arg));
    #else
    scanf(" %39s%[
", 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=k d+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", fail.message);
        exit_status = 1;
    }
    return exit_status;
}
```
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

<table>
<thead>
<tr>
<th>n</th>
<th>kd</th>
<th>uplo</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9.39, 0.00</td>
<td>1.69, 0.00</td>
<td></td>
</tr>
<tr>
<td>1.08, 1.73</td>
<td>-0.04, -0.29</td>
<td></td>
</tr>
<tr>
<td>2.65, 0.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-0.33, -2.24</td>
<td>2.17, 0.00</td>
<td></td>
</tr>
</tbody>
</table>

10.3 Program Results

nag_zpbtrf (f07hrc) Example Program Results

<table>
<thead>
<tr>
<th>Factor</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( 3.0643, 0.0000)</td>
<td>( 1.1167, 0.0000)</td>
<td>( 1.6066, 0.0000)</td>
<td>( 0.4289, 0.0000)</td>
</tr>
<tr>
<td>2</td>
<td>( 0.3524, 0.5646)</td>
<td>( -0.0358, -0.2597)</td>
<td>( 0.2054, -1.3942)</td>
<td>( 0.4289, 0.0000)</td>
</tr>
</tbody>
</table>