NAG C 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

```c
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 \( \text{uplo} = \text{Nag_Upper} \), or \( A = LL^H \) if \( \text{uplo} = \text{Nag_Lower} \), where \( U \) (or \( L \)) is an upper (or lower) triangular band matrix with the same number of super-diagonals (or sub-diagonals) as \( A \).

4 References

Demmel J W (1989) On floating-point errors in Cholesky LAPACK Working Note No. 14 University of Tennessee, Knoxville

5 Parameters

1: \text{order} – Nag_OrderType

\text{On entry:} the \text{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 \text{order} = \text{Nag_RowMajor}. See Section 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.

\text{Constraint:} \text{order} = \text{Nag_RowMajor} or \text{Nag_ColMajor}.

2: \text{uplo} – Nag_UploType

\text{On entry:} indicates whether the upper or lower triangular part of \( A \) is stored and how \( A \) is factorized, as follows:

- if \( \text{uplo} = \text{Nag_Upper} \), the upper triangular part of \( A \) is stored and \( A \) is factorized as \( U^H U \), where \( U \) is upper triangular;
- if \( \text{uplo} = \text{Nag_Lower} \), the lower triangular part of \( A \) is stored and \( A \) is factorized as \( LL^H \), where \( L \) is lower triangular.

\text{Constraint:} \text{uplo} = \text{Nag_Upper} or \text{Nag_Lower}.

3: \text{n} – Integer

\text{On entry:} \( n \), the order of the matrix \( A \).

\text{Constraint:} \text{n} \geq 0.
4: \( \text{kd} \) – Integer

*Input*

*On entry:* \( k \), the number of super-diagonals or sub-diagonals of the matrix \( A \).

*Constraint:* \( \text{kd} \geq 0 \).

5: \( \text{ab}[\text{dim}] \) – Complex

*Input/Output*

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

*On entry:* the \( n \) by \( n \) Hermitian band matrix \( A \). This is stored as a notional two-dimensional array with row elements or column elements stored contiguously. The storage of elements \( a_{ij} \) depends on the \texttt{order} and \texttt{uplo} parameters as follows:

- if \( \texttt{order} = \texttt{Nag\_ColMajor} \) and \( \texttt{uplo} = \texttt{Nag\_Upper} \),
  \( a_{ij} \) is stored in \( \text{ab}[k + i - j + (j - 1) \times \text{pdab}] \), for \( i = 1, \ldots, n \) and
  \( j = i, \ldots, \min(n, i + k) \);

- if \( \texttt{order} = \texttt{Nag\_ColMajor} \) and \( \texttt{uplo} = \texttt{Nag\_Lower} \),
  \( a_{ij} \) is stored in \( \text{ab}[i - j + (j - 1) \times \text{pdab}] \), for \( i = 1, \ldots, n \) and
  \( j = \max(1, i - k), \ldots, i \);

- if \( \texttt{order} = \texttt{Nag\_RowMajor} \) and \( \texttt{uplo} = \texttt{Nag\_Upper} \),
  \( a_{ij} \) is stored in \( \text{ab}[j - i + (i - 1) \times \text{pdab}] \), for \( i = 1, \ldots, n \) and
  \( j = i, \ldots, \min(n, i + k) \);

- if \( \texttt{order} = \texttt{Nag\_RowMajor} \) and \( \texttt{uplo} = \texttt{Nag\_Lower} \),
  \( a_{ij} \) is stored in \( \text{ab}[k + j - i + (i - 1) \times \text{pdab}] \), for \( i = 1, \ldots, n \) and
  \( j = \max(1, i - k), \ldots, i \).

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

6: \( \text{pdab} \) – Integer

*Input*

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

*Constraint:* \( \text{pdab} \geq \text{kd} + 1 \).

7: \( \text{fail} \) – NagError *

*Output*

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

**NE_INT**

- On entry, \( n = \langle \text{value} \rangle \).
  
  Constraint: \( n \geq 0 \).

- On entry, \( \text{kd} = \langle \text{value} \rangle \).
  
  Constraint: \( \text{kd} \geq 0 \).

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

**NE_INT_2**

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

**NE_POS_DEF**

The matrix \( A \) is not positive definite.
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

If \texttt{uplo} = \texttt{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 \textit{machine precision}.

If \texttt{uplo} = \texttt{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 Further Comments

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

A call to this function may be followed by calls to the functions:

- \texttt{nag_zpbtrs (f07hsc)} to solve \(AX = B\);
- \texttt{nag_zpbcon (f07huc)} to estimate the condition number of \(A\).

The real analogue of this function is \texttt{nag_dpbrf (f07hdc)}.

9 Example

To compute the Cholesky factorization of the matrix \(A\), where

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

9.1 Program Text

```c
/* nag_zpbtrf (f07hrc) Example Program. */
/* Copyright 2001 Numerical Algorithms Group. */
/* Mark 7, 2001. */
*/
#include <stdio.h>
#include <nag.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_enum;
```
NagError fail;
Nag_OrderType order;

/* Arrays */
char uplo[2];
Complex *ab=0;

#endif
#define AB_UPPER(I,J) ab[(J-1)*pdab + k + I - J - 1]
#define AB_LOWER(I,J) ab[(J-1)*pdab + I - J]
#endif
#define AB_UPPER(I,J) ab[(I-1)*pdab + J - I]
#define AB_LOWER(I,J) ab[(I-1)*pdab + k + J - I - 1]
#endif

INIT_FAIL(fail);
Vprintf("f07hrc Example Program Results\n\n");

/* Skip heading in data file */
Vscanf("%*[^
\] ");
Vscanf("%ld%ld%*[^
\] ", &n, &kd);
pdab = kd + 1;

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

/* Read A from data file */
Vscanf(" %1s '%*[^
\] ", uplo);
if (*(unsigned char *)uplo == 'L')
  uplo_enum = Nag_Lower;
else if (*(unsigned char *)uplo == 'U')
  uplo_enum = Nag_Upper;
else
{
  Vprintf("Unrecognised character for Nag_UploType type\n");
  exit_status = -1;
  goto END;
}

k = kd + 1;
if (uplo_enum == Nag_Upper)
{
  for (i = 1; i <= n; ++i)
  {
    for (j = i; j <= MIN(i+kd,n); ++j)
      Vscanf(" ( %lf , %lf )", &AB_UPPER(i,j).re, &AB_UPPER(i,j).im);
  }
  Vscanf("%*[\n"]");
}
else
{
  for (i = 1; i <= n; ++i)
  {
    for (j = MAX(1,i-kd); j <= i; ++j)
      Vscanf(" ( %lf , %lf )", &AB_LOWER(i,j).re, &AB_LOWER(i,j).im);
  }
  Vscanf("%*[\n"]");
}

/* Factorize A */
f07hrc(order, uplo_enum, n, kd, ab, pdab, &fail);
if (fail.code != NE_NOERROR)
{
  Vprintf("Error from f07hrc.\n%s\n", fail.message);
  exit_status = 1;
  goto END;
}
/* Print details of factorization */
if (uplo_enum == Nag_Upper)
    x04dfc(order, n, n, 0, kd, ab, pdab, Nag_BracketForm,
        "%7.4f", "Factor", Nag_IntegerLabels,
        0, Nag_IntegerLabels, 0, 80, 0, 0, &fail);
else
    x04dfc(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)
{
    Vprintf("Error from x04dfc.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
END:
if (ab) NAG_FREE(ab);
return exit_status;

9.2 Program Data
f07hrc Example Program Data
4  1
 'L'
( 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

9.3 Program Results
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></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>( 0.3524, 0.5646)</td>
<td>( 1.1167, 0.0000)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>(-0.0358,-0.2597)</td>
<td>( 1.6066, 0.0000)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>(-0.2054,-1.3942)</td>
<td>( 0.4289, 0.0000)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>