NAG Library Function Document

nag_zpotrf (f07frc)

1 Purpose

nag_zpotrf (f07frc) computes the Cholesky factorization of a complex Hermitian positive definite matrix.

2 Specification

```c
#include <nag.h>
#include <nagf07.h>
void nag_zpotrf (Nag_OrderType order, Nag_UploType uplo, Integer n,
              Complex a[], Integer pda, NagError *fail)
```

3 Description

nag_zpotrf (f07frc) forms the Cholesky factorization of a complex Hermitian positive definite matrix $A$ either as $A = U^H U$ if `uplo = Nag_Upper` or $A = LL^H$ if `uplo = Nag_Lower`, where $U$ is an upper triangular matrix and $L$ is lower triangular.

4 References


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 $U^H U$, 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

\textit{Input}

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

\textit{Constraint:} \( n \geq 0 \).

4:  \( a[\text{dim}] \) – Complex

\textit{Input/Output}

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

\textit{On entry:} the \( n \) by \( n \) Hermitian positive definite matrix \( A \).

If \( \text{order} = \text{Nag\_ColMajor} \), \( A_{ij} \) is stored in \( a[(j - 1) \times \text{pda} + i - 1] \).

If \( \text{order} = \text{Nag\_RowMajor} \), \( A_{ij} \) is stored in \( a[(i - 1) \times \text{pda} + j - 1] \).

If \( \text{uplo} = \text{Nag\_Upper} \), the upper triangular part of \( A \) must be stored and the elements of the array below the diagonal are not referenced.

If \( \text{uplo} = \text{Nag\_Lower} \), the lower triangular part of \( A \) must be stored and the elements of the array above the diagonal are not referenced.

\textit{On exit:} the upper or lower triangle of \( A \) is overwritten by the Cholesky factor \( U \) or \( L \) as specified by \( \text{uplo} \).

5:  \( \text{pda} \) – Integer

\textit{Input}

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

\textit{Constraint:} \( \text{pda} \geq \max(1, n) \).

6:  \( \text{fail} \) – NagError*

\textit{Input/Output}

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

6  \textbf{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 \( \langle \text{value} \rangle \) had an illegal value.

**NE\_INT**

On entry, \( n = \langle \text{value} \rangle \).

\textit{Constraint:} \( n \geq 0 \).

On entry, \( \text{pda} = \langle \text{value} \rangle \).

\textit{Constraint:} \( \text{pda} > 0 \).

**NE\_INT\_2**

On entry, \( \text{pda} = \langle \text{value} \rangle \) and \( n = \langle \text{value} \rangle \).

\textit{Constraint:} \( \text{pda} \geq \max(1, n) \).

**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.
7 Accuracy
If uplo = Nag_Upper, the computed factor U is the exact factor of a perturbed matrix A + E, where
\[ |E| \leq c(n)\epsilon \|U\|\|U\|, \]
c(n) is a modest linear function of n, and \( \epsilon \) is the machine precision. If uplo = Nag_Lower, a similar statement holds for the computed factor L. It follows that \( |e_{ij}| \leq c(n)\epsilon \sqrt{a_{ii}a_{jj}}. \)

8 Parallelism and Performance
nag_zpotrf (f07frc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.
nag_zpotrf (f07frc) 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 \( \frac{4}{3}n^3. \)
A call to nag_zpotrf (f07frc) may be followed by calls to the functions:
- nag_zpotrs (f07fsc) to solve \( AX = B; \)
- nag_zpocon (f07fuc) to estimate the condition number of \( A; \)
- nag_zpotri (f07fwc) to compute the inverse of \( A. \)
The real analogue of this function is nag_dpotrf (f07fdc).

10 Example
This example computes the Cholesky factorization of the matrix \( A, \) where
\[
A = \begin{pmatrix}
3.23 + 0.00i & 1.51 - 1.92i & 1.90 + 0.84i & 0.42 + 2.50i \\
1.51 + 1.92i & 3.58 + 0.00i & -0.23 + 1.11i & -1.18 + 1.37i \\
1.90 - 0.84i & -0.23 - 1.11i & 4.09 + 0.00i & 2.33 - 0.14i \\
0.42 - 2.50i & -1.18 - 1.37i & 2.33 + 0.14i & 4.29 + 0.00i
\end{pmatrix}.
\]
10.1 Program Text

/* nag_zpotrf (f07frc) Example Program.  
*  
* Copyright 2014 Numerical Algorithms Group.  
*  
* */

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

int main(void)
{
  /* Scalars */
  Integer i, j, n, pda;
  Integer exit_status = 0;
  Nag_UploType uplo;
  Nag_MatrixType matrix;
  NagError fail;
  Nag_OrderType order;
  /* Arrays */
  char nag_enum_arg[40];
  Complex *a = 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 A(I, J) a[(J-1)*pda +I-1 
    order = Nag_ColMajor;
  #else
    #define A(I, J) a[(I-1)*pda +J-1 
    order = Nag_RowMajor;
  #endif

  INIT_FAIL(fail);

  printf("nag_zpotrf (f07frc) 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"%*[\n] ", &n);
  #else
    scanf("%NAG_IFMT"%*[\n] ", &n);
  #endif
  #ifdef NAG_COLUMN_MAJOR
    pda = n;
  #else
    pda = n;
  #endif

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

  /* Read A from data file */
  #ifdef __WIN32
    scanf_s("%*[\n] ");
  #endif

END:
exit_status = 0;

return exit_status;
}
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);
if (uplo == Nag_Upper)
{
  matrix = Nag_UpperMatrix;
  for (i = 1; i <= n; ++i)
  {
    for (j = i; j <= n; ++j)
#ifdef _WIN32
      scanf_s(" ( %lf , %lf )", &A(i, j).re, &A(i, j).im);
#else
      scanf(" ( %lf , %lf )", &A(i, j).re, &A(i, j).im);
#endif
  }#ifdef _WIN32
  scanf_s("%\n ");
#else
  scanf("%\n ");
#endif
} else
{
  matrix = Nag_LowerMatrix;
  for (i = 1; i <= n; ++i)
  {
    for (j = 1; j <= i; ++j)
#ifdef _WIN32
      scanf_s(" ( %lf , %lf )", &A(i, j).re, &A(i, j).im);
#else
      scanf(" ( %lf , %lf )", &A(i, j).re, &A(i, j).im);
#endif
  }#ifdef _WIN32
  scanf_s("%\n ");
#else
  scanf("%\n ");
#endif
}
/* Factorize A */
/* nag_zpotrf (f07frc).
* Cholesky factorization of complex Hermitian
* positive-definite matrix *
*/
nag_zpotrf(order, uplo, n, a, pda, &fail);
if (fail.code != NE_NOERROR)
{
  printf("Error from nag_zpotrf (f07frc).\n", fail.message);
  exit_status = 1;
goto END;
} /* Print factor */
/* nag_gen_complx_mat_print_comp (x04dbc).
* Print complex general matrix (comprehensive) *
*/
fflush(stdout);
nag_gen_complx_mat_print_comp(order, matrix, Nag_NonUnitDiag, n, n, a, pda, Nag_BracketForm, "%7.4f", "Factor",
Nag_IntegerLabels, 0, Nag_IntegerLabels, 0, 80, 0, 0, &fail);
if (fail.code != NE_NOERROR)
{
  printf("Error from nag_gen_complx_mat_print_comp (x04dbc).\n", fail.message);
10.2 Program Data

nag_zpotrf (f07frc) Example Program Data

4 :Value of n
Nag_Lower :Value of uplo
(3.23, 0.00) (1.51, 1.92) (3.58, 0.00) (1.90,-0.84)
(1.90,-0.84) (-0.23,-1.11) (4.09, 0.00) (-0.23,-1.11)
(0.42,-2.50) (-1.18,-1.37) (2.33, 0.14) (2.33, 0.14) :End of matrix A

10.3 Program Results

nag_zpotrf (f07frc) Example Program Results

Factor

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.7972, 0.0000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.8402, 1.0683</td>
<td>1.3164, 0.0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1.0572,-0.4674</td>
<td>-0.4702, 0.3131</td>
<td>1.5604, 0.0000</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.2337,-1.3910</td>
<td>0.0834, 0.0368</td>
<td>0.9360, 0.9900</td>
<td>0.6603, 0.0000</td>
</tr>
</tbody>
</table>