NAG C Library Function Document
nag_zpptrf (f07grc)

1 Purpose
nag_zpptrf (f07grc) computes the Cholesky factorization of a complex Hermitian positive-definite matrix, using packed storage.

2 Specification

```c
void nag_zpptrf (Nag_OrderType order, Nag_UploType uplo, Integer n, Complex ap[], NagError *fail)
```

3 Description
nag_zpptrf (f07grc) forms the Cholesky factorization of a complex Hermitian positive-definite 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 $U$ is an upper triangular matrix and $L$ is lower triangular, using packed storage.

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

*Input*

*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.

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

2: $\text{uplo}$ – Nag_UploType

*Input*

*On entry:* indicates whether the upper or lower triangular part of $A$ is stored and how $A$ is factorized, as follows:

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

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

3: $n$ – Integer

*Input*

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

*Constraint:* $n \geq 0$.

4: $\text{ap}[]$ – Complex

*Input/Output*

*Note:* the dimension, $\text{dim}$, of the array $\text{ap}$ must be at least $\max(1, n \times (n + 1)/2)$. 
On entry: the Hermitian positive-definite matrix $A$, packed by rows or columns. The storage of elements $a_{ij}$ depends on the order and uplo parameters as follows:

- If order = Nag_ColMajor and uplo = Nag_Upper, $a_{ij}$ is stored in $\text{ap}[\left( j - 1 \right) \times j/2 + i - 1]$, for $i \leq j$;
- If order = Nag_ColMajor and uplo = Nag_Lower, $a_{ij}$ is stored in $\text{ap}[\left(2n - j\right) \times (j - 1)/2 + i - 1]$, for $i \geq j$;
- If order = Nag_RowMajor and uplo = Nag_Upper, $a_{ij}$ is stored in $\text{ap}[\left(2n - i\right) \times (i - 1)/2 + j - 1]$, for $i \leq j$;
- If order = Nag_RowMajor and uplo = Nag_Lower, $a_{ij}$ is stored in $\text{ap}[\left(i - 1\right) \times i/2 + j - 1]$, for $i \geq j$.

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 packed storage format as described above.

5: fail – NagError *

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NER_INT
On entry, $n = \langle value \rangle$.
Constraint: $n \geq 0$.

NER_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 $A$. To factorize a Hermitian matrix which is not positive-definite, call nag_zhptrf (f07prc) instead.

NER_ALLOC_FAIL
Memory allocation failed.

NER_BAD_PARAM
On entry, parameter $\langle value \rangle$ had an illegal value.

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

$|E| \leq c(n)\epsilon|U^H||U|$, 

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 Further Comments

The total number of real floating-point operations is approximately $\frac{4}{3}n^3$.

A call to this function may be followed by calls to the functions:
nag_zpptrs (f07gsc) to solve $AX = B$;
nag_zppcon (f07guc) to estimate the condition number of $A$;
nag_zpptri (f07gwc) to compute the inverse of $A$.

The real analogue of this function is nag_dpptrf (f07gdc).

9 Example

To compute the Cholesky factorization of the matrix $A$, where

$$A = \begin{pmatrix}
3.23 + 0.00i & 1.51 - 1.92i & 1.90 + 0.84i & 0.42 + 2.50i \\
1.51 + 1.92i & 3.58 + 0.00i & -0.23 + 1.11i & -1.18 + 1.37i \\
1.90 - 0.84i & -0.23 - 1.11i & 4.09 + 0.00i & 2.33 - 0.14i \\
0.42 - 2.50i & -1.18 - 1.37i & 2.33 + 0.14i & 4.29 + 0.00i
\end{pmatrix},$$

using packed storage.

9.1 Program Text

/* nag_zpptrf (f07grc) 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 ap_len, i, j, n;
    Integer exit_status=0;
    NagError fail;
    Nag_UploType uplo_enum;
    Nag_OrderType order;

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

    #ifdef NAG_COLUMN_MAJOR
    #define A_UPPER(I,J) ap[J*(J-1)/2 + I - 1]
    #define A_LOWER(I,J) ap[(2*n-J)*(J-1)/2 + I - 1]
    #else
    #define A_LOWER(I,J) ap[I*(I-1)/2 + J - 1]
    #define A_UPPER(I,J) ap[(2*n-I)*(I-1)/2 + J - 1]
    #endif
    order = Nag_ColMajor;
    
    INIT_FAIL(fail);
    Vprintf("f07grc Example Program Results\n\n");

    /* Skip heading in data file */
    Vscanf("%*[\n"");
    Vscanf("%ld%*[\n]", &n);
    ap_len = n * (n + 1)/2;

    /* Allocate memory */
    if( !(ap = NAG_ALLOC(ap_len, Complex)) )
    {
        Vprintf("Allocation failure\n");
    }

    /* ... */
}
exit_status = -1;
goto END;
}
/* Read A from data file */
Vscanf("' %ls '^[\n] ", 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;
}
if (uplo_enum == Nag_Upper)
{
  for (i = 1; i <= n; ++i)
  {
    for (j = i; j <= n; ++j)
      Vscanf(" ( %lf , %lf )", &A_UPPER(i,j).re, &A_UPPER(i,j).im);
  }
  Vscanf("%[*\n] ");
}
else {
  for (i = 1; i <= n; ++i)
  {
    for (j = 1; j <= i; ++j)
      Vscanf(" ( %lf , %lf )", &A_LOWER(i,j).re, &A_LOWER(i,j).im);
  }
  Vscanf("%[*\n] ");
}
/* Factorize A */
f07grc(order, uplo_enum, n, ap, &fail);
if (fail.code != NE_NOERROR)
{
  Vprintf("Error from f07grc.\n%s\n", fail.message);
  exit_status = 1;
  goto END;
}
/* Print details of factorization */
x04ddc(order, uplo_enum, Nag_NonUnitDiag, n, ap,
       Nag_BracketForm, "%7.4f", "Factor", Nag_IntegerLabels,
       0, Nag_IntegerLabels, 0, 80, 0, 0, &fail);
if (fail.code != NE_NOERROR)
{
  Vprintf("Error from x04ddc.\n%s\n", fail.message);
  exit_status = 1;
  goto END;
}
END:
if (ap) NAG_FREE(ap);
return exit_status;

9.2 Program Data

f07grc Example Program Data
4  :Value of N
 'L'  :Value of UPLO
(3.23, 0.00)
(1.51, 1.92) ( 3.58, 0.00)
(1.90,-0.84) (-0.23,-1.11) ( 4.09, 0.00)
(0.42,-2.50) (-1.18,-1.37) ( 2.33, 0.14) ( 4.29, 0.00) :End of matrix A

f07grc.4 [NP3645/7]
### 9.3 Program Results

**f07grc 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>( 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>