1 Purpose

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

2 Specification

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 $\text{uplo} = \text{Nag_Upper}$, or $A = LL^H$ if $\text{uplo} = \text{Nag_Lower}$, where $U$ is an upper triangular matrix and $L$ is lower triangular.

4 References

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

5 Parameters

1: order – Nag_OrderType
   On entry: the 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 order = Nag_RowMajor. See Section 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.
   Constraint: order = Nag_RowMajor or Nag_ColMajor.

2: uplo – Nag_UploType
   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.
   Constraint: uplo = Nag_Upper or Nag_Lower.

3: n – Integer
   On entry: $n$, the order of the matrix $A$.
   Constraint: $n \geq 0$.

4: a[dim] – Complex
   Input/Output
   Note: the dimension, $dim$, of the array a must be at least max(1, pda $\times n$).
If \( \text{order} = \text{Nag\_ColMajor} \), the \((i, j)\)th element of the matrix \( A \) is stored in \( a[(j - 1) \times \text{pda} + i - 1] \) and if \( \text{order} = \text{Nag\_RowMajor} \), the \((i, j)\)th element of the matrix \( A \) is stored in \( a[(i - 1) \times \text{pda} + j - 1] \).

On entry: the \( n \) by \( n \) Hermitian positive-definite matrix \( A \). If \( \text{uplo} = \text{Nag\_Upper} \), the upper triangle 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 triangle of \( A \) must be stored and the elements of the array above the diagonal are not referenced.

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

\( \text{Input} \)

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

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

6: \( \text{fail} \) – NagError*

\( \text{Output} \)

The NAG error *parameter (see the Essential Introduction).

### 6 Error Indicators and Warnings

**NE\_INT**

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

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

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

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

**NE\_INT\_2**

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

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

**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 \( A \). To factorize a Hermitian matrix which is not positive-definite, call nag_zhetrf (f07mrc) instead.

**NE\_ALLOC\_FAIL**

Memory allocation failed.

**NE\_BAD\_PARAM**

On entry, parameter \( \langle \text{value} \rangle \) 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 \( \text{uplo} = \text{Nag\_Upper} \), the computed factor \( U \) is the exact factor of a perturbed matrix \( A + E \), where

\[ |E| \leq c(n)\varepsilon|U^H||U|, \]

\( c(n) \) is a modest linear function of \( n \), and \( \varepsilon \) is the \textit{machine precision}. If \( \text{uplo} = \text{Nag\_Lower} \), a similar statement holds for the computed factor \( L \). It follows that \( |e_{ij}| \leq c(n)\varepsilon\sqrt{a_{ii}a_{jj}} \).
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_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).

9 Example

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

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

9.1 Program Text

/* nag_zpotrf (f07frc) 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 i, j, n, pda;
    Integer exit_status=0;
    Nag_UploType uplo_enum;
    Nag_MatrixType matrix;
    NagError fail;
    Nag_OrderType order;
    /* Arrays */
    char uplo[2];
    Complex *a=0;

#ifdef NAG_COLUMN_MAJOR
#define A(I,J) a[(J-1)*pda+I-1]
#else
#define A(I,J) a[(I-1)*pda+J-1]
#endif

    INIT_FAIL(fail);
    Vprintf("f07frc Example Program Results

");
    /* Skip heading in data file */
    Vscanf("%*
[^
] ");
    Vscanf("%ld%*
[^
] ", &n);
#elifdef NAG_COLUMN_MAJOR
    pda = n;
#else

[NP3645/7]
pda = n;
#endif
/* Allocate memory */
if ( !(a = NAG_ALLOC(n* n, Complex)) )
{
    Vprintf("Allocation failure\n");
    exit_status = -1;
    goto END;
}
/* Read A from data file */
Vscanf(" %>s {%[\n ] ", uplo);
if (*(unsigned char *)uplo == 'L')
{
    uplo_enum = Nag_Lower;
    matrix = Nag_LowerMatrix;
}
else if (*(unsigned char *)uplo == 'U')
{
    uplo_enum = Nag_Upper;
    matrix = Nag_UpperMatrix;
}
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(i,j).re, &A(i,j).im);
    
    Vscanf("%*[\n ] ");
}
else
{
    for (i = 1; i <= n; ++i)
    {
        for (j = 1; j <= i; ++j)
            Vscanf(" ( %lf , %lf )", &A(i,j).re, &A(i,j).im);
    
    Vscanf("%*[\n ] ");
}

/* Factorize A */
f07frc(order, uplo_enum, n, a, pda, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07frc.\n\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Print factor */
x04dbc(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)
{
    Vprintf("Error from x04dbc.\n\n", fail.message);
    exit_status = 1;
    goto END;
}
END:
if (a) NAG_FREE(a);
return exit_status;
9.2 Program Data

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

9.3 Program Results

Example Program Results

Factor 1 2 3 4
1 (1.7972, 0.0000)
2 (0.8402, 1.0683) (1.3164, 0.0000)
3 (1.0572, -0.4674) (-0.4702, 0.3131) (1.5604, 0.0000)
4 (0.2337, -1.3910) (0.0834, 0.0368) (0.9360, 0.9900) (0.6603, 0.0000)