NAG C Library Function Document

nag_dpotrf (f07fdc)

1 Purpose
nag_dpotrf (f07fdc) computes the Cholesky factorization of a real symmetric positive-definite matrix.

2 Specification
void nag_dpotrf (Nag_OrderType order, Nag_UploType uplo, Integer n, double a[],
    Integer pda, NagError *fail)

3 Description
nag_dpotrf (f07fdc) forms the Cholesky factorization of a real symmetric positive-definite matrix $A$ either as $A = U^T U$ if $\text{uplo} = \text{Nag}\_\text{Upper}$, or $A = LL^T$ if $\text{uplo} = \text{Nag}\_\text{Lower}$, 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 $\text{uplo} = \text{Nag}\_\text{Upper}$, the upper triangular part of $A$ is stored and $A$ is factorized as $U^T U$, where $U$ is upper triangular;
   
   if $\text{uplo} = \text{Nag}\_\text{Lower}$, the lower triangular part of $A$ is stored and $A$ is factorized as $LL^T$, 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] – double
   
   Note: the dimension, dim, of the array a must be at least max(1, pda × n).
If order = Nag_ColMajor, the \((i, j)\)th element of the matrix \(A\) is stored in \(a[(j - 1) \times \text{pda} + i - 1]\) and if order = 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 \(n\) symmetric positive-definite matrix \(A\). If uplo = Nag_Upper, the upper triangle of \(A\) must be stored and the elements of the array below the diagonal are not referenced; if uplo = 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 uplo.

5: \text{pda} – Integer
On entry: the stride separating row or column elements (depending on the value of order) of the matrix \(A\) in the array \(a\).
Constraint: \text{pda} \geq \max(1, n).

6: \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, \(\text{pda} = \langle \text{value} \rangle\).
Constraint: \(\text{pda} > 0\).

**NE_INT_2**
On entry, \(\text{pda} = \langle \text{value} \rangle, n = \langle \text{value} \rangle\).
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 symmetric matrix which is not positive-definite, call nag_dsytrf (f07mdc) 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 uplo = Nag_Upper, the computed factor \(U\) is the exact factor of a perturbed matrix \(A + E\), where

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

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

The total number of floating-point operations is approximately $\frac{1}{3}n^3$.

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

- `nag_dpotrs` (f07fec) to solve $AX = B$;
- `nag_dpocon` (f07fgc) to estimate the condition number of $A$;
- `nag_dpotri` (f07fjc) to compute the inverse of $A$.

The complex analogue of this function is `nag_zpotrf` (f07frc).

9 Example

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

$$
A = \begin{pmatrix}
4.16 & -3.12 & 0.56 & -0.10 \\
-3.12 & 5.03 & -0.83 & 1.18 \\
0.56 & -0.83 & 0.76 & 0.34 \\
-0.10 & 1.18 & 0.34 & 1.18
\end{pmatrix}
$$

9.1 Program Text

```c
/* nag_dpotrf (f07fdc) 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;

    INIT_FAIL(fail);
    Vprintf("f07fdc Example Program Results\n\n");

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

    ifndef 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);
    Vprintf("f07fdc Example Program Results\n\n");

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

[NP3645/7]
pda = n;
#endif

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

/* Read A from data file */
Vscanf("' %ls ' %*[\`
]", 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", &A(i,j));
    }
}
else
{
    for (i = 1; i <= n; ++i)
    {
        for (j = 1; j <= i; ++j)
            Vscanf("%lf", &A(i,j));
    }
}

/* Factorize A */
f07fdc(order, uplo_enum, n, a, pda, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07fdc.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Print factor */
x04cac(order, matrix, Nag_SameSignAsInput, n, n, a, pda,
"Factor", 0, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from x04cac.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
END:
if (a) NAG_FREE(a);
return exit_status;
9.2 Program Data

Example Program Data

\[ \begin{array}{cccc}
4 & \text{Value of N} \\
'L' & \text{Value of UPLO} \\
4.16 & \\
-3.12 & 5.03 & \\
0.56 & -0.83 & 0.76 & \\
-0.10 & 1.18 & 0.34 & 1.18 & \text{:End of matrix A} \\
\end{array} \]

9.3 Program Results

Example Program Results

\begin{array}{cccc}
\text{Factor} & 1 & 2 & 3 & 4 \\
1 & 2.0396 & & & \\
2 & -1.5297 & 1.6401 & & \\
3 & 0.2746 & -0.2500 & 0.7887 & \\
4 & -0.0490 & 0.6737 & 0.6617 & 0.5347 \\
\end{array} \]