NAG Library Function Document
nag_dpptrf (f07gdc)

1 Purpose
nag_dpptrf (f07gdc) computes the Cholesky factorization of a real symmetric positive definite matrix, using packed storage.

2 Specification
#include <nag.h>
#include <nagf07.h>
void nag_dpptrf (Nag_OrderType order, Nag_UploType uplo, Integer n, double ap[], NagError *fail)

3 Description
nag_dpptrf (f07gdc) forms the Cholesky factorization of a real symmetric positive definite matrix $A$ either as $A = U^T U$ if uplo = Nag_Upper or $A = LL^T$ if uplo = Nag_Lower, where $U$ is an upper triangular matrix and $L$ is lower triangular, using packed storage.

4 References

5 Arguments
1: order – Nag_OrderType

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

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 $U^T U$, where $U$ is upper triangular.

uplo = Nag_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

\textit{Input}

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

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

4: \( \text{ap}[\text{dim}] \) – double

\textit{Input/Output}

\textbf{Note:} the dimension, \( \text{dim} \), of the array \( \text{ap} \) must be at least \( \max(1, n \times (n + 1)/2) \).

On entry: the \( n \) by \( n \) symmetric matrix \( A \), packed by rows or columns.

The storage of elements \( A_{ij} \) depends on the \textit{order} and \textit{uplo} arguments as follows:

- if \textit{order} = \text{Nag}_\text{ColMajor} and \textit{uplo} = \text{Nag}_\text{Upper},
  \( A_{ij} \) is stored in \( \text{ap}[(j - 1) \times j/2 + i - 1] \), for \( i \leq j \);
- if \textit{order} = \text{Nag}_\text{ColMajor} and \textit{uplo} = \text{Nag}_\text{Lower},
  \( A_{ij} \) is stored in \( \text{ap}[(2n - j) \times (j - 1)/2 + i - 1] \), for \( i \geq j \);
- if \textit{order} = \text{Nag}_\text{RowMajor} and \textit{uplo} = \text{Nag}_\text{Upper},
  \( A_{ij} \) is stored in \( \text{ap}[(2n - i) \times (i - 1)/2 + j - 1] \), for \( i \leq j \);
- if \textit{order} = \text{Nag}_\text{RowMajor} and \textit{uplo} = \text{Nag}_\text{Lower},
  \( A_{ij} \) is stored in \( \text{ap}[(i - 1) \times i/2 + j - 1] \), for \( i \geq j \).

On exit: if \textit{fail.code} = \text{NE_NOERROR}, the factor \( U \) or \( L \) from the Cholesky factorization \( A = U^T U \) or \( A = LL^T \), in the same storage format as \( A \).

5: \( \text{fail} \) – \text{NagError}*

\textit{Input/Output}

\textbf{The NAG error argument} (see Section 3.6 in the Essential Introduction).

6 \textbf{Error Indicators and Warnings}

\textbf{NE_ALLOC_FAIL}

Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

\textbf{NE_BAD_PARAM}

On entry, argument \( \langle \text{value} \rangle \) had an illegal value.

\textbf{NE_INT}

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

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

\textbf{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.

\textbf{NE_NO_LICENCE}

Your licence key may have expired or may not have been installed correctly.
See Section 3.6.5 in the Essential Introduction for further information.

\textbf{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 \text{nag_dsptrf (f07pdc)} instead.
7 Accuracy

If \( \text{uplo} = \text{Nag}_\text{Upper} \), the computed factor \( U \) is the exact factor of a perturbed matrix \( A + E \), where

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

\( c(n) \) is a modest linear function of \( n \), and \( \epsilon \) is the **machine precision**.

If \( \text{uplo} = \text{Nag}_\text{Lower} \), 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

\( \text{nag_dpptrf (f07gdc)} \) is not threaded by NAG in any implementation.

\( \text{nag_dpptrf (f07gdc)} \) 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 floating-point operations is approximately \( \frac{1}{3}n^3 \).

A call to \( \text{nag_dpptrf (f07gdc)} \) may be followed by calls to the functions:

- \( \text{nag_dpptrs (f07gec)} \) to solve \( AX = B \);
- \( \text{nag_dppcon (f07ggc)} \) to estimate the condition number of \( A \);
- \( \text{nag_dpptri (f07gjc)} \) to compute the inverse of \( A \).

The complex analogue of this function is \( \text{nag_zpptrf (f07grc)} \).

10 Example

This example computes the Cholesky factorization of the matrix \( A \), where

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

using packed storage.

10.1 Program Text

```c
/* nag_dpptrf (f07gdc) Example Program. */
/* Copyright 2014 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;
Nag_UploType uplo;
NagError fail;
Nag_OrderType order;

/* Arrays */
char nag_enum_arg[40];
double *ap = 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_UPPER(I, J) ap[J*(J-1)/2 + I - 1]
#define A_LOWER(I, J) ap[(2*n-J)*(J-1)/2 + I - 1]
order = Nag_ColMajor;
#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]
order = Nag_RowMajor;
#endif

INIT_FAIL(fail);
printf("nag_dpptrf (f07gdc) Example Program Results\n\n");

/* Skip heading in data file */
#ifdef __WIN32
scanf_s("%*[\n"]");
#else
scanf("%*[\n"]");
#endif
#ifdef __WIN32
scanf("%"NAG_IFMT"%*[\n"]", &n);
#else
scanf("%"NAG_IFMT"%*[\n"]", &n);
#endif
ap_len = n*(n+1)/2;

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

/* Read A from data file */
#ifdef __WIN32
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)
{
    for (i = 1; i <= n; ++i)
    {
        for (j = i; j <= n; ++j)
#ifdef __WIN32
            scanf_s("%lf", &A_UPPER(i, j));
#else
            scanf("%lf", &A_UPPER(i, j));
#endif
    
}
```c
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif

else
{
    for (i = 1; i <= n; ++i)
    {
        for (j = 1; j <= i; ++j)
        {
            #ifdef _WIN32
                scanf_s("%lf", &A_LOWER(i, j));
            #else
                scanf("%lf", &A_LOWER(i, j));
            #endif
        }
    }
#endif

    /* Factorize A */
    /* nag_dpptrf (f07gdc). */
    /* Cholesky factorization of real symmetric positive-definite matrix, packed storage */
    nag_dpptrf(order, uplo, n, ap, &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_dpptrf (f07gdc). \n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }
    /* Print factor */
    /* nag_pack_real_mat_print (x04ccc). */
    /* Print real packed triangular matrix (easy-to-use) */
    fflush(stdout);
    nag_pack_real_mat_print(order, uplo, Nag_NonUnitDiag, n, ap,
                            "Factor", 0, &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_pack_real_mat_print (x04ccc). \n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }
}

END:
NAG_FREE(ap);
return exit_status;
```

### 10.2 Program Data

**nag_dpptrf (f07gdc) Example Program Data**

<table>
<thead>
<tr>
<th>Value of n</th>
<th>Value of uplo</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Nag_Lower</td>
</tr>
<tr>
<td>-3.12</td>
<td>5.03</td>
</tr>
<tr>
<td>4.16</td>
<td>-0.83</td>
</tr>
<tr>
<td>0.76</td>
<td>-0.10</td>
</tr>
<tr>
<td>1.18</td>
<td>0.34</td>
</tr>
<tr>
<td>1.18</td>
<td>End of matrix A</td>
</tr>
</tbody>
</table>
### 10.3 Program Results

**nag_dpptrf (f07gdc) 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>2.0396</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>-1.5297</td>
<td>1.6401</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.2746</td>
<td>-0.2500</td>
<td>0.7887</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-0.0490</td>
<td>0.6737</td>
<td>0.6617</td>
<td>0.5347</td>
</tr>
</tbody>
</table>