NAG Library Function Document

nag_dpptri (f07gjc)

1 Purpose

nag_dpptri (f07gjc) computes the inverse of a real symmetric positive definite matrix \( A \), where \( A \) has been factorized by nag_dpptrf (f07gdc), using packed storage.

2 Specification

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

3 Description

nag_dpptri (f07gjc) is used to compute the inverse of a real symmetric positive definite matrix \( A \), the function must be preceded by a call to nag_dpptrf (f07gdc), which computes the Cholesky factorization of \( A \), using packed storage.

If \( \text{uplo} = \text{Nag_Upper} \), \( A = U^T U \) and \( A^{-1} \) is computed by first inverting \( U \) and then forming \( (U^{-1})U^{-T} \).

If \( \text{uplo} = \text{Nag_Lower} \), \( A = LL^T \) and \( A^{-1} \) is computed by first inverting \( L \) and then forming \( L^{-T}(L^{-1}) \).

4 References


5 Arguments

1: \( \text{order} \) – Nag_OrderType

\( \text{Input} \)

\( \text{On entry:} \) the \( \text{order} \) 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 \( \text{order} = \text{Nag_RowMajor} \). See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

\( \text{Constraint:} \ \text{order} = \text{Nag_RowMajor} \) or \( \text{Nag_ColMajor} \).

2: \( \text{uplo} \) – Nag_UploType

\( \text{Input} \)

\( \text{On entry:} \) specifies how \( A \) has been factorized.

\( \text{uplo} = \text{Nag_Upper} \)

\( A = U^T U \), where \( U \) is upper triangular.

\( \text{uplo} = \text{Nag_Lower} \)

\( A = LL^T \), where \( L \) is lower triangular.

\( \text{Constraint:} \ \text{uplo} = \text{Nag_Upper} \) or \( \text{Nag_Lower} \).

3: \( \text{n} \) – Integer

\( \text{Input} \)

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

\( \text{Constraint:} \ n \geq 0 \).
4: \( \text{ap}[\text{dim}] – \text{double} \)

**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 Cholesky factor of \( A \) stored in packed form, as returned by \text{nag_dppttrf} (\text{f07gdc}).

- On exit: the factorization is overwritten by the \( n \) by \( n \) matrix \( A^{-1} \).

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

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

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

**Input/Output**

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

6 **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 \( \text{value} \) had an illegal value.

**NE_INT**

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

Constraint: \( n \geq 0 \).

**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.

**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.

**NE_SINGULAR**

Diagonal element \( \text{value} \) of the Cholesky factor is zero; the Cholesky factor is singular and the inverse of \( A \) cannot be computed.

7 **Accuracy**

The computed inverse \( X \) satisfies

\[
\|XA - I\|_2 \leq c(n)\epsilon\kappa_2(A) \quad \text{and} \quad \|AX - I\|_2 \leq c(n)\epsilon\kappa_2(A),
\]

where \( c(n) \) is a modest function of \( n \), \( \epsilon \) is the machine precision and \( \kappa_2(A) \) is the condition number of \( A \) defined by
8 Parallelism and Performance

nag_dpptri (f07gjc) is not threaded by NAG in any implementation.

nag_dpptri (f07gjc) 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 \( \frac{2}{3} n^3 \).

The complex analogue of this function is nag_zpptri (f07gwc).

10 Example

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

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

Here \( A \) is symmetric positive definite, stored in packed form, and must first be factorized by nag_dpptrf (f07gdc).

10.1 Program Text

```c
#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;
    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 link er
     to load floating point support */
    float force_loading_of_ms_float_support = 0;
    #endif /* NAG_LOAD_FP */
```

Mark 25

f07gjc.3
#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;
#else
order = Nag_RowMajor;
#endif

INIT_FAIL(fail);

printf("nag_dpptri (f07gjc) 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

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
    }
#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
    }
#else
}
```c
#include <stdio.h>
#include <nag.h>

int main(void)
{
    int i, j, n = 4, exit_status = 0;
    double A[16] = {-3.12, 5.03, 0.56, -0.83, 0.76, -0.10, 1.18, 0.34, 1.18};

    /* Factorize A */
    nag_dpptrf(order, uplo, n, ap, &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_dpptrf (f07gjc). \n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }

    /* Compute inverse of A */
    nag_dpptri(order, uplo, n, ap, &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_dpptri (f07gjc). \n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }

    /* Print inverse */
    nag_pack_real_mat_print(order, uplo, Nag_NonUnitDiag, n, ap,
                            "Inverse", 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_dpptri (f07gjc) Example Program Data**

<table>
<thead>
<tr>
<th>n</th>
<th>Value of n</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>uplo</th>
<th>Value of uplo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lower</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-3.12</td>
<td>5.03</td>
</tr>
<tr>
<td>0.56</td>
<td>-0.83</td>
</tr>
<tr>
<td>-0.10</td>
<td>1.18</td>
</tr>
</tbody>
</table>

:End of matrix A
### 10.3 Program Results

*nag_dpptri (f07gjc)* Example Program Results

<table>
<thead>
<tr>
<th>Inverse</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.6995</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.7769</td>
<td>1.4239</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.7508</td>
<td>1.8255</td>
<td>4.0688</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-0.9340</td>
<td>-1.8841</td>
<td>-2.9342</td>
<td>3.4978</td>
</tr>
</tbody>
</table>