NAG C 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
void nag_dpptri (Nag_OrderType order, Nag_UploType uplo, Integer n, double ap[],
                 NagError *fail)
```

3 Description

To compute the inverse of a real symmetric positive-definite matrix $A$, this 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 $(U^{-1})(U^{-1})^T$.

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

4 References


5 Parameters

1:  \textit{order} – Nag\_OrderType

\textit{Input}

On entry: the \textit{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 \textit{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: \textit{order} = Nag\_RowMajor or Nag\_ColMajor.

2:  \textit{uplo} – Nag\_UploType

\textit{Input}

On entry: indicates whether $A$ has been factorized as $U^T U$ or $LL^T$ as follows:

- if $\text{uplo} = \text{Nag\_Upper}$, $A = U^T U$, where $U$ is upper triangular;
- if $\text{uplo} = \text{Nag\_Lower}$, $A = LL^T$, where $L$ is lower triangular.

Constraint: \textit{uplo} = Nag\_Upper or Nag\_Lower.

3:  \textit{n} – Integer

\textit{Input}

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

Constraint: $n \geq 0$.

4:  \textit{ap}[\textit{dim}] – double

\textit{Input/Output}

Note: the dimension, \textit{dim}, of the array \textit{ap} must be at least max(1, $n \times (n + 1)/2$).
On entry: the upper triangular matrix $U$ stored in packed form if $\text{uplo} = \text{Nag}_\text{Upper}$ or the lower triangular matrix $L$ stored in packed form if $\text{uplo} = \text{Nag}_\text{Lower}$, as returned by nag_dpptrf (f07gdc).

On exit: $U$ is overwritten by the upper triangle of $A^{-1}$ if $\text{uplo} = \text{Nag}_\text{Upper}$; $L$ is overwritten by the lower triangle of $A^{-1}$ if $\text{uplo} = \text{Nag}_\text{Lower}$, using the same storage scheme as on entry.

5: fail – NagError *

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

**NE_INT**

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

Constraint: $n \geq 0$.

**NE_SINGULAR**

Element $\langle\text{value}\rangle$ of the diagonal of the Cholesky factor is zero. The Cholesky factor is singular, and the inverse of $A$ cannot be computed.

**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

The computed inverse $X$ satisfies

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

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

$$\kappa_2(A) = \|A\|_2 \|A^{-1}\|_2.$$

8 Further Comments

The total number of floating-point operations is approximately $\frac{3}{2}n^3$.

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

9 Example

To compute 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 \texttt{nag_dpptrf} (f07gdc).

### 9.1 Program Text

```c
/* nag_dpptri (f07gjc) 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];
    double *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]
    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);
    Vprintf("f07gjc 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, double)) )
    {
        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)
    {
        f07 – Linear Equations (LAPACK)
        f07gjc
    }
```

[NP3645/7]  f07gjc.3


```c
{ 
  for (i = 1; i <= n; ++i) 
  { 
    for (j = i; j <= n; ++j)
      Vscanf("%lf", &A_UPPER(i,j));
  
  } 
  Vscanf("%*[\n"]");
} 
else 
{ 
  for (i = 1; i <= n; ++i) 
  { 
    for (j = 1; j <= i; ++j)
      Vscanf("%lf", &A_LOWER(i,j));
  
  } 
  Vscanf("%*[\n"]");
}

/* Factorize A */
f07gdc(order, uplo_enum, n, ap, &fail);
if (fail.code != NE_NOERROR) 
{ 
  Vprintf("Error from f07gdc.\n%s\n", fail.message);
  exit_status = 1;
  goto END;
}

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

/* Print inverse */
x04ccc(order, uplo_enum, Nag_NonUnitDiag, n, ap,
      "Inverse", 0, NAGERR_DEFAULT);
if (fail.code != NE_NOERROR) 
{ 
  Vprintf("Error from x04ccc.\n%s\n", fail.message);
  exit_status = 1;
  goto END;
}

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

9.2 Program Data

**f07gjc Example Program Data**

<table>
<thead>
<tr>
<th>Value of N</th>
<th>Value of UPLO</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>'L'</td>
</tr>
<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>
<tr>
<td>0.34</td>
<td>1.18</td>
</tr>
</tbody>
</table>

9.3 Program Results

**f07gjc Example Program Results**

```
Inverse

    1  2  3     4
  1  0.6995
  2  0.7769  1.4239
  3  0.7508  1.8255  4.0688
  4 -0.9340 -1.8841 -2.9342  3.4978
```