NAG C Library Function Document  
nag_dpotri (f07fjc)

1 Purpose

nag_dpotri (f07fjc) computes the inverse of a real symmetric positive-definite matrix $A$, where $A$ has been factorized by nag_dpotrf (f07fdc).

2 Specification

```c
void nag_dpotri (Nag_OrderType order, Nag_UploType uplo, Integer n, double a[],
                Integer pda, 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_dpotrf (f07fdc), which computes the Cholesky factorization of $A$.

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

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

4 References


5 Parameters

1: 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: $\text{order} = \text{Nag_RowMajor}$ or $\text{Nag_ColMajor}$.

2: uplo -- Nag_UploType

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

 if $\text{uplo} = \text{NagUpper}$, $A = U^T U$, where $U$ is upper triangular;

 if $\text{uplo} = \text{NagLower}$, $A = LL^T$, where $L$ is lower triangular.

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

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).

[NP3645/7] f07fjc.1
On entry: the upper triangular matrix $U$ if $\text{uplo} = \text{Nag\_Upper}$ or the lower triangular matrix $L$ if $\text{uplo} = \text{Nag\_Lower}$, as returned by nag_dpotrf (f07fdc).

On exit: $U$ is overwritten by the upper triangle of $A^{-1}$ if $\text{uplo} = \text{Nag\_Upper}$; $L$ is overwritten by the lower triangle of $A^{-1}$ if $\text{uplo} = \text{Nag\_Lower}$.

5: **pda** – Integer  
   *Input*

   On entry: the stride separating row or column elements (depending on the value of order) of the matrix in the array $a$.

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

6: **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\_SINGULAR**

Diagonal element $\langle\text{value}\rangle$ 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 $\frac{4}{3}n^3$. 

f07fjc.2
The complex analogue of this function is nag_zpotri (f07fwc).

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 and must first be factorized by nag_dpotrf (f07fdc).

9.1 Program Text

/* nag_dpotri (f07fjc) Example Program. 
 */
/* Copyright 2001 Numerical Algorithms Group. */
/* Mark 7, 2001. */

#include <stdio.h>
#include <nag.h>
#include <nag_stdbase.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];
    double *a=0;

    #ifdef 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("f07fjc Example Program Results

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

    #ifdef NAG_COLUMN_MAJOR
    pda = n;
    #else
    pda = n;
    #endif

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

    /* ... */

    /* ... */

    /* ... */

    /* ... */

    /* ... */

    return 0;
}

[NP3645/7]

f07fjc.3
/* Read A from data file */
Vscanf("' %ls '\%[\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");
extit_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));
  Vscanf("%*[\n] ");
}
}
else
{
for (i = 1; i <= n; ++i)
{  
  for (j = 1; j <= i; ++j)
   Vscanf("%lf", &A(i,j));
  Vscanf("%*[\n] ");
}
}

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

/* Compute inverse of A */
f07fjc(order, uplo_enum, n, a, pda, &fail);
if (fail.code != NE_NOERROR)
{
Vprintf("Error from f07fjc.\n%\s\n", fail.message);
extit_status = 1;
goto END;
}
/* Print inverse */
x04cac(order, matrix, Nag_NonUnitDiag, n, n, a, pda,
"Inverse", 0, &fail);
if (fail.code != NE_NOERROR)
{
Vprintf("Error from x04cac.\n%\s\n", fail.message);
extit_status = 1;
goto END;
}
END:
if (a) NAG_FREE(a);
return exit_status;
9.2 Program Data

f07fjc Example Program Data

Value of N
'L'
Value of UPLO

4.16
-3.12  5.03
0.56 -0.83  0.76
-0.10  1.18  0.34  1.18 : End of matrix A

9.3 Program Results

f07fjc Example Program Results

Inverse

<table>
<thead>
<tr>
<th></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>