NAG 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 (f07fde).

2 Specification

#include <nag.h>
#include <nagf07.h>

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

3 Description

nag_dpotri (f07fjc) is used to compute the inverse of a real symmetric positive definite matrix $A$, the function must be preceded by a call to nag_dpotrf (f07fde), which computes the Cholesky factorization of $A$.

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

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

4 References


5 Arguments

1: 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 how $A$ has been factorized.

$\text{uplo} = \text{Nag}_\text{Upper}$

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

$\text{uplo} = \text{Nag}_\text{Lower}$

$A = LL^T$, 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: $n \geq 0$. 

Mark 25

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

\textit{Input/Output}  

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

\textit{On entry:} the upper triangular matrix \( U \) if \( \text{uplo} = \text{Nag}_\text{Upper} \) or the lower triangular matrix \( L \) if \( \text{uplo} = \text{Nag}_\text{Lower} \), as returned by \text{nag_dpotrf} (f07fde).

\textit{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 \( \text{uplo} = \text{Nag}_\text{Lower} \).

5: \( \text{pda} \) – Integer  

\textit{Input}  

\textit{On entry:} the stride separating row or column elements (depending on the value of \text{order}) of the matrix in the array \( \text{a} \).  

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

6: \( \text{fail} \) – NagError *  

\textit{Input/Output}  

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

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

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

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

Constraint: \( \text{pda} > 0 \).

\textbf{NE_INT_2}  

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

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

\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_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.
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

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

8 Parallelism and Performance

nag_dpotri (f07fjc) is not threaded by NAG in any implementation.

nag_dpotri (f07fjc) 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_zpotri (f07fwc).

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 and must first be factorized by nag_dpotrf (f07fdc).

10.1 Program Text

/* nag_dpotri (f07fjc) Example Program.
 * Copyright 2014 Numerical Algorithms Group.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdf.h>
#include <nagf07.h>
#include <nagx04.h>

int main(void)
{
    /* Scalars */
    Integer i, j, n, pda;
    Integer exit_status = 0;
    Nag_UploType uplo;
    Nag_MatrixType matrix;
    NagError fail;
    Nag_OrderType order;
    /* Arrays */
char nag_enum_arg[40];
double *a = 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 */

#ifndef NAG_COLUMN_MAJOR
#define A(I, J) a[(J-1)*pda + I - 1]
#else
#define A(I, J) a[(I-1)*pda + J - 1]
#endif

order = Nag_ColMajor;

INIT_FAIL(fail);

printf("nag_dpotri (f07fjc) Example Program Results\n\n");
/* Skip heading in data file */
#ifndef _WIN32
scanf_s("%*[\n ] ");
#else
scanf("%*[\n ] ");
#endif
#ifndef _WIN32
scanf_s("%"NAG_IFMT"%*[\n ] ", &n);
#else
scanf("%"NAG_IFMT"%*[\n ] ", &n);
#endif
#ifndef NAG_COLUMN_MAJOR
pda = n;
#else
pda = n;
#endif

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

/* Read A from data file */
#ifndef _WIN32
scanf_s(" %39s%*[\n ] ", nag_enum_arg, _countof(nag_enum_arg));
#else
scanf(" %39s%*[\n ] ", nag_enum_arg);
#endif

uplo = (Nag_UploType) nag_enum_name_to_value(nag_enum_arg);
if (uplo == Nag_Upper)
{
    matrix = Nag_UpperMatrix;
    for (i = 1; i <= n; ++i)
    {
        for (j = i; j <= n; ++j)
        {
            scanf("%f", &a[i, j]);
        }
    }
}
#ifndef _WIN32
scanf_s("%*[\n ] ");
#else
scanf("%*[\n ] ");
#endif
#endif
scanf("%*[\n] ");
#endif
else
{
    matrix = Nag_LowerMatrix;
    for (i = 1; i <= n; ++i)
    {
        for (j = 1; j <= i; ++j)
        {
            ifdef _WIN32
                scanf_s("%lf", &A(i, j));
            #else
                scanf("%lf", &A(i, j));
            ifdef _WIN32
                scanf_s("%*[\n] ");
            #else
                scanf("%*[\n] ");
            endif
        }
    }
    /* Factorize A */
    /* nag_dpotrf (f07fdc). */
    /* Cholesky factorization of real symmetric 
     * positive-definite matrix */
    nag_dpotrf(order, uplo, n, a, pda, &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_dpotrf (f07fdc).
            %s
            ", fail.message);
        exit_status = 1;
        goto END;
    }
    /* Compute inverse of A */
    /* nag_dpotri (f07fjc). */
    /* Inverse of real symmetric positive-definite matrix, 
     * matrix already factorized by nag_dpotrf (f07fdc) */
    nag_dpotri(order, uplo, n, a, pda, &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_dpotri (f07fjc).
            %s
            ", fail.message);
        exit_status = 1;
        goto END;
    }
    /* Print inverse */
    /* nag_gen_real_mat_print (x04cac). */
    /* Print real general matrix (easy-to-use) */
    fflush(stdout);
    nag_gen_real_mat_print(order, matrix, Nag_NonUnitDiag, n, n, a, pda,
        "Inverse", 0, &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_gen_real_mat_print (x04cac).
            %s
            ", fail.message);
        exit_status = 1;
        goto END;
    }
END:
NAG_FREE(a);
return exit_status;
}
10.2 Program Data

nag_dpotri (f07fjc) Example Program Data
4 :Value of n
Nag_Lower :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

10.3 Program Results

nag_dpotri (f07fjc) 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>