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

```
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 **uplo** = **Nag_Upper**, $A = U^TU$ and A^{-1} is computed by first inverting U and then forming $(U^{-1})(U^{-1})^T$.

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

4 References

Du Croz J J and Higham N J (1992) Stability of methods for matrix inversion *IMA J. Numer. Anal.* **12** 1–19

5 Parameters

1: **order** – Nag_OrderType *Input*

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: **order** = **Nag_RowMajor** or **Nag_ColMajor**.

2: **uplo** – Nag_UploType *Input*

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

if **uplo** = **Nag_Upper**, $A = U^TU$, where U is upper triangular;

if **uplo** = **Nag_Lower**, $A = LL^T$, where L is lower triangular.

Constraint: **uplo** = **Nag_Upper** or **Nag_Lower**.

3: **n** – Integer *Input*

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

Constraint: **n** ≥ 0.

4: **a**[dim] – double *Input/Output*

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

On entry: the upper triangular matrix U if **uplo** $=$ **Nag_Upper** or the lower triangular matrix L if **uplo** $=$ **Nag_Lower**, as returned by nag_dpotrf (f07fdc).

On exit: U is overwritten by the upper triangle of A^{-1} if **uplo** $=$ **Nag_Upper**; L is overwritten by the lower triangle of A^{-1} if **uplo** $=$ **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: **pda** $\geq \max(1, \mathbf{n})$.

6: **fail** – NagError * *Output*

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 0.

On entry, **pda** = ⟨*value*⟩.
Constraint: **pda** > 0.

NE_INT_2

On entry, **pda** = ⟨*value*⟩, **n** = ⟨*value*⟩.
Constraint: **pda** $\geq \max(1, \mathbf{n})$.

NE_SINGULAR

Diagonal element ⟨*value*⟩ 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 ⟨*value*⟩ 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

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

where $c(n)$ is a modest function of n, ϵ 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 $\frac{2}{3}n^3$.

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

```
  /* Read A from data file */
  Vscanf(" ' %1s '%*[^\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");
      exit_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);
      exit_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);
      exit_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);
      exit_status = 1;
      goto END;
    }
 END:
  if (a) NAG_FREE(a);
  return exit_status;
}
```

9.2 Program Data

```
f07fjc Example Program Data
  4                           :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
             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
```