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 (f07fdc).

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 (f07fdc), which computes the Cholesky factorization of A.

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

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

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 Arguments

1: **order** – Nag_OrderType *Input*

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 *Input*

On entry: specifies how A has been factorized.

uplo $=$ Nag_Upper
$A = U^{\mathrm{T}}U$, where U is upper triangular.

uplo $=$ Nag_Lower
$A = LL^{\mathrm{T}}$, 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 $\max(1, \mathbf{pda} \times \mathbf{n})$.

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

6: **fail** – 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 ⟨*value*⟩ had an illegal value.

NE_INT

On entry, $\mathbf{n} = \langle value \rangle$.
Constraint: $\mathbf{n} \geq 0$.

On entry, $\mathbf{pda} = \langle value \rangle$.
Constraint: $\mathbf{pda} > 0$.

NE_INT_2

On entry, $\mathbf{pda} = \langle value \rangle$ and $\mathbf{n} = \langle value \rangle$.
Constraint: $\mathbf{pda} \geq \max(1, \mathbf{n})$.

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 ⟨*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, 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 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.
 *
 * 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;
  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 */

#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);

  printf("nag_dpotri (f07fjc) 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
#ifdef 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 */
#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)
    {
      matrix = Nag_UpperMatrix;
      for (i = 1; i <= n; ++i)
        {
          for (j = i; j <= n; ++j)
#ifdef _WIN32
            scanf_s("%lf", &A(i, j));
#else
            scanf("%lf", &A(i, j));
#endif
        }
#ifdef _WIN32
      scanf_s("%*[^\n] ");
#else
```

```
      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));
#endif
        }
#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).\n%s\n", 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).\n%s\n", 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).\n%s\n",
             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

 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
```