NAG 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

```
#include <nag.h>
#include <nagf07.h>
void nag_dpptri (Nag_OrderType order, Nag_UploType uplo, Integer n,
     double ap[], NagError *fail)
```

3 Description

nag_dpptri (f07gjc) is used to compute the inverse of a real symmetric positive definite matrix A, the function must be preceded by a call to nag_dpptrf (f07gdc), which computes the Cholesky factorization of A, using packed storage.

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: **ap**[dim] – double *Input/Output*

Note: the dimension, *dim*, of the array **ap** must be at least $\max(1, \mathbf{n} \times (\mathbf{n}+1)/2)$.

On entry: the Cholesky factor of A stored in packed form, as returned by nag_dpptrf (f07gdc).

On exit: the factorization is overwritten by the n by n matrix A^{-1}.

The storage of elements A_{ij} depends on the **order** and **uplo** arguments as follows:

if **order** = 'Nag_ColMajor' and **uplo** = 'Nag_Upper',
A_{ij} is stored in **ap**$[(j-1) \times j/2 + i - 1]$, for $i \leq j$;
if **order** = 'Nag_ColMajor' and **uplo** = 'Nag_Lower',
A_{ij} is stored in **ap**$[(2n-j) \times (j-1)/2 + i - 1]$, for $i \geq j$;
if **order** = 'Nag_RowMajor' and **uplo** = 'Nag_Upper',
A_{ij} is stored in **ap**$[(2n-i) \times (i-1)/2 + j - 1]$, for $i \leq j$;
if **order** = 'Nag_RowMajor' and **uplo** = 'Nag_Lower',
A_{ij} is stored in **ap**$[(i-1) \times i/2 + j - 1]$, for $i \geq j$.

5: **fail** – NagError * *Input/Output*

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

6 Error Indicators and Warnings

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_INT

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

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.

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

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

8 Parallelism and Performance

nag_dpptri (f07gjc) is not threaded by NAG in any implementation.

nag_dpptri (f07gjc) 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 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_zpptri (f07gwc).

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, stored in packed form, and must first be factorized by nag_dpptrf (f07gdc).

10.1 Program Text

```
/* 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;
  Nag_OrderType order;
  /* Arrays */
  char          nag_enum_arg[40];
  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);

  printf("nag_dpptri (f07gjc) Example Program Results\n\n");

  /* Skip heading in data file */
  scanf("%*[^\n] ");
  scanf("%ld%*[^\n] ", &n);
  ap_len = n * (n + 1)/2;
```

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

  /* Read A from data file */
  scanf(" %39s%*[^\n] ", nag_enum_arg);
  /* 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)
    {
      for (i = 1; i <= n; ++i)
        {
          for (j = i; j <= n; ++j)
            scanf("%lf", &A_UPPER(i, j));
        }
      scanf("%*[^\n] ");
    }
  else
    {
      for (i = 1; i <= n; ++i)
        {
          for (j = 1; j <= i; ++j)
            scanf("%lf", &A_LOWER(i, j));
        }
      scanf("%*[^\n] ");
    }

  /* Factorize A */
  /* nag_dpptrf (f07gdc).
   * Cholesky factorization of real symmetric
   * positive-definite matrix, packed storage
   */
  nag_dpptrf(order, uplo, n, ap, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_dpptrf (f07gdc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  /* Compute inverse of A */
  /* nag_dpptri (f07gjc).
   * Inverse of real symmetric positive-definite matrix,
   * matrix already factorized by nag_dpptrf (f07gdc), packed
   * storage
   */
  nag_dpptri(order, uplo, n, ap, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_dpptri (f07gjc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  /* Print inverse */
  /* nag_pack_real_mat_print (x04ccc).
   * Print real packed triangular matrix (easy-to-use)
   */
  fflush(stdout);
  nag_pack_real_mat_print(order, uplo, Nag_NonUnitDiag, n, ap,
                          "Inverse", 0, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_pack_real_mat_print (x04ccc).\n%s\n",
              fail.message);
      exit_status = 1;
```

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

10.2 Program Data

```
nag_dpptri (f07gjc) 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_dpptri (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
```