NAG Library Function Document

nag_dpptrf (f07gdc)

1 Purpose

nag_dpptrf (f07gdc) computes the Cholesky factorization of a real symmetric positive definite matrix, using packed storage.

2 Specification

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

3 Description

nag_dpptrf (f07gdc) forms the Cholesky factorization of a real symmetric positive definite matrix A either as $A = U^{\mathrm{T}}U$ if **uplo** = Nag_Upper or $A = LL^{\mathrm{T}}$ if **uplo** = Nag_Lower, where U is an upper triangular matrix and L is lower triangular, using packed storage.

4 References

Demmel J W (1989) On floating-point errors in Cholesky *LAPACK Working Note No. 14* University of Tennessee, Knoxville

Golub G H and Van Loan C F (1996) *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

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 whether the upper or lower triangular part of A is stored and how A is to be factorized.

uplo = Nag_Upper
The upper triangular part of A is stored and A is factorized as $U^{\mathrm{T}}U$, where U is upper triangular.

uplo = Nag_Lower
The lower triangular part of A is stored and A is factorized as 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: **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 n by n symmetric matrix A, packed by rows or columns.

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 $i \geq j$.

On exit: if **fail.code** = NE_NOERROR, the factor U or L from the Cholesky factorization $A = U^{\mathrm{T}}U$ or $A = LL^{\mathrm{T}}$, in the same storage format as A.

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_POS_DEF

The leading minor of order ⟨*value*⟩ is not positive definite and the factorization could not be completed. Hence A itself is not positive definite. This may indicate an error in forming the matrix A. To factorize a symmetric matrix which is not positive definite, call nag_dsptrf (f07pdc) instead.

7 Accuracy

If **uplo** = Nag_Upper, the computed factor U is the exact factor of a perturbed matrix $A + E$, where

$$|E| \leq c(n)\epsilon \left|U^{\mathrm{T}}\right||U|,$$

$c(n)$ is a modest linear function of n, and ϵ is the ***machine precision***.

If **uplo** = Nag_Lower, a similar statement holds for the computed factor L. It follows that $\left|e_{ij}\right| \leq c(n)\epsilon\sqrt{a_{ii}a_{jj}}$.

8 Parallelism and Performance

nag_dpptrf (f07gdc) is not threaded by NAG in any implementation.

nag_dpptrf (f07gdc) 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 $\frac{1}{3}n^3$.

A call to nag_dpptrf (f07gdc) may be followed by calls to the functions:

nag_dpptrs (f07gec) to solve $AX = B$;

nag_dppcon (f07ggc) to estimate the condition number of A;

nag_dpptri (f07gjc) to compute the inverse of A.

The complex analogue of this function is nag_zpptrf (f07grc).

10 Example

This example computes the Cholesky factorization of the matrix A, where

$$A = \begin{pmatrix} 4.16 & -3.12 & 0.56 & -0.10 \\ -3.12 & 5.03 & -0.83 & 1.18 \\ 0.56 & -0.83 & 0.76 & 0.34 \\ -0.10 & 1.18 & 0.34 & 1.18 \end{pmatrix},$$

using packed storage.

10.1 Program Text

```
/* nag_dpptrf (f07gdc) 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;
  Nag_UploType  uplo;
  NagError      fail;
  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_dpptrf (f07gdc) 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;
    }
  /* Print factor */
  /* 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,
                          "Factor", 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_dpptrf (f07gdc) 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_dpptrf (f07gdc) Example Program Results

 Factor
            1          2          3          4
 1      2.0396
 2     -1.5297     1.6401
 3      0.2746    -0.2500     0.7887
 4     -0.0490     0.6737     0.6617     0.5347
```