

# NAG Library Function Document

## nag\_dpbtrf (f07hdc)

### 1 Purpose

nag\_dpbtrf (f07hdc) computes the Cholesky factorization of a real symmetric positive definite band matrix.

### 2 Specification

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

void nag_dpbtrf (Nag_OrderType order, Nag_UploType uplo, Integer n,
                Integer kd, double ab[], Integer pdab, NagError *fail)
```

### 3 Description

nag\_dpbtrf (f07hdc) forms the Cholesky factorization of a real symmetric positive definite band matrix  $A$  either as  $A = U^T U$  if **uplo** = Nag\_Upper or  $A = L L^T$  if **uplo** = Nag\_Lower, where  $U$  (or  $L$ ) is an upper (or lower) triangular band matrix with the same number of superdiagonals (or subdiagonals) as  $A$ .

### 4 References

Demmel J W (1989) On floating-point errors in Cholesky *LAPACK Working Note No. 14* University of Tennessee, Knoxville <http://www.netlib.org/lapack/lawnspdf/lawn14.pdf>

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 2.3.1.3 in How to Use the NAG Library and its Documentation 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  $U^T U$ , where  $U$  is upper triangular.  
**uplo** = Nag\_Lower  
 The lower triangular part of  $A$  is stored and  $A$  is factorized as  $L L^T$ , 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:*  $n \geq 0$ .
- 4: **kd** – Integer *Input*  
*On entry:*  $k_d$ , the number of superdiagonals or subdiagonals of the matrix  $A$ .  
*Constraint:*  $kd \geq 0$ .
- 5: **ab**[*dim*] – double *Input/Output*  
**Note:** the dimension, *dim*, of the array **ab** must be at least  $\max(1, \mathbf{pdab} \times \mathbf{n})$ .  
*On entry:* the  $n$  by  $n$  symmetric positive definite band matrix  $A$ .  
This is stored as a notional two-dimensional array with row elements or column elements stored contiguously. The storage of elements of  $A_{ij}$ , depends on the **order** and **uplo** arguments as follows:
- if **order** = Nag\_ColMajor and **uplo** = Nag\_Upper,  
 $A_{ij}$  is stored in **ab**[ $k_d + i - j + (j - 1) \times \mathbf{pdab}$ ], for  $j = 1, \dots, n$  and  
 $i = \max(1, j - k_d), \dots, j$ ;  
if **order** = Nag\_ColMajor and **uplo** = Nag\_Lower,  
 $A_{ij}$  is stored in **ab**[ $i - j + (j - 1) \times \mathbf{pdab}$ ], for  $j = 1, \dots, n$  and  
 $i = j, \dots, \min(n, j + k_d)$ ;  
if **order** = Nag\_RowMajor and **uplo** = Nag\_Upper,  
 $A_{ij}$  is stored in **ab**[ $j - i + (i - 1) \times \mathbf{pdab}$ ], for  $i = 1, \dots, n$  and  
 $j = i, \dots, \min(n, i + k_d)$ ;  
if **order** = Nag\_RowMajor and **uplo** = Nag\_Lower,  
 $A_{ij}$  is stored in **ab**[ $k_d + j - i + (i - 1) \times \mathbf{pdab}$ ], for  $i = 1, \dots, n$  and  
 $j = \max(1, i - k_d), \dots, i$ .
- On exit:* the upper or lower triangle of  $A$  is overwritten by the Cholesky factor  $U$  or  $L$  as specified by **uplo**, using the same storage format as described above.
- 6: **pdab** – Integer *Input*  
*On entry:* the stride separating row or column elements (depending on the value of **order**) of the matrix  $A$  in the array **ab**.  
*Constraint:*  $\mathbf{pdab} \geq \mathbf{kd} + 1$ .
- 7: **fail** – NagError \* *Input/Output*  
The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

## 6 Error Indicators and Warnings

### NE\_ALLOC\_FAIL

Dynamic memory allocation failed.

See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

### NE\_BAD\_PARAM

On entry, argument  $\langle value \rangle$  had an illegal value.

**NE\_INT**

On entry, **kd** =  $\langle value \rangle$ .

Constraint: **kd**  $\geq 0$ .

On entry, **n** =  $\langle value \rangle$ .

Constraint: **n**  $\geq 0$ .

On entry, **pdab** =  $\langle value \rangle$ .

Constraint: **pdab**  $> 0$ .

**NE\_INT\_2**

On entry, **pdab** =  $\langle value \rangle$  and **kd** =  $\langle value \rangle$ .

Constraint: **pdab**  $\geq \mathbf{kd} + 1$ .

**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 2.7.6 in How to Use the NAG Library and its Documentation for further information.

**NE\_NO\_LICENCE**

Your licence key may have expired or may not have been installed correctly.

See Section 2.7.5 in How to Use the NAG Library and its Documentation for further information.

**NE\_POS\_DEF**

The leading minor of order  $\langle value \rangle$  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  $A$ . There is no function specifically designed to factorize a symmetric band matrix which is not positive definite; the matrix must be treated either as a nonsymmetric band matrix, by calling `nag_dgbtrf` (f07bdc) or as a full symmetric matrix, by calling `nag_dsytrf` (f07mdc).

**7 Accuracy**

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

$$|E| \leq c(k+1)\epsilon |U^T| |U|,$$

$c(k+1)$  is a modest linear function of  $k+1$ , and  $\epsilon$  is the *machine precision*.

If **uplo** = Nag\_Lower, a similar statement holds for the computed factor  $L$ . It follows that  $|e_{ij}| \leq c(k+1)\epsilon \sqrt{a_{ii}a_{jj}}$ .

**8 Parallelism and Performance**

`nag_dpbtrf` (f07hdc) 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  $n(k+1)^2$ , assuming  $n \gg k$ .

A call to `nag_dpbtrf` (f07hdc) may be followed by calls to the functions:

nag\_dpbtrs (f07hec) to solve  $AX = B$ ;

nag\_dpbcon (f07hgc) to estimate the condition number of  $A$ .

The complex analogue of this function is nag\_zpbtrf (f07hrc).

## 10 Example

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

$$A = \begin{pmatrix} 5.49 & 2.68 & 0.00 & 0.00 \\ 2.68 & 5.63 & -2.39 & 0.00 \\ 0.00 & -2.39 & 2.60 & -2.22 \\ 0.00 & 0.00 & -2.22 & 5.17 \end{pmatrix}.$$

### 10.1 Program Text

```

/* nag_dpbtrf (f07hdc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

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

int main(void)
{
    /* Scalars */
    Integer i, j, k, kd, n, pdab;
    Integer exit_status = 0;
    Nag_UploType uplo;
    NagError fail;
    Nag_OrderType order;
    /* Arrays */
    char nag_enum_arg[40];
    double *ab = 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 AB_UPPER(I, J) ab[(J-1)*pdab + k + I - J - 1]
#define AB_LOWER(I, J) ab[(J-1)*pdab + I - J]
    order = Nag_ColMajor;
#else
#define AB_UPPER(I, J) ab[(I-1)*pdab + J - I]
#define AB_LOWER(I, J) ab[(I-1)*pdab + k + J - I - 1]
    order = Nag_RowMajor;
#endif

    INIT_FAIL(fail);

    printf("nag_dpbtrf (f07hdc) 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 "%" NAG_IFMT "%*[\n] ", &n, &kd);
#else
    scanf("%" NAG_IFMT "%" NAG_IFMT "%*[\n] ", &n, &kd);
#endif
pdab = kd + 1;

/* Allocate memory */
if (!(ab = NAG_ALLOC((kd + 1) * 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, (unsigned)_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);

k = kd + 1;
if (uplo == Nag_Upper) {
    for (i = 1; i <= n; ++i) {
        for (j = i; j <= MIN(i + kd, n); ++j)
#ifdef _WIN32
            scanf_s("%lf", &AB_UPPER(i, j));
#else
            scanf("%lf", &AB_UPPER(i, j));
#endif
    }
}
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif
}
else {
    for (i = 1; i <= n; ++i) {
        for (j = MAX(1, i - kd); j <= i; ++j)
#ifdef _WIN32
            scanf_s("%lf", &AB_LOWER(i, j));
#else
            scanf("%lf", &AB_LOWER(i, j));
#endif
    }
}
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif
}

/* Factorize A */
/* nag_dpbtrf (f07hdc).
 * Cholesky factorization of real symmetric
 * positive-definite band matrix
 */
fflush(stdout);
nag_dpbtrf(order, uplo, n, kd, ab, pdab, &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_dpbtrf (f07hdc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Print details of factorization */
if (uplo == Nag_Upper)

```

```

/* nag_band_real_mat_print (x04cec).
 * Print real packed banded matrix (easy-to-use)
 */
nag_band_real_mat_print(order, n, n, 0, kd, ab, pdab, "Factor", 0, &fail);
else
/* nag_band_real_mat_print (x04cec), see above. */
fflush(stdout);
nag_band_real_mat_print(order, n, n, kd, 0, ab, pdab, "Factor", 0, &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_band_real_mat_print (x04cec).\n%s\n",
        fail.message);
    exit_status = 1;
    goto END;
}
END:
    NAG_FREE(ab);
    return exit_status;
}

```

## 10.2 Program Data

```

nag_dpbtrf (f07hdc) Example Program Data
4 1 :Values of n and kd
Nag_Lower :Value of uplo
5.49
2.68 5.63
-2.39 2.60
-2.22 5.17 :End of matrix A

```

## 10.3 Program Results

nag\_dpbtrf (f07hdc) Example Program Results

Factor	1	2	3	4
1	2.3431			
2	1.1438	2.0789		
3		-1.1497	1.1306	
4			-1.9635	1.1465

---