

# NAG Library Function Document

## nag\_dpftf (f07wdc)

### 1 Purpose

nag\_dpftf (f07wdc) computes the Cholesky factorization of a real symmetric positive definite matrix stored in Rectangular Full Packed (RFP) format.

### 2 Specification

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

void nag_dpftf (Nag_OrderType order, Nag_RFP_Store transr,
               Nag_UploType uplo, Integer n, double ar[], NagError *fail)
```

### 3 Description

nag\_dpftf (f07wdc) forms the Cholesky factorization of a real symmetric positive definite matrix  $A$  either as  $A = U^T U$  if **uplo** = Nag\_Upper or  $A = L L^T$  if **uplo** = Nag\_Lower, where  $U$  is an upper triangular matrix and  $L$  is a lower triangular, stored in RFP format. The RFP storage format is described in Section 3.3.3 in the f07 Chapter Introduction.

### 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>

Gustavson F G, Waśniewski J, Dongarra J J and Langou J (2010) Rectangular full packed format for Cholesky's algorithm: factorization, solution, and inversion *ACM Trans. Math. Software* **37**, 2

### 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: **transr** – Nag\_RFP\_Store *Input*  
*On entry:* specifies whether the RFP representation of  $A$  is normal or transposed.  
**transr** = Nag\_RFP\_Normal  
 The matrix  $A$  is stored in normal RFP format.  
**transr** = Nag\_RFP\_Trans  
 The matrix  $A$  is stored in transposed RFP format.  
*Constraint:* **transr** = Nag\_RFP\_Normal or Nag\_RFP\_Trans.

- 3: **uplo** – Nag\_UploType *Input*  
*On entry:* specifies whether the upper or lower triangular part of  $A$  is stored.  
**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  $LL^T$ , where  $L$  is lower triangular.  
*Constraint:* **uplo** = Nag\_Upper or Nag\_Lower.
- 4: **n** – Integer *Input*  
*On entry:*  $n$ , the order of the matrix  $A$ .  
*Constraint:*  $n \geq 0$ .
- 5: **ar**[ $n \times (n + 1)/2$ ] – double *Input/Output*  
*On entry:* the upper or lower triangular part (as specified by **uplo**) of the  $n$  by  $n$  symmetric matrix  $A$ , in either normal or transposed RFP format (as specified by **transr**). The storage format is described in detail in Section 3.3.3 in the f07 Chapter Introduction.  
*On exit:* if **fail.code** = NE\_NOERROR, the factor  $U$  or  $L$  from the Cholesky factorization  $A = U^T U$  or  $A = LL^T$ , in the same storage format as  $A$ .
- 6: **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,  $n = \langle value \rangle$ .

Constraint:  $n \geq 0$ .

### 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.

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\_MAT\_NOT\_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 matrix stored in

RFP format which is not positive definite; the matrix must be treated as a full symmetric matrix, by calling nag\_dsytrf (f07mdc).

## 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.

## 7 Accuracy

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

$$|E| \leq c(n)\epsilon|U^T||U|,$$

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

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

## 8 Parallelism and Performance

nag\_dpftf (f07wdc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag\_dpftf (f07wdc) 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  $\frac{1}{3}n^3$ .

A call to nag\_dpftf (f07wdc) may be followed by calls to the functions:

nag\_dpftfs (f07wec) to solve  $AX = B$ ;

nag\_dpftri (f07wjc) to compute the inverse of  $A$ .

The complex analogue of this function is nag\_zpftf (f07wrc).

## 10 Example

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

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

and is stored using RFP format.

### 10.1 Program Text

```
/* nag_dpftf (f07wdc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
```

```

*/

#include <nag.h>
#include <nag_stdlib.h>
#include <nagf01.h>
#include <nagf07.h>
#include <nagx04.h>

int main(void)
{
    /* Scalars */
    Integer exit_status = 0;
    Integer i, j, k, lar1, lar2, lenar, n, pdar, pda, q;
    /* Arrays */
    double *ar = 0, *a = 0;
    char nag_enum_arg[40];
    /* NAG types */
    Nag_RFP_Store transr;
    Nag_UploType uplo;
    Nag_OrderType order;
    Nag_MatrixType matrix;
    NagError fail;

#ifdef NAG_COLUMN_MAJOR
    order = Nag_ColMajor;
#define AR(I,J) ar[J*pdar + I]
#else
    order = Nag_RowMajor;
#define AR(I,J) ar[I*pdar + J]
#endif

    INIT_FAIL(fail);

    printf("nag_dpftf (f07wdc) Example Program Results\n");
    /* Skip heading in data file */
#ifdef _WIN32
    scanf_s("%*[\n] ");
#else
    scanf("%*[\n] ");
#endif
#ifdef _WIN32
    scanf_s("%" NAG_IFMT " ", &n);
#else
    scanf("%" NAG_IFMT " ", &n);
#endif
#ifdef _WIN32
    scanf_s("%39s", nag_enum_arg, (unsigned)_countof(nag_enum_arg));
#else
    scanf("%39s", nag_enum_arg);
#endif
    uplo = (Nag_UploType) nag_enum_name_to_value(nag_enum_arg);
#ifdef _WIN32
    scanf_s("%39s%*[\n]", nag_enum_arg, (unsigned)_countof(nag_enum_arg));
#else
    scanf("%39s%*[\n]", nag_enum_arg);
#endif
    transr = (Nag_RFP_Store) nag_enum_name_to_value(nag_enum_arg);

    lenar = (n * (n + 1)) / 2;
    pda = n;
    if (!(ar = NAG_ALLOC(lenar, double)) || !(a = NAG_ALLOC(pda * n, double)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    /* Setup dimensions for RFP array ar. */
    k = n / 2;
    q = n - k;
    if (transr == Nag_RFP_Normal) {

```

```

        lar1 = 2 * k + 1;
        lar2 = q;
    }
    else {
        lar1 = q;
        lar2 = 2 * k + 1;
    }
    if (order == Nag_RowMajor) {
        pdar = lar2;
    }
    else {
        pdar = lar1;
    }
    /* Read matrix into RFP array ar. */
    for (i = 0; i < lar1; i++) {
        for (j = 0; j < lar2; j++) {
#ifdef _WIN32
            scanf_s("%lf ", &AR(i, j));
#else
            scanf("%lf ", &AR(i, j));
#endif
        }
    }

    /* Factorize A using nag_dpfttrf (f07wdc) which performs a
     * Cholesky factorization of real symmetric positive definite matrix in
     * Rectangular Full Packed format.
     */
    nag_dpfttrf(order, transr, uplo, n, ar, &fail);
    if (fail.code != NE_NOERROR) {
        printf("Error from nag_dpfttrf (f07wdc)\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }
    printf("\n");

    /* Convert factor to full array form for printing using nag_dtftrr (f01vgc).
     * Copies a real triangular matrix from Rectangular Full Packed format
     * to full matrix format
     */
    nag_dtftrr(order, transr, uplo, n, ar, a, pda, &fail);
    if (fail.code != NE_NOERROR) {
        printf("Error from nag_dtftrr (f01vgc)\n%s\n", fail.message);
        exit_status = 2;
        goto END;
    }

    matrix = (uplo == Nag_Lower ? Nag_LowerMatrix : Nag_UpperMatrix);
    /* 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,
                          "Factor", 0, &fail);
    if (fail.code != NE_NOERROR) {
        printf("Error from nag_gen_real_mat_print (x04cac)\n%s\n", fail.message);
        exit_status = 3;
    }
}

END:
NAG_FREE(ar);
NAG_FREE(a);
return exit_status;
}

```

## 10.2 Program Data

```
nag_dpftf (f07wdc) Example Program Data
  4      Nag_Lower   Nag_RFP_Normal   : n, uplo, transr

    0.76    0.34
    4.16    1.18
   -3.12    5.03
    0.56   -0.83
   -0.10    1.18                                : matrix A in RFP, ar[]
```

## 10.3 Program Results

```
nag_dpftf (f07wdc) 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

---