NAG Library Function Document

nag_dtftri (f07wkc)

1 Purpose

nag_dtftri (f07wkc) computes the inverse of a real triangular matrix stored in Rectangular Full Packed (RFP) format.

2 Specification

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

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

3 Description

nag_dtftri (f07wkc) forms the inverse of a real triangular matrix $A$, stored using RFP format. The RFP storage format is described in Section 3.3.3 in the f07 Chapter Introduction. Note that the inverse of an upper (lower) triangular matrix is also upper (lower) triangular.

4 References


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: 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: \( \text{uplo} \) – Nag_UploType

\textit{Input}

\textit{On entry:} specifies whether \( A \) is upper or lower triangular.

\( \text{uplo} = \text{NagUpper} \)

\( A \) is upper triangular.

\( \text{uplo} = \text{NagLower} \)

\( A \) is lower triangular.

\textit{Constraint:} \( \text{uplo} = \text{NagUpper} \) or \( \text{NagLower} \).

4: \( \text{diag} \) – Nag_DiagType

\textit{Input}

\textit{On entry:} indicates whether \( A \) is a nonunit or unit triangular matrix.

\( \text{diag} = \text{NagNonUnitDiag} \)

\( A \) is a nonunit triangular matrix.

\( \text{diag} = \text{NagUnitDiag} \)

\( A \) is a unit triangular matrix; the diagonal elements are not referenced and are assumed to be 1.

\textit{Constraint:} \( \text{diag} = \text{NagNonUnitDiag} \) or \( \text{NagUnitDiag} \).

5: \( \text{n} \) – Integer

\textit{Input}

\textit{On entry:} \( n \), the order of the matrix \( A \).

\textit{Constraint:} \( n \geq 0 \).

6: \( \text{ar}[\text{n} \times (\text{n} + 1)/2] \) – double

\textit{Input/Output}

\textit{On entry:} the upper or lower triangular part (as specified by \( \text{uplo} \)) of the \( n \) by \( n \) symmetric matrix \( A \), in either normal or transposed RFP format (as specified by \( \text{transr} \)). The storage format is described in detail in Section 3.3.3 in the \textit{f07} Chapter Introduction.

\textit{On exit:} \( A \) is overwritten by \( A^{-1} \), in the same storage format as \( A \).

7: \( \text{fail} \) – NagError *

\textit{Input/Output}

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

6 \quad \textbf{Error Indicators and Warnings}

\textbf{NE_ALLOC_FAIL}

Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

\textbf{NE_BAD_PARAM}

On entry, argument \( \langle \text{value} \rangle \) had an illegal value.

\textbf{NE_INT}

On entry, \( n = \langle \text{value} \rangle \).
Constraint: \( n \geq 0 \).

\textbf{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.
7 Accuracy

The computed inverse $X$ satisfies

$$|XA - I| \leq c(n)\epsilon|X||A|,$$

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

Note that a similar bound for $|AX - I|$ cannot be guaranteed, although it is almost always satisfied.

The computed inverse satisfies the forward error bound

$$|X - A^{-1}| \leq c(n)\epsilon|A^{-1}||A||X|.$$


8 Parallelism and Performance

`nag_dtftri (f07wkc)` is not threaded by NAG in any implementation.

`nag_dtftri (f07wkc)` 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 $\frac{1}{3}n^3$.

The complex analogue of this function is `nag_ztftri (f07wxc)`.

10 Example

This example computes the inverse of the matrix $A$, where

$$A = \begin{pmatrix}
4.30 & 0.00 & 0.00 & 0.00 \\
-3.96 & -4.87 & 0.00 & 0.00 \\
0.40 & 0.31 & -8.02 & 0.00 \\
-0.27 & 0.07 & -5.95 & 0.12 \\
\end{pmatrix}$$

and is stored using RFP format.

10.1 Program Text

```c
/* nag_dtftri (f07wkc) Example Program. */
/* Copyright 2014 Numerical Algorithms Group. */
/* Mark 23, 2011. */
*/
#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_DiagType diag;
    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_dtftri (f07wkc) 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);
    #else
    scanf("%"NAG_IFMT", &n);
    #endif
    #ifdef _WIN32
    scanf_s("%39s", nag_enum_arg, _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("%"NAG_IFMT", &n);
    #else
    scanf("%"NAG_IFMT", &n);
    #endif
    #ifdef _WIN32
    scanf_s("%39s", nag_enum_arg, _countof(nag_enum_arg));
    #else
    scanf("%39s", nag_enum_arg);
    #endif
    diag = (Nag_DiagType) nag_enum_name_to_value(nag_enum_arg);
    lenar = (n * (n + 1))/2;
    if (!(ar = NAG_ALLOC((lenar), double)) ||
        !(a = NAGALLOC((n)*(n), double)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
    pda = n;

    f07wkc
/* Setup dimensions of 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 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
  }
}
/* Compute inverse of A using nag_dtftri (f07wkc) which Inverts a real *
* triangular matrix in Rectangular Full Packed format. */
nag_dtftri(order, transr, uplo, diag, n, ar, &fail);
if (fail.code != NE_NOERROR) {
  printf("%s
", fail.message);
  exit_status = 1;
  goto END;
}
/* Convert inverse to full array form using nag_dtfttr (f01vgc). */
nag_dtfttr(order, transr, uplo, n, ar, a, pdar, &fail);
if (fail.code != NE_NOERROR) {
  printf("%s
", fail.message);
  exit_status = 2;
  goto END;
}
/* nag_gen_real_mat_print (x04cac).
* Print real general matrix (easy-to-use) */
if (uplo == Nag_Lower)
  matrix = Nag_LowerMatrix;
else
  matrix = Nag_UpperMatrix;
nag_gen_real_mat_print(order, matrix, diag, n, n, a, pdar, "Inverse", 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_dtftri (f07wkc) Example Program Data
4 Nag_Lower
Nag_RFP_Normal
Nag_NonUnitDiag : n, uplo, transr, diag
-8.02 -5.95
4.30 0.12
-3.96 -4.87
0.40 0.31
-0.27 0.07 : ar[]

10.3 Program Results

nag_dtftri (f07wkc) Example Program Results

Inverse          1  2  3  4
1     0.2326
2    -0.1891 -0.2053
3     0.0043 -0.0079 -0.1247
4     0.8463 -0.2738 -6.1825  8.3333