nag_dsytri (f07mjc) computes the inverse of a real symmetric indefinite matrix $A$, where $A$ has been factorized by nag_dsytrf (f07mdc).

#include <nag.h>
#include <nagf07.h>
void nag_dsytri (Nag_OrderType order, Nag_UploType uplo, Integer n,
                 double a[], Integer pda, const Integer ipiv[], NagError *fail)

nag_dsytri (f07mjc) is used to compute the inverse of a real symmetric indefinite matrix $A$, the function must be preceded by a call to nag_dsytrf (f07mdc), which computes the Bunch–Kaufman factorization of $A$.

If $\text{uplo} = \text{Nag\_Upper}$, $A = \text{PUDU}^T\text{PT}$ and $A^{-1}$ is computed by solving $U^T\text{PTXPU} = D^{-1}$ for $X$.
If $\text{uplo} = \text{Nag\_Lower}$, $A = \text{PLDL}^T\text{PT}$ and $A^{-1}$ is computed by solving $L^T\text{PTXPL} = D^{-1}$ for $X$.


5 Arguments

1: order – Nag_OrderType

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

On entry: specifies how $A$ has been factorized.

uplo = Nag_Upper
$A = \text{PUDU}^T\text{PT}$, where $U$ is upper triangular.
uplo = Nag_Lower
$A = \text{PLDL}^T\text{PT}$, where $L$ is lower triangular.

Constraint: uplo = Nag_Upper or Nag_Lower.

3: n – Integer

On entry: $n$, the order of the matrix $A$.

Constraint: $n \geq 0$. 

4: \texttt{a[dim]} – double \hspace{1cm} \textit{Input/Output}

\textbf{Note:} the dimension, \texttt{dim}, of the array \texttt{a} must be at least \texttt{max(1, pda \times n)}.

\textit{On entry:} details of the factorization of \(A\), as returned by \texttt{nag_dsytrf (f07mdc)}.

\textit{On exit:} the factorization is overwritten by the \(n \times n\) symmetric matrix \(A^{-1}\).

If \texttt{uplo} = \texttt{NagUpper}, the upper triangle of \(A^{-1}\) is stored in the upper triangular part of the array.

If \texttt{uplo} = \texttt{NagLower}, the lower triangle of \(A^{-1}\) is stored in the lower triangular part of the array.

5: \texttt{pda} – Integer \hspace{1cm} \textit{Input}

\textit{On entry:} the stride separating row or column elements (depending on the value of \texttt{order}) of the matrix in the array \texttt{a}.

\textbf{Constraint:} \texttt{pda} \geq \texttt{max(1, n)}.

6: \texttt{ipiv[dim]} – const Integer \hspace{1cm} \textit{Input}

\textbf{Note:} the dimension, \texttt{dim}, of the array \texttt{ipiv} must be at least \texttt{max(1, n)}.

\textit{On entry:} details of the interchanges and the block structure of \(D\), as returned by \texttt{nag_dsytrf (f07mdc)}.

7: \texttt{fail} – \texttt{NagError*} \hspace{1cm} \textit{Input/Output}

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

6 \hspace{0.5cm} \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 value\rangle had an illegal value.

\textbf{NE_INT}

\textit{On entry,} \texttt{n} = \langle value\rangle.

\textbf{Constraint:} \texttt{n} \geq 0.

\textit{On entry,} \texttt{pda} = \langle value\rangle.

\textbf{Constraint:} \texttt{pda} > 0.

\textbf{NE_INT_2}

\textit{On entry,} \texttt{pda} = \langle value\rangle and \texttt{n} = \langle value\rangle.

\textbf{Constraint:} \texttt{pda} \geq \texttt{max(1, n)}.

\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.

\textbf{NE_NO_LICENCE}

Your licence key may have expired or may not have been installed correctly. See Section 3.6.5 in the Essential Introduction for further information.
Element \(\text{value} \) of the diagonal is exactly zero. \( D \) is singular and the inverse of \( A \) cannot be computed.

7 Accuracy

The computed inverse \( X \) satisfies a bound of the form

\[
|DU^TP^TXPU - I| \leq c(n)\epsilon(|D||U^TP^TPX||P|U| + |D||D^{-1}|);
\]

\[
|DL^TP^TXPL - I| \leq c(n)\epsilon(|D||L^TP^TPX||P|L| + |D||D^{-1}|),
\]

\( c(n) \) is a modest linear function of \( n \), and \( \epsilon \) is the \textbf{machine precision}.

8 Parallelism and Performance

\nmag_dsymtri (f07mjc) is not threaded by NAG in any implementation.

\nmag_dsymtri (f07mjc) 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 \( \frac{3}{2}n^3 \).

The complex analogues of this function are \texttt{mag_dsytri} (f07mwc) for Hermitian matrices and \texttt{mag_zsytri} (f07nwc) for symmetric matrices.

10 Example

This example computes the inverse of the matrix \( A \), where

\[
A = \begin{pmatrix}
  2.07 & 3.87 & 4.20 & -1.15 \\
  3.87 & -0.21 & 1.87 & 0.63 \\
  4.20 & 1.87 & 1.15 & 2.06 \\
 -1.15 & 0.63 & 2.06 & -1.81
\end{pmatrix}.
\]

Here \( A \) is symmetric indefinite and must first be factorized by \texttt{mag_dsyrfd} (f07mdc).

10.1 Program Text

\/* mag_dsymtri (f07mjc) Example Program.  
*  Copyright 2014 Numerical Algorithms Group.  
*/
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf07.h>
#include <nagxm04.h>

int main(void)
{
  /* Scalars */
  Integer i, j, n, pda;
  Integer exit_status = 0;

  // Code to compute the inverse of A

  // Example code
  // ...

}
Nag_UploType uplo;
Nag_MatrixType matrix;
NagError fail;
Nag_OrderType order;

/* Arrays */
char nag_enum_arg[40];
Integer *ipiv = 0;
double *a = 0;
#endif /* 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 */

#define A(I, J) a[(J-1)*pda + I-1]
#endif
#define A(I, J) a[(I-1)*pda + J-1]
#endif

INIT_FAIL(fail);

printf("nag_dsytri (f07mjc) Example Program Results\n\n");
/* Skip heading in data file */
#endif
else
scanf("%*[\n] ");
#endif
#endif
#define NAG_COLUMN_MAJOR
#define A(I, J) a[(J-1)*pda + I - 1]
order = Nag_ColMajor;
#else
#define A(I, J) a[(I-1)*pda + J - 1]
order = Nag_RowMajor;
#endif

INIT_FAIL(fail);

printf("nag_dsytri (f07mjc) Example Program Results\n\n");
/* Skip heading in data file */
#endif
else
scanf("%*[\n] ");
#endif
#endif
#define NAG_COLUMN_MAJOR
#define A(I, J) a[(J-1)*pda + I - 1]
order = Nag_ColMajor;
#else
#define A(I, J) a[(I-1)*pda + J - 1]
order = Nag_RowMajor;
#endif

/* Allocate memory */
if (!(ipiv = NAG_ALLOC(n, Integer)) || !(a = NAG_ALLOC(n * n, double)) )
{
printf("Allocation failure\n");
exit_status = -1;
goto END;
}

/* Read A from data file */
#endif
else
scanf(" %39s%*[\n] ", nag_enum_arg, _countof(nag_enum_arg));
#endif
#endif

uplo = (Nag_UploType) nag_enum_name_to_value(nag_enum_arg);
if (uplo == Nag_Upper)
{
matrix = Nag_UpperMatrix;
for (i = 1; i <= n; ++i)
{
for (j = i; j <= n; ++j)
#endif
else
scanf("%lf", &A(i, j));
#endif
```c
scanf("%lf", &A(i, j));
#endif
#endif
scanf_s("%*[\n ] ");
#else
scanf("%*[\n ] ");
#endif
else
{
    matrix = Nag_LowerMatrix;
    for (i = 1; i <= n; ++i)
    {
        for (j = 1; j <= i; ++j)
            scanf("%lf", &A(i, j));
    }
#endif
#if defined _WIN32
    scanf_s("%*[\n ] ");
#else
    scanf("%*[\n ] ");
#endif
}
#else
    scanf("%*[\n ] ");
#endif
/* Factorize A */
/* nag_dsytrf (f07mdc). */
/* Bunch-Kaufman factorization of real symmetric indefinite */
/* matrix */
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dsytrf (f07mdc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Compute inverse of A */
/* nag_dsytri (f07mjc). */
/* Inverse of real symmetric indefinite matrix, matrix */
/* already factorized by nag_dsytrf (f07mdc) */
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dsytri (f07mjc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Print inverse */
/* nag_gen_real_mat_print (x04cac). */
/* Print real general matrix (easy-to-use) */
fflush(stdout);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_gen_real_mat_print (x04cac).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
END:
NAG_FREE(ipiv);
NAG_FREE(a);
return exit_status;
```
10.2 Program Data

nag_dsytri (f07mjc) Example Program Data
4 :Value of n
Nag_Lower :Value of uplo
2.07
3.87 -0.21
4.20 1.87 1.15
-1.15 0.63 2.06 -1.81 :End of matrix A

10.3 Program Results

nag_dsytri (f07mjc) Example Program Results

inverse

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.7485</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.5221</td>
<td>-0.1605</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-1.0058</td>
<td>-0.3131</td>
<td>1.3501</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>-1.4386</td>
<td>-0.7440</td>
<td>2.0667</td>
<td>2.4547</td>
</tr>
</tbody>
</table>