NAG C Library Function Document

nag_dsytri (f07mjc)

1 Purpose

nag_dsytri (f07mjc) computes the inverse of a real symmetric indefinite matrix $A$, where $A$ has been factorized by nag_dsyr (f07mlc).

2 Specification

```c
void nag_dsytri (Nag_OrderType order, Nag_UploType uplo, Integer n, double a[],
    Integer pda, const Integer ipiv[], NagError *fail)
```

3 Description

To compute the inverse of a real symmetric indefinite matrix $A$, this function must be preceded by a call to nag_dsyr (f07mlc), which computes the Bunch–Kaufman factorization of $A$.

If \texttt{uplo} = \texttt{Nag_Upper}, $A = PUDU^T P^T$ and $A^{-1}$ is computed by solving $U^T P^T X U = D$ for $X$.

If \texttt{uplo} = \texttt{Nag_Lower}, $A = PLDL^T P^T$ and $A^{-1}$ is computed by solving $L^T P^T X L = D$ for $X$.

4 References


5 Parameters

1: \texttt{order} – Nag_OrderType  
   \textit{Input}
   
   \textit{On entry}: the \texttt{order} parameter specifies the two-dimensional storage scheme being used, i.e., row-major ordering or column-major ordering. C language defined storage is specified by \texttt{order} = \texttt{Nag_RowMajor}. See Section 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.
   
   \textit{Constraint}: \texttt{order} = \texttt{Nag_RowMajor} or \texttt{Nag_ColMajor}.

2: \texttt{uplo} – Nag_UploType  
   \textit{Input}
   
   \textit{On entry}: indicates how $A$ has been factorized as follows:
   
   if \texttt{uplo} = \texttt{Nag_Upper}, $A = PUDU^T P^T$, where $U$ is upper triangular;
   
   if \texttt{uplo} = \texttt{Nag_Lower}, $A = PLDL^T P^T$, where $L$ is lower triangular.
   
   \textit{Constraint}: \texttt{uplo} = \texttt{Nag_Upper} or \texttt{Nag_Lower}.

3: \texttt{n} – Integer  
   \textit{Input}
   
   \textit{On entry}: $n$, the order of the matrix $A$.
   
   \textit{Constraint}: $n \geq 0$.

4: \texttt{a[\textit{dim}]} – double  
   \textit{Input/Output}
   
   \textit{Note}: the dimension, \textit{dim}, of the array \texttt{a} must be at least $\text{max}(1, pda \times n)$.
   
   \textit{On entry}: details of the factorization of $A$, as returned by nag_dsyr (f07mlc).
On exit: the factorization is overwritten by the $n$ by $n$ symmetric matrix $A^{-1}$. If
$\text{uplo} = \text{Nag\_Upper}$, the upper triangle of $A^{-1}$ is stored in the upper triangular part of the array;
if $\text{uplo} = \text{Nag\_Lower}$, the lower triangle of $A^{-1}$ is stored in the lower triangular part of the array.

5: $\text{pda}$ – Integer

$\text{Input}$

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

$\text{Constraint:}$ $\text{pda} \geq \text{max}(1, n)$.

6: $\text{ipiv}[\text{dim}]$ – const Integer

$\text{Input}$

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

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

7: $\text{fail}$ – NagError *

$\text{Output}$

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

$\text{NE\_INT}$

$\text{On entry,}$ $n = \langle\text{value}\rangle$.

$\text{Constraint:}$ $n \geq 0$.

$\text{On entry,}$ $\text{pda} = \langle\text{value}\rangle$.

$\text{Constraint:}$ $\text{pda} > 0$.

$\text{NE\_INT\_2}$

$\text{On entry,}$ $\text{pda} = \langle\text{value}\rangle$, $n = \langle\text{value}\rangle$.

$\text{Constraint:}$ $\text{pda} \geq \text{max}(1, n)$.

$\text{NE\_SINGULAR}$

The block diagonal matrix $D$ is exactly singular.

$\text{NE\_ALLOC\_FAIL}$

Memory allocation failed.

$\text{NE\_BAD\_PARAM}$

$\text{On entry,}$ parameter $\langle\text{value}\rangle$ had an illegal value.

$\text{NE\_INTERNAL\_ERROR}$

An internal error has occurred in this function. Check the function call and any array sizes. If the
call is correct then please consult NAG for assistance.

7 Accuracy

The computed inverse $X$ satisfies a bound of the form

if $\text{uplo} = \text{Nag\_Upper}$, $|DU^TXPU - I| \leq c(n)\varepsilon(|D||U^T|P^T|X|P|U| + |D||D^{-1}|)$;

if $\text{uplo} = \text{Nag\_Lower}$, $|DL^TXPL - I| \leq c(n)\varepsilon(|D||L^T|P^T|X|P|L| + |D||D^{-1}|)$,

$c(n)$ is a modest linear function of $n$, and $\varepsilon$ is the machine precision.
8 Further Comments

The total number of floating-point operations is approximately $\frac{2}{3}n^3$.

The complex analogues of this function are nag_zhetri (f07mwc) for Hermitian matrices and nag_zsytri (f07nwc) for symmetric matrices.

9 Example

To compute 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 nag_dsytrf (f07mdc).

9.1 Program Text

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

int main(void)
{
    /* Scalars */
    Integer i, j, n, pda;
    Integer exit_status=0;
    Nag_UploType uplo_enum;
    Nag_MatrixType matrix;
    NagError fail;
    Nag_OrderType order;
    /* Arrays */
    char uplo[2];
    Integer *ipiv=0;
    double *a=0;

    INIT_FAIL(fail);
    Vprintf("f07mjc Example Program Results\n\n");
    /* Skip heading in data file */
    Vscanf("%*\[^
\] ");
    Vscanf("%ld%*\[^
\] ", &n);
    if(NAG_COLUMN_MAJOR)
        pda = n;
    else
        pda = n;
    #endif
    /* Allocate memory */
    /* Compute the inverse of the matrix A */
    /* Call nag_dsytrf */
    /* Call nag_dsytri */
}
```

[NP3645/7]
if (!(ipiv = NAG_ALLOC(n, Integer)) ||
    !(a = NAG_ALLOC(n * n, double)))
{
    Vprintf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

/* Read A from data file */
Vscanf("' %ls '*[\n] ", uplo);
if (*((unsigned char *)uplo == 'L')
{
    uplo_enum = Nag_Lower;
    matrix = Nag_LowerMatrix;
}
else if (*((unsigned char *)uplo == 'U')
{
    uplo_enum = Nag_Upper;
    matrix = Nag_UpperMatrix;
}
else
{
    Vprintf("Unrecognised character for Nag_UploType type\n");
    exit_status = -1;
    goto END;
}

if (uplo_enum == Nag_Upper)
{
    for (i = 1; i <= n; ++i)
    {
        for (j = i; j <= n; ++j)
            Vscanf("%lf", &A(i,j));
    }
    Vscanf("%*[\n] ");
}
else
{
    for (i = 1; i <= n; ++i)
    {
        for (j = 1; j <= i; ++j)
            Vscanf("%lf", &A(i,j));
    }
    Vscanf("%*[\n] ");
}

/* Factorize A */
f07mdc(order, uplo_enum, n, a, pda, ipiv, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07mdc.\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Compute inverse of A */
f07mjc(order, uplo_enum, n, a, pda, ipiv, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07mjc.\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Print inverse */
x04cac(order, matrix, Nag_NonUnitDiag, n, n, a, pda,
        "Inverse", 0, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from x04cac.\n", fail.message);
    exit_status = 1;
    goto END;
}
END:
    if (ipiv) NAG_FREE(ipiv);
    if (a) NAG_FREE(a);
    return exit_status;
}

9.2 Program Data

f07mjc Example Program Data

:Value of N
4

:Value of UPLO
'L'

2.07
3.87 -0.21
4.20 1.87 1.15
-1.15 0.63 2.06 -1.81 :End of matrix A

9.3 Program Results

f07mjc Example Program Results

Inverse

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