NAG C Library Function Document

nag_dsptri (f07pjc)

1 Purpose

nag_dsptri (f07pjc) computes the inverse of a real symmetric indefinite matrix \( A \), where \( A \) has been factorized by nag_dsptrf (f07pdc), using packed storage.

2 Specification

```c
void nag_dsptri (Nag_OrderType order, Nag_UploType uplo, Integer n, double ap[],
        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_dsptrf (f07pdc), which computes the Bunch–Kaufman factorization of \( A \) using packed storage.

If \( \text{uplo} = \text{NagUpper} \), \( A = P U D U^T P^T \) and \( A^{-1} \) is computed by solving \( U^T P^T X P U = D^{-1} \).

If \( \text{uplo} = \text{NagLower} \), \( A = P L D L^T P^T \) and \( A^{-1} \) is computed by solving \( L^T P^T X P L = D^{-1} \).

4 References


5 Parameters

1: \( \text{order} \) – Nag_OrderType

\( \text{Input} \)

\( \text{On entry:} \) the \( \text{order} \) 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 \( \text{order} = \text{NagRowMajor} \). See Section 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.

\( \text{Constraint:} \ \text{order} = \text{NagRowMajor} \text{ or } \text{NagColMajor}. \)

2: \( \text{uplo} \) – Nag_UploType

\( \text{Input} \)

\( \text{On entry:} \) indicates how \( A \) has been factorized as follows:

- if \( \text{uplo} = \text{NagUpper} \), \( A = P U D U^T P^T \), where \( U \) is upper triangular;
- if \( \text{uplo} = \text{NagLower} \), \( A = P L D L^T P^T \), where \( L \) is lower triangular.

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

3: \( n \) – Integer

\( \text{Input} \)

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

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

4: \( \text{ap}[\text{dim}] \) – double

\( \text{Input/Output} \)

\( \text{Note:} \) the dimension, \( \text{dim} \), of the array \( \text{ap} \) must be at least \( \max(1,n \times (n + 1)/2) \).

\( \text{On entry:} \) details of the factorization of \( A \) stored in packed form, as returned by nag_dsptrf (f07pdc).
On exit: the factorization is overwritten by the \( n \) by \( n \) symmetric matrix \( A^{-1} \) stored in packed form.

5: \( \text{ipiv}[\text{dim}] \) – const Integer

\textbf{Input}

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

On entry: details of the interchanges and the block structure of \( D \), as returned by \text{nag_dspttrf} (f07pdc).

6: \( \text{fail} \) – NagError *

\textbf{Output}

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

\textbf{NE_INT}

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

Constraint: \( n \geq 0 \).

\textbf{NE_SINGULAR}

The block diagonal matrix \( D \) is exactly singular.

\textbf{NE_ALLOC_FAIL}

Memory allocation failed.

\textbf{NE_BAD_PARAM}

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

\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 consult NAG for assistance.

7 Accuracy

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

\[
|DU^{-1}P^{-1}XPU - I| \leq c(n)\epsilon(|D||U^{-T}|P^{-1}X|P|U| + |D||D^{-1}|);
\]

\[
|DL^{-1}P^{-1}XPL - I| \leq c(n)\epsilon(|D||L^{-T}|P^{-1}X|P|L| + |D||D^{-1}|),
\]

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

8 Further Comments

The total number of floating-point operations is approximately \( \frac{1}{3}n^3 \).

The complex analogues of this function are \text{nag_zhptri} (f07pwc) for Hermitian matrices and \text{nag_zsptri} (f07qwc) 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, stored in packed form, and must first be factorized by nag_dsptrf (f07pdc).

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 ap_len, i, j, n;
    Integer exit_status=0;
    NagError fail;
    Nag_UploType uplo_enum;
    Nag_OrderType order;

    /* Arrays */
    Integer *ipiv=0;
    char uplo[2];
    double *ap=0;

    INIT_FAIL(fail);
    Vprintf("f07pjc Example Program Results\n\n");
    /* Skip heading in data file */
    Vscanf("%\[\n\]");
    Vscanf("%\[\ss%\[\n\] %", &n);
    ap_len = n * (n + 1)/2;

    /* Allocate memory */
    if ( !(ipiv = NAG_ALLOC(n, Integer)) ||
        !(ap = NAG_ALLOC(n * (n + 1)/2, double)) )
    {
        Vprintf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    INIT_FAIL(fail);
    Vprintf("f07pjc Example Program Results\n\n");
    /* Skip heading in data file */
    Vscanf("%\[\n\]");
    Vscanf("%\[\ss%\[\n\] %", &n);
    ap_len = n * (n + 1)/2;

    /* Allocate memory */
    if ( !(ipiv = NAG_ALLOC(n, Integer)) ||
        !(ap = NAG_ALLOC(n * (n + 1)/2, double)) )
    {
        Vprintf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    /* Read A from data file */
    Vscanf(" ' %1s ' '%\[\n\] %", uplo);
```

[NP3645/7]
if (*(unsigned char *)uplo == 'L')
    uplo_enum = Nag_Lower;
else if (*(unsigned char *)uplo == 'U')
    uplo_enum = Nag_Upper;
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_UPPER(i,j));
            }
    }
else
    { 
        for (i = 1; i <= n; ++i) 
            { 
                for (j = 1; j <= i; ++j) 
                    Vscanf("%lf", &A_LOWER(i,j));
            }
    }
/* Factorize A */
f07pdc(order, uplo_enum, n, ap, ipiv, &fail);
if (fail.code != NE_NOERROR)
    { 
        Vprintf("Error from f07pdc.\n", fail.message);
        exit_status = 1;
        goto END;
    }
/* Compute inverse of A */
f07pjc(order, uplo_enum, n, ap, ipiv, &fail);
if (fail.code != NE_NOERROR)
    { 
        Vprintf("Error from f07pjc.\n", fail.message);
        exit_status = 1;
        goto END;
    }
/* Print inverse */
x04ccc(order, uplo_enum, Nag_NonUnitDiag, n, ap,
        "Inverse", 0, NAGERR_DEFAULT);
if (fail.code != NE_NOERROR)
    { 
        Vprintf("Error from x04ccc.\n", fail.message);
        exit_status = 1;
        goto END;
    }
END:
if (ipiv) NAG_FREE(ipiv);
if (ap) NAG_FREE(ap);
return exit_status;

9.2 Program Data
f07pjc Example Program Data
4 :Value of N
  'L' :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
## 9.3 Program Results

f07pjc Example Program Results

<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>