

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_dsptf (f07pdc), using packed storage.

2 Specification

```
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_dsptf (f07pdc), which computes the Bunch–Kaufman factorization of A using packed storage.

If **uplo** = **Nag_Upper**, $A = PUDU^T P^T$ and A^{-1} is computed by solving $U^T P^T XPU = D^{-1}$.

If **uplo** = **Nag_Lower**, $A = PLDL^T P^T$ and A^{-1} is computed by solving $L^T P^T XPL = D^{-1}$.

4 References

Du Croz J J and Higham N J (1992) Stability of methods for matrix inversion *IMA J. Numer. Anal.* **12** 1–19

5 Parameters

- 1: **order** – Nag_OrderType *Input*
On entry: the **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 **order** = **Nag_RowMajor**. See Section 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.
Constraint: **order** = **Nag_RowMajor** or **Nag_ColMajor**.
- 2: **uplo** – Nag_UploType *Input*
On entry: indicates how A has been factorized as follows:
 if **uplo** = **Nag_Upper**, $A = PUDU^T P^T$, where U is upper triangular;
 if **uplo** = **Nag_Lower**, $A = PLDL^T P^T$, where L is lower triangular.
Constraint: **uplo** = **Nag_Upper** or **Nag_Lower**.
- 3: **n** – Integer *Input*
On entry: n , the order of the matrix A .
Constraint: $n \geq 0$.
- 4: **ap**[*dim*] – double *Input/Output*
Note: the dimension, *dim*, of the array **ap** must be at least $\max(1, n \times (n + 1)/2)$.
On entry: details of the factorization of A stored in packed form, as returned by nag_dsptf (f07pdc).

On exit: the factorization is overwritten by the n by n symmetric matrix A^{-1} stored in packed form.

5: **ipiv**[*dim*] – const Integer *Input*

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

On entry: details of the interchanges and the block structure of D , as returned by nag_dsptf (f07pdc).

6: **fail** – NagError * *Output*

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

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

NE_SINGULAR

The block diagonal matrix D is exactly singular.

NE_ALLOC_FAIL

Memory allocation failed.

NE_BAD_PARAM

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

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 **uplo** = **Nag_Upper**, $|DU^T P^T XPU - I| \leq c(n)\epsilon(|D||U^T|P^T|X|P|U| + |D||D^{-1}|)$;

if **uplo** = **Nag_Lower**, $|DL^T P^T XPL - I| \leq c(n)\epsilon(|D||L^T|P^T|X|P|L| + |D||D^{-1}|)$,

$c(n)$ is a modest linear function of n , and ϵ 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_zhptri (f07pwc) for Hermitian matrices and 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

```
/* nag_dsptri (f07pjc) Example Program.
 *
 * Copyright 2001 Numerical Algorithms Group.
 *
 * Mark 7, 2001.
 */

#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;

#ifdef NAG_COLUMN_MAJOR
#define A_UPPER(I,J) ap[J*(J-1)/2 + I - 1]
#define A_LOWER(I,J) ap[(2*n-J)*(J-1)/2 + I - 1]
    order = Nag_ColMajor;
#else
#define A_LOWER(I,J) ap[I*(I-1)/2 + J - 1]
#define A_UPPER(I,J) ap[(2*n-I)*(I-1)/2 + J - 1]
    order = Nag_RowMajor;
#endif

    INIT_FAIL(fail);
    Vprintf("f07pjc Example Program Results\n\n");

    /* Skip heading in data file */
    Vscanf("%*[^\\n] ");
    Vscanf("%ld%*[^\\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);
```

```

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

/* Factorize A */
f07pdc(order, uplo_enum, n, ap, ipiv, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f07pdc.\n%s\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%s\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%s\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

Inverse	1	2	3	4
1	0.7485			
2	0.5221	-0.1605		
3	-1.0058	-0.3131	1.3501	
4	-1.4386	-0.7440	2.0667	2.4547
