NAG C Library Function Document
nag_dsptrf (f07pdc)

1 Purpose
nag_dsptrf (f07pdc) computes the Bunch–Kaufman factorization of a real symmetric indefinite matrix, using packed storage.

2 Specification
void nag_dsptrf (Nag_OrderType order, Nag_UploType uplo, Integer n, double ap[], Integer ipiv[], NagError *fail)

3 Description
nag_dsptrf (f07pdc) factorizes a real symmetric matrix $A$, using the Bunch–Kaufman diagonal pivoting method and packed storage. $A$ is factorized as either $A = PUDU^T P^T$ if $uplo = \text{Nag\_Upper}$, or $A = PLDL^T P^T$ if $uplo = \text{Nag\_Lower}$, where $P$ is a permutation matrix, $U$ (or $L$) is a unit upper (or lower) triangular matrix and $D$ is a symmetric block diagonal matrix with 1 by 1 and 2 by 2 diagonal blocks; $U$ (or $L$) has 2 by 2 unit diagonal blocks corresponding to the 2 by 2 blocks of $D$. Row and column interchanges are performed to ensure numerical stability while preserving symmetry.

This method is suitable for symmetric matrices which are not known to be positive-definite. If $A$ is in fact positive-definite, no interchanges are performed and no 2 by 2 blocks occur in $D$.

4 References

5 Parameters
1: $\text{order} - \text{Nag\_OrderType}$
   $\text{Input}$
   
   $\text{On entry:}$ 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 $\text{order} = \text{Nag\_RowMajor}$. See Section 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.
   
   $\text{Constraint:} \text{order} = \text{Nag\_RowMajor}$ or $\text{Nag\_ColMajor}$.

2: $\text{uplo} - \text{Nag\_UploType}$
   $\text{Input}$
   
   $\text{On entry:}$ indicates whether the upper or lower triangular part of $A$ is stored and how $A$ is to be factorized, as follows:
   
   if $\text{uplo} = \text{Nag\_Upper}$, the upper triangular part of $A$ is stored and $A$ is factorized as $PUDU^T P^T$, where $U$ is upper triangular;

   if $\text{uplo} = \text{Nag\_Lower}$, the lower triangular part of $A$ is stored and $A$ is factorized as $PLDL^T P^T$, where $L$ is lower triangular.
   
   $\text{Constraint:} \text{uplo} = \text{Nag\_Upper}$ or $\text{Nag\_Lower}$.

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

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

**Input/Output**

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

**On entry:** the symmetric indefinite matrix \( A \), packed by rows or columns. The storage of elements \( a_{ij} \) depends on the \textit{order} and \textit{uplo} parameters as follows:

- If \( \text{order} = \text{Nag}_\text{ColMajor} \) and \( \text{uplo} = \text{Nag}_\text{Upper} \),
  \( a_{ij} \) is stored in \( \text{ap}[j - 1] \times j/2 + i - 1] \), for \( i \leq j \);
- If \( \text{order} = \text{Nag}_\text{ColMajor} \) and \( \text{uplo} = \text{Nag}_\text{Lower} \),
  \( a_{ij} \) is stored in \( \text{ap}[(2n - j) \times (j - 1)/2 + i - 1] \), for \( i \geq j \);
- If \( \text{order} = \text{Nag}_\text{RowMajor} \) and \( \text{uplo} = \text{Nag}_\text{Upper} \),
  \( a_{ij} \) is stored in \( \text{ap}[(i - 1) \times i/2 + j - 1] \), for \( i \leq j \);
- If \( \text{order} = \text{Nag}_\text{RowMajor} \) and \( \text{uplo} = \text{Nag}_\text{Lower} \),
  \( a_{ij} \) is stored in \( \text{ap}[(i - 1) \times i/2 + j - 1] \), for \( i \geq j \).

**On exit:** \( A \) is overwritten by details of the block diagonal matrix \( D \) and the multipliers used to obtain the factor \( U \) or \( L \) as specified by \textit{uplo}.

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

**Output**

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

**On exit:** details of the interchanges and the block structure of \( D \). More precisely, if \( \text{ipiv}[i - 1] = k > 0 \), \( d_{ii} \) is a 1 by 1 pivot block and the \( i \)th row and column of \( A \) were interchanged with the \( k \)th row and column.

If \( \text{uplo} = \text{Nag}_\text{Upper} \) and \( \text{ipiv}[i - 2] = \text{ipiv}[i - 1] = -l < 0 \), \( \begin{pmatrix} d_{i-1,i-1} & d_{i,i-1} \\ d_{i,i-1} & d_{ii} \end{pmatrix} \) is a 2 by 2 pivot block and the \((i - 1)\)th row and column of \( A \) were interchanged with the \( l \)th row and column.

If \( \text{uplo} = \text{Nag}_\text{Lower} \) and \( \text{ipiv}[i - 1] = \text{ipiv}[i] = -m < 0 \), \( \begin{pmatrix} d_{ii} & d_{i+1,i} \\ d_{i+1,i} & d_{i+1,i+1} \end{pmatrix} \) is a 2 by 2 pivot block and the \((i + 1)\)th row and column of \( A \) were interchanged with the \( m \)th row and column.

6: \( \text{fail} \) – NagError *

**Output**

The NAG error parameter (see the Essential Introduction).

### 6 Error Indicators and Warnings

**NE_INT**

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

Constraint: \( n \geq 0 \).

**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
If $\text{uplo} = \text{Nag\_Upper}$, the computed factors $U$ and $D$ are the exact factors of a perturbed matrix $A + E$, where

$$|E| \leq c(n)\epsilon|U||D||U^T|P^T,$$

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

If $\text{uplo} = \text{Nag\_Lower}$, a similar statement holds for the computed factors $L$ and $D$.

8 Further Comments
The elements of $D$ overwrite the corresponding elements of $A$; if $D$ has 2 by 2 blocks, only the upper or lower triangle is stored, as specified by $\text{uplo}$.

The unit diagonal elements of $U$ or $L$ and the 2 by 2 unit diagonal blocks are not stored. The remaining elements of $U$ or $L$ overwrite elements in the corresponding columns of $A$, but additional row interchanges must be applied to recover $U$ or $L$ explicitly (this is seldom necessary). If $\text{ipiv}[i - 1] = i$, for $i = 1, 2, \ldots, n$ (as is the case when $A$ is positive-definite), then $U$ or $L$ are stored explicitly in packed form (except for their unit diagonal elements which are equal to 1).

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

A call to this function may be followed by calls to the functions:

- nag_dsptrs (f07pec) to solve $AX = B$;
- nag_ds spcon (f07pgc) to estimate the condition number of $A$;
- nag_dsptri (f07pjc) to compute the inverse of $A$.

The complex analogues of this function are nag_zhptrf (f07prc) for Hermitian matrices and nag_zsp trf (f07qrc) for symmetric matrices.

9 Example
To compute the Bunch–Kaufman factorization of the matrix $A$, where

$$A = \begin{pmatrix} 2.07 & 3.87 & 4.20 & -1.15 \\ 3.87 & -0.21 & 1.87 & 0.63 \\ 4.20 & 1.87 & 1.15 & 2.06 \\ -1.15 & 0.63 & 2.06 & -1.81 \end{pmatrix},$$

using packed storage.

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;
    Nag_UploType uplo_enum;
```

[NP3645/7]
NagError fail;
Nag_OrderType order;

/* Arrays */
char uplo[2];
Integer *ipiv=0;
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]
#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]
#endif

INIT_FAIL(fail);
Vprintf("f07pdc 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(ap_len, 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;
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%ls\n", fail.message);
    exit_status = 1;
}
goto END;
} /* Print factor */
x04ccc(order, uplo_enum, Nag_NonUnitDiag, n, ap,
"Factor", 0, NAGERR_DEFAULT);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from x04ccc.\n\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Print pivot indices */
Vprintf("\nIPIV\n");
for (i = 1; i <= n; ++i)
    Vprintf("%6ld%s", ipiv[i-1], i%7==0 ?"\n":" ");
Vprintf("\n");
END:
if (ap) NAG_FREE(ap);
if (ipiv) NAG_FREE(ipiv);
return exit_status;

9.2 Program Data

f07pdc Example Program Data

4 :Value of N
'U' :Value of UPLO
2.07 3.87 4.20 -1.15 :Value of matrix A
-0.21 1.87 0.63
1.15 2.06
-1.81 :End of matrix A

9.3 Program Results

f07pdc Example Program Results

<table>
<thead>
<tr>
<th>Factor</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.0700</td>
<td>4.2000</td>
<td>0.2230</td>
<td>0.6537</td>
</tr>
<tr>
<td>2</td>
<td>1.1500</td>
<td>0.8115</td>
<td>-0.5960</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-2.5907</td>
<td>0.3031</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.4074</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IPIV</th>
<th>-3</th>
<th>-3</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
</table>