1 Purpose

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

2 Specification

#include <nag.h>
#include <nagf07.h>

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 $uplo = Nag\_Upper$ or $A = PLDL^T P^T$ if $uplo = Nag\_Lower$, 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 Arguments

1: order – Nag\_OrderType

Input

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

Input

On entry: specifies whether the upper or lower triangular part of $A$ is stored and how $A$ is to be factorized.

uplo = Nag\_Upper
The upper triangular part of $A$ is stored and $A$ is factorized as $PUDU^T P^T$, where $U$ is upper triangular.

uplo = Nag\_Lower
The lower triangular part of $A$ is stored and $A$ is factorized as $PLDL^T P^T$, where $L$ is lower triangular.

Constraint: uplo = Nag\_Upper or Nag\_Lower.
3: \ n \quad – \text{Integer} \quad \text{Input}

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

Constraint: \( n \geq 0 \).

4: \ ap[dim] \quad – \text{double} \quad \text{Input/Output}

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

On entry: the \( n \) by \( n \) symmetric matrix \( A \), packed by rows or columns.

The storage of elements \( A_{ij} \) depends on the \texttt{order} and \texttt{uplo} arguments as follows:

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

5: \ ipiv[n] \quad – \text{Integer} \quad \text{Output}

On exit: details of the interchanges and the block structure of \( D \). More precisely,

- if \texttt{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 \texttt{uplo} = \texttt{Nag Upper} and \texttt{ipiv}[i-2] = \texttt{ipiv}[i-1] = -l < 0, \( \begin{pmatrix} d_{l-1,i-1} & \tilde{d}_{i,j-1} \\ \tilde{d}_{i,j-1} & d_{ii} \end{pmatrix} \) 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 \texttt{uplo} = \texttt{Nag Lower} and \texttt{ipiv}[i-1] = \texttt{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: \ fail \quad – \text{NagError *} \quad \text{Input/Output}

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

6 Error Indicators and Warnings

NE_ALLOC_FAIL
Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

NE_BAD_PARAM
On entry, argument (\textit{value}) had an illegal value.

NE_INT
On entry, \( n = (\text{value}) \).
Constraint: \( n \geq 0 \).
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.

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.

NE_SINGULAR

Element (value) of the diagonal is exactly zero. The factorization has been completed, but the block diagonal matrix $D$ is exactly singular, and division by zero will occur if it is used to solve a system of equations.

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

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

$c(n)$ 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 Parallelism and Performance

nag_dsptrf (f07pdc) is not threaded by NAG in any implementation.

nag_dsptrf (f07pdc) 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 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 nag_dsptrf (f07pdc) may be followed by calls to the functions:

nag_dsptrs (f07pec) to solve $AX = B$;

nag_dspcon (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_zsptrf (f07qrc) for symmetric matrices.
This example computes 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.

### 10.1 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, nrhs, pdb;
    Integer exit_status = 0;
    NagError fail;
    Nag_UploType uplo;
    Nag_OrderType order;
    /* Arrays */
    Integer *ipiv = 0;
    char nag_enum_arg[40];
    double *ap = 0, *b = 0;

    INIT_FAIL(fail);
    printf("nag_dsptrf (f07pdc) Example Program Results\n\n");
    /* Skip heading in data file */
    #ifdef _WIN32
    scanf_s("%*[^\n] ");
    #else
    scanf("%*[^\n] ");
    #endif
    #ifdef _WIN32
    scanf_s("%"NAG_IFMT "%"NAG_IFMT "%"NAG_IFMT "%"NAG_IFMT "%"NAG_IFMT"");
    #else
    scanf("%*[\"n ] ", &n, &nrhs);
    #endif
```
```c
#ifndef NAG_COLUMN_MAJOR
    pdb = n;
#else
    pdb = nrhs;
#endif

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

/* Read A and B from data file */
#ifdef _WIN32
    scanf_s(" %39s%[\n ] ", nag_enum_arg, _countof(nag_enum_arg));
#else
    scanf(" %39s%[\n ] ", nag_enum_arg);
#endif
/* nag_enum_name_to_value (x04nac).
* Converts NAG enum member name to value */
uplo = (Nag_UploType) nag_enum_name_to_value(nag_enum_arg);
if (uplo == Nag_Upper)
{
    for (i = 1; i <= n; ++i)
        { 
            for (j = i; j <= n; ++j)
                { 
                    #ifdef _WIN32
                        scanf_s("%lf", &A_UPPER(i, j));
                    #else
                        scanf("%lf", &A_UPPER(i, j));
                    #endif
                }
        }
    #ifdef _WIN32
    scanf_s("%*[\n ] ");
#else
    scanf("%*[\n ] ");
#endif
    else
{ 
    for (i = 1; i <= n; ++i)
        { 
            for (j = 1; j <= i; ++j)
                { 
                    #ifdef _WIN32
                        scanf_s("%lf", &A_LOWER(i, j));
                    #else
                        scanf("%lf", &A_LOWER(i, j));
                    #endif
                }
        }
    #ifdef _WIN32
    scanf_s("%*[\n ] ");
#else
    scanf("%*[\n ] ");
#endif
    for (i = 1; i <= n; ++i)
        { 
            for (j = 1; j <= nrhs; ++j)
                { 
                    #ifdef _WIN32
                        scanf_s("%lf", &B(i, j));
                    #else
                        scanf("%lf", &B(i, j));
                    #endif
                }
        }
        }
    #else
        } 
#endif
```

---

The code snippet above demonstrates reading matrices and vectors from a file in C. It includes code for allocating memory, reading matrices A and B from a file, and converting NAG enum member names to values.
```c
scanf("%lf", &B(i, j));
#endif
#ifdef _WIN32
scanf_s("%*[\n ]");
#else
scanf("%*[\n ]");
#endif

/* Factorize A */
/* nag_dsptrf (f07pdc).
 * Bunch-Kaufman factorization of real symmetric indefinite
 * matrix, packed storage
 */
nag_dsptrf(order, uplo, n, ap, ipiv, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dsptrf (f07pdc).
%s
", fail.message);
    exit_status = 1;
    goto END;
}

/* Compute solution */
/* nag_dsprtrs (f07pec).
 * Solution of real symmetric indefinite system of linear
 * equations, multiple right-hand sides, matrix already
 * factorized by nag_dsptrf (f07pdc), packed storage
 */
nag_dsprtrs(order, uplo, n, nrhs, ap, ipiv, b, pdb, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dsprtrs (f07pec).
%s
", fail.message);
    exit_status = 1;
    goto END;
}

/* Print solution */
/* nag_gen_real_mat_print (x04cac).
 * Print real general matrix (easy-to-use)
 */
fflush(stdout);
nag_gen_real_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, nrhs, b, pdb, "Solution(s)", 0, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_gen_real_mat_print (x04cac).
%s
", fail.message);
    exit_status = 1;
    goto END;
}
END:
NAG_FREE(ap);
NAG_FREE(ipiv);
NAG_FREE(b);
return exit_status;
}
```

### 10.2 Program Data

**nag_dsptrf (f07pdc) Example Program Data**

<table>
<thead>
<tr>
<th>n</th>
<th>nrhs</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

- Values of uplo
- Lower
- Nag_Lower
- Value of n
- Values of n and nrhs
- 2.07
- 1.87
- 3.87
- -0.21
- 4.20
- 1.87
- 1.15
- -1.15
- 0.63
- 2.06
- 1.15
- -1.81
- -9.50
- 27.85
- -8.38
- 9.90
- -6.07
- 19.25
- -0.96
- 3.93

:End of matrix A

- -9.50
- 27.85
- -8.38
- 9.90
- -6.07
- 19.25
- -0.96
- 3.93

:End of matrix B
### 10.3 Program Results

**nag_dsprtf (f07pdc) Example Program Results**

<table>
<thead>
<tr>
<th>Solution(s)</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-4.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>2</td>
<td>-1.0000</td>
<td>4.0000</td>
</tr>
<tr>
<td>3</td>
<td>2.0000</td>
<td>3.0000</td>
</tr>
<tr>
<td>4</td>
<td>5.0000</td>
<td>2.0000</td>
</tr>
</tbody>
</table>