NAG C Library Function Document

nag_dsytrf (f07mdc)

1 Purpose

nag_dsytrf (f07mdc) computes the Bunch–Kaufman factorization of a real symmetric indefinite matrix.

2 Specification

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

3 Description

nag_dsytrf (f07mdc) factorizes a real symmetric matrix A, using the Bunch–Kaufman diagonal pivoting method. A is factorized as either $A = PUDU^T P^T$ if $\text{uplo} = \text{Nag_Upper}$, or $A = PLDL^T P^T$ if $\text{uplo} = \text{Nag_Lower}$, 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: 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 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.

   Constraint: $\text{uplo} = \text{Nag_Upper}$ or $\text{Nag_Lower}$.

3: n – Integer

   Input

   On entry: $n$, the order of the matrix $A$.

   Constraint: $n \geq 0$. 
4: \(a[\text{dim}]\) – double

**Input/Output**

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

If \(\text{order} = \text{Nag_ColMajor}\), the \((i, j)\)th element of the matrix \(A\) is stored in \(a[(j - 1) \times pda + i - 1]\) and if \(\text{order} = \text{Nag_RowMajor}\), the \((i, j)\)th element of the matrix \(A\) is stored in \(a[(i - 1) \times pda + j - 1]\).

**On entry:** the \(n\) by \(n\) symmetric indefinite matrix \(A\). If \(\text{uplo} = \text{Nag_Upper}\), the upper triangle of \(A\) must be stored and the elements of the array below the diagonal are not referenced; if \(\text{uplo} = \text{Nag_Lower}\), the lower triangle of \(A\) must be stored and the elements of the array above the diagonal are not referenced.

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

5: \(pda\) – Integer

**Input**

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

**Constraint:** \(pda \geq \max(1, n)\).

6: \(\text{ipiv}[\text{dim}]\) – 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 \(\text{ipiv}[i - 1] = k > 0\), \(d_{i,k}\) 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 \(\text{uplo} = \text{Nag_Upper}\) and \(\text{ipiv}[i - 2] = \text{ipiv}[i - 1] = -l < 0\), \(\begin{pmatrix} d_{i-1,j-1} & d_{i,j-1} \\ d_{i,j-1} & d_{i} \end{pmatrix}\) 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 \(\text{uplo} = \text{Nag_Lower}\) and \(\text{ipiv}[i - 1] = \text{ipiv}[i] = -m < 0\), \(\begin{pmatrix} d_{i} & d_{i+1,j} \\ d_{i+1,j} & d_{i+1,j+1} \end{pmatrix}\) 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.

7: \(\text{fail}\) – 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:** \(n \geq 0\).

**On entry:** \(pda = \langle\text{value}\rangle\).

**Constraint:** \(pda > 0\).

**NE_INT_2**

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

**Constraint:** \(pda \geq \max(1, n)\).

**NE_SINGULAR**

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

**NE_ALLOC_FAIL**

Memory allocation failed.
NE_BAD_PARAM

On entry, parameter (value) 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 uplo = 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 uplo = 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 uplo.

The unit diagonal elements of $U$ or $L$ and the 2 by 2 unit diagonal blocks are not stored. The remaining elements of $U$ and $L$ are stored in the corresponding columns of the array $a$, 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 $L$ is stored explicitly (except for its 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_dsystrs (f07mec) to solve $AX = B$;
- nag_dsycon (f07mgc) to estimate the condition number of $A$;
- nag_dsytri (f07mjc) to compute the inverse of $A$.

The complex analogues of this function are nag_zhetrf (f07mrc) for Hermitian matrices and nag_zsytrf (f07nrc) for symmetric matrices.

9 Example

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

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

9.1 Program Text

/* nag_dsytrf (f07mdc) Example Program. */
/* Copyright 2001 Numerical Algorithms Group. */
/* Mark 7, 2001. */
*
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf07.h>
```c
#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;

    #ifdef NAG_COLUMN_MAJOR
    #define A(I,J) a[(J-1)*pda + I-1]
    order = Nag_ColMajor;
    #else
    #define A(I,J) a[(I-1)*pda + J-1]
    order = Nag_RowMajor;
    #endif

    INIT_FAIL(fail);
    Vprintf("f07mdc Example Program Results\n\n");

    /* Skip heading in data file */
    Vscanf("%*[^\n] ");
    Vscanf("%ld%*[^\n] ", &n);
    #ifdef NAG_COLUMN_MAJOR
    pda = n;
    #else
    pda = n;
    #endif
    /* Allocate memory */
    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("' %1s' '%*[^\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.
%s
", fail.message);
    exit_status = 1;
    goto END;
}

/* Print factor */
x04cac(order, matrix, Nag_NonUnitDiag, n, n, a, pda,
"Details of Factorization", 0, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from x04cac.
%s
", fail.message);
    exit_status = 1;
    goto END;
}

/* Print pivot indices */
Vprintf("\nIPIV\n");
for (i = 1; i <= n; ++i)
    Vprintf("%11ld%s", ipiv[i-1], i%7==0 ?"\n":" ");
Vprintf("\n");
END:
if (ipiv) NAG_FREE(ipiv);
if (a) NAG_FREE(a);
return exit_status;

9.2 Program Data

f07mdc Example Program Data
4 :Value of N
'U' :Value of UPLO
2.07  3.87  4.20  -1.15
  -0.21  1.87  0.63
  1.15  2.06
  -1.81 :End of matrix A

9.3 Program Results

f07mdc Example Program Results

Details of Factorization

<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>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></td>
<td>0.4074</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

IPIV

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
</tr>
<tr>
<td>-3</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>