NAG C Library Function Document

nag_zsytrf (f07nrc)

1 Purpose

nag_zsytrf (f07nrc) computes the Bunch–Kaufman factorization of a complex symmetric matrix.

2 Specification

```
void nag_zsytrf (Nag_OrderType order, Nag_UploType uplo, Integer n, Complex a[],
     Integer pda, Integer ipiv[], NagError *fail)
```

3 Description

nag_zsytrf (f07nrc) factorizes a complex symmetric matrix A, using the Bunch–Kaufman diagonal pivoting method. A is factorized as either $A = PUDU^TP^T$ if **uplo** = **Nag_Upper**, or $A = PLDL^TP^T$ 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.

4 References

Golub G H and Van Loan C F (1996) *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

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 **uplo** = **Nag_Upper**, the upper triangular part of A is stored and A is factorized as $PUDU^TP^T$, where U is upper triangular;

if **uplo** = **Nag_Lower**, the lower triangular part of A is stored and A is factorized as $PLDL^TP^T$, 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** ≥ 0.

4: **a**[dim] – Complex *Input/Output*

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

If **order** = **Nag_ColMajor**, the (i,j)th element of the matrix A is stored in **a**$[(j-1) \times$ **pda** $+ i - 1]$ and if **order** = **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 **uplo** = **Nag_Upper**, the upper triangle of A must be stored and the elements of the array below the diagonal are not referenced; if **uplo** = **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 **uplo**.

5: **pda** – Integer *Input*

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

Constraint: **pda** $\geq \max(1, \mathbf{n})$.

6: **ipiv**$[dim]$ – Integer *Output*

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

On exit: details of the interchanges and the block structure of D.

More precisely, if **ipiv**$[i-1] = k > 0$, d_{ii} is a 1 by 1 pivot block and the ith row and column of A were interchanged with the kth row and column.

If **uplo** = **Nag_Upper** and **ipiv**$[i-2]$ = **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 lth row and column.

If **uplo** = **Nag_Lower** and **ipiv**$[i-1]$ = **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 mth row and column.

7: **fail** – NagError * *Output*

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 0.

On entry, **pda** = ⟨*value*⟩.
Constraint: **pda** > 0.

NE_INT_2

On entry, **pda** = ⟨*value*⟩, **n** = ⟨*value*⟩.
Constraint: **pda** $\geq \max(1, \mathbf{n})$.

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 ϵ 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 or 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 **ipiv**$[i-1] = i$, for $i = 1, 2, \ldots, n$, then U or L is stored explicitly (except for its unit diagonal elements which are equal to 1).

The total number of real floating-point operations is approximately $\frac{4}{3}n^3$.

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

nag_zsytrs (f07nsc) to solve $AX = B$;

nag_zsycon (f07nuc) to estimate the condition number of A;

nag_zsytri (f07nwc) to compute the inverse of A.

The real analogue of this function is nag_dsytrf (f07mdc).

9 Example

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

$$A = \begin{pmatrix} -0.39 - 0.71i & 5.14 - 0.64i & -7.86 - 2.96i & 3.80 + 0.92i \\ 5.14 - 0.64i & 8.86 + 1.81i & -3.52 + 0.58i & 5.32 - 1.59i \\ -7.86 - 2.96i & -3.52 + 0.58i & -2.83 - 0.03i & -1.54 - 2.86i \\ 3.80 + 0.92i & 5.32 - 1.59i & -1.54 - 2.86i & -0.56 + 0.12i \end{pmatrix}.$$

9.1 Program Text

```
/* nag_zsytrf (f07nrc) 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 i, j, n, pda;
  Integer exit_status=0;
```

```
  Nag_UploType uplo_enum;
  Nag_MatrixType matrix;

  NagError fail;
  Nag_OrderType order;
  /* Arrays */
  Integer *ipiv=0;
  char    uplo[2];
  Complex *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("f07nrc 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, Complex)) )
    {
      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 , %lf )", &A(i,j).re, &A(i,j).im);
        }
      Vscanf("%*[^\n] ");
    }
  else
    {
      for (i = 1; i <= n; ++i)
        {
          for (j = 1; j <= i; ++j)
```

```
            Vscanf(" ( %lf , %lf )", &A(i,j).re, &A(i,j).im);
        }
      Vscanf("%*[^\n] ");
    }

  /* Factorize A */
  f07nrc(order, uplo_enum, n, a, pda, ipiv, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from f07nrc.\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  /* Print factor */
  x04dbc(order, matrix, Nag_NonUnitDiag, n, n, a, pda, Nag_BracketForm,
         "%7.4f", "Details of Factorixation", Nag_IntegerLabels, 0,
         Nag_IntegerLabels, 0, 80, 0, 0, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from x04dbc.\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  /* Print pivot indices */
  Vprintf("\nIPIV\n");
  for (i = 1; i <= n; ++i)
    Vprintf("%3ld%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

```
f07nrc Example Program Data
  4                                                            :Value of N
  'U'                                                          :Value of UPLO
 (-0.39,-0.71) ( 5.14,-0.64) (-7.86,-2.96) ( 3.80, 0.92)
               ( 8.86, 1.81) (-3.52, 0.58) ( 5.32,-1.59)
                             (-2.83,-0.03) (-1.54,-2.86)
                                           (-0.56, 0.12)  :End of matrix A
```

9.3 Program Results

```
f07nrc Example Program Results

 Details of Factorixation
                  1                  2                  3                  4
 1  (-0.3900,-0.7100)  (-7.8600,-2.9600)  ( 0.5279,-0.3715)  ( 0.4426, 0.1936)
 2                     (-2.8300,-0.0300)  (-0.6078, 0.2811)  (-0.4823, 0.0150)
 3                                        ( 4.4079, 5.3991)  (-0.1071,-0.3157)
 4                                                           (-2.0954,-2.2011)

IPIV
 -3                 -3                 3                 4
```