

# NAG C Library Function Document

## nag\_zsptf (f07qrc)

### 1 Purpose

nag\_zsptf (f07qrc) computes the Bunch–Kaufman factorization of a complex symmetric matrix, using packed storage.

### 2 Specification

```
void nag_zsptf (Nag_OrderType order, Nag_UploType uplo, Integer n, Complex ap[],
               Integer ipiv[], NagError *fail)
```

### 3 Description

nag\_zsptf (f07qrc) factorizes a complex 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.

### 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  $PUDU^T P^T$ , where  $U$  is upper triangular;  
     if **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** – Integer *Input*  
*On entry:*  $n$ , the order of the matrix  $A$ .  
*Constraint:*  $n \geq 0$ .

4: **ap**[*dim*] – Complex

*Input/Output*

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

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

if **order** = **Nag\_ColMajor** and **uplo** = **Nag\_Upper**,  
 $a_{ij}$  is stored in **ap**[(*j* − 1) × *j*/2 + *i* − 1], for *i* ≤ *j*;  
 if **order** = **Nag\_ColMajor** and **uplo** = **Nag\_Lower**,  
 $a_{ij}$  is stored in **ap**[(2*n* − *j*) × (*j* − 1)/2 + *i* − 1], for *i* ≥ *j*;  
 if **order** = **Nag\_RowMajor** and **uplo** = **Nag\_Upper**,  
 $a_{ij}$  is stored in **ap**[(2*n* − *i*) × (*i* − 1)/2 + *j* − 1], for *i* ≤ *j*;  
 if **order** = **Nag\_RowMajor** and **uplo** = **Nag\_Lower**,  
 $a_{ij}$  is stored in **ap**[(*i* − 1) × *i*/2 + *j* − 1], for *i* ≥ *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 **uplo**.

5: **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 *i*th row and column of *A* were interchanged with the *k*th 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 *l*th 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 *m*th row and column.

6: **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.

### 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  $U$  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 **ipiv** $[i - 1] = i$ , for  $i = 1, 2, \dots, n$ , 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 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\_zsptrs (f07qsc) to solve  $AX = B$ ;

nag\_zspcon (f07quc) to estimate the condition number of  $A$ ;

nag\_zsptri (f07qwc) to compute the inverse of  $A$ .

The real analogue of this function is nag\_dsptrf (f07pdc).

## 9 Example

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

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

using packed storage.

### 9.1 Program Text

```
/* nag_zsptrf (f07qrc) 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];
Complex *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("f07qrc 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, Complex)) )
{
  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 , %lf )", &A_UPPER(i,j).re, &A_UPPER(i,j).im);
  }
  Vscanf("%*[^\\n] ");
}
else
{
  for (i = 1; i <= n; ++i)
  {
    for (j = 1; j <= i; ++j)
      Vscanf(" ( %lf , %lf )", &A_LOWER(i,j).re, &A_LOWER(i,j).im);
  }
  Vscanf("%*[^\\n] ");
}

/* Factorize A */
f07qrc(order, uplo_enum, n, ap, ipiv, &fail);
if (fail.code != NE_NOERROR)
{
  Vprintf("Error from f07qrc.\n%s\n", fail.message);
  exit_status = 1;
  goto END;
}

```

```

    }
    /* Print details of factorization */
    x04ddc(order, uplo_enum, Nag_NonUnitDiag, n, ap,
           Nag_BracketForm, "%7.4f", "Factor", Nag_IntegerLabels,
           0, Nag_IntegerLabels, 0, 80, 0, 0, &fail);
    if (fail.code != NE_NOERROR)
    {
        Vprintf("Error from x04ddc.\n%s\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 (ipiv) NAG_FREE(ipiv);
    if (ap) NAG_FREE(ap);
    return exit_status;
}

```

## 9.2 Program Data

f07qrc 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

f07qrc Example Program Results

```

Factor
      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

```

---