

## NAG Library Function Document

### nag\_real\_cholesky\_skyline\_solve (f04mcc)

#### 1 Purpose

nag\_real\_cholesky\_skyline\_solve (f04mcc) computes the approximate solution of a system of real linear equations with multiple right-hand sides,  $AX = B$ , where  $A$  is a symmetric positive definite variable-bandwidth matrix, which has previously been factorized by nag\_real\_cholesky\_skyline (f01mcc). Related systems may also be solved.

#### 2 Specification

```
#include <nag.h>
#include <nagf04.h>

void nag_real_cholesky_skyline_solve (Nag_SolveSystem selct, Integer n,
    Integer nrhs, const double a[], Integer lal, const double d[],
    const Integer row[], const double b[], Integer tdb, double x[],
    Integer tdx, NagError *fail)
```

#### 3 Description

The normal use of nag\_real\_cholesky\_skyline\_solve (f04mcc) is the solution of the systems  $AX = B$ , following a call of nag\_real\_cholesky\_skyline (f01mcc) to determine the Cholesky factorization  $A = LDL^T$  of the symmetric positive definite variable-bandwidth matrix  $A$ .

However, the function may be used to solve any one of the following systems of linear algebraic equations:

$$LDL^T X = B \text{ (usual system)} \quad (1)$$

$$LDX = B \text{ (lower triangular system)} \quad (2)$$

$$DL^T X = B \text{ (upper triangular system)} \quad (3)$$

$$LL^T X = B \quad (4)$$

$$LX = B \text{ (unit lower triangular system)} \quad (5)$$

$$L^T X = B \text{ (unit upper triangular system)} \quad (6)$$

$L$  denotes a unit lower triangular variable-bandwidth matrix of order  $n$ ,  $D$  a diagonal matrix of order  $n$ , and  $B$  a set of right-hand sides.

The matrix  $L$  is represented by the elements lying within its **envelope**, i.e., between the first nonzero of each row and the diagonal (see Section 10 for an example). The width **row**[ $i$ ] of the  $i$ th row is the number of elements between the first nonzero element and the element on the diagonal inclusive.

#### 4 References

Wilkinson J H and Reinsch C (1971) *Handbook for Automatic Computation II, Linear Algebra* Springer-Verlag

## 5 Arguments

- 1: **selct** – Nag\_SolveSystem *Input*  
*On entry:* **selct** must specify the type of system to be solved, as follows:  
 if **selct** = Nag\_LDLTX: solve  $LDL^T X = B$ ;  
 if **selct** = Nag\_LDX: solve  $LDX = B$ ;  
 if **selct** = Nag\_DLTX: solve  $DL^T X = B$ ;  
 if **selct** = Nag\_LLTX: solve  $LL^T X = B$ ;  
 if **selct** = Nag\_LX: solve  $LX = B$ ;  
 if **selct** = Nag\_LTX: solve  $L^T X = B$ .  
*Constraint:* **selct** = Nag\_LDLTX, Nag\_LDX, Nag\_DLTX, Nag\_LLTX, Nag\_LX or Nag\_LTX.
- 2: **n** – Integer *Input*  
*On entry:*  $n$ , the order of the matrix  $L$ .  
*Constraint:*  $n \geq 1$ .
- 3: **nrhs** – Integer *Input*  
*On entry:*  $r$ , the number of right-hand sides.  
*Constraint:* **nrhs**  $\geq 1$ .
- 4: **al[lal]** – const double *Input*  
*On entry:* the elements within the envelope of the lower triangular matrix  $L$ , taken in row by row order, as returned by nag\_real\_cholesky\_skyline (f01mcc). The unit diagonal elements of  $L$  must be stored explicitly.
- 5: **lal** – Integer *Input*  
*On entry:* the dimension of the array **al**.  
*Constraint:* **lal**  $\geq \mathbf{row}[0] + \mathbf{row}[1] + \dots + \mathbf{row}[n - 1]$ .
- 6: **d[n]** – const double *Input*  
*On entry:* the diagonal elements of the diagonal matrix  $D$ . **d** is not referenced if **selct** = Nag\_LLTX, Nag\_LX or Nag\_LTX
- 7: **row[n]** – const Integer *Input*  
*On entry:* **row** $[i]$  must contain the width of row  $i$  of  $L$ , i.e., the number of elements between the first (left-most) nonzero element and the element on the diagonal, inclusive.  
*Constraint:*  $1 \leq \mathbf{row}[i] \leq i + 1$  for  $i = 0, 1, \dots, n - 1$ .
- 8: **b[n  $\times$  tdb]** – const double *Input*  
**Note:** the  $(i, j)$ th element of the matrix  $B$  is stored in **b** $[(i - 1) \times \mathbf{tdb} + j - 1]$ .  
*On entry:* the  $n$  by  $r$  right-hand side matrix  $B$ . See also Section 9.
- 9: **tdb** – Integer *Input*  
*On entry:* the stride separating matrix column elements in the array **b**.  
*Constraint:* **tdb**  $\geq \mathbf{nrhs}$ .

- 10: **x**[**n** × **tdx**] – double *Output*  
**Note:** the  $(i, j)$ th element of the matrix  $X$  is stored in  $\mathbf{x}[(i - 1) \times \mathbf{tdx} + j - 1]$ .  
*On exit:* the  $n$  by  $r$  solution matrix  $X$ . See also Section 9.
- 11: **tdx** – Integer *Input*  
*On entry:* the stride separating matrix column elements in the array  $\mathbf{x}$ .  
*Constraint:*  $\mathbf{tdx} \geq \mathbf{nrhs}$ .
- 12: **fail** – NagError \* *Input/Output*  
The NAG error argument (see Section 3.6 in the Essential Introduction).

## 6 Error Indicators and Warnings

### NE\_2\_INT\_ARG\_GT

On entry,  $\mathbf{row}[i] = \langle \text{value} \rangle$  while  $i = \langle \text{value} \rangle$ . These arguments must satisfy  $\mathbf{row}[i] \leq i + 1$ .

### NE\_2\_INT\_ARG\_LT

On entry,  $\mathbf{lal} = \langle \text{value} \rangle$  while  $\mathbf{row}[0] + \dots + \mathbf{row}[n - 1] = \langle \text{value} \rangle$ . These arguments must satisfy  $\mathbf{lal} \geq \mathbf{row}[0] + \dots + \mathbf{row}[n - 1]$ .

On entry,  $\mathbf{tdb} = \langle \text{value} \rangle$  while  $\mathbf{nrhs} = \langle \text{value} \rangle$ . These arguments must satisfy  $\mathbf{tdb} \geq \mathbf{nrhs}$ .

On entry,  $\mathbf{tdx} = \langle \text{value} \rangle$  while  $\mathbf{nrhs} = \langle \text{value} \rangle$ . These arguments must satisfy  $\mathbf{tdx} \geq \mathbf{nrhs}$ .

### NE\_BAD\_PARAM

On entry, argument **selct** had an illegal value.

### NE\_INT\_ARG\_LT

On entry,  $\mathbf{n} = \langle \text{value} \rangle$ .

Constraint:  $\mathbf{n} \geq 1$ .

On entry,  $\mathbf{nrhs} = \langle \text{value} \rangle$ .

Constraint:  $\mathbf{nrhs} \geq 1$ .

On entry,  $\mathbf{row}[\langle \text{value} \rangle]$  must not be less than 1:  $\mathbf{row}[\langle \text{value} \rangle] = \langle \text{value} \rangle$ .

### NE\_NOT\_UNIT\_DIAG

The lower triangular matrix  $L$  has at least one diagonal element which is not equal to unity. The first non-unit element has been located in the array  $\mathbf{al}[\langle \text{value} \rangle]$ .

### NE\_ZERO\_DIAG

The diagonal matrix  $D$  is singular as it has at least one zero element. The first zero element has been located in the array  $\mathbf{d}[\langle \text{value} \rangle]$ .

## 7 Accuracy

The usual backward error analysis of the solution of triangular system applies: each computed solution vector is exact for slightly perturbed matrices  $L$  and  $D$ , as appropriate (see pages 25-27 and 54-55 of Wilkinson and Reinsch (1971)).

## 8 Parallelism and Performance

`nag_real_cholesky_skyline_solve (f04mcc)` is not threaded by NAG in any implementation.

nag\_real\_cholesky\_skyline\_solve (f04mcc) 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 Users' Note for your implementation for any additional implementation-specific information.

## 9 Further Comments

The time taken by nag\_real\_cholesky\_skyline\_solve (f04mcc) is approximately proportional to  $pr$ , where  $p = \mathbf{row}[0] + \mathbf{row}[1] + \dots + \mathbf{row}[n-1]$ .

The function may be called with the same actual array supplied for the arguments  $\mathbf{b}$  and  $\mathbf{x}$ , in which case the solution matrix will overwrite the right-hand side matrix.

## 10 Example

To solve the system of equations  $AX = B$ , where

$$A = \begin{pmatrix} 1 & 2 & 0 & 0 & 5 & 0 \\ 2 & 5 & 3 & 0 & 14 & 0 \\ 0 & 3 & 13 & 0 & 18 & 0 \\ 0 & 0 & 0 & 16 & 8 & 24 \\ 5 & 14 & 18 & 8 & 55 & 17 \\ 0 & 0 & 0 & 24 & 17 & 77 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 6 & -10 \\ 15 & -21 \\ 11 & -3 \\ 0 & 24 \\ 51 & -39 \\ 46 & 67 \end{pmatrix}.$$

Here  $A$  is symmetric and positive definite and must first be factorized by nag\_real\_cholesky\_skyline (f01mcc).

### 10.1 Program Text

```

/* nag_real_cholesky_skyline_solve (f04mcc) Example Program.
 *
 * Copyright 1996 Numerical Algorithms Group.
 *
 * Mark 4, 1996.
 * Mark 8 revised, 2004.
 */

#include <nag.h>
#include <math.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagf01.h>
#include <nagf04.h>

#define B(I, J) b[(I) *tdb + J]
#define X(I, J) x[(I) *tdx + J]

int main(void)
{
    Integer          exit_status = 0, i, k, k1, k2, lal, n, nrhs, *row = 0, tdb,
                    tdx;
    Nag_SolveSystem select;
    double           *a = 0, *al = 0, *b = 0, *d = 0, *x = 0;
    NagError         fail;

    INIT_FAIL(fail);

    printf(
        "nag_real_cholesky_skyline_solve (f04mcc) Example Program Results\n");
    /* Skip heading in data file */
    scanf("%*[\n]");
    scanf("%ld", &n);
    if (n >= 1)
        {

```

```

        if (!(row = NAG_ALLOC(n, Integer)))
        {
            printf("Allocation failure\n");
            exit_status = -1;
            goto END;
        }
    }
else
    {
        printf("Invalid n.\n");
        exit_status = 1;
        return exit_status;
    }

lal = 0;
for (i = 0; i < n; ++i)
    {
        scanf("%ld", &row[i]);
        lal += row[i];
    }
if (!(a = NAG_ALLOC(lal, double)) ||
    !(al = NAG_ALLOC(lal, double)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
k2 = 0;
for (i = 0; i < n; ++i)
    {
        k1 = k2;
        k2 = k2 + row[i];
        for (k = k1; k < k2; ++k)
            scanf("%lf", &a[k]);
    }
scanf("%ld", &nrhs);
if (nrhs >= 1)
    {
        if (!(b = NAG_ALLOC(n*nrhs, double)) ||
            !(d = NAG_ALLOC(n, double)) ||
            !(x = NAG_ALLOC(n*nrhs, double)))
            {
                printf("Allocation failure\n");
                exit_status = -1;
                goto END;
            }
        tdb = nrhs;
        tdx = nrhs;
    }
else
    {
        printf("Invalid nrhs.\n");
        exit_status = 1;
        return exit_status;
    }
for (i = 0; i < n; ++i)
    for (k = 0; k < nrhs; ++k)
        scanf("%lf", &B(i, k));
/* nag_real_cholesky_skyline (f01mcc).
 * LDL^T factorization of real symmetric positive-definite
 * variable-bandwidth (skyline) matrix
 */
nag_real_cholesky_skyline(n, a, lal, row, al, d, &fail);
if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_real_cholesky_skyline (f01mcc).\n%s\n",
            fail.message);
        exit_status = 1;
        goto END;
    }
select = Nag_LDLTX;

```

```

/* nag_real_cholesky_skyline_solve (f04mcc).
 * Approximate solution of real symmetric positive-definite
 * variable-bandwidth simultaneous linear equations
 * (coefficient matrix already factorized by
 * nag_real_cholesky_skyline (f01mcc))
 */
nag_real_cholesky_skyline_solve(select, n, nrhs, al, lal, d, row, b, tdb,
                               x, tdx, &fail);
if (fail.code != NE_NOERROR)
{
    printf(
        "Error from nag_real_cholesky_skyline_solve (f04mcc).\n%s\n",
        fail.message);
    exit_status = 1;
    goto END;
}
printf("\n Solution\n");
for (i = 0; i < n; ++i)
{
    for (k = 0; k < nrhs; ++k)
        printf("%9.3f", X(i, k));
    printf("\n");
}
END:
NAG_FREE(row);
NAG_FREE(b);
NAG_FREE(d);
NAG_FREE(x);
NAG_FREE(a);
NAG_FREE(al);
return exit_status;
}

```

## 10.2 Program Data

nag\_real\_cholesky\_skyline\_solve (f04mcc) Example Program Data

```

6
1 2 2 1 5 3
1.0
2.0 5.0
3.0 13.0
16.0
5.0 14.0 18.0 8.0 55.0
24.0 17.0 77.0
2
6.0 -10.0
15.0 -21.0
11.0 -3.0
0.0 24.0
51.0 -39.0
46.0 67.0

```

## 10.3 Program Results

nag\_real\_cholesky\_skyline\_solve (f04mcc) Example Program Results

```

Solution
-3.000 4.000
2.000 -2.000
-1.000 3.000
-2.000 1.000
1.000 -2.000
1.000 1.000

```

---