NAG Library Function Document

nag_real_cholesky_skyline (f01mcc)

1 Purpose

nag_real_cholesky_skyline (f01mcc) computes the Cholesky factorization of a real symmetric positive definite variable-bandwidth matrix.

2 Specification

```
#include <nag.h>
#include <nagf01.h>
void nag_real_cholesky_skyline (Integer n, const double a[], Integer lal,
     Integer row[], double al[], double d[], NagError *fail)
```

3 Description

nag_real_cholesky_skyline (f01mcc) determines the unit lower triangular matrix L and the diagonal matrix D in the Cholesky factorization $A = LDL^{\mathrm{T}}$ of a symmetric positive definite variable-bandwidth matrix A of order n. (Such a matrix is sometimes called a 'sky-line' matrix.)

The matrix A is represented by the elements lying within the **envelope** of its lower triangular part, that is, between the first nonzero of each row and the diagonal (see Section 10 for an example). The width **row**$[i]$ of the ith row is the number of elements between the first nonzero element and the element on the diagonal, inclusive. Although, of course, any matrix possesses an envelope as defined, this function is primarily intended for the factorization of symmetric positive definite matrices with an **average** bandwidth which is small compared with n (also see Section 9).

The method is based on the property that during Cholesky factorization there is no fill-in outside the envelope.

The determination of L and D is normally the first of two steps in the solution of the system of equations $Ax = b$. The remaining step, viz. the solution of $LDL^{\mathrm{T}}x = b$ may be carried out using nag_real_cholesky_skyline_solve (f04mcc).

4 References

Jennings A (1966) A compact storage scheme for the solution of symmetric linear simultaneous equations *Comput. J.* **9** 281–285

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

5 Arguments

1: **n** – Integer *Input*

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

Constraint: **n** ≥ 1.

2: **a**[**lal**] – const double *Input*

On entry: the elements within the envelope of the lower triangle of the positive definite symmetric matrix A, taken in row by row order. The following code assigns the matrix elements within the envelope to the correct elements of the array

```
k=0;
for(i=0; i<n; ++i)
  for(j=i-row[i]+1; j<=i; ++j)
   a[k++]=matrix[i][j];
```

See also Section 9

3: **lal** – Integer *Input*

On entry: the smaller of the dimensions of the arrays **a** and **al** as declared in the function from which nag_real_cholesky_skyline (f01mcc) is called.

Constraint: **lal** $\geq$ **row**$[0] + $ **row**$[1] + \cdots +$ **row**$[n-1]$.

4: **row**[**n**] – Integer *Input*

On entry: **row**$[i]$ must contain the width of row i of the matrix A, i.e., the number of elements between the first (left-most) nonzero element and the element on the diagonal, inclusive.

Constraint: $1 \leq$ **row**$[i] \leq i+1$, for $i = 0, 1, \ldots, n-1$.

5: **al**[**lal**] – double *Output*

On exit: the elements within the envelope of the lower triangular matrix L, taken in row by row order. The envelope of L is identical to that of the lower triangle of A. The unit diagonal elements of L are stored explicitly. See also Section 9

6: **d**[**n**] – double *Output*

On exit: the diagonal elements of the diagonal matrix D. Note that the determinant of A is equal to the product of these diagonal elements. If the value of the determinant is required it should not be determined by forming the product explicitly, because of the possibility of overflow or underflow. The logarithm of the determinant may safely be formed from the sum of the logarithms of the diagonal elements.

7: **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, **row**$[\langle value \rangle] = \langle value \rangle$ while $i = \langle value \rangle$. These arguments must satisfy **row**$[i] \leq i+1$.

NE_2_INT_ARG_LT

On entry, **lal** $= \langle value \rangle$ while **row**$[0] + \cdots +$ **row**$[n-1] = \langle value \rangle$. These arguments must satisfy **lal** $\geq$ **row**$[0] + \cdots +$ **row**$[n-1]$.

NE_INT_ARG_LT

On entry, **n** $= \langle value \rangle$.
Constraint: **n** ≥ 1.

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

NE_NOT_POS_DEF

The matrix is not positive definite, possibly due to rounding errors.

NE_NOT_POS_DEF_FACT

The matrix is not positive definite, possibly due to rounding errors. The factorization has been completed but may be very inaccurate.

7 Accuracy

On successful exit then the **computed** L and D satisfy the relation $LDL^{\mathrm{T}} = A + F$, where

$$\|F\|_2 \leq km^2\epsilon \max_i a_{ii}$$

and

$$\|F\|_2 \leq km^2\epsilon\|A\|_2,$$

where k is a constant of order unity, m is the largest value of **row**$[i]$, and ϵ is the ***machine precision***. See pages 25–27 and 54–55 or Wilkinson and Reinsch (1971). If the error NE_NOT_POS_DEF_FACT is reported then the factorization has been completed although the matrix was not positive definite. However the factorization may be very inaccurate and should be used only with great caution. For instance, if it is used to solve a set of equations $Ax = b$ using nag_real_cholesky_skyline_solve (f04mcc), the residual vector $b - Ax$ should be checked.

8 Parallelism and Performance

Not applicable.

9 Further Comments

The time taken by nag_real_cholesky_skyline (f01mcc) is approximately proportional to the sum of squares of the values of **row**$[i]$.

The distribution of row widths may be very non-uniform without undue loss of efficiency. Moreover, the function has been designed to be as competitive as possible in speed with functions designed for full or uniformly banded matrices, when applied to such matrices.

The function may be called with the same actual array supplied for arguments **a** and **al**, in which case L overwrites the lower triangle of A.

10 Example

To obtain the Cholesky factorization of the symmetric matrix, whose lower triangle is

.

For this matrix, the elements of **row** must be set to 1, 2, 2, 1, 5, 3, and the elements within the envelope must be supplied in row order as

$$1, 2, 5, 3, 13, 16, 5, 14, 18, 8, 55, 24, 17, 77.$$

10.1 Program Text

```
/* nag_real_cholesky_skyline (f01mcc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 4, 1996.
 * Mark 8 revised, 2004.
 */

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

int main(void)
{
  Integer  exit_status = 0, i, k, k1, k2, lal, n, *row = 0;
  NagError fail;
  double   *a = 0, *al = 0, *d = 0;
```

```
  INIT_FAIL(fail);

  printf(
          "nag_real_cholesky_skyline (f01mcc) Example Program Results\n");
  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif
#ifdef _WIN32
  scanf_s("%"NAG_IFMT"", &n);
#else
  scanf("%"NAG_IFMT"", &n);
#endif
  if (n >= 1)
    {
      if (!(row = NAG_ALLOC(n, Integer)) ||
          !(d = NAG_ALLOC(n, double)))
        {
          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++)
    {
#ifdef _WIN32
      scanf_s("%"NAG_IFMT"", &row[i]);
#else
      scanf("%"NAG_IFMT"", &row[i]);
#endif
      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++)
#ifdef _WIN32
        scanf_s("%lf", &a[k]);
#else
        scanf("%lf", &a[k]);
#endif
    }
  /* 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;
    }
  printf("\n");
  printf("   i    d[i]   Row i of unit lower triangle\n\n");
  k2 = 0;
  for (i = 0; i < n; ++i)
    {
      k1 = k2;
      k2 = k2+row[i];
      printf(" %3"NAG_IFMT"%8.3f", i, d[i]);
      for (k = k1; k < k2; k++)
        printf("%8.3f", al[k]);
      printf("\n");
    }
 END:
  NAG_FREE(row);
  NAG_FREE(a);
  NAG_FREE(al);
  NAG_FREE(d);
  return exit_status;
}
```

10.2 Program Data

```
nag_real_cholesky_skyline (f01mcc) 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
```

10.3 Program Results

```
nag_real_cholesky_skyline (f01mcc) Example Program Results

   i    d[i]   Row i of unit lower triangle

   0   1.000   1.000
   1   1.000   2.000   1.000
   2   4.000   3.000   1.000
   3  16.000   1.000
   4   1.000   5.000   4.000   1.500   0.500   1.000
   5  16.000   1.500   5.000   1.000
```