NAG Toolbox for MATLAB

f11ja

1 Purpose

f11ja computes an incomplete Cholesky factorization of a real sparse symmetric matrix, represented in symmetric co-ordinate storage format. This factorization may be used as a preconditioner in combination with f11jc or f11ge.

2 Syntax

```
[a, irow, icol, ipiv, istr, nnzc, npivm, ifail] = f11ja(nnz, a, irow,
icol, lfill, dtol, mic, dscale, ipiv, 'n', n, 'la', la, 'pstrat',
pstrat)
```

3 Description

f11ja computes an incomplete Cholesky factorization (see Meijerink and Van der Vorst 1977) of a real sparse symmetric n by n matrix A. It is designed specifically for positive-definite matrices, but may also work for some mildly indefinite cases. The factorization is intended primarily for use as a preconditioner with one of the symmetric iterative solvers f11jc or f11ge.

The decomposition is written in the form

$$A = M + R$$

where

$$M = PLDL^{\mathrm{T}}P^{\mathrm{T}}$$

and P is a permutation matrix, L is lower triangular with unit diagonal elements, D is diagonal and R is a remainder matrix.

The amount of fill-in occurring in the factorization can vary from zero to complete fill, and can be controlled by specifying either the maximum level of fill **lfill**, or the drop tolerance **dtol**. The factorization may be modified in order to preserve row sums, and the diagonal elements may be perturbed to ensure that the preconditioner is positive-definite. Diagonal pivoting may optionally be employed, either with a user-defined ordering, or using the Markowitz strategy (see Markowitz 1957), which aims to minimize fill-in. For further details see Section 8.

The sparse matrix A is represented in symmetric co-ordinate storage (SCS) format (see Section 2.1.2 in the F11 Chapter Introduction). The array **a** stores all the nonzero elements of the lower triangular part of A, while arrays **irow** and **icol** store the corresponding row and column indices respectively. Multiple nonzero elements may not be specified for the same row and column index.

The preconditioning matrix M is returned in terms of the SCS representation of the lower triangular matrix

$$C = L + D^{-1} - I.$$

4 References

Chan T F 1991 Fourier analysis of relaxed incomplete factorization preconditioners *SIAM J. Sci. Statist. Comput.* **12(2)** 668–680

Markowitz H M 1957 The elimination form of the inverse and its application to linear programming *Management Sci.* **3** 255–269

Meijerink J and Van der Vorst H 1977 An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix *Math. Comput.* **31** 148–162

Salvini S A and Shaw G J 1995 An evaluation of new NAG Library solvers for large sparse symmetric linear systems *NAG Technical Report TR1/95*

Van der Vorst H A 1990 The convergence behaviour of preconditioned CG and CG-S in the presence of rounding errors *Lecture Notes in Mathematics* (ed O Axelsson and L Y Kolotilina) **1457** Springer–Verlag

5 Parameters

5.1 Compulsory Input Parameters

1: **nnz – int32 scalar**

the number of nonzero elements in the lower triangular part of the matrix A.

Constraint: $1 \leq \mathbf{nnz} \leq \mathbf{n} \times (\mathbf{n} + 1)/2$.

2: **a(la) – double array**

The nonzero elements in the lower triangular part of the matrix A, ordered by increasing row index, and by increasing column index within each row. Multiple entries for the same row and column indices are not permitted. The function f11zb may be used to order the elements in this way.

3: **irow(la) – int32 array**
4: **icol(la) – int32 array**

The row and column indices of the nonzero elements supplied in **a**.

Constraints:

$1 \leq \mathbf{irow}(i) \leq \mathbf{n}$ and $1 \leq \mathbf{icol}(i) \leq \mathbf{irow}(i)$, for $i = 1, 2, \ldots, \mathbf{nnz}$;
$\mathbf{irow}(i-1) < \mathbf{irow}(i)$ or $\mathbf{irow}(i-1) = \mathbf{irow}(i)$ and $\mathbf{icol}(i-1) < \mathbf{icol}(i)$, for $i = 2, 3, \ldots, \mathbf{nnz}$.

irow and **icol** must satisfy the following constraints (which may be imposed by a call to f11zb):

5: **lfill – int32 scalar**

If $\mathbf{lfill} \geq 0$ its value is the maximum level of fill allowed in the decomposition (see Section 8.2). A negative value of **lfill** indicates that **dtol** will be used to control the fill instead.

6: **dtol – double scalar**

If $\mathbf{lfill} < 0$ then **dtol** is used as a drop tolerance to control the fill-in (see Section 8.2). Otherwise **dtol** is not referenced.

Constraint: if $\mathbf{lfill} < 0$, $\mathbf{dtol} \geq 0.0$.

7: **mic – string**

Indicates whether or not the factorization should be modified to preserve row sums (see Section 8.3).

mic = 'M'

The factorization is modified (**mic**).

mic = 'N'

The factorization is not modified.

Constraint: **mic** = 'M' or 'N'.

8: **dscale – double scalar**

The diagonal scaling parameter. All diagonal elements are multiplied by the factor $(1 + \mathbf{dscale})$ at the start of the factorization. This can be used to ensure that the preconditioner is positive-definite. See Section 8.3.

9: **ipiv(n) – int32 array**

If **pstrat** = 'U', then **ipiv**(i) must specify the row index of the diagonal element used as a pivot at elimination stage i. Otherwise **ipiv** need not be initialized.

Constraint: if **pstrat** = 'U', **ipiv** must contain a valid permutation of the integers on [1,**n**].

5.2 Optional Input Parameters

1: **n – int32 scalar**

Default: The dimension of the array **ipiv**.

n, the order of the matrix A.

Constraint: **n** ≥ 1.

2: **la – int32 scalar**

Default: The dimension of the arrays **a**, **irow**, **icol**. (An error is raised if these dimensions are not equal.)

These arrays must be of sufficient size to store both A (**nnz** elements) and C (**nnzc** elements).

Constraint: **la** $\geq 2 \times$ **nnz**.

3: **pstrat – string**

Specifies the pivoting strategy to be adopted.

pstrat = 'N'

No pivoting is carried out.

pstrat = 'M'

Diagonal pivoting aimed at minimizing fill-in is carried out, using the Markowitz strategy.

pstrat = 'U'

Diagonal pivoting is carried out according to the user-defined input value of **ipiv**.

Suggested value: **pstrat** = 'M'.

Default: 'M'

Constraint: **pstrat** = 'N', 'M' or 'U'.

5.3 Input Parameters Omitted from the MATLAB Interface

iwork, liwork

5.4 Output Parameters

1: **a(la) – double array**

The first **nnz** elements of **a** contain the nonzero elements of A and the next **nnzc** elements contain the elements of the lower triangular matrix C. Matrix elements are ordered by increasing row index, and by increasing column index within each row.

2: **irow(la) – int32 array**
3: **icol(la) – int32 array**

The row and column indices of the nonzero elements returned in **a**.

4: **ipiv(n) – int32 array**

The pivot indices. If **ipiv**(i) $= j$ then the diagonal element in row j was used as the pivot at elimination stage i.

5: **istr**($\mathbf{n}+\mathbf{1}$) – **int32 array**

istr(i), for $i = 1, 2, \ldots, \mathbf{n}$, holds the starting address in the arrays **a**, **irow** and **icol** of row i of the matrix C. **istr**($\mathbf{n}+1$) holds the address of the last nonzero element in C plus one.

6: **nnzc – int32 scalar**

The number of nonzero elements in the lower triangular matrix C.

7: **npivm – int32 scalar**

The number of pivots which were modified during the factorization to ensure that M was positive-definite. The quality of the preconditioner will generally depend on the returned value of **npivm**. If **npivm** is large the preconditioner may not be satisfactory. In this case it may be advantageous to call f11ja again with an increased value of either **lfill** or **dscale**. See also Section 8.4.

8: **ifail – int32 scalar**

0 unless the function detects an error (see Section 6).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

On entry, $\mathbf{n} < 1$,
or $\mathbf{nnz} < 1$,
or $\mathbf{nnz} > \mathbf{n} \times (\mathbf{n}+1)/2$,
or $\mathbf{la} < 2 \times \mathbf{nnz}$,
or $\mathbf{dtol} < 0.0$,
or **mic** $\neq$ 'M' or 'N',
or **pstrat** $\neq$ 'N', 'M' or 'U',
or $\mathbf{liwork} < 2 \times \mathbf{la} - 3 \times \mathbf{nnz} + 7 \times \mathbf{n} + 1$, and $\mathbf{lfill} \geq 0$,
or $\mathbf{liwork} < \mathbf{la} - \mathbf{nnz} + 7 \times \mathbf{n} + 1$, and $\mathbf{lfill} < 0$.

ifail $= 2$

On entry, the arrays **irow** and **icol** fail to satisfy the following constraints:

$1 \leq \mathbf{irow}(i) \leq \mathbf{n}$ and $1 \leq \mathbf{icol}(i) \leq \mathbf{irow}(i)$, for $i = 1, 2, \ldots, \mathbf{nnz}$;

$\mathbf{irow}(i-1) < \mathbf{irow}(i)$, or $\mathbf{irow}(i-1) = \mathbf{irow}(i)$ and $\mathbf{icol}(i-1) < \mathbf{icol}(i)$, for $i = 2, 3, \ldots, \mathbf{nnz}$.

Therefore a nonzero element has been supplied which does not lie in the lower triangular part of A, is out of order, or has duplicate row and column indices. Call f11zb to reorder and sum or remove duplicates.

ifail $= 3$

On entry, **pstrat** = 'U', but **ipiv** does not represent a valid permutation of the integers in $[1, \mathbf{n}]$. An input value of **ipiv** is either out of range or repeated.

ifail $= 4$

la is too small, resulting in insufficient storage space for fill-in elements. The decomposition has been terminated before completion. Either increase **la** or reduce the amount of fill by setting **pstrat** = 'M', reducing **lfill**, or increasing **dtol**.

ifail $= 5$ (f11zb)

A serious error has occurred in an internal call to the specified function. Check all (sub)program calls and array sizes. Seek expert help.

7 Accuracy

The accuracy of the factorization will be determined by the size of the elements that are dropped and the size of any modifications made to the diagonal elements. If these sizes are small then the computed factors will correspond to a matrix close to A. The factorization can generally be made more accurate by increasing **lfill**, or by reducing **dtol** with **lfill** < 0.

If f11ja is used in combination with f11ge or f11jc, the more accurate the factorization the fewer iterations will be required. However, the cost of the decomposition will also generally increase.

8 Further Comments

8.1 Timing

The time taken for a call to f11ja is roughly proportional to $(\mathbf{nnzc})^2/\mathbf{n}$.

8.2 Control of Fill-in

If **lfill** ≥ 0 the amount of fill-in occurring in the incomplete factorization is controlled by limiting the maximum **level** of fill-in to **lfill**. The original nonzero elements of A are defined to be of level 0. The fill level of a new nonzero location occurring during the factorization is defined as

$$k = \max(k_{\mathrm{e}}, k_{\mathrm{c}}) + 1,$$

where k_{e} is the level of fill of the element being eliminated, and k_{c} is the level of fill of the element causing the fill-in.

If **lfill** < 0 the fill-in is controlled by means of the **drop tolerance dtol**. A potential fill-in element a_{ij} occurring in row i and column j will not be included if

$$|a_{ij}| < \mathbf{dtol} \times \sqrt{|a_{ii}a_{jj}|}.$$

For either method of control, any elements which are not included are discarded if **mic** = 'N', or subtracted from the diagonal element in the elimination row if **mic** = 'M'.

8.3 Choice of Parameters

There is unfortunately no choice of the various algorithmic parameters which is optimal for all types of symmetric matrix, and some experimentation will generally be required for each new type of matrix encountered.

If the matrix A is not known to have any particular special properties the following strategy is recommended. Start with **lfill** $= 0$, **mic** = 'N' and **dscale** $= 0.0$. If the value returned for **npivm** is significantly larger than zero, i.e., a large number of pivot modifications were required to ensure that M was positive-definite, the preconditioner is not likely to be satisfactory. In this case increase either **lfill** or **dscale** until **npivm** falls to a value close to zero. Once suitable values of **lfill** and **dscale** have been found try setting **mic** = 'M' to see if any improvement can be obtained by using **modified** incomplete Cholesky.

f11ja is primarily designed for positive-definite matrices, but may work for some mildly indefinite problems. If **npivm** cannot be satisfactorily reduced by increasing **lfill** or **dscale** then A is probably too indefinite for this function.

If A has nonpositive off-diagonal elements, is nonsingular, and has only nonnegative elements in its inverse, it is called an 'M-matrix'. It can be shown that no pivot modifications are required in the incomplete Cholesky factorization of an M-matrix (see Meijerink and Van der Vorst 1977). In this case a good preconditioner can generally be expected by setting **lfill** $= 0$, **mic** = 'M' and **dscale** $= 0.0$.

For certain mesh-based problems involving M-matrices it can be shown in theory that setting **mic** = 'M', and choosing **dscale** appropriately can reduce the order of magnitude of the condition number of the preconditioned matrix as a function of the mesh steplength (see Chan 1991). In practise this property often holds even with **dscale** $= 0.0$, although an improvement in condition can result from increasing **dscale** slightly (see Van der Vorst 1990).

Some illustrations of the application of f11ja to linear systems arising from the discretization of two-dimensional elliptic partial differential equations, and to random-valued randomly structured symmetric positive-definite linear systems, can be found in Salvini and Shaw 1995.

8.4 Direct Solution of Positive-definite Systems

Although it is not their primary purpose, f11ja and f11jb may be used together to obtain a **direct** solution to a symmetric positive-definite linear system. To achieve this the call to f11jb should be preceded by a **complete** Cholesky factorization

$$A = PLDL^{\mathrm{T}}P^{\mathrm{T}} = M.$$

A complete factorization is obtained from a call to f11ja with **lfill** < 0 and **dtol** $= 0.0$, provided **npivm** $= 0$ on exit. A nonzero value of **npivm** indicates that **a** is not positive-definite, or is ill-conditioned. A factorization with nonzero **npivm** may serve as a preconditioner, but will not result in a direct solution. It is therefore **essential** to check the output value of **npivm** if a direct solution is required.

The use of f11ja and f11jb as a direct method is illustrated in Section 9 of the document for f11jb.

9 Example

```
nnz = int32(16);
a = [4;
     1;
     5;
     2;
     2;
     3;
     -1;
     1;
     4;
     1;
     -2;
     3;
     2;
     -1;
     -2;
     5;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0;
     0];
irow = [int32(1);
     int32(2);
     int32(2);
```

```
      int32(3);
      int32(4);
      int32(4);
      int32(5);
      int32(5);
      int32(5);
      int32(6);
      int32(6);
      int32(6);
      int32(7);
      int32(7);
      int32(7);
      int32(7);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
      int32(0)];
icol = [int32(1);
      int32(1);
      int32(2);
      int32(3);
      int32(2);
      int32(4);
      int32(1);
      int32(4);
      int32(5);
      int32(2);
      int32(5);
      int32(6);
      int32(1);
      int32(2);
      int32(3);
      int32(7);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
      int32(0);
```

```
     int32(0);
     int32(0);
     int32(0);
     int32(0);
     int32(0);
     int32(0);
     int32(0)];
lfill = int32(0);
dtol = 0;
mic = 'N';
dscale = 0;
ipiv = [int32(0);
     int32(0);
     int32(0);
     int32(0);
     int32(0);
     int32(0);
     int32(0)];
[aOut, irowOut, icolOut, ipivOut, istr, nnzc, npivm, ifail] = ...
    f11ja(nnz, a, irow, icol, lfill, dtol, mic, dscale, ipiv)
```

```
aOut =
    4.0000
    1.0000
    5.0000
    2.0000
    2.0000
    3.0000
   -1.0000
    1.0000
    4.0000
    1.0000
   -2.0000
    3.0000
    2.0000
   -1.0000
   -2.0000
    5.0000
    0.5000
    0.3333
    0.3333
    0.2727
   -0.5455
    0.5238
   -0.2727
    0.2683
    0.6667
    0.5238
    0.2683
    0.3479
   -1.0000
    0.5366
   -0.5345
    0.9046
         0
         0
         0
         0
         0
         0
         0
         0
irowOut =
           1
           2
           2
           3
           4
           4
           5
```

```
            5
            5
            6
            6
            6
            7
            7
            7
            7
            1
            2
            3
            3
            4
            4
            5
            5
            6
            6
            6
            6
            7
            7
            7
            7
            0
            0
            0
            0
            0
            0
            0
            0
icolOut =
            1
            1
            2
            3
            2
            4
            1
            4
            5
            2
            5
            6
            1
            2
            3
            7
            1
            2
            2
            3
            3
            4
            3
            5
            2
            4
            5
            6
            1
            5
            6
            7
            0
            0
            0
            0
```

```
           0
           0
           0
           0
ipivOut =
           3
           4
           5
           6
           1
           2
           7
istr =
          17
          18
          19
          21
          23
          25
          29
          33
nnzc =
          16
npivm =
           0
ifail =
           0
```