NAG Toolbox for MATLAB

f11jc

1 Purpose

f11jc solves a real sparse symmetric system of linear equations, represented in symmetric co-ordinate storage format, using a conjugate gradient or Lanczos method, with incomplete Cholesky preconditioning.

2 Syntax

```
[x, rnorm, itn, ifail] = f11jc(method, nnz, a, irow, icol, ipiv, istr,
b, tol, maxitn, x, 'n', n, 'la', la)
```

3 Description

f11jc solves a real sparse symmetric linear system of equations

$$Ax = b,$$

using a preconditioned conjugate gradient method (see Meijerink and Van der Vorst 1977), or a preconditioned Lanczos method based on the algorithm SYMMLQ (see Paige and Saunders 1975). The conjugate gradient method is more efficient if A is positive-definite, but may fail to converge for indefinite matrices. In this case the Lanczos method should be used instead. For further details see Barrett *et al.* 1994.

f11jc uses the incomplete Cholesky factorization determined by f11ja as the preconditioning matrix. A call to f11jc must always be preceded by a call to f11ja. Alternative preconditioners for the same storage scheme are available by calling f11je.

The matrix A, and the preconditioning matrix M, are represented in symmetric co-ordinate storage (SCS) format (see Section 2.1.2 in the F11 Chapter Introduction) in the arrays **a**, **irow** and **icol**, as returned from f11ja. The array **a** holds the nonzero entries in the lower triangular parts of these matrices, while **irow** and **icol** hold the corresponding row and column indices.

4 References

Barrett R, Berry M, Chan T F, Demmel J, Donato J, Dongarra J, Eijkhout V, Pozo R, Romine C and Van der Vorst H 1994 *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods* SIAM, Philadelphia

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

Paige C C and Saunders M A 1975 Solution of sparse indefinite systems of linear equations *SIAM J. Numer. Anal.* **12** 617–629

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*

5 Parameters

5.1 Compulsory Input Parameters

1: **method – string**

Specifies the iterative method to be used.

method = 'CG'

Conjugate gradient method.

method = 'SYMMLQ'

Lanczos method (SYMMLQ).

Constraint: **method** = 'CG' or 'SYMMLQ'.

2: **nnz – int32 scalar**

The number of nonzero elements in the lower triangular part of the matrix A. This **must** be the same value as was supplied in the preceding call to f11ja.

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

3: **a(la) – double array**

The values returned in the array **a** by a previous call to f11da.

4: **irow(la) – int32 array**

5: **icol(la) – int32 array**

6: **ipiv(n) – int32 array**

7: **istr(n + 1) – int32 array**

The values returned in arrays **irow**, **icol**, **ipiv** and **istr** by a previous call to f11ja.

8: **b(n) – double array**

The right-hand side vector b.

9: **tol – double scalar**

The required tolerance. Let x_k denote the approximate solution at iteration k, and r_k the corresponding residual. The algorithm is considered to have converged at iteration k if

$$\|r_k\|_\infty \leq \tau \times \left(\|b\|_\infty + \|A\|_\infty \|x_k\|_\infty\right).$$

If $\mathbf{tol} \leq 0.0$, $\tau = \max\left(\sqrt{\epsilon}, \sqrt{n}\epsilon\right)$ is used, where ϵ is the ***machine precision***. Otherwise $\tau = \max\left(\mathbf{tol}, 10\epsilon, \sqrt{n}\epsilon\right)$ is used.

Constraint: **tol** < 1.0.

10: **maxitn – int32 scalar**

The maximum number of iterations allowed.

Constraint: **maxitn** ≥ 1.

11: **x(n) – double array**

An initial approximation to the solution vector x.

5.2 Optional Input Parameters

1: **n – int32 scalar**

Default: The dimension of the arrays **ipiv**, **b**, **x**. (An error is raised if these dimensions are not equal.)

n, the order of the matrix A. This **must** be the same value as was supplied in the preceding call to f11ja.

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.)

This **must** be the same value as was supplied in the preceding call to f11ja.

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

5.3 Input Parameters Omitted from the MATLAB Interface

work, lwork

5.4 Output Parameters

1: **x(n) – double array**

An improved approximation to the solution vector x.

2: **rnorm – double scalar**

The final value of the residual norm $\|r_k\|_\infty$, where k is the output value of **itn**.

3: **itn – int32 scalar**

The number of iterations carried out.

4: **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, **method** $\neq$ 'CG' or 'SYMMLQ',
or **n** < 1,
or **nnz** < 1,
or **nnz** $>$ **n** $\times$ (**n** $+ 1)/2$,
or **la** too small,
or **tol** ≥ 1.0,
or **maxitn** < 1,
or **lwork** too small.

ifail $= 2$

On entry, the SCS representation of A is invalid. Further details are given in the error message. Check that the call to f11jc has been preceded by a valid call to f11ja, and that the arrays **a**, **irow**, and **icol** have not been corrupted between the two calls.

ifail $= 3$

On entry, the SCS representation of the preconditioning matrix M is invalid. Further details are given in the error message. Check that the call to f11jc has been preceded by a valid call to f11ja, and that the arrays **a**, **irow**, **icol**, **ipiv** and **istr** have not been corrupted between the two calls.

ifail $= 4$

The required accuracy could not be obtained. However, a reasonable accuracy has been obtained and further iterations could not improve the result.

ifail = 5

Required accuracy not obtained in **maxitn** iterations.

ifail = 6

The preconditioner appears not to be positive-definite.

ifail = 7

The matrix of the coefficients appears not to be positive-definite (conjugate gradient method only).

ifail = 8 (f11gd, f11ge or f11gf)

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

7 Accuracy

On successful termination, the final residual $r_k = b - Ax_k$, where $k =$ **itn**, satisfies the termination criterion

$$\|r_k\|_\infty \leq \tau \times \left(\|b\|_\infty + \|A\|_\infty \|x_k\|_\infty\right).$$

The value of the final residual norm is returned in **rnorm**.

8 Further Comments

The time taken by f11jc for each iteration is roughly proportional to the value of **nnzc** returned from the preceding call to f11ja. One iteration with the Lanczos method (SYMMLQ) requires a slightly larger number of operations than one iteration with the conjugate gradient method.

The number of iterations required to achieve a prescribed accuracy cannot be easily determined *a priori*, as it can depend dramatically on the conditioning and spectrum of the preconditioned matrix of the coefficients $\bar{A} = M^{-1}A$.

Some illustrations of the application of f11jc 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.

9 Example

```
method = 'CG    ';
nnz = int32(16);
a = [4;
     1;
     5;
     2;
     2;
     3;
     -1;
     1;
     4;
     1;
     -2;
     3;
     2;
     -1;
     -2;
     5;
     0.5;
     0.3333333333333333;
     0.3333333333333333;
     -1;
```

```
     0.3333333333333333;
     0.6666666666666666;
     0.3333333333333333;
     -0.3333333333333333;
     0.3333333333333333;
     0.3333333333333333;
     -0.6666666666666666;
     0.4285714285714286;
     0.6666666666666666;
     0.5555555555555555;
     -0.4285714285714286;
     0.762096774193548;
     0.762096774193548;
     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(1);
     int32(2);
     int32(3);
     int32(4);
     int32(4);
     int32(5);
     int32(5);
     int32(5);
     int32(5);
     int32(6);
     int32(6);
     int32(6);
     int32(7);
     int32(7);
     int32(7);
     int32(7);
     int32(7);
     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(1);
     int32(2);
     int32(3);
     int32(1);
     int32(4);
     int32(2);
     int32(3);
     int32(4);
     int32(5);
     int32(2);
     int32(3);
     int32(6);
     int32(4);
     int32(5);
     int32(6);
     int32(7);
     int32(7);
     int32(0);
     int32(0);
     int32(0);
     int32(0);
     int32(0);
     int32(0);
     int32(0)];
ipiv = [int32(3);
     int32(4);
     int32(6);
     int32(7);
     int32(2);
     int32(5);
     int32(1)];
istr = [int32(17);
     int32(18);
     int32(19);
     int32(20);
     int32(22);
     int32(26);
     int32(29);
     int32(33)];
b = [15;
     18;
     -8;
     21;
     11;
     10;
     29];
tol = 1e-06;
maxitn = int32(100);
x = [0;
     0;
     0;
     0;
     0;
     0;
     0];
[xOut, rnorm, itn, ifail] = ...
    f11jc(method, nnz, a, irow, icol, ipiv, istr, b, tol, maxitn, x)
```

```
xOut =
    1.0000
    2.0000
    3.0000
    4.0000
    5.0000
```

```
    6.0000
    7.0000
rnorm =
   7.1054e-15
itn =
           1
ifail =
           0
```