

## NAG Toolbox

### nag\_sparse\_real\_symm\_solve\_ichol (f11jc)

#### 1 Purpose

nag\_sparse\_real\_symm\_solve\_ichol (f11jc) solves a real sparse symmetric system of linear equations, represented in symmetric coordinate storage format, using a conjugate gradient or Lanczos method, with incomplete Cholesky preconditioning.

#### 2 Syntax

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

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

#### 3 Description

nag\_sparse\_real\_symm\_solve\_ichol (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).

nag\_sparse\_real\_symm\_solve\_ichol (f11jc) uses the incomplete Cholesky factorization determined by nag\_sparse\_real\_symm\_precon\_ichol (f11ja) as the preconditioning matrix. A call to nag\_sparse\_real\_symm\_solve\_ichol (f11jc) must always be preceded by a call to nag\_sparse\_real\_symm\_precon\_ichol (f11ja). Alternative preconditioners for the same storage scheme are available by calling nag\_sparse\_real\_symm\_solve\_jacssor (f11je).

The matrix  $A$ , and the preconditioning matrix  $M$ , are represented in symmetric coordinate storage (SCS) format (see Section 2.1.2 in the F11 Chapter Introduction) in the arrays **a**, **irow** and **icol**, as returned from nag\_sparse\_real\_symm\_precon\_ichol (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** – CHARACTER(\*)

Specifies the iterative method to be used.

**method** = 'CG'

Conjugate gradient method.

**method** = 'SYMMLQ'

Lanczos method (SYMMLQ).

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

2: **nz** – INTEGER

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 `nag_sparse_real_symm_precon_ichol` (f11ja).

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

3: **a(la)** – REAL (KIND=`nag_wp`) array

The values returned in the array **a** by a previous call to `nag_sparse_real_symm_precon_ichol` (f11ja).

4: **irow(la)** – INTEGER array

5: **icol(la)** – INTEGER array

6: **ipiv(n)** – INTEGER array

7: **istr(n + 1)** – INTEGER array

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

8: **b(n)** – REAL (KIND=`nag_wp`) array

The right-hand side vector  $b$ .

9: **tol** – REAL (KIND=`nag_wp`)

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

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

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

*Constraint:* **tol**  $< 1.0$ .

10: **maxitn** – INTEGER

The maximum number of iterations allowed.

*Constraint:* **maxitn**  $\geq 1$ .

11: **x(n)** – REAL (KIND=`nag_wp`) array

An initial approximation to the solution vector  $x$ .

## 5.2 Optional Input Parameters

1: **n** – INTEGER

*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 `nag_sparse_real_symm_precon_ichol` (f11ja).

*Constraint:*  $n \geq 1$ .

2: **la** – INTEGER

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

The dimension of the arrays **a**, **irow** and **icol**. this **must** be the same value as was supplied in the preceding call to `nag_sparse_real_symm_precon_ichol` (f11ja).

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

## 5.3 Output Parameters

1: **x(n)** – REAL (KIND=nag\_wp) array

An improved approximation to the solution vector *x*.

2: **rnorm** – REAL (KIND=nag\_wp)

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

3: **itn** – INTEGER

The number of iterations carried out.

4: **ifail** – INTEGER

**ifail** = 0 unless the function detects an error (see Section 5).

## 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**  $\geq 1.0$ ,

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 `nag_sparse_real_symm_solve_ichol` (f11jc) has been preceded by a valid call to `nag_sparse_real_symm_precon_ichol` (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 `nag_sparse_real_symm_solve_ichol` (f11jc) has been preceded by a valid call to `nag_sparse_real_symm_precon_ichol` (f11ja) and that the arrays **a**, **irow**, **icol**, **ipiv** and **istr** have not been corrupted between the two calls.

**ifail** = 4 (*warning*)

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 (`nag_sparse_real_symm_basic_setup` (f11gd), `nag_sparse_real_symm_basic_solver` (f11ge) or `nag_sparse_real_symm_basic_diag` (f11gf))

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

**ifail** = -99

An unexpected error has been triggered by this routine. Please contact NAG.

**ifail** = -399

Your licence key may have expired or may not have been installed correctly.

**ifail** = -999

Dynamic memory allocation failed.

## 7 Accuracy

On successful termination, the final residual  $r_k = b - Ax_k$ , where  $k = \mathbf{itn}$ , satisfies the termination criterion

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

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

## 8 Further Comments

The time taken by `nag_sparse_real_symm_solve_ichol` (f11jc) for each iteration is roughly proportional to the value of **nnzc** returned from the preceding call to `nag_sparse_real_symm_precon_ichol` (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 `nag_sparse_real_symm_solve_ichol` (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

This example solves a symmetric positive definite system of equations using the conjugate gradient method, with incomplete Cholesky preconditioning.

### 9.1 Program Text

```
function f11jc_example

fprintf('f11jc example results\n\n');

% Solve Ax = b by CG with Incomplete Cholesky preconditioning

% Matrix A and RHS b

n    = nag_int(7);
nz   = nag_int(16);
a = zeros(3*nz, 1);
irow = zeros(3*nz, 1, nag_int_name);
icol = irow;
a(1:nz)    = [4; 1; 5; 2; 2; 3;-1; 1; 4; 1;-2; 3; 2;-1;-2; 5];
irow(1:nz) = [1; 2; 2; 3; 4; 4; 5; 5; 5; 6; 6; 6; 7; 7; 7; 7];
icol(1:nz) = [1; 1; 2; 3; 2; 4; 1; 4; 5; 2; 5; 6; 1; 2; 3; 7];

b = [15; 18; -8; 21; 11; 10; 29];

% Calculate incomplete Cholesky factorization
lfill = nag_int(1);
dtol   = 0;
mic    = 'N';
dscale = 0;
ipiv = zeros(n,1,nag_int_name);

[a, irow, icol, ipiv, istr, nnzc, npivm, ifail] = ...
f11ja( ...
    nz, a, irow, icol, lfill, dtol, mic, dscale, ipiv);

% Solve Ax = b
method = 'CG';
tol    = 1e-06;
maxitn = nag_int(100);
x      = zeros(n,1);

[x, rnorm, itn, ifail] = f11jc( ...
    method, nz, a, irow, icol, ipiv, istr, ...
    b, tol, maxitn, x);

fprintf('\nConverged in %d iterations\n', itn);
fprintf('Final residual norm = %16.3e\n\n', rnorm);
disp('Solution');
disp(x);
```

### 9.2 Program Results

```
f11jc example results

Converged in 1 iterations
Final residual norm =          7.105e-15

Solution
  1.0000
  2.0000
```

3.0000  
4.0000  
5.0000  
6.0000  
7.0000

---