NAG Toolbox

nag_sparse_complex_herm_solve_ilu (f11jq)

1 Purpose

nag_sparse_complex_herm_solve_ilu (f11jq) solves a complex sparse Hermitian 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_complex_herm_solve_ilu(method, nnz, a, irow, icol, ipiv, istr, b, tol, maxitn, x, 'n', n, 'la', la)
[x, rnorm, itn, ifail] = f11jq(method, nnz, a, irow, icol, ipiv, istr, b, tol, maxitn, x, 'n', n, 'la', la)
```

3 Description

nag_sparse_complex_herm_solve_ilu (f11jq) solves a complex sparse Hermitian 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_complex_herm_solve_ilu (f11jq) uses the incomplete Cholesky factorization determined by nag_sparse_complex_herm_precon_ilu (f11jn) as the preconditioning matrix. A call to nag_sparse_complex_herm_solve_ilu (f11jq) must always be preceded by a call to nag_sparse_complex_herm_precon_ilu (f11jn). Alternative preconditioners for the same storage scheme are available by calling nag_sparse_complex_herm_solve_jacssor (f11js).

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_complex_herm_precon_ilu (f11jn). 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

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_complex_herm_precon_ilu (f11jn).

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

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

The values returned in the array **a** by a previous call to nag_sparse_complex_herm_precon_ilu (f11jn).

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_complex_herm_precon_ilu (f11jn).

8: **b**(**n**) – COMPLEX (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

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

If $\mathbf{tol} \leq 0.0$, $\tau = \max\sqrt{\epsilon}, 10\epsilon, \sqrt{n}\epsilon$ is used, where ϵ 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** ≥ 1.

11: **x**(**n**) – COMPLEX (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_complex_herm_precon_ilu (f11jn).

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

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_complex_herm_precon_ilu (f11jn).

Constraint: $\mathbf{la} \geq 2 \times \mathbf{nnz}$.

5.3 Output Parameters

1: **x**(**n**) – COMPLEX (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** $> \mathbf{n} \times (\mathbf{n}+1)/2$,
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 nag_sparse_complex_herm_solve_ilu (f11jq) has been preceded by a valid call to nag_sparse_complex_herm_precon_ilu (f11jn), and that the arrays **a**, **irow**, and **icol** have not been corrupted between the two calls.

ifail $= 3$

On entry, the SCS representation of M is invalid. Further details are given in the error message. Check that the call to nag_sparse_complex_herm_solve_ilu (f11jq) has been preceded by a valid call to nag_sparse_complex_herm_precon_ilu (f11jn), 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 $= 8$

A serious error has occurred in an internal call to an auxiliary function. 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 $k =$ **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 nag_sparse_complex_herm_solve_ilu (f11jq) for each iteration is roughly proportional to the value of **nnzc** returned from the preceding call to nag_sparse_complex_herm_precon_ilu (f11jn). 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 easily be determined *a priori*, as it can depend dramatically on the conditioning and spectrum of the preconditioned matrix of the coefficients $\bar{A} = M^{-1}A$.

9 Example

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

9.1 Program Text

```
     function f11jq_example

fprintf('f11jq example results\n\n');

% Solve sparse Hermitian system Ax = b using CG method with
% Incomplete Cholesky preconditioning (IC)

% Define A and b
n  = nag_int(9);
nz = nag_int(23);
a    = zeros(3*nz,1);
irow = zeros(3*nz, 1, nag_int_name);
icol = zeros(3*nz, 1, nag_int_name);
a(1:nz) = [ 6 + 0.i; -1 + 1.i;  6 + 0.i;  0 + 1.i;
            5 + 0.i;  5 + 0.i;  2 - 2.i;  4 + 0.i;
            1 + 1.i;  2 + 0.i;  6 + 0.i; -4 + 3.i;
            0 + 1.i; -1 + 0.i;  6 + 0.i; -1 - 1.i;
            0 - 1.i;  9 + 0.i;  1 + 3.i;  1 + 2.i;
           -1 + 0.i;  1 + 4.i;  9 + 0.i];
b       = [ 8 + 54i;-10 - 92i; 25 + 27i; 26 - 28i;
           54 + 12i; 26 - 22i; 47 + 65i; 71 - 57i;
           60 + 70i];
irow(1:nz) = nag_int([1;2;2;3;3;4;5;5;6;6;6;7;7;7;7;8;8;8;9;9;9;9;9]);
icol(1:nz) = nag_int([1;1;2;2;3;4;1;5;3;4;6;2;5;6;7;4;6;8;1;5;6;8;9]);

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

[a, irow, icol, ipiv, istr, nnzc, npivm, ifail] = ...
  f11jn( ...
         nz, a, irow, icol, lfill, dtol, mic, dscale, ipiv);

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

[x, rnorm, itn, ifail] = ...
  f11jq( ...
         method, nz, a, irow, icol, ipiv, istr, b, tol, maxitn, x);

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

9.2 Program Results

```
     f11jq example results

Converged in 5 iterations
Final redidual norm =        3.197e-14

Solution
   1.0000 + 9.0000i
   2.0000 - 8.0000i
   3.0000 + 7.0000i
```

```
   4.0000 - 6.0000i
   5.0000 + 5.0000i
   6.0000 - 4.0000i
   7.0000 + 3.0000i
   8.0000 - 2.0000i
   9.0000 + 1.0000i
```