

NAG Toolbox for MATLAB

f11jq

1 Purpose

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

3 Description

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.

f11jq uses the incomplete Cholesky factorization determined by f11jn as the preconditioning matrix. A call to f11jq must always be preceded by a call to f11jn. Alternative preconditioners for the same storage scheme are available by calling f11js.

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 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** – 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 f11jn.

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

3: **a(la)** – complex array

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

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 f11jn.

8: **b(n)** – complex 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

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

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

Constraint: **tol** < 1.0 .

10: **maxitn** – int32 scalar

The maximum number of iterations allowed.

Constraint: **maxitn** ≥ 1 .

11: **x(n)** – complex 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 f11jn.

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 f11jn.

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

5.3 Input Parameters Omitted from the MATLAB Interface

work, lwork

5.4 Output Parameters1: **x(n) – complex 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 f11jq has been preceded by a valid call to 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 f11jq has been preceded by a valid call to f11jn, 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

A serious error has occurred in an internal call to an auxiliary function. 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 $k = \mathbf{itn}$, 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 f11jq for each iteration is roughly proportional to the value of **nnzc** returned from the preceding call to 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

```
method = 'CG      ';
nnz = int32(23);
a = [complex(6, +0);
     complex(-1, +1);
     complex(6, +0);
     complex(0, +1);
     complex(5, +0);
     complex(5, +0);
     complex(2, -2);
     complex(4, +0);
     complex(1, +1);
     complex(2, +0);
     complex(6, +0);
     complex(-4, +3);
     complex(0, +1);
     complex(-1, +0);
     complex(6, +0);
     complex(-1, -1);
     complex(0, -1);
     complex(9, +0);
     complex(1, +3);
     complex(1, +2);
     complex(-1, +0);
     complex(1, +4);
     complex(9, +0);
     complex(0.2, -0);
```

[illegible]

```
int32(7);  
int32(8);  
int32(8);  
int32(8);  
int32(9);  
int32(9);  
int32(9);  
int32(9);  
int32(1);  
int32(2);  
int32(2);  
int32(3);  
int32(3);  
int32(4);  
int32(5);  
int32(5);  
int32(6);  
int32(6);  
int32(6);  
int32(7);  
int32(7);  
int32(7);  
int32(7);  
int32(8);  
int32(8);  
int32(8);  
int32(9);  
int32(9);  
int32(9);  
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);  
int32(0);  
int32(0);  
int32(0);  
int32(0);  
int32(0);  
icol = [int32(1);  
int32(1);  
int32(2);  
int32(2);
```

[illegible]

```

        int32(0);
        int32(0);
        int32(0);
        int32(0);
        int32(0);
        int32(0)];
    ipiv = [int32(3);
        int32(2);
        int32(1);
        int32(4);
        int32(8);
        int32(9);
        int32(6);
        int32(5);
        int32(7)];
    istr = [int32(24);
        int32(25);
        int32(27);
        int32(29);
        int32(30);
        int32(32);
        int32(35);
        int32(40);
        int32(43);
        int32(47)];
    b = [complex(8, +54);
        complex(-10, -92);
        complex(25, +27);
        complex(26, -28);
        complex(54, +12);
        complex(26, -22);
        complex(47, +65);
        complex(71, -57);
        complex(60, +70)];
    tol = 1e-06;
    maxitn = int32(100);
    x = [complex(0, +0);
        complex(0, +0);
        complex(0, +0);
        complex(0, +0);
        complex(0, +0);
        complex(0, +0);
        complex(0, +0);
        complex(0, +0);
        complex(0, +0)];
    [xOut, rnorm, itn, ifail] = ...
        f11jq(method, nnz, a, irow, icol, ipiv, istr, b, tol, maxitn, x)

```

```

xOut =
    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
rnorm =
    4.2633e-14
itn =
    5
ifail =
    0

```