NAG Toolbox for MATLAB

f11jp

1 Purpose

f11jp solves a system of complex linear equations involving the incomplete Cholesky preconditioning matrix generated by f11jn.

2 Syntax

```
[x, ifail] = f11jp(a, irow, icol, ipiv, istr, check, y, 'n', n, 'la', la)
```

3 Description

f11jp solves a system of linear equations

$$Mx = y$$

involving the preconditioning matrix $M = PLDL^{\mathrm{H}}P^{\mathrm{T}}$, corresponding to an incomplete Cholesky decomposition of a complex sparse Hermitian matrix stored in symmetric co-ordinate storage (SCS) format (see Section 2.1.2 in the F11 Chapter Introduction), as generated by f11jn.

In the above decomposition L is a complex lower triangular sparse matrix with unit diagonal, D is a real diagonal matrix and P is a permutation matrix. L and D are supplied to f11jp through the matrix

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

which is a lower triangular n by n complex sparse matrix, stored in SCS format, as returned by f11jn. The permutation matrix P is returned from f11jn via the array **ipiv**.

f11jp may also be used in combination with f11jn to solve a sparse complex Hermitian positive-definite system of linear equations directly (see f11jn). This is illustrated in Section 9.

4 References

None.

5 Parameters

5.1 Compulsory Input Parameters

1: **a**(**la**) – **complex array**

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

2: **irow**(**la**) – **int32 array**

3: **icol**(**la**) – **int32 array**

4: **ipiv**(**n**) – **int32 array**

5: **istr**(**n** + **1**) – **int32 array**

The values returned in arrays **irow**, **icol**, **ipiv** and **istr** by a previous call to f11jn.

6: **check** – **string**

Specifies whether or not the input data should be checked.

check = 'C'

Checks are carried out on the values of **n**, **irow**, **icol**, **ipiv** and **istr**.

check = 'N'

None of these checks are carried out.

Constraint: **check** = 'C' or 'N'.

7: **y(n) – complex array**

The right-hand side vector *y*.

5.2 Optional Input Parameters

1: **n – int32 scalar**

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

n, the order of the matrix *M*. 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.

5.3 Input Parameters Omitted from the MATLAB Interface

None.

5.4 Output Parameters

1: **x(n) – complex array**

The solution vector *x*.

2: **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, **check** $\neq$ 'C' or 'N'.

ifail = 2

On entry, **n** < 1.

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 f11jp 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.

7 Accuracy

The computed solution x is the exact solution of a perturbed system of equations $(M + \delta M)x = y$, where

$$|\delta M| \leq c(n)\epsilon P|L||D|\left|L^{\mathrm{H}}\right|P^{\mathrm{T}},$$

$c(n)$ is a modest linear function of n, and ϵ is the ***machine precision***.

8 Further Comments

8.1 Timing

The time taken for a call to f11jp is proportional to the value of **nnzc** returned from f11jn.

9 Example

```
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);
     complex(0.2, -0);
     complex(-0.2, -0.2);
     complex(0.1162790697674419, -0);
     complex(0.1162790697674419, +0.4651162790697674);
     complex(0.1423841059602649, -0);
     complex(0.1666666666666667, -0);
     complex(0.1423841059602649, -0.4271523178807947);
     complex(0.2185238784370478, -0);
     complex(0.2, +0.2);
     complex(0.4, +0);
     complex(0.04651162790697674, +0.06976744186046512);
     complex(-0.1887417218543046, +0.01655629139072848);
     complex(-0.1666666666666667, +0);
     complex(0.05209840810419682, +0.1201157742402315);
     complex(0.2357208646509671, -0);
     complex(0.1423841059602649, -0.2847682119205298);
     complex(0, -0.1666666666666667);
     complex(0.2192474674384949, -0.4681620839363242);
     complex(0.1006333647931046, -0.06964738523816563);
     complex(0.5449841356402756, -0);
     complex(0, -0.2);
     complex(-0.6666666666666666, -0.5);
     complex(-0.2185238784370478, +0.2185238784370478);
     complex(-0.1260361825273188, -0.1113107353626781);
     complex(0.1135355212929453, +0.5451700461057954);
     complex(1.971123543351331, -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);
      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);
      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);
      complex(0, +0);
      complex(0, +0);
      complex(0, +0);
      complex(0, +0)];
irow = [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(9);
      int32(9);
      int32(1);
      int32(2);
      int32(3);
      int32(3);
      int32(4);
      int32(4);
      int32(5);
      int32(6);
      int32(6);
      int32(7);
      int32(7);
      int32(7);
      int32(7);
      int32(7);
      int32(7);
      int32(7);
      int32(8);
```

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

```
     int32(4);
     int32(6);
     int32(1);
     int32(2);
     int32(3);
     int32(4);
     int32(5);
     int32(6);
     int32(7);
     int32(4);
     int32(5);
     int32(6);
     int32(7);
     int32(8);
     int32(1);
     int32(5);
     int32(6);
     int32(7);
     int32(8);
     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);
     int32(0);
     int32(0);
     int32(0)];
ipiv = [int32(3);
     int32(4);
     int32(8);
     int32(9);
     int32(7);
     int32(1);
     int32(6);
     int32(5);
     int32(2)];
istr = [int32(24);
     int32(25);
     int32(26);
     int32(28);
     int32(30);
     int32(31);
     int32(33);
     int32(40);
     int32(45);
     int32(51)];
check = 'C';
```

```
y = [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)];
[x, ifail] = f11jp(a, irow, icol, ipiv, istr, check, y)
```

```
x =
   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
ifail =
           0
```