# NAG Toolbox for MATLAB

# f11dp

## 1 Purpose

f11dp solves a system of complex linear equations involving the incomplete $LU$ preconditioning matrix generated by f11dn.

## 2 Syntax

```
[x, ifail] = f11dp(trans, a, irow, icol, ipivp, ipivq, istr, idiag,
check, y, 'n', n, 'la', la)
```

## 3 Description

f11dp solves a system of complex linear equations

$$Mx = y, \qquad \text{or} \qquad M^{\mathrm{T}}x = y,$$

according to the value of the parameter **trans**, where the matrix $M = PLDUQ$ corresponds to an incomplete $LU$ decomposition of a complex sparse matrix stored in co-ordinate storage (CS) format (see Section 2.1.1 in the F11 Chapter Introduction), as generated by f11dn.

In the above decomposition $L$ is a lower triangular sparse matrix with unit diagonal elements, $D$ is a diagonal matrix, $U$ is an upper triangular sparse matrix with unit diagonal elements and, $P$ and $Q$ are permutation matrices. $L$, $D$ and $U$ are supplied to f11dp through the matrix

$$C = L + D^{-1} + U - 2I$$

which is an **n** by **n** sparse matrix, stored in CS format, as returned by f11dn. The permutation matrices $P$ and $Q$ are returned from f11dn via the arrays **ipivp** and **ipivq**.

It is envisaged that a common use of f11dp will be to carry out the preconditioning step required in the application of f11bs to sparse complex linear systems. f11dp is used for this purpose by the Black Box function f11dq.

f11dp may also be used in combination with f11dn to solve a sparse system of complex linear equations directly (see Section 8.5 of the document for f11dn). This use of f11dp is illustrated in Section 9.

## 4 References

None.

## 5 Parameters

### 5.1 Compulsory Input Parameters

1: **trans – string**

Specifies whether or not the matrix $M$ is transposed.

**trans** = 'N'

$Mx = y$ is solved.

**trans** = 'T'

$M^{\mathrm{T}}x = y$ is solved.

*Constraint*: **trans** = 'N' or 'T'.

2: **a**(**la**) – **complex array**

The values returned in the array **a** by a previous call to f11dn.

3: **irow**(**la**) – **int32 array**
4: **icol**(**la**) – **int32 array**
5: **ipivp**(**n**) – **int32 array**
6: **ipivq**(**n**) – **int32 array**
7: **istr**(**n** + **1**) – **int32 array**
8: **idiag**(**n**) – **int32 array**

The values returned in arrays **irow**, **icol**, **ipivp**, **ipivq**, **istr** and **idiag** by a previous call to f11da.

9: **check** – **string**

Specifies whether or not the CS representation of the matrix $M$ should be checked.

**check** = 'C'

Checks are carried on the values of **n**, **irow**, **icol**, **ipivp**, **ipivq**, **istr** and **idiag**.

**check** = 'N'

None of these checks are carried out.

See also Section 8.2.

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

10: **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 **ipivp**, **ipivq**, **idiag**, **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 f11dn.

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

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 f11dn.

## 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, **trans** ≠ 'N' or 'T',
or **check** ≠ 'C' or 'N'.

**ifail** = 2

On entry, **n** < 1.

**ifail** = 3

On entry, the CS representation of the preconditioning matrix $M$ is invalid. Further details are given in the error message. Check that the call to f11dp has been preceded by a valid call to f11dn and that the arrays **a**, **irow**, **icol**, **ipivp**, **ipivq**, **istr** and **idiag** have not been corrupted between the two calls.

## 7 Accuracy

If **trans** = 'N' the computed solution $x$ is the exact solution of a perturbed system of equations $(M + \delta M)x = y$, where

$$|\delta M| \leq c(n)\epsilon P|L||D||U|Q,$$

$c(n)$ is a modest linear function of $n$, and $\epsilon$ is the ***machine precision***. An equivalent result holds when **trans** = 'T'.

## 8 Further Comments

### 8.1 Timing

The time taken for a call to f11dp is proportional to the value of **nnzc** returned from f11dn.

### 8.2 Use of check

It is expected that a common use of f11dp will be to carry out the preconditioning step required in the application of f11bs to sparse complex linear systems. In this situation f11dp is likely to be called many times with the same matrix $M$. In the interests of both reliability and efficiency, you are recommended to set **check** to 'C' for the first of such calls, and to 'N' for all subsequent calls.

## 9 Example

```
trans = 'N';
a = [complex(1, +2);
     complex(1, +3);
     complex(-1, -3);
     complex(2, +0);
     complex(0, +4);
     complex(3, +4);
     complex(-2, +0);
     complex(1, -1);
     complex(-2, -1);
     complex(1, +0);
     complex(1, +3);
     complex(0.1, -0.3);
     complex(0.7, -0.09999999999999998);
     complex(0.2, -0.6);
     complex(0, -0.25);
     complex(-0.75, +0.25);
```

```
     complex(0.04999999999999999, +0.35);
     complex(0, +0.5);
     complex(0.06666666666666667, -0.2);
     complex(0.1466666666666667, +0.02666666666666666);
     complex(0.1, -0.3);
     complex(0.75, -0.25);
     complex(0.3666666666666667, -0.4333333333333333);
     complex(-0.3015768725361367, +0.2385019710906702);
     complex(0, +0);
     complex(0, +0);
     complex(0, +0);
     complex(0, +0);
     complex(0, +0);
     complex(0, +0)];
irow = [int32(1);
     int32(1);
     int32(2);
     int32(2);
     int32(2);
     int32(3);
     int32(3);
     int32(4);
     int32(4);
     int32(4);
     int32(4);
     int32(1);
     int32(1);
     int32(2);
     int32(2);
     int32(2);
     int32(2);
     int32(3);
     int32(3);
     int32(3);
     int32(4);
     int32(4);
     int32(4);
     int32(4);
     int32(0);
     int32(0);
     int32(0);
     int32(0);
     int32(0);
     int32(0)];
icol = [int32(2);
     int32(3);
     int32(1);
     int32(3);
     int32(4);
     int32(1);
     int32(4);
     int32(1);
     int32(2);
     int32(3);
     int32(4);
     int32(1);
     int32(4);
     int32(1);
     int32(2);
     int32(3);
     int32(4);
     int32(2);
     int32(3);
     int32(4);
     int32(1);
     int32(2);
     int32(3);
     int32(4);
     int32(0);
     int32(0);
```

```
     int32(0);
     int32(0);
     int32(0);
     int32(0)];
ipivp = [int32(1);
     int32(2);
     int32(3);
     int32(4)];
ipivq = [int32(3);
     int32(4);
     int32(1);
     int32(2)];
istr = [int32(12);
     int32(14);
     int32(18);
     int32(21);
     int32(25)];
idiag = [int32(12);
     int32(15);
     int32(19);
     int32(24)];
check = 'C';
y = [complex(5, +14);
     complex(21, +5);
     complex(-21, +18);
     complex(14, +4)];
[x, ifail] = f11dp(trans, a, irow, icol, ipivp, ipivq, istr, idiag,
check, y)
```

```
x =
   1.0000 + 4.0000i
   2.0000 + 3.0000i
   3.0000 - 2.0000i
   4.0000 - 1.0000i
ifail =
           0
```