NAG Toolbox for MATLAB

f11dn

1 Purpose

f11dn computes an incomplete LU factorization of a complex sparse non-Hermitian matrix, represented in co-ordinate storage format. This factorization may be used as a preconditioner in combination with f11dq or f11bs.

2 Syntax

```
[a, irow, icol, ipivp, ipivq, istr, idiag, nnzc, npivm, ifail] =
f11dn(nnz, a, irow, icol, lfill, dtol, milu, ipivp, ipivq, 'n', n, 'la',
la, 'pstrat', pstrat)
```

3 Description

f11dn computes an incomplete LU factorization (see Meijerink and Van der Vorst 1977 and Meijerink and Van der Vorst 1981) of a complex sparse non-Hermitian n by n matrix A. The factorization is intended primarily for use as a preconditioner with one of the iterative solvers f11dq or f11bs.

The decomposition is written in the form

$$A = M + R,$$

where

$$M = PLDUQ$$

and L is lower triangular with unit diagonal elements, D is diagonal, U is upper triangular with unit diagonals, P and Q are permutation matrices, and R is a remainder matrix.

The amount of fill-in occurring in the factorization can vary from zero to complete fill, and can be controlled by specifying either the maximum level of fill **lfill**, or the drop tolerance **dtol**.

The parameter **pstrat** defines the pivoting strategy to be used. The options currently available are no pivoting, user-defined pivoting, partial pivoting by columns for stability, and complete pivoting by rows for sparsity and by columns for stability. The factorization may optionally be modified to preserve the row-sums of the original matrix.

The sparse matrix A is represented in co-ordinate storage (CS) format (see Section 2.1.1 in the F11 Chapter Introduction). The array **a** stores all the nonzero elements of the matrix A, while arrays **irow** and **icol** store the corresponding row and column indices respectively. Multiple nonzero elements may not be specified for the same row and column index.

The preconditioning matrix M is returned in terms of the CS representation of the matrix

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

Further algorithmic details are given in Section 8.3.

4 References

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

Meijerink J and Van der Vorst H 1981 Guidelines for the usage of incomplete decompositions in solving sets of linear equations as they occur in practical problems *J. Comput. Phys.* **44** 134–155

5 Parameters

5.1 Compulsory Input Parameters

1: **nnz – int32 scalar**

the number of nonzero elements in the matrix A.

Constraint: $1 \leq \mathbf{nnz} \leq \mathbf{n}^2$.

2: **a(la) – complex array**

The nonzero elements in the matrix A, ordered by increasing row index, and by increasing column index within each row. Multiple entries for the same row and column indices are not permitted. The function f11zn may be used to order the elements in this way.

3: **irow(la) – int32 array**
4: **icol(la) – int32 array**

The row and column indices of the nonzero elements supplied in **a**.

Constraints:

$1 \leq \mathbf{irow}(i) \leq \mathbf{n}$ and $1 \leq \mathbf{icol}(i) \leq \mathbf{n}$, for $i = 1, 2, \ldots, \mathbf{nnz}$;
$\mathbf{irow}(i-1) < \mathbf{irow}(i)$ or $\mathbf{irow}(i-1) = \mathbf{irow}(i)$ and $\mathbf{icol}(i-1) < \mathbf{icol}(i)$, for $i = 2, 3, \ldots, \mathbf{nnz}$.

irow and **icol** must satisfy these constraints (which may be imposed by a call to f11zn).

5: **lfill – int32 scalar**

If $\mathbf{lfill} \geq 0$ its value is the maximum level of fill allowed in the decomposition (see Section 8.2). A negative value of **lfill** indicates that **dtol** will be used to control the fill instead.

6: **dtol – double scalar**

If $\mathbf{lfill} < 0$ then **dtol** is used as a drop tolerance to control the fill-in (see Section 8.2). Otherwise **dtol** is not referenced.

Constraint: if $\mathbf{lfill} < 0$, $\mathbf{dtol} \geq 0.0$.

7: **milu – string**

Indicates whether or not the factorization should be modified to preserve row-sums (see Section 8.4).

milu = 'M'

The factorization is modified (**milu**).

milu = 'N'

The factorization is not modified.

Constraint: **milu** = 'M' or 'N'.

8: **ipivp(n) – int32 array**
9: **ipivq(n) – int32 array**

If **pstrat** = 'U', then $\mathbf{ipivp}(k)$ and $\mathbf{ipivq}(k)$ must specify the row and column indices of the element used as a pivot at elimination stage k. Otherwise **ipivp** and **ipivq** need not be initialized.

Constraint: if **pstrat** = 'U', **ipivp** and **ipivq** must both hold valid permutations of the integers on [1,**n**].

5.2 Optional Input Parameters

1: **n – int32 scalar**

Default: The dimension of the arrays **ipivp**, **ipivq**, **idiag**. (An error is raised if these dimensions are not equal.)

n, the order of the matrix A.

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

2: **la – int32 scalar**

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

These arrays must be of sufficient size to store both A (**nnz** elements) and C (**nnzc** elements).

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

3: **pstrat – string**

Specifies the pivoting strategy to be adopted.

pstrat = 'N'

No pivoting is carried out.

pstrat = 'U'

Pivoting is carried out according to the user-defined input values of **ipivp** and **ipivq**.

pstrat = 'P'

Partial pivoting by columns for stability is carried out.

pstrat = 'C'

Complete pivoting by rows for sparsity, and by columns for stability, is carried out.

Suggested value: **pstrat** = 'C'.

Default: 'C'

Constraint: **pstrat** = 'N', 'U', 'P' or 'C'.

5.3 Input Parameters Omitted from the MATLAB Interface

iwork, liwork

5.4 Output Parameters

1: **a(la) – complex array**

The first **nnz** entries of **a** contain the nonzero elements of A and the next **nnzc** entries contain the elements of the matrix C. Matrix elements are ordered by increasing row index, and by increasing column index within each row.

2: **irow(la) – int32 array**
3: **icol(la) – int32 array**

The row and column indices of the nonzero elements returned in **a**.

4: **ipivp(n) – int32 array**
5: **ipivq(n) – int32 array**

The pivot indices. If $\mathbf{ipivp}(k) = i$ and $\mathbf{ipivq}(k) = j$ then the element in row i and column j was used as the pivot at elimination stage k.

6: **istr**$(\mathbf{n}+\mathbf{1})$ – **int32 array**

istr(i), for $i = 1, 2, \ldots, \mathbf{n}$ holds the index of arrays **a**, **irow** and **icol** where row i of the matrix C starts. **istr**$(\mathbf{n}+1)$ holds the address of the last nonzero element in C plus one.

7: **idiag**$(\mathbf{n})$ – **int32 array**

idiag(i), for $i = 1, 2, \ldots, \mathbf{n}$ holds the index of arrays **a**, **irow** and **icol** which holds the diagonal element in row i of the matrix C.

8: **nnzc** – **int32 scalar**

The number of nonzero elements in the matrix C.

9: **npivm** – **int32 scalar**

If **npivm** > 0 it gives the number of pivots which were modified during the factorization to ensure that M exists.

If **npivm** $= -1$ no pivot modifications were required, but a local restart occurred (see Section 8.3). The quality of the preconditioner will generally depend on the returned value of **npivm**.

If **npivm** is large the preconditioner may not be satisfactory. In this case it may be advantageous to call f11dn again with an increased value of **lfill**, a reduced value of **dtol**, or set **pstrat** = 'C'. See also Section 8.5.

10: **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, **n** < 1,
or **nnz** < 1,
or **nnz** $> \mathbf{n}^2$,
or **la** $< 2 \times$ **nnz**,
or **lfill** < 0 and **dtol** < 0.0,
or **pstrat** $\neq$ 'N', 'U', 'P' or 'C',
or **milu** $\neq$ 'M' or 'N',
or **liwork** $< 7 \times \mathbf{n} + 2$.

ifail $= 2$

On entry, the arrays **irow** and **icol** fail to satisfy the following constraints:

$1 \le \mathbf{irow}(i) \le \mathbf{n}$ and $1 \le \mathbf{icol}(i) \le \mathbf{n}$, for $i = 1, 2, \ldots, \mathbf{nnz}$;

$\mathbf{irow}(i-1) < \mathbf{irow}(i)$, or $\mathbf{irow}(i-1) = \mathbf{irow}(i)$ and $\mathbf{icol}(i-1) < \mathbf{icol}(i)$, for $i = 2, 3, \ldots, \mathbf{nnz}$.

Therefore a nonzero element has been supplied which does not lie within the matrix A, is out of order, or has duplicate row and column indices. Call f11zn to reorder and sum or remove duplicates.

ifail $= 3$

On entry, **pstrat** = 'U', but one or both of **ipivp** and **ipivq** does not represent a valid permutation of the integers in [1,**n**]. An input value of **ipivp** or **ipivq** is either out of range or repeated.

ifail = 4

> **la** is too small, resulting in insufficient storage space for fill-in elements. The decomposition has been terminated before completion. Either increase **la** or reduce the amount of fill by reducing **lfill**, or increasing **dtol**.

ifail = 5 (f11zn)

> A serious error has occurred in an internal call to the specified function. Check all (sub)program calls and array sizes. Seek expert help.

7 Accuracy

The accuracy of the factorization will be determined by the size of the elements that are dropped and the size of any modifications made to the pivot elements. If these sizes are small then the computed factors will correspond to a matrix close to A. The factorization can generally be made more accurate by increasing **lfill**, or by reducing **dtol** with **lfill** < 0.

If f11dn is used in combination with f11bs or f11dq, the more accurate the factorization the fewer iterations will be required. However, the cost of the decomposition will also generally increase.

8 Further Comments

8.1 Timing

The time taken for a call to f11dn is roughly proportional to $(\mathbf{nnzc})^2/\mathbf{n}$.

8.2 Control of Fill-in

If **lfill** ≥ 0 the amount of fill-in occurring in the incomplete factorization is controlled by limiting the maximum level of fill-in to **lfill**. The original nonzero elements of A are defined to be of level 0. The fill level of a new nonzero location occurring during the factorization is defined as:

$$k = \max(k_{\mathrm{e}}, k_{\mathrm{c}}) + 1,$$

where k_{e} is the level of fill of the element being eliminated, and k_{c} is the level of fill of the element causing the fill-in.

If **lfill** < 0 the fill-in is controlled by means of the **drop tolerance dtol**. A potential fill-in element a_{ij} occurring in row i and column j will not be included if:

$$|a_{ij}| < \mathbf{dtol} \times \alpha,$$

where α is the maximum modulus element in the matrix A.

For either method of control, any elements which are not included are discarded unless **milu** = 'M', in which case their contributions are subtracted from the pivot element in the relevant elimination row, to preserve the row-sums of the original matrix.

Should the factorization process break down a local restart process is implemented as described in Section 8.3. This will affect the amount of fill present in the final factorization.

8.3 Algorithmic Details

The factorization is constructed row by row. At each elimination stage a row index is chosen. In the case of complete pivoting this index is chosen in order to reduce fill-in. Otherwise the rows are treated in the order given, or some user-defined order.

The chosen row is copied from the original matrix A and modified according to those previous elimination stages which affect it. During this process any fill-in elements are either dropped or kept according to the values of **lfill** or **dtol**. In the case of a modified factorization (**milu** = 'M') the sum of the dropped terms for the given row is stored.

Finally the pivot element for the row is chosen and the multipliers are computed for this elimination stage. For partial or complete pivoting the pivot element is chosen in the interests of stability as the element of largest absolute value in the row. Otherwise the pivot element is chosen in the order given, or some user-defined order.

If the factorization breaks down because the chosen pivot element is zero, or there is no nonzero pivot available, a local restart recovery process is implemented. The modification of the given pivot row according to previous elimination stages is repeated, but this time keeping all fill. Note that in this case the final factorization will include more fill than originally specified by the user-supplied value of **lfill** or **dtol**. The local restart usually results in a suitable nonzero pivot arising. The original criteria for dropping fill-in elements is then resumed for the next elimination stage (hence the **local** nature of the restart process). Should this restart process also fail to produce a nonzero pivot element an arbitrary unit pivot is introduced in an arbitrarily chosen column. f11dn returns an integer parameter **npivm** which gives the number of these arbitrary unit pivots introduced. If no pivots were modified but local restarts occurred **npivm** $= -1$ is returned.

8.4 Choice of Parameters

There is unfortunately no choice of the various algorithmic parameters which is optimal for all types of matrix, and some experimentation will generally be required for each new type of matrix encountered. The recommended approach is to start with **lfill** $= 0$ and **pstrat** = 'C'. If the value returned for **npivm** is significantly larger than zero, i.e., a large number of pivot modifications were required to ensure that M existed, the preconditioner is not likely to be satisfactory. In this case increase **lfill** until **npivm** falls to a value close to zero.

For certain classes of matrices (typically those arising from the discretization of elliptic or parabolic partial differential equations) the convergence rate of the preconditioned iterative solver can sometimes be significantly improved by using an incomplete factorization which preserves the row-sums of the original matrix. In these cases try setting **milu** = 'M'.

8.5 Direct Solution of Sparse Linear Systems

Although it is not their primary purpose f11dn and f11dp may be used together to obtain a **direct** solution to a nonsingular sparse complex non-Hermitian linear system. To achieve this the call to f11dp should be preceded by a **complete** LU factorization

$$A = PLDUQ = M.$$

a complete factorization is obtained from a call to f11dn with **lfill** < 0 and **dtol** $= 0.0$, provided **npivm** ≤ 0 on exit. A positive value of **npivm** indicates that A is singular, or ill-conditioned. A factorization with positive **npivm** may serve as a preconditioner, but will not result in a direct solution. It is therefore **essential** to check the output value of **npivm** if a direct solution is required.

The use of f11dn and f11dp as a direct method is illustrated in f11dp.

9 Example

```
nnz = int32(11);
a = [complex(1, +3);
     complex(1, +0);
     complex(-1, -2);
     complex(2, -2);
     complex(2, +1);
     complex(0, +5);
     complex(-2, +0);
     complex(1, +1);
     complex(-2, +4);
     complex(1, -3);
     complex(0, +7);
     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(1);
     int32(2);
     int32(2);
     int32(2);
     int32(3);
     int32(3);
     int32(4);
     int32(4);
     int32(4);
     int32(4);
     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(2);
     int32(3);
     int32(1);
     int32(3);
     int32(4);
     int32(1);
     int32(4);
     int32(1);
     int32(2);
     int32(3);
     int32(4);
     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)];
lfill = int32(0);
dtol = 0;
milu = 'N';
ipivp = [int32(0);
     int32(0);
     int32(0);
     int32(0)];
ipivq = [int32(0);
     int32(0);
     int32(0);
     int32(0)];
[aOut, irowOut, icolOut, ipivpOut, ipivqOut, istr, idiag, nnzc, npivm,
ifail] = ...
    f11dn(nnz, a, irow, icol, lfill, dtol, milu, ipivp, ipivq)
```

```
aOut =
   1.0000 + 3.0000i
   1.0000
  -1.0000 - 2.0000i
   2.0000 - 2.0000i
   2.0000 + 1.0000i
        0 + 5.0000i
  -2.0000
   1.0000 + 1.0000i
  -2.0000 + 4.0000i
   1.0000 - 3.0000i
        0 + 7.0000i
   0.1000 - 0.3000i
   0.1000 - 0.3000i
        0 - 0.2000i
        0 + 0.4000i
  -0.4000 + 0.2000i
   0.2500 + 0.2500i
  -0.0500 + 0.6500i
   1.0000 + 1.0000i
   0.2000 - 0.2000i
   1.0000 - 1.0000i
  -0.0480 - 0.1397i
        0
        0
        0
        0
        0
        0
        0
        0
irowOut =
           1
           1
           2
           2
           2
           3
           3
           4
           4
           4
           4
           1
           1
           2
           2
           3
           3
           3
           4
           4
           4
           4
           0
           0
           0
           0
           0
           0
           0
           0
icolOut =
           2
           3
           1
           3
           4
           1
           4
           1
           2
           3
           4
           1
           3
           2
           4
           2
           3
           4
           1
           2
           3
           4
           0
           0
           0
           0
           0
           0
           0
           0
ipivpOut =
           1
           3
           2
           4
ipivqOut =
           2
           1
           3
           4
istr =
          12
          14
          16
          19
          23
idiag =
          12
          14
          17
          22
nnzc =
          11
npivm =
           0
ifail =
                0
```