NAG Library Routine Document

F11ZAF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

F11ZAF sorts the nonzero elements of a real sparse nonsymmetric matrix, represented in coordinate storage format.

2 Specification

```
SUBROUTINE F11ZAF (N, NNZ, A, IROW, ICOL, DUP, ZER, ISTR, IWORK, IFAIL)
INTEGER            N, NNZ, IROW(*), ICOL(*), ISTR(N+1), IWORK(N), IFAIL
REAL (KIND=nag_wp) A(*)
CHARACTER(1)       DUP, ZER
```

3 Description

F11ZAF takes a coordinate storage (CS) representation (see Section 2.1.1 in the F11 Chapter Introduction) of a real n by n sparse nonsymmetric matrix A, and reorders the nonzero elements by increasing row index and increasing column index within each row. Entries with duplicate row and column indices may be removed, or the values may be summed. Any entries with zero values may optionally be removed.

F11ZAF also returns a pointer ISTR to the starting address of each row in A. This can be used to construct a compressed column storage (CCS) representation of the matrix (see Section 9).

4 References

None.

5 Arguments

1: N – INTEGER *Input*

On entry: n, the order of the matrix A.

Constraint: N ≥ 1.

2: NNZ – INTEGER *Input/Output*

On entry: the number of nonzero elements in the matrix A.

Constraint: NNZ ≥ 0.

On exit: the number of nonzero elements with unique row and column indices.

3: A$(*)$ – REAL (KIND=nag_wp) array *Input/Output*

Note: the dimension of the array A must be at least $\max(1, \text{NNZ})$.

On entry: the nonzero elements of the matrix A. These may be in any order and there may be multiple nonzero elements with the same row and column indices.

On exit: the nonzero elements ordered by increasing row index, and by increasing column index within each row. Each nonzero element has a unique row and column index.

4: IROW($*$) – INTEGER array *Input/Output*

Note: the dimension of the array IROW must be at least $\max(1, \text{NNZ})$.

On entry: the row indices corresponding to the nonzero elements supplied in the array A.

Constraint: $1 \leq \text{IROW}(i) \leq \text{N}$, for $i = 1, 2, \ldots, \text{NNZ}$.

On exit: the first NNZ elements contain the row indices corresponding to the nonzero elements returned in the array A.

5: ICOL($*$) – INTEGER array *Input/Output*

Note: the dimension of the array ICOL must be at least $\max(1, \text{NNZ})$.

On entry: the column indices corresponding to the nonzero elements supplied in the array A.

Constraint: $1 \leq \text{ICOL}(i) \leq \text{N}$, for $i = 1, 2, \ldots, \text{NNZ}$.

On exit: the first NNZ elements contain the row indices corresponding to the nonzero elements returned in the array A.

6: DUP – CHARACTER(1) *Input*

On entry: indicates how any nonzero elements with duplicate row and column indices are to be treated.

DUP = 'R'
The entries are removed.

DUP = 'S'
The relevant values in A are summed.

DUP = 'F'
The routine fails on detecting a duplicate, with IFAIL $= 3$.

Constraint: DUP = 'R', 'S' or 'F'.

7: ZER – CHARACTER(1) *Input*

On entry: indicates how any elements with zero values in A are to be treated.

ZER = 'R'
The entries are removed.

ZER = 'K'
The entries are kept.

ZER = 'F'
The routine fails on detecting a zero, with IFAIL $= 4$.

Constraint: ZER = 'R', 'K' or 'F'.

8: ISTR(N + 1) – INTEGER array *Output*

On exit: ISTR(i), for $i = 1, 2, \ldots, \text{N}$, is the starting address in the arrays A, IROW and ICOL of row i of the matrix A. ISTR(N + 1) is the address of the last nonzero element in A plus one.

9: IWORK(N) – INTEGER array *Workspace*

10: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, if you are not familiar with this argument, the

recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: IFAIL $= 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL $= 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

IFAIL $= 1$

On entry, N < 1,
or NNZ < 0,
or DUP $\neq$ 'R','S', or `F',
or ZER $\neq$ 'R','K', or `F'.

IFAIL $= 2$

On entry, a nonzero element has been supplied which does not lie within the matrix A, i.e., one or more of the following constraints has been violated:

$1 \leq \text{IROW}(i) \leq \text{N}$,

$1 \leq \text{ICOL}(i) \leq \text{N}$,

for $i = 1, 2, \ldots, \text{NNZ}$.

IFAIL $= 3$

On entry, DUP $=$ 'F' and nonzero elements have been supplied which have duplicate row and column indices.

IFAIL $= 4$

On entry, ZER $=$ 'F' and at least one matrix element has been supplied with a zero coefficient value.

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

Not applicable.

8 Parallelism and Performance

F11ZAF is not threaded in any implementation.

9 Further Comments

The time taken for a call to F11ZAF is proportional to NNZ.

Note that the resulting matrix may have either rows or columns with no entries. If row i has no entries then $\mathrm{ISTR}(i) = \mathrm{ISTR}(i+1)$.

It is also possible to use this routine to convert between coordinate storage (CS) and compressed column storage (CCS) formats. To achieve this the CS storage format arrays IROW and ICOL must be interchanged in the call to F11ZAF. On exit from F11ZAF, the CCS representation of the matrix is then defined by arrays A, IROW and ISTR. This is illustrated in Section 10.

10 Example

This example reads the CS representation of a real sparse matrix A, calls F11ZAF to reorder the nonzero elements, and outputs the original and the reordered representations. It then calls F11ZAF again with the alternative ordering, creating a CCS representation which is then passed to a routine that computes a matrix norm for that representation.

$$A = \begin{pmatrix} 2.00 & 1.00 & 0 & 0 & 0 \\ 0 & 0 & 1.00 & -1.00 & 0 \\ 4.00 & 0 & 1.00 & 0 & 1.00 \\ 0 & 0 & 0 & 1.00 & 2.00 \\ 0 & -2.00 & 0 & 0 & 3.00 \end{pmatrix}.$$

10.1 Program Text

```
    Program f11zafe

!     F11ZAF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: f11mlf, f11zaf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)               :: anorm
      Integer                          :: i, ifail, n, nnz
      Character (1)                    :: dup, norm, zer
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: a(:)
      Integer, Allocatable             :: icol(:), irow(:), istr(:), iwork(:)
!     .. Executable Statements ..
      Write (nout,*) 'F11ZAF Example Program Results'
      Write (nout,*)
!     Skip heading in data file
      Read (nin,*)

!     Read order of matrix and number of nonzero entries

      Read (nin,*) n
      Read (nin,*) nnz

      Allocate (a(nnz),icol(nnz),irow(nnz),istr(n+1),iwork(n))

!     Read and output the original nonzero elements

      Do i = 1, nnz
```

```
        Read (nin,*) a(i), irow(i), icol(i)
      End Do
      Write (nout,*) 'Original elements'
      Write (nout,99999) nnz
      Write (nout,99997)
      Do i = 1, nnz
        Write (nout,99998) i, a(i), irow(i), icol(i)
      End Do

!     Reorder, sum duplicates and remove zeros

      dup = 'S'
      zer = 'R'

!     ifail: behaviour on error exit
!             =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
      ifail = 0
      Call f11zaf(n,nnz,a,irow,icol,dup,zer,istr,iwork,ifail)

!     Output results

      Write (nout,*) 'Reordered elements, along rows first'
      Write (nout,99999) nnz
      Do i = 1, nnz
        Write (nout,99998) i, a(i), irow(i), icol(i)
      End Do

!     Reorder down columns, fail on duplicates or zeros.
!     Creates CCS storage format as side-effect

      dup = 'F'
      zer = 'F'
      ifail = 0

      Call f11zaf(n,nnz,a,icol,irow,dup,zer,istr,iwork,ifail)

!     Output results

      Write (nout,*) 'Reordered elements, along columns first'
      Write (nout,99999) nnz
      Do i = 1, nnz
        Write (nout,99998) i, a(i), irow(i), icol(i)
      End Do
      Write (nout,99996)
      Do i = 1, n + 1
        Write (nout,99995) i, istr(i)
      End Do

!     Calculate 1-norm in Compressed Column Storage format

      norm = '1'
      Call f11mlf(norm,anorm,n,istr,irow,a,ifail)

!     Output norm

      Write (nout,99994) anorm

99999 Format (1X,'NNZ = ',I4)
99998 Format (1X,I8,1P,E16.4,2I8)
99997 Format (24X,'A',4X,'IROW',4X,'ICOL')
99996 Format (13X,'ISTR')
99995 Format (1X,2I8)
99994 Format (1X,'One norm ',1P,E16.4)
    End Program f11zafe
```

10.2 Program Data

```
F11ZAF Example Program Data
  5                  N
  15                 NNZ
  4.   3    1
 -2.   5    2
  1.   4    4
 -2    4    2
 -3    5    5
  1.   1    2
  0.   1    5
  1.   3    5
 -1.   2    4
  6.   5    5
  2.   1    1
  2.   4    2
  1.   2    3
  1.   3    3
  2.   4    5         A(I), IROW(I), ICOL(I), I=1,...,NNZ
```

10.3 Program Results

```
 F11ZAF Example Program Results

 Original elements
 NNZ =   15
                         A    IROW    ICOL
        1      4.0000E+00       3       1
        2     -2.0000E+00       5       2
        3      1.0000E+00       4       4
        4     -2.0000E+00       4       2
        5     -3.0000E+00       5       5
        6      1.0000E+00       1       2
        7      0.0000E+00       1       5
        8      1.0000E+00       3       5
        9     -1.0000E+00       2       4
       10      6.0000E+00       5       5
       11      2.0000E+00       1       1
       12      2.0000E+00       4       2
       13      1.0000E+00       2       3
       14      1.0000E+00       3       3
       15      2.0000E+00       4       5
 Reordered elements, along rows first
 NNZ =   11
        1      2.0000E+00       1       1
        2      1.0000E+00       1       2
        3      1.0000E+00       2       3
        4     -1.0000E+00       2       4
        5      4.0000E+00       3       1
        6      1.0000E+00       3       3
        7      1.0000E+00       3       5
        8      1.0000E+00       4       4
        9      2.0000E+00       4       5
       10     -2.0000E+00       5       2
       11      3.0000E+00       5       5
 Reordered elements, along columns first
 NNZ =   11
        1      2.0000E+00       1       1
        2      4.0000E+00       3       1
        3      1.0000E+00       1       2
        4     -2.0000E+00       5       2
        5      1.0000E+00       2       3
        6      1.0000E+00       3       3
        7     -1.0000E+00       2       4
        8      1.0000E+00       4       4
        9      1.0000E+00       3       5
       10      2.0000E+00       4       5
       11      3.0000E+00       5       5
             ISTR
```

```
        1       1
        2       3
        3       5
        4       7
        5       9
        6      12
 One norm       6.0000E+00
```