NAG Library Routine Document

F11ZPF

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

F11ZPF sorts the nonzero elements of a sparse complex Hermitian matrix, represented in symmetric coordinate storage format.

2 Specification

```
SUBROUTINE F11ZPF (N, NNZ, A, IROW, ICOL, DUP, ZER, ISTR, IWORK, IFAIL)

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

3 Description

F11ZPF takes a symmetric coordinate storage (SCS) representation (see Section 2.1.2 in the F11 Chapter Introduction) of a sparse n by n complex Hermitian 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.

The routine also returns a pointer array ISTR to the starting address of each row in A.

4 References

None.

5 Parameters

1: N – INTEGER *Input*

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

Constraint: $\text{N} \geq 1$.

2: NNZ – INTEGER *Input/Output*

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

Constraint: $\text{NNZ} \geq 0$.

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

3: A($*$) – COMPLEX (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 lower triangular part of the complex 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 lower triangular 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 $\max(1, \mathrm{NNZ})$.

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

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

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 $\max(1, \mathrm{NNZ})$.

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

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

On exit: the first NNZ elements contain the column 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 with IFAIL $= 3$ on detecting a duplicate.

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

7: ZER – CHARACTER(1) *Input*

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

ZER = 'R'
: The entries are removed.

ZER = 'K'
: The entries are kept.

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

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

8: ISTR($\mathrm{N}+1$) – INTEGER array *Output*

On exit: ISTR(i), for $i = 1, 2, \ldots, \mathrm{N}$, is the starting address in the arrays A, IROW and ICOL of row i of the matrix A. ISTR($\mathrm{N}+1$) 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 parameter you should refer to Section 3.3 in the Essential Introduction 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 parameter, 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 in the lower triangular part of A, i.e., one or more of the following constraints have been violated:

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

$$1 \leq \mathrm{ICOL}(i) \leq \mathrm{IROW}(i),$$

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

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.

7 Accuracy

Not applicable.

8 Further Comments

The time taken for a call to F11ZPF 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 ISTR$(i) =$ ISTR$(i+1)$.

9 Example

This example reads the SCS representation of a complex sparse Hermitian matrix A, calls F11ZPF to reorder the nonzero elements, and outputs the original and the reordered representations.

9.1 Program Text

```
    Program f11zpfe

!     F11ZPF Example Program Text

!     Mark 24 Release. NAG Copyright 2012.

!     .. Use Statements ..
      Use nag_library, Only: f11zpf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: i, ifail, n, nnz
      Character (1)                    :: dup, zer
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: a(:)
      Integer, Allocatable             :: icol(:), irow(:), istr(:), iwork(:)
!     .. Executable Statements ..
      Write (nout,*) 'F11ZPF Example Program Results'
!     Skip heading in data file
      Read (nin,*)

!     Read order of matrix and number of non-zero entries

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

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

!     Read and output the original non-zero elements

      Do i = 1, nnz
        Read (nin,*) a(i), irow(i), icol(i)
      End Do
      Write (nout,*) 'Original elements'
      Write (nout,99997) 'NNZ = ', nnz
      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 f11zpf(n,nnz,a,irow,icol,dup,zer,istr,iwork,ifail)

!     Output results

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

99999 Format (1X,A,I4)
99998 Format (I8,5X,'(',E16.4,',',E16.4,')',2I8)
99997 Format (1X,A,I16)
    End Program f11zpfe
```

9.2 Program Data

```
F11ZPF Example Program Data
  4                  N
  9                  NNZ
  (1., 2.)   3    2
  (0., 0.)   2    1
  (0., 3.)   3    2
  (3.,-5.)   4    4
  (4., 2.)   1    1
  (0., 3.)   2    2
  (2., 4.)   3    3
  (1.,-1.)   3    2
  (1., 3.)   3    2   A(I), IROW(I), ICOL(I), I=1,...,NNZ
```

9.3 Program Results

```
 F11ZPF Example Program Results
 Original elements
 NNZ =               9
       1     (      0.1000E+01,      0.2000E+01)       3       2
       2     (      0.0000E+00,      0.0000E+00)       2       1
       3     (      0.0000E+00,      0.3000E+01)       3       2
       4     (      0.3000E+01,     -0.5000E+01)       4       4
       5     (      0.4000E+01,      0.2000E+01)       1       1
       6     (      0.0000E+00,      0.3000E+01)       2       2
       7     (      0.2000E+01,      0.4000E+01)       3       3
       8     (      0.1000E+01,     -0.1000E+01)       3       2
       9     (      0.1000E+01,      0.3000E+01)       3       2
 Reordered elements
 NNZ =    5
       1     (      0.4000E+01,      0.2000E+01)       1       1
       2     (      0.0000E+00,      0.3000E+01)       2       2
       3     (      0.3000E+01,      0.7000E+01)       3       2
       4     (      0.2000E+01,      0.4000E+01)       3       3
       5     (      0.3000E+01,     -0.5000E+01)       4       4
```