NAG Library Routine Document

F11XSF

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

F11XSF computes a matrix-vector product involving a complex sparse Hermitian matrix stored in symmetric coordinate storage format.

2 Specification

```
SUBROUTINE F11XSF (N, NNZ, A, IROW, ICOL, CHECK, X, Y, IFAIL)
INTEGER                 N, NNZ, IROW(NNZ), ICOL(NNZ), IFAIL
COMPLEX (KIND=nag_wp) A(NNZ), X(N), Y(N)
CHARACTER(1)            CHECK
```

3 Description

F11XSF computes the matrix-vector product

$$y = Ax$$

where A is an n by n complex Hermitian sparse matrix, of arbitrary sparsity pattern, stored in symmetric coordinate storage (SCS) format (see Section 2.1.2 in the F11 Chapter Introduction). The array A stores all the nonzero elements in the lower triangular part of A, while arrays IROW and ICOL store the corresponding row and column indices respectively.

4 References

None.

5 Parameters

1: N – INTEGER *Input*

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

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

2: NNZ – INTEGER *Input*

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

Constraint: $1 \leq \mathrm{NNZ} \leq \mathrm{N} \times (\mathrm{N}+1)/2$.

3: A(NNZ) – COMPLEX (KIND=nag_wp) array *Input*

On entry: the nonzero elements in the lower triangular part of 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 routine F11ZPF may be used to order the elements in this way.

4: IROW(NNZ) – INTEGER array *Input*

5: ICOL(NNZ) – INTEGER array *Input*

On entry: the row and column indices of the nonzero elements supplied in array A.

Constraints:

IROW and ICOL must satisfy the following constraints (which may be imposed by a call to F11ZPF):

$1 \leq \text{IROW}(i) \leq \text{N}$ and $1 \leq \text{ICOL}(i) \leq \text{IROW}(i)$, for $i = 1, 2, \ldots, \text{NNZ}$;
$\text{IROW}(i-1) < \text{IROW}(i)$ or $\text{IROW}(i-1) = \text{IROW}(i)$ and $\text{ICOL}(i-1) < \text{ICOL}(i)$, for $i = 2, 3, \ldots, \text{NNZ}$.

6: CHECK – CHARACTER(1) *Input*

On entry: specifies whether or not the SCS representation of the matrix A, values of N, NNZ, IROW and ICOL should be checked.

CHECK = 'C'
Checks are carried out on the values of N, NNZ, IROW and ICOL.

CHECK = 'N'
None of these checks are carried out.

Constraint: CHECK = 'C' or 'N'.

7: X(N) – COMPLEX (KIND=nag_wp) array *Input*

On entry: the vector x.

8: Y(N) – COMPLEX (KIND=nag_wp) array *Output*

On exit: the vector y.

9: 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, CHECK $\neq$ 'C' or 'N'.

IFAIL $= 2$

On entry, N < 1,
or NNZ < 1,
or NNZ $>$ N $\times$ (N $+ 1)/2$.

IFAIL $= 3$

On entry, the arrays IROW and ICOL fail to satisfy the following constraints:

$1 \leq \text{IROW}(i) \leq \text{N}$ and $1 \leq \text{ICOL}(i) \leq \text{IROW}(i)$, for $i = 1, 2, \ldots, \text{NNZ}$;

$\text{IROW}(i-1) < \text{IROW}(i)$ or $\text{IROW}(i-1) = \text{IROW}(i)$ and $\text{ICOL}(i-1) < \text{ICOL}(i)$, for $i = 2, 3, \ldots, \text{NNZ}$.

Therefore a nonzero element has been supplied which does not lie in the lower triangular part of A, is out of order, or has duplicate row and column indices. Call F11ZPF to reorder and sum or remove duplicates.

IFAIL $= -99$

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

See Section 3.8 in the Essential Introduction for further information.

IFAIL $= -399$

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

See Section 3.7 in the Essential Introduction for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.6 in the Essential Introduction for further information.

7 Accuracy

The computed vector y satisfies the error bound

$$\|y - Ax\|_\infty \leq c(n)\epsilon\|A\|_\infty\|x\|_\infty,$$

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

8 Parallelism and Performance

F11XSF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

F11XSF makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

9.1 Timing

The time taken for a call to F11XSF is proportional to NNZ.

10 Example

This example reads in a complex sparse Hermitian positive definite matrix A and a vector x. It then calls F11XSF to compute the matrix-vector product $y = Ax$.

10.1 Program Text

```
    Program f11xsfe

!     F11XSF Example Program Text

!     Mark 25 Release. NAG Copyright 2014.

!     .. Use Statements ..
      Use nag_library, Only: f11xsf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter              :: nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                         :: i, ifail, n, nnz
      Character (1)                   :: check
!     .. Local Arrays ..
      Complex (Kind=nag_wp), Allocatable :: a(:), x(:), y(:)
      Integer, Allocatable            :: icol(:), irow(:)
!     .. Executable Statements ..
      Write (nout,*) 'F11XSF 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),x(n),y(n),icol(nnz),irow(nnz))

!     Read the matrix A

      Do i = 1, nnz
        Read (nin,*) a(i), irow(i), icol(i)
      End Do

!     Read the vector x

      Read (nin,*) x(1:n)

!     Calculate matrix-vector product

      check = 'C'

!     ifail: behaviour on error exit
!             =0 for hard exit, =1 for quiet-soft, =-1 for noisy-soft
      ifail = 0
      Call f11xsf(n,nnz,a,irow,icol,check,x,y,ifail)

!     Output results

      Write (nout,*) ' Matrix-vector product'
      Write (nout,99999) y(1:n)

99999 Format (1X,'(',E16.4,',',E16.4,')')
    End Program f11xsfe
```

10.2 Program Data

```
F11XSF Example Program Data
  9                   N
 23                   NNZ
 ( 6., 0.)   1    1
 (-1., 1.)   2    1
 ( 6., 0.)   2    2
 ( 0., 1.)   3    2
 ( 5., 0.)   3    3
 ( 5., 0.)   4    4
```

```
( 2.,-2.)   5    1
( 4., 0.)   5    5
( 1., 1.)   6    3
( 2., 0.)   6    4
( 6., 0.)   6    6
(-4., 3.)   7    2
( 0., 1.)   7    5
(-1., 0.)   7    6
( 6., 0.)   7    7
(-1.,-1.)   8    4
( 0.,-1.)   8    6
( 9., 0.)   8    8
( 1., 3.)   9    1
( 1., 2.)   9    5
(-1., 0.)   9    6
( 1., 4.)   9    8
( 9., 0.)   9    9   A(I), IROW(I), ICOL(I), I=1,...,NNZ
(1., 9.) (2.,-8.) (3., 7.)
(4.,-6.) (5., 5.) (6.,-4.)
(7., 3.) (8.,-2.) (9., 1.)  X(I), I=1,...,N
```

10.3 Program Results

```
F11XSF Example Program Results
 Matrix-vector product
(      0.8000E+01,      0.5400E+02)
(     -0.1000E+02,     -0.9200E+02)
(      0.2500E+02,      0.2700E+02)
(      0.2600E+02,     -0.2800E+02)
(      0.5400E+02,      0.1200E+02)
(      0.2600E+02,     -0.2200E+02)
(      0.4700E+02,      0.6500E+02)
(      0.7100E+02,     -0.5700E+02)
(      0.6000E+02,      0.7000E+02)
```