NAG Library Routine Document

F11XEF

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

F11XEF computes a matrix-vector product involving a real sparse symmetric matrix stored in symmetric coordinate storage format.

2 Specification

```
SUBROUTINE F11XEF (N, NNZ, A, IROW, ICOL, CHECK, X, Y, IFAIL)

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

3 Description

F11XEF computes the matrix-vector product

$$y = Ax$$

where A is an n by n symmetric 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 nonzero elements in the lower triangular part of A, while arrays IROW and ICOL store the corresponding row and column indices respectively.

It is envisaged that a common use of F11XEF will be to compute the matrix-vector product required in the application of F11GEF to sparse symmetric linear systems. An illustration of this usage appears in F11JDF.

4 References

None.

5 Parameters

1: N – INTEGER *Input*

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

Constraint: $N \geq 1$.

2: NNZ – INTEGER *Input*

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

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

3: A(NNZ) – REAL (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 F11ZBF 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 these constraints (which may be imposed by a call to F11ZBF):

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

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.

See also Section 8.2.

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

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

On entry: the vector x.

8: Y(N) – REAL (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, $\mathrm{N} < 1$,
or $\mathrm{NNZ} < 1$,
or $\mathrm{NNZ} > \mathrm{N} \times (\mathrm{N}+1)/2$.

IFAIL = 3

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

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

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

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 F11ZBF to reorder and sum or remove duplicates.

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 Further Comments

8.1 Timing

The time taken for a call to F11XEF is proportional to NNZ.

8.2 Use of CHECK

It is expected that a common use of F11XEF will be to compute the matrix-vector product required in the application of F11GEF to sparse symmetric linear systems. In this situation F11XEF is likely to be called many times with the same matrix A. In the interests of both reliability and efficiency you are recommended to set CHECK = 'C' for the first of such calls, and to set CHECK = 'N' for all subsequent calls.

9 Example

This example reads in a symmetric positive definite sparse matrix A and a vector x. It then calls F11XEF to compute the matrix-vector product $y = Ax$.

9.1 Program Text

```
    Program f11xefe

!     F11XEF Example Program Text

!     Mark 24 Release. NAG Copyright 2012.

!     .. Use Statements ..
      Use nag_library, Only: f11xef, 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 ..
      Real (Kind=nag_wp), Allocatable  :: a(:), x(:), y(:)
      Integer, Allocatable             :: icol(:), irow(:)
!     .. Executable Statements ..
      Write (nout,*) 'F11XEF 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 f11xef(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)
    End Program f11xefe
```

9.2 Program Data

```
F11XEF Example Program Data
  9                   N
 23                   NNZ
  4.   1    1
 -1.   2    1
  6.   2    2
  1.   3    2
  2.   3    3
  3.   4    4
  2.   5    1
  4.   5    5
  1.   6    3
  2.   6    4
  6.   6    6
 -4.   7    2
  1.   7    5
 -1.   7    6
  6.   7    7
 -1.   8    4
 -1.   8    6
  3.   8    8
  1.   9    1
  1.   9    5
 -1.   9    6
  1.   9    8
  4.   9    9         A(I), IROW(I), ICOL(I), I=1,...,NNZ
 0.70 0.16 0.52
 0.77 0.28 0.21
 0.93 0.20 0.90       X(I), I=1,...,N
```

9.3 Program Results

```
 F11XEF Example Program Results
  Matrix-vector product
       0.4100E+01
      -0.2940E+01
       0.1410E+01
       0.2530E+01
       0.4350E+01
       0.1290E+01
       0.5010E+01
       0.5200E+00
       0.4570E+01
```