NAG Toolbox for MATLAB

f11xs

1 Purpose

f11xs computes a matrix-vector product involving a complex sparse Hermitian matrix stored in symmetric co-ordinate storage format.

2 Syntax

```
[y, ifail] = f11xs(a, irow, icol, check, x, 'n', n, 'nnz', nnz)
```

3 Description

f11xs 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 co-ordinate 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

5.1 Compulsory Input Parameters

1: **a**(**nnz**) – **complex array**

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 function f11zp may be used to order the elements in this way.

2: **irow**(**nnz**) – **int32 array**

3: **icol**(**nnz**) – **int32 array**

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

Constraints:

$1 \leq \mathbf{irow}(i) \leq \mathbf{n}$ and $1 \leq \mathbf{icol}(i) \leq \mathbf{irow}(i)$, 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 the following constraints (which may be imposed by a call to f11zp):

4: **check** – **string**

Specifies whether or not the input data 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'.

5: **x(n) – complex array**

The vector *x*.

5.2 Optional Input Parameters

1: **n – int32 scalar**

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

n, the order of the matrix *A*.

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

2: **nnz – int32 scalar**

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

the number of nonzero elements in the lower triangular part of the matrix *A*.

Constraint: $1 \leq \mathbf{nnz} \leq \mathbf{n} \times (\mathbf{n}+1)/2$.

5.3 Input Parameters Omitted from the MATLAB Interface

None.

5.4 Output Parameters

1: **y(n) – complex array**

The vector *y*.

2: **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, **check** $\neq$ 'C' or 'N'.

ifail = 2

On entry, $\mathbf{n} < 1$,
or $\mathbf{nnz} < 1$,
or $\mathbf{nnz} > \mathbf{n} \times (\mathbf{n}+1)/2$.

ifail = 3

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

$1 \leq \mathbf{irow}(i) \leq \mathbf{n}$ and $1 \leq \mathbf{icol}(i) \leq \mathbf{irow}(i)$, 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 in the lower triangular part of A, is out of order, or has duplicate row and column indices. Call f11zp 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 f11xs is proportional to **nnz**.

9 Example

```
a = [complex(6, +0);
     complex(-1, +1);
     complex(6, +0);
     complex(0, +1);
     complex(5, +0);
     complex(5, +0);
     complex(2, -2);
     complex(4, +0);
     complex(1, +1);
     complex(2, +0);
     complex(6, +0);
     complex(-4, +3);
     complex(0, +1);
     complex(-1, +0);
     complex(6, +0);
     complex(-1, -1);
     complex(0, -1);
     complex(9, +0);
     complex(1, +3);
     complex(1, +2);
     complex(-1, +0);
     complex(1, +4);
     complex(9, +0)];
irow = [int32(1);
     int32(2);
     int32(2);
     int32(3);
     int32(3);
     int32(4);
     int32(5);
     int32(5);
     int32(6);
     int32(6);
     int32(6);
     int32(7);
     int32(7);
     int32(7);
     int32(7);
     int32(8);
     int32(8);
```

```
     int32(8);
     int32(9);
     int32(9);
     int32(9);
     int32(9);
     int32(9)];
icol = [int32(1);
     int32(1);
     int32(2);
     int32(2);
     int32(3);
     int32(4);
     int32(1);
     int32(5);
     int32(3);
     int32(4);
     int32(6);
     int32(2);
     int32(5);
     int32(6);
     int32(7);
     int32(4);
     int32(6);
     int32(8);
     int32(1);
     int32(5);
     int32(6);
     int32(8);
     int32(9)];
check = 'C';
x = [complex(1, +9);
     complex(2, -8);
     complex(3, +7);
     complex(4, -6);
     complex(5, +5);
     complex(6, -4);
     complex(7, +3);
     complex(8, -2);
     complex(9, +1)];
[y, ifail] = f11xs(a, irow, icol, check, x)
```

```
y =
   8.0000 +54.0000i
 -10.0000 -92.0000i
  25.0000 +27.0000i
  26.0000 -28.0000i
  54.0000 +12.0000i
  26.0000 -22.0000i
  47.0000 +65.0000i
  71.0000 -57.0000i
  60.0000 +70.0000i
ifail =
           0
```