NAG Toolbox

nag_sparse_complex_gen_matvec (f11xn)

1 Purpose

nag_sparse_complex_gen_matvec (f11xn) computes a matrix-vector or conjugate transposed matrix-vector product involving a complex sparse non-Hermitian matrix stored in coordinate storage format.

2 Syntax

```
[y, ifail] = nag_sparse_complex_gen_matvec(trans, a, irow, icol, check, x, 'n', n, 'nnz', nnz)
[y, ifail] = f11xn(trans, a, irow, icol, check, x, 'n', n, 'nnz', nnz)
```

3 Description

nag_sparse_complex_gen_matvec (f11xn) computes either the matrix-vector product $y = Ax$, or the conjugate transposed matrix-vector product $y = A^{\mathrm{H}}x$, according to the value of the argument **trans**, where A is a complex n by n sparse non-Hermitian matrix, of arbitrary sparsity pattern. The matrix A is stored in coordinate storage (CS) format (see Section 2.1.1 in the F11 Chapter Introduction). The array **a** stores all the nonzero elements of A, while arrays **irow** and **icol** store the corresponding row and column indices respectively.

It is envisaged that a common use of nag_sparse_complex_gen_matvec (f11xn) will be to compute the matrix-vector product required in the application of nag_sparse_complex_gen_basic_solver (f11bs) to sparse complex linear systems. This is illustrated in Section 10 in nag_sparse_complex_gen_precon_ssor_solve (f11dr).

4 References

None.

5 Parameters

5.1 Compulsory Input Parameters

1: **trans** – CHARACTER(1)

Specifies whether or not the matrix A is conjugate transposed.

trans = 'N'
$y = Ax$ is computed.

trans = 'T'
$y = A^{\mathrm{H}}x$ is computed.

Constraint: **trans** = 'N' or 'T'.

2: **a**(**nnz**) – COMPLEX (KIND=nag_wp) array

The nonzero elements in 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 nag_sparse_complex_gen_sort (f11zn) may be used to order the elements in this way.

3: **irow**(**nnz**) – INTEGER array
4: **icol**(**nnz**) – INTEGER array

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

Constraints:

$1 \leq \mathbf{irow}(i) \leq \mathbf{n}$ and $1 \leq \mathbf{icol}(i) \leq \mathbf{n}$, 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 $i = 2, 3, \ldots, \mathbf{nnz}$.

5: **check** – CHARACTER(1)

Specifies whether or not the CS representation of the matrix A, values of **n**, **nnz**, **irow** and **icol** should be checked.

check = 'C'
Checks are carried on the values of **n**, **nnz**, **irow** and **icol**.

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

See also Section 9.2.

Constraint: **check** = 'C' or 'N'.

6: **x**(**n**) – COMPLEX (KIND=nag_wp) array

The vector x.

5.2 Optional Input Parameters

1: **n** – INTEGER

Default: the dimension of the array **x**.

n, the order of the matrix A.

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

2: **nz** – INTEGER

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 matrix A.

Constraint: $1 \leq \mathbf{nnz} \leq \mathbf{n}^2$.

5.3 Output Parameters

1: **y**(**n**) – COMPLEX (KIND=nag_wp) array

The vector y.

2: **ifail** – INTEGER

ifail = 0 unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail = 1

On entry, **trans** ≠ 'N' or 'T',
or **check** ≠ 'C' or 'N'.

ifail $= 2$

On entry, **n** < 1,
or **nnz** < 1,
or **nnz** $>$ **n**^{2}.

ifail $= 3$

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

$1 \leq$ **irow**$(i) \leq$ **n** and $1 \leq$ **icol**$(i) \leq$ **n**, for $i = 1, 2, \ldots,$ **nnz**;

irow$(i-1) <$ **irow**(i), or **irow**$(i-1) =$ **irow**(i) and **icol**$(i-1) <$ **icol**(i), for $i = 2, 3, \ldots,$ **nnz**.

Therefore a nonzero element has been supplied which does not lie within the matrix A, is out of order, or has duplicate row and column indices. Call nag_sparse_complex_gen_sort (f11zn) to reorder and sum or remove duplicates.

ifail $= -99$

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

ifail $= -399$

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

ifail $= -999$

Dynamic memory allocation failed.

7 Accuracy

The computed vector y satisfies the error bound:

$\|y - Ax\|_\infty \leq c(n)\epsilon\|A\|_\infty\|x\|_\infty$, if **trans** $=$ 'N', or

$\|y - A^{\mathrm{H}}x\|_\infty \leq c(n)\epsilon\|A^{\mathrm{H}}\|_\infty\|x\|_\infty$, if **trans** $=$ 'T',

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 nag_sparse_complex_gen_matvec (f11xn) is proportional to **nnz**.

8.2 Use of check

It is expected that a common use of nag_sparse_complex_gen_matvec (f11xn) will be to compute the matrix-vector product required in the application of nag_sparse_complex_gen_basic_solver (f11bs) to sparse complex linear systems. In this situation nag_sparse_complex_gen_matvec (f11xn) 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 complex sparse matrix A and a vector x. It then calls nag_sparse_complex_gen_matvec (f11xn) to compute the matrix-vector product $y = Ax$ and the conjugate transposed matrix-vector product $y = A^{\mathrm{H}}x$.

9.1 Program Text

```
     function f11xn_example

fprintf('f11xn example results\n\n');

a = [ 2 + 3i   1 - 4i                                  ...
                       1 + 0i  -1 - 2i                 ...
      4 + 1i           0 + 1i             1 + 3i ...
                                 0 - 1i    2 - 6i ...
             -2 + 0i                      3 + 1i];
irow = [nag_int(1) 1   2 2   3 3 3   4 4   5 5];
icol = [nag_int(1) 2   3 4   1 3 5   4 5   2 5];
x = [ 0.70 + 0.21i;
      0.16 - 0.43i;
      0.52 + 0.97i;
      0.77 + 0.00i;
      0.28 - 0.64i];

% Calculate matrix-vector product
trans = 'N';
check = 'C';
[y, ifail] = f11xn( ...
                    trans, a, irow, icol, check, x);
fprintf('\nMatrix-vector product\n');
disp(y);

% Calculate conjugate transposed matrix-vector product
trans = 'T';
check = 'N';
[y, ifail] = f11xn( ...
                    trans, a, irow, icol, check, x);
fprintf('\nConjugate transposed matrix-vector product\n');
disp(y);
```

9.2 Program Results

```
     f11xn example results

Matrix-vector product
  -0.7900 + 1.4500i
  -0.2500 - 0.5700i
   3.8200 + 2.2600i
  -3.2800 - 3.7300i
   1.1600 - 0.7800i

Conjugate transposed matrix-vector product
   5.0800 + 1.6800i
  -0.7000 + 4.2900i
   1.1300 - 0.9500i
   0.7000 + 1.5200i
   5.1700 + 1.8300i
```