NAG Toolbox

nag_sparse_complex_herm_sort (f11zp)

1 Purpose

nag_sparse_complex_herm_sort (f11zp) sorts the nonzero elements of a sparse complex Hermitian matrix, represented in symmetric coordinate storage format.

2 Syntax

```
[nnz, a, irow, icol, istr, ifail] = nag_sparse_complex_herm_sort(n, nnz, a, irow, icol, dup, zer)
[nnz, a, irow, icol, istr, ifail] = f11zp(n, nnz, a, irow, icol, dup, zer)
```

3 Description

nag_sparse_complex_herm_sort (f11zp) 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 function also returns a pointer array **istr** to the starting address of each row in A.

4 References

None.

5 Parameters

5.1 Compulsory Input Parameters

1: **n** – INTEGER

n, the order of the matrix A.

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

2: **nz** – INTEGER

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

Constraint: $\mathbf{nnz} \geq 0$.

3: **a**(:) – COMPLEX (KIND=nag_wp) array

The dimension of the array **a** must be at least $\max(1, \mathbf{nnz})$

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.

4: **irow**(:) – INTEGER array

The dimension of the array **irow** must be at least $\max(1, \mathbf{nnz})$

The row indices corresponding to the nonzero elements supplied in the array **a**.

Constraint: $1 \leq \mathbf{irow}(i) \leq \mathbf{n}$, for $i = 1, 2, \ldots, \mathbf{nnz}$.

5: **icol**(:) – INTEGER array

The dimension of the array **icol** must be at least $\max(1, \mathbf{nnz})$

The column indices corresponding to the nonzero elements supplied in the array **a**.

Constraint: $1 \leq \mathbf{icol}(i) \leq \mathbf{irow}(i)$, for $i = 1, 2, \ldots, \mathbf{nnz}$.

6: **dup** – CHARACTER(1)

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 function fails with **ifail** = 3 on detecting a duplicate.

Constraint: **dup** = 'R', 'S' or 'F'.

7: **zer** – CHARACTER(1)

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 function fails with **ifail** = 4 on detecting a zero.

Constraint: **zer** = 'R', 'K' or 'F'.

5.2 Optional Input Parameters

None.

5.3 Output Parameters

1: **nz** – INTEGER

The number of lower triangular nonzero elements with unique row and column indices.

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

The dimension of the array **a** will be $\max(1, \mathbf{nnz})$

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.

3: **irow**(:) – INTEGER array

The dimension of the array **irow** will be $\max(1, \mathbf{nnz})$

The first **nnz** elements contain the row indices corresponding to the nonzero elements returned in the array **a**.

4: **icol**(:) – INTEGER array

The dimension of the array **icol** will be $\max(1, \mathbf{nnz})$

The first **nnz** elements contain the column indices corresponding to the nonzero elements returned in the array **a**.

5: **istr**$(\mathbf{n}+\mathbf{1})$ – INTEGER array

istr(i), for $i = 1, 2, \ldots, \mathbf{n}$, is the starting address in the arrays **a**, **irow** and **icol** of row i of the matrix A. **istr**$(\mathbf{n}+1)$ is the address of the last nonzero element in A plus one.

6: **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, **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:

$$1 \leq \mathbf{irow}(i) \leq \mathbf{n},$$

$$1 \leq \mathbf{icol}(i) \leq \mathbf{irow}(i),$$

for $i = 1, 2, \ldots, \mathbf{nnz}$.

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.

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

Not applicable.

8 Further Comments

The time taken for a call to nag_sparse_complex_herm_sort (f11zp) 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 nag_sparse_complex_herm_sort (f11zp) to reorder the nonzero elements, and outputs the original and the reordered representations.

9.1 Program Text

```
function f11zp_example

fprintf('f11zp example results\n\n');

n = nag_int(4);
nz = nag_int(9);
irow = zeros(nz,1,nag_int_name);
icol = zeros(nz,1,nag_int_name);
a = zeros(nz,1);
a(1:nz)    = [ 1 + 2i;     0 + 0i;      0 + 3i;
               3 - 5i;     4 + 2i;      0 + 3i;
               2 + 4i;     1 - 1i;      1 + 3i];
irow(1:nz) = [ 3;          2;           3;
               4;          1;           2;
               3;          3;           3];
icol(1:nz) = [ 2;          1;           2;
               4;          1;           2;
               3;          2;           2];
fprintf('Number of elements in original sparse matrix: %5d\n',nz);
disp('Original matrix ordering:');
fprintf('   k      a(k)      i_k  j_k\n');
for j = 1:nz
  fprintf('%4d  (%4.1f,%4.1f)%5d%5d\n', j, real(a(j)), imag(a(j)), ...
   irow(j),icol(j));
end

% Sum duplicates and remove zeros
dup = 'S';
zero = 'R';
[nz, a, irow, icol, istr, ifail] = ...
  f11zp( ...
  n, nz, a, irow, icol, dup, zero);

fprintf('\nNumber of elements in reordered sparse matrix: %5d\n',nz);
disp('New ordering:');
fprintf('   k      a(k)      i_k  j_k\n');
for j = 1:nz
  fprintf('%4d  (%4.1f,%4.1f)%5d%5d\n', j, real(a(j)), imag(a(j)), ...
   irow(j),icol(j));
end
```

9.2 Program Results

```
f11zp example results

Number of elements in original sparse matrix:     9
Original matrix ordering:
   k      a(k)      i_k  j_k
   1  ( 1.0, 2.0)    3    2
   2  ( 0.0, 0.0)    2    1
   3  ( 0.0, 3.0)    3    2
   4  ( 3.0,-5.0)    4    4
   5  ( 4.0, 2.0)    1    1
   6  ( 0.0, 3.0)    2    2
   7  ( 2.0, 4.0)    3    3
   8  ( 1.0,-1.0)    3    2
   9  ( 1.0, 3.0)    3    2

Number of elements in reordered sparse matrix:     5
New ordering:
   k      a(k)      i_k  j_k
```

```
1  ( 4.0, 2.0)    1    1
2  ( 0.0, 3.0)    2    2
3  ( 3.0, 7.0)    3    2
4  ( 2.0, 4.0)    3    3
5  ( 3.0,-5.0)    4    4
```