NAG Toolbox

nag_sort_realvec_rank_rearrange (m01ea)

1 Purpose

nag_sort_realvec_rank_rearrange (m01ea) rearranges a vector of real numbers into the order specified by a vector of ranks.

2 Syntax

```
[rv, irank, ifail] = nag_sort_realvec_rank_rearrange(rv, m1, irank, 'm2', m2)
[rv, irank, ifail] = m01ea(rv, m1, irank, 'm2', m2)
```

3 Description

nag_sort_realvec_rank_rearrange (m01ea) is designed to be used typically in conjunction with the M01D ranking functions. After one of the M01D functions has been called to determine a vector of ranks, nag_sort_realvec_rank_rearrange (m01ea) can be called to rearrange a vector of real numbers into the rank order. If the vector of ranks has been generated in some other way, then nag_sort_permute_check (m01zb) should be called to check its validity before nag_sort_realvec_rank_rearrange (m01ea) is called.

4 References

None.

5 Parameters

5.1 Compulsory Input Parameters

1: **rv**(**m2**) – REAL (KIND=nag_wp) array

Elements **m1** to **m2** of **rv** must contain real values to be rearranged.

2: **m1** – INTEGER

m1 and **m2** must specify the range of the ranks supplied in **irank** and the elements of **rv** to be rearranged.

Constraint: $0 < \mathbf{m1} \leq \mathbf{m2}$.

3: **irank**(**m2**) – INTEGER array

Elements **m1** to **m2** of **irank** must contain a permutation of the integers **m1** to **m2**, which are interpreted as a vector of ranks.

5.2 Optional Input Parameters

1: **m2** – INTEGER

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

m1 and **m2** must specify the range of the ranks supplied in **irank** and the elements of **rv** to be rearranged.

Constraint: $0 < \mathbf{m1} \leq \mathbf{m2}$.

5.3 Output Parameters

1: **rv**(**m2**) – REAL (KIND=nag_wp) array

These values are rearranged into rank order. For example, if **irank**$(i) =$ **m1**, then the initial value of **rv**(i) is moved to **rv**(**m1**).

2: **irank**(**m2**) – INTEGER array

Used as internal workspace prior to being restored and hence is unchanged.

3: **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, **m2** < 1,
or **m1** < 1,
or **m1** $>$ **m2**.

ifail $= 2$

Elements **m1** to **m2** of **irank** contain a value outside the range **m1** to **m2**.

ifail $= 3$

Elements **m1** to **m2** of **irank** contain a repeated 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.

If **ifail** $= 2$ or 3, elements **m1** to **m2** of **irank** do not contain a permutation of the integers **m1** to **m2**. On exit, the contents of **rv** may be corrupted. To check the validity of **irank** without the risk of corrupting **rv**, use nag_sort_permute_check (m01zb).

7 Accuracy

Not applicable.

8 Further Comments

The average time taken by the function is approximately proportional to n, where $n =$ **m2** $-$ **m1** $+ 1$.

9 Example

This example reads a matrix of real numbers and rearranges its rows so that the elements of the kth column are in ascending order. To do this, the program first calls nag_sort_realvec_rank (m01da) to rank the elements of the kth column, and then calls nag_sort_realvec_rank_rearrange (m01ea) to rearrange each column into the order specified by the ranks. The value of k is read from the datafile.

9.1 Program Text

```
     function m01ea_example

fprintf('m01ea example results\n\n');

k  = nag_int(1);

rm = [6 5 4;
      5 2 1;
      2 4 9;
      4 9 6;
      4 9 5;
      4 1 2;
      3 4 1;
      2 4 6;
      1 6 4;
      9 3 2;
      6 2 5;
      4 9 6];

m1 = nag_int(1);
m2 = nag_int(size(rm,1));
n  = nag_int(size(rm,2));

% Rank by column k
order = 'Ascending';
[irank, ifail] = m01da(rm(:,k), m1, order);

% Sort columns by ranking
for j = 1:n
  [rm(:,j), irank, ifail] = m01ea(rm(:,j), m1, irank);
end

fprintf('Matrix sorted on column %2d\n', k);
for i = m1:m2
  fprintf('%7.1f',rm(i,:));
  fprintf('\n');
end
```

9.2 Program Results

```
     m01ea example results

Matrix sorted on column  1
    1.0    6.0    4.0
    2.0    4.0    9.0
    2.0    4.0    6.0
    3.0    4.0    1.0
    4.0    9.0    6.0
    4.0    9.0    5.0
    4.0    1.0    2.0
    4.0    9.0    6.0
    5.0    2.0    1.0
    6.0    5.0    4.0
    6.0    2.0    5.0
    9.0    3.0    2.0
```