NAG C Library Function Document

nag_2d_panel_sort (e02zac)

1 Purpose

nag_2d_panel_sort (e02zac) sorts two-dimensional data into rectangular panels.

2 Specification

```
#include <nag.h>
#include <nage02.h>
```

```
void nag_2d_panel_sort (Integer px, Integer py, const double lamda[],
     const double mu[], Integer m, const double x[], const double y[],
     Integer point[], NagError *fail)
```

3 Description

A set of m data points with rectangular Cartesian co-ordinates x_r, y_r are sorted into panels defined by lines parallel to the y and x axes. The intercepts of these lines on the x and y axes are given in **lamda**$[i-1]$, for $i = 5, 6, \ldots, \mathbf{px} - 4$ and **mu**$[j-1]$, for $j = 5, 6, \ldots, \mathbf{py} - 4$, respectively. The function orders the data so that all points in a panel occur before data in succeeding panels, where the panels are numbered from bottom to top and then left to right, with the usual arrangement of axes, as shown in the diagram. Within a panel the points maintain their original order.

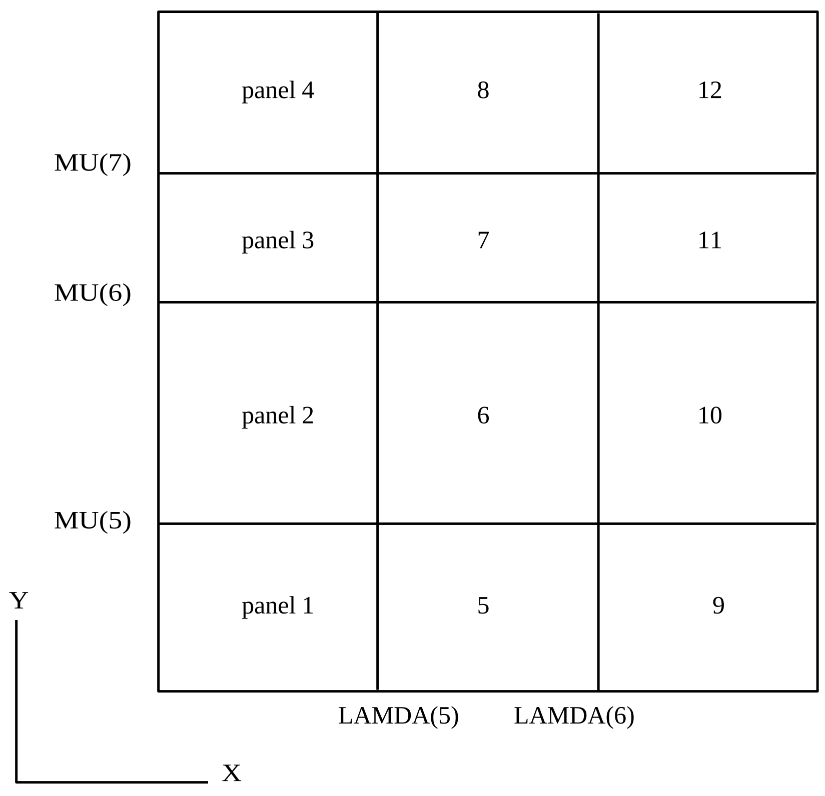

Figure 1

A data point lying exactly on one or more panel sides is taken to be in the highest-numbered panel adjacent to the point. The function does not physically rearrange the data, but provides the array **point** which contains a linked list for each panel, pointing to the data in that panel. The total number of panels is $(\mathbf{px} - 7) \times (\mathbf{py} - 7)$.

4 References

None.

5 Arguments

1: **px** – Integer *Input*
2: **py** – Integer *Input*

On entry: **px** and **py** must specify eight more than the number of intercepts on the x axis and y axis, respectively.

Constraint: $\mathbf{px} \geq 8$ and $\mathbf{py} \geq 8$.

3: **lamda**[**px**] – const double *Input*

On entry: **lamda**[4] to **lamda**[**px** − 5] must contain, in non-decreasing order, the intercepts on the x axis of the sides of the panels parallel to the y axis.

4: **mu**[**py**] – const double *Input*

On entry: **mu**[4] to **mu**[**py** − 5] must contain, in non-decreasing order, the intercepts on the y axis of the sides of the panels parallel to the x axis.

5: **m** – Integer *Input*

On entry: the number m of data points.

6: **x**[**m**] – const double *Input*
7: **y**[**m**] – const double *Input*

On entry: the co-ordinates of the rth data point (x_r, y_r), for $r = 1, 2, \ldots, m$.

8: **point**[*dim*] – Integer *Output*

Note: the dimension, *dim*, of the array **point** must be at least $\mathbf{m} + (\mathbf{px} - 7) \times (\mathbf{py} - 7)$.

On exit: for $i = 1, 2, \ldots, (\mathbf{px} - 7) \times (\mathbf{py} - 7)$, $\mathbf{point}[m + i - 1] = \mathrm{I1}$ is the index of the first point in panel i, $\mathbf{point}[\mathrm{I0}] = \mathrm{I2}$ is the index of the second point in panel i and so on.

$\mathbf{point}[\mathrm{I}n - 1] = 0$ indicates that $\mathbf{x}[\mathrm{I}n - 1], \mathbf{y}[\mathrm{I}n - 1]$ was the last point in the panel.

9: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 2.6 of the Essential Introduction).

6 Error Indicators and Warnings

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_BAD_PARAM

On entry, argument $\langle value \rangle$ had an illegal value.

NE_INT

On entry, $\mathbf{m} = \langle value \rangle$.
Constraint: $\mathbf{m} > 0$.

On entry, **px** $= \langle value \rangle$.
Constraint: **px** ≥ 8.

On entry, **py** $= \langle value \rangle$.
Constraint: **py** ≥ 8.

NE_INT_2

On entry, **px** $= \langle value \rangle$, **py** $= \langle value \rangle$.
Constraint: **px** ≥ 8 and **py** ≥ 8.

NE_INTERNAL_ERROR

An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please consult NAG for assistance.

NE_NOT_NON_DECREASING

On entry, **lamda**$[i-1] <$ **lamda**$[i-2]$:$i = \langle value \rangle$, **lamda**$[i-1] = \langle value \rangle$, **lamda**$[i-2] = \langle value \rangle$.

On entry, **mu**$[i-1] <$ **mu**$[i-2]$:$i = \langle value \rangle$, **mu**$[i-1] = \langle value \rangle$, **mu**$[i-2] = \langle value \rangle$.

7 Accuracy

Not applicable.

8 Further Comments

The time taken is approximately proportional to $m \times \log((\mathbf{px} - 7) \times (\mathbf{py} - 7))$.

This function was written to sort two dimensional data in the manner required by function nag_2d_spline_fit_panel (e02dac). The first 9 arguments of nag_2d_panel_sort (e02zac) are the same as the arguments in nag_2d_spline_fit_panel (e02dac) which have the same name.

9 Example

This example program reads in data points and the intercepts of the panel sides on the x and y axes; it calls nag_2d_panel_sort (e02zac) to set up the index array **point**; and finally it prints the data points in panel order.

9.1 Program Text

```
/* nag_nag_2d_panel_sort (e02zac) Example Program.
 *
 * Copyright 2005 Numerical Algorithms Group.
 *
 * Mark 8, 2004.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nage02.h>

int main(void)
{
  /* Scalars */
  Integer exit_status, i, iadres, m, nadres, npoint, px, py;

  /* Arrays */
  double *lamda=0, *mu=0, *x=0, *y=0;
  Integer *point=0;

  /* Nag types */
  NagError fail;

  exit_status = 0;
  INIT_FAIL(fail);
  Vprintf( "nag_2d_panel_sort (e02zac) Example Program Results\n");

  /* Skip heading in data file */
  Vscanf("%*[^\n] ");

  while (scanf("%ld%*[^\n] ",   &m) != EOF)
    {
      if ( m > 0 )
        {
          /* Allocate memory */
          if ( !(x = NAG_ALLOC(m, double)) ||
               !(y = NAG_ALLOC(m, double)) )
            {
              Vprintf("Allocation failure\n");
              exit_status = -1;
              goto END;
            }
        }
      else
        {
          Vprintf("Invalid m.\n");
          exit_status = 1;
          return exit_status;
        }
      Vscanf("%ld%ld%*[^\n] ",   &px,  &py);
      nadres = (px - 7) * (py - 7);
      npoint = m+nadres;
      if (px >= 8 && py >= 8)
        {
          /* Allocate memory */
          if ( !(lamda = NAG_ALLOC(px, double)) ||
               !(mu = NAG_ALLOC(py, double)) ||
               !(point = NAG_ALLOC(npoint, Integer)) )
            {
              Vprintf("Allocation failure\n");
              exit_status = -1;
              goto END;
            }
        }
      else
        {
          Vprintf("Invalid px or py.\n");
          exit_status = 1;
          goto END;
        }

      /* Read data points and intercepts of panel sides */
      for (i = 1; i <= m; ++i)
        {
          Vscanf("%lf%lf",   &x[i - 1],  &y[i - 1]);
        }
      Vscanf("%*[^\n] ");

      if (px > 8)
        {
          for (i = 5; i <= px - 4; ++i)
            {
              Vscanf("%lf", &lamda[i - 1]);
            }
          Vscanf("%*[^\n] ");
        }
      if (py > 8)
        {
          for (i = 5; i <= py - 4; ++i)
            {
              Vscanf("%lf", &mu[i - 1]);
            }
          Vscanf("%*[^\n] ");
        }
      /* Sort points into panel order */

      /* nag_2d_panel_sort (e02zac).
       * Sort two-dimensional data into panels for fitting bicubic
       * splines
       */
      nag_2d_panel_sort(px, py, lamda, mu, m, x, y, point, &fail);
      if (fail.code != NE_NOERROR)
        {
          Vprintf("Error from nag_2d_panel_sort (e02zac).\n%s\n", fail.message);
          exit_status = 1;
          goto END;
        }

      /* Output points in panel order */
      for (i = 1; i <= nadres; ++i)
        {

          Vprintf("\n%s%4ld\n\n",   "Panel",  i);
          iadres = m + i;
          while ( (iadres = point[iadres - 1]) > 0)
            {
              Vprintf("%7.2f%7.2f\n",   x[iadres - 1],  y[iadres - 1]);
            }
        }
    END:
      if (lamda) NAG_FREE(lamda);
      if (mu) NAG_FREE(mu);
      if (x) NAG_FREE(x);
      if (y) NAG_FREE(y);
      if (point) NAG_FREE(point);
    }
  return exit_status;
}
```

9.2 Program Data

```
nag_2d_panel_sort (e02zac) Example Program Data
  10
   9
  10
0       0.77
0.70    1.06
1.44    0.33
0.21    0.44
1.01    0.50
1.84    0.02
0.71    1.95
1.00    1.20
0.54    0.04
1.53    0.18
1.00
0.80
1.20
```

9.3 Program Results

```
nag_2d_panel_sort (e02zac) Example Program Results

Panel   1

   0.00   0.77
   0.21   0.44
   0.54   0.04

Panel   2

   0.70   1.06
```

```
Panel   3

   0.71   1.95

Panel   4

   1.44   0.33
   1.01   0.50
   1.84   0.02
   1.53   0.18

Panel   5


Panel   6

   1.00   1.20
```