NAG Library Function Document

nag_running_median_smoother (g10cac)

1 Purpose

nag_running_median_smoother (g10cac) computes a smoothed data sequence using running median smoothers.

2 Specification

```
#include <nag.h>
#include <nagg10.h>
void nag_running_median_smoother (Nag_Smooth_Type smoother, Integer n,
     const double y[], double smooth[], double rough[], NagError *fail)
```

3 Description

Given a sequence of n observations recorded at equally spaced intervals, nag_running_median_smoother (g10cac) fits a smooth curve through the data using one of two smoothers. They are based on the use of running medians and averages to summarise the overlapping segments. The fit is called the smooth, the residuals the rough and they obey the following:

Data = Smooth + Rough

The two smoothers are :

1. 4253H, twice consisting of a running median of 4, then 2, then 5, then 3 followed by Hanning. Hanning is a running weighted average, the weights being 1/4, 1/2 and 1/4. The result of this smoothing is then reroughed by computing residuals, applying the same smoother to them and adding the result to the smooth of the first pass.
2. 3RSSH, twice consisting of a running median of 3, two splitting operations named S to improve the smooth sequence, each of which is followed by a running median of 3, and finally Hanning. The end points are dealt with using the method described by Velleman and Hoaglin (1981). The full smoother 3RSSH, twice is produced by reroughing as described above.

The compound smoother 4253H, twice is recommended. The smoother 3RSSH, twice is popular when calculating by hand as it requires simpler computations and is included for comparison purposes.

4 References

Tukey J W (1977) *Exploratory Data Analysis* Addison–Wesley

Velleman P F and Hoaglin D C (1981) *Applications, Basics, and Computing of Exploratory Data Analysis* Duxbury Press, Boston, MA

5 Arguments

1: **smoother** – Nag_Smooth_Type *Input*

On entry: **smoother** must specify the method to be used.

smoother = Nag_4253H
4253H, twice is used.

smoother = Nag_3RSSH
3RSSH, twice is used.

Constraint: **smoother** = Nag_4253H or Nag_3RSSH.

2: **n** – Integer *Input*

On entry: the number, n, of the observations.

Constraint: $\mathbf{n} > 6$.

If $\mathbf{n} \leq 6$ then the sequence is not long enough to carry out smoothing.

3: **y**[**n**] – const double *Input*

On entry: the sample observations.

4: **smooth**[**n**] – double *Output*

On exit: contains the smooth.

5: **rough**[**n**] – double *Output*

On exit: contains the rough.

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

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_BAD_PARAM

On entry, argument **smoother** had an illegal value.

NE_INT_ARG_LE

On entry, $\mathbf{n} = \langle value \rangle$.
Constraint: $\mathbf{n} > 6$.

7 Accuracy

Not applicable.

8 Parallelism and Performance

Not applicable.

9 Further Comments

None.

10 Example

The example program reads in a sequence of 49 data taken from Tukey (1977), above. Results are obtained using the two smoothing methods described.

10.1 Program Text

```
/* nag_running_median_smoother (g10cac) Example Program.
 *
 * Copyright 1992 Numerical Algorithms Group.
 *
 * Mark 3, 1992.
 * Mark 8 revised, 2004.
 */

#include <nag.h>
```

```
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagg10.h>

int main(void)
{
  Integer         exit_status = 0, i, isel, n;
  NagError        fail;
  Nag_Smooth_Type smoother;
  double          *rough = 0, *smooth = 0, *y = 0;

  INIT_FAIL(fail);

  printf(
          "nag_running_median_smoother (g10cac) Example Program Results\n");
  /* Skip heading in data file */
  scanf("%*[^\n]");
  scanf("%ld", &n);
  if (n >= 1)
    {
      if (!(rough = NAG_ALLOC(n, double)) ||
          !(smooth = NAG_ALLOC(n, double)) ||
          !(y = NAG_ALLOC(n, double)))
        {
          printf("Allocation failure\n");
          exit_status = -1;
          goto END;
        }
    }
  else
    {
      printf("Invalid n.\n");
      exit_status = 1;
      return exit_status;
    }
  for (i = 0; i < n; ++i)
    scanf("%lf", &y[i]);

  for (isel = 1; isel <= 2; ++isel)
    {
      /* Select the smoothing method. */
      if (isel == 1)
        smoother = Nag_4253H;
      else if (isel == 2)
        smoother = Nag_3RSSH;

      if (smoother == Nag_4253H)
        printf("\nExample 1: Data smoothing using  4253H method.\n\n");
      else if (smoother == Nag_3RSSH)
        printf(
                "\n\nExample 2: Data smoothing using  3RSSH method.\n\n");

      /* nag_running_median_smoother (g10cac).
       * Compute smoothed data sequence using running median
       * smoothers
       */
      nag_running_median_smoother(smoother, n, y, smooth, rough, &fail);
      if (fail.code != NE_NOERROR)
        {
          printf(
                  "Error from nag_running_median_smoother (g10cac).\n%s\n",
                  fail.message);
          exit_status = 1;
          goto END;
        }

      printf("     index        data          smooth          rough\n");
      for (i = 0; i < n; ++i)
        printf("%9ld%15.4f%15.4f%15.4f\n", i, y[i], smooth[i],
                rough[i]);
    }
```

```
 END:
  NAG_FREE(rough);
  NAG_FREE(smooth);
  NAG_FREE(y);
  return exit_status;
}
```

10.2 Program Data

```
nag_running_median_smoother (g10cac) Example Program Data
49
569.0 416.0 422.0 565.0 484.0 520.0 573.0 518.0 501.0 505.0
468.0 382.0 310.0 334.0 359.0 372.0 439.0 446.0 349.0 395.0
461.0 511.0 583.0 590.0 620.0 578.0 534.0 631.0 600.0 438.0
516.0 534.0 467.0 457.0 392.0 467.0 500.0 493.0 410.0 412.0
416.0 403.0 422.0 459.0 467.0 512.0 534.0 552.0 545.0
```

10.3 Program Results

```
nag_running_median_smoother (g10cac) Example Program Results

Example 1: Data smoothing using  4253H method.

     index        data          smooth          rough
        0       569.0000       416.0000       153.0000
        1       416.0000       416.0000         0.0000
        2       422.0000       431.5000        -9.5000
        3       565.0000       473.0000        92.0000
        4       484.0000       509.5000       -25.5000
        5       520.0000       520.6875        -0.6875
        6       573.0000       521.5625        51.4375
        7       518.0000       518.0000         0.0000
        8       501.0000       510.0000        -9.0000
        9       505.0000       496.5000         8.5000
       10       468.0000       455.2500        12.7500
       11       382.0000       387.5000        -5.5000
       12       310.0000       339.7500       -29.7500
       13       334.0000       334.9375        -0.9375
       14       359.0000       353.9375         5.0625
       15       372.0000       376.1250        -4.1250
       16       439.0000       392.2500        46.7500
       17       446.0000       396.2500        49.7500
       18       349.0000       403.0000       -54.0000
       19       395.0000       427.2500       -32.2500
       20       461.0000       461.3750        -0.3750
       21       511.0000       513.3125        -2.3125
       22       583.0000       567.5625        15.4375
       23       590.0000       590.0000         0.0000
       24       620.0000       593.5000        26.5000
       25       578.0000       595.2500       -17.2500
       26       534.0000       590.9375       -56.9375
       27       631.0000       566.8125        64.1875
       28       600.0000       531.5000        68.5000
       29       438.0000       516.0000       -78.0000
       30       516.0000       516.0000         0.0000
       31       534.0000       501.8750        32.1250
       32       467.0000       473.6250        -6.6250
       33       457.0000       457.0000         0.0000
       34       392.0000       452.0000       -60.0000
       35       467.0000       440.1250        26.8750
       36       500.0000       421.3750        78.6250
       37       493.0000       412.0000        81.0000
       38       410.0000       412.0000        -2.0000
       39       412.0000       412.0000         0.0000
       40       416.0000       411.0625         4.9375
       41       403.0000       410.6875        -7.6875
       42       422.0000       422.0000         0.0000
       43       459.0000       446.6250        12.3750
       44       467.0000       476.3750        -9.3750
       45       512.0000       509.0000         3.0000
```

46	534.0000	534.0000	0.0000
47	552.0000	545.0000	7.0000
48	545.0000	547.7500	-2.7500

Example 2: Data smoothing using 3RSSH method.

index	data	smooth	rough
0	569.0000	491.3750	77.6250
1	416.0000	491.3750	-75.3750
2	422.0000	491.3750	-69.3750
3	565.0000	498.8828	66.1172
4	484.0000	514.9375	-30.9375
5	520.0000	524.6602	-4.6602
6	573.0000	525.0352	47.9648
7	518.0000	521.1602	-3.1602
8	501.0000	512.5742	-11.5742
9	505.0000	493.1680	11.8320
10	468.0000	449.7422	18.2578
11	382.0000	391.6133	-9.6133
12	310.0000	353.4297	-43.4297
13	334.0000	343.8438	-9.8438
14	359.0000	355.1602	3.8398
15	372.0000	382.7930	-10.7930
16	439.0000	405.5469	33.4531
17	446.0000	411.8633	34.1367
18	349.0000	411.5586	-62.5586
19	395.0000	420.9375	-25.9375
20	461.0000	456.1250	4.8750
21	511.0000	513.8516	-2.8516
22	583.0000	565.2422	17.7578
23	590.0000	589.4688	0.5312
24	620.0000	594.7188	25.2812
25	578.0000	594.5625	-16.5625
26	534.0000	591.8125	-57.8125
27	631.0000	583.8438	47.1562
28	600.0000	569.0312	30.9688
29	438.0000	546.3438	-108.3438
30	516.0000	517.2578	-1.2578
31	534.0000	489.6445	44.3555
32	467.0000	471.2383	-4.2383
33	457.0000	463.4844	-6.4844
34	392.0000	464.1875	-72.1875
35	467.0000	468.4688	-1.4688
36	500.0000	470.6094	29.3906
37	493.0000	462.2617	30.7383
38	410.0000	438.5703	-28.5703
39	412.0000	416.1094	-4.1094
40	416.0000	408.8711	7.1289
41	403.0000	412.1836	-9.1836
42	422.0000	424.8750	-2.8750
43	459.0000	448.1445	10.8555
44	467.0000	478.7578	-11.7578
45	512.0000	510.0234	1.9766
46	534.0000	534.1250	-0.1250
47	552.0000	547.0000	5.0000
48	545.0000	550.9375	-5.9375