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

```c
#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

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:* \( n > 6 \).

If \( n \leq 6 \) then the sequence is not long enough to carry out smoothing.

3: \( y[n] \) – const double

*Input*

*On entry:* the sample observations.

4: \( \text{smooth}[n] \) – double

*Output*

*On exit:* contains the smooth.

5: \( \text{rough}[n] \) – double

*Output*

*On exit:* contains the rough.

6: \( \text{fail} \) – NAG Error *

*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,* \( n = \text{(value)} \).

*Constraint:* \( n > 6 \).

## 7 Accuracy

Not applicable.

## 8 Parallelism and Performance

Not applicable.

## 9 Further Comments

Internal changes have been made to this function as follows:

At Mark 25: nag_running_median_smother (g10cac) is a smoothing function with two possible smoothing methods. The function was previously using the incorrect method (i.e., if you asked for method A you would get method B, and vice versa).

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

int main(void)
{
    Integer exit_status = 0, i, n;
    NagError fail;
    Nag_Smooth_Type smoother;
    double *rough0 = 0, *smooth0 = 0, *rough1 = 0, *smooth1 = 0, *y = 0;

    INIT_FAIL(fail);

    printf("nag_running_median_smoother (g10cac) Example Program Results\n");
    /* Skip heading in data file */
    #ifdef _WIN32
        scanf_s("%*[\n]");
    #else
        scanf("%*[\n]");
    #endif

    #ifdef _WIN32
        scanf("%"NAG_IFMT"", &n);
    #else
        scanf("%"NAG_IFMT"", &n);
    #endif

    if (n >= 1)
    {
        if (!(rough0 = NAG_ALLOC(n, double)) ||
            !(smooth0 = NAG_ALLOC(n, double)) ||
            !(rough1 = NAG_ALLOC(n, double)) ||
            !(smooth1 = 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)
    {
        #ifdef _WIN32
            scanf_s("%lf", &y[i]);
        #else
            scanf("%lf", &y[i]);
        #endif
    }
}

/* nag_running_median_smoother (g10cac).
   * Compute smoothed data sequence using running median smoothers
   *
   * Smooth sequence using 3RSSH,twice */

    smoother = Nag_3RSSH;
    nag_running_median_smoother(smoother, n, y, smooth1, rough1, &fail);
    if (fail.code != NE_NOERROR)
printf("Error from nag_running_median_smother (g10cac).\n%s\n", fail.message);
exit_status = 1;
goto END;
}
/* Smooth sequence using 4253H,twice */
smoother = Nag_4253H;
nag_running_median_smother(smoother, n, y, smooth0, rough0, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_running_median_smother (g10cac).\n%s\n", fail.message);
    exit_status = 1;
goto END;
}
/* Display results */
printf("\n");
printf(" Index Data Smooth Rough Smooth Rough\n");
for (i = 0; i < n; ++i)
    printf("%4"NAG_IFMT" %10.1f %12.1f %12.1f %12.1f %12.1f\n", i, y[i], smooth1[i], rough1[i], smooth0[i], rough0[i]);
END:
NAG_FREE(rough0);
NAG_FREE(smooth0);
NAG_FREE(rough1);
NAG_FREE(smooth1);
NAG_FREE(y);
return exit_status;
}

10.2 Program Data
nag_running_median_smother (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_smother (g10cac) Example Program Results

<table>
<thead>
<tr>
<th>Index</th>
<th>Data</th>
<th>Using 3RSSH,twice</th>
<th>Using 4253H,twice</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Smooth</td>
<td>Rough</td>
</tr>
<tr>
<td>0</td>
<td>569.0</td>
<td>416.0</td>
<td>153.0</td>
</tr>
<tr>
<td>1</td>
<td>416.0</td>
<td>416.0</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>422.0</td>
<td>431.5</td>
<td>-9.5</td>
</tr>
<tr>
<td>3</td>
<td>565.0</td>
<td>473.0</td>
<td>92.0</td>
</tr>
<tr>
<td>4</td>
<td>484.0</td>
<td>509.5</td>
<td>-25.5</td>
</tr>
<tr>
<td>5</td>
<td>520.0</td>
<td>520.7</td>
<td>-0.7</td>
</tr>
<tr>
<td>6</td>
<td>573.0</td>
<td>521.6</td>
<td>51.4</td>
</tr>
<tr>
<td>7</td>
<td>512.0</td>
<td>518.0</td>
<td>0.0</td>
</tr>
<tr>
<td>8</td>
<td>501.0</td>
<td>510.0</td>
<td>-9.0</td>
</tr>
<tr>
<td>9</td>
<td>505.0</td>
<td>496.5</td>
<td>8.5</td>
</tr>
<tr>
<td>10</td>
<td>468.0</td>
<td>455.2</td>
<td>12.8</td>
</tr>
<tr>
<td>11</td>
<td>382.0</td>
<td>387.5</td>
<td>-5.5</td>
</tr>
<tr>
<td>12</td>
<td>310.0</td>
<td>339.8</td>
<td>-29.8</td>
</tr>
<tr>
<td>13</td>
<td>334.0</td>
<td>334.9</td>
<td>-0.9</td>
</tr>
<tr>
<td>14</td>
<td>359.0</td>
<td>353.9</td>
<td>5.1</td>
</tr>
<tr>
<td>15</td>
<td>372.0</td>
<td>376.1</td>
<td>-4.1</td>
</tr>
<tr>
<td>16</td>
<td>439.0</td>
<td>392.2</td>
<td>46.8</td>
</tr>
<tr>
<td>17</td>
<td>446.0</td>
<td>396.2</td>
<td>49.8</td>
</tr>
<tr>
<td>18</td>
<td>349.0</td>
<td>403.0</td>
<td>-54.0</td>
</tr>
<tr>
<td>19</td>
<td>395.0</td>
<td>427.2</td>
<td>-32.2</td>
</tr>
<tr>
<td>20</td>
<td>461.0</td>
<td>461.4</td>
<td>-0.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>21</td>
<td>511.0</td>
<td>513.3</td>
<td>-2.3</td>
</tr>
<tr>
<td>22</td>
<td>563.0</td>
<td>567.6</td>
<td>15.4</td>
</tr>
<tr>
<td>23</td>
<td>590.0</td>
<td>590.0</td>
<td>0.0</td>
</tr>
<tr>
<td>24</td>
<td>620.0</td>
<td>593.5</td>
<td>26.5</td>
</tr>
<tr>
<td>25</td>
<td>578.0</td>
<td>595.2</td>
<td>-17.2</td>
</tr>
<tr>
<td>26</td>
<td>534.0</td>
<td>590.9</td>
<td>-56.9</td>
</tr>
<tr>
<td>27</td>
<td>631.0</td>
<td>566.8</td>
<td>64.2</td>
</tr>
<tr>
<td>28</td>
<td>600.0</td>
<td>531.5</td>
<td>68.5</td>
</tr>
<tr>
<td>29</td>
<td>438.0</td>
<td>516.0</td>
<td>-78.0</td>
</tr>
<tr>
<td>30</td>
<td>516.0</td>
<td>516.0</td>
<td>0.0</td>
</tr>
<tr>
<td>31</td>
<td>534.0</td>
<td>501.9</td>
<td>32.1</td>
</tr>
<tr>
<td>32</td>
<td>467.0</td>
<td>473.6</td>
<td>-6.6</td>
</tr>
<tr>
<td>33</td>
<td>457.0</td>
<td>457.0</td>
<td>0.0</td>
</tr>
<tr>
<td>34</td>
<td>392.0</td>
<td>452.0</td>
<td>-60.0</td>
</tr>
<tr>
<td>35</td>
<td>467.0</td>
<td>440.1</td>
<td>26.9</td>
</tr>
<tr>
<td>36</td>
<td>500.0</td>
<td>421.4</td>
<td>78.6</td>
</tr>
<tr>
<td>37</td>
<td>493.0</td>
<td>412.0</td>
<td>81.0</td>
</tr>
<tr>
<td>38</td>
<td>410.0</td>
<td>412.0</td>
<td>-2.0</td>
</tr>
<tr>
<td>39</td>
<td>412.0</td>
<td>412.0</td>
<td>0.0</td>
</tr>
<tr>
<td>40</td>
<td>416.0</td>
<td>411.1</td>
<td>4.9</td>
</tr>
<tr>
<td>41</td>
<td>403.0</td>
<td>410.7</td>
<td>-7.7</td>
</tr>
<tr>
<td>42</td>
<td>422.0</td>
<td>422.0</td>
<td>0.0</td>
</tr>
<tr>
<td>43</td>
<td>459.0</td>
<td>446.6</td>
<td>12.4</td>
</tr>
<tr>
<td>44</td>
<td>467.0</td>
<td>476.4</td>
<td>-9.4</td>
</tr>
<tr>
<td>45</td>
<td>512.0</td>
<td>509.0</td>
<td>3.0</td>
</tr>
<tr>
<td>46</td>
<td>534.0</td>
<td>534.0</td>
<td>0.0</td>
</tr>
<tr>
<td>47</td>
<td>552.0</td>
<td>545.0</td>
<td>7.0</td>
</tr>
<tr>
<td>48</td>
<td>545.0</td>
<td>547.8</td>
<td>-2.8</td>
</tr>
</tbody>
</table>