NAG Library Routine Document

G10CAF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

G10CAF computes a smoothed data sequence using running median smoothers.

2 Specification

```
SUBROUTINE G10CAF (ITYPE, N, Y, SMOOTH, ROUGH, IFAIL)
INTEGER            ITYPE, N, IFAIL
REAL (KIND=nag_wp) Y(N), SMOOTH(N), ROUGH(N)
```

3 Description

Given a sequence of n observations recorded at equally spaced intervals, G10CAF fits a smooth curve through the data using one of two smoothers. The two smoothers are based on the use of running medians and averages to summarise overlapping segments. The fit and the residuals are called the smooth and the rough respectively. They obey the following:

$$\text{Data} = \text{Smooth} + \text{Rough}.$$

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: ITYPE – INTEGER *Input*

On entry: specifies the method to be used.

If ITYPE $= 0$, 4253H,twice is used.

If ITYPE $= 1$, 3RSSH,twice is used.

Constraint: ITYPE $= 0$ or 1.

2: N – INTEGER *Input*

On entry: n, the number of observations.

Constraint: N > 6.

3: Y(N) – REAL (KIND=nag_wp) array *Input*

On entry: the sample observations.

4: SMOOTH(N) – REAL (KIND=nag_wp) array *Output*

On exit: contains the smooth.

5: ROUGH(N) – REAL (KIND=nag_wp) array *Output*

On exit: contains the rough.

6: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, if you are not familiar with this argument, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: IFAIL $= 0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry IFAIL $= 0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

IFAIL $= 1$

On entry, ITYPE < 0,
or ITYPE > 1.

IFAIL $= 2$

On entry, N ≤ 6.

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

Not applicable.

8 Parallelism and Performance

G10CAF makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

Alternative methods of smoothing include the use of splines; see G10ABF and G10ACF.

10 Example

This example reads in a sequence of 49 observations on bituminous coal production (in millions of net tons per year) in the USA., 1920–1968 and is taken from Tukey (1977). For comparison purposes, both smoothers are applied to the data and the results are printed.

10.1 Program Text

```
    Program g10cafe

!     G10CAF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: g10caf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nin = 5, nout = 6
!     .. Local Scalars ..
      Integer                          :: i, ifail, itype, n
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable  :: rough0(:), rough1(:), smooth0(:),    &
                                          smooth1(:), y(:)
!     .. Executable Statements ..
      Write (nout,*) ' G10CAF Example Program Results'
      Write (nout,*)

!     Skip heading in data file
      Read (nin,*)

!     Read in the problem size
      Read (nin,*) n

      Allocate (y(n),rough0(n),smooth0(n),rough1(n),smooth1(n))

!     Read in data
      Read (nin,*) y(1:n)

!     Smooth sequence using 3RSSH,twice
      itype = 1
      ifail = 0
      Call g10caf(itype,n,y,smooth1,rough1,ifail)

!     Smooth sequence using 4253H,twice
      itype = 0
```

```
      ifail = 0
      Call g10caf(itype,n,y,smooth0,rough0,ifail)

!     Display results
      Write (nout,*)                                                    &
        '                    Using 3RSSH,twice         Using 4253H,twice'
      Write (nout,*)                                                    &
        ' Index     Data      Smooth        Rough       Smooth       Rough'
      Write (nout,99999)(i,y(i),smooth1(i),rough1(i),smooth0(i),rough0(i),i=1, &
        n)

99999 Format (1X,I4,F11.1,2F13.4,2F13.1)
    End Program g10cafe
```

10.2 Program Data

```
G10CAF Example Program Data
49                                                      :: N
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       :: End of Y
```

10.3 Program Results

```
 G10CAF Example Program Results

                     Using 3RSSH,twice         Using 4253H,twice
 Index     Data      Smooth        Rough       Smooth       Rough
   1      569.0     416.0000     153.0000        491.4         77.6
   2      416.0     416.0000       0.0000        491.4        -75.4
   3      422.0     431.5000      -9.5000        491.4        -69.4
   4      565.0     473.0000      92.0000        498.9         66.1
   5      484.0     509.5000     -25.5000        514.9        -30.9
   6      520.0     520.6875      -0.6875        524.7         -4.7
   7      573.0     521.5625      51.4375        525.0         48.0
   8      518.0     518.0000       0.0000        521.2         -3.2
   9      501.0     510.0000      -9.0000        512.6        -11.6
  10      505.0     496.5000       8.5000        493.2         11.8
  11      468.0     455.2500      12.7500        449.7         18.3
  12      382.0     387.5000      -5.5000        391.6         -9.6
  13      310.0     339.7500     -29.7500        353.4        -43.4
  14      334.0     334.9375      -0.9375        343.8         -9.8
  15      359.0     353.9375       5.0625        355.2          3.8
  16      372.0     376.1250      -4.1250        382.8        -10.8
  17      439.0     392.2500      46.7500        405.5         33.5
  18      446.0     396.2500      49.7500        411.9         34.1
  19      349.0     403.0000     -54.0000        411.6        -62.6
  20      395.0     427.2500     -32.2500        420.9        -25.9
  21      461.0     461.3750      -0.3750        456.1          4.9
  22      511.0     513.3125      -2.3125        513.9         -2.9
  23      583.0     567.5625      15.4375        565.2         17.8
  24      590.0     590.0000       0.0000        589.5          0.5
  25      620.0     593.5000      26.5000        594.7         25.3
  26      578.0     595.2500     -17.2500        594.6        -16.6
  27      534.0     590.9375     -56.9375        591.8        -57.8
  28      631.0     566.8125      64.1875        583.8         47.2
  29      600.0     531.5000      68.5000        569.0         31.0
  30      438.0     516.0000     -78.0000        546.3       -108.3
  31      516.0     516.0000       0.0000        517.3         -1.3
  32      534.0     501.8750      32.1250        489.6         44.4
  33      467.0     473.6250      -6.6250        471.2         -4.2
  34      457.0     457.0000       0.0000        463.5         -6.5
  35      392.0     452.0000     -60.0000        464.2        -72.2
  36      467.0     440.1250      26.8750        468.5         -1.5
  37      500.0     421.3750      78.6250        470.6         29.4
  38      493.0     412.0000      81.0000        462.3         30.7
  39      410.0     412.0000      -2.0000        438.6        -28.6
  40      412.0     412.0000       0.0000        416.1         -4.1
```

41	416.0	411.0625	4.9375	408.9	7.1
42	403.0	410.6875	-7.6875	412.2	-9.2
43	422.0	422.0000	0.0000	424.9	-2.9
44	459.0	446.6250	12.3750	448.1	10.9
45	467.0	476.3750	-9.3750	478.8	-11.8
46	512.0	509.0000	3.0000	510.0	2.0
47	534.0	534.0000	0.0000	534.1	-0.1
48	552.0	545.0000	7.0000	547.0	5.0
49	545.0	547.7500	-2.7500	550.9	-5.9