Applying Nonlinear Smoothers to Remove Impulsive Noise from Experimentally Sampled Data

A. E. Marquardt,* L. M. Toerien† and E. Terblanche.‡

(Received July 1990)

Abstract

Experimentally sampled data found to be corrupted by pulses of outliers is decontaminated by the application of a novel pair of unsymmetric smoothers. Basic concepts of nonlinear smoothing are introduced and the implementation of the particular smoothers is discussed.

Introduction

Figure 1 shows a set of measured data points from a towed model boat experiment. Buffer overflow during data transfer contaminated the signals with random spikes.

A well-developed theory exists for *linear filtering* (for example Hamming [1]). A linear filter, also referred to as a *linear smoother,* basically replaces each data point with the (weighted) average of it's closest neighbours. This is suitable for the processing of "well behaved" (e.g. *Gaussian*) noise. However, transient outliers of unreasonable amplitude can greatly distort some originally sound data and the resulting "smoothed" data remains smeared with unacceptable noise. Linear smoothing was therefore clearly not appropriate for the detection and removal of the impulsive noise found in the experimental data.

Various publications have reported on investigations into the theory, properties and applications of *nonlinear smoothers* [2-5], but analytical investigation is difficult and theoretical understanding still far from adequate. The usual task of a nonlinear smoother is to replace impulsive noise by acceptable points from its immediate surroundings, while leaving the values of other data points as far as possible unaltered.

For this specific application a novel pair of unsymmetric smoothers (*UL* and *LU*) by Rohwer [2] was applied. In the succeeding sections the basic concepts of nonlinear smoothers and the principles involved in the construction of the smoothers *UL* and *LU* are explained rather intuitively.

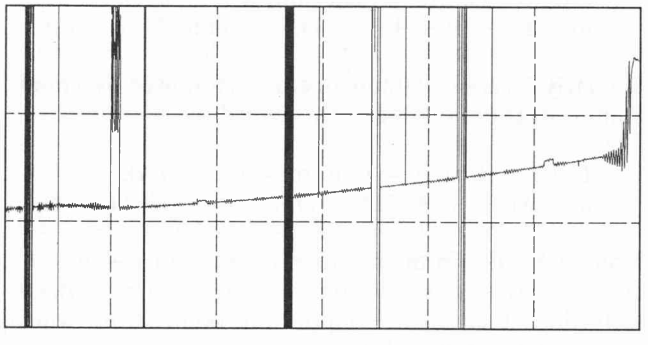

Figure 1 – Corrupted Experimental Data

*Engineer, M.Eng. student, University of Stellenbosch.
†Ph.D. student, Department of Applied Mathematics, University of Stellenbosch.
‡Senior Lecturer, Department of Mechanical Engineering, University of Stellenbosch.

Basic Concepts of Nonlinear Smoothers

For convenience the notations and conventions used in [2] are adhered to as far as possible.

1. A complete series of data points is denoted by a lower case letter while the letter followed by an index enclosed in parenthesis specifies a single point of the series. Thus x represents the series $\ldots, x(i-1), x(i), x(i+1), \ldots$.
2. The *set* of points (real numbers) obtained by selecting a *subinterval* of a series is denoted by an upper case letter followed by an index pair enclosed in parenthesis. The indices are the lower and upper index limits of the selected points. Thus $X(s, t) = \{x(s), x(s+1), \ldots, x(t-1), x(t)\}$ where it is understood that $s \leq t$.
3. A *smoother,* represented by an upper case letter, is defined as an operator (or algorithm) which maps each point of an *input series* to a corresponding point of an *output (smoothed) series*. Thus the statement $y = Sx$ signifies that smoother S operates on the input series x to produce the output series y. An index enclosed in parenthesis refers to a specific point of the series, e.g. $y(i) = Sx(i)$.
4. Nonlinear smoothers (henceforth referred to simply as *smoothers*) are typically *rank-based selectors* and concatenations of these. A rank-based selector determines the smoothed value at a point by *selecting* an input point from a *window* (which includes the particular point) based purely on the relative ranks amongst all the points presently in the window.
5. The *concatenation* (combination) of two smoothers refers to the resulting smoother obtained by letting one smoother operate on the output of another smoother, i.e. if $y = Sx$ and $z = Qy$ then smoother QS signifies smoother S followed by smoother Q and we write $z = QSx$. It is easy to see that a smoother of window size $n+1$ points followed by a second smoother of similar window size results in the final smoothed value at a point depending on $2n+1$ points. This effective total of a concatenation is termed the *support* of the resulting smoother (equivalent to the window size if a single rank order selector is under consideration).

Two central concepts involved in Rohwer's analysis are *ordering* and *idempotency* of smoother operators.

1. Smoothers are *ordered* by a comparison of their output series for general input. A series is equal to ($=$), less than or equal to ($\leq$) or greater than or equal to ($\geq$) another series, containing an equal number of points, if the ordering applies to *all* corresponding points of the

two series. Thus smoothers S and Q are either *equal* ($S = Q$) or S is *less than or equal to* Q ($S \leq Q$) or S is *greater than or equal to* Q ($S \geq Q$) if the ordering applies to the smoother outputs for all input series processed by both smoothers, otherwise S and Q are *not comparable*.

2. A smoother is *idempotent* if it does not alter its own output on repeated application. In other words, if the concatenation of smoother S with itself results in a smoother equal to S ($SS = S$) then S is idempotent (which obviously means that any power of S is equal to S).

As an example of simple nonlinear smoothers the popular median smoothers can be considered. A *median smoother* is a rank-based selector which, for the smoothed value of the point centred in the selector window, selects the median of all the points in the window. Mn, or simply M, is a median smoother with a window size of $2n + 1$ points. The smoothed value of each point $x(i)$ is the median of the $2n + 1$ points $X(i - n, i + n)$, i.e. the values of at least $n + 1$ points in the window $X(i - n, i + n)$ are less than or equal to $Mx(i)$ and at least $n + 1$ points of $X(i - n, i + n)$ are greater than or equal to $Mx(i)$. $Mx(i)$ is the $(n + 1)$-th ranked point of $X(i - n, i + n)$.

In Figure 2 a 5-point *running median* (i.e. $M2$) is compared to a 5-point *running average* (which is a simple linear filter). Note the stark contrast in the vacinity of transient outliers. The median smoother is *trend preserving*, points of monotone (increasing or decreasing) regions are unaltered.

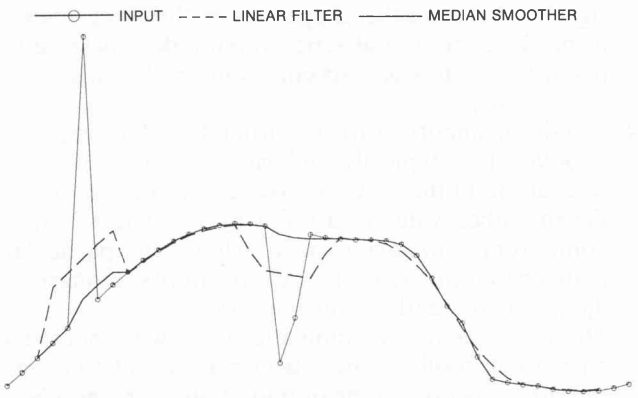

Figure 2 – Comparison of a 5-point Running Median and a 5-point Running Average

In theoretical studies the absence of data before the first point and after the last point of an input series is conveniently circumvented by defining the series as doubly infinite sequences of real numbers. In practice a finite input series is usually extended at the start by duplicates of the first point and at the end by duplicates of the last point as far as is needed to accommodate the window (or support in the case of a concatenation) for determining the smoothed value of the first and last points. An obvious advantage of this is that upward or downward trends at the start/end of the series are maintained.

Smoothers ***UL*** *and* ***LU***

Refering to Figure 3, consider for the moment a constant sequence with an occasional outlier in the upward direction. A procedure for removing such *upward pulses* from the otherwise constant sequence would be to apply a *running minimum*, i.e. a rank-order selector selecting the smallest (lowest ranked) element from the window. Such algorithm will remove all the upward pulses from the sequence (except for cases where the number of adjacent pulses is greater than or equal to the window size). However, if the series also contains transient *downward pulses* these will be *widened* by the running minimum. Furthermore, an *upward trend* in the series is *retarded* while a *downward trend* is *advanced*. These shortcomings can be overcome by following the running minimum by a *running maximum* which will then *reduce* the *downward pulses* to their original widths, *advance upward trends* and *retard downward trends* to restore monotone increasing/decreasing sections to their original state (except possibly some points at the transitions between upward and downward trends). Similarly it can be argued that a *running maximum followed by a running minimum* will *remove downward pulses* while *retaining upward and downward trends* and *upward pulses* more or less *in tact*. These fundamental ideas lead to the *basic pair of unsymmetric smoothers*, namely L and U.

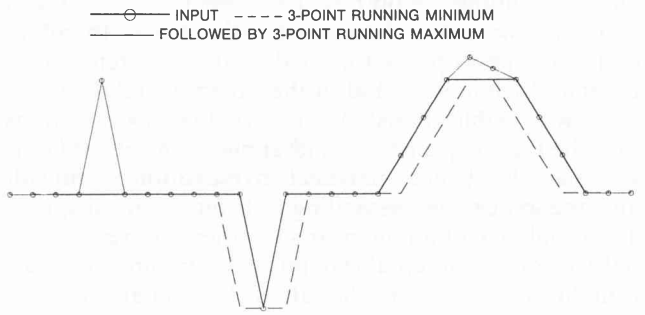

Figure 3 – Smoothing by following a Running Minimum by a Running Maximum

Ln, or L for short, is the concatenation of a minimum operation followed by a maximum operation each with a window size of $n + 1$ points for a total support of $2n + 1$. The points of Lx can conveniently be defined by

$$Lx(i) = Lnx(i) = \max \{\min X(i - n, i), \min X(i + n + 1, i + 1), \ldots, \min X(i, i + n)\}$$

Similarly Un is a maximum operation followed by a minimum operation, conveniently defined by

$$Ux(i) = Unx(i) = \min \{\max X(i - n, i), \max X(i - n + 1, i + 1), \ldots, \max X(i, i + n)\}$$

Note that $x(i)$ is in the centre of the range $x(i - n), \ldots, x(i), \ldots, x(i + n)$ which comprises $X(i - n, i + n)$, namely the set of $2n + 1$ supporting points from which $Lx(i)$ and $Ux(i)$ are selected.

L and U can be shown to be idempotent, i.e. $LL = L$ and $UU = U$. Idempotency is a most desirable property for a nonlinear smoother. If a smoother is not idempotent successive applications of the smoother can give different interpretations to the same original input data.

Median smoothers are not idempotent, $Ln \leq Mn \leq Un$,

i.e. *Ln* and *Un* bound the median smoother of the same support (*Mn*). This means that any upward outliers removed by a running median will also be removed by *L* of the same support and any downward outliers removed by a running median will also be removed by *U* of the same support. *L* and *U* also bound their input ($Lx \leq x \leq Ux$) but Mx and x are not comparable.

Groups of not more than n consecutive upward/downward outliers are removed by *L*/*U*. In designing *L* and *U*, n should therefore be chosen as *at least the maximum number of consecutive expected outliers*. A sequence of more than n "outliers" (in the same direction) is interpreted as a significant pattern in the data. *L* removes upward outliers and retains downward outliers while *U* removes downward outliers and retains upward outliers. This defect is overcome by concatenating the basic unsymmetric smoothers to obtain *UL* and *LU* which both remove upward and downward outliers.

UL and *LU* can also be shown to be idempotent and $LnUn \geq Mn \geq UnLn$. From $Lx \leq x \leq Ux$ it follows that $LU \leq U$ and $UL \geq L$ and therefore $Un \geq LnUn \geq Mn \geq UnLn \geq Ln$, i.e. *LnUn* and *UnLn* are narrower bounds on *Mn* than *Ln* and *Un*. (Note that the support of *UnLn* and *LnUn* is $4n + 1$ while *Ln*, *Un* and *Mn* have a support of $2n + 1$.)

LU and *UL* both satisfy the design criterion and their lack of symmetry should be seen as natural, indicating an interval of fundamental uncertainty associated with the recognition of impulsive noise [2].

Implementation of the Smoothers ***UL*** *and* ***LU***

Consider the construction of the smoother *UL*. *UnLn* is the concatenation of four rank-order selectors; a running minimum followed by a running maximum followed by a second running maximum followed by a final running minimum, each with a window size of $n + 1$ points. It is clear that two successive running maximum operations each with a window size of $n + 1$ points are equivalent to a single running maximum with a window size of $2n + 1$. Therefore:

$$\begin{aligned} ULx(i) &= \min Z(i, i + n) \\ &= \min \{z(i), \ldots, z(i + n)\}, \text{ where} \\ z(i) &= \max Y(i - n, i + n) \\ &= \max \{y(i - n), \ldots, y(i), \ldots, y(i + n)\}, \text{ with} \\ y(i) &= \min X(i - n, i) \\ &= \min \{x(i - n), \ldots, x(i)\} \end{aligned}$$

(The placement of the individual windows for the min-/max operations is not important, all that matters is that $ULx(i)$ is supported by the $4n + 1$ points $X(i - 2n, i + 2n)$. This requirement is clearly met by the above representation because the sums of the offsets from i of the lower and upper limits of the *Z*, *Y* and *X* ranges respectively are $0 - n - n = -2n$ and $n + n + 0 = +2n$.)

UL can be implemented as a procedure which returns successive points of *ULx* as it receives x point by point, with an obvious time lag of $2n$ points. With the above representation the memory requirement for such a procedure is two buffers of $n + 1$ points each (*X* and *Z*) plus one buffer of $2n + 1$ points (*Y*). In the following formal presentation of the procedure for calculating a single point of *UL* the individual memory locations of the buffers are referred to as $x_0, x_1, \ldots, x_n$; $y_0, y_1, \ldots, y_{2n}$ and $z_0, z_1, \ldots, z_n$. The buffers are used circularly; as the oldest value in a buffer is discarded the latest value entering the buffer is placed in the same memory location and a buffer pointer is updated by an appropriate modulo function in order to keep track of the "front end" of the buffer. Assuming that the necessary initializations of buffers (*X*, *Y*, *Z*) and pointers (i for *X* and *Z*, j for *Y*) have been done before the first execution of the procedure, the procedure for calculating the next point of *UL* (UL_{result}), given the next point of x (x_{in}), with a time lag of $2n$ points between them, is as follows:

Begin *UL*
$k \leftarrow (i + 1) \bmod (n + 1)$; $l \leftarrow (j + 1) \bmod (2n + 1)$;
$x_{min} \leftarrow \min (\{x_{in}, x_0, \ldots, x_n\} \backslash x_k)$;
$y_{max} \leftarrow \max (\{x_{min}, y_0, \ldots, y_{2n}\} \backslash y_l)$;
$UL_{result} \leftarrow \min (\{y_{max}, z_0, \ldots, z_n\} \backslash z_k)$;
$i \leftarrow k$; $j \leftarrow l$; $x_i \leftarrow x_{in}$; $y_j \leftarrow x_{min}$; $z_i \leftarrow y_{max}$;
End *UL*

(Note: Operator "\" is used to signify the exclusion of an element from a set; thus $\{x_{in}, x_0, \ldots, x_n\} \backslash x_k$ means "all elements of the set $\{x_{in}, x_0, \ldots, x_n\}$ but excluding x_k".)

An easy way to initialize the process is to fill all three buffers with zeros (say) and then to successively run the procedure with each of the $2n$ points preceding the start of the input series in order to support the first point as explained earlier (typically all duplicates of the first point). Then $x(1), x(2), \ldots, x(2n + 1)$ are run through the procedure before $ULx(1)$ emerges, etc. After processing the last point of the input series a further $2n$ points are needed in order to obtain the last $2n$ *UL* values (typically duplicates of the last input point are used). Before the first execution of the procedure, buffer points i and j must be initialized to any values in the ranges $0 \leq i \leq n$ and $0 \leq j \leq 2n$ (e.g. $i = j = 0$). (The contents of memory locations k, l, x_{min}, and y_{max} need not be retained between procedure calls.)

If a *vector processor* is available an interesting option is to calculate *LU* and *UL* simultaneously in one procedure by noting that $LU(x) = -UL(-x)$.

Note that the calculation of x_{min}, y_{max} and UL_{result} (which is the minimum of the *Z* buffer) need not necessarily involve a complete scan of the buffers for each execution of the procedure. The previous values of the minima and maxima can be compared to the latest values entering the buffers to determine if complete buffer scans are necessary or whether only the previous and latest values need to be considered for a new maximum or minimum. This saving of computing time can be very significant for large supports (large n). The previous values of x_{min} and y_{max} are available as y_j and z_i respectively. The minimum of the *Z* buffer, z_{min} (say), must now also be retained by the procedure. Before the first execution z_{min} should be initialized to a value not less than the smallest value in the initial *Z* buffer (e.g. initialize $z_{min} = 0$). With the preceding in mind the "faster" version of the procesure is as follows:

Begin *UL*
$k \leftarrow (i + 1) \bmod (n + 1); l \leftarrow (j + 1) \bmod (2n + 1);$
if $x_k > y_j$ **then** $x_{\min} \leftarrow$ **min**(y_j, x_{in}) **else** $x_{\min} \leftarrow \min(\{x_{\text{in}}, x_0, \ldots, x_n\} \backslash x_k)$;
if $y_l < z_i$ **then** $y_{\max} \leftarrow \max(z_i, x_{\min})$ **else**
$y_{\max} \leftarrow \max(x_{\min}, y_0, \ldots, y_{2n}\} \backslash y_l)$;
if $z_k > z_{\min}$ **then** $z_{\min} \leftarrow \min(z_{\min}, y_{\max})$ **else**
$z_{\min} \leftarrow \min(\{y_{\max}, z_0, \ldots, z_n\} \backslash z_k)$;
$UL_{\text{result}} \leftarrow z_{\min}$;
$i \leftarrow k; j \leftarrow l; x_i \leftarrow x_{\text{in}}; y_j \leftarrow x_{\min}; z_i \leftarrow y_{\max}$
End *UL*

LU can be calculated by a similar procedure but with the max and min operations (and of course the < and > tests) interchanged. A verbatim copy of the procedure can also be used to calculate *LU* by using $LU(x) = -UL(-x)$. If *LU* and *UL* are to be calculated simultaneously (in parallel) by the latter method two copies of the procedure are of course needed in order to maintain the integrity of the buffers, pointers, etc. of each of the two concurrent processes.

Smoothing of the Experimental Data

In the case of our experimentally sampled data, outliers were replaced by the average of *UnLn* and *LnUn*. The width of the impulsively corrupted segments dictated an *n* of about 1 000 which, however, far exceeded the period of relatively small oscillations occurring in the sequences of uncorrupted data (see Figure 4). The removal of wide impulses and the retention of oscillations with small periods are conflicting requirements. *UL* and *LU* (and M or, for that matter, any rank-based selector) are clearly only trend preserving as far as trends of relatively long duration are concerned. Under the assumption that the amplitude of impulsive noise far exceeded the amplitude of the small oscillations in the uncorrupted data (as seems reasonable from inspection of Figures 1 and 4) it was possible to differentiate directly between outliers and valid data. If $ULx(i) - \delta < x(i) < LUx(i) + \delta$ where δ is an appropriate tolerance, data point $x(i)$ was considered valid otherwise $x(i)$ was replaced by $(ULx(i) + LUx(i))/2$. Figures 4 and 5 depict the results.

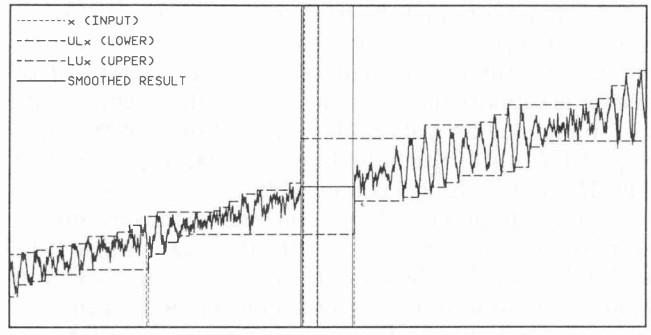

Figure 4 – Segment of Corrupted Data and Decontaminated Result (10 000 points)

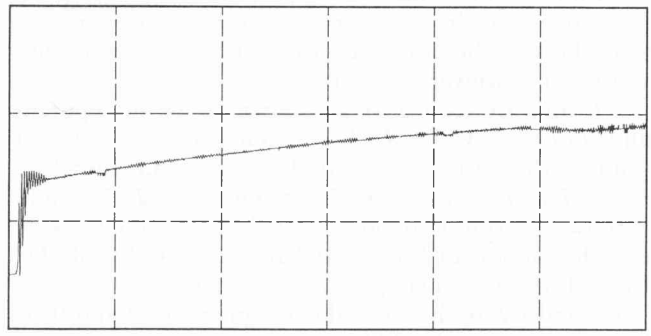

Figure 5 – Decontaminated Experimental Data. (Compare to Figure 1)

Conclusion

The pair of unsymmetric smoothers *LU* and *UL* has been found to be useful in the removal of impulsive noise contained in important experimental data. The example illustrates the ease of implementation and the flexibility in innovative adaptation.

References

1. R. W. Hamming, *Digital Filters*. Prentice-Hall, Englewood Cliffs, N.Y., 1977.
2. C. H. Rohwer, *Idempotent One-Sided Approximation of Median Smoothers, Journal of Approximation Theory*, Vol. 58, No. 2 (1989).
3. J. W. Tukey, *Exploratory Data Analysis*, Addison-Wesley, Reading, MA, 1977.
4. C. L. Mallows, *Some theory of nonlinear smoothers, Ann. Statist.* 8, No. 4 (1980), 695-715.
5. N. C. Callagher, *A theoretical analysis of the properties of median filters, IEEE Trans. Acoust. Speech Signal Process.* ASSP-29, No. 6 (1981).