By Ron Pearson, Swiss Federal Institute of Technology

THE IDEA MAY BE COUNTERINTUITIVE, BUT FOR A LARGE CLASS OF FILTERING TASKS, NONLINEAR FILTERS CAN PRESERVE SIGNAL FIDELITY BETTER THAN THEIR LINEAR COUSINS.

Scrub data with scale-invariant nonlinear digital filters

Simple linear digital filters are useful in many applications. Linearity is a significant practical advantage that you can exploit to develop useful classifications (such as lowpass, highpass, or bandpass filters), rules of thumb (such as "Highpass filters require both positive and negative filter coefficients"), and design procedures (see, for example, **Reference 1**). Unfortunately, some applications require nonlinear filters. Such filters are capable of implementing a vastly wider range of qualitative behavior, but they are harder to design and analyze than linear filters. One reason for this difficulty is that the range of possible nonlinear-filter types is enormous and extremely heterogeneous: Two classes of nonlinear filters may be as dissimilar to each other as they are to the class of linear filters. In fact, as others have repeatedly observed, the difficulty is partly inherent in the term "nonlinear," which characterizes an enormous class of systems by their lack of a unique property. The situation is analogous to considering "nonkangaroo zoology." The class of nonkangaroos includes both all species of octopus and all species of beetle, two subsets that seem as dissimilar to each other as they are to kangaroos.

A useful class of nonlinear digital filters exhibits the property of scale invariance. If {x(k)} and {y(k)} represent the filter's input and output sequences, respectively, the output of a scale-invariant filter in response to the input {Ax(k)+b} is simply {Ay(k)+b} for any positive constant A and any real number b. Note that all linear filters with unity dc gain satisfy this condition, but most nonlinear filters do not. However, a number of useful nonlinear digital filters are scale-invariant. The practical significance of this behavioral restriction is that the response of a scale-invariant filter depends only on the shape of the input sequence and not the units in which the original data sequence is expressed. As a specific example, if you apply any scale-invariant filter to a sequence of temperature measurements expressed in Celsius units, and then convert the results to Fahrenheit or Kelvin, the filter will give the corresponding results, expressed in those units. In contrast, more general nonlinear filters (filters based on polynomials, artificial neural networks, or piecewise linear functions) can respond differently to the same temperature-data sequence expressed in different units.

CLEANING DATA WITH A NONLINEAR FILTER

Linear-system characteristics that are profound advantages in one application can be equally profound disadvantages in another application. For example, one fundamental property of linear systems is that they preserve the shape and frequency of sinusoids and can alter only their magnitude and phase. This fact is the basis for frequency-response characterizations used extensively in the design of linear filters that separate signals according to their frequency content. Conversely, this same property means that linear circuits are inadequate for tasks such as modulation or demodulation, in which you wish to move these components from one frequency range to another.

Data cleaning—automatically removing outliers (anomalous data points) and replacing them with values that are more representative—is another case that requires nonlinear filters. **Figure 1**, which depicts a sequence of 200 physical-property measurements obtained from an industrial manufacturing process, illustrates this problem. Open circles designate most of these points, but the Hampel filter described below detects eight outliers—the points marked with solid circles. Three of these points (k=42, 74, and 80), lying above the main collection of data, do not appear to be unusual. They illustrate that even the best outlier-detection procedures can behave somewhat unpredictably, finding either more or fewer outliers

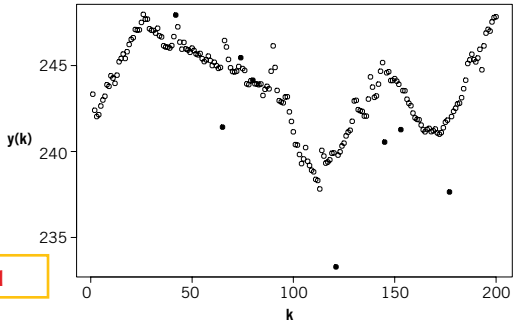

Figure 1

A data sequence that an industrial process measurement generates includes eight outliers, designated by filled circles.

in a data set than your eyes or other manual analyses might. Still, the five points at k=65, 121, 145, 153, and 177, which lie below the main collection of points, appear anomalous with respect to the variation in the rest of the data sequence. The effect of these spikes on subsequent data processing and analysis can be substantial: Two estimated power spectra, each computed using exactly the same method, differ at high frequencies by about 10 dB, due entirely to the treatment of these eight data points (**Figure 2**). The upper curve corresponds to the original, unmodified data sequence's power spectrum, and the lower curve corresponds to the spectrum of the cleaned data sequence using the Hampel filter. These two data sequences (original and clean) are identical except for these eight data points, which the Hampel filter replaces with more consistent values that it calculates from their neighbors.

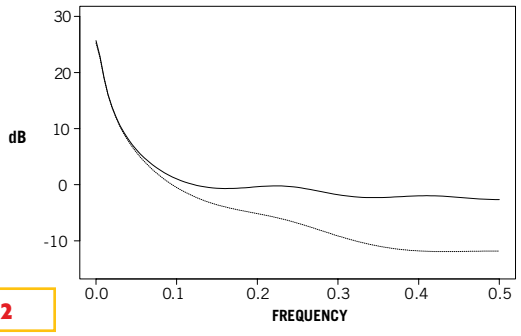

Figure 2

The power spectra estimated from the raw (top) and cleaned (bottom) data sequences differ by about 10 dB at high frequency.

I estimated these spectra by first fitting a linear prediction model to the data sequence and then computing the power spectrum from the coefficients of the prediction model. **Reference 2** details this approach to spectrum estimation. The method computes the spectrum from these estimated model parameters; examining them for this example to see how much the eight outlying points influence these results is instructive (**Table 1**).

How you treat anomalous data points, such as those in **Figure 1**, can profoundly influence the results you obtain when using the data in subsequent processing and analysis. You can achieve similar results using other spectrum-estimation procedures. Isolated outliers, such as the ones here, frequently raise the power spectrum's noise floor, possibly obscuring important high-frequency details (**references 3**, **4**, and **5**).

WHY LINEAR FILTERS ARE INADEQUATE

Linear filters succeed in a variety of noise-reduction applications, so it is reasonable to ask why they are inadequate in data-cleaning applications such as the one just described. The filters considered here, both linear and nonlinear, belong to the nonrecursive, FIR, or moving-window class, defined by

$$y(k)=F[x(k-m),(k-m-1),\ldots,(k-n)].$$

Here, {x(k)} and {y(k)} represent, respectively, the filter's input and output sequences, and F[] can be any function that maps the n−m+1 input samples from x(k−m) through x(k−n) into a real number. You can assume that m is larger than or equal to n or redefine the function F[] to make it so. If m>n>0, this equation defines a filter that is "strictly causal," meaning that the current filter response y(k) depends only on past filter inputs x(j), where j is less than k. This restriction is important for filters that must operate in real time, but it is unnecessary for filters you use in offline processing applications, such as image enhancement or data analysis. Alternatively, note that you may convert any noncausal filter, for which n<0, into a causal filter by delaying the filter output |n|+1 or more steps. This delay can have undesirable effects in some applications, such as feedback control, but in many applications, such a delay is acceptable; the filters considered here are easier to analyze in the noncausal case. In particular, it is convenient to consider filters that are based on a symmetric data window, for which n=−K and m=K for some positive integer K, giving a filter of the form

$$y(k)=F[x(k-K),\ldots,(k),\ldots,(k+K)].$$

Delaying this filter's output by K+1 time steps gives you a strictly causal filter:

$$z(k)=y(k-K-1)=$$
$$F[x(k-2K-1),\ldots,(k-K-1),\ldots,(k-1)].$$

If you restrict F[] to the form

$$y(k)=a(-K)x(k-K)+\ldots+ a(0)x(k)+\ldots+a(K)x(k+K), \quad (1)$$

you obtain a linear nonrecursive filter.

These filters satisfy the "principle of superposition," meaning that if {u(k)} and {v(k)} are input sequences that generate responses {y(k)} and {z(k)}, respectively, the filter's response to the input {au(k)+bv(k)} is simply {ay(k)+bz(k)}, for any real constants a and b. In fact, you can show that essentially the only moving window filters satisfying the principle of superposition are the linear filters that **Equation 1** defines (**Reference 4**).

One useful model for a data sequence contaminated by a few large, isolated spikes, such as the one depicted in **Figure 1**, is the "additive outlier model," which assumes that

$$x(k)=v(k)+o(k).$$

Here, {v(k)} represents the uncontaminated data sequence you would like to have available for subsequent processing, and {o(k)} represents the sequence of outliers whose effects you would like to overcome. The general characteristic of these outliers is that they are zero most of the time, but when they are not zero, they are often quite large. If you apply a linear FIR filter L to the observed sequence {x(k)}, it follows from the principle of superposition that the response is given by

$$L\{x(k)\}=L\{v(k)\}+L\{o(k)\}.$$

For a large, isolated outlier of amplitude A occurring at time k−j, it follows from **Equation 1** that the filter's response at time k will be

$$y(k)=L\{v(k)\}+Aa(j).$$

Thus, to remove the effects of large outliers occurring at arbitrary times, it follows that a(j)=0 for all j, meaning that y(k)=0 for all k, for all input sequences {x(k)}. This filter is insensitive to outliers, but it throws the baby out with the bath water, because its response is never useful. The basic problem is that the data-cleaning filter must, first, reduce or eliminate the effects of outliers, and second, minimize the distortion of the nominal part of the input-data sequence. Linear filters are inherently unsuited to the task of data cleaning because they cannot achieve both of these objectives together, but some nonlinear FIR filters nicely achieve both of these objectives.

IS THE MEDIAN THE MESSAGE?

Taking a(j)=1/(2K+1) in **Equation 1** gives the average—the most common measure of a data sequence's "center." A fundamental difficulty with the average, however, is its great sensitivity to outliers. The median, an alternative measure of the data sequence's center with much less

sensitivity to outliers, provides the basis for the nonlinear scale-invariant filters discussed below.

You can find the median by sorting a data sequence from its most negative to its most positive values. For a collection containing an odd number of data values $\{x(1),x(2),...,x(N)\}$, the median is simply the middle value in the sorted list. A seven-point data sequence serves as an example:

$$\{-1.5,6.8,-1.1,0.3,0.2,1.3,1.1\}\rightarrow$$
$$\{-1.5,-1.1,0.2,0.3,1.1,1.3,6.8\}.$$

The median of this data sequence is the middle (that is, the fourth) element from this sorted list, which equals 0.3. A group containing an even number of points has no middle element, so the median is defined as the average of the two central elements in the sorted list. For example, omitting the largest value (6.8) from this sequence produces a list containing six elements. Sorting this list then gives you

$$\{-1.5,-1.1,0.3,0.2,1.3,1.1\}\rightarrow$$
$$\{-1.5,-1.1,0.2,0.3,1.1,1.3\}.$$

The two middle elements in this sorted list are the third and fourth elements, 0.2 and 0.3; their average is 0.25.

This example illustrates the difference in outlier sensitivities of the mean and the median. In particular, note that the original data sequence is almost in ascending order to begin with, aside from the glaring exception of the point $x(2)=6.8$. The anomalous character of this point is similar to that of the outliers visible in the real data sequence in **Figure 1**. Treating this point as an outlier and simply removing it from the data sequence change the mean from approximately 1.01 for the original seven data points to 0.05 for the six remaining data points. In contrast, the median changes from a value of 0.20 for the original data sequence to 0.25 for the sequence without the largest point.

An important feature the median shares with the mean is scale invariance. The mean is scale-invariant because it is a linear combination of data values, and the coefficients in that linear combination sum to 1. In contrast, the median is scale-invariant because the sorting operation that finds it is also scale-invariant. Suppose that $\{x(k)\}$ is a data sequence, and $\{y(k)\}$ results from sorting $\{x(k)\}$ into ascending order. If you multiply all of the elements of the original data sequence by some positive number A, sorting $\{Ax(k)\}$ into ascending order will give you $\{Ay(k)\}$. This result follows from the fact that the sorting operation makes comparisons that essentially ask the question, "Is x(j) larger than x(k)?" Clearly, if A is a positive number and x(j) is larger than x(k), then Ax(j) is also larger than Ax(k). Similarly, adding any constant to all of the values of the sequence does not change their relative order: If x(j) is larger than x(k), then x(j)+b is larger than x(k)+b for any real number b. The sorting operation is scale-invariant: If x(j) is greater than x(k), then Ax(j)+b is greater than Ax(k)+b for any positive A and any real b. Because you obtain the median either by simply selecting one value from this sorted list or, in the case of a list containing an even number of elements, by averaging two elements from this sorted list, then for Z equal to the median of the data sequence $\{x(k)\}$, the median of the rescaled data sequence $\{Ax(k)+b\}$ is AZ+b.

Many sorting algorithms exist, and inefficient ones can take a very long time to execute. Fortunately, because sorting is such an important operation in many computations, you can readily find efficient sorting algorithms. One of the most widely used sorting algorithms is the Quicksort, which you can find in the C++ Standard Template Library (**Reference 6**).

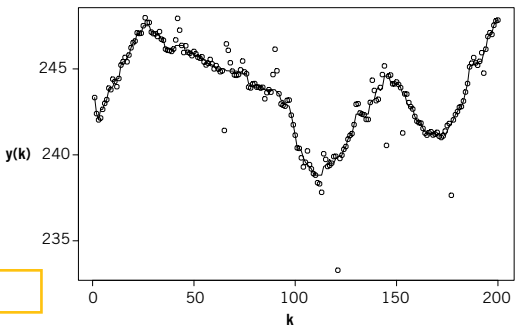

Figure 3

The seven-point median filter eliminates the outliers in the data sample but also introduces significant changes at other data points.

THE MEDIAN FILTER

If you set

$$a(j)=\frac{1}{2K+1}$$

for all j in **Equation 1**, you obtain an unweighted-average moving-window filter: The filter's response at sample time k is simply the average of the current value, the past K values, and the next K values. If you replace this average with the median, you obtain the median filter:

$$y(k)=\text{median}\{x(k-K),\ldots,x(k),\ldots,x(k+K)\}. \quad (2)$$

Note that because this moving data window contains an odd number of data points for any choice of K, the median filter's output is simply the middle element in the list that you obtain by sorting the values in this data window from most negative to most positive. Because the median is scale-invariant, it follows that the median filter is also scale-invariant. On the other hand, the following example shows that the median is not a linear operation on the data, so the resulting scale-invariant filter is a nonlinear one. Specifically, if the median were a linear operation, then the median of the sum of two sequences $\{u(k)\}$ and $\{v(k)\}$ would simply be the sum of the individual medians. To see that this scenario is not the case, consider the following two sequences:

$$\{u(k)\}=\{-3,-2,-1,2,4\},$$

and

$$\{v(k)\}=\{3,2,-1,-2,-2\}.$$

Because $\{u(k)\}$ is an increasing sequence, the median is simply the central element in this list. Increasing sequences require no further sorting, so median filters do not alter them. Similarly, $\{v(k)\}$ is a decreasing sequence, so sorting this sequence simply reverses it. This reversal does not change the central element of the sequence, however, so again, the median is simply the central element of the original data sequence. Consequently, the median filter does not alter decreasing sequences, such as $\{v(k)\}$.

For this example, the medians of

TABLE 1—SPECTRUM-ESTIMATE PARAMETERS

Parameter	Estimate Original	Estimate Cleaned	Change (%)
1	0.53	1.084	105
2	0.192	−0.177	−193
3	0.131	0.047	−65
4	0.101	0.075	−26
5	0.038	−0.053	−241
6	−0.047	−0.005	89

{u(k)} and {v(k)} both equal −1, so the sum of these medians is −2. Therefore, if the median filter were linear, its response to the sum {u(k)+v(k)} would be −2, but this sum yields:

$$\{u(k)+v(k)\}=\{0,0,-2,0,2\}.$$

This sequence is neither increasing nor decreasing, and its median value is the central value in the following sorted list:

$$\{u(k)+v(k)\}\rightarrow\{-2,0,0,0,2\},$$

which is 0. Because the median of the sum does not equal the sum of the medians in this example, it follows that the median filter is nonlinear.

Applying the median filter to the industrial-process data set in **Figure 1** removes the outliers from the data set (**Figure 3**). But this filter, which you obtain by setting K=3 in **Equation 2**, introduces significant changes in the rest of the data sequence, causing distortion that is highly undesirable in many applications (**Reference 5**).

An important feature of the median filter and many other nonlinear filters is the existence of "root signals," which are sequences that the filter does not modify. Both increasing and decreasing sequences represent root signals of the median filter. This observation holds for all choices of the window-width parameter K. Other root signals also exist for the median filter but generally depend on K (**Reference 7**). A desirable nonlinear filter for applications such as data cleaning is one whose root signals include all of the nominal data sequences of interest, but exclude the spikes and other undesirable signal features you wish to eliminate. In general, the median filter tends to be too aggressive as a data-cleaning filter; it tends to introduce undesirable distortions in the nominal part of the data sequence. This observation means that the class of root signals for the median filter is often too small. One way to enlarge this class is by replacing the median filter with the Hampel filter.

SETTING DOWN ROOTS

The Hampel filter belongs to the general class of "decision-based filters" (**Reference 7**). These filters look for anomalous data points and replace them with more representative values. The Hampel filter uses a moving data window that you define as discussed for the median filter, including the points x(k−K) through x(k+K). The filter uses these 2K+1 data values to decide the "representativeness" of the current data sample x(k): If the filter deems x(k) representative, it makes no change, and the filter's output is simply y(k)=x(k). Otherwise, if the filter decides that the current data sample is unrepresentative, it computes a more representative replacement from the available data. For the Hampel filter, this representative value is the median. The remaining task is to establish the criterion for deciding whether x(k) is representative.

To decide whether a given data point is an outlier, the filter must have some quantitative characterization of "nominal behavior." One of the most popular working assumptions concerning nominal behavior is the Gaussian, or normal-random-variable, model. A detailed discussion of this idea is beyond the scope of this article, but two points are important. First, this assumption is popular and is often reasonable as an initial approximation in describing nominal behavior. Second, if you adopt this approximation for some data sequence {x(k)}, the mean,

$$M=\frac{x(1)+x(2)+\ldots+x(N)}{N},$$

and the standard deviation,

$$\sigma=\sqrt{\frac{[x(1)-M]^2+\ldots+[x(N)-M]^2}{N-1}},$$

provide a complete characterization of the statistical behavior of {x(k)}. From these assumptions, the probability of observing a value x(k) lying further than 3σ from the mean M should be only about 0.3%. These observations lead to the following simple outlier-detection strategy, called the "3σ edit rule": After computing the mean M and standard deviation σ, if |x(k)−M|<3σ, declare the point x(k) nominal; otherwise, declare x(k) an outlier and take appropriate action.

Unfortunately, this procedure often fails because outliers in the data sequence tend to shift the mean and increase the estimated standard deviation. Often, these outlier-induced distortions are severe enough to cause the 3σ edit rule to completely break down. As an example, if you apply this procedure to a seven-point moving data window for the industrial data sequence in **Figure 1**, it fails to detect any outliers.

Recalling that the median is much less sensitive to outliers than the mean, the Hampel filter is a decision-based filter that replaces the mean with the median. It replaces the standard deviation with another measure of the "scale" or "spread" of the data sequence that is based on the median and that is much less sensitive to outliers than is the standard deviation. Specifically, let Z denote the median of the data sequence {x(k)}. The expression

$$\{d(k)\}=\{|x(1)-Z|,|x(2)-Z|,\ldots,|x(N)-Z|\},\quad(3)$$

gives a measure of how far each point lies from this reference value.

Therefore, you may view the median of this list of distances as a measure of how far x(k) typically lies from the median value Z. Scaling this value by a factor of 1.4826 results in the MAD (median-absolute-deviation) scale estimate, whose value is equal, on average, to the standard deviation for Gaussian data sequences. The MAD scale estimate is the alternative to the standard deviation σ that makes the 3σ edit rule effective against outliers. Statistician FR Hampel proposed the MAD scale estimate as an alternative to σ, so this modification of the 3σ edit rule is sometimes called the "Hampel identifier."

After computing the median Z of the data sequence {x(k)}, compute the sequence {d(k)} defined in **Equation 3**. From this sequence, compute the MAD scale estimate, which is equal to 1.4826 median {d(k)}. For each data point x(k) in the sequence, if |x(k)−Z|<3MAD, classify x(k) as nominal; otherwise, classify x(k) as an outlier.

The Hampel filter uses a moving-window implementation of this outlier-detection procedure. For each sample k, the filter forms a collection of samples x(k−K) through x(k+K) from which it computes the median and MAD scale estimate and performs the outlier test. If the Hampel filter declares x(k) to be a nominal data point, its output is y(k)=x(k), but if the filter declares x(k) to be an outlier, its output is the median, y(k)=Z.

A Hampel filter with K=3 produced the results in **Figure 1** for the industrial data sequence. Comparing **figures 1** and **3**, it is clear that the Hampel filter introduces far less distortion in the nominal part of the data sequence than does the median filter. In particular, the median filter modifies 123 of the 200 points in this data sequence, and the Hampel filter based on the same data window modifies only the eight points.

It is useful to generalize the definition of the Hampel filter given here, both to obtain a more flexible filter and to illustrate the relationship between the Hampel filter and the median filter. Specifically, note that the numerical value 3 in the 3σ edit rule comes from an assumption of a Gaussian distribution for the nominal data and defines the degree of aggressiveness in detecting and rejecting outliers. Increasing this threshold parameter from 3 to, say, 4 or 5 would result in a more conservative outlier-detection strategy that is less likely to reject data points; decreasing it from 3 to 1 or 2 would give a much more aggressive data-cleaning strategy. The general form of the Hampel filter makes this value a variable tuning parameter, t.

To apply any of the moving-window data filters, you need to handle the end effects. That is, the moving data window on which these filters are based consists of the points x(k−K) through x(k+K) for some positive integer K. As a consequence, this data window is undefined for values of k smaller than K+1 or larger than N−K for a data sequence indexed from 1 to N. The most popular way to overcome this problem is by artificially extending the data sequence, defining the values x(−K+1) through x(0) as equal to x(1) and x(N+1) through x(N+K) as equal to x(N). This way, you may construct the moving data window for all values of k from 1 to N, although, in unfortunate cases, the resulting filter may behave strangely at the ends of the data record.

Combining these ideas leads to the following general implementation of the Hampel filter:

Extend the data sequence from length N to length N+2K:

$$\{x(1),x(2),\ldots,x(N)\}\rightarrow$$
$$\{x(1),\ldots,x(1),x(2),\ldots,x(N),x(N),\ldots,x(N)\}.$$

For (k=1,2,...,N)

{

Construct the moving data window

$$\{w(-K),\ldots,w(0),\ldots,w(K)\}=$$
$$\{w(k-K),\ldots,w(k),\ldots,w(k+K)\}$$

Compute

$$Z=\text{median}\{w(k)\}$$

Compute

$$Q=\text{MAD}\{w(k)\}=$$
$$1.4826\cdot\text{median}\{|w(k)-Z|\}$$

Compute

$$D=|w(0)-Z|$$

if D<tQ then y(k)=w(0)

else y(k)=Z

}

You can adjust the characteristics of this filter with two tuning parameters, the window width K and the threshold t. Both of these parameters have a significant effect on the filter's performance. The window-width parameter K influences both the general consistency of the filter's behavior and its ability to handle outliers that occur in "patches" or "clusters," such as those that a persistent external source lasting longer than the time between successive data samples causes. As a general rule, it is undesirable to make K too large. Doing so increases the length of the "end segments," which the filter sometimes doesn't deal with very well. Making K too large also restricts the filter's ability to follow systematic changes in the nominal part of the data sequence while still rejecting outliers that violate this trend. Conversely, it is also important not to make K too small, both to avoid difficulties with outlier patches and because the filter's behavior becomes less consistent as K decreases. Experience suggests that K values of 3 to 5 (implying seven- to 11-point moving data windows) often give reasonable results, but you should regard this suggestion only as a rough rule of thumb, as performance can be highly application-specific. It is therefore important to try this filter on some representative test cases.

The threshold parameter t directly determines the filter's aggressiveness. Increasing t reduces the likelihood of declaring x(k) an outlier and replacing it with the median value Z. Note that if the point x(k) is unmodified by the Hampel filter for some fixed value of t, a Hampel filter with the same window-width parameter K and any larger value of t cannot modify it. Conversely, if you choose t=0, the Hampel filter remains well-defined, but the input data never satisfies the selection criterion D<tQ, so the filter's output is y(k)=Z for all k. Because Z is the median value from the data window, the Hampel filter reduces to the median filter when t=0.

The Hampel filter belongs to the class of scale-invariant nonlinear filters, so if you rescale the original data sequence {x(k)} to {A x(k)+b}, the quantities appearing in the Hampel filter pseudocode change as follows:

$$w(k)\rightarrow Aw(k)+b$$
$$Z\rightarrow AZ+b$$
$$|w(k)-Z|\rightarrow A|w(k)-Z|,$$

so

$$Q\rightarrow AQ$$
$$D\rightarrow AD.$$

As a consequence, the test condition D<tQ is rescaled to AD<tAQ, which is exactly equivalent to the original for any positive number A. Therefore, rescaling the data sequence {x(k)} to {Ax(k)+b} does not alter the decision criterion for the Hampel filter at all. The filter now chooses between the rescaled but otherwise unmodified data value Ax(k)+b, when it deems the point nominal, and the rescaled median value AZ+b, when it deems the point an outlier.□

References

1. Hamming, RW, "Digital Filters," 2nd edition, Prentice-Hall, 1983.
2. Kay, SM, "Modern Spectrum Estimation: Theory and Applications," Prentice-Hall, 1996.
3. Martin, RD, and DJ Thomson, "Robust-resistant spectrum estimation," Proceedings of the IEEE, vol 70, 1982, pg 1097 to 1114.
4. Pearson, RK, "Discrete-Time Dynamic Models," Oxford, 1999.
5. Pearson, RK, "Data Cleaning for Dynamic Modeling and Control," Proceedings of the European Control Conference, 1999, Karlsruhe, Germany.
6. Sedgewick, R, "Algorithms in C++," 3rd edition, Addison-Wesley, 1998.
7. Astola, J, and P Kuosmanen, "Fundamentals of Nonlinear Digital Filtering," CRC Press, 1997.

Author's biography

At the time of this writing, Ron Pearson was a senior member of the research staff for the Automatic Control Laboratory at the Swiss Federal Institute of Technology, where he was involved with nonlinear dynamic-model identification, data analysis, nonlinear digital-signal processing, and nonideal sensor modeling. He currently works for the Tampere International Center for Signal Processing at the Tampere University of Technology (Finland). He holds a BS in physics from the University of Arkansas (Monticello) and an MSEE and PhD from the Massachusetts Institute of Technology (Cambridge). You can reach him at pearson@cs.tut.fi.