

DATA COMPRESSION FOR PROCESS HISTORIANS

Copyright © 1995 by Peter A. James,
Chevron Research and Technology Company,
Richmond, CA 94802-0627, paja@chevron.com

Historization

This paper deals with storing and retrieving the time history of plant sensor data. Historization is an important part of modern plant monitoring and information systems. Operators look at real-time multivariable trends for clues to "what's going on", post-mortem committees need to reconstruct the circumstances leading up to plant "incidents", and engineering studies benefit from a way-back look at a not-so-recent mode of operation.

Compression

Why compress, why not just store all data? The reasons are pretty obvious:

- to conserve storage space
- to live within limited channel bandwidth - this could be the input-output channel to disk or the message channel from a data collecting/filtering process to a history storage process.

Efficient utilization of storage space is important even with today's huge storage volumes. Operations and Engineering want months of data online in addition to last shift's. Compression is not the only means to fulfill this need: storage of "averaged" data eg. hourly, shift, daily, weekly averages may fulfill the really long term data storage requirement. Nonetheless, for some "short" time scale most clients want "raw" data or some really close approximation thereto. And "short" may mean months or longer to some clients.

Data compression is not effective at dealing with limited I/O bandwidth at all times. This is because there will be intervals during which little or no compression will take place: eg. computer or instrumentation subsystem startup, plant upsets, neurotic sensors, etc. Compression can only help insofar as memory buffers are able to provide a temporary repository when the I/O channels are "maxed out".

Compression is not the only means to deal with limited I/O bandwidth. Multiple channels can be provided. This especially convenient along the lines of datatype (eg. analogs versus discreties) since the different data types are usually dealt with separately anyway.

Who's the Client?

The client is an operator, an engineer, a plant supervisor, an accountant, a government agency -- in short, anybody or any body that may want a peek at the historical record. Choosing a compression method is as much a matter of clientele proclivity as mathematical proof.

The Customer is Always Right

Our clients ask three questions about each compression method:

- how good a representation of the original data will I get?
- what will it look like (when reconstructed from the historical record)? Some reconstructed data looks digitized. Other plots "sharper" than the original data, or has suspicious "glitches" in it. These subjective assessments may, in the end, carry far more weight than mathematical or statistical reassurances.
- How efficient is it in terms of reducing the amount of storage I have to buy?

Details vary from client to client, but two views cover the gamut. Which one of the following views do your clients espouse?

View # 1 - "My only requirement is that any reconstructed point be within $\pm\text{max_dev}$ of the actual value (as originally presented to the historization process)." This requirement will hereafter be referred to as *the iron-clad guarantee*. Note that the iron-clad guarantee eliminates store-every-nth point, timeslicing (periodic sampling), and averaging methods.

View # 2 - "I want two criteria to be met. First, any reconstructed point be within $\pm\text{max_dev}$ of the actual value (the iron-clad guarantee). *Second, any point stored in history be exactly the value as received from the scanner.*"

If you subscribe to view #1 then you can use any of a wide variety of compression methods including some very efficient ones. View #2 does not lead to quite the same economies.

Time and Time Again

Bear in mind that a data point is a time-value pair. The compression algorithm may have permission to alter a value (but by no more than $\pm\text{max_dev}$). But the scanning subsystem indelibly time-stamps a value and this time-stamp must pass to the historical record unscathed---if it makes it into the record at all. Probably some economies could be achieved by relaxing this requirement. In one view the compression process should have complete freedom to invent timestamps. For instance, an historical record that represents

the data by a spline fit does not explicitly contain any time-value pairs at all.

STATUS and other qualifiers/modifiers

Operational requirements may dictate a maximum age (`max_age`) as well as a maximum deviation. Thus, a new data point may trigger a store simply because it is more than one hour older than the last stored point. This is both a credibility issue (eg. a point that is scanned once a minute but hasn't been stored in 2.5 months because it hasn't changed enough...nevertheless raises eyebrows) and a backup to whatever other ways the programmer designer provides for failure-tolerance.

The compression methods we use accommodate maximum age and status gracefully, with an unavoidable toll on efficiency. While they add to other interesting problems (such as startup, normal shutdown, and sudden shutdown) for the system designer and programmer, they are not dealt with further in this paper.

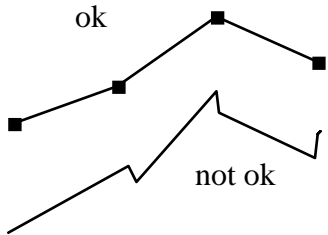
Compression Methods

Space telemetry needs motivated researchers to devise and analyze data compression schemes in the 1960's and 70's (1), and the methods developed were screened with real telemetry data (2, 3, 4). In the early eighties, Hale and Sellars (5) promoted the "boxcar" and "backslope" compression algorithms (predictors), which have since seen some service in process monitoring historians (6, 7). New schemes are described from time to time (e.g. 8). Recently "Swinging Door" algorithm(s) were introduced and to some extent popularized (9, 10, 11). Today, much data compression research is focused on image compression---e.g. by wavelet-based spectral transformations (12). Possible application of the latter to process historians has not been considered by this author and is not covered by this paper.

The methods we have found particularly useful are the "fan interpolators" (2), which we nicknamed SLIM (Straight-Line-Interpolative-Methods). A definition of these and other methods is contained in the Appendix.

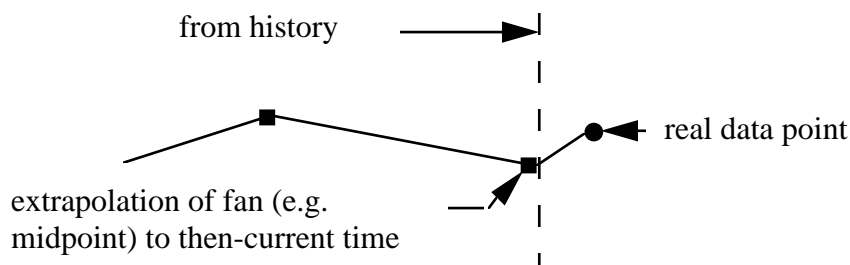
Table I lists the properties of several compression methods. Table II gives some indication, at least, of relative compression efficiencies in terms of compression ratio, which is the # points received / # points stored. Depending on `max_dev`, the compression ratio for any particular set of data can vary between 1.0 and infinity.

In choosing a compression method, we need to consider how we intend to recreate the historical record (retrieval). This will always be by interpolation, or something like interpolation, because of trending:

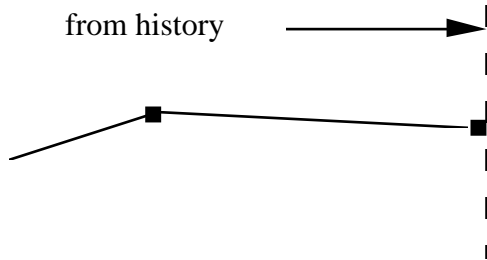


The downside to choosing joined line segments is that it may exacerbate oscillating situations (ref. 3, p. 84).

Let's create a strip chart, starting from historical data, moving the "curve" to the left each time a new point is stored and a real "uncompressed" point is plotted.



If we then replot the data, now entirely from the historical record, we may get:



which shows that retrieval is like scuba-diving; you have to be careful on decompression or you'll get the bends!

It is probably not a good idea to recreate the unclosed reconstruction epoch each time the chart is bumped to the left by a new data point, because this would result in noticeable vertical wobble of the existing trace (as the slope of the best-fit line changes), and would require more coordination between storage and retrieval.

Retrieval versus Storage

A curious dichotomy possesses many compression methods. When the compression process receives a new point it compares it to one or more extrapolators. If the extrapolator does not predict the value to within $\pm\text{max_dev}$, then some action is taken, e.g. the extrapolator(s) is/are terminated at the prior point's timestamp. The line or

function may still be an extrapolator, however, because its value there may be an extrapolation of previous values, not the value as received.

The decompression process, however, is purely one of interpolation. Why? Because it looks better on a trend.

Does the disparity between compression by extrapolation and decompression by interpolation pose a problem? Yes, if it leads to a breakdown of our iron-clad guarantee.

For value and for modified boxcar compression, extrapolation results in a staircase appearance, which is unacceptable for trending. At least the staircase meets our iron-clad guarantee for reconstructed data. Straight line interpolation may not. In order to use straight line interpolation to reconstruct data, we may have to halve the value of `max_dev` used during compression. This applies to Modified Boxcar, Modified Backslope, and SLIM2 (see the Appendix), and unavoidably reduces the compression ratio. But happily, if we always reconstruct by interpolating, we no longer need to remember which of the methods was used to compress the data.

More Gotchas

Retrieved data carries more meaning if you're aware of what value for `max_dev` was used. Indeed `max_dev` probably doesn't change very often, if ever. Nevertheless, provision must be made for recovering that number with certainty.

The instrumentation system may itself impose limits on a value (clamping). It is possible for reconstructed points to lie outside the clamped range by as much as `max_dev`. In this way, a trend of absolute pressure might dip below zero, whereas the "live" engineering values were originally non-negative. SLIM1-compressed data suffers from this anomaly.

The compression ratio experienced depends on the method chosen, on the nature of (e.g. variance in) the data, and on `max_dev`. It can also be highly serendipitous. Take for example a constant process subjected to some sort of sinusoidal disturbance and sampled frequently:

$$y = C + \sin(t)$$

and

$$\text{max_dev} = 1.0$$

We discover that the compression ratio is large indeed, limited only by `max_age`, provided the very first point falls on $y = C$, as will happen if we start at $t = 0$. Otherwise, the compression ratio could be as low as one.

Finally, we note that send-on-change instrumentation systems may introduce their own error which is not covered by our iron-clad guarantee unless specifically accommodated by reducing the value of `max_dev`. This could be important where substantiating environmental compliance is an issue.

And the Winner is....

It comes down to choosing between SLIM3, with its faithful reproduction of original data points, and SLIM1 with its higher compression ratio. Both methods guarantee that any reconstructed points will be within $\pm \text{max_dev}$ of the original value. When a sensor is likely to yield measurements very close to an implausible region (e.g. subzero absolute pressure), use SLIM3. Otherwise enjoy the maximum compression SLIM1 offers. But check with the historian's owner first!

Appendix--- Compression Algorithms

Pseudocode is used to describe the first few methods.

1. Value Compression

```

IF the value of the current point differs from
  the last recorded point by more than max_dev
THEN {
  IF previous point is later than last recorded point
  THEN {
    record projected point =
      (previous point's time, last recorded point's value)
    record current point
  }
  ELSE record current point}

```

Although this is basically an extrapolative method, an interpolative reconstructor can be used transparently.

2. Hale and Sellars (4) Box Car

```

IF the value of the current point differs from
  the last recorded point by more than max_dev
THEN {
  IF previous point is later than last recorded point
  THEN record previous point
  ELSE record current point}

```

3. Hale and Sellars (4) Backward Slope

Project the last two recorded (time, value) points to the current time.

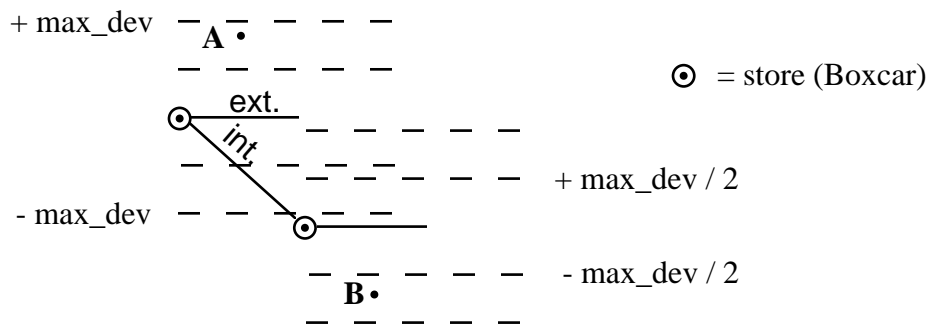
```

IF the value of the current point differs from
  the projected point by more than max_dev
THEN {
  IF previous point is later than last recorded point
  THEN record previous point
  ELSE record current point}

```

4. Modified Boxcar and Modified Backslope

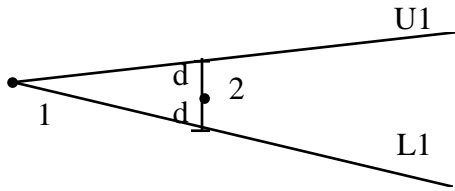
Our ironclad guarantee forces us to modify the boxcar and backslope procedures, which record only the previous point when a current point turns up out of tolerance with the zero or first order projection. We now must also store the current point if its value is beyond \pm one-half of max_dev from the projection through the new last recorded point (= previous point). Point B, below, is an example.



Since these are extrapolative methods, we really ought to use an extrapolator for reconstructing data. If an interpolator is to be used (as recommended), our iron-clad guarantee requires us to set the tolerance bands to \pm one-half of max_dev . Point A, above, illustrates why.

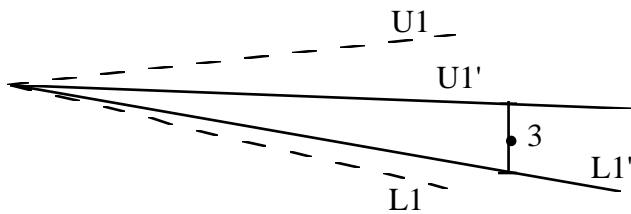
5. SLIM1

We'll describe this method by working through an example.



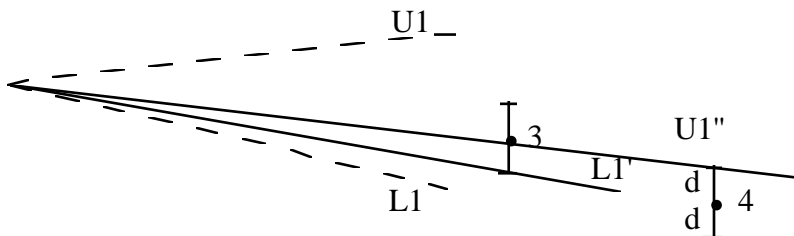
Given points 1 and 2, i.e. (t_1, y_1) and (t_2, y_2) , we construct a pair of lines of slope L_1 and U_1 as in the above sketch (d is max_dev and defines upper and lower limits or "tolerance band").

Now point 3 comes along.



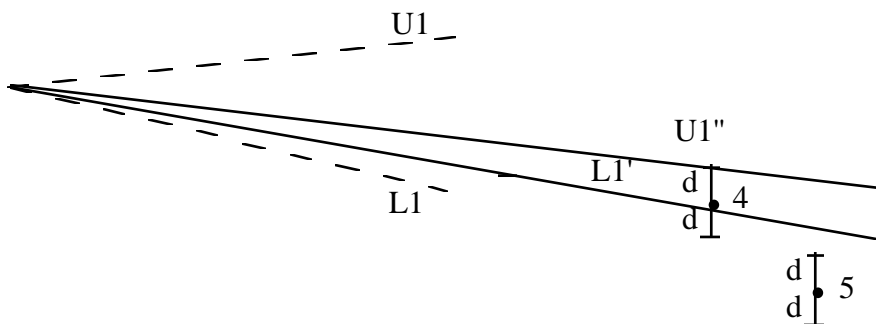
We change slope L_1 to L_1' and U_1 to U_1' to subtend y_3 's tolerance band. Notice that we close the fan (reduce slope U_1 and/or increase slope L_1); we never open it (else we might go outside 2's tolerance band).

Now point 4 arrives.



This time we change only slope U_1' to U_1'' .

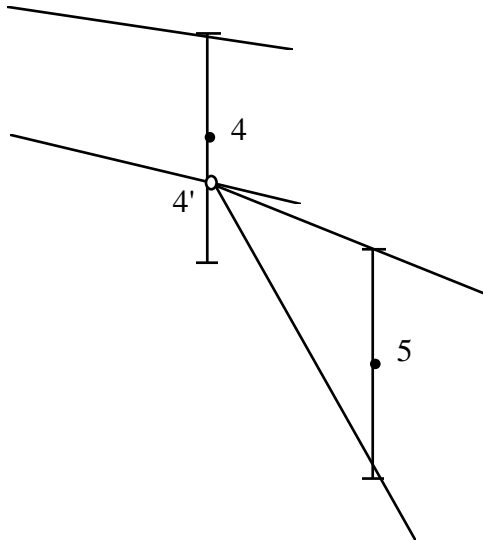
When point 5 arrives,



we discover that this time we are out of luck. 5's tolerance band does not overlap the region subtended by the two lines at all. So we go back to t_4 and record pseudopoint (t_4, y_4') which is at the intersection of 4's tolerance band and the line of slope L_1' . (Had

point 5 lay above line U1, we would have chosen to record the value y_4+d instead.)

Now we start anew, starting with a pair of lines emanating from (t_4, y_4') :



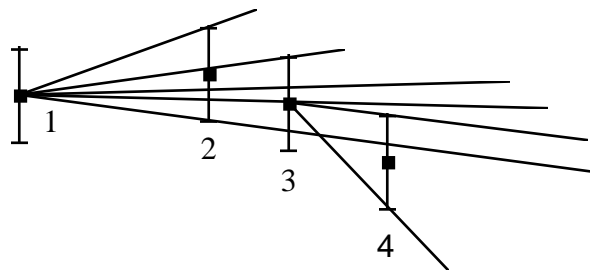
This, then, is SLIM1.

6. SLIM2

SLIM2 is similar to SLIM1 except that we record the actual previous value and time whenever a new point's tolerance band falls outside the "fan". Alas, to meet our iron-clad guarantee we must set the tolerance bands to \pm one-half of max_dev .

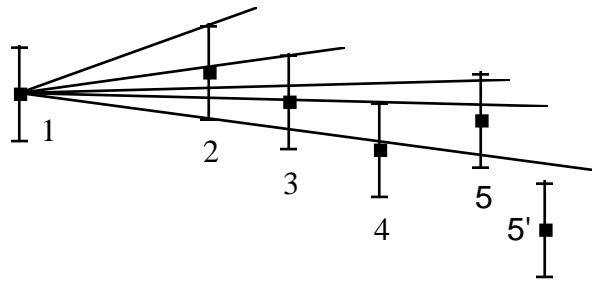
7. SLIM3

We revert to using $\pm \text{max_dev}$ instead of \pm one-half of max_dev for the tolerance bands, and tentatively mark a previous point for recording when its value (rather than its tolerance band) falls outside the "fan". Consider:



When point 4 arrives we go back and store point 3, and start a new fan from point 3 through the extremes of point 4's tolerance band.

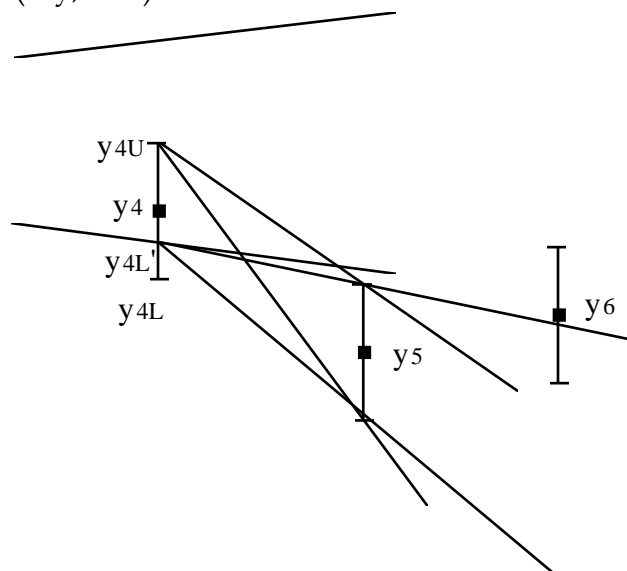
The performance of SLIM3 *could* be improved. When point 4 arrives,



do we *have* to go back and store point 3? Not necessarily because the next point may be such that it is unnecessary to store 3, as for example is the case with point 5. But if we have point 5' instead of point 5, then we must go back and store point 3, and proceed from there. Since we may have to store point 3, we must keep track of the fan emanating from 3 at least until such time we discover that we will not have to store it (i.e. a point within the fan like 5 comes along.) Of course, if point 4's tolerance band had not overlapped the fan, we would have immediately stored point 3. We might expect a small-to-modest increase in compression ratio, at the expense of more complicated programming. We haven't evaluated this proposed improvement.

8. SLIMMER

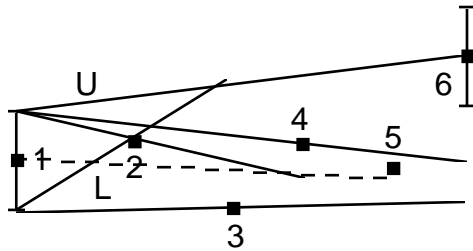
Yet another variant (on either SLIM1 or SLIM2, but illustrated with SLIM1) uses multiple (say, two) fans.



Instead of storing y_4 or y_4^L , we remember y_4^L and y_4^U . Now we start two fans, one at y_4^L and one at y_4^U . When we finally eliminate one, (e.g. as when point y_6 comes along) we store the apex of the surviving fan (in this case y_4^U) and continue. Indeed there

could be any number of fans emanating from points, say, equidistant on the subtended tolerance band of the last recorded point. There results potentially better compression, same or better noise immunity, and same or better fit; but additional fans to keep track of and more complicated logic to program.

6. Swinging Door as described by Bristol (11) with certain modifications.



Starting with points 1 and 2 we construct a pair of lines emanating from the extrema of 1's tolerance band and intersecting at point 2. For each new point we either:

- rotate L clockwise through the new point e.g. point 3,
- rotate U counterclockwise through the new point e.g. point 4,
- do nothing, e.g. point 5

When the lines L and U no longer intersect at some time more recent than point 1 (i.e. the swinging doors swing past parallel), as happens with point 6, we store the line segment (that is, the *two* end points) midway between the *previous* L and U (dotted line in the example) and start new swinging doors at the offending point.

Acknowledgments

I acknowledge my colleagues Roger Humphrey, Art McCready, and George Whittlesey without whose prodding this work might never have been done, leaving our clients to rely on less-satisfying methods for compressing their process data.

References

1. L. Ehrman, "Analysis of Some Redundancy Removal Bandwidth Compression Techniques", *Proceedings of the IEEE*, 55(3), March 1967, pp. 278-287.
2. C. M. Kortman, "Redundancy Reduction--- A Practical Method of Data Compression", *Proceedings of the IEEE*, 55(3), March 1967, pp. 253-263.

3. V. Cappellini, *et. al.* , *A Study on Data-Compression Techniques*, European Space Research Organisation, April 1974.
4. G. Benelli, V. Cappellini, F. Lotti, "Data Compression Techniques and Applications, *The Radio and Electronic Engineer*, 50(1/2), pp. 29-53, January/February 1980.
5. John C. Hale, Harold L. Sellars, "Historical Data Recording for Process Computers", *CEP*, November, 1981, pp. 38-43.
6. F. P. Bader, T. W. Tucker, "Real-Time Data Compression Improves Plant Performance Assessment, *InTech*, August 1987, pp. 53-56
7. J. P. Kennedy, "Data Storage for CIM", *Process Engineering*, February 1988, pp. 31-35.
8. K. R. Henry, "Data Compression Speeds Graphics, Saves Disk Space", *Control*, June 1994, pp. 47-49.
9. E. H. Bristol, "Swinging Door Trending: Adaptive Trend Recording?", *ISA National Conf. Proc.*, 1990, pp. 749-754.
10. J. P. Kennedy, "Data Treatment and Applications Future of the Desktop", *Proceeding of Foundations of Computer-Aided Process Operations*, 1993, pp. 21-43.
11. R. S. H. Mah, *et. al.* "Process Trending with Piecewise Linear Smoothing", *Computers Chem Engng*, 19(2), 1995, pp.129-137.
12. *Data Compression (Mar 70 - Present)*, Citations from the NTIS Bibliographic Database, Published Search PB95-859872, National Technical Information Service, Springfield, VA, 1995 [one of many NTIS searches on data compression].

TABLE I---COMPRESSION-METHOD GUARANTEES

Reconstruction compression method	Actual \pm max_dev of actual	Continuous values stored	Line segments
none	YES	YES	YES
boxcar	NO	YES	NO
backslope	NO	YES	YES
modified boxcar	YES	YES	NO
modified backslope	YES	YES	YES
value	YES	NO	NO
SLIM1	YES	NO	YES
SLIM2	YES	YES	YES
SLIM3	YES	YES	YES
modified Swinging Door	YES	NO	NOT REALLY

TABLE II - PERFORMANCE

Signal	Pure Sine Wave			Noisy Sine Wave		
Parameters	y[i] = 100 sin(t[i]), sampled 360x per period for four periods, max_dev = 1.5			y[i] = 100 sin(t[i]) + G[i], sampled 360x per period for four periods, max_dev = 1.5 G[i] is a random number normally distributed about 0.0 with standard deviation = 0.5		
Compression Method	Compression Ratio	Max Reconstruction Error		Compression Ratio	Max Reconstruction Error	
		Interpolative	Extrapolative		Interpolative	Extrapolative
None	1	0.00	0.00	1.00	0.00	0.00
Boxcar	1.27	1.37	1.21	1.32	2.35	2.57
Backslope	8.9	0.85	1.40	6.2	2.41	2.63
Modified Boxcar	1.13	0.85	0.55	1.14	1.06	0.73
Modified Backslope	6.2	0.52	0.70	1.47	1.23	0.75
Value	1.20	1.49	1.49	1.28	1.50	1.50
SLIM1	35	1.50	NA	16.7	1.50	NA
SLIM2	22.1	1.17	NA	5.3	1.32	NA
SLIM3	25.3	1.47	NA	9.8	1.50	NA
Modified Swinging Door	17.8	1.48	NA	10.1	1.50	NA

Notes

1. In general adding noise degrades performance as measured by compression ratio. However, if the compression ratio is near unity, almost every point is stored anyway, so noise may actually improve performance by providing data within the tolerance band.
2. Zero order (i.e. zero slope) compression methods (Boxcar, Modified Boxcar) may exhibit a smaller error (e.g. zero error for a triangular waveform synchronous with the sampling function), or a larger error (e.g. for staircase or rectangular signals) for interpolative reconstruction than for extrapolative reconstruction.
3. max_dev is the maximum deviation required. Modified Boxcar, Modified Backslope, and SLIM2 use a tolerance equal to one-half of max_dev in order to meet this requirement (see text).