Time Series Reduction

Scaling Data Visualisation

WHITE PAPER

By

Dr. Tim BUTTERS

Data Assimilation &
Numerical Analysis Specialist

tim.butters@sabisu.co
www.sabisu.co
Contents

1 Introduction 2

2 Challenge 2
   2.1 The Data Explosion 2
   2.2 Limitations of Simplistic Approaches 3
      2.2.1 Linear Data Skipping 3
      2.2.2 Data Averaging 4
   2.3 Desired Algorithmic Features 5

3 Solution 6
   3.1 Peak Based Reduction 6
      3.1.1 Peak Based Algorithm (PBA) 6
   3.2 Example 7

4 Conclusions 7

List of Figures

1 Linear data skip results for rapidly spiking time series 4
2 Data averaging reduction method for rapidly spiking time series 5
3 Peak based reduction algorithm results 8
1 Introduction

Time series are an ubiquitous tool throughout industrial operations and business. Be it the temperature trend of a heating element or the sale price of a commodity, a Cartesian time series plot is the most intuitive and easily understood way to display time dependant information.

With the rapid advancement of instrumentation and process historian capabilities it has become possible to record variables with higher frequencies for long periods of time. This generates huge amounts of data that can be utilised for decision support through mathematical analysis. However, this continual data growth poses a challenge in its visualisation. The data needs to be passed from server to client, and plotted on-screen, as rapidly as possible.

This white paper outlines the method developed by Sabisu to solve this problem through server-side processing, allowing the drastic reduction in the number of data points required to accurately reproduce the data trend.

2 Challenge

2.1 The Data Explosion

With high sampling rates of physical sensors, affordable data storage and regulatory requirements it is common for industrial installations to store vast quantities of process data that can be invaluable for the plant’s continual optimisation. To maximise the usability of this data it is vital that it can be efficiently and accurately displayed to the operator. To view one variable recorded continuously at one second intervals over a period of 7 days would require over 1.2M data values (time and quantity). This is data that must be sent from its storage location (e.g. an on-site process historian or database server) to the user’s machine, processed for plotting and drawn to the screen. This is not only an intensive computational process, there are a large number of redundant operations performed. Although there are 600,000 horizontally spaced points to be plotted, there will be far fewer pixels available on the screen. For example, a good quality 24” high definition monitor would provide fewer than 4000 horizontal pixels. Even for a variable stored at the much lower rate of one record per minute, more than 2.5 times more data points would be produced than there are available screen pixels. As it is highly unlikely that the graph would take up the entire screen space, it is
common that the actual number of available horizontal pixels will be of the order of a few hundred.

With this information it is clear that an algorithm that could reduce the number of data points in a given time series would be of benefit, as with the network load and processing time reduced the information could be delivered to the user in a much shorter time. It is, however, important that with this reduction the key characteristics of the data remain unchanged.

For example, one reason to view a large data set would be to ensure that a particular part of a plant or processes is operating within its safe limits. If, as a result of some data reduction transformation, the peaks and troughs of the data were distorted this could lead to inaccurate information being delivered to the operator concealing periods when parameters were above or below safe limits, or introducing apparent ‘unsafe’ behaviour that does not represent the actual conditions. It is for this reason that simplistic approaches should be employed with caution.

2.2 Limitations of Simplistic Approaches

When approaching the task of data reduction it is tempting to employ an intuitive algorithm using either a raw stripping procedure or some form of data averaging. However, it is unlikely that these approaches will yield satisfactory results in many cases.

2.2.1 Linear Data Skipping

The easiest method for reducing a data set would be to extract every $n^{th}$ point, were $n$ is some fixed integer. This also provides a predetermined reduction in data size, e.g. if every $4^{th}$ data point is taken, the reduced data will be 25% the size of the original set.

Although this approach is trivial to implement, it has clear limitations. It would be possible to completely exclude features in the data such as sharp spikes if they appear solely in an excluded block of data. For peaks that are partially within excluded blocks there could be an apparent reduction in magnitude, leading to false information being relayed to operators as periods when parameters are outside safe limits could be hidden.

This is demonstrated in figure 1, where panels B–D show the data skip reduction for increasing block sizes of the time series displayed in panel A.
Figure 1: Linear data skip results for rapidly spiking time series. Panel A shows the original data, panels B–D show the reduction for skip-block sizes of 2, 3 and 5 respectively.

As can be seen, peaks are concealed with even the smallest block size, an effect which increases as the block size gets larger.

This method also limits the possible compression ratio (defined as original data size / reduced data size), as each section of the time series is assumed to be equally important. This is unlikely, and an effective solution would reduce the number of points heterogeneously throughout the data, concentrating reduction around ‘uninteresting’ sections such as flat line areas.

2.2.2 Data Averaging

A simple averaging scheme goes some way to address the faults of the linear skip method. Rather than discarding data points, this solution would take the average value of consecutive ‘chunks’ of data (size $n$). This reduces the probability of missing peaks completely, but it is still likely that their size will be distorted by the algorithm; a problem that becomes more profound as $n$ increases. This can be seen in figure 2 which shows the results of such a scheme applied to a time series containing rapid spikes. Although peaks are not removed completely, their magnitude is greatly reduced, which could
Figure 2: Data averaging reduction method for rapidly spiking time series. Panel A shows the original data, panels B–D show the reduction for average-block sizes of 2, 3 and 5 respectively.

pose a safety risk as there is potential to conceal parameters operating in unsafe regimes.

Like the linear skipping method, this averaging scheme also treats each chunk of data equally, limiting the possible compression ratio.

2.3 Desired Algorithmic Features

Through consideration of the core problem, and the analysis of these simplistic data reduction methods, it is possible to formulate the desired specification of an effective time series reduction algorithm:

- **Fast Performance:** As the principle motivation behind time series reduction is speed, the resultant algorithm must be highly efficient with minimal overhead to facilitate rapid execution.

- **High Compression Ratio:** Due to the high sampling frequencies now in use throughout industry the algorithm must be capable of achieving high compression ratios. Although this may not be possible in all cases,
where possible the algorithm should exploit a heterogeneous approach to data reduction to ensure maximum speed-up.

- **Preservation of Data Range:** To ensure users can trust the data produced through the reduction procedure the limits of the reduced set should match the original data as closely as possible. Specifically, peak heights must not be changed.

## 3 Solution

To solve this data reduction problem a method has been devised that meets the criteria specified in section 2.3. This algorithm intelligently selects areas of the time series that can be reduced, whilst preserving segments with important features.

### 3.1 Peak Based Reduction

To ensure the inclusion of important features in the reduced data set the peak based reduction algorithm identifies key points that must be included in the reduced set. It then only excludes points that can be considered uninteresting (i.e. not distinctly different from neighbouring values).

#### 3.1.1 Peak Based Algorithm (PBA)

**FUNCTION:** PBA(data)

- Approximate data with line.
- IF: line fits data within tolerance
  - Save endpoints → STOP
- ELSE:
  - Add point furthest from line to reduced set.
  - Split data at point.
  - Call PBA on each new data chunk.

This pseudo-code details the way in which the peak based algorithm reduces data. It first approximates the data with a straight line, and measures the goodness-of-fit of the line to the data. If this is within a certain tolerance the end points of the line are added to the reduced set and the algorithm stops. If the data does not fit the line well, then the point furthest from this line...
is found and added to the reduced set. The data is then split at this point, and the PBA algorithm is performed on the left and right ‘chunks’ of data from the beginning. This recursive calling continues until all of the data has been processed.

This algorithm intelligently selects the sections of the time series that do not contain interesting information, which are identifiable by their good approximation to a straight best-fit line. It removes the points at the interior of the line, saving only the end points, giving data reduction. The algorithm preserves the detailed sections such as the peaks and troughs. This ensures the maximum relevant information is displayed, with the highest compression rate possible whilst preserving these features.

### 3.2 Example

Figure 3Ai shows a typical time series containing 13,121 points. This is of a size that was shown to cause significant delay if transmitted over an internal network and displayed within a Sabisu widget. Panel Bi shows an artificially generated time series with sharp, rapid, spikes. This was chosen to test the algorithm’s ability to respond to rapid changes as it is important that the reduction does not distort or conceal sharp spikes.

As can be seen from figure 3 the reduction algorithm performs well, reproducing the peak and trough magnitudes precisely, whilst giving a high compression ratio. The reduction shown in panels Ai and Aii from 13,121 points to 501 gives a compression ratio of 26.2, which is a 96.2% reduction in the number of data points. This is more than enough to allow for the rapid transmission and plotting of the data. The reduction from 1,425 to 39 data points shown in panels Bi and Bii represent a compression ratio of 36.5 (a reduction of 97.3%). Such a high ratio is obtained because a large amount of reduction can be carried out on the early flat section of the series. Importantly, the peaks are reproduced accurately, with no distortion of magnitude.

### 4 Conclusions

As shown in figure 3, the new reduction algorithm developed by Sabisu produces excellent results. The combination of high compression ratio and levels of detail make peak based reduction an ideal solution to the problem of large
Figure 3: Peak based reduction algorithm. A typical time series (Ai) and the results of its peak based reduction (Aii). Panels Bi and Bii show an example of a rapidly spiked time series and its corresponding reduction.
data visualisation.

As outlined in section 2.2 this is not the simplest solution to data reduction. However, as demonstrated through the examples of data skipping and averaging, simplistic approaches are often only valid in specific cases and are not suitable for general use. This is especially true when a certain level of accuracy is required by the user.

Time series reduction is an important challenge in modern business and operational environments. To fully leverage the wealth of information available within data it is paramount that it is readily accessible, a challenge that becomes more difficult as the size of the data increases.

Sabisu has developed a novel solution to this problem that allows the platform to scale with the client data seamlessly.