Automatic Tuning of the Window Size in the Box Car Backslope Data Compression Algorithm

Jens Pettersson*
Division of Automatic Control
Royal Institute of Technology
100 44 Stockholm, Sweden

Per-Olof Gutman†
Faculty of Agricultural Engineering
Technion — Israel Institute of Technology
Haifa 32000, Israel

Abstract

An automatic algorithm to tune the Box Car Backslope (BCBS) data reduction algorithm is given in this paper.

1 Introduction

In process industries such as the pulp and paper industry, thousands or ten of thousand variables are measured and stored. For most of these variables the sampling rate might be large, e.g. one reading/storage per minute. The data is used not only for immediate display on the operator monitors but is also saved for future reference and analysis. A typical storage period is one year. Even though computer storage media have become cheap and compact, the huge amount of data to be stored still necessitates the use of data reduction algorithms whose purpose is to reduce the amount of stored data points while retaining the essential features of the measured signals.

This paper will not discuss the relative merits of various data reduction algorithms, but concentrate on the Box Car Back Slope method (BCBS) (Hale and Sellars, 1981), which is popular in the process industry.

The BCBS is a real time algorithm, i.e. at sampling instant n, when the measured value $y(n)$ is sampled, it decides whether $y(n-1)$ should be stored, or not. The BCBS contains no mechanism to post-process the data in order to change the stored data points. Hence the BCBS is a filter. It is obvious that better data reduction can be achieved with a smoothing or post-processing algorithm, which would require, however, that measured data points be stored for some time.

The underlying assumption of the BCBS is that the signal itself (without measurement noise) can be well approximated by a piece-wise linear function and that the frequency of transitions from one line segment to another is considerably smaller than the measurement noise frequency. The crucial parameter of the BCBS is the window size, h. If both $y(n)$ and $y(n-1)$ fall within the current window(s), $y(n-1)$ is not stored, otherwise $y(n-1)$ may be stored. An approximant of the original signal is then formed by linear interpolation of the stored values. The larger h is, the less data points will be stored and the less faithful the approximant will be. Our experience with the carton board machine at Assi-Domän Frövi, Sweden, shows that for many measured

*E-mail: *jep@s3.kth.se*
†E-mail: *peo@tx.technion.ac.il*

signals, BCBS with a window size approximately equal to four noise standard deviations achieves satisfactory signal approximation with only about 10% of the original data. It is however easy to synthesize noisy signals for which this rule of thumb gives too modest a data reduction.

It is however impossible to tune the window size manually for thousands of signal channels. In most of them h remains at its default start up setting, e.g. 1% of the allowed signal span which may be much larger than the actual measured signal variation. The result may be that almost no data points are stored (Hanimann, 1998).

In this chapter we propose an algorithm to automatically tune the BCBS window size. This is a non-trivial problem even if the measurement noise standard deviation was known, as indicated above. The algorithm is based on the minimization of a criterion that weighs the data reduction and the deviation of the approximant from the measured signal. The criterion is estimated during each piece of the piece-wise linear approximant, and a modified descent search is performed to find the optimal window size.

The algorithm is analysed for the simple synthetic signal consisting of a linear function with identically independent Gaussian measurement noise, and is illustrated with simulations on synthetic signals.

First, however, the BCBS algorithm is reviewed and illustrated in section 2.

2 The Box Car Back Slope algorithms

Assume without loss of generality that the sampling period equals one and that the sampling instants are $t = 1, 2, 3, \ldots$. Let the current sampling instant be $t = n$, with $n \geq 3$. The measured scalar data up to and including the current sampling instant are denoted $y(1), y(2), \ldots, y(n-1), y(n)$. The set of stored data, in the form of time-value pairs, up to and including the current sampling instant, is

$$S_n = \{[t_1, y(t_1)], [t_2, y(t_2)], \ldots, [t_{m-1}, y(t_{m-1})], [t_m, y(t_m)]\}, \tag{1}$$

where m is a increasing function of n, and in particular $m \leq n-1$. The BCBS algorithm is initialized with $t_1 = 1$ and $t_2 = 2$.

There are two windows in the BCBS algorithm: the Box Car window (BCwindow), and the Back Slope window (BSwindow). The windows are illustrated in figure 2, and are defined as follows:

$$\text{BCwindow} = \{x(t) | y(t_m) - h \leq x(t) \leq y(t_m) + h,\ t \geq t_m\} \tag{2}$$

$$\begin{aligned} \text{BSwindow} &= \{x(t) | y(t_m) - h + \frac{y(t_m) - y(t_{m-1})}{t_m - t_{m-1}}(t - t_m) \leq x(t) \leq \\ &\leq y(t_m) + h + \frac{y(t_m) - y(t_{m-1})}{t_m - t_{m-1}}(t - t_m),\ t \geq t_m\} \end{aligned} \tag{3}$$

where h is the window size. Note that the width of the window intersected along the y-axis is $2h$.

A window is said to be *active* if the incoming measurement, $y(n+1)$ and its predecessor $y(n)$ are tested against it: the *Box Car test* fails if at least one of two measurements does not belong to BCwindow, and, *the Back Slope test* fails if at least one of the two measurements does not belong to BSwindow. Which windows are active depends on the state p that may assume the values 0 (both windows are active), 1 (BCwindow active), and 2 (BSwindow active). The state transition diagram is found in figure 1.

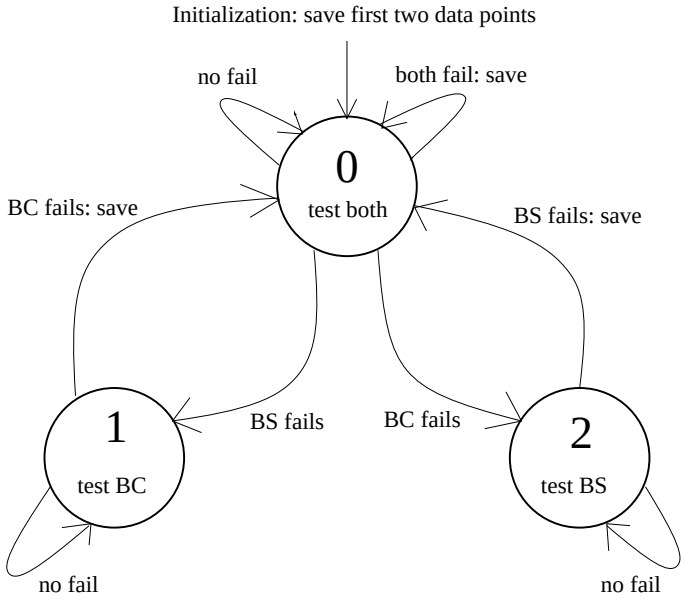

Figure 1: State transition diagram for the Box Car Back Slope data reduction algorithm

The algorithm is initialized with $p = 0$. When $p = 0$ the state remains there as long the incoming measurement $y(n+1)$ and its predecessor $y(n)$ both belong to BCwindow and to BSwindow, or both of them fail the two tests. In the latter case, denoted as "both fail" in Figure 1, $[n, y(n)]$ is appended to S_{n+1}, unless $[n, y(n)]$ already belongs to S which happens for $n = 2$. If the Back Slope test fails alone then $p = 1$, and BSwindow ceases to be active. As long as the Box Car test does not fail the state remains $p = 1$. When the Box Car test fails, p returns to 0 and $[n, y(n)]$ is appended to S_{n+1}. When $p = 0$, and the Box Car test fails, p becomes 2, and BCwindow ceases to be active. As long as the Back Slope test does not fail the state remains $p = 2$. When the Back Slope test fails, p returns to 0 and $[n, y(n)]$ is appended to S_{n+1}. Clearly, when data is appended to S_{n+1}, m is increased by one, and the windows change.

2.1 Example

Figures 2 and 3 illustrates the BCBS algorithm, with $h = 1$, on each consecutive measurement of the sequence $y(1), y(2), \ldots, y(10)$ which simulates e.g the measured step response of a controlled system. At $t = 2$ the two initial values have been stored and they are marked with a star (*). The state $p = 0$, and both windows are active. The edge of BCwindow is drawn with a solid line, and the edge of BCwindow is dashed.

At time $t = 3$ the measurement $y(3) = 4$ is recorded, marked with a ring (o). Since $y(3)$ and $y(2)$ both belong to BCwindow and BSwindow ("no fail" in figure 1), p remains equal to 0 and no value is stored into S_3.

The next measurement is $y(4) = 7$. Clearly, only the BC test fails, since $y(4) \notin$ BCwindow while $y(4) \in$ BSwindow, $y(3) \in$ BCwindow, and $y(3) \in$ BSwindow. Hence, p becomes 2, and only the Back Slope window remains active.

At $t = 5$, both the new measurement $y(5) = 9$ and the preceding $y(4)$ belong to BSwindow, and p remains equal to 2. At $t = 6$, the current measurement $y(6) = 6$ fails the Back Slope test,

and p becomes 0. As a result $[5, y(5)]$ is appended to S_5, and a new BCwindow (solid) and a new BSwindow (dash-dotted) are defined as seen in figure 3 f). Note that the new Back Slope window by chance coincides with the previous one, but it is marked dash-dotted to emphasize that it is new.

At $t = 7$, both the new measurement $y(7) = 5$ and the preceding $y(6)$ fail both tests; "both fail" in figure 1. Hence p remains 0, and $[6, y(6)]$ is appended to S_7, and a new BCwindow (solid) and a new BSwindow (dashed) are computed, see figure 3 g).

The following measurement, $y(8) = 6$ in figure 3 g), makes the BS test fail, since $y(8)$ and $y(7)$ both belong to BCwindow and do not belong to BSwindow. The state p then switches to 1 implying that only the BC window remains active. Notice that if $y(7)$ had had a value less than 5 or larger than 7, then the state transition at $t = 7$ in figure 3 f) would have remained unchanged $0 \to 0$, but at time $t = 8$ in figure 3 g) both tests would have failed and the state would have remained $p = 0$.

This is however not the case and, as noted above, the state is initially $p = 1$ at time $t = 9$ and only the Box Car window is active as seen in figure 3 h) when the measurement $y(9) = 6$ comes in. The state remains $p = 1$ since BCwindow contains both $y(9)$ and $y(8)$.

At $t = 10$ however, $y(10) = 7.5$ is outside BCwindow, the state returns to $p = 0$, and $[9, y(9)]$ is appended to S_{10}. The computation of the two new windows is left as an exercise for the reader.

One may appreciate the goodness of the BCBS approximant in figure 3 i). The approximant is given by the straight line segments between the stored values which include the points marked by stars, and $[9, y(9])$, i.e. $S = \{[1, 1]$, [2,3],[5,9],[6,6], $[9, 6]\}$ The drawing of the approximant is left to (the imagination of) the reader. The RMS of the approximation error for $t = 1, 2, \ldots, 9$ is

$$\sqrt{(0 + 0 + (-1)^2 + 0 + 0 + 0 + (-1)^2 + 0 + 0)/9} = 0.47.$$

If no data reduction had taken place, 9 real numbers would have had to be stored. After the reduction 10 numbers[1] are stored! Hence, in this example there is no reduction at all. This concludes the example.

3 An off-line algorithm for determining the window size

It was stated above that the Box Car Back Slope algorithm is a filter implying that measured consecutive data are not saved temporally for post-processing. In order to develop an efficient window sizing algorithm it is however instructive to investigate how the window size would have been chosen if a batch of measured data was at hand.

Assume therefore that a batch of N measurements have been recorded, and that it is desired to select the stored data set, S_N (1) with the Box Car Back Slope algorithm. Since the window size h reflects a trade-off between the data reduction ratio and the goodness of the approximant, it is reasonable that h should be chosen such that an appropriate criterion is minimized. In this section a criterion will be proposed, and analysed in a simple case, for which also a numerical example is given.

Define $e(t) = y(t) - \hat{y}(t)$ as the approximation error at time t, where $y(t)$ is the measured value, and $\hat{y}(t)$ the BCBS generated approximant. Let $R_N = 1/N \sum_{t=1}^{N} e^2(t)$ be the sample mean square of the approximation error.

Let $r_N = \text{card } S_N/N$ define the *data reduction ratio*, where card S_N denotes the number of stored pairs in S_N, i.e. half the amount of numbers actually stored.

[1] In general the time stamp does not have to be stored as a real number

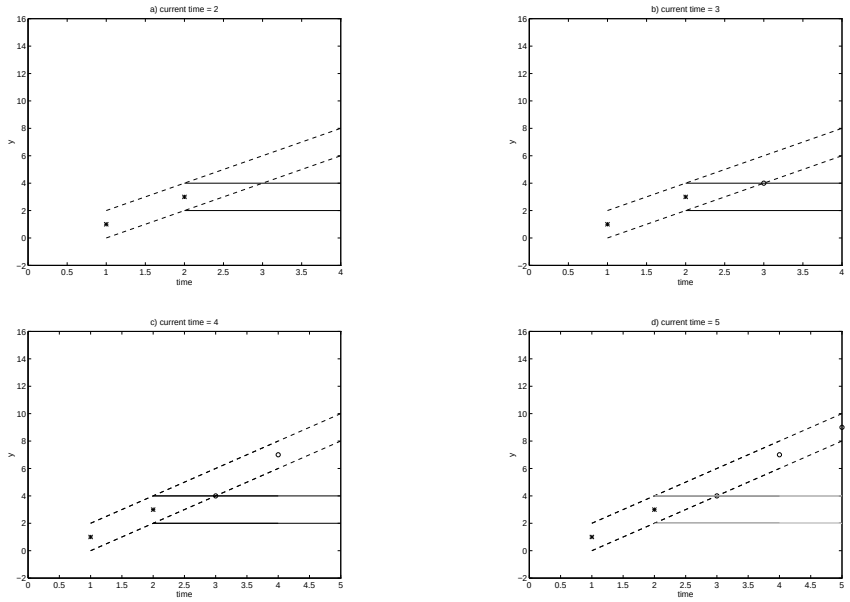

Figure 2: An illustration of the Box Car Back Slope algorithm for Example 2.1. Solid lines denote the edges of Box Car windows, and dashed or dash-dotted lines denote the edges of Back Slope windows. Values saved in the set S are marked with a star, and currently unsaved vales are marked with a ring. The state transitions are in a) $0 \to 0$, b) $0 \to 0$, c) $0 \to 2$, d) $2 \to 2$.

An appropriate criterion is then

$$V_N(h) = (1-\alpha) r_N + \alpha \frac{R_N(e)}{\sigma_N^2(y)} \tag{4}$$

where $\alpha \in [0,1]$ and $\sigma_N^2(y) = 1/(N-1)\sum_{t=1}^{N}[y^2(t) - 1/N(\sum_{t=1}^{N} y(t))^2]$ is the variance of $y(t)$. The constant α is a user-chosen constant that reflects the trade-off between data reduction and the reproduction of the original signal. For example, α close to 0 will emphasise data reduction. The window size is then found from

$$\hat{h} = \arg \min_h V_N(h) \tag{5}$$

3.1 Analysis of V_N

We vill now for a simple case analyse exact expression of $V_N(h)$, given a set of data. In order to simplify the calculations we consider only the Box Car algorithm, and in particular the *true* Box Car algorithm, i.e. when only the value that fails the boxcar test is stored and the data reproduction is done by zero-order holding of each stored value.

Consider now the case when $y(t) \in \mathcal{N}(0, \sigma^2)$. Assume that some value, $y(t_0)$, at time t_0 was stored. Then we are interested in the probability that $y(t_k)$, where $k \geq 1$ will be stored. We are also interested in the variance of the error $e(t_k)$ if $y(t_k)$ is not stored. Since $y(t_1)$ and $y(t_k)$ are independent, we have that for the process $z(t_k) = y(t_k) - y(t_0)$, $z(t_k) \in N(0, 2\sigma^2)$. Now, let $r = \mathrm{P}(\text{store})$, i.e. the probability that a value is stored. Thus,

$$\mathrm{P}(\text{store}) \quad = \quad \mathrm{P}(|z(t_k)| > h)$$

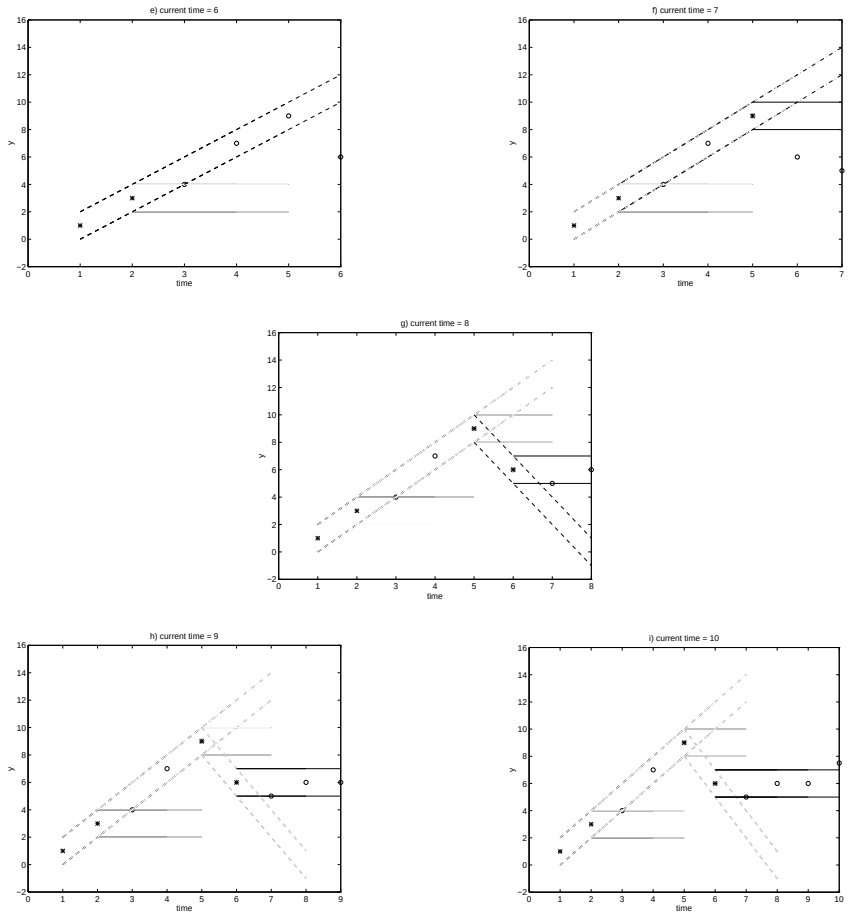

Figure 3: An illustration of the Box Car Back Slope algorithm for example 2.1 (cont.). Solid lines denote the edges of Box Car windows, and dashed or dash-dotted lines denote the edges of Back Slope windows. Values saved in the set S are marked with a star, and currently unsaved vales are marked with a ring. The state transitions are in e) $2 \rightarrow 0$, f) $0 \rightarrow 0$, g) $0 \rightarrow 1$, h) $1 \rightarrow 1$, i) $1 \rightarrow 0$.

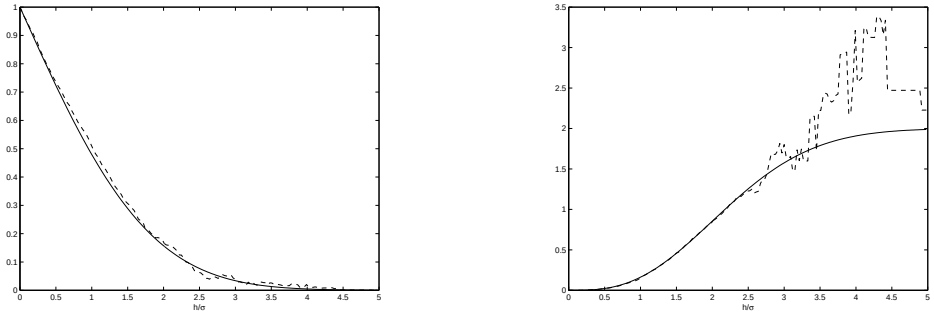

Figure 4: Analytical values (solid lines) and simulated values (dashed lines) of r_N (left) and $R_N^2(e)/\sigma_N^2(y)$ (right) as functions of h/σ. In the simulation $N = 1000$ was used. Notice the big difference between the analytical and simulated values of $R_N^2(e)/\sigma_N^2(y)$ for large h.

$$\begin{aligned} &= 1 - \frac{1}{2\sigma\sqrt{\pi}} \int_{-h}^{h} e^{-x^2/(4\sigma^2)} dx \\ &= 1 - \mathrm{erf}(h/2\sigma) \end{aligned} \tag{6}$$

For the error, $e(t_k)$, we have

$$e(t_k) = \begin{cases} z(t_k), & \text{if } |z(t_k)| < h \\ 0, & \text{otherwise} \end{cases}$$

The statistical properties of the error are then

$$\mathrm{E}\{e(t_k)\} = \frac{1}{2\sigma\sqrt{\pi}} \int_{-h}^{h} x e^{-x^2/(4\sigma^2)} dx = 0 \tag{7}$$

and

$$\begin{aligned} \mathrm{E}\{e^2(t_k)\} &= \frac{1}{2\sigma\sqrt{\pi}} \int_{-h}^{h} x^2 e^{-x^2/(4\sigma^2)} dx \\ &= \frac{2\sigma}{\sqrt{\pi}} (\sigma\sqrt{\pi}\mathrm{erf}(h/2\sigma) - h e^{-h^2/4\sigma^2}) \end{aligned} \tag{8}$$

where $\mathrm{erf}(x) = (2/\sqrt{\pi}) \int_0^x e^{-x^2} dx$. Hence, $R(e) = \mathrm{E}\{e^2(t_k)\}$.

In figure 4 the analytical functions for r and $R(e)/\sigma^2(y)$ are plotted together with calculated values of r_N and $R_N^2(e)/\sigma_N^2(y)$, respectively, from a simulation of $y(t)$ with $N = 1000$. As can be seen, the analytical functions are smooth, while the curves from the simulation contains large peaks for higher values of h. Hence, the conclusion is that V_N may have many local minima for finite N while the theoretical analysis shows it only to have a global minimum.

4 An on-line algorithm for determining the window size

In this section we will present an on-line algorithm that adaptively tunes the window size h. The basic assumption is that the signal $y(t)$ can be described by a low frequency signal with an additive Gaussian measurement noise, i.e. $y(t) = y_0(t) + n(t)$, where $y_0(t)$ is the "true" signal and $n(t) \in \mathcal{N}(0, \sigma^2)$. If σ was known an intuitive condition on h would be $h > d \cdot \sigma$ where the

constant d would be equal to, say 4, since otherwise a too low data reduction will be achieved, see figure 4. On the other hand, a higher value of d might be bad, since it could give poor reproduction of the original signal.

A first step in the on-line algorithm is thus to estimate the noise standard deviation σ. This is done by introducing the process $x(t_i) = y(t_i) - y(t_{i-1})$. Since we have assumed that $y_0(t)$ is a low-frequency signal, $y_0(t_i) \approx y_0(t_{i-1})$ and $x(t_i) \in \mathcal{N}(0, 2\sigma^2)$. The estimate $\hat{\sigma}$ can be calculated in a recursive maner by

$$\hat{\sigma}^2(t_i) = \lambda \cdot \hat{\sigma}^2(t_{i-1}) + (1-\lambda)x^2(t_i)/2 \tag{9}$$

where the forgetting factor λ typically is 0.95-0.99 (Ljung, 1987). The use of a forgetting factor enables us to deal with time varying noise.

The above method may give a large bias in the case of a signal with a slope. However, the slope is detected by the BCBS-algorithm, and the estimate $\hat{\sigma}$ can be modified when in Back Slope mode, by assuming that $y(t) = m \cdot (t - t_0) + y_0 + n(t)$, when in Back Slope mode. Here the time t_0 is when the BCBS algorithm enters the Back Slope mode, m is the slope and y_0 is a constant. Now $x(t_i) = y(t_i) - y(t_{i-1}) = m \cdot (t_i - t_{i-1}) + n(t_i) - n(t_{i-1})$. Assume now that the time is normalized so that $t_i - t_{i-1} = 1$, then $x(t) \in \mathcal{N}(m, 2\sigma^2)$. The recursive estimate of σ is now

$$\begin{aligned} \hat{m}(t_i) &= \hat{m}(t_{i-1}) + 1/(t_i - t_0)\left[x(t_i) - \hat{m}(t_{i-1})\right] \\ \hat{\sigma}^2(t_i) &= \lambda \cdot \hat{\sigma}^2(t_{i-1}) + (1-\lambda)\left[x(t_i) - \hat{m}(t_i)\right]^2/2 \end{aligned} \tag{10}$$

Equation 10 is initialized with $\hat{m}(t_0) = x(t_0)$ and is proceeded until a recording is made and the BCBS algorithm leaves the Back Slope mode.

We now have the estimate $\hat{\sigma}$ of the noise standard deviation. The question then is how to choose the factor d. As mentioned in section 1 using the *ad hoc* $d = 4$ may give either poor data reproduction or poor data reduction. Instead, d may be chosen such that the criterion like equation (4) is minimized. The minimization can be done on-line with a recursive minimization algorithm presented in (Baril and Gutman, 1997). The minimization algorithm functions as follows:

Let the function to be minimized be given by

$$V_n = (1-\alpha)r_n + \alpha\frac{R_n(e)}{\sigma_n^2(y)} \tag{11}$$

which is a function of h, and since $h = d \cdot \hat{\sigma}$, it is subsequently a function of d. Assume that we during the time interval $\mathcal{T}_n = [T_n, T_{n+1}]$, where $T_{n+1} - T_n = N_n$ and values are stored at the interval endpoints, have $d = d_n$. This means that d only is updated after each interval $\mathcal{T}_n$. Now, let

$$\Sigma_n = \text{sign}\left[\frac{V_n - V_{n-1}}{d_n - d_{n-1}}\right] \tag{12}$$

The recursive algorithm is then

$$d_{n+1} = d_n - \gamma_n\Sigma_n \tag{13}$$

where

$$\gamma_n = \begin{cases} b\gamma_{n-1}, & \text{if } \Sigma_n\Sigma_{n-1} = -1 \\ a\gamma_{n-1}, & \text{otherwise} \end{cases}$$

with $\gamma_0 > 0$. In (Baril and Gutman, 1997) a value set for a and b is given for which the algorithm converge. To use this algorithm, we must calculate V_n for the time interval T_n without storing

any more values then stored by the BCBS algorithm. First we will consider the mean square of the error,

$$R_n(e) = \frac{1}{N_n} \sum_{t_i=T_n}^{T_{n+1}} e^2(t_i) \tag{14}$$

To calculate $R_n(e)$ it is thus essential to know $\sum_{t_i=T_n}^{T_{n+1}} e^2(t_i)$. From the choice of $\mathcal{T}_n$, it consists of L_n intervals of unstored values. Assume that during interval j, $j = [1, \ldots, L_n]$, values are stored at time t_0 and t_k and that no values are stored in between t_0 and t_k. At time t_i, where $t_0 < t_i < t_k$ we naturally do not know neither the time t_k, nor the value of $y(t_k)$. We therefore introduce the estimated error at time t_i,

$$\hat{e}(t_i) = y(t_i) - y(t_0) \tag{15}$$

The real error, $e(t_i)$, after the BCBS algorithm has stored the value at time t_k and interpolated between $y(t_0)$ and $y(t_k)$, is given by

$$e(t_i) = y(t_i) - (t_i - t_0)\frac{y(t_k) - y(t_0)}{t_k - t_0} - y(t_0) \tag{16}$$

Now, let $s = (y(t_k) - y(t_0))/(t_k - t_0)$ and $i = t_i - t_0$, then

$$e(t_i) = \hat{e}(t_i) - i \cdot s \tag{17}$$

Then

$$\sum_{i=0}^{k-1} e^2(t_i) = \sum_{i=0}^{k-1} \hat{e}^2(t_i) + 2s \sum_{i=0}^{k-1} i\hat{e}^2(t_i) + (\frac{k^3}{3} - \frac{k^2}{2} + \frac{k}{6}) \cdot s^2 \tag{18}$$

The sums $\sum_{i=0}^{k-1} \hat{e}^2(t_i)$ and $\sum_{i=0}^{k-1} i\hat{e}(t_i)$ can be computed recursivly, for example, $\sum_{i=0}^{k-1} i\hat{e}(t_i) = (k-1)\hat{e}(t_{k-1}) + \sum_{i=0}^{k-2} i\hat{e}(t_i)$. At time t_k, when the recording is made, s and k becomes known, and $R_j = \sum_{i=0}^{k-1} e^2(t_i)$ can be calculated. Thus, the total mean square error during $\mathcal{T}_n$ is,

$$R_n(e) = \frac{1}{N_n} \sum_{j=1}^{L_n} R_j \tag{19}$$

We also need to calculate the variance $\sigma_n^2(y)$ during $\mathcal{T}_n$. We have that

$$\sigma_n^2(y) = \frac{1}{N_n - 1}(\sum_{T_n}^{T_{n+1}} y^2(t_i) - [\frac{1}{N_n} \sum_{T_n}^{T_{n+1}} y(t_i)]^2) \tag{20}$$

Here $\sum_{T_n}^{T_{n+1}} y^2(t_i)$ and $\sum_{T_n}^{T_{n+1}} y(t_i)$ can be computed recursivly. The number of stored values during $\mathcal{T}_n$, S_n, is simply calculated by letting $S_n = S_n + 1$ if a value is stored. The data reduction ratio is then

$$r_n = S_n / N_n \tag{21}$$

Finally, at time T_{n+1}, V_n can be calculated.

During simulations of the on-line tuning algorithm, it became evident that the search algorithm may find a local minimum. The reason for this is that V_n is realization dependent. This was solved by a) choosing $N_n > M$ where M is large enough to make V_n smooth and b) by modifying equation(13) in the following way

$$d_{n+1} = d_n - \gamma_n \Sigma_n + c \cdot w_n \tag{22}$$

where c is a constant and w_n is a stochastic variable, with $w_n \in \mathcal{N}(0, 1)$. The search algorithm will thus always take a step in some direction and eventually find a global minimum.

In appendix A the full on-line tuning algorithm is shown in pseudo-code.

4.1 Simulations

To verify that the proposed algorithm functions as designed, it has been simulated on both process data and superficial data. In the simulations the following numerical values were used: $\lambda = 0.95$, $M = 600$, $c = 0.2$ and $\alpha = 0.01$.

In figure 5 a sequence from one of the process signals is shown. The original signal, reproduced signal and stored values are shown in figure 5 a. One can clearly see that the tuning algorithm finds a window size h which captures most of the variation in the original signal without storing too many values. For this particular signal a compression ratio of 99% was achieved. Values of d are presented in figure 5 b, the algorithm converges to $d \approx 7.5$.

The second test signal is a sinus without noise. Although such a signal hopefully is not representative of signals in the process industry, it is interesting to see how the tuning algorithm can handle this case. The result of the simulation is shown in figure 6 a and one sees that the interpolated signal would capture most of the variations in this signal too. Also, d converges to a higher values in this case, see figure 6 b.

5 Conclusions and Discussion

We have in this chapter studied the Box Car Back Slope algorithm for compressing process data. The performance of the BCBS algorithm depends strongly on the chosen window size h. A too small window will result in a low data reduction ratio, while a too large window will result in loss of information.

An on-line tuning algorithm for the window size was proposed. The tuning algorithm first estimates the measurement noise, then chooses the window size as a factor times the estimated measurement noise. The factor is chosen by adaptively minimizing a loss function.

Also, we believe that process information systems using the BCBS algorithm must, beside on-line tuning algorithms, utilize some kind of fault diagnosis function. For example, we have shown it possible to recursivly calculate the statistics of the error for the reproduced signal. The statistics could then be shown to the user of the reproduced signal, such that he/she gets a figure of the accuracy of the signal. It would also be possible to include alarm functions, which alerts when the error becomes too large or the reduction ratio becomes too small, indicating that the window size must be re-tuned.

A Algorithm for on-line tuning of the Box Car Back Slope algorithm

We will now show the full algorithm for the on-line tuning of the window h.

Initialize all variables
`repeat until`
  [store,p]=BCBStest$(y(t), t, \ldots, p)$
  `if` store
    $r_n := r_n + 1$
    `if` $i > 1$
      $s := (y(t-1) - y(t_r))/i$
      $R_j = R_j + \Sigma_{\hat{e}^2} + 2s \cdot \Sigma_{i\hat{e}} + (\frac{k^3}{3} - \frac{k^2}{2} + \frac{k}{6}) \cdot s^2$
      $i := 0$
      $\Sigma_{\hat{e}^2} := 0$

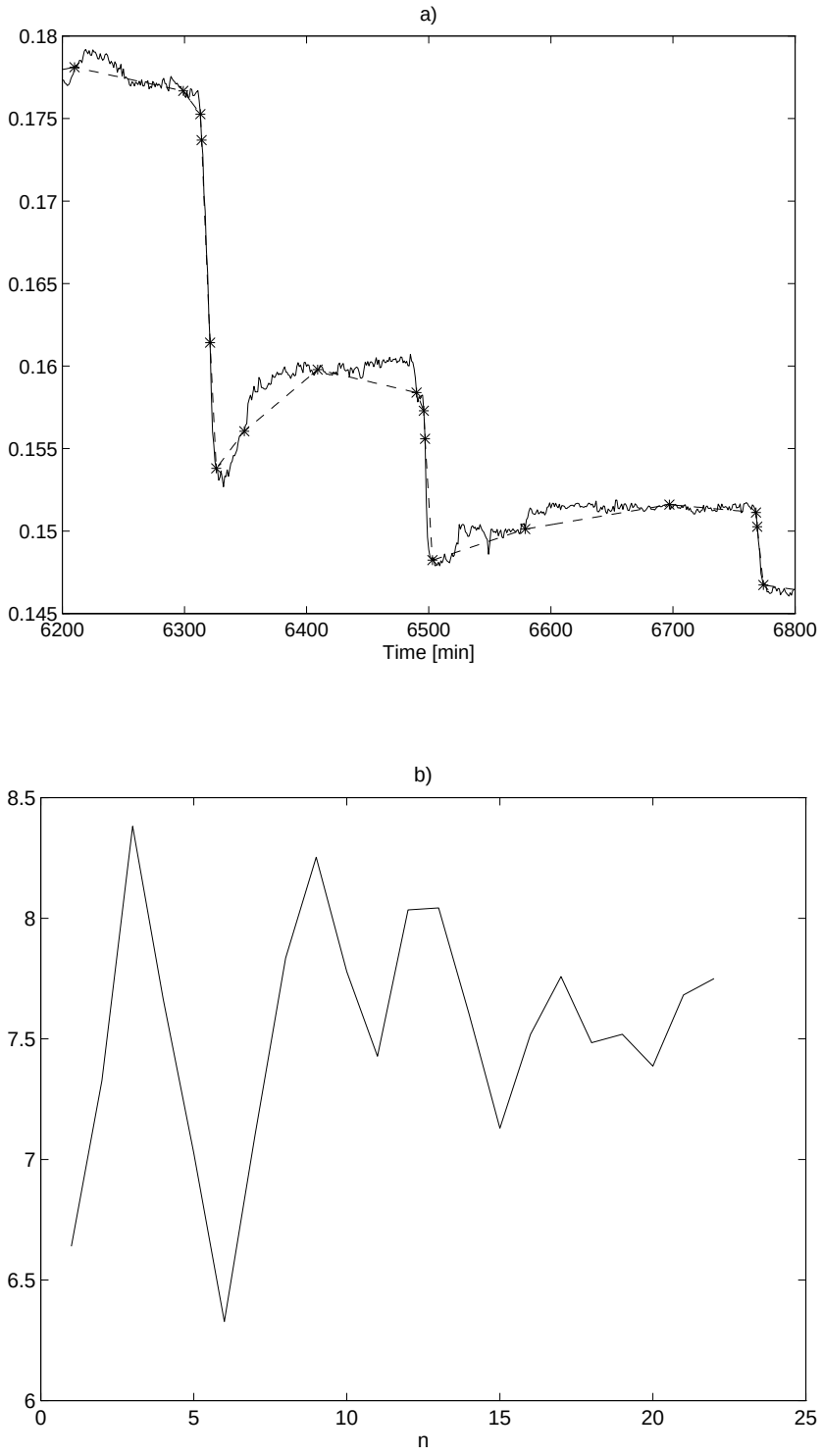

Figure 5: Simulation of the BCBS tuning algorithm with the first test signal. In a) the original signal (solid line), stored values ("*") and reproduced signal (dashed line) are shown. The reproduced signal is close to the original signal. In b) the values of d are displayed.

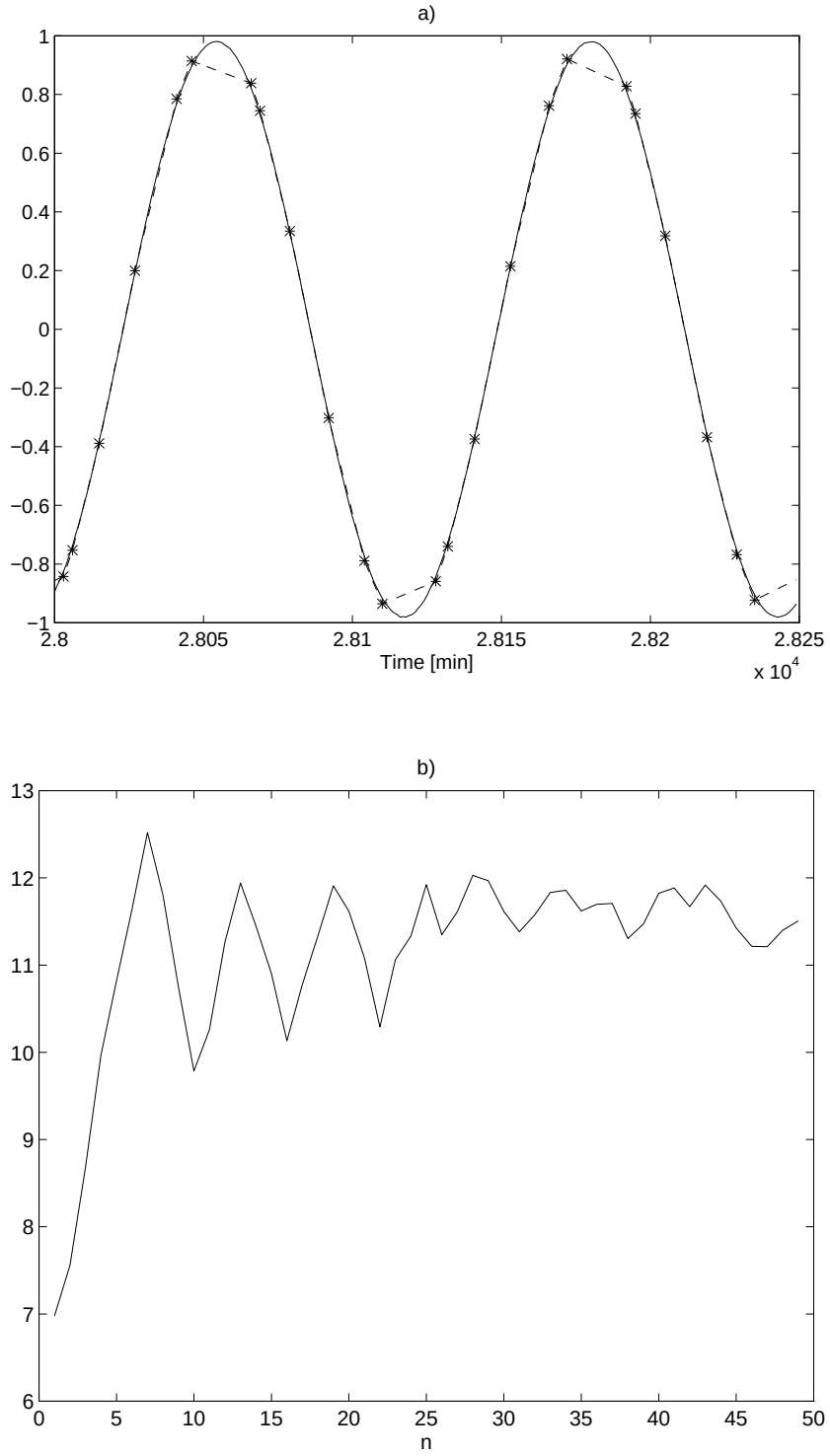

Figure 6: Simulation of the BCBS tuning algorithm with the second test signal. In a) the original signal (solid line), stored values ("*") and reproduced signal (dashed line) are shown. The BCBS algorithm "cuts" the tops of original signal but the overall reproduction is fairly good. In b) the values of d are displayed. As can be seen, it converges to a larger value than for the first signal.

```
            Σ_iê := 0
          end
          m̂ := 0;  t_m := 0
          if M_n > M
            R_e := R_j/M_n
            r_n := r_n/M_n
            σ_y^2 := (Σ_y² − (Σ_y/M_n)²)/(M_n − 1)
            V_n := (1 − α)r_n + αR_e/σ_y^2
            S_n := sign((V_n − V_{n−1})/(d_n − d_{n−1}))
            if S_n S_{n−1} = 1
              γ_n := aγ_{n_1}
              else γ_n := bγ_{n_1}
            end
            d_{n+1} := d_n − γ_n S_n + cw_n
            d_{n−1} := d_n;  d_n := d_{n+1}
            S_{n−1} := S_n;  S_n := S_{n+1}
            V_{n−1} := V_n
            r_n := 0;  E_n := 0
            Σ_y² := 0;  Σ_y := 0
          end
        end
        else
            z := y(t) − y(t − 1)
            if p = 1
              σ̂² := λσ̂² + (1 − λ)z²/2
              h := d√σ̂²
            end
            if p = 2
              t_m := t_m + 1
              m̂ := m̂ + 1/t_m(z − m̂)
              σ̂² := λσ̂² + (1 − λ)(z − m̂)²/2
              h := d√σ̂²
            end
            M_n := M_n + 1;  i := i + 1
            ê := y(t) − y(t_r)
            Σ_ê² := Σ_ê² + ê²
            Σ_iê := Σ_iê + iê
        end
        Σ_y := Σ_y + y(t)
        Σ_y² := Σ_y² + y(t)²
end
```

References

Baril C.G., and P.O. Gutman (1997). "Performance enhancing adaptive friction compensation for uncertain systems," *Transactions on Control Systems Technology*, **5**, no. 5, pp. 465–479.

Hale J.C., and H.L. Sellars (1981). "Historical Data Recording For Process Computers", *Chemical Engineering Progress*, **37**, no. 11.

Hanimann M. (1998). Personal communication with M. Hanimann, AssiDomän Frövi AB.

Ljung L. (1987). *System Identification, Theory for the user*. Prentice Hall.