On-Line Data Compression and Error Analysis Using Wavelet Technology

Manish Misra and S. Joe Qin
Dept. of Chemical Engineering, The University of Texas at Austin, Austin, TX 78712

Shailesh Kumar
Dept. of Computer Sciences, The University of Texas at Austin, Austin, TX 78712

Dick Seemann
Fisher Rosemount Systems Inc., Austin, TX 78754

Wavelet representation of a signal is efficient for process data compression. An on-line compression algorithm based on Haar wavelets is proposed here. As a new data point arrives, the algorithm computes all the approximation coefficients and updates the multiresolution tree before it prepares to receive the next data point. An efficient bookkeeping and indexing scheme improves compression ratio more significantly than batch-mode wavelet compression. Reconstruction algorithms and historian format for this bookkeeping are developed. Various analytical results on the bounds on compression ratio and sum of the square error that can be achieved using this algorithm are derived. Experimental evaluation over two sets of plant data shows that wavelet compression is superior to conventional interpolative methods (such as boxcar, backward slope, and SLIM3) in terms of quality of compression measured both in time and frequency domain and that the proposed on-line wavelet compression algorithm performs better than the batch-mode wavelet compression algorithm due to the efficient indexing and bookkeeping scheme. The on-line algorithm combines the high quality of compression of wavelet-based methods and on-line implementation of interpolative compression algorithms at the same time.

Introduction

Chemical process data are a rich source of information and can be used to automate and improve the operations of a chemical plant. Due to recent improvements in sensors and faster sampling methods, advances in computer technology, and an increasing need for safe and efficient plant operation, there has been an explosion of measured process data that needs to be managed efficiently. Extensive storage of process histories and their subsequent retrieval for computer integrated operation of chemical processes, transfer of process data over the information highway for remote and autonomous operations, and collection and storage of real-time process data for quality management systems are some of the issues that need to be addressed to handle this explosion. Data compression has, therefore, become an important part of computer integrated operations of chemical processes.

Data compression is achieved by removing insignificant and redundant components of the signal and storing only the relevant ones. In other words, if the regularities in the data are captured, then a *compact representation* of these regularities could be stored instead of storing the raw data itself. This transformation of representation into a more compact form is usually associated with some loss of information. For most process signals, the high frequency, low amplitude components are typically noise and their loss is not only acceptable but is also desirable. This loss of information needs to be properly controlled and monitored. It is important that data compression techniques applied to chemical process data retain the transient and dynamic behavior present in the measured variable and provide acceptably accurate estimates of

Correspondence concerning this article should be addressed to S. J. Qin.

the raw signal at all relevant frequencies.

Some of the desirable properties of a good compression technique are as follows,

(1) Compressed data should require minimum storage. In other words, if N is the number of data units required to store (represent) the raw signal and $\hat{N}$ is the number of data units required to store (represent) the compressed signal (including the bookkeeping information), then the ratio $N/\hat{N}$ should be as high as possible.

(2) Clear and explicit measures to judge the quality of compression, in terms of how ''closely'' the raw signal is approximated by the signal reconstructed from its compressed representation, should be available. Three such measures of closeness namely, *root mean square error* (RMSE), *local point error* (LPE) and *transfer function estimate* (TFE), defined formally in the Experimental Results section, will be used in this work. With respect to these measures, compression algorithms should lead to compression such that the reconstructed signal is as close to the raw signal as possible for a given amount of compression.

(3) The compression technique should efficiently filter out undesirable ''noise'' from the raw signal and should capture all the ''desired features'' at the same time.

(4) Typically, the plant data are sampled at high frequencies. To be able to transfer them to the monitoring or storage site over a network and to store them in an on-line fashion, it is required that the time of compression and subsequent retrieval of the original signal from compressed signal should be fast.

These requirements are conflicting and depend on the *bias* of the compression technique as to which features or components of the signal it prefers to eliminate over others and by how much. The parameters of the compression technique need to be properly adjusted to meet these conflicting requirements and constraints.

Early data compression techniques were developed in the 1960s and 1970s (Ehrman, 1967) and were tested with real telemetry data (Benelli et al., 1980; Kortman, 1967). Hale and Sellars (1981) and Bader and Tucker (1987) demonstrated the use of real-time data compression in chemical engineering. The conventional data compression techniques include *interpolative techniques* like boxcar (Hale and Sellars, 1981; Bader and Tucker, 1987), backward slope (Hale and Sellars, 1981; Bader and Tucker, 1987), straight line interpolation methods (SLIM) (Kortman, 1967), piecewise linear on-line trending (PLOT) (Mah et al., 1995) and the swinging door method (Bristol, 1990). These methods produce piecewise linear interpolation of the data. Basically, the signal is represented as a straight line until a user-specified error criterion is violated. Then, a new linear segment is initiated to continue the approximation. Data compression is achieved since only the end points of the linear segments need to be stored. Other techniques like multivariate statistical methods (MacGregor, 1994) and vector quantization (Gray, 1984) have also been used for chemical process data compression.

The existing methods suffer from the intrinsic drawback that they are short-sighted in the sense that they use only local temporal information as the criteria for compression, and thus give suboptimal compression with respect to the root mean square error and the scale error criteria. The long-term frequency regularities are not captured for a given compression ratio; thus, information loss occurs. Process signals are mostly nonstationary, and they represent the cumulative effect of many underlying process phenomena, such as process dynamics, measurement noise, effects of external disturbances, effects of feedback control, each of which may manifest on a different scale. Wavelet decompositions, which result in representations on different time scales, seems a natural framework for analyzing a process signal (Chikkula and Lee, 1994). Using wavelets, the signal is first decomposed into fine and coarse details at multiple levels of resolution and then ignoring the fine details that are below certain threshold, compression is achieved (Bakshi and Stephanopoulos, 1996; Watson et al., 1996). For a batch-mode compression, Kudic and Thornhill (1996) have used discrete wavelets, while Watson et al. (1998) have compared transform compression (wavelet, discrete cosine, and Fourier) to vector quantization and piecewise linear compression techniques (boxcar, backward slope, and so on). Yamatake-Honeywell have developed a wavelet-based algorithm for the compression of control and sensor data such as temperature, pressure, liquid flow and liquid levels (Kazato and Hosoi, 1995).

It has been shown (Bakshi and Stephanopoulos, 1996; Watson et al., 1996, 1998) that wavelet-based compression is superior to the conventional techniques with respect to various performance indices. However, still, the interpolative methods, being more suitable for on-line compression, are widely used in the industry. In this article, we propose an on-line compression algorithm using Haar wavelets. It captures both the quality of compression property of wavelet-based compression and the practicability of an on-line compression algorithm. In this on-line compression algorithm, the multiresolution tree of coarse approximations is built, bottom up, as new data points arrive. An efficient bookkeeping scheme which makes use of the tree structure is proposed. Theoretical and empirical results on how the efficient indexing and bookkeeping improves compression ratio over batch mode compression is given. Furthermore, analytical results for the bounds on compression ratio and sum square error are obtained. These bounds can be used as guidelines for choosing thresholds.

A brief overview of wavelets is given and the generic batch mode compression algorithm using wavelets is introduced. The complete on-line compression algorithm based on Haar wavelets, is introduced together with the reconstruction algorithm, bookkeeping and historian format. The theoretical performance bounds on compression ratio and sum of square error are discussed. Experimental results on: (1) the comparison of the conventional techniques with batch mode wavelet-based compression; and (2) the comparison of the batch mode compression using Haar wavelets with the on-line compression developed in this work are presented. Key results and future directions conclude the article.

Batch-Mode Data Compression Using Wavelets

The theory of wavelet transforms is based on multiresolution analysis (Strang and Nguyen, 1996), which implies that the space of the finite energy square integrable functions $L^2(R)$ (or any Hilbert space of functions for that matter) can be decomposed into nested closed subspaces at multiple resolutions, that is, subspace at resolution $j-1$ (V_{j-1}) is contained in the subspace at resolution j (V_j)

$$\mathbf{0} \subseteq \ldots \subseteq V_{-1} \subseteq V_0 \subseteq V_1 \subseteq \ldots \subseteq V_j \subseteq V_{j+1} \subseteq \ldots \subseteq L^2(R) \quad (1)$$

The space V_j is defined by a set of orthonormal basis functions $\{\phi_{j,n}(x)\}_{n=-\infty}^{\infty}$. Any function in this space can be expressed as a linear combination of these bases. Now consider a signal $x(t)$ at some high resolution V_J [if it were a continuous signal with finite energy, V_J would be the same as $L^2(R)$]. This signal can be broken down into two components, one of which lying in the subspace V_{J-1}, denoted by $a_{J-1}(t)$, is called the *coarse approximation* and the other in $V_{J-1}^{\perp}$, denoted by $b_{J-1}(t)$, is called the *fine detail* of $x(t)$ [$\equiv a_J(t)$]. The subspace $V_j^{\perp}$, denoted by W_j, hereafter, is essentially the orthogonal subspace of V_j such that the relationship between the three subspaces V_j, W_{j-1} and V_{j-1} in the multiresolution framework is given by (Strang and Nguyen, 1996)

$$V_j = V_{j-1} \oplus W_{j-1} \quad (2)$$

Equation 2 states that any function in the subspace V_j can be written as a sum of two functions, one from the subspace V_{j-1} and the other from its orthogonal subspace W_{j-1}. More specifically, $a_J(t)$ $(\in V_J) = a_{J-1}(t)$ $(\in V_{J-1}) + b_{J-1}(t)$ $(\in W_{J-1})$. If $\{\psi_{j,n}(x)\}_{n=-\infty}^{\infty}$ denotes the complete orthonormal bases in W_j, then functions $a_{J-1}(t)$ and $b_{J-1}(t)$ can be written as

$$a_{J-1}(t) = \sum_n \hat{a}_{J-1,n} \phi_{J-1,n}(t) \quad (3)$$

$$b_{J-1}(t) = \sum_n \hat{b}_{J-1,n} \psi_{J-1,n}(t) \quad (4)$$

where $\hat{a}_{J-1,n}$ and $\hat{b}_{J-1,n}$ are the nth *wavelet coefficients* for the coarse approximation and the fine details part of the signal $x(t)$, respectively.

The simplest wavelet transform that one can think of is the Haar wavelet with filter length $L = 2$. The scaling function $\phi(t) = 1$ for $0 \le t < 1$, and 0 otherwise is a box function. The wavelet function $\psi(t) = 1$ for $0 \le t < 1/2$, and -1 for $1/2 < t \le 1$ is a square wave.

The process of obtaining wavelet coefficients $\hat{a}_{j-1,n}$ and $\hat{b}_{j-1,n}$ from the coefficients at higher resolution $\hat{a}_{j,m}$s, such that $\hat{a}_{J,k}$ represents the discrete raw signal $a_k = a_J(k\Delta t)$ obtained at the highest resolution J, is called analysis. For Haar wavelets, the analysis equations are

$$\hat{a}_{j-1,n} = \frac{1}{\sqrt{2}}\left(\hat{a}_{j,2n} + \hat{a}_{j,2n+1}\right) \quad (5)$$

$$\hat{b}_{j-1,n} = \frac{1}{\sqrt{2}}\left(\hat{a}_{j,2n} - \hat{a}_{j,2n+1}\right) \quad (6)$$

The process of generating the signal $a_k = a_J(k\Delta t)$ back from the coefficients at lower resolutions $\hat{a}_{j,n}$ and $\hat{b}_{j,n}$ is called synthesis. The Haar wavelet synthesis equations for obtaining coefficient $\hat{a}_{j,2n}$ and $\hat{a}_{j,2n+1}$ from coefficients $\hat{a}_{j-1,n}$ and $\hat{b}_{j-1,n}$, obtained by adding and subtracting the analysis Eqs. 5 and 6, are as follows

$$\hat{a}_{j,2n} = \frac{1}{\sqrt{2}}\left(\hat{a}_{j-1,n} + \hat{b}_{j-1,n}\right) \quad (7)$$

$$\hat{a}_{j,2n+1} = \frac{1}{\sqrt{2}}\left(\hat{a}_{j-1,n} - \hat{b}_{j-1,n}\right) \quad (8)$$

The multiresolution tree for Haar wavelets can be generated on-line as the signal a_k arrives, which will be discussed later.

Batch mode data compression using wavelets is a simple two step process. First, the raw signal $x = \{x_k\}_{k=0}^{N}$ is represented in terms of wavelet coefficients $\{c_k\}_{k=0}^{N'}$. This is done by taking the wavelet transform of the signal using a particular wavelet and level of decomposition. The number of coefficients generated in this transform is approximately the same as the number of data points ($N \approx N'$).

The second step is thresholding. Coefficients whose magnitude is less than a predefined threshold θ are set to zero and ignored, that is, *if* ($|c_k| < \theta$), *then* $\hat{c}_k = 0$; *otherwise* $\hat{c}_k = c_k$; $\forall k = 1, \ldots, N$. Additional bookkeeping information also needs to be stored along with the coefficients. If it takes $\rho(C)$ units of memory to store a coefficient and $\alpha\rho(C)$ units to store the corresponding bookkeeping information, then the compression ratio is given by

$$CR(\text{batch}) = \frac{N\rho(C)}{(N-M)(\rho(C))(1+\alpha)} = \frac{1}{\left(1 - \frac{M}{N}\right)(1+\alpha)} \quad (9)$$

where M is the number of coefficients ignored. The complete process of data compression using wavelets is given in Figure 1.

To reconstruct the signal $\{\hat{x}_k\}_{k=1}^{N}$ back from the thresholded coefficients $\{\hat{c}_k\}_{k=1}^{N'}$, an inverse wavelet transform using the same mother wavelet used for compression and at the same level of decomposition is used. It will be shown that with respect to the *sum of square error* (SSE) criterion of information loss, wavelet-based compression is optimal. In fact,

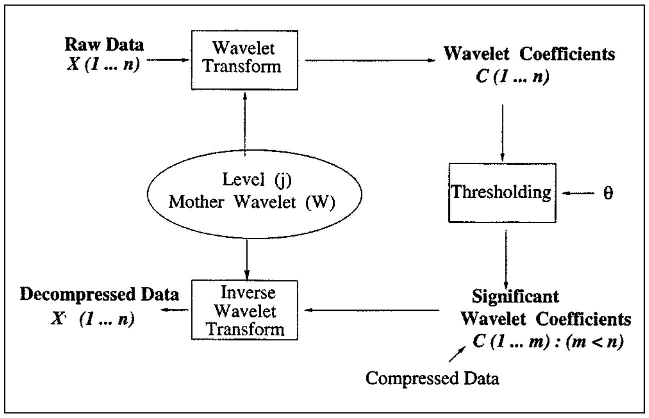


Figure 1. Data compression using wavelets: batch-mode.

the energy (information) lost in compression is the energy lost in the coefficients that are ignored; hence, the sum of square error is given by

$$SSE = \sum_{k=1}^{N} (x_k - \hat{x}_k)^2 = \sum_{k=1}^{N} I(c_k \neq \hat{c}_k) c_k^2, \qquad (10)$$

where $I(q) = 1$ if q is true and 0 otherwise. Equation 10 states that the sum of squared error of reconstruction is the same as the sum of square of all the coefficients that have been ignored.

A compression method should remove noise and preserve significant signal frequencies. Generally, noise is high frequency, low magnitude, and is additive in nature. When the wavelets decompose a noisy signal, the details (high frequency components) tend to include noise and the approximations represent the relevant (low) frequencies. Since the noise has low magnitude, the absolute values of the detailed coefficients are small. Hence, when thresholding is applied, these detail coefficients are set to zero and noise is automatically removed during compression using wavelets. If the threshold is chosen properly, the process of de-noising is accompanied by the process of compression.

On-Line Compression Algorithm

The on-line compression is based on the generation of a multiresolution tree. As new data points arrive, the multiresolution tree is continuously grown, and each detail coefficient is compared with a set threshold value. If detail coefficient is less than threshold, tree generation continues to the maximum resolution level possible. The moment a detail coefficient exceeds threshold value, tree generation is stopped, and the minimal set of coefficients together with the necessary bookkeeping information is sent to the historian and a new tree is started from the next incoming point. This section deals with the on-line tree generation as the discrete signal $a_n \equiv a_J(n\Delta t)$ arrives. This signal being the finest level of resolution, (say resolution J), we denote these coefficients as $a_{J,n}$ and they form the leaf nodes of the multiresolution tree under construction.

On-line tree generation

Consider the arrival of the first point at time step 0, $a_{J,0}$. Starting from the null tree, this point is added to the tree (Figure 2a). Now at time step 1, point $a_{J,1}$ arrives. Using the Haar wavelet analysis Eqs. 5 and 6, coefficients $a_{J-1,0}$, and $b_{J-1,0}$ are computed. If the magnitude of the detail coefficient $b_{J-1,0}$ is less than the threshold at this level, then it is ignored while the approximation coefficient $a_{J-1,0}$ is added to the tree (Figure 2b). The next point $a_{J,2}$ arriving at time step 2 is added to the tree and no further coefficients can be computed at this stage (Figure 2c). With the arrival of $a_{J,3}$, however, the coefficients $a_{J-1,1}$ and $b_{J-1,1}$ can be computed. Further, using the coefficients $a_{J-1,0}$ and $a_{J-1,1}$, the coefficients $a_{J-2,0}$ and $b_{J-2,0}$ are computed. The status of the tree after ignoring $b_{J-2,0}$ is shown in Figure 2d.

This process is continued until the magnitude of a detail coefficient is larger than threshold. When this occurs, tree generation is stopped and the tree is ready for dispatch. A minimal set of coefficients along with bookkeeping information is dispatched to the historian. The next subsection deals with practical on-line implementation issues of thresholding along with bookkeeping and historian format. A complete flow chart for the on-line compression algorithm is given in Figure 3.

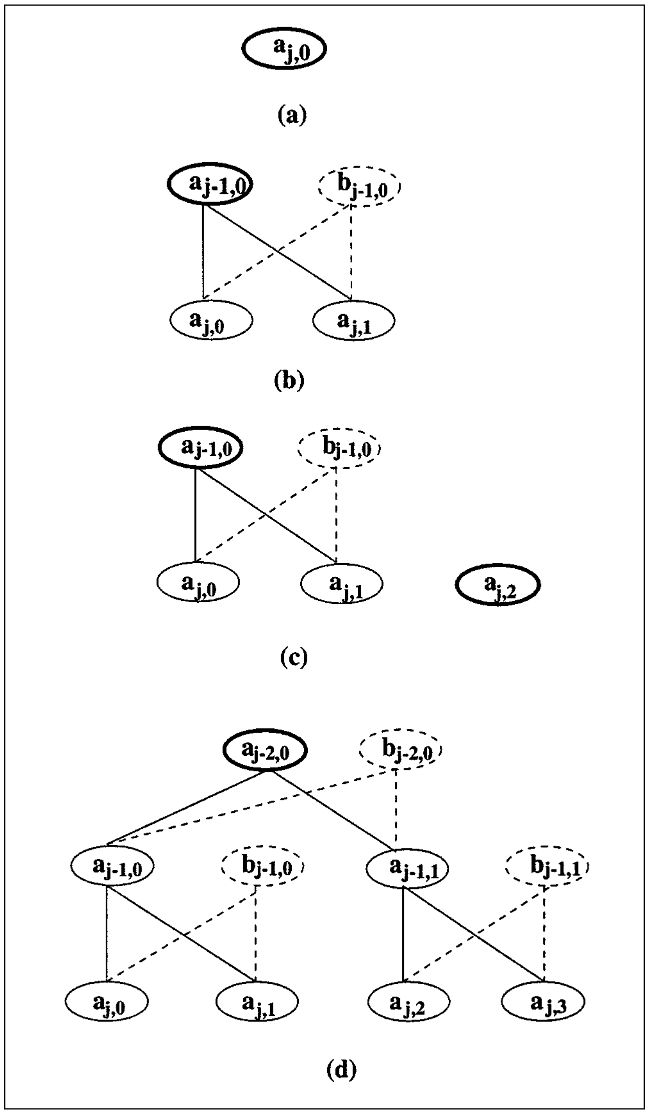


Figure 2. On-line tree generation

Tree is shown after (a) the first point has arrived, (b) the second point has arrived, (c) the third point has arrived, and (d) the fourth point has arrived. Solid circles are approximation coefficients, dashed circles are detail coefficients. Dark circle coefficients represent the minimal set of coefficients, **C**.

Thresholding and compression

Once it is clear how the tree is being generated, we see how the compression is achieved using this tree. As mentioned in the previous section, the basic mechanism of obtaining (lossy) compression is that some detail coefficients

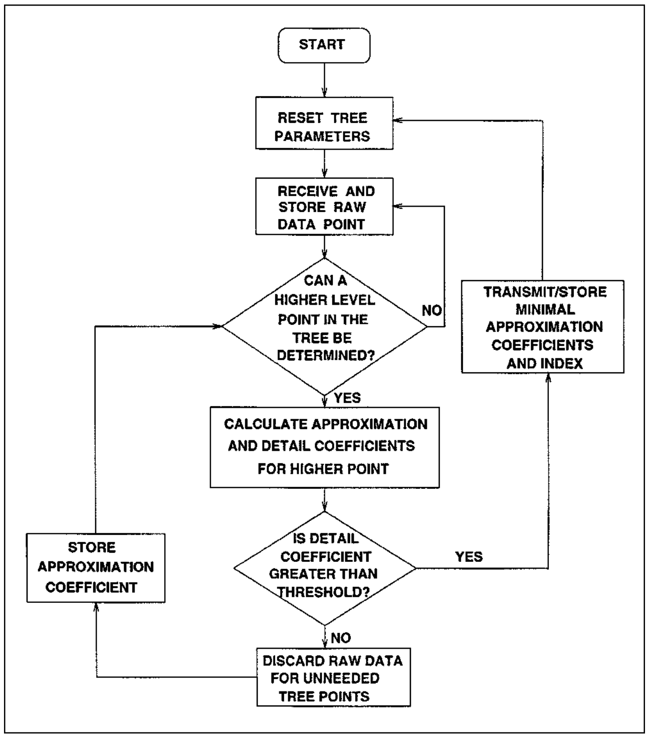

START
RESET TREE PARAMETERS
RECEIVE AND STORE RAW DATA POINT
CAN A HIGHER LEVEL POINT IN THE TREE BE DETERMINED?
NO
YES
TRANSMIT/STORE MINIMAL APPROXIMATION COEFFICIENTS AND INDEX
CALCULATE APPROXIMATION AND DETAIL COEFFICIENTS FOR HIGHER POINT
STORE APPROXIMATION COEFFICIENT
IS DETAIL COEFFICIENT GREATER THAN THRESHOLD?
YES
NO
DISCARD RAW DATA FOR UNNEEDED TREE POINTS

Figure 3. Flow chart for the on-line compression algorithm.

which are below certain threshold are ignored (set to 0) as they represent low magnitude components in the signal which are mostly noise. At each level, when a new approximation coefficient $a_{j,k}$ is computed, a corresponding detail coefficient $b_{j,k}$ is also computed. Now, if we assign the threshold of θ_j to level j, then the basic rule that decides whether to continue growing the tree or not can be stated as follows:

Rule: If the magnitude of the detail coefficient $b_{j,k}$ exceeds the threshold θ_j for the corresponding resolution j, then, the loss of information by ignoring this detail coefficient is "significant" and generation of the tree any further should be stopped.

At this point, the minimal set of coefficients necessary to reconstruct the portion of the signal used to generate this tree, along with the necessary bookkeeping information, is dispatched to the historian. The minimum set of coefficients,

Table 1. The Minimal Set, C, of Coefficients Which Completely Approximate All Coefficients Seen so far and the Corresponding Bookkeeping Information $B = (B(r), B(t))$.

Time	Minimal Set **C**	$[B(r), B(t)]$
0	$\{a_{J,0}\}$	$(J, 0)$
1	$\{a_{J-1,0}\}$	$(J-1, 0)$
2	$\{a_{J-1,0}, a_{J,2}\}$	$(J, 2)$
3	$\{a_{J-2,0}\}$	$(J-2, 0)$
4	$\{a_{J-2,0}, a_{J,4}\}$	$(J, 4)$
5	$\{a_{J-2,0}, a_{J-1,2}\}$	$(J-1, 2)$
6	$\{a_{J-2,0}, a_{J-1,2}, a_{J,6}\}$	$(J, 6)$
7	$\{a_{J-3,0}\}$	$(J-3, 0)$

bookkeeping information, and the historian format are discussed below.

Minimum set of coefficients, bookkeeping and historian format

At any stage in the tree generation process, a *minimal set of coefficients C* is maintained. This set contains only those coefficients which are necessary to reconstruct the portion of the signal used to generate the current tree. When a new coefficient (say $a_{j,k}$) is added to the tree at any resolution $j < J$, it contains relevant information in $a_{j+1,2k}$ and $a_{j+1,2k+1}$ that was used to generate this coefficient. Thus, the set **C** should now be modified to contain $a_{j,k}$ instead of $a_{j+1,2k}$ and $a_{j+1,2k+1}$. The bookkeeping information associated with the current set **C** is a pair of integers denoting the resolution level $B(r)$, and time index $B(t)$ of the *last* coefficient added to set **C**. Table 1 gives the minimal set of coefficients as new data points arrive. The second column in Table 1 gives the bookkeeping information that is maintained at different time steps.

The historian format for this algorithm is given in Figure 4. It is essentially a sequence of *Dispatch chunks* (DC_i) each of which contains bookkeeping information (B_i) which requires $\alpha\rho(C)$ units of memory, followed by a set of $(N - M)$ coefficients C_i in a specific order, which takes $(M - N)\rho(C)$ units of memory. The compression ratio for this dispatch chunk is given by

$$CR(\text{recurs}) = \frac{N\rho(C)}{\rho(C)[(N-M)+\alpha]} = \frac{1}{\left(1-\frac{M}{N}\right)+\frac{\alpha}{N}} \quad (11)$$

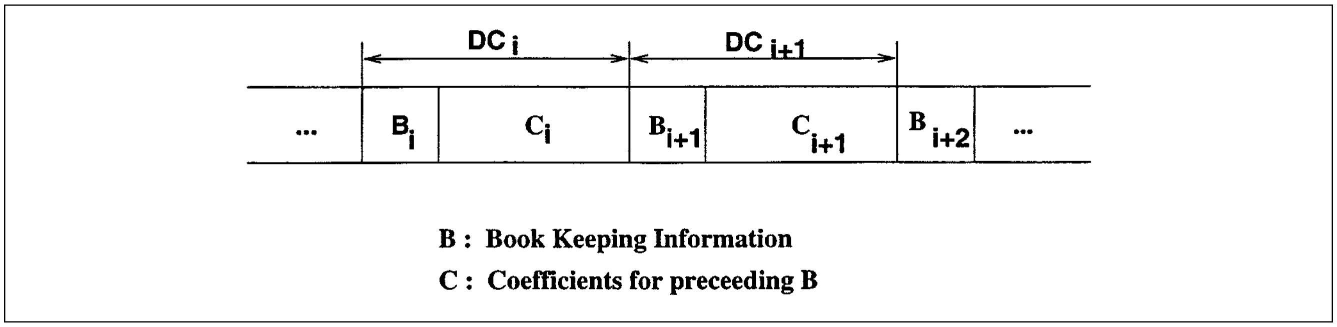

DC i
DC i+1
...
Bi
Ci
Bi+1
Ci+1
B i+2
...
B : Book Keeping Information
C : Coefficients for preceeding B

Figure 4. Historian format.

Historian is a collection of dispatch chunks each of which is the bookkeeping part B followed by coefficient part **C**.

Comparing the compression ratio of the on-line algorithm with that of the corresponding chunk with a batch mode approach (Eq. 9), we get

$$\frac{CR(\text{recurs})}{CR(\text{batch})} = \frac{(N-M)\rho(C)(1+\alpha)}{\rho(C)[(N-M)+\alpha]} = \frac{1+\alpha}{1+\dfrac{\alpha}{(N-M)}} > 1 \qquad (12)$$

Hence, the compression ratio for any dispatch chunk in on-line algorithm is larger than that in the corresponding chunk in batch mode. As $(N-M)$ becomes large, the difference in compression ratios becomes more significant. Empirical evidence for the gain in compression ratio for the same compression error is given in the subsection on batch-mode wavelets. It may be noted here that this comparison between the compression ratios of on-line and batch-mode formulations is on a per dispatch chunk basis.

Main differences between wavelet on-line and batch-mode formulations

The following remarks should be made regarding the difference between the on-line and batch compression approaches.

(1) The ultimate savings in on-line compression are due to the proposed bookkeeping strategy. For instance, with a small data set of 16 points, as shown in Figure 5a, if $b_{J-1,7}$, the detail coefficient corresponding to $a_{J,14}$ and $a_{J,15}$ exceeds the threshold value (shown as a filled black circle in Figures 5b and 5c), the on-line approach would require dispatch of $a_{J-3,0}$, $a_{J-2,2}$, $a_{J-1,6}$, $a_{J,14}$ and $a_{J,15}$ (shown as dark circle in Figure 5b), along with bookkeeping information for only $a_{J,15}$ (double dark circle in Figure 5b), resulting in a compression of ratio of $16/(5+1) = 2.67$ (assuming $\alpha = 1$). The batch mode approach, under similar situations, would require saving $a_{J-4,0}$, $b_{J-4,0}$, $b_{J-3,1}$, $b_{J-2,3}$, and $b_{J-1,7}$ along with bookkeeping information for each of these (shown in double dark circle in Figure 5c), giving a compression ratio of $16/(5+5) = 1.60$. About 67% more compression is achieved by on-line approach than the batch-mode approach. In the event of $b_{J-1,7}$ exceeding the threshold, it is very likely that the values of the coefficients related to $b_{J-1,7}$ would also be high. In other words, it is likely that $a_{J-4,0}$, $b_{J-4,0}$, $b_{J-3,1}$, and $b_{J-2,3}$ also exceed the threshold. However, depending on the type of signal and the threshold value, it is possible that in a best case scenario for batch-mode compression, only $a_{J-4,0}$, and $b_{J-1,7}$ exceed the threshold, thereby resulting in a compression ratio of $16/(2+2) = 4.0$.

For a larger data set, if the detail coefficient after 1,024 points exceeds the threshold value, the on-line approach would require storing only 11 approximation coefficients and one pair of bookkeeping information, giving a compression ratio of $1{,}024/(11+1) = 85.33$, whereas the batch-mode approach would give a compression ratio of $1{,}024/(11+11) = 46.54$. Here again the on-line compression ratio is about 83% over that of the batch-mode.

(2) The batch compression thresholds both detail and approximation coefficients, while the on-line compression thresholds the detail coefficients only. This is justified for the following reasons:

(a) Approximations typically represent the process information rather than noise. Therefore, it is desirable not to threshold the approximations at all.

(b) Approximations of the signal are typically low-frequency and large magnitude coefficients. Therefore, most of them are likely larger than threshold, that is, little compression is expected from thresholding the approximations.

(c) The proposed bookkeeping structure requires not to threshold the approximations.

(3) The batch compression proceeds to the next level even though a significant detail coefficient is encountered at the current level, while the on-line compression does not proceed to the next level. This treatment is justified because with a large detail coefficient at the current level, the coefficients at the next level are likely to be large as well (for example, due to a drastic change in signal).

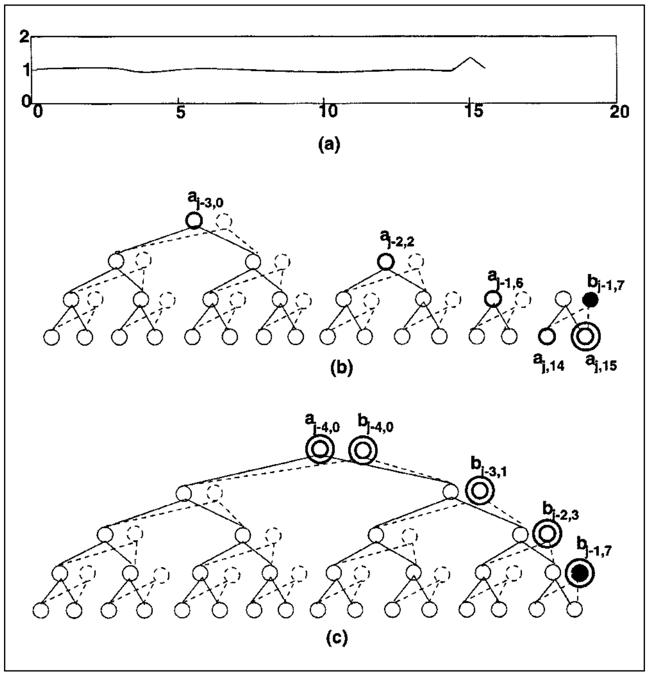


Figure 5. Comparison between wavelet on-line and wavelet batch-mode approaches.

(a) Original signal, (b) wavelet tree with on-line approach, and (c) wavelet tree with batch-mode approach. Solid circles are approximation coefficients; dashed circles are detail coefficients; dark circles are a minimal set of coefficients; double dark circles are bookkeeping information to be dispatched; solid dark circles are detail coefficients exceeding the threshold.

Complete on-line algorithm

Two types of indexes used in this algorithm are *the level index* (j) and for every level index a *time index* (k_j). Each level j has its own threshold θ_j. All indexes are initialized and updated such that the index k_j denotes the next coefficient time index that is to be added to the current tree at level j, that is, the tree already contains coefficients $a_{j,0}$, $a_{j,1}$, $\ldots$, a_{j,k_j-1} and the next coefficient that will be added at level j will be a_{j,k_j}. The basic algorithm starts with initializing all these coefficients $(k_J, k_{J-1}, \ldots, k_0)$ to 0. The complete algorithm is given below:

(1) **Initialize:**

- $k_j = 0$, for all $j = J, J-1, \ldots, 0$.
- $C = \{\}$ and $(B(r), B(t)) = (-1, -1)$ (some invalid values)

(2) **Receive next point** (a_{J,k_J}) and update set C and B's as follows

$$C = C + \{a_{J,k_J}\} \tag{13}$$

$$[B(r), B(t)] = (J, k_J) \tag{14}$$

(3) **Update tree** as much as possible starting from level $j = J-1$, that is, while $k_j - 1$ is divisible by 2, do the following:

(a) Compute the detail coefficient b_{j,k_j} using

$$b_{j,k_j} = \frac{1}{\sqrt{2}}(a_{j+1,2k_j} - a_{j+1,2k_j+1})$$
$$= \frac{1}{\sqrt{2}}(a_{j+1,k_{j+1}-1} - a_{j+1,k_{j+1}-2}) \tag{15}$$

(b) If $|b_{j,kj}| < \theta_j$, then

- Compute the approximation coefficient a_{j,k_j} using

$$a_{j,k_j} = \frac{1}{\sqrt{2}}(a_{j+1,2k_j} + a_{j+1,2k_j+1})$$
$$= \frac{1}{\sqrt{2}}(a_{j+1,k_{j+1}-1} + a_{j+1,k_{j+1}-2}) \tag{16}$$

- Update set C and B as follows

$$C = C + \{a_{j,k_j}\} - \{a_{j+1,k_{j+1}-1}, a_{j+1,k_{j+1}-1}\} \tag{17}$$

$$[B(r), B(t)] = (j, k_j) \tag{18}$$

- set $k_j = k_j + 1$ (increment index at level j)
- set $j = j - 1$ (move on to next lower resolution)

(c) If $|b_{j,k_j}| > \theta_j$ then

- DISPATCH (C, B)
- go to step (1) and start a new tree

(4) $k_J = k_J + 1$ and go to step 2.

Reconstruction

A reconstruction algorithm for reconstructing the signal $\hat{x}$ from the compressed data using the bookkeeping and coefficients information is given in this section. Each dispatch chunk is processed separately. Consider the ith dispatch chunk. The bookkeeping part B_i of this chunk gives the level and time index of the last coefficient in its C_i part. Let $j = B_i(r) = J - l$, where J is the predefined highest resolution at which the raw signal is obtained, and $k = B_i(t)$, resolution level and time index, respectively. The following algorithm computes all the $(k+1)2^l$ coefficients $\hat{x}_i = \{a_{J,i}\}$, $i = 0 \ldots (k+1)2^l - 1$, as follows:

Reconstruct (DC_i)

(1) READ $(j,k) = [B_i(r), B_i(t)]$; set $l = J - j$.

(2) SET $n = (k+1)2^l$; n denotes the number of data points to be generated.

(3) READ COEFFICIENT $a_{J-l,k}$; the next coefficient in C_i.

(4) COMPUTE 2^l coefficients as follows

$$a_{J,m} = \frac{1}{(\sqrt{2})^l} a_{J-l,k} \quad \forall m = k2^l \ldots (k+1)2^l - 1. \tag{19}$$

(5) UPDATE $n = n - 2^l$.

(6) COMPUTE unique (l,k) such that $n = k2^l$.

(7) IF $n > 0$ GOTO step 3 ELSE done.

Theoretical Bounds on Compression Ratio and Error

From the binary structure of the multiresolution tree generated and the properties of the on-line algorithm, properties and performance bounds on compression ratio and sum of square error are derived in this section. In the results that follow:

- n denotes the number of points arrived so far. (Note that the index of the nth point is $n-1$; first index is 0).
- $\gamma(n)$ denotes the number of levels the tree can be grown when the nth point (or the point with index $n-1$) arrives.
- $C_A(n)$ denotes the set of $(q+1)$ coefficients added to the tree when the nth point arrives, where n is uniquely represented as $n = p2^q$ (note that $q = \gamma(n)$ and $p = n/[2^{\gamma(n)}]$);

$$C_A(n) = \{a_{J-i,\kappa(p,q,i)}\} \quad (i = 0, 1, \ldots, q), \tag{21}$$

where

$$\kappa(p, q, i) = p2^{q-i}. \tag{22}$$

- $T^k(n)$ denotes the tree after nth point arrives and the tree is grown only till level $k < \gamma(n)$.
- $\tilde{T}(n)$ denotes the fully grown tree after nth point arrives [note that $\tilde{T}(n) = T^{\gamma(n)}(n)$].
- $\tilde{\lambda}_j(n)$ denotes the total number of coefficients at level j in $\tilde{T}(n)$

$$\tilde{\lambda}_j(n) = \begin{cases} n & j = J \\ \left\lfloor \dfrac{\tilde{\lambda}_{j+1}(n)}{2} \right\rfloor & \text{otherwise} \end{cases} \tag{23}$$

- $\tilde{\lambda}_j^k(n)$ denotes the total number of coefficients at level j in $T^k(n)$

$$\lambda_j^k(n) = \begin{cases} \tilde{\lambda}_j(n) & k \le j \\ \tilde{\lambda}_j(n-1) & k > j \end{cases} \tag{24}$$

- $\lambda^k(n)$ denotes the total number of coefficients in $T^k(n)$

$$\lambda^k(n) = \sum_{j \le J} \lambda_j^k(n) \tag{25}$$

- $\tilde{\pi}_j(n)$ denotes the number of significant coefficients at level j in $\tilde{T}(n)$

$$\tilde{\pi}_j(n) = \text{rem}\left(\tilde{\lambda}_j(n),2\right) \tag{26}$$

where rem$(x,\ y)$ = remainder in the integer division of x by y.

- $\pi_j^k(n)$ denotes the number of significant coefficients at level j in $T^k(n)$

$$\pi_j^k(n) = \begin{cases} \tilde{\pi}_j(n) & k < j \\ \tilde{\pi}_j(n-1)+1 & k = j \geq J-\gamma(n) \\ \tilde{\pi}_j(n-1) & k = j < J-\gamma(n) \\ \tilde{\pi}_j(n-1) & k > j \end{cases} \tag{27}$$

- $\pi^k(n)$ denotes the number of significant coefficients in $T^k(n)$

$$\pi^k(n) = \sum_{j \leq J} \pi_j^k(n) \tag{28}$$

- $\tilde{E}_j(n)$ denotes the upper bound on the mean-square error at level j in $\tilde{T}(n)$

$$\tilde{E}_j(n) = \frac{1}{n}\tilde{\lambda}_j(n)\theta_j^2 \tag{29}$$

- $E_j^k(n)$ denotes the upper bound on the mean-square error at level j in $T^k(n)$

$$E_j^k(n) = \frac{1}{n}\lambda_j^k(n)\theta_j^2 \tag{30}$$

- $E^k(n)$ denotes the upper bound on the mean-square error in $T^k(n)$

$$E^k(n) = \sum_{j<J} E_j^k(n) = \frac{1}{n}\sum_{j<J}\lambda_j^k(n)\theta_j^2 \tag{31}$$

- $C^k(n)$ denotes the compression ratio of tree $T^k(n)$

$$C^k(n) = \frac{n}{\pi^k(n)} \tag{32}$$

- $\tilde{C}(n)$ denotes the compression ratio of tree $\tilde{T}(n)$

$$\tilde{C}(n) = \frac{n}{\tilde{\pi}(n)} \tag{33}$$

(1) *Bounds on Compression Ratio.* The compression ratio after $n+1$ points is highest if the tree is fully grown, that is, for the tree $\tilde{T}(n)$. Equation 33 computes the upper bound which is $\tilde{C}(n)$. The lower bound on compression ratio is obtained when the tree is not grown at all after $n+1$ points, that is, for tree $T^J(n)$. Equation 32 gives the lower bound which is $C^J(n)$. Thus, the bounds on compression ratio after n points are given by

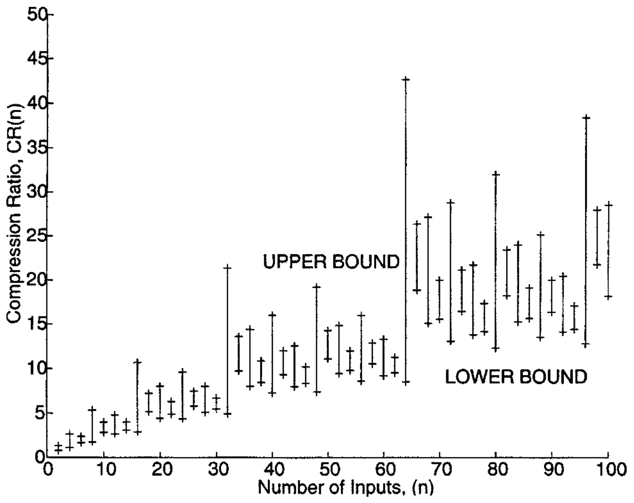


Figure 6. Bounds on compression ratio.

The lower bound is obtained when the tree is not grown at all after nth point arrives and the upper bound is obtained when the tree is grown full after nth point arrives.

$$C^J(n) = \frac{n}{\pi^J(n)} \leq C(n) \leq \tilde{C}(n) = \frac{n}{\tilde{\pi}(n)} \tag{34}$$

Figure 6 shows the bounds on the compression ratio for different number of input points.

(2) *Upper Bound on the Sum of Square Error.* Bakshi and Stephanopoulos (1996) have shown that the total sum of square error of reconstruction of a signal from only significant coefficients is essentially the sum of squares of the coefficients that are ignored in wavelet compression. Thus, mean-square error $e(n)$ accumulated in the recursion tree, after n points have been compressed, is the mean of squares of all the detail coefficients that have been ignored so far. Since any detail coefficient b_i^j at level j is less in magnitude than the threshold θ_j for level j, the upper bound on how much ignoring b_i^j will contribute to $e(n)$ is θ_j^2. Thus, the upper bound on the total mean-square error after n points and the tree is grown until level k is given by $E^k(n)$, in particular

$$e(n) \leq E^k(n) = \frac{1}{n}\sum_{j<J}\lambda_j^k(n)\theta_j^2 \tag{35}$$

Figure 7 shows the upper bounds on the mean-square error for different numbers of input points and different levels of tree growth. Figure 8 shows the upper bound on the mean-square error for different possible compression ratios that can be achieved using the on-line compression algorithm.

(3) *Time Complexity.* Time complexity of the on-line algorithm is crucial to the maximum sampling rate. The time between two successive points should be enough to allow processing of tree growth due to the last point. From the algorithm, it is evident that for the nth coefficient coming in the tree, the maximum number of coefficients that can be

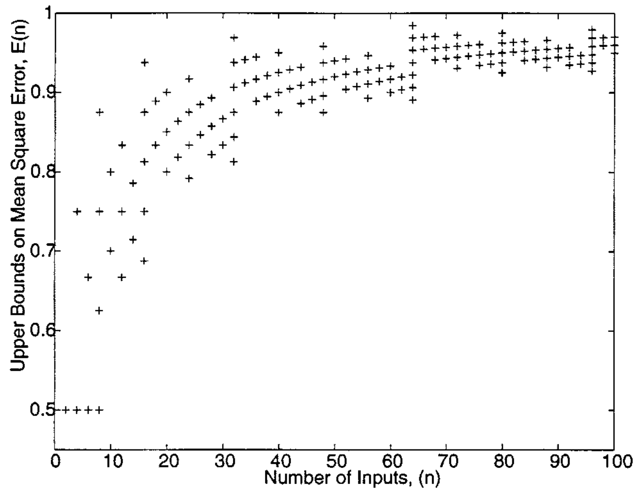


Figure 7. Bounds on mean-square error.

For each point that arrives, the tree can be grown to different levels. The kth + sign from the lowest for any n indicates the upper bound on the mean-square error that can be incurred if the tree is grown until level k after arrival of that point. Threshold is assumed 1. These bounds should be scaled by a factor of θ^2 where θ is the threshold at all levels.

computed is $\lfloor \log_2(n) \rfloor$; thus, the worst case time complexity is $O(\log_2(n))$ after the arrival of n points, which is reasonably low.

(4) *Space Complexity* (*Run-Time Memory Requirement*). This is important in the case where compression is to be done at the site of data generation before dispatching the data on the network to the storage or monitoring site. At any point of time in the tree generation process, the run-time memory required is just the minimal set of coefficients C and the bookkeeping pair B. The maximum size of the set C after n data points have been compressed in the tree is $\pi(n)$, which is again $O(\log_2(n))$. The size of B is constant. Hence, the run-time memory also does not exceed $O(\log_2(n))$.

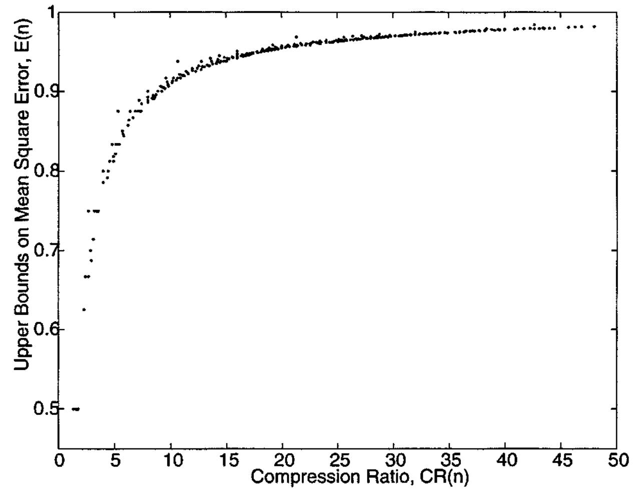


Figure 8. Bounds on mean-square error vs. compression ratio.

For all possible compression ratios that can be attained in the Haar recursive tree, the upper bounds on the mean-square error are shown here. The threshold is assumed 1. These bounds should be scaled by a factor of θ^2 where θ is the threshold at all levels.

Experimental Results

Two sets of experiments were conducted in this work. In the first set, the interpolative conventional techniques for compression, namely, boxcar, backward slope, and straight line interpolation method, are compared with the batch mode wavelet-based compression. In the second set of experiments, the batch mode compression using wavelets is compared with the on-line compression using Haar wavelets, the algorithm developed in this work. Before these two sets of experiments are discussed later in this section, the metrics on which these algorithms are compared are discussed next.

Performance metrics

Performance of a compression algorithm is measured in terms of how close the raw signal is to the reconstructed signal. Two performance metrics in time domain, namely, *root mean-square error* (also called the L^2 error) and *local point error* (also called the L^∞ error), and one performance metrics in frequency domain, namely, *transfer function estimate*, were used in experiments which compared batch mode wavelet compression with conventional interpolative compression. These performance metrics are defined below.

- **Root Mean Square Error:** If $\{x_k\}_{k=1}^n$ denotes the raw signal and $\{\hat{x}_k\}_{k=1}^n$ denotes the reconstructed signal, then the root mean-square error is defined as

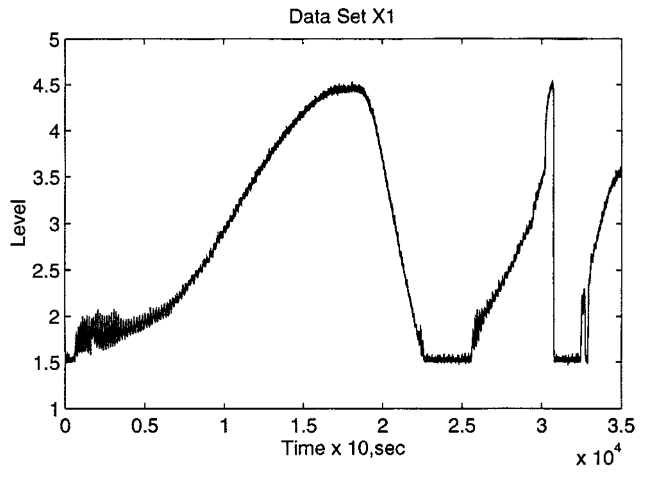
$$\text{RMSE} = \sqrt{\frac{\sum_{k=1}^n (x_k - \hat{x}_k)^2}{n}} = \sqrt{\frac{\text{SSE}}{n}} \tag{36}$$

where SSE is the sum-squared error, bounds on which are given in the previous section. Note that RMSE is averaged over the entire signal. Also, SSE is related to the ignored wavelet coefficients in wavelet compression as given in Eq. 23.

- **Local Point Error:** For the raw signal $\{x_k\}_{k=1}^n$ and the reconstructed signal $\{\hat{x}_k\}_{k=1}^n$, local point error is defined as the maximum deviation of any data point x_k in the raw signal from the corresponding point $\hat{x}_k$ in the reconstructed signal. More specifically

$$\text{LPE} = \max_k |x_k - \hat{x}_k| \tag{37}$$

- **Transfer Function Estimate:** The L^p norms described above are measures in the time domain. Frequency characteristics of chemical process signals are important as they contain dynamic information about the signal. A transfer function is a measure in the frequency domain which describes the frequency components of the signal that were retained during the compression and reconstruction of the signal. Although process data compression is a nonlinear operation due to thresholding, a transfer function can be used to approximately describe the nonlinear operation in the sense of a describing function. Considering the raw signal $\{x_k\}_{k=1}^n$ and the reconstructed signal $\{\hat{x}_k\}_{k=1}^n$ as the input and output



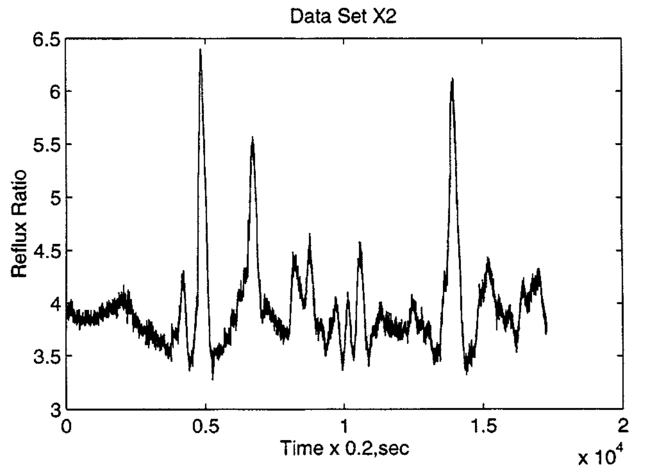


Figure 9. Data set $X1$: Level of a concentrator tank; data set $X2$: Reflux ratio of a distillation column.

of the compression/reconstruction process, the transfer function estimate (TFE) from x_k to $\hat{x}_k$ represents information loss in the frequency domain. It is defined as the ratio of the cross-power spectrum $P_{x\hat{x}}$ between the raw and reconstructed signal to the power spectrum P_{xx} of the raw signal.

$$\text{TFE}_{x\hat{x}}(\omega) = \frac{P_{x\hat{x}}(\omega)}{P_{xx}(\omega)} \tag{38}$$

A TFE close to 1 indicates no information loss at that frequency. A TFE significantly smaller than one indicates significant information loss.

Results

Figure 9 shows two sets of plant data which were used for comparison purposes. The first data set $X1$ is the level of a concentrator tank sampled at 0.1 s for almost an hour during normal plant operation. It has a mean of 2.73 and a variance of 0.99. $X2$, the second data set, which is the reflux ratio of a distillation column, was sampled at 5 s for 24 h, during normal plant operation. Its mean is 3.95 and variance 0.20.

For experiments with the conventional techniques (boxcar, backslope, SLIM3), the 35,000 data points of $X1$ were treated as one batch. Each one of these values was compared with a predefined threshold which was chosen so that all these techniques yield the same root mean-square error or local point error, as the case might be. The points were then rejected if they were less than the threshold, and stored if otherwise. For the batch-mode wavelet compression runs, the 35,000 points were taken as a batch and decomposed into approximation and detail coefficients using Haar mother wavelet, with level 10 as the decomposition level. The coefficients were then compared to a predefined threshold, such that the root mean-square error or the local point error is the same as that for the interpolative techniques. This was done to facilitate easy comparisons.

The Haar on-line algorithm treated each of the points from the 35,000 points data set as though they were coming on-line from a sensor and compared each calculated wavelet detail coefficient with a predefined threshold. It may be noted here that the results were generated using different thresholds so as to yield the same RMSE and LPE. Similar runs were conducted with the $X2$ data set which had 17,281 data points. All the batch-mode experiments were performed in Matlab on a pentium PC. The Haar on-line algorithm was implemented in C and experiments were performed on a workstation.

Conventional interpolative vs. batch mode wavelets-based compression

In the first set of experiments, the Haar wavelet-based compression method was compared with boxcar, backslope, and SLIM3 interpolative techniques. The results for the different performance indices are shown in Figures 10 to 12. The results indicate that wavelet-based data compression methods outperform the interpolative-based methods. Some results with the time domain indices like RMSE and LPE were also reported by Bakshi and Stephanopoulos (1996) and Watson et al. (1998). This section also explains the reasons why wavelet-based data compression methods outperform interpolative based methods.

Figure 10 shows the root mean-square error vs. compression ratio for the data sets $X1$ and $X2$. For all compression algorithms, as the compression ratio increases, loss of information also increases and, hence, the root mean-square error increases. A desirable feature is high compression ratio with low root mean-square error.

Wavelet-based compression is optimal with respect to the root mean-square error measure of loss of information. As mentioned before, the root mean-square error is related to the sum of square of all the detail coefficients ignored in the wavelet-based compression. Since in wavelet-based compression, only the smallest details which are below a certain threshold are ignored, the root mean-square error is kept to a minimum. Wavelets compression relies on the fact that given an orthonormal family of bases, the signal can be uniquely represented as a linear combination of these bases from multiple resolutions. As a result of this, the information content in the signal is now distributed among the wavelet coefficients. The higher the magnitude of a wavelet coefficient is, the more informative it is. In other words, wavelets is a suit-

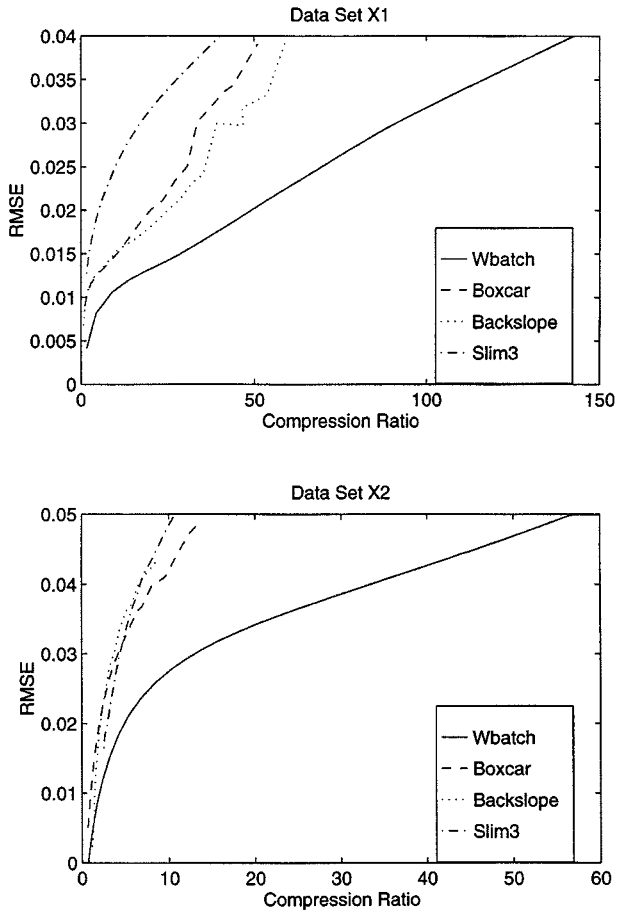


Figure 10. Root mean-square error vs. compression ratio for data sets $X1$ and $X2$.

Comparison of wavelet batch mode (Wbatch) with interpolative techniques.

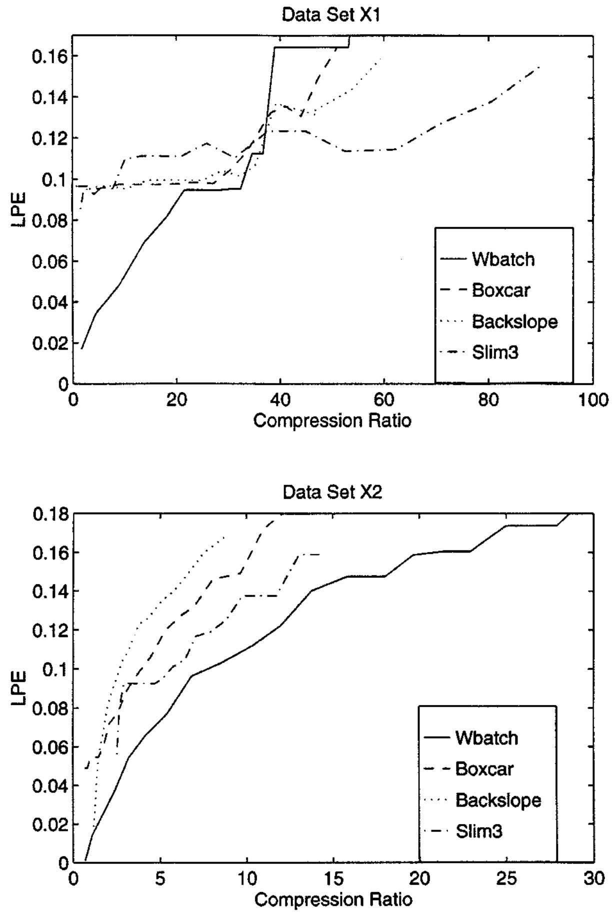


Figure 11. Local point error vs. compression ratio for data sets $X1$ and $X2$.

Comparison of wavelet batch mode (Wbatch) with interpolative techniques.

able way of representing information in a minimum number of coefficients or data units. Due to this efficient signal representation, the loss of information can be kept at a minimum by eliminating the smallest coefficients.

Interpolative methods, on the other hand, work in time domain only and this domain is not necessarily the most efficient representation of the signal. Since interpolative methods rely on ignoring the actual data points and try to interpolate these data points using zeroth-(boxcar) or first-(backslope) order interpolation on the data points they store, the information loss is not handled optimally. Thus, the root mean-square error in interpolative methods is significantly higher than that in wavelet-based compression for a given compression ratio.

Figure 11 shows the local point errors for the two data sets vs. compression ratio. Again, the wavelet-based compression outperforms the interpolative methods as they give a higher compression ratio for a given local point error.

Figure 12 is a plot of the transfer function estimate (TFE) vs. frequency for two different compression ratios of 2 and 50 for both data sets. It is desired that TFE be as close to 1 as possible, which is the case with wavelet-based batch mode compression for both the data sets. This shows that the reconstructed signal has captured almost all the features in the input signal. For the interpolative techniques, the TFE is significantly smaller than that of the wavelet compression (with the exception of the SLIM3 method for data set $X2$ at a compression ratio of 2). Hence, the reconstructed signal from interpolative methods tends to lose more process features, which is undesirable. At the lower compression ratio of 2, these trends are less pronounced, as the thresholds for low compression ratio is low, resulting in less loss in the reconstructed signal. For the high compression of 50, the losses are more due to higher thresholds, which is expected.

The difference between wavelet-based compression and interpolative compression shows the bias of these two classes of compression algorithms towards which *frequency components* they prefer to remove first in a lossy compression. It is desirable that high frequency noise should be removed before the low frequencies are affected. For wavelet-based methods, the information loss is small for low frequencies. That is, it preserves the frequency characteristics of the signal for low frequencies. Loss is more towards the higher frequency region. The reason for this behavior is the fact that the wavelet coefficients in some sense relate to the frequency of the signal. The approximation coefficients represent the low frequency

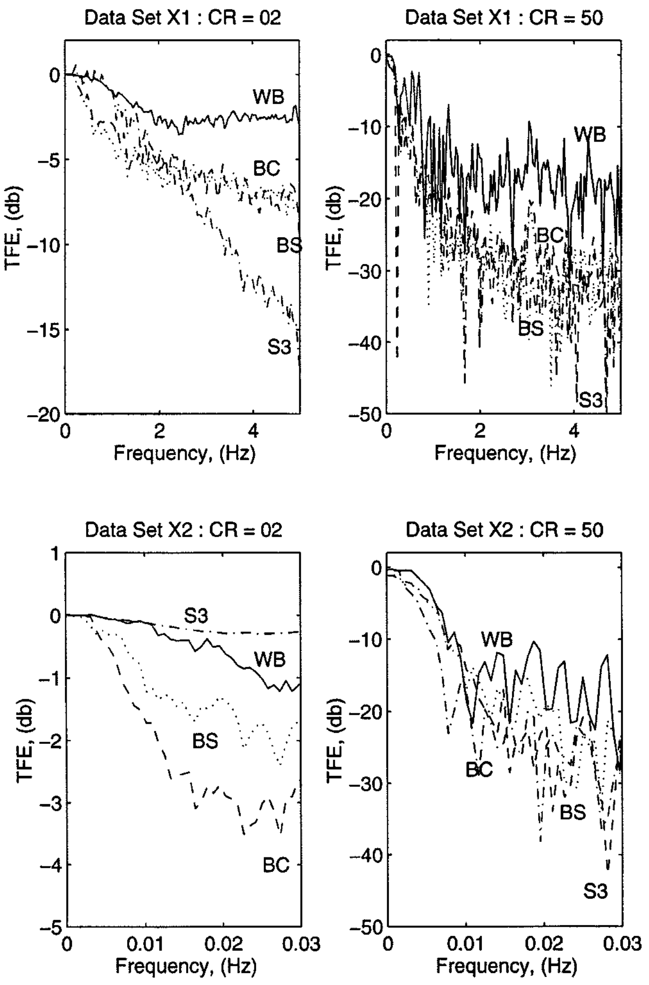


Figure 12. Transfer function estimate for data sets $X1$ and $X2$ at compression ratios of 02 and 50.

WB = Wavelet Batch; BC = Boxcar; BS = Backslope; S3 = Slim3.

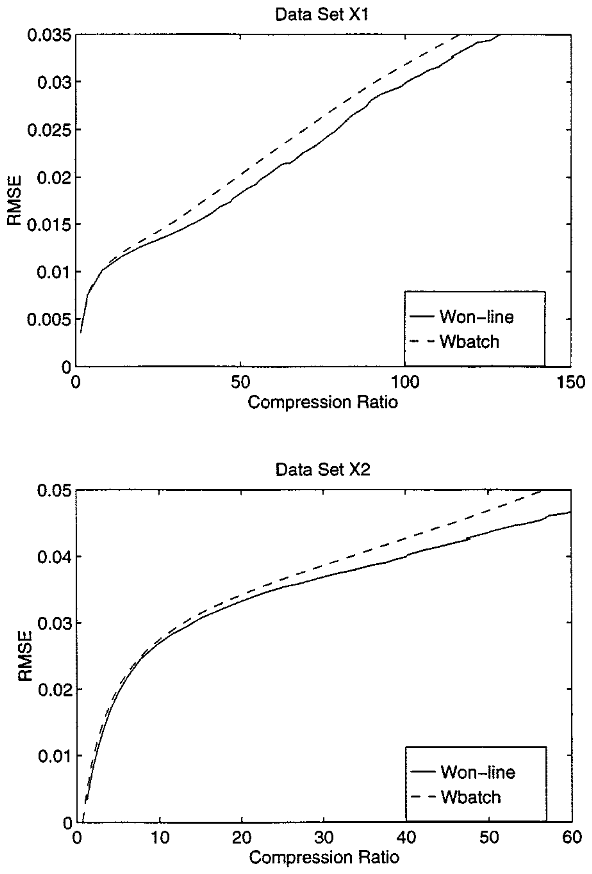


Figure 13. Root mean-square error vs. compression ratio for data sets $X1$ and $X2$.

Comparison of wavelet on-line (Won-line) technique with wavelet batch mode (Wbatch) technique.

components and the detail coefficients represent the high frequency components. Thus, wavelet representation in terms of these approximation and detail coefficients breaks the signal into different frequency components. In general the detail coefficients are smaller in magnitude than the approximation coefficients. Therefore, it is the detail coefficients that are chosen to be ignored during the thresholding in wavelet compression. As a result, only the high frequency components represented by these ignored details are lost during compression. In interpolative methods, on other hand, there is no distinction between the low and high frequency components in the signal since the signal is dealt with in time domain only. As a result, the interpolative methods do not show any specific bias towards removing high frequency components over low frequency components and all frequencies are equally likely to be affected.

From these results, it can be safely concluded that wavelet compression is superior to interpolative methods as far as quality of compression is concerned. For the same compression ratio, the raw signal is closer to the reconstructed signal both in time domain and frequency domain for wavelet compression than it is for interpolative methods.

Batch-mode wavelets vs. on-line wavelet-based compression

Batch mode compression using Haar wavelets is compared with on-line compression using Haar wavelets in this section. In batch mode compression, the time and scale indexes of each significant coefficient are stored along with the significant wavelet coefficients. In on-line compression, the time and scale indexes of only the last coefficient added to the tree are stored. As a result of this efficient indexing scheme, even for the same number of wavelet coefficients stored, the compression ratio of the on-line compression is higher than that of batch mode compression, as shown in Eq. 24.

Figure 13 shows the root mean-square error of batch mode compression vs. on-line compression for different compression ratios. For both the data sets, the performance of on-line compression is slightly better than that of batch mode compression. That is, the root mean-square error for the on-line compression is slightly smaller than that for batch mode for the same compression ratio.

This can be attributed to efficient bookkeeping used in on-line compression. The basic difference between the way batch mode and on-line compression works is that in batch mode some of the detail coefficients, which are larger than the threshold in magnitude, are stored and others are ignored, while in the on-line algorithm, all the details are ignored until a detail whose magnitude is greater than the threshold is encountered. When such a detail is encountered, the tree obtained so far is dispatched and a new tree is started afresh. As a result, if the thresholds for on-line compression is kept too low, chances of obtaining a detail more than the threshold early on will increase. This will lead to a large number of dispatches each with only a small number of coefficients. Thus, the performance of both batch mode and on-line are comparable at low compression ratios which in turn are obtained as a result of low thresholds.

When the thresholds are high, that is, for higher compression ratios, the number of dispatches is low. For each dispatch, there is only one pair of bookkeeping indices that is sent along with the dispatch. Thus, the larger the dispatch chunk, the more is saved in bookkeeping and, therefore, the differences between compression ratios in batch mode and on-line become more significant at higher thresholds.

Figure 14 is the local point errors of batch mode and on-line compression using Haar wavelets for both data sets. The values of local point error remain the same, but the curve seems to be shifted a little towards the right for the wavelet on-line technique. This is again because of the gain in compression ratio due to efficient bookkeeping.

Transfer function estimates of both data sets at compression ratios of 2 and 50 are plotted in Figure 15. Wavelet on-line compression technique does well to preserve the process features and the only loss is high frequency noise. Wavelet batch-mode compression removes the high frequency noise as well as some relevant process information, and thus performs suboptimally with respect to the on-line version. This effect is more pronounced at low compression ratios. High compression ratios are achieved with high thresholds, and, thus, both the wavelet-based methods suffer almost an equal amount of information loss and their TFE plots are almost the same for both the data sets, with the wavelet on-line slightly outperforming the wavelet batch-mode method.

In the end, a sample comparison between the original and compressed data is shown in Figure 16. For the same compression ratio, a sample from each of the data sets $X1$ and $X2$ is plotted along with the reconstructed data compressed by wavelet on-line and wavelet batch-mode methods. For both data sets, data compressed by the wavelet on-line method follows the original data more closely than that compressed by the batch-mode method, once again testifying that wavelet on-line approach outperforms the wavelet batch-mode approach.

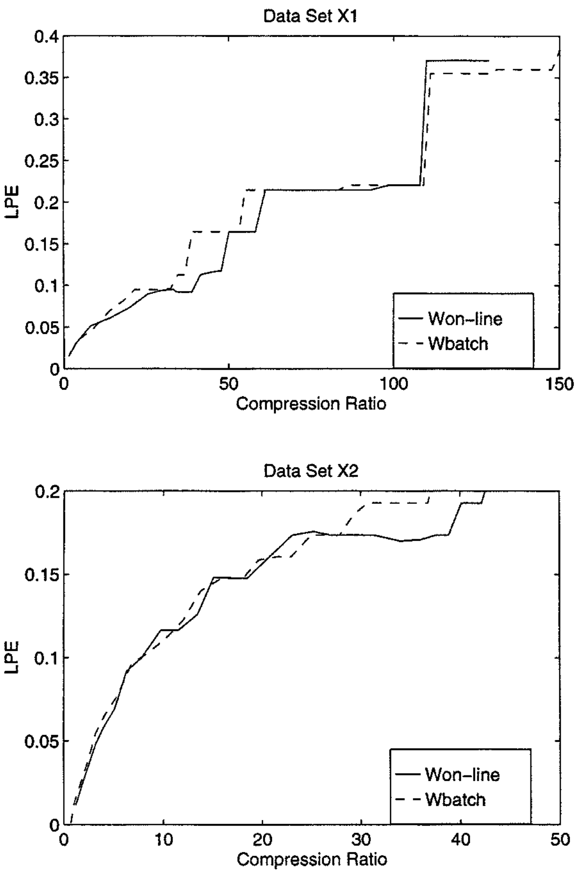


Figure 14. Local point error vs. compression ratio for data sets $X1$ and $X2$.

Comparison of wavelet on-line (Won-line) with wavelet batch mode (Wbatch) technique.

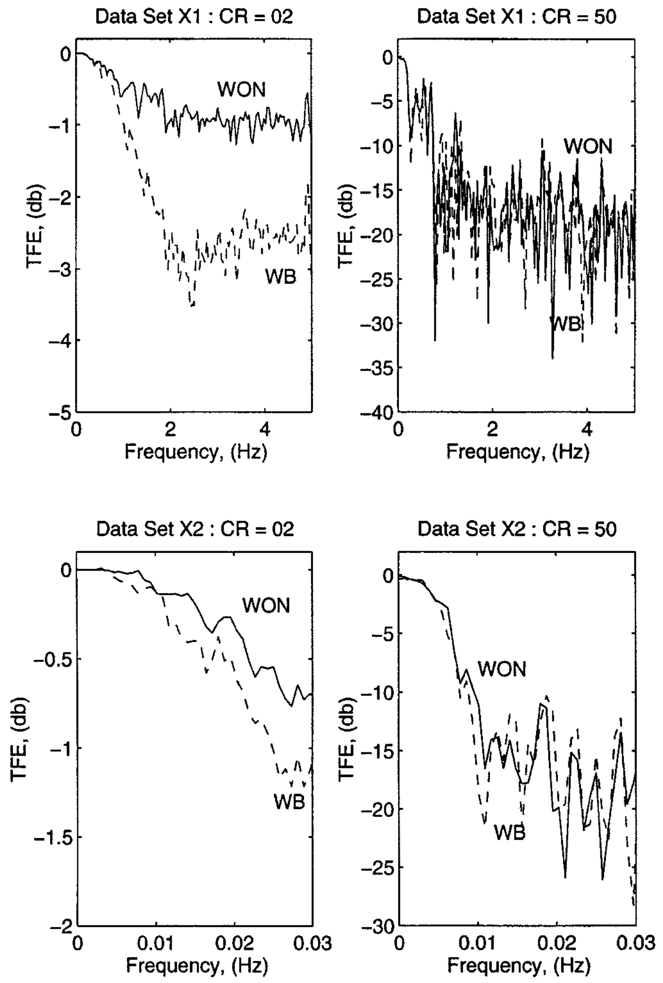
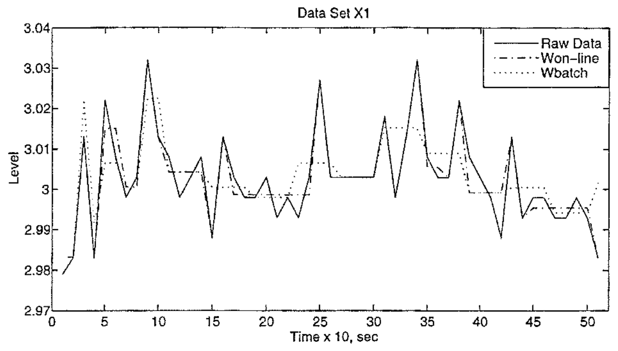


Figure 15. Transfer function estimate vs. compression ratio for data sets $X1$ and $X2$.

Comparison of wavelet on-line (WON) with wavelet batch mode (WB) technique.



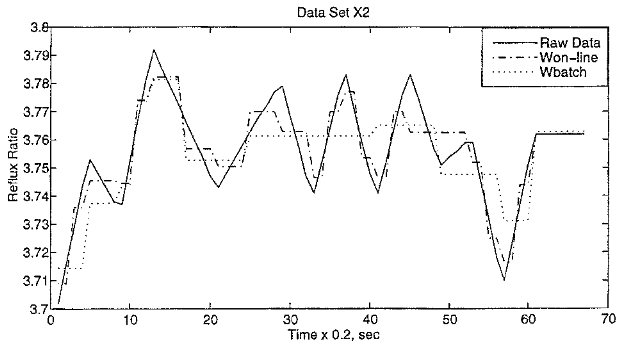


Figure 16. Comparison of wavelet on-line with wavelet batch mode technique.

Samples from original data sets $X1$ and $X2$ and compressed data sets.

Conclusions

A novel recursive on-line compression algorithm using Haar wavelets is proposed in this work. An efficient bookkeeping scheme which leads to significant gains in compression ratio over batch mode compression is also proposed. The reconstruction algorithm using this bookkeeping and the historian format is developed with easy on-line implementation features. Various analytical results on the bounds on compression ratio and sum of square error have been derived. These bounds form guidelines for choosing the compression parameters such as the thresholds. Experimental evaluation over two sets of plant data show that batch mode compression is superior to conventional interpolative methods in terms of the quality of compression measured both in time and frequency domain. On-line compression using wavelets shows improved performance over batch mode compression because of the efficient bookkeeping. The on-line algorithm proposed for compression captures the high quality of compression using wavelets, and the ability to implement it in an on-line fashion, making them as practicable as interpolative methods. Extension of the on-line wavelet-based compression to other families of wavelets is straightforward and will be studied in the future.

Notation

$B(r)$, $B(t)$	= bookkeeping information for resolution level j and time index t
c_k, $\hat{c}_k$	= kth wavelet coefficient in raw and threshold data set
$C(n)$	= compression ratio of tree $T(n)$
J	= index of the finest subspace
k	= time index for on-line incoming data set
V_j	= subspace at resolution j
W_j	= orthogonal subspace of V_j, at resolution j
x_k, $\hat{x}_k$	= kth sample point in raw and reconstructed data set
α	= parameter for units of memory used
$\phi(t)$	= scaling function (father wavelet)
$\psi(t)$	= wavelet function (mother wavelet)
ω	= frequency

Literature Cited

Bader, F. P., and T. W. Tucker, "Real-Time Data Compression Improves Plant Performance Assessment," *InTech*, **53** (1987).

Bakshi, B. R., and G. Stephanopoulos, "Compression of Chemical Process Data by Functional Approximation and Feature Extraction," *AIChE J.*, **42**, 477 (1996).

Benelli, D., V. Cappellini, and F. Lotti, "Data Compression Techniques and Applications," *The Radio and Electronic Engr.*, **50**, 29 (1980).

Bristol, E. H., "Swinging Door Trending: Adaptive Trend Recording?," *ISA Conf. Proc.*, (1990).

Chikkula, Y., and J. H. Lee, "Application of Wavelets in Process Control," *Wavelet Applications in Chemical Engineering*, R. L. Motard and B. Joseph, eds., Kluwer Academic Press Wellesley, MA (1994).

Ehrman, J., "Analysis of Some Redundancy Removal Bandwidth Compression Techniques," *Proc. IEEE*, **55**, 278 (1967).

Gray, R. M., "Vector Quantization," *IEEE ASSP Magazine*, **4** (1984).

Hale, J. C., and H. L. Sellars, "Historical Data Recording for Process Computers," *Chem. Eng. Prog.*, **38**, (Nov. 1981).

Kazato, H., and T. Hosoi, "Data Compression for Measured Variable by Wavelet Transform," *Proc. Digital Signal Process. Symp.*, IEICE (Nov. 1, 1995).

Kortman, M., "Redundancy Reduction—a Practical Method of Data Compression," *Proc. IEEE*, **53**(3), 253 (1967).

Kudic, A., and N. Thornhill, "Data Compression for Process Monitoring," *AIChE Symp. Ser. No. 312*, **92**, CACHE, AIChE (1996).

MacGregor, J. F., "Statistical Process Control of Multivariate Processes," *IFAC ADCHEM*, Kyoto, Japan (1994).

Mah, R. S. H., A. C. Tamhane, S. H. Tung, and A. N. Patel, "Process Treading with Piecewise Linear Smoothing," *Comp. Chem. Eng.*, **19**, 129 (1995).

Strang, G., and T. Nguyen, *Wavelets and Filter Banks*, Wellesley-Cambridge Press, Wellesley, MA (1996).

Watson, M. J., A. Liakopoulos, D. Brzakovic, and C. Georgakis, "Wavelet Techniques in the Compression of Process Data," *Proc. American Control Conf.*, Seattle, WA (1996).

Watson, M. J., A. Liakopoulos, D. Brzakovic, and C. Georgakis, "A Practical Assessment of Process Data Compression Techniques," *Ind. Eng. Chem. Res.*, **37**, 267 (1998).

Manuscript received Apr. 27, 1998, and revision received Apr. 30, 1999.