Effect of Data Compression on Pattern Matching in Historical Data

Ashish Singhal*

Controls Research Group, Johnson Controls, Inc., 507 East Michigan Street, Milwaukee, Wisconsin 53202

Dale E. Seborg†

Department of Chemical Engineering, University of California, Santa Barbara, California 93106

It is a common practice in the process industry to compress process data before it is archived. However, compression may alter the original data in a manner that makes extracting useful information from it more difficult. In this paper, popular data compression methods and their effect on pattern matching in historical data are evaluated. Pattern matching is performed using principal-component analysis based similarity factors. Simulation results indicate that wavelet-based compression provides the best compression for pattern matching, while compression using OSI PI software produces the best reconstruction of data.

1. Introduction

Because of advances in information technology, large amounts of data produced by industrial plants are recorded as frequently as every 1 s using data historians. With data storage media being relatively inexpensive, the cost of storing data has dropped sharply. The mechanical and electronic aspects of data acquisition, organization, and storage are well developed. However, with rapid globalization and decentralization, many companies require data to be made available at different sites around the world. The focus for data historian software is not only how to record data in an efficient and reliable manner but also how to make it available on demand in a timely fashion to a wide variety of personnel at different locations. To speed up transfer of relevant data across data networks and via the Internet, it is beneficial to transmit data in a compressed form. Data compression also is becoming relevant for control networks with wireless sensors that transmit data intermittently to extend battery life.

An efficient data compression method must maximize the degree of compression while retaining key features of the original data. There have been extensive studies performed in the areas of image compression and acoustic signal compression, but there are only a few relevant publications for compression of process data. A recent paper by Thornhill et al. described the detrimental effect of popular compression methods on basic data statistics such as the average, variance, and spectral density.

One of the earliest papers on compression of process data was published by Hale and Sellars. They provided an excellent overview of the issues in the compression of process data and also described piecewise linear compression methods that were being employed at the DuPont company. By compressing data, they were able to increase computer capability and satisfy the demand for process information by operators and engineers. Data compression thus permitted ever-increasing stored data to remain accessible with short response times.

Kennedy and Bascur and Kennedy have extended the ideas proposed by Hale and Sellars by developing information systems that integrate data recording as well as efficient retrieval and presentation to aid operations management, maintenance, crisis management, warehousing, engineering, business, and scheduling, among others.

Other researchers have developed several algorithms to compress time-varying signals in efficient ways. Bristol modified the piecewise linear compression methods of Hale and Sellars to provide a swinging-door data compression algorithm. Mah et al. proposed a complex piecewise linear online trending algorithm that performed better than the classical boxcar, backward-slope, and swinging-door methods. Although their algorithm could adapt to process variability and noise, it was slower than the swinging-door method.

Bakshi and Stephanopoulos published a series of papers on representing trends and compressing process data using wavelet methods. Their methods were batch methods because a batch of data was required before it could be compressed. Recently, Misra et al. developed an online data compression method using wavelets, where the algorithm computes and updates the wavelet decomposition tree before receiving the next data point. Their approach achieved better compression than batch methods.

In this paper, we evaluate different data compression methods in terms of pattern matching. The data compression methods are evaluated on the basis of not only how accurately they represent process data but also how they effect the identification of similar patterns from historical data. For example, a particular method may compress data so that the compressed data require very little storage while accurately representing the original process data. However, it may transform the original data in a way that is detrimental to methods that attempt to extract useful information from it, such as the pattern-matching methodology presented by Singhal and Seborg.

Preliminary results on this research topic were presented previously. This paper provides a detailed description of the different compression methods and their effects on pattern matching.

* To whom correspondence should be addressed. Tel.: (414) 524-4688. Fax: (414) 524-5810. E-mail: Ashish.Singhal@jci.com.
† E-mail: seborg@engineering.ucsb.edu.
2. Popular Data Compression and Reconstruction Methods for Time-Series Data

This section describes some of the popular compression methods for time-series data. Because the accuracy of retrieved data depends not only on the method that was used for compression but also on the method that was used for reconstruction, some simple reconstruction techniques such as zero-order hold and linear interpolation are also discussed briefly.

2.1. Data Compression Methods. The popular boxcar piecewise linear compression method is considered in this paper. Boxcar is also sometimes referred to as a “change-of-value” compression method and is illustrated in Figure 1. It is summarized as algorithm 1. Another common data compression method used in the process industries, data averaging, is described. Wavelet-based compression, a new method that is gaining popularity, is discussed in section 2.1.2. Finally, data compression performed by the commercially available Plant Information, or PI software, is described briefly in section 2.1.3.

Algorithm 1. Boxcar compression algorithm.

If \( V \) is the current value of the process variable, \( V_R \) is the most recent recorded value, \( V_L \) is the last value processed, and \( H \) is the recording limit, then do the following:
1. Calculate \( |V - V_R| \).
2. If \( |V - V_R| \geq H \), then record the previous value processed, \( V_L \), not the current value \( V \), which caused the triggering of the recording.
3. Recording is accomplished by setting \( V_R = V_L \).

2.1.1. Data-Averaging Compression. A common compression technique for time-series data is to simply average the data over a specified period of time. In this case, the compression is performed offline, rather than online, because a batch of data must be collected before it can be averaged. For example, if the averaging period is 2 min while the sampling period is 5 s, then 24 samples need to be collected for averaging before the next compressed data point is recorded.

2.1.2. Wavelet-Based Compression. The development of wavelets has received considerable interest during the past 15 years. Wavelet analysis is an emerging field of mathematics that has provided new methods and algorithms suited for the types of problems encountered in process monitoring and control. The wavelet transform is a method that divides data, functions, or operators into different frequency components and then processes each component with a resolution matched to its scale. The wavelet transform of a time-varying signal depends on two variables: scale (frequency) and time. Wavelets provide a tool for time-frequency localization. This time-frequency representation is of particular interest in analyzing time-varying patterns that are characteristic of process data.

A time-varying signal \( f(t) \) can be represented through the discrete wavelet transformation as follows:

\[
f(t) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} d(k,l) 2^{-k/2}(t-l)
\]

where the integer \( k \) represents the frequency scale at which the signal is decomposed and \( l \) represents its position in time. The wavelet decomposition coefficients, \( d(k,l) \), describe the frequency content of the signal \( f(t) \) at different times. The function \( \psi(t) \) is defined as the mother wavelet function. The most common mother wavelet functions used in the literature are the Haar function and the Daubechies family of orthogonal wavelet functions. The Haar wavelet function is used in this paper and is defined as

\[
\psi(t) = \begin{cases} 
1 & 0 \leq t < 1/2 \\
-1 & 1/2 \leq t < 1 \\
0 & \text{otherwise}
\end{cases}
\]

Wavelet transforms can be used to compress time-series data by thresholding the wavelet coefficients \( d(k,l) \). Thresholding is a method by which wavelet coefficients that are greater than a specified threshold are retained, while those smaller than the threshold are discarded. This type of thresholding is called hard thresholding. Another type of thresholding, known as soft thresholding, scales the wavelet coefficients that are smaller than the specified threshold. Hard thresholding is used in this paper.

If \( d(k,l) \) is the wavelet coefficient at frequency scale \( k \) and at time location \( l \) and \( \phi \) is the wavelet threshold, then the thresholded wavelet coefficients \( d^\phi(k,l) \) are defined as

\[
d^\phi(k,l) = \begin{cases} 
d(k,l) & d(k,l) < \phi \\
d(k,l) & d(k,l) \geq \phi
\end{cases}
\]

For data compression, only the nonzero-thresholded wavelet coefficients \( d^\phi(k,l) \) are stored. These thresholded coefficients can then be used to reconstruct data when needed. For data reconstruction, the missing wavelet detail coefficients are assumed to be zero. In this paper, the recording limits for each process variable are used as threshold values. The wavelet-based compression algorithm will now be described. Like the data-averaging method, it is also a batch method. Online versions of the algorithm are also available.

2.1.3. Compression Using Commercial PI Software. Commercial data archiving software, such as PI from OSIsoft, Inc., is very efficient at storing data because it automatically compresses time-series data while storing them in a way that provides the best reconstruction. Because PI is widely used for data archiving, it is informative to compare this commercial software with the classical techniques. In particular, the BatchFile Interface for the PI software was used for data compression.
2.2. Data Reconstruction Methods. All of the data compression methods described in the previous section result in *lossy compression*; i.e., it is not possible to reconstruct the compressed data to exactly match the original data. The accuracy by which compressed data can describe the original uncompressed data depends not only on the compression algorithm but also on the method of data reconstruction. Many reconstruction methods are available such as zero-order hold, where the value of a variable is held at the last recorded value until the next recording. Although the zero-order hold is the simplest method, it is not the best method for reconstructing compressed data because holding the value of the variable constant since the last recording causes a loss of much of the information about the possible values of the variable between recordings. However, a zero-order hold can be useful in situations where the process is at steady state.

Linear interpolation is a simple method that can overcome a part of this limitation by reconstructing data between recordings. It can provide more accurate reconstruction for situations where the process is at steady state and for situations where process variables show trends. Higher-order methods such as cubic-spline interpolation can be used to obtain better reconstruction accuracy, but they are computationally more intensive and produce large deviations from the process trend line when the time between consecutive recordings is large. We use linear interpolation to reconstruct compressed data throughout this paper.

More sophisticated methods such as the use of the expectation–maximization algorithm for data reconstruction have also been proposed. However, these methods are sensitive to the amount of missing data and do not perform well when more than 20% of the data are missing.

3. Pattern-Matching Approach

A new pattern-matching strategy developed by the authors is summarized in the flowchart of Figure 2. First, the user defines the *snapshot data* that serve as a template for searching the historical database. The snapshot specifications consist of (i) the relevant process variables and (ii) the duration of the period of interest. These specifications can be arbitrarily chosen by the user; no special plant tests or preimposed conditions are necessary.

To find periods of historical data that are similar to the snapshot data, a window of the same size as the snapshot data is moved through the historical data. The similarity between the snapshot and historical data in the moving window is characterized by the $S_{PCA}$ and $S_{dist}$ similarity factors that are described in section 4. The historical data windows with the largest values of the similarity factors are collected in a *candidate pool*. The individual data windows in the candidate pool are called *records*. After the candidate pool has been formed, a person familiar with the process can then perform a more detailed examination of the records. Details of the moving-window approach for pattern matching are provided by Singhal and Seborg.

4. Pattern Matching Using Similarity Factors

Pattern matching in historical data involves comparisons of different datasets. Suppose that it is desired to find historical datasets that are similar to a particular dataset of current interest. This latter dataset will be referred to as the snapshot data. Because process data are available as multivariate time series where variables are correlated to each other, metrics are required to compare the historical multivariate data to the snapshot data. The authors have proposed and used similarity factors based on principal-component analysis (PCA) for comparing such multivariate datasets. These similarity factors are briefly described next.

4.1. PCA Similarity Factor. Krzanowski developed a method for measuring the similarity of two datasets using a PCA similarity factor, $S_{PCA}$. Consider two datasets that contain the same $n$ variables but not necessarily the same number of measurements. We assume that the PCA model for each dataset contains $k$ principal components, where $k < n$. The number of principal components, $k$, is chosen such that $k$ principal components describe at least 95% of the total variance in each dataset. The similarity between the two datasets is then quantified by comparing their principal components. The appeal of the similarity factor approach is that the similarity between two multivariate datasets is quantified by a single number, $S_{PCA}$.

Consider a current snapshot dataset $S$ and a historical dataset $H$ with each dataset consisting of $m$ measurements of the same $n$ variables. Let $k_S$ be the number of principal components that describe at least 95% of the variance in dataset $S$ and $k_H$ be the number of principal components that describe at least 95% of the variance in dataset $H$. Let $k = \max(k_S, k_H)$, which ensures that $k$ principal components describe at least 95% of the variance in each dataset. Then subspaces of the $S$ and $H$ datasets can be constructed by selecting only the first $k$ principal components for each dataset. The PCA similarity factor compares these reduced subspaces and can be calculated from the angles between principal components$

\[
S_{PCA} = \sum_{i=1}^{k} \sum_{j=1}^{k} \cos^2 \theta_{ij}
\]

where $\theta_{ij}$ is the angle between the $i$th principal component of dataset $S$ and the $j$th principal component of dataset $H$. A value of $S_{PCA}$ of close to 1 means that the datasets $S$ and $H$ are similar.
4.2. Distance Similarity Factor. The distance similarity factor, \( S_{\text{dist}} \), compares two datasets that may have similar spatial orientation but are located far apart. This similarity factor is particularly useful when two data windows have similar principal components, but the values of the process variables may be different because of disturbances of varying magnitudes or setpoint changes. The distance similarity factor can be used to distinguish between these cases.

The Mahalanobis distance, \( S_{\text{Mah}} \), from the center of the historical dataset (\( \bar{x}_H \)) to the center of the current snapshot dataset, \( \bar{x}_S \), is defined as

\[
\Phi = \sqrt{\bar{x}_H - \bar{x}_S}^\top \Sigma_s^{-1} (\bar{x}_H - \bar{x}_S)
\]

where \( \bar{x}_S \) and \( \bar{x}_H \) are sample mean vectors, \( \Sigma_s \) is the covariance matrix for dataset \( S \), and \( \Sigma_s^{-1} \) is the pseudoinverse of \( \Sigma_s \) calculated using singular-value decomposition. The number of singular values used to calculate the pseudoinverse is \( k = \max(k_S, k_H) \). The new distance similarity factor, \( S_{\text{dist}} \), is defined as the probability that the center of the historical dataset, \( \bar{x}_H \), is at least a distance \( \Phi \) from the snapshot dataset \( S \):

\[
S_{\text{dist}} = 2 \frac{1}{\sqrt{2\pi}} \int_0^\infty e^{-z^2/2} \, dz
\]

Note that a one-sided Gaussian distribution is used in eq 6 because \( \Phi \geq 0 \). The error function in eq 6 can be evaluated using standard tables or software packages. Because the distance similarity factor provides a natural complement to the PCA similarity factor, the proposed pattern-matching approach is based on both \( S_{\text{PCA}} \) and \( S_{\text{dist}} \). This approach compares datasets at two levels: (i) orientation of their subspaces and (ii) the distance between them.

It is interesting to note that the widely used Mahalanobis distance metric in the pattern-matching literature and Hotelling’s \( T^2 \) statistic used in process monitoring are related to each other. If \( k \) singular values are used to calculate \( \Sigma_s^{-1} \), then the square of the Mahalanobis distance, \( \Phi \), is equal to Hotelling’s \( T^2 \) statistic for the point \( \bar{x}_H \). The novelty of the proposed distance similarity factor is that it assigns a probability value between 0 and 1 to the Mahalanobis distance between the snapshot and historical datasets.

4.3. Selection of the Candidate Pool. The candidate pool is constructed by selecting the most similar historical data windows. Figure 3 illustrates pattern matching using the moving-window methodology. Suppose that a historical data window has been selected for the candidate pool based on its \( S_{\text{PCA}} \) and \( S_{\text{dist}} \) values. In general, this record will contain data from two operating periods, as shown in Figure 3. For the simulation case study in section 5, the record is considered to have been correctly identified if the center of the moving data window is located in the same type of operating period as the snapshot data. This situation is illustrated in Figure 3, where the snapshot data are for an operating condition called F10. Then a correct match between the moving historical window and the F10 snapshot data occurs when the center of the moving window lies inside a F10 period of historical data. To avoid redundant counting of data windows that represent the same period of operation in the simulated historical database (i.e., historical data windows that are too close to each other), the historical data window that has the largest values of the similarity factors is chosen among the historical data windows that are within \( m \) observations of each other. This restriction results in the selection of data windows that are at least \( m \) observations apart. In the case study, \( m \) is chosen to be the number of data points in the snapshot data. This value is also the duration of each operating period in the historical database.

4.4. Performance Measures for Pattern Matching. Two important metrics are proposed to quantify the effectiveness of a pattern-matching technique. However, first several definitions are introduced:

- \( N_P \): The size of the candidate pool. \( N_P \) is the number of historical data windows that have been labeled similar to the snapshot data by a pattern-matching technique. The data windows collected in the candidate pool are called records.
- \( N_1 \): The number of records in the candidate pool that are actually similar to the current snapshot, i.e., the number of correctly identified records.
- \( N_2 \): The number of records in the candidate pool that are actually not similar to the current snapshot, i.e., the number of incorrectly identified records.
- \( N_{DB} \): The total number of historical data windows that are actually similar to the current snapshot. In general, \( N_{DB} \neq N_P \).

The first metric, the pool accuracy \( p \), characterizes the accuracy of the candidate pool:

\[
p = \frac{N_1}{N_P} \times 100\%
\]

A second metric, the pattern-matching efficiency \( \eta \), characterizes how effective the pattern-matching technique is in locating similar records in the historical database. It is defined as

\[
\eta = \frac{N_1}{N_{DB}} \times 100\%
\]

When the pool size \( N_P \) is small (\( N_P < N_{DB} \)), then the efficiency \( \eta \) will be small because \( N_1 \leq N_P \). A theoretical maximum efficiency, \( \eta_{\text{max}} \), for a given pool size \( N_P \) can be calculated as follows:

\[
\eta_{\text{max}} = \begin{cases} 
N_P \times 100\% & \text{for } N_P < N_{DB} \\
100\% & \text{for } N_P \geq N_{DB}
\end{cases}
\]
of the two quantities ($\xi$) is used as a measure of the overall effectiveness of pattern matching:

$$\xi = \frac{D + \eta}{2}$$  (10)

In pattern-matching problems, the relative importance of $p$ and $\eta$ metrics is application dependent. For example, a busy engineer may want to locate a small number of previous occurrences of an abnormal situation without having to waste time evaluating incorrectly identified records (false positives). In this situation, a large value of $p$ is more important than a large value of $\eta$, and $N_P$ should be small (e.g., 2–5). In other types of applications, it might be desirable to locate most (or even all) previous occurrences of an abnormal situation, for business or legal reasons. This type of situation could arise during the investigation of a serious plant accident or after the recall of a defective product. Here, $\eta$ is more important than $p$, and a relatively large value of the candidate pool size, $N_P$, is acceptable.

5. Simulation Case Study: Continuous Stirred Tank Reactor (CSTR) Example

To compare the effect of data compression on pattern matching, a case study was performed for a simulated chemical reactor. A nonlinear CSTR with cooling jacket dynamics, a variable liquid level, and a first-order irreversible reaction, $A \rightarrow B$, was simulated. A total of 28 different operating conditions that included a number of faults of varying magnitudes, disturbances, and setpoint changes were simulated for the CSTR, and 14 process variables for each operating condition were recorded. The details of the simulation study and 28 different operating conditions for the CSTR are available in previous publications.\textsuperscript{17,19,32}

The simulation generated two groups of data. The first group contained 28 snapshot datasets, each corresponding to the nominal magnitude of a fault, while the second group contained a large historical dataset of over 474 000 observations that included faults, disturbances, and setpoint changes of varying magnitudes. Data compression was performed for each of the 28 snapshot
6. Results and Discussion

The data compression methods were evaluated in two ways: (1) reconstruction error and (2) degree of similarity between the original and reconstructed data. The reconstruction error analysis calculates the difference between the original and compressed time series, while the calculation of similarity factors compares the datasets from a pattern-matching perspective. The $S_{PCA}$ and $S_{dist}$ similarity factors were used to quantify the similarity between the original and reconstructed data. The results with respect to the reconstruction error are presented first, and the results for pattern matching are presented in section 6.2.

6.1. Comparison of Different Methods with Respect to the Reconstruction Error. The data compression methods described in section 2.1 were first compared on the basis of the reconstruction error for a constant compression ratio (CR). It is easier to compare different methods with respect to the reconstruction error if all compression methods have the same CR. The CR is defined as

$$CR = \frac{\text{number of data points in the original dataset}}{\text{number of data points in the compressed dataset}} \quad (11)$$

The mean-squared error (MSE) of reconstruction is defined as

$$\text{MSE} = \frac{1}{m \times n} \sum_{i=1}^{m} \sum_{j=1}^{n} (x_{i,j} - \hat{x}_{i,j})^2 \quad (12)$$

where $m$ is the number of measurements in the original dataset; $n$ is the number of variables; and $\epsilon_{i,j} = x_{i,j} - \hat{x}_{i,j}$, where $x_{i,j}$ represents the $j$ measurement of the $i$ variable in the original data and $\hat{x}_{i,j}$ is the corresponding reconstructed value.

Obtaining a constant CR for all methods requires adjustment of the recording limits individually for each method. As mentioned in the previous section, the recording limits for a given method and each process variable are proportional to their standard deviations in Table 2. For example, the OSI PI recording limits were chosen as 3$\sigma$, while the recording limits for the boxcar method were adjusted to produce the same CR as that in the PI method. Thus, the recording limits for the boxcar method were 2.23$\sigma$.

Data compression was performed using PI’s proprietary algorithm. The CR for the 28 snapshot datasets was 14.8 using PI. The recording limits for all other methods were then adjusted using numerical root-finding techniques, such as the bisection method, to obtain an average CR of approximately 14.8 for each method. The average results for the 28 operating conditions presented in Table 3 show that the PI algorithm provides the best reconstruction of the compressed data, while wavelet-based compression is second best. The common practice of averaging the data provides the worst reconstruction.

6.2. Effect of Data Compression on Pattern Matching. An objective of this paper is to evaluate the effect of data compression on pattern matching. Data compression affects pattern matching because the origi-
Table 4. $S_{PCA}$ Values between the Reconstructed and Original Datasets for the CSTR Example

<table>
<thead>
<tr>
<th></th>
<th>PCA</th>
<th></th>
<th></th>
<th>PCA</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>boxcar</td>
<td>averaging</td>
<td>wavelet</td>
<td>PI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$S_{PCA}$</td>
<td>$S_{dist}$</td>
<td>$S_{PCA}$</td>
<td>$S_{dist}$</td>
<td>$S_{PCA}$</td>
<td>$S_{dist}$</td>
<td>$S_{PCA}$</td>
</tr>
<tr>
<td>average</td>
<td>0.88</td>
<td>0.67</td>
<td>0.92</td>
<td>0.97</td>
<td>0.95</td>
<td>&gt;0.99</td>
</tr>
<tr>
<td>N</td>
<td>0.80</td>
<td>0.69</td>
<td>0.96</td>
<td>0.98</td>
<td>0.99</td>
<td>&gt;0.99</td>
</tr>
<tr>
<td>F1</td>
<td>0.84</td>
<td>0.42</td>
<td>0.99</td>
<td>0.98</td>
<td>0.99</td>
<td>&gt;0.99</td>
</tr>
<tr>
<td>F2</td>
<td>0.87</td>
<td>0.87</td>
<td>0.83</td>
<td>0.99</td>
<td>0.99</td>
<td>&gt;0.99</td>
</tr>
<tr>
<td>F3</td>
<td>0.83</td>
<td>0.89</td>
<td>0.95</td>
<td>0.99</td>
<td>0.97</td>
<td>&gt;0.99</td>
</tr>
<tr>
<td>F4</td>
<td>0.83</td>
<td>0.76</td>
<td>0.93</td>
<td>0.99</td>
<td>0.96</td>
<td>&gt;0.99</td>
</tr>
<tr>
<td>F5</td>
<td>0.92</td>
<td>0.78</td>
<td>0.88</td>
<td>0.98</td>
<td>0.98</td>
<td>&gt;0.99</td>
</tr>
<tr>
<td>F6</td>
<td>0.87</td>
<td>0.70</td>
<td>0.81</td>
<td>0.98</td>
<td>0.98</td>
<td>&gt;0.99</td>
</tr>
<tr>
<td>F7</td>
<td>0.93</td>
<td>0.84</td>
<td>0.92</td>
<td>0.99</td>
<td>0.94</td>
<td>&gt;0.99</td>
</tr>
<tr>
<td>F8</td>
<td>0.95</td>
<td>0.92</td>
<td>0.94</td>
<td>0.99</td>
<td>0.96</td>
<td>&gt;0.99</td>
</tr>
<tr>
<td>F9</td>
<td>0.98</td>
<td>0.81</td>
<td>0.79</td>
<td>0.98</td>
<td>0.99</td>
<td>&gt;0.99</td>
</tr>
<tr>
<td>F10</td>
<td>0.83</td>
<td>0.63</td>
<td>0.91</td>
<td>0.98</td>
<td>0.99</td>
<td>&gt;0.99</td>
</tr>
<tr>
<td>F11</td>
<td>0.83</td>
<td>0.79</td>
<td>0.98</td>
<td>0.98</td>
<td>0.99</td>
<td>&gt;0.99</td>
</tr>
<tr>
<td>F12</td>
<td>0.84</td>
<td>0.84</td>
<td>0.93</td>
<td>0.98</td>
<td>0.98</td>
<td>&gt;0.99</td>
</tr>
<tr>
<td>F13</td>
<td>0.86</td>
<td>0.34</td>
<td>0.99</td>
<td>0.98</td>
<td>0.99</td>
<td>&gt;0.99</td>
</tr>
<tr>
<td>S1</td>
<td>0.85</td>
<td>0.73</td>
<td>0.99</td>
<td>0.98</td>
<td>0.99</td>
<td>&gt;0.99</td>
</tr>
<tr>
<td>S2</td>
<td>0.90</td>
<td>0.69</td>
<td>0.89</td>
<td>0.98</td>
<td>0.98</td>
<td>&gt;0.99</td>
</tr>
<tr>
<td>S3</td>
<td>0.88</td>
<td>0.39</td>
<td>0.93</td>
<td>0.99</td>
<td>0.98</td>
<td>&gt;0.99</td>
</tr>
<tr>
<td>S4</td>
<td>0.94</td>
<td>0.30</td>
<td>0.94</td>
<td>0.99</td>
<td>0.88</td>
<td>&gt;0.99</td>
</tr>
<tr>
<td>F1</td>
<td>0.87</td>
<td>0.70</td>
<td>0.92</td>
<td>0.99</td>
<td>0.88</td>
<td>&gt;0.99</td>
</tr>
<tr>
<td>F2</td>
<td>0.80</td>
<td>0.89</td>
<td>0.83</td>
<td>0.99</td>
<td>0.80</td>
<td>&gt;0.99</td>
</tr>
<tr>
<td>F3</td>
<td>0.82</td>
<td>0.88</td>
<td>0.90</td>
<td>0.99</td>
<td>0.99</td>
<td>&gt;0.99</td>
</tr>
<tr>
<td>F4</td>
<td>0.96</td>
<td>0.84</td>
<td>0.97</td>
<td>0.99</td>
<td>0.99</td>
<td>&gt;0.99</td>
</tr>
<tr>
<td>S5</td>
<td>0.80</td>
<td>0.88</td>
<td>0.95</td>
<td>0.99</td>
<td>0.98</td>
<td>&gt;0.99</td>
</tr>
<tr>
<td>S6</td>
<td>0.89</td>
<td>0.01</td>
<td>0.87</td>
<td>0.99</td>
<td>0.88</td>
<td>&gt;0.99</td>
</tr>
<tr>
<td>S7</td>
<td>0.85</td>
<td>0.20</td>
<td>0.81</td>
<td>0.99</td>
<td>0.86</td>
<td>&gt;0.99</td>
</tr>
<tr>
<td>S8</td>
<td>0.90</td>
<td>0.75</td>
<td>0.79</td>
<td>1.00</td>
<td>0.97</td>
<td>&gt;0.99</td>
</tr>
<tr>
<td>S9</td>
<td>0.99</td>
<td>0.85</td>
<td>0.99</td>
<td>1.00</td>
<td>&gt;0.99</td>
<td>&gt;0.99</td>
</tr>
<tr>
<td>S10</td>
<td>&gt;0.99</td>
<td>0.99</td>
<td>&gt;0.99</td>
<td>&gt;0.99</td>
<td>&gt;0.99</td>
<td>&gt;0.99</td>
</tr>
</tbody>
</table>

The PCA and averaging methods produced low $S_{PCA}$ and $S_{dist}$ values for the normal operation. The low values occur for normal operation because there is not much variation in the measurements. This results in small eigenvalues for the covariance matrix and makes the Mahalanobis distance very sensitive to mean shifts. The averaging method produces high $S_{PCA}$ and $S_{dist}$ values for all operating conditions. This suggests that although it is not a very accurate method for data compression, it is still very effective in representing the data characteristics for pattern matching.

On the other hand, even though the PI algorithm results in a very low MSE, it does not represent the data very well for pattern matching. The wavelet method produces both low MSE and high similarity factor values. The wavelet transform preserves the essential dynamic features of the signal in the detail coefficients while retaining the correlation structure between the variables in the approximation coefficients. These two features of the wavelet transform produce low MSE and high $S_{PCA}$ values between the original and reconstructed data. These features also minimize mean shifts and result in high $S_{dist}$ values. By contrast, the PI method records data very accurately and produces very low MSE values, but its variable sampling rates change the correlation structure between variables and produce low $S_{PCA}$ values. Variable sampling also affects the mean value of the reconstructed data and produces low $S_{dist}$ values.

The effect of compression on pattern matching using the PI software was investigated further by comparing the reconstructed and original datasets using similarity factors. Similarity factors were calculated for all possible pairs of the snapshot datasets. The results are shown in Figures 5 and 6. The original datasets were considered to be the reference datasets for the purpose of scaling. The numerical values represented as gray scale in a row are the similarity factors between the original snapshot and reconstructed datasets obtained from data compression using PI. The diagonal entries and the entries that are larger than the diagonal value in the same row are crossed.

The $S_{PCA}$ values for the original snapshot and reconstructed datasets are presented in Figure 5. This figure shows the effect of data compression on the correlation between the process variables. The diagonal and off-diagonal values in a row that are greater than the diagonal value are crossed in Figure 5b. These off-diagonal values indicate that data compression (using PI) changes the correlation between variables such that a snapshot now is more similar to reconstructed datasets of other operating conditions than it is to its own reconstructed dataset.

Figure 5b indicates that the normal operation and operating conditions F1, F4, F_4, F_6, F_9, S_1, F13, and O2 are less similar to their reconstructed datasets than to other reconstructed datasets. For example, normal operation is more similar to reconstructed datasets corresponding to operating conditions F1, F6, F_11, and F12; F8 is more similar to reconstructed datasets corresponding to operating conditions F2, F_4, F_5, F_5, and S_1. These results suggest that mismatches between these operating conditions would be very high if the historical data were compressed and the snapshot data were not compressed. Thus, it would be very difficult to locate these operating conditions in the historical database if the data were compressed and averaged and wavelet methods.
recorded using PI. In Figure 5a, the diagonal $S_{PCA}$ values are exactly equal to 1 because the datasets being compared were not compressed, while in Figure 5b, data compression results in diagonal values that are less than 1. These results suggest that compression has a detrimental effect on pattern matching.

Figure 6 shows the $S_{dist}$ values for the original snapshot and reconstructed datasets. Note that the diagonal values for normal operation and operating conditions F10, F11, and F12 are very small. In particular, the $S_{dist}$ value for normal operation is only 0.05. The reason for the low $S_{dist}$ values is that there is very little variation in the values of the process variables for these operating conditions. Consequently, the eigenvalues of the covariance matrix are small and, thus, the Mahalanobis distance becomes very sensitive to even small mean shifts. The normal operation is affected most strongly by mean shifts because the process variables have very small variations, and the data compression algorithm records only those values that have large deviations from the original mean. When the datasets being compared are not compressed, the diagonal $S_{dist}$ values would be equal to 1, as indicated by Figure 6a.

Figure 6b indicates that data compression using PI causes significant mean shifts for normal operation and operating conditions F11, F12, F13, O1, O2, and O4 because the off-diagonal $S_{dist}$ values are larger than the diagonal values. For example, significant mean shifts can be observed for the normal operation that has larger $S_{dist}$ values for reconstructed datasets corresponding to operating conditions F12, F13, and O1–O4 compared to the reconstructed normal operation.

6.2.1. Finding Similar Patterns in Compressed Historical Data. We now evaluate the effect of data compression on locating similar patterns in the historical data. Two situations are considered: (i) both the snapshot and historical data are compressed, and (ii) only the historical data are compressed. When snapshot data were compressed, the same compression method was used for both the snapshot and historical data. The historical data for the CSTR example described in section 5 were compressed using three different methods: boxcar, averaging, and wavelet. The performance of the proposed pattern-matching technique for compressed historical data was then evaluated. As described by Singhal19 and Singhal and Seborg,17 a data window

![Figure 5. $S_{PCA}$ self-similarity matrices for original and compressed datasets.](image1)

![Figure 6. $S_{dist}$ self-similarity matrices for original and compressed datasets.](image2)
that was the same size as the snapshot data \((S)\) was moved through the historical database, 100 observations at a time (i.e., \(w = 100\)). The \(i\)th moving window was denoted as \(H_i\).

The snapshot data were scaled to zero mean and unit variance, and the historical data were scaled using the scaling factors for the snapshot data. Similarity factors were then calculated for each instance of the window moving through the historical data. After the entire historical dataset was analyzed for one set of snapshot data, the analysis was repeated for a new snapshot dataset. A total of 28 different snapshot datasets, one for each of the 28 operating conditions, were used for pattern matching.

Table 5 compares the pattern-matching results for different data compression methods. The best pattern-matching results were obtained when the data were compressed using the wavelet method. In Table 5, the optimum \(N_F\) was calculated by calculating \(\xi\) for different \(N_F\) values and then choosing the value of \(N_F\) for which \(\xi\) was the largest. Table 5 indicates that pattern matching is adversely affected by data compression when the data are compressed using either the averaging method or the boxcar compression methods. By contrast, wavelet-based compression has very little effect on pattern matching because similar results are obtained for both compressed and uncompressed data. Table 6 presents the results for the situation when the snapshot data are not compressed and only the historical data are compressed. The \(p\), \(\eta\), and \(\xi\) values in Table 6 are somewhat lower compared to those in Table 5, particularly for wavelet compression. Thus, if the historical data are compressed, it may be beneficial to compress the snapshot data as well to obtain better pattern matching. In this situation, wavelet compression provides the best pattern matching. The \(\xi\) values for the averaging method are higher than those of the wavelet method because of large \(N_F\) and \(\eta\) values. In general, data compression had a detrimental effect on pattern matching, but \(S_{PCA}\) and \(S_{dist}\) methods performed very well.

### Table 5. Effect of Data Compression on Pattern Matching When Both Snapshot and Historical Data Are Compressed Using the Same Method

<table>
<thead>
<tr>
<th>compression method</th>
<th>similarity factor</th>
<th>(N_F)</th>
<th>(p) (%)</th>
<th>(\eta) (%)</th>
<th>(\eta_{max}) (%)</th>
<th>(\xi) (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>original data ((\text{uncompressed})) (S_{PCA})</td>
<td>34</td>
<td>43</td>
<td>90</td>
<td>99</td>
<td>66</td>
<td></td>
</tr>
<tr>
<td>(S_{dist})</td>
<td>25</td>
<td>41</td>
<td>68</td>
<td>100</td>
<td>53</td>
<td></td>
</tr>
<tr>
<td>boxcar (S_{PCA})</td>
<td>56</td>
<td>24</td>
<td>84</td>
<td>100</td>
<td>46</td>
<td></td>
</tr>
<tr>
<td>(S_{dist})</td>
<td>60</td>
<td>19</td>
<td>73</td>
<td>100</td>
<td>46</td>
<td></td>
</tr>
<tr>
<td>averaging (S_{PCA})</td>
<td>21</td>
<td>49</td>
<td>65</td>
<td>95</td>
<td>57</td>
<td></td>
</tr>
<tr>
<td>(S_{dist})</td>
<td>24</td>
<td>40</td>
<td>65</td>
<td>96</td>
<td>53</td>
<td></td>
</tr>
<tr>
<td>wavelet (S_{PCA})</td>
<td>34</td>
<td>38</td>
<td>82</td>
<td>99</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>(S_{dist})</td>
<td>52</td>
<td>25</td>
<td>83</td>
<td>100</td>
<td>54</td>
<td></td>
</tr>
</tbody>
</table>

### Table 6. Effect of Data Compression on Pattern Matching When Snapshot Data Are Not Compressed and Historical Data Are Compressed

<table>
<thead>
<tr>
<th>compression method</th>
<th>similarity factor</th>
<th>(N_F)</th>
<th>(p) (%)</th>
<th>(\eta) (%)</th>
<th>(\eta_{max}) (%)</th>
<th>(\xi) (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>original data ((\text{uncompressed})) (S_{PCA})</td>
<td>34</td>
<td>43</td>
<td>90</td>
<td>99</td>
<td>66</td>
<td></td>
</tr>
<tr>
<td>(S_{dist})</td>
<td>25</td>
<td>41</td>
<td>68</td>
<td>100</td>
<td>53</td>
<td></td>
</tr>
<tr>
<td>boxcar (S_{PCA})</td>
<td>56</td>
<td>24</td>
<td>84</td>
<td>100</td>
<td>46</td>
<td></td>
</tr>
<tr>
<td>(S_{dist})</td>
<td>60</td>
<td>19</td>
<td>73</td>
<td>100</td>
<td>46</td>
<td></td>
</tr>
<tr>
<td>averaging (S_{PCA})</td>
<td>21</td>
<td>49</td>
<td>65</td>
<td>95</td>
<td>57</td>
<td></td>
</tr>
<tr>
<td>(S_{dist})</td>
<td>24</td>
<td>40</td>
<td>65</td>
<td>96</td>
<td>53</td>
<td></td>
</tr>
<tr>
<td>wavelet (S_{PCA})</td>
<td>34</td>
<td>38</td>
<td>82</td>
<td>99</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>(S_{dist})</td>
<td>52</td>
<td>25</td>
<td>83</td>
<td>100</td>
<td>54</td>
<td></td>
</tr>
</tbody>
</table>

### 7. Conclusions

A comparison of some popular data compression methods has shown that the classical methods such as boxcar and data-averaging compression do not accurately represent data in terms of the reconstruction error. Data compressed using the PI software very accurately represent the original data but result in low similarity factor values for pattern-matching applications. Compression using the wavelet method produces reconstruction errors that are higher than those obtained with PI but much lower than conventional compression methods such as boxcar, etc. Data compressed using wavelets also show a high degree of similarity with the original data. This feature tends to minimize the misclassifications that result from data compression. Thus, if the objective is to accurately represent data with minimum reconstruction errors, then PI software provides the best compression. However, if pattern matching using similarity factors is the primary concern, then wavelet compression would provide better results. When the historical data are compressed, it is beneficial to compress the snapshot data as well for pattern-matching applications.

### Acknowledgment

The authors thank OSI Software, Inc., for providing financial support and the data archiving software PI and Gregg LeBlanc at OSI for providing software support during the research. Financial support from ChevronTexaco Research and Technology Co. is also acknowledged.

### Nomenclature

- \(C_A\) = concentration of species A in the reactor (mol/L)
- \(C_{AP}\) = concentration of species A in the reactor feed stream (mol/L)
- \(h\) = liquid level in the reactor (dm)
- \(hc\) = level controller signal (mA)
- \(Q\) = flow rate of the reactor outlet stream (L/min)
- \(QC\) = flow rate controller signal (mA)
- \(Q_C\) = coolant flow rate (L/min)
- \(Q_F\) = feed flow rate of the reactor feed stream (L/min)
- \(T\) = temperature in the reactor (K)
- \(TC\) = reactor temperature controller signal (mA)
- \(TC_F\) = temperature of the coolant in the cooling jacket (K)
- \(TV\) = temperature of the reactor feed stream (K)

### Literature Cited


