Linear and Neural-Network Methods for Condensing High-Dimensional Measurements

Michael L. Mavrovouniotis, Venkatramana N. Reddy
Northwestern University (Evanston, IL)

Introduction

Process data are the foundation of process monitoring, evaluation and control. Advancements in automation allow the collection of large volumes of process data. A process may be equipped with hundreds or even thousands of sensors with sampling intervals of seconds or minutes. As an important step towards process understanding, engineers need to uncover the significant patterns hidden in process data. Dimensionality reduction is a way of summarizing information carried by a large number of observed variables with a few latent variables. Through dimensionality reduction, we obtain not only a reduced data set but also a model that relates all observed variables to a few latent variables; such a model is valuable to many types of data screening tasks, such as data noise reduction, missing sensor replacement, gross error detection and correction, and fault detection.

Given an $m \times n$ matrix representing m measurements made on n variables, n is the observed dimensionality of the data set. Reduction of data dimensionality aims to map the original data matrix to a much smaller matrix of dimension $m \times f$ ($f << n$), which is able to reproduce the original matrix with minimum distortion. The dimensionality reduction is useful when there exist correlations among the observed variables. The reduced matrix describes latent variables extracted from the original matrix. Ideally, the f latent variables should retain all nonrandom variations in the observed variables, exploiting correlations and redundancies in the measurements.

Linear Approach

The linear technique of Principal Component Analysis (PCA) is the most commonly used technique, and it has been shown to facilitate many types of data analysis in process engineering, including data validation and fault detection (Wise and Ricker, 1989), quality control (MacGregor, 1989), data visualization (Stephanopoulos and Guterman, 1989), and process monitoring (Raich and Cinar, 1994).

Singular value decomposition (SVD) provides a computationally efficient method for PCA. Any m x n matrix $\mathbf{A}$ of rank r can be decomposed into the following form (Strang, 1988):

$$\mathbf{A} = \mathbf{u}_1 s_1 \mathbf{v}_1^T + \mathbf{u}_2 s_2 \mathbf{v}_2^T + \ldots + \mathbf{u}_r s_r \mathbf{v}_r^T \quad (s_1 \geq s_2 \geq \ldots \geq s_r > 0)$$

where s_i (i = 1, 2, ..., r) are positive scalars in descending order, $\mathbf{u}_i$ (i = 1, 2, ..., r) are m x 1 orthonormal vectors and $\mathbf{v}_i$ (i = 1, 2, ..., r) are n x 1 orthonormal vectors. The first f terms of the above decomposition provide the best approximation to $\mathbf{A}$ with f principal components.

PCA is a linear technique in the sense that it uses linear functions to model relationships between observed variables and latent variables. Factor analysis (FA) is another linear technique for

dimensionality reduction. Both PCA and FA try to represent some aspect of the covariance (or correlation) matrix of the observed data as well as possible. PCA concentrates on the variances of individual variables, whereas in FA the interest is in the covariances of different variables (Jolliffe, 1986).

The linear techniques reduce data dimensionality by exploiting linear correlations between observed variables. If there exist nonlinear correlations between observed variables, as usually occurs in industrial processes, a nonlinear method will describe the data with greater accuracy and/or by fewer latent variables than a linear method. The necessity of nonlinear methods has long been recognized. In this paper, we present a new method for reducing data dimensionality using neural networks as nonlinear models for observed variables and latent variables.

Concept of Input Training

We have developed Input-Training Networks (Tan and Mavrovouniotis, 1995) as a neural network architecture that carries out a non-linear version of PCA. An Input-Training net (or IT-Net) has three layers; its inputs are the latent variables and its outputs are (approximations of) the process measurements. When training an IT-net, we adjust not only the internal network parameters but also input values to reproduce the given data as accurately as possible. When network inputs are adjusted, each output sample should be uniquely associated with one input vector. Figure 1 illustrates a 2-4-5 network (i.e., a network with 2 input nodes, 4 hidden nodes, and 5 output nodes) with input training used for reducing the dimensionality of a data set from five to two. Each input vector $(x_{p1}\ \ x_{p2})^T$ is adjusted to minimize only the error of its *corresponding output* vector $(z_{p1}\ \ z_{p2}\ \ ...\ z_{p5})^T$ while the internal network parameters are trained using *all* output samples.

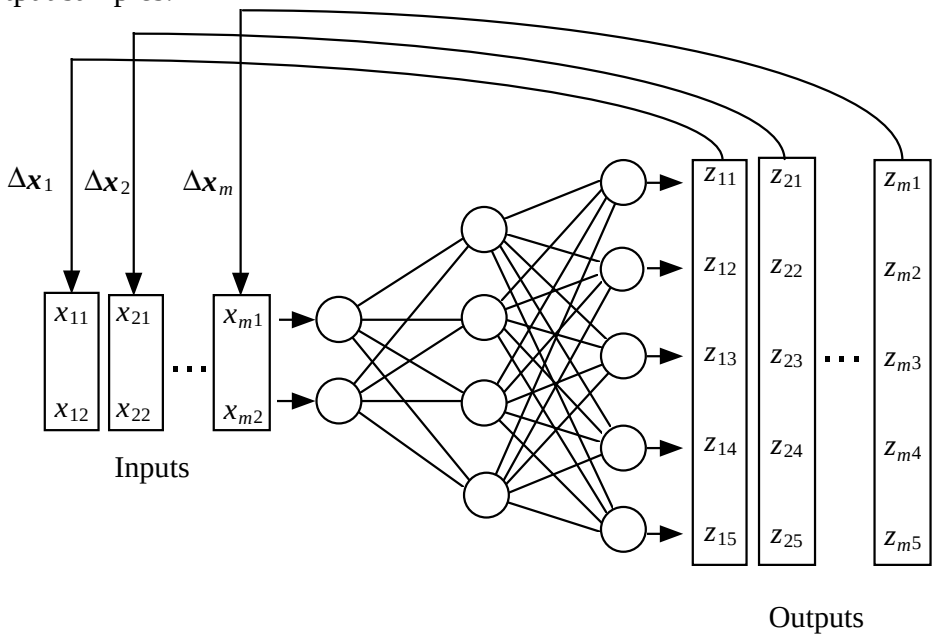

Figure 1. *Concept of input training*

After the weights and inputs are properly trained, we obtain a reduced matrix and a model (in the form of a neural network) that maps latent variables to the original high-dimensional space. Thus all requirements for data dimensionality reduction can be fulfilled through training a single-hidden-layer network and its input simultaneously. Two characteristics are basic for an IT-net:

the input layer has fewer nodes than any other layer; and inputs are adjusted according to corresponding outputs.

It should be noted that the term *input* in the context of input training is slightly different from what is used for traditional neural networks, where inputs are always given. However, it is not unusual to adjust inputs to a model while the model parameters are being modified to minimize the output error. Examples of this model-fitting strategy include the polynomial PCA and polynomial FA reviewed in the introduction of this paper. Input training is an application of the same strategy in neural networks.

To gain better understanding of input training, we study the simplest IT-net, one with linear nodes only and no hidden layer. The objective of training such an IT-net is to minimize the sum of square errors:

$$\min E = \left\| \mathbf{X}\mathbf{W}^{\mathrm{T}} - \mathbf{T} \right\|^2 \tag{1}$$

where $\mathbf{T}$ is the original data matrix (m x n), $\mathbf{X}$ the reduced matrix (m x f), and $\mathbf{W}$ the weight matrix (n x f) of the IT-net. To solve the minimization problem, we obtain the partial derivatives of E with respect to all variables, including network inputs:

$$\frac{\partial E}{\partial \mathbf{W}} = \begin{bmatrix} \frac{\partial E}{\partial w_{11}} & \Lambda & \frac{\partial E}{\partial w_{1f}} \\ \mathrm{M} & & \mathrm{M} \\ \frac{\partial E}{\partial w_{n1}} & \Lambda & \frac{\partial E}{\partial w_{nf}} \end{bmatrix} = 2(\mathbf{X}\mathbf{W}^{\mathrm{T}} - \mathbf{T})^{\mathrm{T}}\mathbf{X} \tag{2}$$

$$\frac{\partial E}{\partial \mathbf{X}} = \begin{bmatrix} \frac{\partial E}{\partial x_{11}} & \Lambda & \frac{\partial E}{\partial x_{1f}} \\ \mathrm{M} & & \mathrm{M} \\ \frac{\partial E}{\partial x_{m1}} & \Lambda & \frac{\partial E}{\partial x_{mf}} \end{bmatrix} = 2(\mathbf{X}\mathbf{W}^{\mathrm{T}} - \mathbf{T})\mathbf{W} \tag{3}$$

In order to reach the minimum training error, all derivatives in Eqs. 2 and 3 are necessarily zero, which leads to the following equations:

$$\mathbf{W} = \mathbf{T}^{\mathrm{T}}\mathbf{X}(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1} \tag{4}$$

$$\mathbf{X} = \mathbf{T}\mathbf{W}(\mathbf{W}^{\mathrm{T}}\mathbf{W})^{-1} \tag{5}$$

A straightforward way of using the above equations is to start with random input values and apply Eqs. 4 and 5 alternately to calculate weights and new input values. This process is equivalent to the following iterative procedure:

$$\mathbf{X}^{(\mathrm{new})} = \mathbf{T}\mathbf{T}^{\mathrm{T}}\mathbf{X}(\mathbf{X}^{\mathrm{T}}\mathbf{T}\mathbf{T}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{X} \tag{6}$$

This iteration is close in spirit to the block power method (Strang, 1988) for calculating multiple eigenvectors of $\mathbf{T}\mathbf{T}^{\mathrm{T}}$. The difference is that the orthogonalization in each iteration of the block power method is replaced with right multiplication of an f x f matrix, $(\mathbf{X}^{\mathrm{T}}\mathbf{T}\mathbf{T}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{X}$. Specially, if the IT-net has only one input node, $\mathbf{X}$ becomes a vector and the above iteration reduces to:

$$\mathbf{x}^{(\mathrm{new})} = \frac{\mathbf{x}^{\mathrm{T}}\mathbf{x}}{\mathbf{x}^{\mathrm{T}}\mathbf{T}\mathbf{T}^{\mathrm{T}}\mathbf{x}}\mathbf{T}\mathbf{T}^{\mathrm{T}}\mathbf{x} \tag{7}$$

which is the power method for finding the largest eigenvalue of $\mathbf{TT}^T$ using the Rayleigh quotient as the scaling factor in each iteration. It can be shown that this iteration converges to an eigenvector of $\mathbf{TT}^T$ corresponding to the largest eigenvalue, which is exactly the first principal component of **T**. Therefore, training an IT-net with one input node and no hidden layer is equivalent to PCA.

IT-nets are basically feedforward networks. With one hidden layer of sigmoidal nodes, a feedforward network is able to approximate any nonlinear function to an arbitrary accuracy given sufficient hidden nodes (Cybenko, 1989). Let $\phi_k(\lambda_1, ..., \lambda_f)$, $k = 1,...,n$, denote the nonlinear mappings by an IT-net. When the output error for a given data vector, $(t_{p1}, ..., t_{pn})^T$, is minimized at an input vector, $(x_{p1}, ..., x_{pf})^T$, we have:

$$\left[\frac{\partial}{\partial \lambda_i}\sum_k (\phi_k - t_{pk})^2\right]_{\lambda_i = x_{pi}} = 0, \quad i = 1,\ldots,f \tag{8}$$

which can be rearranged into:

$$\sum_k (z_{pk} - t_{pk})\left[\frac{\partial \phi_k}{\partial \lambda_i}\right]_{\lambda_i = x_{pi}} = 0, \quad i = 1,\ldots,f \tag{9}$$

where $z_{pk} = \phi_k(x_{p1}, ..., x_{pf})$. If the IT-net has exactly one input node, the functions $\phi_k(\lambda)$, $k = 1,...,n$, represent a smooth curve in n-dimensional space. Eq. 9 indicates that the vector of output errors, $\boldsymbol{z}_p - \boldsymbol{t}_p$, is orthogonal to the tangent of the curve at $\lambda = x_p$ when the sum of square errors is minimized through input training.

Training IT-Nets

For IT-nets with hidden layers, a direct iterative procedure for network inputs such as Eq. 6 is not available. We extend the backpropagation training method to network inputs. Similar to network weights, network inputs are modified using errors backpropagated from the output layer. The steepest descent direction for minimizing the output errors through adjustment of network inputs is derived below.

Let t_{pk} be the value of the kth observed variable in the pth training sample and z_{pk} the corresponding IT-net approximation. Then the objective function to be minimized in network training is:

$$E = \sum_p \sum_k (z_{pk} - t_{pk})^2 \tag{10}$$

The steepest descent direction for optimizing network inputs x_{pi} is given by:

$$\Delta x_{pi} = -\frac{\partial E}{\partial x_{pi}} = \sum_k (t_{pk} - z_{pk})\frac{\partial z_{pk}}{\partial x_{pi}} \tag{11}$$

Assuming that input and output nodes use the identity activation function while hidden nodes use a sigmoidal function, the network output is given by:

$$z_{pk} = \sum_j w_{kj}\sigma(b_j + \sum_i v_{ji} x_{pi}) \tag{12}$$

where $\sigma(.)$ is a sigmoidal function, b_j is the bias of the jth hidden node, and V_{ji} and W_{kj} are network weights. Hence, the steepest descent direction for training network inputs is:

$$\Delta x_{pi} = \sum_j v_{ji} \delta_{pj} \tag{13}$$

where δ_{pj} is the propagated error at the hidden layer and has been defined as:

$$\delta_{pj} = \sigma'(b_j + \sum_i v_{ji} x_{pi}) \sum_k w_{kj} (t_{pk} - z_{pk}) \tag{14}$$

Note that the steepest descent direction for training network weights between the input layer and the hidden layer is given by:

$$\Delta v_{ji} = \sum_p x_{pi} \delta_{pj} \tag{15}$$

Therefore, the extra computation required for training the inputs is negligible compared with training the rest of the network. In the above derivation, we have assumed that only hidden nodes use sigmoidal functions and that input and output nodes are linear. The same derivation can be carried out for networks with sigmoidal output and/or input nodes and Eq. 13 still holds but with different δ_{pj}.

Testing and Using IT-Nets

A trained IT-net can be tested through cross-validation. In this sense, testing and using an IT-net involves the same computing task. We will need to describe only testing. Because the network inputs are unknown for testing sample, the appropriate way to test a trained network is to adjust the network inputs while freezing all internal network parameters (weights and biases). In other words, testing an IT-net still requires a searching procedure which optimizes each input pattern to yield a good approximation for its corresponding output sample.

It should be noted that optimization of inputs for testing is much less time-consuming than training a whole IT-net. First of all, testing can be done for each individual sample and it involves much fewer searching variables than training the whole network. In addition, since the inputs of training data are available, we can apply a nearest neighbor algorithm to obtain good initial guesses for the inputs of testing samples. Finally, replacing the small fixed learning rate with a one-dimensional search significantly improves the speed of optimizing network inputs.

Applications to Process Measurements

The data for the examples that follow come from temperature sensors in the air heaters of an electric power plant. A total of 44 variables will be considered.

We first examine prediction errors of models constructed from PCA and from IT-Nets. Process data obtained as measurements from the plant often contain noise. The reduced set of variables should retain the most important non-random variation within the data set. When the reduced set is projected back to the original dimension, a corrected set of values is reconstructed based on the non-random information captured by the reduced set of variables.

The results (Table 1) show that, when the data dimension is reduced from 44 to 3, the 3-11-44 IT-net, with RMSE (Root Mean Square Error) of 0.0057 performs better than the linear 3-PC model (RMSE = 0.0281). When the dimension of the data is reduced further, from 44 to 2, the RMSE for the IT-net model increases to 0.0108 yet has a lower prediction error than the 3-PC model. Plots of a representative variable estimated by a 2-11-44 IT-net, a 2-PC model and a 3-PC model are shown in Fig. 2. The estimates obtained from the IT-net barely deviate from the

plots of the actual data whereas there are significant deviations in the estimates obtained from the 2-PC and 3-PC models (for instance, in the range of sample numbers 10 through 40).

***Table 1**: The prediction errors for the linear model and the IT-net. The 2-11-44 IT-net achieves data dimensionality reduction with lower error than the 3-PC model.*

Model	RMSE		Model	RMSE
3-PC Model	0.0281		2-PC Model	0.0383
3-11-44 IT-net	0.0057		2-11-44 IT-net	0.0108
3-33-44 IT-net	0.0060		2-33-44 IT-net	0.0557

Increasing the number of hidden nodes in the IT-net overfits the data, increasing the RMSE in both cases (3-11-44 to 3-33-44; 2-11-44 to 2-33-44). Increasing the number of hidden nodes, in effect increasing the degrees of freedom in the model, allows the neural network to fit the training data more closely. The trained model now contains information about the process along with some of the randomness specific only to the training set. This leads to poor estimates for the testing data. Over-training is inherent to all types of neural networks when the number of hidden nodes is too large.

Figure 2. *Estimates for variable #20 derived by 2-factor IT-net, 2-PC and 3-PC models. Model estimates (dotted line) are compared with the actual data (solid line).*

Sensor measurements that do not reach the control system, due to a sensor failure or a fault in signal delivery, may cause a loss in the control action, leading to a disruption in the process. IT-nets are capable of predicting the values of missing sensors in incomplete data.

The input values for an IT-net are ordinarily optimized so that the prediction error of all the output variables is minimized. When output variables are missing, the input variables are optimized based only on the available variables, i.e., the contribution to the prediction error from the missing variables is ignored during the optimization. In an IT-net, the input values are optimized based on the errors back-propagated through the network. If the prediction errors of missing variables are not back-propagated during input adjustment, the input values obtained are optimal with respect to the remaining available variables. The input values so obtained are fed-forward through the IT-net, generating the optimal (in a least-squares sense) estimates for all variables, including an estimate for the missing variables. In other words, the error-minimization projects the incomplete data onto the hyper-surface of latent variables, thus providing estimates for the missing sensors.

This procedure can be carried out in essentially the same manner for linear PCA and nonlinear IT-nets. In PCA, a data matrix **X** of size (m×n) is decomposed into matrices with independent vectors such that $\tilde{\mathbf{X}}$ (the least-squares estimate of **X**) can be reconstructed by the first few independent vectors, $\tilde{\mathbf{X}} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathrm{T}}$. The principal component model contained in the loading matrix, **V**, is modified to remove the loadings corresponding to the missing variables. The incomplete data is projected onto the subspace spanned by the modified matrix, $\tilde{\mathbf{V}}$. The principal component scores obtained are then re-projected back to the full space of variables using the complete matrix, **V**, producing estimates for the missing variables. The procedure is as follows:

$$\tilde{V} = VW \tag{16}$$

where **V** is the loading matrix with as many columns as the necessary principal components and **W** is a square diagonal matrix with ones (1) along the diagonal except for the positions corresponding to missing variables, which are set to zero (0).

$$\tilde{x} = X\tilde{V}\left(\tilde{V}^T\tilde{V}\right)^{-1} \tag{17}$$

where $\tilde{\mathbf{x}}$ are the projections of the matrix **X** (incomplete data) on the subspace of $\tilde{\mathbf{V}}$.

$$\hat{X} = \tilde{x}V^T \tag{18}$$

where $\hat{\mathbf{X}}$ is the reconstructed data matrix which now includes estimates for the missing variables.

A group of five variables (#16 to #20) were tested as simultaneously missing sensors. The RMSE is calculated only for the estimates of the missing variables. We emphasize that the IT-nets in this example are trained on complete measurement patterns and only *tested* on incomplete ones. The tests on the IT-nets and the PC models (Table 2) show that the prediction is much better with the 2-11-44 IT-net (RMSE = 0.0212), than with a 3-PC model (RMSE = 0.0533). This is another indication that two non-linear latent variables explain the non-random variation in the data better than three linear latent variables. As was the case earlier, an excessive number of hidden nodes leads to deteriorating performance of the 2-input IT-net (though the 3-input IT-net continues to perform well). Fig. 3 shows the estimates of the measured values for the missing variable #20 using a 2-11-44 IT-net, a 2-PC model and a 3-PC model.

***Table 2**: The RMSE of the missing variables #16 to #20. The results indicate that the 2-11-44 IT-net provides a much better estimate of the missing sensors than a linear 3-PC model.*

Model	RMSE	Model	RMSE

3-PC Model	0.0533		2-PC Model	0.0588
3-11-44 IT-net	0.0134		2-11-44 IT-net	0.0212
3-33-44 IT-net	0.0089		2-33-44 IT-net	0.0870

This EPS image does not contain a screen preview.
It will print correctly to a PostScript printer.
File Name : F6-v20miss.eps
Title : v20plots.eps
Creator : MATLAB, The Mathworks, Inc.
CreationDate : 09/09/96 11:32:39
Pages : 1

***Figure 3**. Sensor replacement when variables #16 to #20 are missing. Estimates for variable #20 (dotted lines), derived by 2-factor IT-net, 2-PC and 3-PC models, are compared with the actual data (solid lines).*

Process measurements collected over time can encounter problems such as corrupted readings due to temporary faults in measuring devices, unavailable readings due to device failure and loss of data during storage and retrieval from databases. All of these situations lead to sparse pockets of unreliable or unavailable data for different variables at different times. The measurement data

matrix is usually dense but incomplete. In most cases, the situation is remedied by substituting the missing elements by their local means or simply by zeros. However, IT-nets have the capability of summarizing information from incomplete data, without the need for substitute values, provided that the remaining variables contain sufficient redundancy. Remarkably, IT-nets can be trained with incomplete data.

An IT-net optimizes its input values so that the estimation error of the corresponding output pattern is minimized. When there are missing elements in the output pattern, the objective function is optimized based only on those values that are available. The overall objective function—without the contribution from the missing data values—is minimized to obtain the optimal network parameters. Thus, the missing data values are filled in with estimated values such that the orthogonal distance from the nonlinear principal surface is minimized for all patterns.

Concluding Remarks

Reduction of data dimensionality is an important step towards understanding complex processes. By introducing the concept of input training, we have developed a new type of network, called IT-net, for reducing data dimensionality. The IT-net approach works by extending backpropagation training to network inputs. Using IT-nets for nonlinear PCA is analogous to using ordinary feedforward networks for nonlinear regression. Much of the knowledge on constructing and training backpropagation networks can be applied to IT-nets for better performance.

IT-nets can model nonlinear correlation among variables with fewer latent variables than linear PCA, hence are preferable for reducing the dimensionality of data containing many variables. Process data containing a large number of correlated variables can be reduced to a smaller set of variables through data dimensionality reduction methods. The IT-net is a nonlinear method for data dimensionality reduction and a useful tool for process monitoring tasks, such as missing sensor replacement, gross error detection and rectification. As shown in our tests, IT-nets are far more effective than linear PCA in the detection, identification, and rectification of gross errors as well as in sensor replacement tasks. IT-nets can even be trained with incomplete data; this is particularly useful when process data are scarce or their acquisition is costly.

Acknowledgments

This work was supported by the Electric Power Research Institute (Dr. Martin Wildberger; RP 8017-4).

References

MacGregor, J., "Multivariate Statistical Methods for Monitoring Large Data Sets from Chemical Processes," paper 164a, AIChE Meeting, San Francisco, CA (1989).

Raich, A. and A. Cinar, "Statistical Process Monitoring and Disturbance Isolation," Midwest Process Control Workshop, West Lafayette, IN (1994).

Wise, B.M. and N.L. Ricker, "Upset and Sensor Failure Detection in Multivariate Processes," paper 164b, AIChE Meeting, San Francisco, CA (1989).

Stephanopoulos, G.N., and H. Guterman, "Pattern Recognition in Fermentation Processes," paper 163, ACS Meeting, Miami Beach, FL (1989).

Strang, G., *Linear Algebra and Applications*, Academic Press, New York, NY (1988).

Jolliffe, I.T., *Principal Component Analysis*, Springer-Verlag, New York, NY, 122 (1986).

Tan, S. and Mavrovouniotis, M. L., 1995, "Reducing data dimensionality through optimizing neural network inputs," *AIChE J.* 41: 1471–1480.

Cybenko, G., "Approximation by Superpositions of a Sigmoidal Function," *Math. of Control, Signals, and Systems*, **2**, 303 (1989).

MICHAEL L. MAVROVOUNIOTIS is Associate Professor of Chemical Engineering at Northwestern University. He received his Ph.D. from the Massachusetts Institute of Technology in 1989. His recent teaching has included Process Design, Statistics, and Separations. He is the recipient of the Woody Everett Award of ASEE, two Best Paper awards from *Computers and Chemical Engineering*, and the Ted Peterson award of AIChE (CAST Division).