Chapter 5

Process Forecasting, Modeling, and Control of Time-Dependent Systems

5.1 Introduction
5.2 Data Compression and Filtering
5.3 Recurrent Networks for Process Forecasting
- A. Process Modeling
- B. Network Architecture
- C. Error Correction

5.4 Illustrative Case Study : Development of a Time-Dependent Network for Predictive Modeling of a Batch Fermentation Process.
- A. Introduction
- B. Objectives and Tasks : Predictive Modeling of Batch Fermentation
- C. Experimental Approach and Results
  - 1. Experimental Set-up
  - 2. Experimental Design Strategy
  - 3. Experimental Results and Data Files
- D. Neural Network Architecture
  - 1. Overall Network Architecture
  - 2. Data-Compression Network
    - a. Network Design
    - b. Network Training
    - c. Network Performance
  - 3. Growth-Phase Classification Network
    - a. Network Design
    - b. Network Training and Results
  - 4. Fermentation-Processing Network
    - a. Network Design
    - b. Network Training and Results
    - c. Illustrative Predictions
      - c1. Recalling an Experimental Run from the Training Data
      - c2. Forecasting the Cell Concentration with Extrapolation from Training Data
- E. Conclusions

c.2

LD
5635
V856
1995
B384
v.2
c.2

5.5 Illustrative Case Study: Tennessee Eastman Plantwide Control Problem
- A. Introduction
- B. Development of Preliminary Control Strategy
- C. Replacing Three Feed Rate Outer-Loop Controllers with One Supervisory Controller
- D. Neural Network Modeling of the Reactor
  - 1. Feed-Composition Network
    - a. Network Architecture
    - b. Network Training and Results
    - c. Illustrative Predictions of Feed-Composition Network
  - 2. Reactor-Pressure Network
    - a. Network Architecture
    - b. Network Training and Results
  - 3. Attachment of Feed-Composition Network to the Reactor-Pressure Network
- E. Conclusions

5.6 Neural Networks for Process Control
- A. Overview of Process Control Applications
- B. Direct Network Control
  - 1. Direct Network Control Scheme
  - 2. An Example of Commercial Implementation: The Intelligent Arc Furnace (IAF) Controller
- C. Inverse Network Control
  - 1. Inverse and Forward Modeling by Neural Networks
  - 2. An Illustrative Example of Inverse and Forward Modeling: Neutralization Operation in a Continuous Stirred-Tank Reactor
  - 3. Inverse Network Control Schemes for the Neutralization Operation
- D. Model-Based Control
  - 1. Model-Based Approaches to Process Control
  - 2. Internal Model Control
  - 3. Model-Predictive Control
- E. Neural-Fuzzy Control
  - 1. Neural-Fuzzy Control of a Batch Fermentation Process Using Predictions of Future Operating Conditions
  - 2. Neural-Fuzzy Control of a Bioreactor System Using Temporal Pattern Recognition
    - a. Temporal-Pattern-Recognition Neural Networks
    - b. Fuzzy Control Algorithm
  - 3. Neural-Fuzzy Control of High-Cell-Density Cultivation of Recombinant Escherichia Coli Using a Fuzzy Neural Network.
    - a. Structure of a Fuzzy Neural Network
    - b. Implementation of a Fuzzy Network

5.7 Chapter Summary

Nomenclature
Practice Problems
Appendices
References and Further Reading

5.1 Introduction

There are two major applications of neural networks for process forecasting, modeling, and control of time-dependent processes: (1) process optimization; and (2) adaptive process control. Modeling chemical and biological processes using conventional techniques is often difficult because the processes are very nonlinear and complicated. For example, many chemical processes have a large number of reacting components and follow very complex reaction mechanisms. The neural network approach provides a method to develop a dynamic model accounting for instabilities and unsteady-state operating conditions that often occur in chemical systems.

One problem in the neural network approach is noise. Many of the input signals present in these systems are very noisy and must be smoothed out to obtain the best results within the moving-window network (Section 2.6.B.2). Raich et al. (1992) have compared the training and recall of time-dependent networks with and without noise in the input signals. They show that large quantities of noise can degrade prediction accuracy, while smaller amounts have minimal effects. For systems that have noisy periods in the input signals, we propose the use of a data-compression network (Section 5.2) to smooth out the incoming signals before forwarding them to the time-dependent network . The

noise can also be handled using a larger window for the input signals entering the time-dependent network, which has a data-filtering capability. However, this latter method requires a larger network structure, making the training algorithm less efficient.

Section 5.3 introduces the basic principles of recurrent networks for process forecasting. Section 5.4 presents an illustrative case study, the practical application of neural networks to the predictive modeling of an experimental fermentation process. This application involves using a data-compression network (Sections 2.6.A and 5.2), a classification network (Chapter 3), and a recurrent network (Sections 2.6.C and 5.3). Section 5.5 describes another illustrative case study, dealing with the Tennessee Eastman plantwide control problem, a well-known industrial problem for testing control strategies in a complex, chemical reacting process (Downs and Vogel, 1993). We illustrate the development of a neural network predictor for determining the control setpoints in a supervisory control system for the Tennessee Eastman problem. Section 5.6 gives a survey of the recent developments of neural networks for process control.

5.2 Data Compression and Filtering

The neural networks used for data compression and data filtering are divided into the two main categories: signal-processing networks and image-processing networks. Because image processing is not particularly important to bioprocessing and chemical engineering, we will not include it in this text. Signal processing, on the other hand, is very important

in preprocessing sensor data prior to using them in other prediction algorithms or process-control strategies. The data-compression capabilities of the neural network reduce the amount of random noise in the input signal. The neural network also filters gross errors in data resulting from sensor-malfunctions, drift, and bias. The neural network approach simplifies noise reduction by replacing the standard procedures of data rectification, gross-error detection, failure identification, and sensor-value replacement estimation with a single feedforward network (Kramer, 1991).

The most commonly used neural network for signal processing is the autoassociative backpropagation network. As described in Section 2.6.A, an autoassociative network correlates the input pattern with itself and we provide only the input vector to the network which automatically defines the output vector for the user. Both the input and output patterns for a signal are represented as vectors having *m* incoming data points.

$$\mathbf{In\ (m) = Out\ (m)} \tag{5.1}$$

We obtain the input vector, **In(m)**, from process measurements taken over a given time span defined by *m* data points. The input signal has three distinct components: a pure signal (p), measurement noise (n), and measurement errors (e). Thus, we may represent the input vector as a linear combination of the three components:

$$\mathbf{In\ (m) = In_p(m) + In_n(m) + In_e(m)} \tag{5.2}$$

The objective of filtering a process signal is to map only the pure signal (**In_p(m)**) onto the output vector, while removing all measurement noise (**In_n(m)**) and measurement errors (**In_e(m)**).

Figure 5.1 shows the standard architecture for a three-layer autoassociative network (Section 2.6.A) used as a data-compression network, where the vectors **In(m)**, **B(n)**, and **Out(m)** represent the input layer, hidden layer, and output layer, respectively. As shown, the network maps the noisy input vector of *m* elements onto a smaller intermediate vector, known as a bottleneck vector having *n* elements (where $m > n$). The smoothed noise-free output vector has *m* elements.

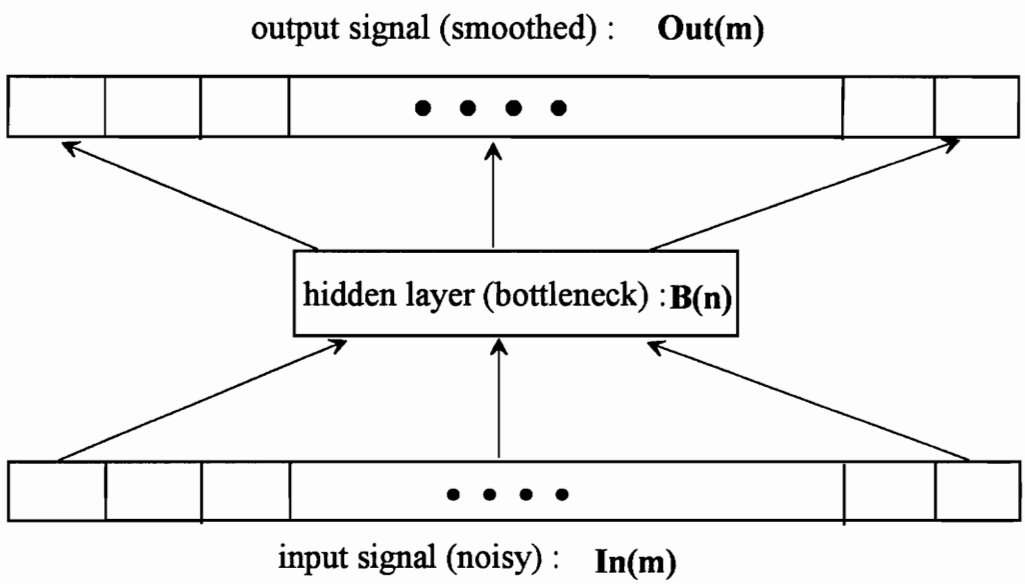


Figure 5.1 : A three-layer, data-compression network.

An important consideration in developing a data-compression network is selecting the optimal compression ratio, discussed in Section 2.6.A. Once again, we define the compression ratio as the ratio of elements in the input vector (m) to elements in the bottleneck vector (n).

$$\text{compression ratio} = \frac{m}{n} \tag{5.3}$$

This compression ratio is directly related to the amount of noise eliminated in the process signal (e.g., higher compression ratios leads to greater noise reduction). We also refer the reader back to Figure 2.35 which is a standard plot of the RMS error versus compression ratio used to determine the optimal compression ratio.

Selecting the size of the input vector is also important in developing an effective data-compression network. The vector size is initially constrained by the sampling frequency of the measuring device and the overall processing time. These two factors are fixed for a given processing system and offer no flexibility in network design without major adjustments to the process or the measurement equipment. We suggest using a minimum of ten nodes in the input layer with a compression ratio of at least 2.0 for systems that have a significant quantity of noise.

The data supplied to the network within each input vector should also be as linear as possible. For nonlinear systems, we may represent the signal as a series of linear

segments. This is not always possible in practice, but one should make an effort to minimize the nonlinearity of a given input vector. For example, Figure 5.2 illustrates a nonlinear input signal divided into seven time segments (t1, t2,.., t7).

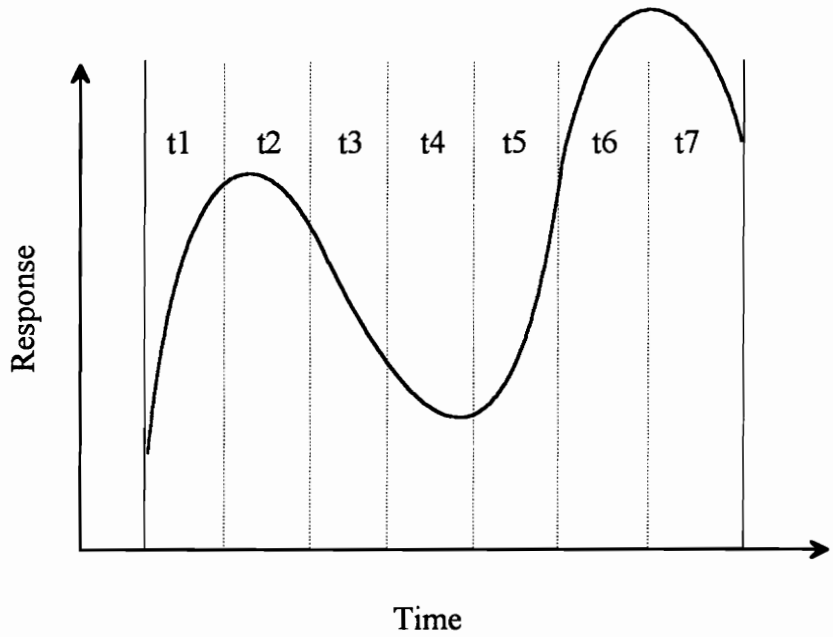


Figure 5.2. The partitioning of a signal into a multiple linear input segments for a data-compression network.

If the measuring device permits, the time intervals of the input vector should not exceed those of the seven time segments. If, for example, we double that interval to include both t1 and t2 in the input vector, the network will have to store increasing linearly, decreasing

linearly, and parabolic functions within the bottleneck layer. That will require more nodes in the hidden layer, decreasing the compression ratio and lowering of effectiveness of the network in performing its primary task - filtering noise. If instead, we maintain an adequate compression ratio, the network will not be able to retain all the relevant information in the input signal and will treat the nonlinearity as noise. This will compress the nonlinear signal, producing a linear representation of a nonlinear region. Further expanding the time interval to include additional time segments will, naturally, deteriorate the effectiveness of the network.

Other factors, such as the internal dynamics of the system, can also play a significant role in network development, particularly when the network is being used for process control, as in modeling dead time.

Other architectures used in data compression are learning-vector-quantization networks, self-organizing map networks, and recirculation networks (NeuralWare, 1993).

5.3 Recurrent Networks for Process Forecasting

As mentioned in the introduction, many of the advanced control and optimization techniques require an accurate time-dependent model of the process. Developing traditional models based on first principles is typically very time-consuming and it is difficult to achieve accurate results. It is especially difficult for biological and chemical processes which normally have large variable sets for processes that are not well

understood. To bypass many of the problems associated with traditional models, recurrent networks (Section 2.6.C) have been employed to accurately model the process (Blum et al., 1992; Raich et al., 1992; You and Nikolaou, 1993).

The standard approach to the neural network modeling of time-dependent processes uses a "black box" model, which requires no previous knowledge of the system or process. This type of model utilizes previously recorded input/output processing patterns to predict future responses to a given set of operating conditions. Since we are mainly interested in process forecasting in this chapter, we will use a simple process model, similar to that presented by Blum et al. (1992), to illustrate the prediction of time-dependent systems.

A. Process Modeling

Figure 5.3 shows the standard process model consisting of three main variable groups: (1) manipulated variables, $u_i(t)$ ($i = 1$ to m); (2) measured variables at past time intervals, $y_j(t)$ ($j = 1$ to n); and (3) predicted response of the process, $\hat{y}_k(t)$ ($k = 1$ to n). In the figure, the arrow passing diagonally through the process model represents the error signal, $e_k(t) = \hat{y}_k(t) - y_k(t)$, used for model development (e.g., network training).

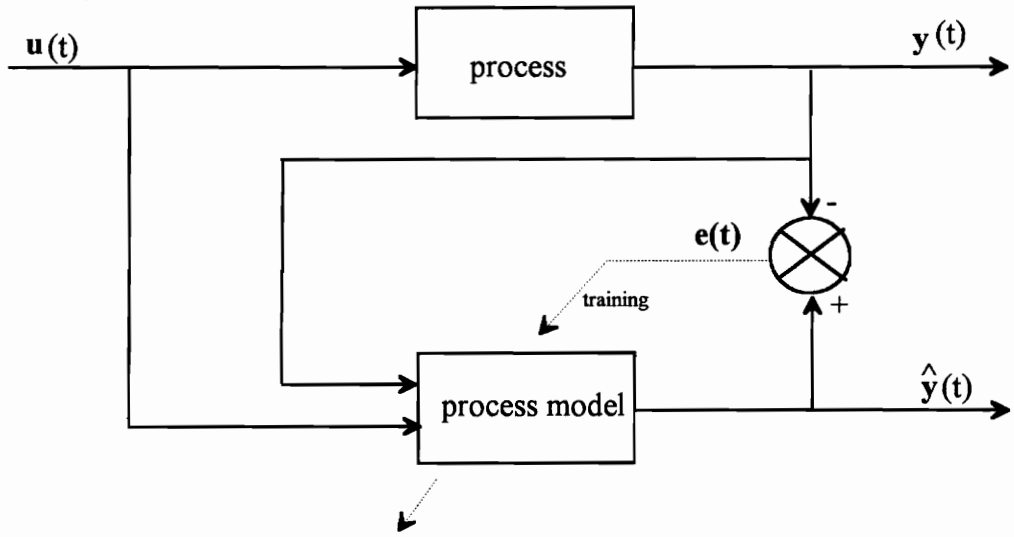


Figure 5.3. An illustrative model of a time-dependent processing system (Blum et al., 1992).

The vectors for the four variables, **u**(t), **y**(t), $\hat{\mathbf{y}}$ (t), and **e**(t), consist of m manipulated and n measured variables, and can be represented as:

$$\mathbf{u}(t) = \begin{pmatrix} u_1(t) \\ \bullet \\ \bullet \\ \bullet \\ u_m(t) \end{pmatrix} \quad \mathbf{y}(t) = \begin{pmatrix} y_1(t) \\ \bullet \\ \bullet \\ \bullet \\ y_n(t) \end{pmatrix} \quad \hat{\mathbf{y}}(t) = \begin{pmatrix} \hat{y}_1(t) \\ \bullet \\ \bullet \\ \bullet \\ \hat{y}_n(t) \end{pmatrix} \quad \mathbf{e}(t) = \begin{pmatrix} e_1(t) \\ \bullet \\ \bullet \\ \bullet \\ e_n(t) \end{pmatrix} \tag{5.4}$$

The vector for the predicted future responses, $\hat{\mathbf{y}}$ (t+Δt), is a function of the manipulated variables, **u**(t), and the measured variables at past time intervals, **y**(t). The functional operator f ( ) represents the system shown in Figure 5.3, to which we will apply the neural network.

We represent the functional form of any predicted response variable, $\hat{y}_k$ (t+Δt) (k = 1 to n), as:

$$\hat{y}_k\ (t+\Delta t,\cdots,t+p_k\Delta t) = f\ [y_1(t-s_1\Delta t)\cdots y_1(t),\cdots,y_n(t-s_n\Delta t)\cdots y_n(t),\ u_1(t-r_1\Delta t)\cdots u_1(t),\cdots,u_m(t-r_m\Delta t)\cdots u_m(t)] \tag{5.5}$$

where Δt is a fixed time interval, and r_i (i = 1 to m), s_j (j = 1 to n), and p_k (k = 1 to n) are the numbers of time intervals used in the process model for the manipulated variables, the measured variables at past time intervals, and the predicted future responses, respectively. Therefore, the total time spans, τ's, for the input and output windows of the manipulated, the measured, and the predicted variables are:

$$\tau_{ui} = r_i \cdot \Delta t \quad (i = 1 \text{ to } m) \tag{5.6}$$

$$\tau_{yj} = s_j \cdot \Delta t \quad (j = 1 \text{ to } n) \tag{5.7}$$

$$\tau_{\hat{y}k} = p_k \cdot \Delta t \quad (k = 1 \text{ to } n) \tag{5.8}$$

B. Network Architecture

Figure 5.4 shows the standard network architecture for modeling time-dependent processes. The input vectors, **u**(t) and **y**(t), that affect the process forecasting of all measured variables are included in the input section of the neural network. This section makes up the first three layers of our network architecture: (1) the individual values for each input variable at every time period specified within the moving window; (2) the subnetworks that contains each manipulated and each measured variable as a separate group; and (3) the subnetwork that contains the group of all manipulated variables and the group of all measured variables.

The input section proceeds through the hidden layers, which vary in size and structure based on the nonlinearity of the system. In general, process forecasting requires at least two hidden layers to obtain good recall and generalization of data sets.

The output layer of the network contains the future predictions of the measured variables, $\hat{\mathbf{y}}$ (t+Δt). The number of future prediction time intervals, p_k, or time span of the moving window, τ_{yk}, is normally based on the specific application. We can use one output time response for each measured variable, $p_k = 1$ (k = 1 to n) and predict values at many future times by continually recirculating predicted values to the input layer and making subsequent predictions. This process will cause prediction errors to propagate from one cycle to the next, resulting in poor long-range forecasting. The alternative is to have the network predict multiple future time periods for each measured variable, $p_k > 1$ (k=1 to n).

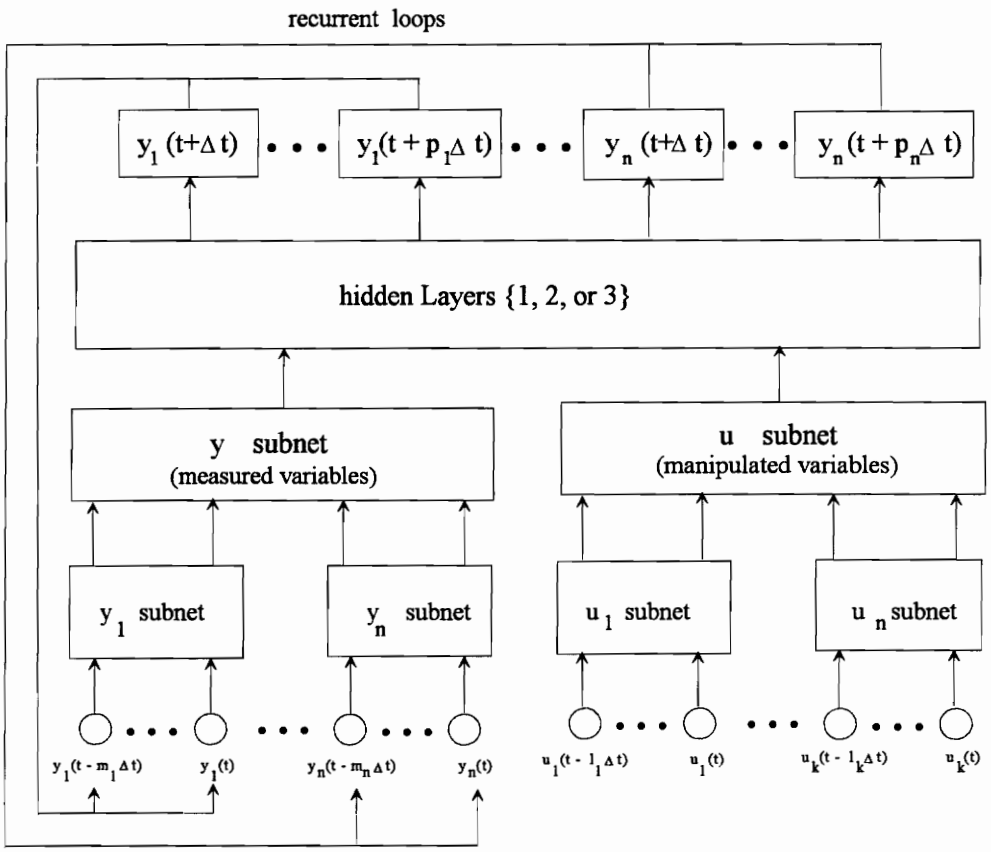


Figure 5.4. The standard recurrent network architecture for time-dependent process modeling.

This method reduces the propagation of the prediction error from one time period to the next, but will also require larger and more complex networks. Therefore, we must be careful to select the output-vector size to minimize prediction errors, while maintaining an efficient network.

We note in Figure 5.4 that there is a time lag between the output signal $\hat{y}_1$ (t+Δt) and the input signal $y_1(t)$ on the same recurrent loop. As mentioned in Section 2.6.C, we call the recurrent network architecture of Figure 5.4 *a time-lag recurrent network* or *an externally recurrent network* (Werbos, 1988; Su et al., 1992b). In applying neural networks to modeling an experimental fermentation process, Karim and Rivera (1992a,b), and Rivera and Karim (1992) employ another class of recurrent network, called *the Grossberg-Hopfield recurrent network* (Section 2.6.C). The latter is similar to Figure 5.4, except that the input-output dynamics of each node are governed by a first-order ordinary differential equation (Pearlmutter, 1989; NeuralWare, 1993).

C. Error Correction

As discussed, one major problem with recirculating predicted values is error propagation. That is, we may obtain small prediction errors for each single training example, but still observe large deviations from actual process measurements at processing times in the future. This is especially true when processing data that do not resemble the training examples (i.e., when network predictions require extrapolation).

There is a basic method to dampen the error associated with processing data that require extrapolation. We first assume that the average error between measured values and predicted values for a fixed number of previously recorded time intervals is constant throughout the process. This is not always the case, as shown in Section 5.4 where the

error follows a skewed Gaussian density function, but is normally offset in one direction. Although more advanced techniques are available for such cases, we will limit our discussion here to the assumption of constant error. A simple example to illustrate error dampening uses the average of the ten previously recorded measurements. The average error, e_{avg}, is given as:

$$e_{avg} = \text{avg}\left[\hat{y}_k(t) - y_k(t), \bullet\bullet\bullet, \hat{y}_k(t-9\Delta t) - y_k(t-9\Delta t)\right] \tag{5.9}$$

The adjusted prediction, $\hat{y}_{k,adj}(t+\Delta t)$, is represent as :

$$\hat{y}_{k,adj}(t+\Delta t) = \hat{y}_k(t+\Delta t) + e_{avg} \tag{5.10}$$

We must note that this type of error adjustment may not be accurate if the training examples for the network do not result from a good experimental design strategy, or if knowledge of the process outside of normal operating regions is limited.

The overall architecture of the neural network in Figure 5.4 is valid for any type of neural network discussed throughout this book, including backpropagation networks, radial-basis-function networks, etc. In other words, the function $f(\,)$ can represent any training algorithm we wish.

5.4. Illustrative Case Study : Development of a Time-Dependent Network for Predictive Modeling of a Batch Fermentation Process.

A. Introduction

This case study deals with the process forecasting and modeling of fermentation processes. As an example, we will consider the batch fermentation process in which yeast is the primary product and ethanol is a poison produced from the ethanol-tolerant yeast strain, saccharomyces cerevisiae. Improving process forecasting, process control, and process optimization techniques of this system can lead to increased production rates and higher product quality.

Bioreactions are complex chemical systems that are usually nonlinear in nature. The overall reaction mechanism is much more complicated than that of a nonbiochemical reaction because of the genetic and enzymatic control mechanisms involved. These control mechanisms are fundamentally programmed by the DNA sequence of the organism (Baily and Ollis, 1993; Shuler and Karigi, 1992). The difficulties present in using conventional numerical techniques for modeling such a complex system makes the neural network approach ideal.

In batch fermentation, there are four phases of cell growth: induction, growth, stationary, and death, as illustrated in Figure 5.5 (Fogler, 1992). In the induction phase, the cells begin to adapt to their new environment and minimal reproduction occurs. Most of the cell growth occurs in the growth phase, when cells are dividing at their maximum

rate. The cell growth rate is proportional to the cell concentration in this phase. In the stationary phase, the cell growth rate is virtually zero due to depletion of nutrients (organic feed) or crowding of cells. Finally, in the death phase, the cells begin to die and the growth rate becomes negative as a result of the lack of nutrients or the presence of poisonous byproducts from the reaction.

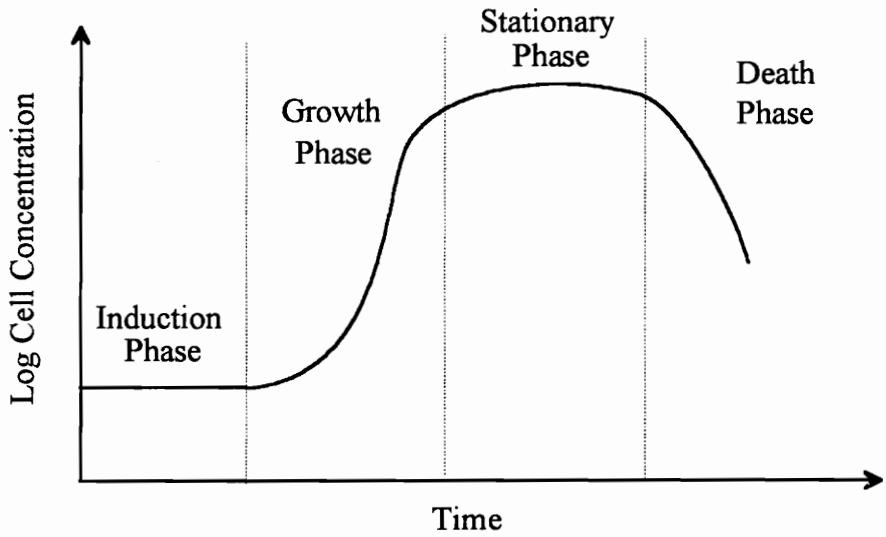


Figure 5.5. The four phases of the batch-fermentation cell growth.

Most predictive techniques for fermentation modeling are based on kinetic studies using the Monod model or a modified form of the Monod model considering additional inhibition reactions. We can represent the overall reaction as:

$$\text{Glucose (G)} + \text{Cells (C)} \rightarrow \text{Ethanol (E)} + \text{More Cells (C)} \tag{5.11}$$

where an organic feed or substrate (e.g., glucose) comes in contact with cells to produce ethanol and more cells.

The operating conditions of the fermentation unit determines the reaction path in the production of ethanol and cells. For example, under *aerobic* conditions (i.e., in the presence of oxygen), cells will be the primary product and ethanol represents a "poison" that inhibits the cell growth. Under *anaerobic* conditions, ethanol takes precedence and equation (5.11) becomes:

$$\text{Glucose (G)} \xrightarrow{\text{Cells (C)}} 2\cdot\text{Ethanol (E)} + 2\cdot CO_2 \uparrow \tag{5.12}$$

We refer to equation (5.11) as *the microbial or cell-growth fermentation* and equation (5.12) as the *enzymatic or alcoholic fermentation*.

The reaction rate for the Monod kinetic model for a poison-free system is given as:

$$r_c = k \frac{[C]\,[G]}{[G] + K_M} \tag{5.13}$$

where r_c is the production rate of yeast cells, k is the reaction rate constant, K_M is the Monod constant, [G] and [C] are the concentrations of glucose and cells, respectively.

When a poison, such as ethanol, is included in the model, the reaction rate is given as (Levenspiel, 1980; Mairolle et al., 1983, 1984) :

$$r_c = k\left(1 - \frac{[E]}{[E]^*}\right)^n \frac{[C]\,[G]}{[G] + K_M} \tag{5.14}$$

where [E] is the concentration of ethanol and $[E]^*$ is the limiting ethanol concentration at which no cells are present, and n is an empirical parameter for poison limitation.

Monod types of models only account for the *initial* concentrations of cells, glucose, and ethanol in predicting the reaction path. Moreover, Monod models include all operating conditions of the fermentation unit within empirical parameters (e.g., k, K_M, and n) that must be experimentally determined. Therefore, the model *cannot* account for changes in operating conditions during a process without adjusting those parameters.

B. Objectives and Tasks : Predictive Modeling of Batch Fermentation

We wish to develop a time-dependent network for the predictive modeling of batch fermentation using saccharomyces cerevisiae. To do so, we will use a hybrid system, linking three different types of networks, for process forecasting. First, an autoassociative backpropagation network for data compression and filtering will smooth a continuous cell-concentration signal. A radial-basis-function classification network will then identify the four phases of the fermentation process (induction, growth, stationary and death

phases). Finally, a recurrent time-dependent network attached to the growth-phase output signals will forecast future cell concentrations. We will also incorporate a Taguchi-based experimental design technique (Taguchi, 1986) to carry out pilot-scale, batch fermentations for a wide range of initial cell concentrations, initial glucose concentrations, temperatures, pH values, agitation rates, and aeration rates.

Our goal is to develop a realistic network structure that includes the primary manipulative variables of the process in addition to the feed and product concentrations. We will utilize a moving-window network structure for time-dependent processes to track the fermentation as it proceeds. This recurrent network structure also provides a mechanism for predicting the response to setpoint changes or ramping variations in any of the manipulative variables during the process. Therefore, we can perform rapid process optimizations for any desired output. In addition, the network can be used in conjunction with an adaptive control algorithm to maintain desired levels of a given product. The framework of the network developed here can easily be expanded to fed-batch processes and other bioconversion processes.

Table 5.1 identifies the main control and measured variables used in fermentation processes as represented by Endo and Nagamune (1987) in their development of an expert system. The controlled variables are divided into setpoint control and optimal control based on standard operations. The pressure, $pO_{2,in}$, aeration rate, and agitation rate are controlled at a setpoint determined from their optimal response. The temperature, pH

value, and feed rate are controlled based on their optimal conditions at any given time in the process.

Table 5.1 : Standardized measured and control variables in the characterization of fermentation processes (Endo and Nagamune, 1987).

Measured variables	
temperature	
pressure	
pH-Value	
dissolved oxygen content	
aeration rate	
aAgitation rate	
pO_2 in	
pO_2 out	
pCO_2 in	
pCO_2 out	
working volume	
torque	
turbitity	on-line
product concentration	on-line
substrate concentration	on-line
specific rates	on-line

Controlled variables	
temperature	optimal control
pressure	setpoint control
pH value	optimal control
pO_2 in	setpoint control
aeration rate	setpoint control
agitation speed	setpoint control
feed rate	optimal control

Our network structure will consist of four control variables and two measured response variables. The control variables are temperature, pH, aeration rate, and agitation rate, but can easily include pressure, $pO_{2,in}$, and feed rate as well. The temperature, pH value, agitation rate, and aeration rate will be set in our experimental design strategy (Section 5.4.C.2). The pressure and $pO_{2,in}$ will be set at normal operating levels and the feed rate does not apply to a batch system. The measured response variables consist of the glucose and the cell concentrations. The initial concentrations of substrate and product are fixed at the values defined in our experimental design strategy. We do not measure ethanol concentration directly in our experiments, but it appears indirectly within the network's nodes as a function of the given manipulative variables and the cell-growth response curve. Future work should include ethanol concentrations, especially when operating under anaerobic conditions. The dissolved oxygen content can also be added easily to the network, although the agitation and aeration rates adequately define the dissolved oxygen operating levels.

C. Experimental Approach and Results

1. Experimental Set-up

Figure 5.6 shows the schematic diagram of the fermentation unit and control system used in our experiments. A Camile controller (Dow Chemicals) monitors and controls this

process. The pressure, aeration rate, and agitation rate are set in the controller for each run. The pH and temperature are monitored by sensors and controlled by PID control algorithms. A stepper motor controls the pH using acid/base additions. To maintain the defined temperature profile, the fermentation unit is jacketed with water flowing in contact with steam and cooling water . The dissolved oxygen content is monitored on-line and recorded by the Camile system. Spectrophotometric methods and off-line test kits monitor the cell concentration and glucose concentration, respectively.

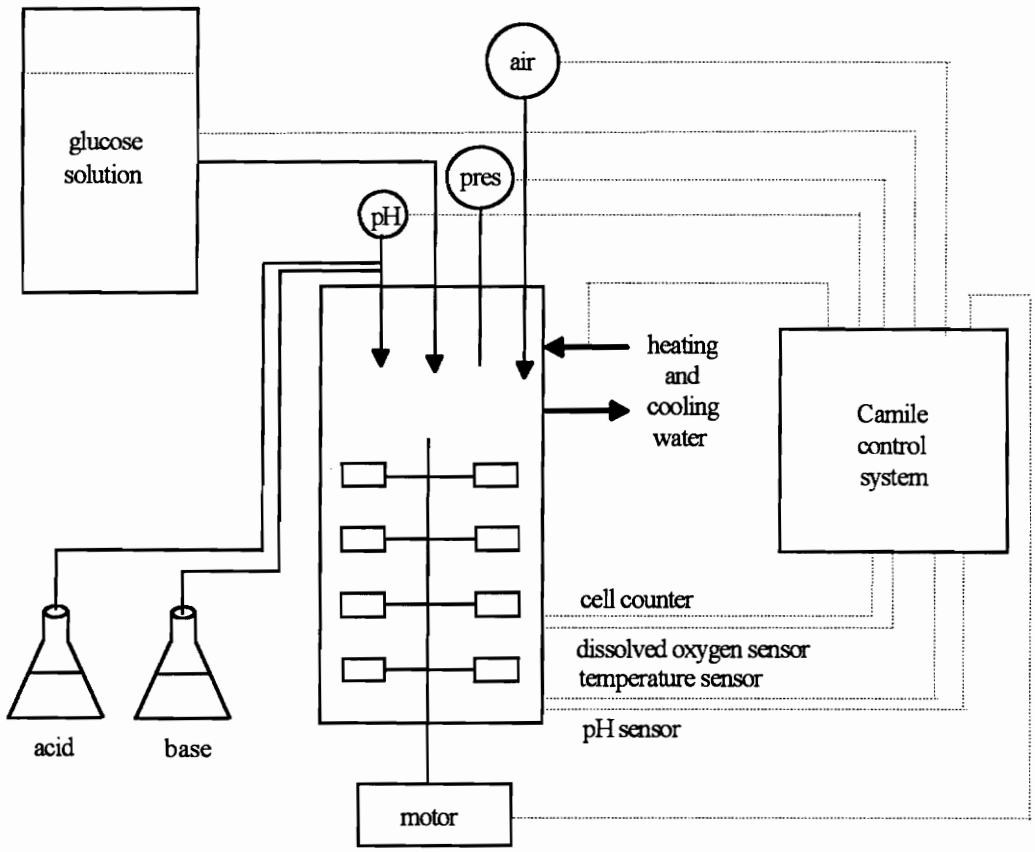


Figure 5.6. A schematic diagram of the fermentation unit and control system .

The fermentation process is carried out in 20-liter batches of deionized water. The following formulation is used to provide the required energy sources and nutrients:

Glucose	:	50, 75, and 100 grams/liter
Yeast Extract	:	3.0 grams/liter
Peptone	:	5.0 grams/liter
Dextrin	:	0.25 grams/liter
Maltose	:	1.13 gram/liter
Agar	:	1.33 grams/liter

2. Experimental Design Strategy

Our experimental design strategy focuses on *a preliminary design* that includes experimental conditions listed in Table 5.2 spanning the *entire* feasible operating range for *both* the cell-growth fermentation, equation (5.11), *and* the alcoholic fermentation, equation (5.12). Our broad-based experimental design attempts to avoid extrapolation problems associated with having fermentation conditions outside of normal operating conditions. If only normal operating conditions are used to train the neural network, the network will not function well during process drifts or abnormal conditions. Future work should include *a secondary design* that is targeted around the desired process operating conditions. We highly recommend this multiple-tiered design strategy for neural network modeling of processes that are not presently on-line with a large database. Figure 5.7 shows a 3-dimensional feasible process operating region and the normal operating conditions, with both the *preliminary and secondary design areas* identified.

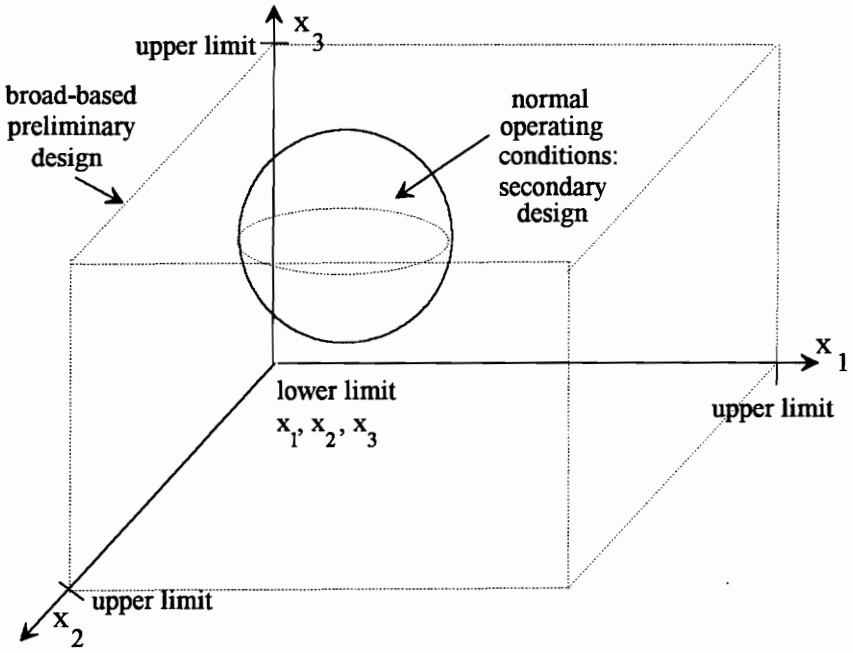


Figure 5.7. A 3-dimensional feasible operating region and the desired operating region.

We begin by combining two Taguchi's $L_9(3^4)$ designs to obtain six experimental categories for the fermentation process, with temperature and pH orthogonal to the other four properties (Taguchi, 1986). We then expand this design after obtaining preliminary results, producing the experimental design shown in Table 5.2. Next, we set the six processing variables, including two initial conditions (glucose and cell concentrations) and four manipulative variables (temperature, pH-value, agitation rate, and aeration rate), at three different levels each within the feasible operating regions. We also modify the aeration rate category after beginning the experimentation to include anaerobic operating conditions (aeration rate equals 0.0 Nm^3/hour).

Table 5.2. The experimental design for fermentation processes.

Trial number	Initial glucose A	Initial cell B	Temperature C	pH D	Agitation rate E	Aeration rate F
1	A2	B3	C3	D2	E3	F4
2	A3	B2	C2	D1	E1	F4
3	A1	B3	C1	D1	E1	F2
4	A2	B3	C3	D3	E2	F3
5	A3	B1	C2	D3	E2	F2
6	A1	B1	C3	D2	E1	F1
7	A2	B1	C2	D3	E3	F1
8	A1	B1	C1	D1	E2	F1
9	A3	B1	C2	D2	E1	F1
10	A2	B2	C3	D1	E3	F1
11	A1	B2	C1	D3	E3	F3
12	A3	B2	C3	D2	E3	F3
13	A2	B3	C2	D1	E1	F2
14	A3	B2	C1	D3	E3	F2
15	A1	B3	C3	D2	E2	F4

A : Initial glucose concentration
- A1 : 50 g/l
- A2 : 75 g/l
- A3 : 100 g/l

B : Initial cell concentration
- B1 : $[C]_0 < 0.30$ g/l
- B2 : $0.30 < [C]_0 < 0.46$ g/l
- B3 : $[C]_0 > 0.46$ g/l

C : Temperature
- C1 : 25 °C
- C2 : 30 °C
- C3 : 35 °C

D : pH
- D1 : 3.5
- D2 : 4.5
- D3 : 5.5

E : Agitation rate
- E1 : 200 rpm
- E2 : 400 rpm
- E3 : 600 rpm

F: Aeration rate
- F1 : 0.0 Nm^3/hr
- F2 : 2.0 Nm^3/hr
- F3 : 3.5 Nm^3/hr
- F4 : 5.0 Nm^3/hr

Table 5.3 compares our work with reported applications of neural networks to fermentation studies in terms of fermentation type, neural network architecture, and network training database. Our work improves upon reported studies by having a broad-based experimental design, considering both cell-growth and alcoholic fermentations.

Table 5.3. Comparison of this work with reported applications of neural networks to fermentation studies.

Investigators	Fermentation type	Neural network architecture	Training database	Remarks
Bocherean et al. (1991)	alcoholic	recurrent I[a]	experimental	
Clearn et al. (1991)	alcoholic	propagation	experimental	
DiMassio et al. (1991)	alcoholic	backpropagation	industrial	filter dynamics included[b]
Glassey et al. (1992)	alcoholic	backpropagation	simulated	filter dynamics included
Karim and Rivera (1992a,b)	alcoholic	backpropagation and recurrent II[c]	experimental	
Kurtanjek (1994)	alcoholic	backpropagation	experimental	
Lant et al. (1990)	alcoholic	backpropagation	experimental	
Latrille et al. (1990)	alcoholic	backpropagation	experimental	
Linko and Zhu (1991)	alcoholic	backpropagation	simulated	
Linko and Zhu (1992)	alcoholic	recurrent I	simulated	
Syo and Tsao (1993)	cell-growth	backpropagation	experimental	
Linko and Zhu (1992)	cell-growth	backpropagation	simulated	continuous reactor
DiMassio et al. (1992)	call-growth and alcoholic	backpropagation	industrial	filter dynamics included
This work	call-growth and alcoholic	backpropagation and recurrent I	experimental	data compression and growth-phase identification included

a Recurrent I = time-lag or externally recurrent network.
b First-order filter included to represent network dynamics.
c Recurrent II = Grossberg-Hopfield recurrent network (Section 2.6.E).

3. Experimental Results and Data Files

Figure 5.8 presents the smoothed cell-growth curves obtained from our fifteen experiments. The cell concentrations range from plateaus of approximately 5 to 25 grams/liter.

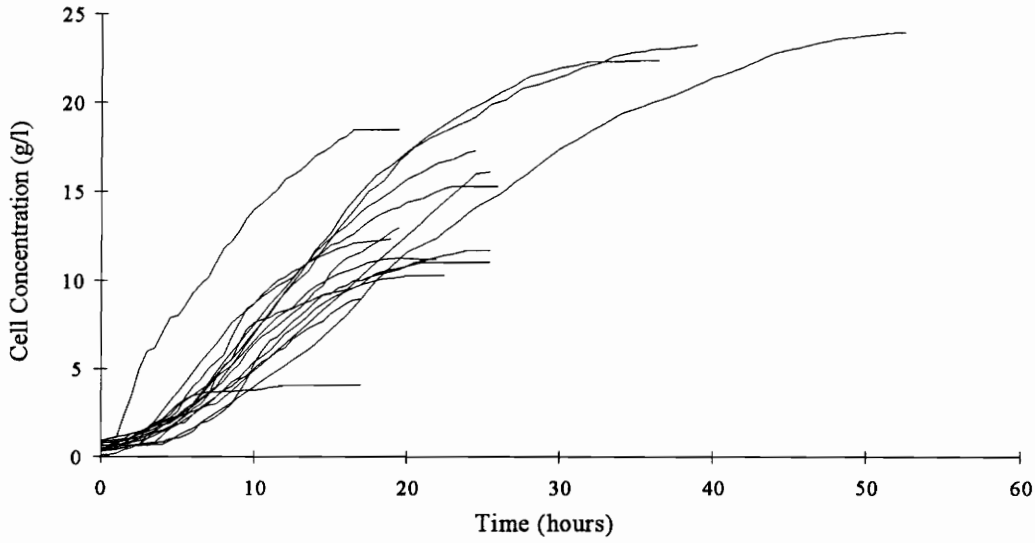


Figure 5.8. The cell-growth curves for fifteen experimental runs in the growth phase.

The complete set of experimental results used for training the network is provided in file *ferm.nna*. Table 5.4 lists the fourteen input variables and one output variable supplied to the network according to input file column. All variables in the data file are

normalized to between 0 and 1 by the normalization factor in the last column. This example uses three time periods, t to t-2Δt, for all input variables except aeration and agitation rates, which are held constant at their setpoints for all experiments.

Table 5.4. The format of the file *ferm.nna* used for training the fermentation-processing network.

Column number	Variable type	Normalized variable	Time period	Normalization factor
1	input	cell concentration	t - 2	
2	input	cell concentration	t - 1	100 g/l
3	input	cell concentration	t	
4	input	glucose concentration	t - 2	
5	input	glucose concentration	t - 1	100 g/l
6	input	glucose concentration	t	
7	input	pH	t - 2	
8	input	pH	t - 1	6.5
9	input	pH	t	
10	input	temperature	t - 2	
11	input	temperature	t - 1	35 °C
12	input	temperature	t	
13	input	aeration rate	setpoint	5 Nm^3/hr
14	input	agitation rate	setpoint	600 rpm
15	output	cell concentration	t + 1	100 g/l

Table 5.5 lists the sequential order in which the experimental runs appear in the data file. The experimental runs are assigned to one of five groups (numbered one to five) randomly and then added to the data file based on generalization error, as described in Section 5.4.D.4. For each group, the experiments are run in order from left to right, and the groups are run in ascending order. Note that the data in file *ferm.nna* is **not** in the same order as our experimental design (Table 5.2).

Table 5.5. The sequential order of experimental runs as they are present in file *ferm.nna* which is used for training the fermentation-processing network.

Group number	Sequential experiment numbers		
1	3	14	6
2	11	5	12
3	8	4	13
4	1	2	9
5	15	7	10

D. Neural Network Architecture

1. Overall Neural Network Architecture

To forecast this fermentation process, we use three neural networks in series, as described in Section 5.4.B (Figure 5.9). Note that the induction, stationary, and death phases are

only identified and are not used for any predictions. This study focuses only on the growth phase, hence the other phases do not have forecasting networks attached to their output category. The smoothed signal from the data-compression network becomes the input to the recurrent time-dependent network, and the classification network only activates the time-dependent network.

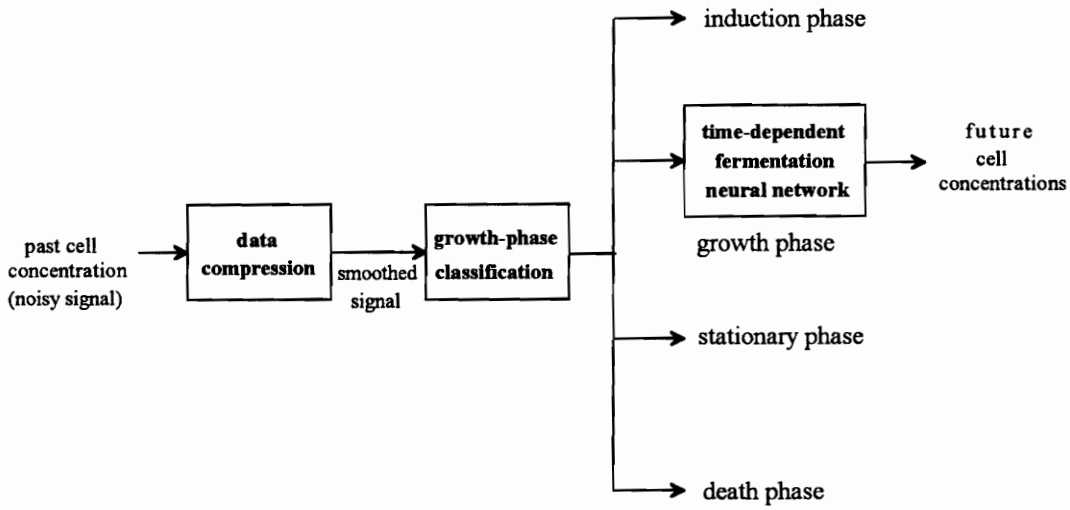


Figure 5.9. The overall network architecture for forecasting future cell concentrations.

Most previously developed recurrent networks for time-dependent processes (e.g., Blum et al., 1992) reduce the noise within their own structure using large moving

windows for the variables being recirculated. An independent data-compression network prior to the recurrent network, however, significantly reduces the size of the window, making the network much more efficient.

The classification network provides a means to keep the network inactive while the process is not operating in the growth phase, which is the phase of interest in this study. The induction, stationary, and death phases follow significantly different behavioral patterns than the exponential growth phase, with cell growth being constant in the induction and stationary phases, and decreasing in the death phase. Therefore, including the other three phases adds more error to the system for a network of similar size. To compensate for this nonlinearity, we must either sacrifice prediction accuracy or increase the size of the network to store more complex input patterns. Increasing the size of the network decreases the efficiency in training the network.

Simutis et al. (1992) and Raju and Cooney (1992) have both stressed the importance of the growth-phase identification in the modeling and interpretation of fermentation kinetics using either an expert system or a neural network. Both studies, however, have not explicitly presented any expert system or neural network architecture for accomplishing this task. We demonstrate here the use of a radial-basis-function network for the growth-phase identification.

2. Data-Compression Network

a. Network Design

The overall structure of the fermentation-forecasting network we present in this section is designed to be used with a continuous processing signal. The autoassociative backpropagation network described in Section 5.2 smoothes the noisy signal retrieved from an on-line cell counter. Since the cell counter was not installed until the final runs of our experiments, the cell concentrations are measured off-line using a spectrophotometer and the data are smoothed by fitting them to a polynomial and interpolating the interior points.

The cell counter measures cell concentrations at two-minute time intervals and the network inputs them into a moving window that spans thirty-minutes. A sampling time of two minutes is selected based on our cell counter's testing frequency and the overall fermentation time. The thirty-minute time span corresponds to the time differential, Δt, used in the recurrent forecasting network. Therefore, the smoothed values at time t are the only data points that proceed to the final forecasting network.

As Figure 5.10 shows, data points at fifteen time intervals enter the input layer and are mapped onto the hidden layer (bottleneck). The more compact pattern stored in the hidden layer is then mapped onto the output layer to obtain a representation of the same size as that in the input layer, but excluding a portion of the noise in the original signal.

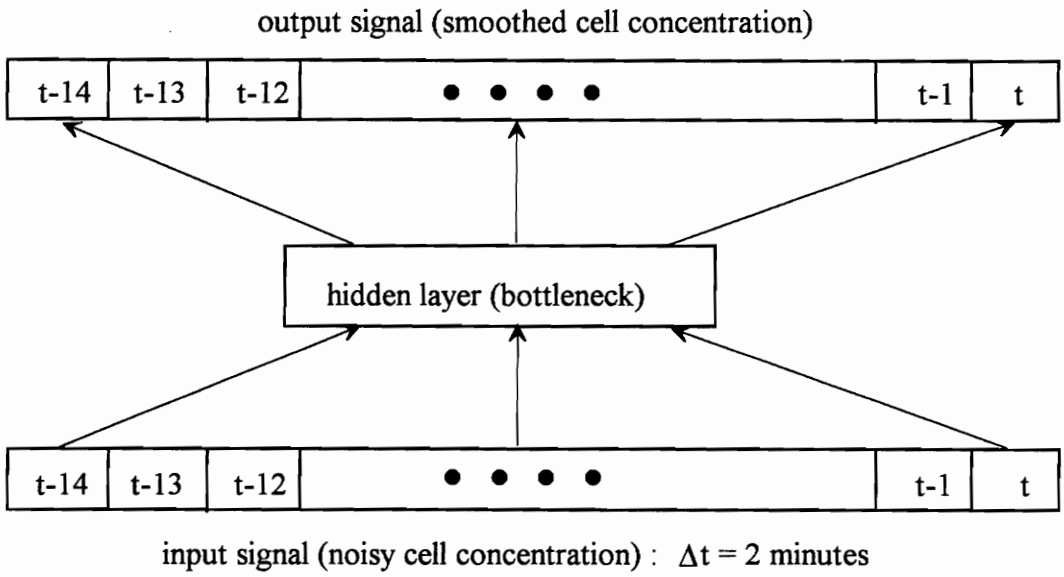


Figure 5.10. The architecture of the autoassociative network for the data compression of a noisy cell-concentration signal.

b. Network Training

We train the network using four simulated cell-growth curves (Figure 5.11) spanning the entire range of possible cell-growth profiles, having maximum normalized cell concentrations of 0.2, 0.4, 0.6, and 0.8, respectively. Therefore, this data-compression network can be used for all cell-growth curves that are normalized to between 0 and 1, and are generated by similar reaction times, approximately 24 to 48 hours. Note that we train this network with data that are already smooth. Smooth training data are not required, but they can reduce the number of growth curves necessary for effective training using noisy data signals. Moreover, one needs multiple signals in a given region of the

curve to average out prediction errors that occur because the noise is not normally distributed.

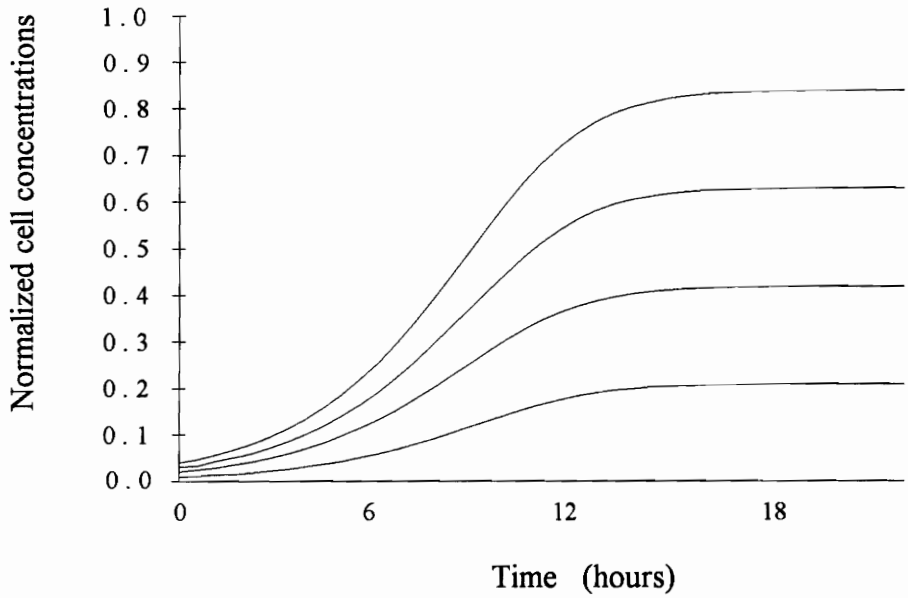


Figure 5.11. The four cell-growth curves used for training the autoassociative backpropagation network for data compression.

The training set for the data-compression network includes the four cell-growth curves, which consist of 675 data points for each curve, divided into 45 training examples consisting of 15 data points each, producing a total training set of 180 examples (file: *cellflt.nna*). The data are presented to the network in fifteen columns, one for each input node in the moving window. Table 5.6 shows fifteen columns of the data file *cellflt.nna.*

These fifteen columns form both the input and the output vectors for network training. Hence, if your software package does not have an autoassociative feature, enabling you to supply the data only once, you will need to add another fifteen columns to your data file and copy the fifteen columns from the file.

Table 5.6. The format of the file *cellflt.nna* used for training the data-compression network used for reducing noise in the cell-concentration signal.

Column number	Variable type	Normalized variable	Time period	Normalization factor
1	input/output	cell concentration	t - 14	
2	input/output	cell concentration	t - 13	
•	•	•	•	
•	•	•	•	* 100 g/l
•	•	•	•	
14	input/output	cell concentration	t - 1	
15	input/output	cell concentration	t	

Table 5.7 shows all the main characteristics used in training the data-compression network. We use a standard backpropagation network, with the input data presented autoassociatively and the output vector equal to the input vector, as described in Section 5.2. We select the hyperbolic tangent transfer function and the delta rule for this network training.

Table 5.7. The specifications for the data-compression network.

Network type	backpropagation
Training file name	*cellflt.nna*
Transfer function (input layer)	linear
Transfer function (hidden layers)	tanh
Transfer function (output layer)	tanh
Learning rule	delta rule
Summation	sum
Error	standard
Network weight distribution	normal distribution : 3 σ limits of [-1, 1]

input Layer

Training iteration	5,000
Noise	0
Learning rate	0.9
Momentum coefficient	0.6
Error tolerance	0

hidden layer

Training iteration	10,000	30,000	70,000
Noise	0	0	0
Learning rate	0.3	0.15	0.04
Momentum coefficient	0.4	0.2	0.05
Error tolerance	0.1	0.1	0.1

output layer

Training iteration	10,000	30,000	70,000
Noise	0	0	0
Learning rate	0.15	0.08	0.02
Momentum coefficient	0.4	0.2	0.05
Error tolerance	0.1	0.1	0.1

To determine the optimal compression ratio, we train many network configurations, varying the number of nodes in the hidden layer from one to twelve. Table 5.8 and Figure 5.12 show the effect of hidden layer size on network performance. Each network is trained with 50,000 iterations. As mentioned in Section 2.6.A, we compare the RMS error for network training and the corresponding compression ratio to determine the optimum configuration. As the number of nodes in the hidden layer increases, the RMS error decreases to 0.009 at a compression ratio of 3:1 where it remains constant. We find the optimal compression ratio to be between 2:5 and 4:1.

Table 5.8. A comparison of the average error and compression ratio for the data-compression network.

Number of nodes (hidden layer)	RMS error	Compression ratio
1	0.135	15 : 1
2	0.013	5 : 1
5	0.009	3 : 1
8	0.009	~ 2 : 1
10	0.009	3 : 2
12	0.009	5 : 4

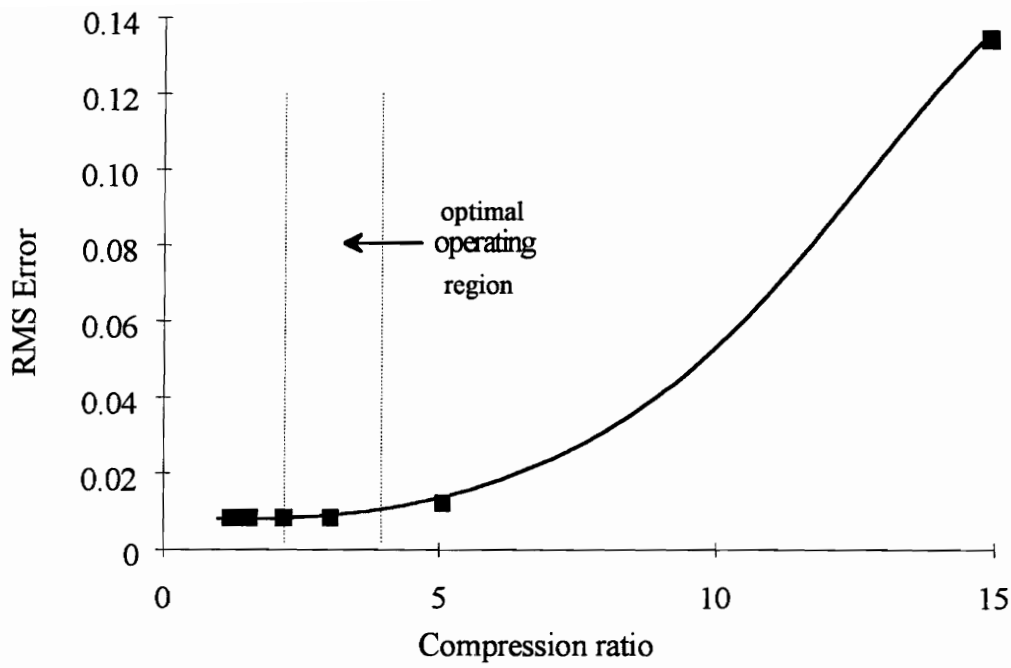


Figure 5.12. A plot of RMS error versus compression ratio used for the determination of the optimal compression ratio.

We select a 3:1 compression ratio (i.e., 15 nodes in both the input and output layers being compressed to 5 nodes in the hidden layer) for the network. As shown in Figure 5.13, the network is effectively trained to an acceptable error limit after 10,000 to 15,000 training iterations.

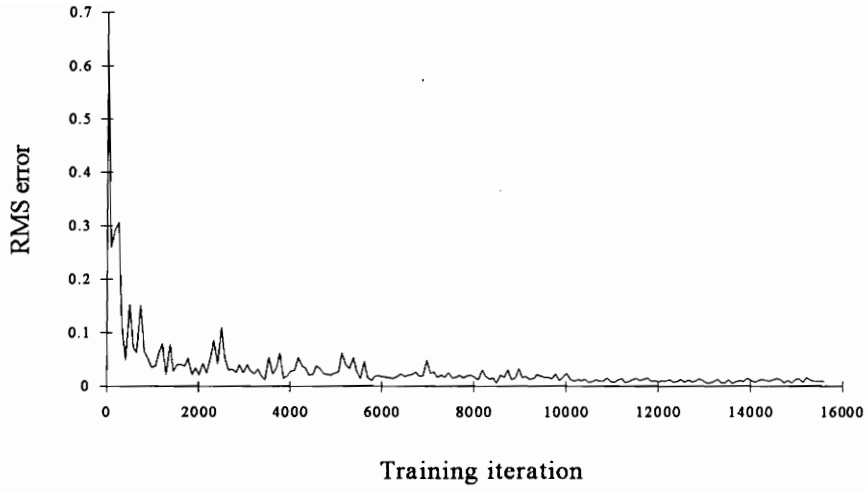


Figure 5.13. The training of the data-compression network for the reduction of noise in a cell-concentration signal.

c. Network Performance

Figure 5.14 shows the four cell-growth curves predicted from the data-compression network using the original curves from the network training. The two sets of growth curves (original and predicted) are essentially identical and hence appear as a single set on the diagram. The average error of the normalized cell concentration for the training example is 0.0027. Such a small error in prediction is expected for the original system, which only maps a smoothed curve onto itself, and it demonstrates that the compression ratio is not too large and that the network retains all relevant information about the given signal.

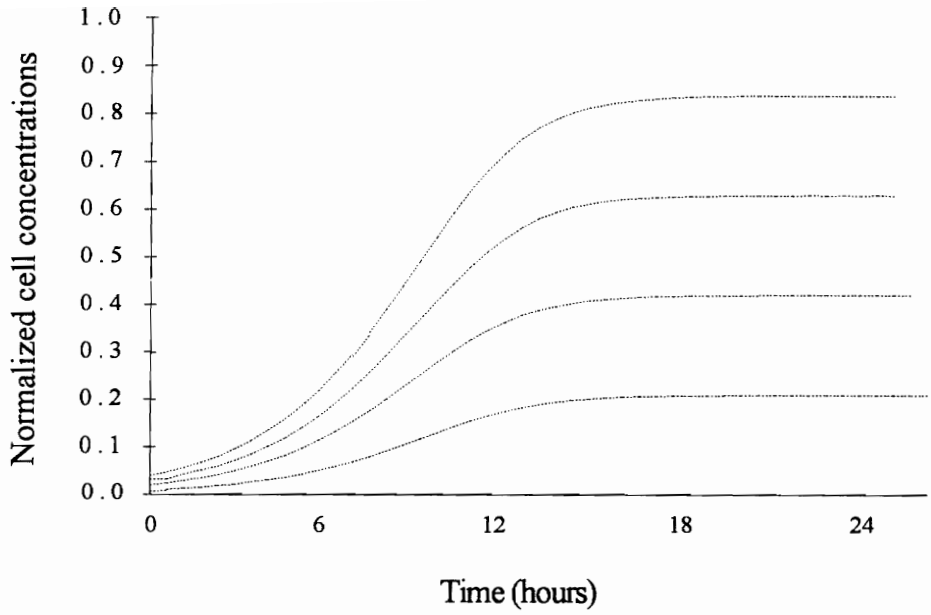


Figure 5.14. The prediction of cell concentration from training the autoassociative backpropagation network for the data compression of noisy signals.

To demonstrate using the network to reduce noise in the cell-growth signal, we now add random noise to the four signals used for training. We generate the noise using a normal distribution for the normalized cell concentration that has 3 σ limits of +/- 0.025 (Figure 5.15). This much noise is larger than we would expect from most measuring devices for cell concentrations, but will be instructive for demonstrating the performance of this type of network.

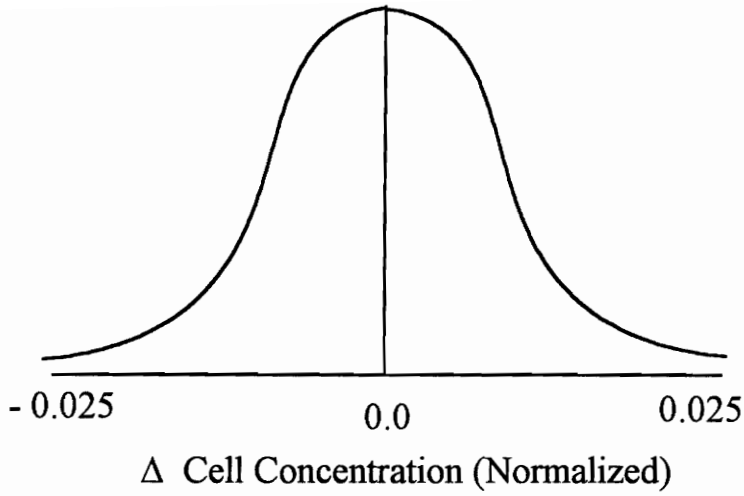
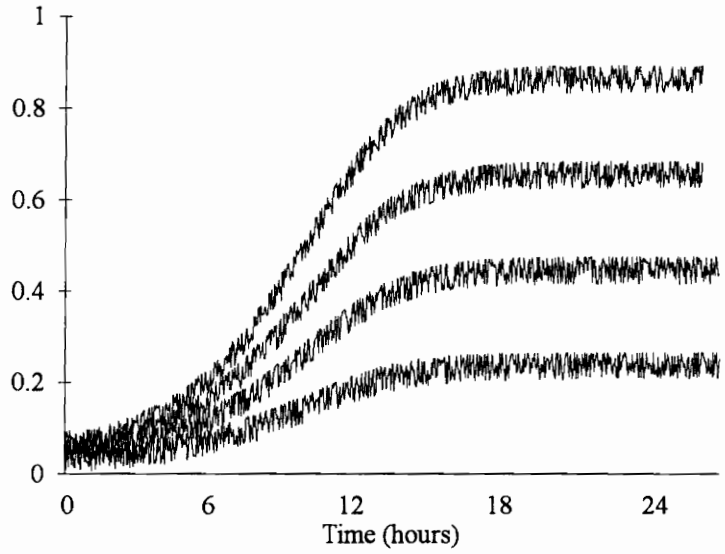


Figure 5.15. The random noise added to four cell-growth curves for testing the data-compression network.

Figure 5.16 shows the four cell-growth curves with random noise added, and the results of the data-compression network using those input signals (file: *cellnse.nna*). Signals with less noise result in much smoother signals than those shown in Figure 5.16.

(a)



(b)

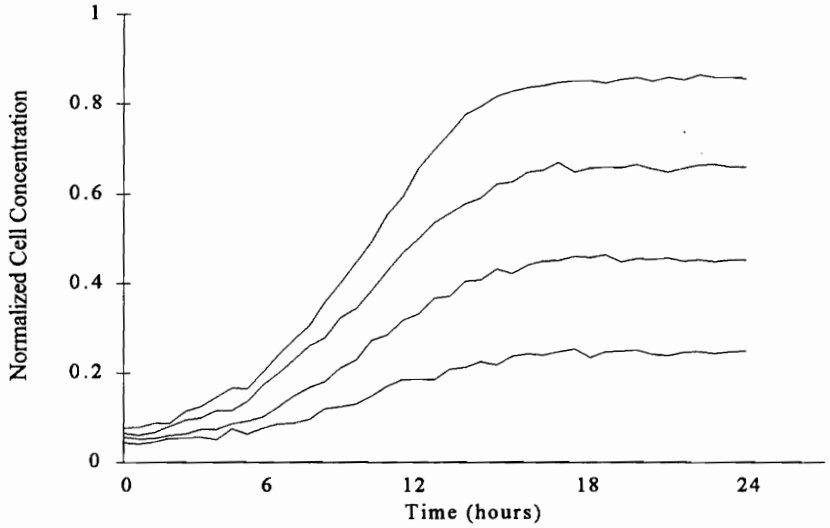


Figure 5.16. The four cell-growth curves with noise added at 3 σ limits of +/- 0.02 normally distributed and corresponding cell-growth curves after data compression using the data-compression network of Figure 5.10 with a 3:1 compression ratio.

3. Growth-Phase Classification Network

a. Network Design

To classify cell growth into one of the four phases, we use a radial-basis-function network (described in Chapter 3). Figure 5.17 shows the architecture of this network. We use the smoothed cell-concentration signal from the data-compression network as the primary input. In addition to those fifteen normalized cell concentrations, though, we also use the time t as an input variable because both the induction and stationary phases have similar slopes around zero time. As a result, the network has trouble distinguishing the different magnitudes of cell concentrations for growth curves in the initial time periods. Adding the time variable provides a means to characterize the induction phase in the early time periods and the stationary phase in the later periods. However, this variable may not be required if the fermentation process normally operates around a targeted cell-growth profile, as is the case with most industrial applications. Finally, although Figure 5.17 shows the death phase as an output parameter, we are not using it in our example. Because the death phase cannot be measured using our spectrophotometric testing methods, we simply do not continue to operate past the stationary phase.

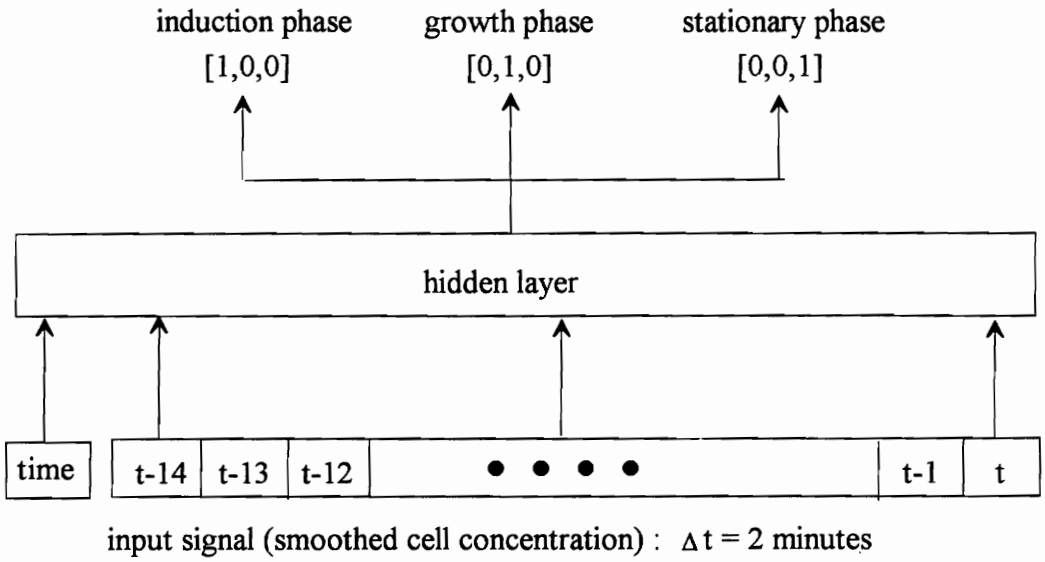


Figure 5.17. The architecture of the cell-growth-phase classification network.

Each output of the cell-growth classification network is fixed at either "0" if the fermentation process is not in that specific phase, or "1" if the process is in that phase. For any specified time period within a given moving window, only one phase can be identified as "1", and the other two will be set at "0". Therefore, the three possible output vectors used in the training of the network are :

(1) induction phase : [1 0 0]

(2) growth phase : [0 1 0]

(3) stationary phase : [0 0 1]

The radial-basis-function network classifies the phase based on which output is greater in testing/recall. There is a transition region when the process switches from one phase to another (e.g., stationary to growth phase), where both phases have a positive response that intersects around 0.5, as will be illustrated in Figure 5.20. A major objective in the design of this network is to minimize this transition region where the inferences are fuzzy, thus minimizing the total number of phases which are misclassified. Chapter 3 discusses techniques for performing this task in greater detail, and we will demonstrate those techniques in this section as we select the optimal network size based on misclassification percentages.

b. Network Training and Results

To train this network, we use a data set similar to the one used to train the data-compression network (Figure 5.18). The only difference is that here, we add five extra examples to the induction phase. The complete training data set file is given in "*cellphs.nna*". This data set also contains 200 training examples, or 50 for each cell-growth curve. Table 5.9 show the sixteen input columns and three output columns of the data file.

Table 5.10 summarizes the main specifications of the cell-growth-phase classification network. The design uses a radial-basis-function network with one hidden layer.

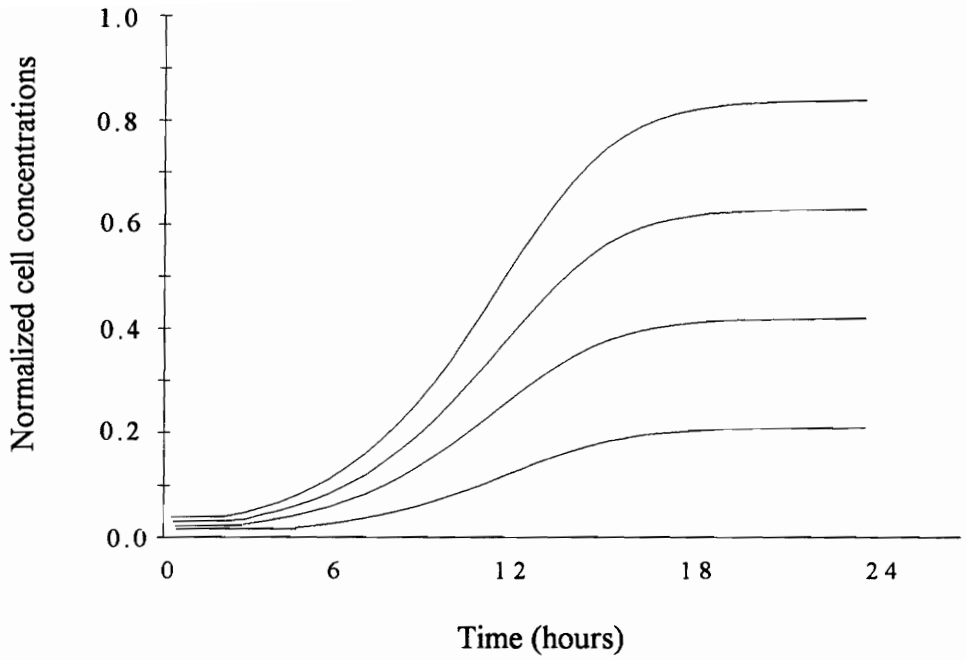


Figure 5.18. The four cell-growth curves used for training the autoassociative backpropagation network for data compression.

Table 5.9. The format of the file *cellphs.nna* used for training the cell-growth-phase classification network.

Column number	Variable type	Normalized variable	Time period	Normalization factor
1	input	time	t	100 hours
2	input	cell concentration	t - 14	
3	input	cell concentration	t - 13	
•	•	•	•	
•	•	•	•	100 g/l
•	•	•	•	
15	input	cell concentration	t - 1	
16	input	cell concentration	t	
17	output	induction phase	-----	class [1,0,0]
18	output	growth phase	-----	class [0,1,0]
19	output	stationary phase	-----	class [0,0,1]

Table 5.10. The specifications for the cell-growth-phase classification network.

Network type	radial-basis-function network
Training file name	*cellphs.nna*
Transfer function (input layer)	linear
Transfer function (hidden layer)	radial basis
Transfer function (output layer)	tanh
Learning rule (hidden layer)	K-means
Learning rule (output layer)	delta rule
Summation (hidden layer)	Euclidean
Summation (output layer)	sum
Error (hidden layer)	2-NN
Error (output layer)	standard
Network weight distribution	distribution range: [-1.0, 1.0]

input layer

Training iteration	5,000
Noise	0.0
Learning rate	0.9
Momentum coefficient	0.6
Error tolerance	0.0

hidden layer

Training iteration	500	1,000	1,500	2,000
Noise	0.0	0.0	0.0	0.0
Learning rate	0.3	0.15	0.075	0.0375
Cluster threshold	0.1	0.05	0.025	0.0

output layer

Training iteration	2,000	12,000	32,000	72,000
Noise	0.0	0.0	0.0	0.0
Learning rate	0.0	0.15	0.075	0.01875
Momentum coefficient	0.0	0.4	0.20	0.05
Error tolerance	0.0	0.1	0.1	0.1

To identify the optimal network architecture, we train four networks using 10, 20, 35, and 50 nodes in the hidden layer. Table 5.11 lists the number of training examples misclassified for each network. Each network is trained with 50,000 iterations. The percent misclassified decreases as the number of nodes in the hidden layer increases, reaching a minimum between 30 and 40 nodes. We will set the optimal number of hidden nodes at 35 to achieve an acceptable classification accuracy while maximizing training efficiency.

Table 5.11. A comparison of misclassification percentages for the radial-basis-function network having four different hidden-layer sizes.

Number of nodes in the hidden layer	Number of misclassified training examples	% Misclassified
10	16	8.00 %
20	2	1.00 %
35	1	0.50 %
50	0	0.00 %

Figure 5.19 shows the training of the optimal network configuration (35 nodes in the hidden layer) for the first 20,000 iterations. Minimal reduction in RMS error occurs after 5000 iterations.

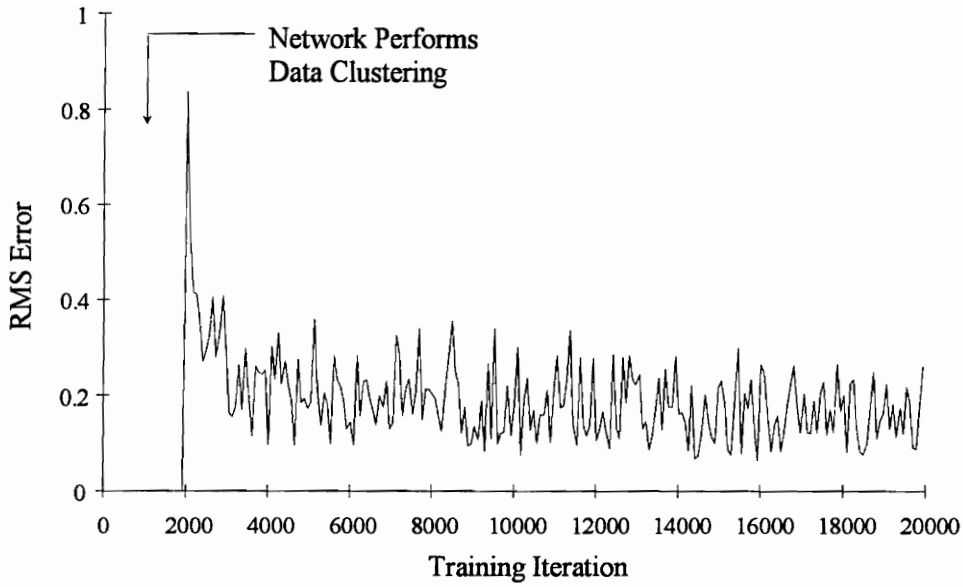


Figure 5.19. The training of cell-growth-phase classification network for 35 nodes in the hidden layer.

As seen, the RMS error is much greater than expected based on low misclassification percentages. This high error occurs because the crossover of responses in the transition region between two phases. In this region, one phase response decreases from 1 to 0, while the next increases from 0 to 1. Therefore, we can have proper classification from responses of 0.6 and 0.4, but still have large errors from 1 and 0, respectively. Figure 5.20 further illustrates this problem, displaying the three network responses for the third cell-growth curve, which reaches an upper limit of 0.6 normalized cell concentration. The dashed lines represent the transition regions between two phases as predicted by the network. As seen, visual classification is no more accurate than the

network predictions, and we can make only fuzzy inferences. Originally, we want to minimize these two transition regions. The transition from induction [1,0,0] to the growth [0,1,0] lasts one hour, while the transition from growth [0,1,0] to stationary [0,0,1] lasts two hours.

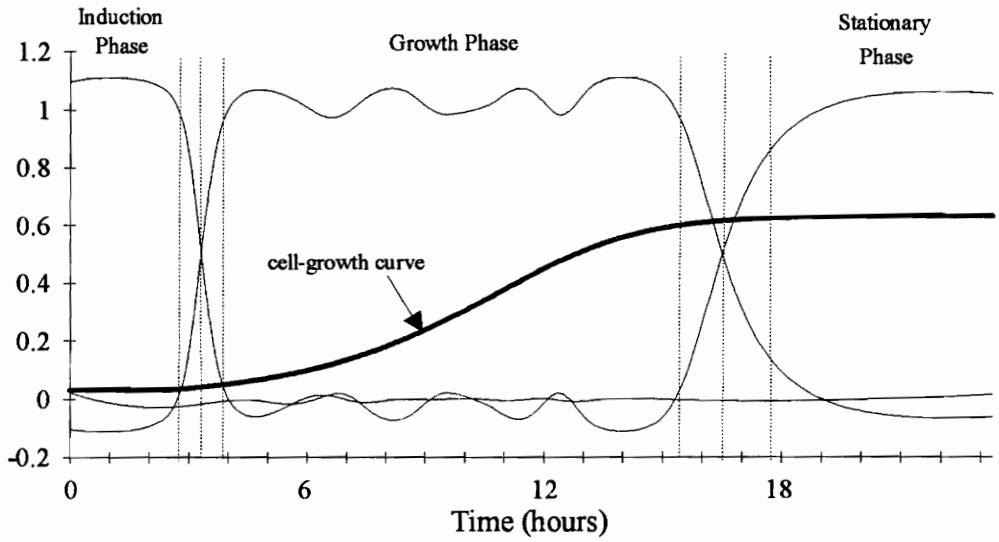


Figure 5.20. The cell-growth-phase classification responses for the third cell-growth curve.

4. Fermentation-Processing Network

a. Network Design

The output of the classification network then activates the *fermentation-processing network* when the process is in the growth phase. Figure 5.21 shows the architecture of

the *fermentation-processing network*, which is designed based on the recurrent network for process forecasting (Section 5.3) and the hierarchical structured moving-window network (Section 2.6.B.1). Every input variable uses a process-trend scanning window including three times (t-2Δt, t-Δt, and t).

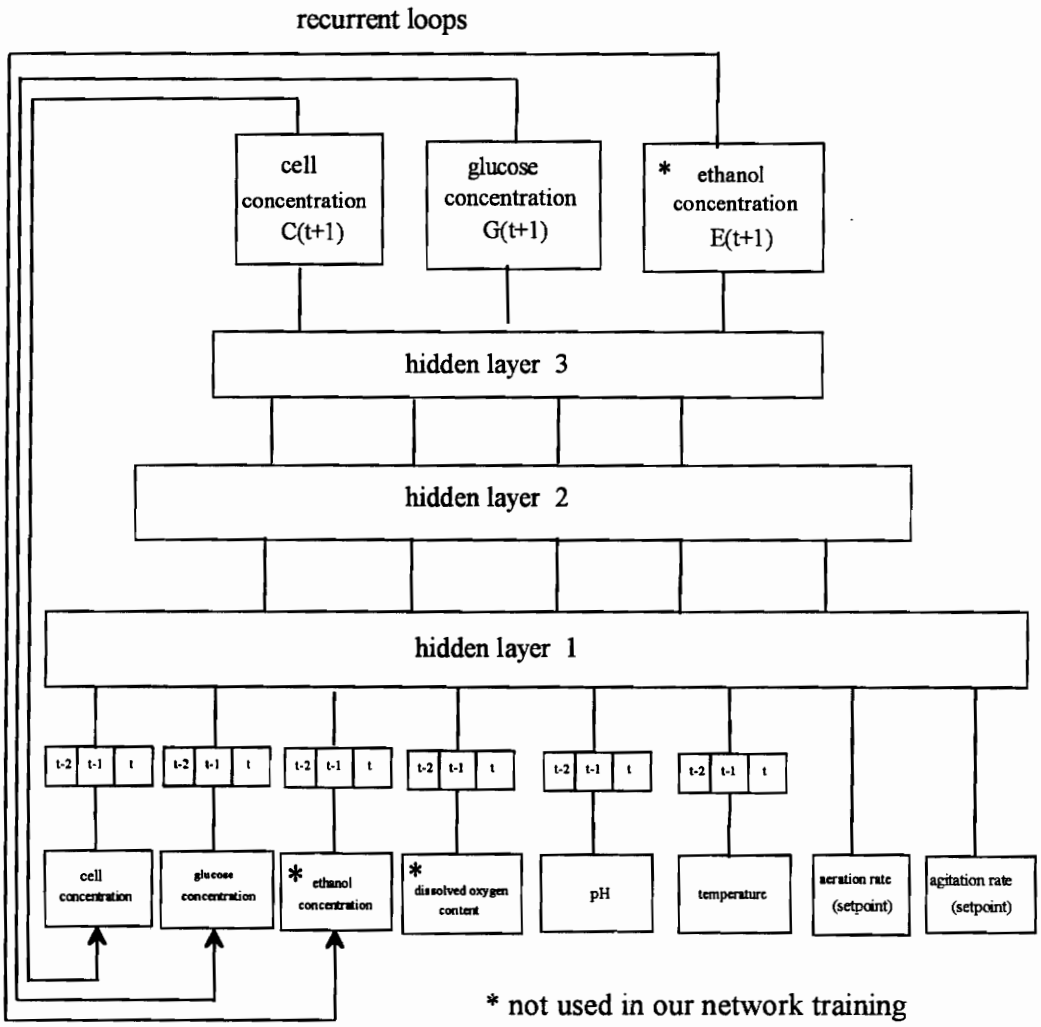


Figure 5.21. The fermentation-processing network.

The network has two groups of properties entering the input layer. The first group includes the manipulated variables, u_i, which are set based on the experimental design strategy discussed in Section 5.4.C.2. This group is further divided into optimal control and setpoint control variables. The optimal control variables include temperature, pH value, and dissolved oxygen content, with three input values (at $t-2\Delta t$, $t-\Delta t$, and t) for each variable. The setpoint control variables are aeration rate and agitation rate, and each has one constant input value for the entire process. The second group are measured variables which are both input to the network, $y_j(t-2\Delta t, t-\Delta t, t)$, and output from the network, $y_j(t+\Delta t)$. The measured variables include cell, glucose, and ethanol concentrations. When the network is in testing/recall mode, the predicted values at time t+1 are recirculated into the input layer by recurrent loops to predict the next response.

The dissolved oxygen content and ethanol concentration variables, which are included in the standard network architecture (Figure 5.21), are not used in training our particular fermentation-processing network. In our experimentation, the dissolved oxygen content always starts near 100% of its maximum level, causing a high degree of nonlinearity and variability between runs in the early stages of the process prior to steady state. Therefore, including the dissolved oxygen content in our model is detrimental to network training. However, although we do not use dissolve oxygen content directly, its steady-state level appears indirectly as a function of the aeration and agitation rates. We exclude ethanol concentration from the model because we have not measured it in our experiments. Moreover, ethanol concentration is not required for predicting cell and

glucose concentrations, because it is indirectly included in the network structure through the manipulated and measured variables.

Note that our model is not limited to the variables specified in Figure 5.21. All the standard measured and controlled variables shown in Table 5.1 (e.g., pressure, feed rate, etc.) can easily be added to our network model. In addition, including an expanded set of products and critical reactants used in more complex bioprocesses should not cause any difficulties either.

b. Network Training and Results

Table 5.12 shows the primary specifications used in training the fermentation-processing network. We use a standard backpropagation network having three hidden layers, with the output vector recirculated to input layer via recurrent loops, as described in Section 5.3. The hyperbolic tangent transfer function and the delta rule are effective for this network. The data for network training are supplied in the file: *ferm.nna* (format given in Table 5.3).

Table 5.12. The specifications of the fermentation-processing network.

Network type	backpropagation
Training file name	*ferm.nna*
Transfer function (input layer)	linear
Transfer function (hidden layers)	tanh
Transfer function (output layer)	tanh
Learning rule	delta rule
Summation	sum
Error	standard
Network weight distribution	normal distribution : 3 σ limits of [-1, 1]

input Layer			
Training iteration	5,000		
Noise	0		
Learning rate	0.9		
Momentum coefficient	0.6		
Error tolerance	0		

hidden layer 1			
Training iteration	10,000	30,000	70,000
Noise	0	0	0
Learning rate	0.3	0.15	0.04
Momentum coefficient	0.4	0.2	0.05
Error tolerance	0.1	0.1	0.1

hidden layer 2			
Training iteration	10,000	30,000	70,000
Noise	0	0	0
Learning rate	0.25	0.13	0.03
Momentum coefficient	0.4	0.2	0.05
Error tolerance	0.1	0.1	0.1

hidden layer 3			
Training iteration	10,000	30,000	70,000
Noise	0	0	0
Learning rate	0.2	0.1	0.03
Momentum coefficient	0.4	0.2	0.05
Error tolerance	0.1	0.1	0.1

output layer			
Training iteration	10,000	30,000	70,000
Noise	0	0	0
Learning rate	0.15	0.08	0.02
Momentum coefficient	0.4	0.2	0.05
Error tolerance	0.1	0.1	0.1

We determine the optimal configuration for the hidden layers of the fermentation-processing network by varying the number of hidden layers from 1 to 3, and varying the total number of nodes within the layers from 10 to 45. We maintain the two-layer networks at a 2:1 ratio of nodes in the first layer to nodes in the second layer. The three-layer networks have ratios close to 3:2:1, for layer numbers 1, 2, and 3, respectively. Table 5.13 shows how the number of layers and their sizes affect network performance. We train each network for 50,000 iterations.

Table 5.13. A comparison of the effectiveness of different hidden-layer configurations for training the fermentation-processing network.

Hidden layer 1 number of nodes	Hidden layer 2 number of nodes	Hidden layer 3 number of nodes	Total number of nodes	RMS error
10	0	0	10	0.037
20	0	0	20	0.032
30	0	0	30	0.037
40	0	0	40	0.038
8	4	0	12	0.021
14	7	0	21	0.021
20	10	0	30	0.021
30	15	0	45	0.017
10	7	3	20	0.020
15	10	5	30	0.019
20	12	7	39	0.018

Figure 5.22 shows how the total number of nodes in the hidden layer affects the RMS error for recall of training data. Prediction capability increases significantly going from 1 to 2 hidden layers, but adding a third hidden layer produces minimal further variation. Hence, we will use two hidden layers in our configuration.

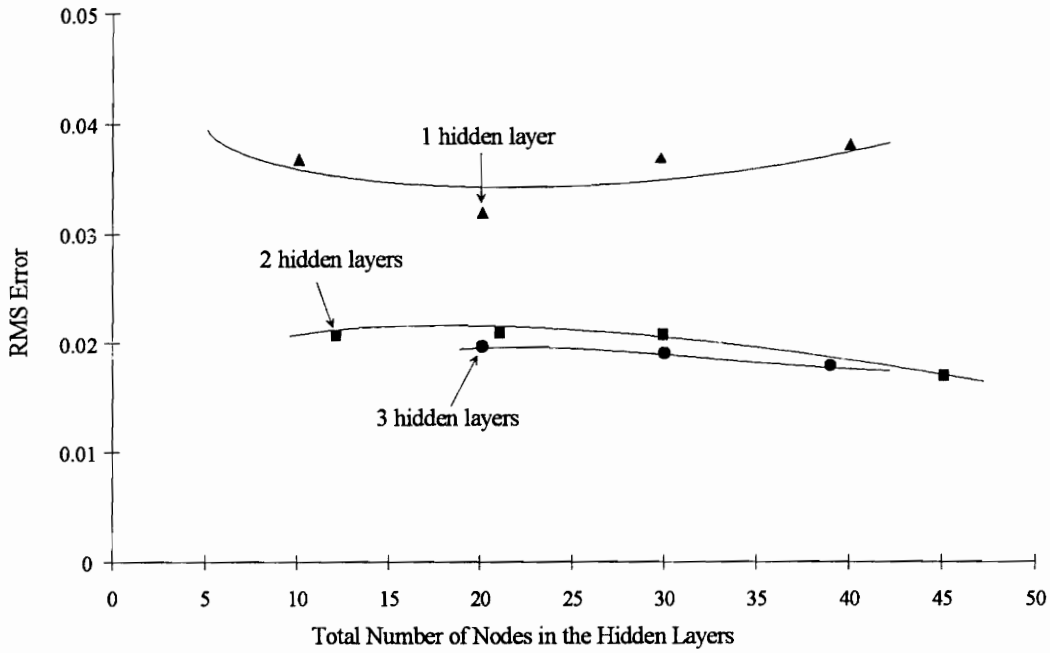


Figure 5.22. A comparison of the RMS error for the recall of training data for a varying number of total nodes in 1, 2 and 3 hidden-layer fermentation-processing networks.

Adding nodes within a layer, e.g. going from 10 to 40 total nodes in any of the three network structures, also produces little improvement. Based only on training-data

recall, then, we select a network with 30 nodes in the first hidden layer and 15 nodes in the second as the most effective configuration. Always select the network that has the lowest number of layers and yet effectively models the system to avoid generalization problems in the future. We will later demonstrate how different network structures affect generalization with new training examples.

Figure 5.23 shows the 50,000 iterations in training the fermentation-processing network. The network is effectively trained in the first 10,000 to 20,000 iterations.

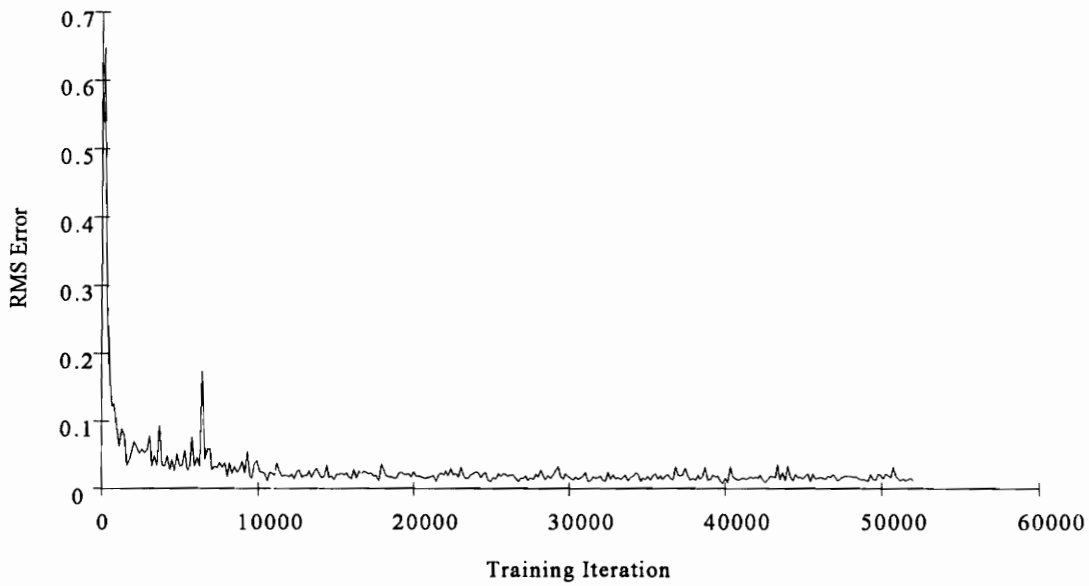


Figure 5.23. The training of the fermentation-processing network for predicting the cell concentration using a two-layer network configured with 30 and 15 nodes for hidden layers 1 and 2, respectively.

Figure 5.24 shows the scatter plot of predicted versus actual values from recall of the training data file *ferm.nna* using the fermentation-processing network having 30 nodes in hidden layer 1 and 15 nodes in hidden layer 2.

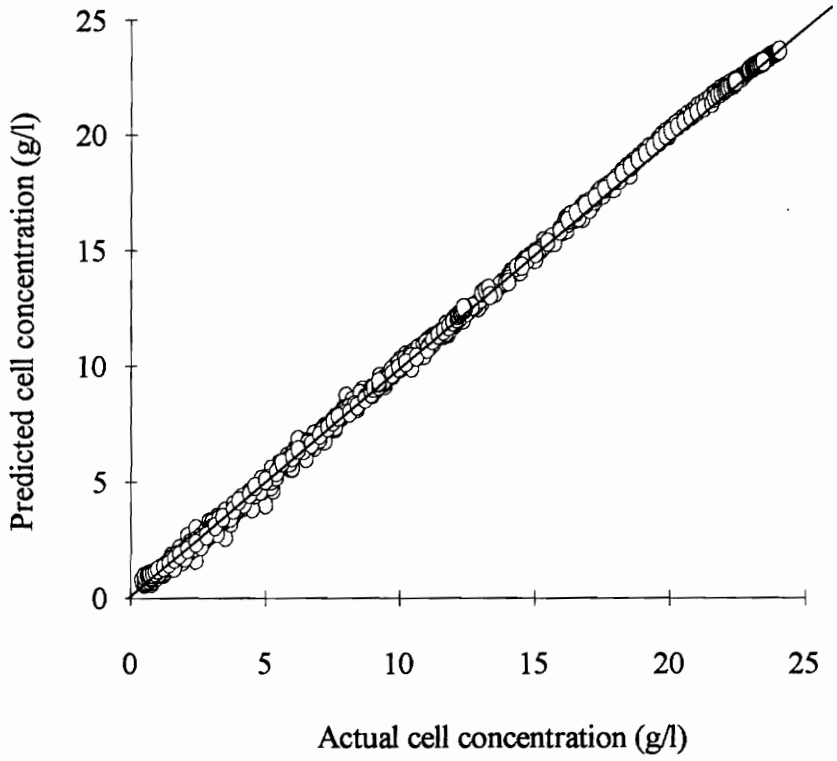


Figure 5.24. Scatter plot for recall of data file *ferm.nna* using the fermentation-processing network for predicting cell concentration using a two-layer network configured with 30 and 15 nodes for hidden layers 1 and 2, respectively.

To test the network for its ability to generalize new training examples, we generate a learning curve as described in Section 2.1.C.3 and 2.3.F. We divide the fifteen

experimental runs used for training into five groups of three experiments each (see Table 5.5), and train the network with one group initially and test for generalization of the other four. We then move one group from the generalization test groups to the training groups. We continue this process until no generalization data groups remain. We determine the overall effectiveness of the network in predicting a cell-growth profile based on how closely the average error for generalization data set approaches the average error for the training data set. Note that we set the iteration counter back to 10,000 at the end of each training run, so the next training run has the desired set of the network specifications (e.g., learning rate of hidden layer 1 = 0.3 from Table 5.12).

As Table 5.5 shows, the first group of data for initial training includes the lower limit of cell-growth profile (experiment 3), an intermediate cell-growth profile (experiment 6), and the upper limit cell-growth profile (experiment 14). The first group of data should include examples that span a broad set of operating conditions and the entire distribution of output responses. This eliminates many of the problems associated with extrapolated predictions and new training examples outside the previously modeled operating limits. Sometimes, including new data sets that do not follow past pattern behavior requires retraining the entire network from the beginning. This retraining results from the many barriers in the training algorithm that prevent the internal model representation from being drastically modified without large learning rates and momentum coefficients that destroy all past network learning patterns.

Once the initial training set is established, the remaining four data sets are randomly put into the groups. Table 5.14 shows the average errors obtained for recall and generalization in this process.

Table 5.14. The average errors for the recall and generalization of the fermentation-processing network (30:15 hidden-layer configuration) used in the generation of the learning curve.

Training	Additional	Recall	Generalization : error (g/l)				
group # 's	run numbers	avg error (g/l)	group 2	group 3	group 4	group 5	Average
1	3, 14, 6	0.125	3.54	3.50	3.42	2.74	3.30
1, 2	11, 5, 12	0.168	----	2.54	1.37	2.42	2.11
1, 2, 3	8, 4, 3	0.180	----	----	1.14	0.53	0.84
1, 2, 3, 4	1, 2, 9	0.186	----	----	----	0.37	0.37
1, 2, 3, 4, 5	15, 7, 10	0.165	----	----	----	----	----

Figure 5.25 shows the learning curves generated from the average errors presented in Table 5.14. The error associated with training-data recall remains at a reasonable level throughout all stages of the learning process, increasing only slightly but insignificantly with additional training examples. In comparison, the generalization curve has large prediction errors when only three cell-growth profiles are used for training. The average error decreases rapidly with each additional group of experimental runs. In the later stages of the learning curve, the average error for the generalization data sets approaches that for the recall data set, indicating that training is near completion. The network may

now require only a few new experimental runs, to produce accurate predictions within the operating conditions set in the experimental design strategy (Table 5.2).

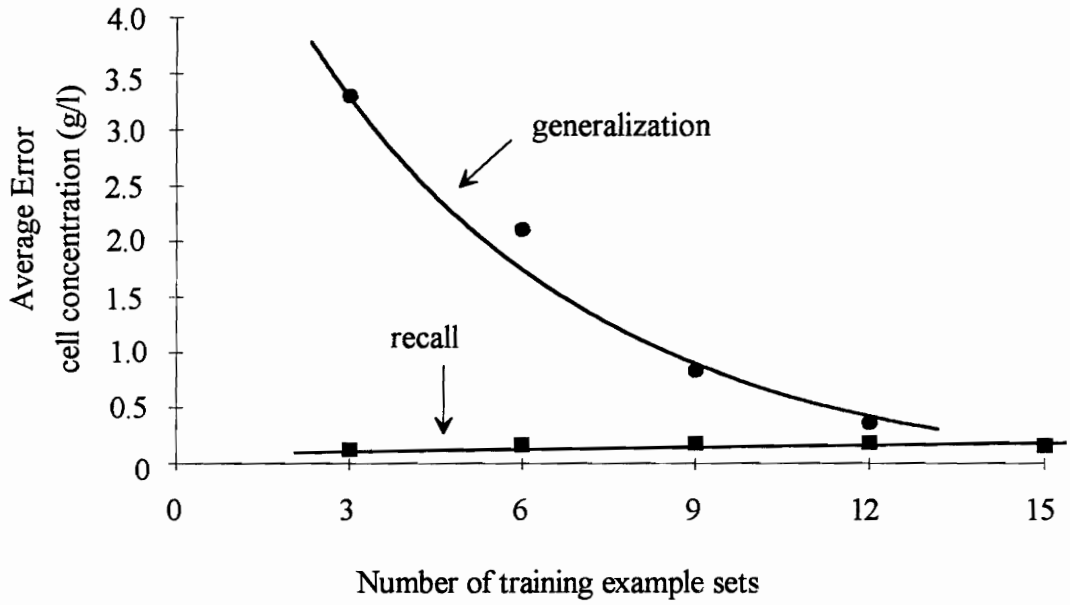


Figure 5.25. The learning curve for training of the fermentation-processing network (30:15 hidden-layer configuration).

We also compare the learning curves for all the network configurations used in our original recall-capability testing (Table 5.13). Table 5.15 shows the average error for the generalization of the fermentation-processing network trained with 3, 6, 9, and 12 training example sets. The data groups are added into the training set in the same order used for

the 30:15 network configuration just described (Table 5.14). Comparing network configurations for generalization, we find that using a two-layer network with a total of 45 nodes outperforms networks with a fewer number of nodes.

Table 5.15. The average errors for the generalization of the fermentation-processing network, trained with 3, 6, 9, and 12 training example sets used for the generation of learning curves for multiple network configurations.

Hidden-layer configuration			3 training runs	6 training runs	9 training runs	12 training runs
layer 1	layer 2	layer 3				
10	-----	-----	2.66	1.92	0.58	0.56
20	-----	-----	2.29	1.29	0.67	0.54
30	-----	-----	2.45	1.84	0.52	0.56
40	-----	-----	2.34	1.67	0.61	0.60
8	4	-----	3.06	2.27	1.12	0.77
14	7	-----	2.12	2.46	0.65	0.60
20	10	-----	2.84	1.75	0.79	0.59
30	15	-----	1.92	1.52	0.53	0.34
10	7	3	2.05	1.16	0.61	0.67
15	10	5	2.64	2.17	0.62	0.58
20	12	7	1.97	1.08	0.56	0.38

c. Illustrative Predictions

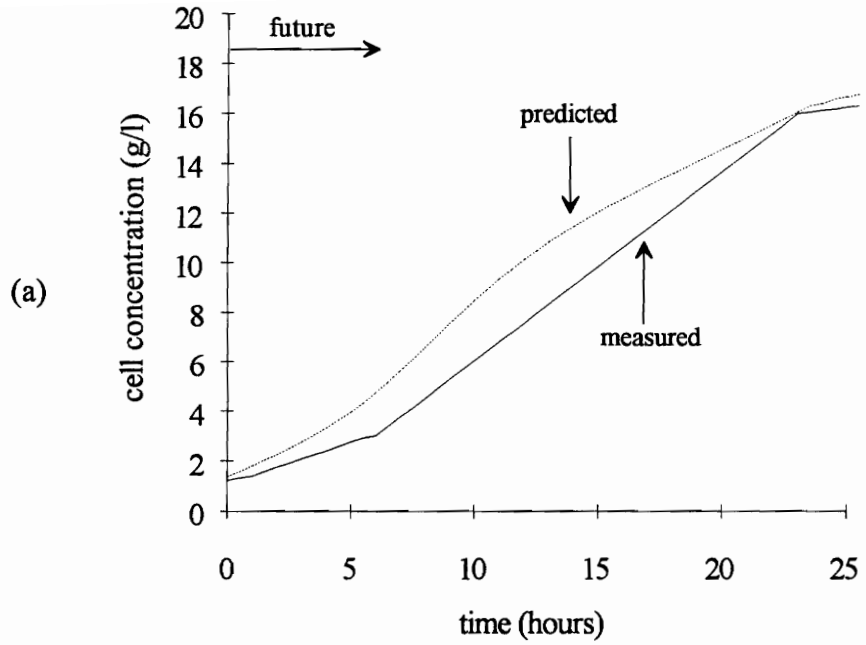
We will now illustrate the fermentation-processing network's performance using two examples. First, we examine network performance when recalling an experimental run

from the training data set. Second, we show a limitation of neural networks using an example where the processing conditions are far beyond those used in network training.

c1. Recalling an Experimental Run from the Training Data

We test the performance of experiment 5, which has a cell-growth profile close to the average profile of the fifteen experiments. Figures 5.26a-d show the network forecasting ability beginning from several times in the process (0, 5, 10, and 15 hours) and proceeding to the completion of the process at 25 hours. All four cases predict the final cell concentration at 25 hours within 0.5 g/l from those measured values, although there is some variations at times throughout the process, especially for forecasts that begin at early times (e.g., 0 and 5 hours). This is probably due to variations in cell preparation, causing different patterns of cell growth in the early stage of the growth phase from one experiment to another (e.g., a different percentage of cells being in an active state when the process begins). The network's prediction capability improves significantly as we proceed to later stages of the process where cell growth follows much more consistent patterns from batch to batch.

This example is characteristic of training examples that have average cell-growth curves. Training examples near the lower and upper bounds of the cell-growth curves will show slightly greater errors.



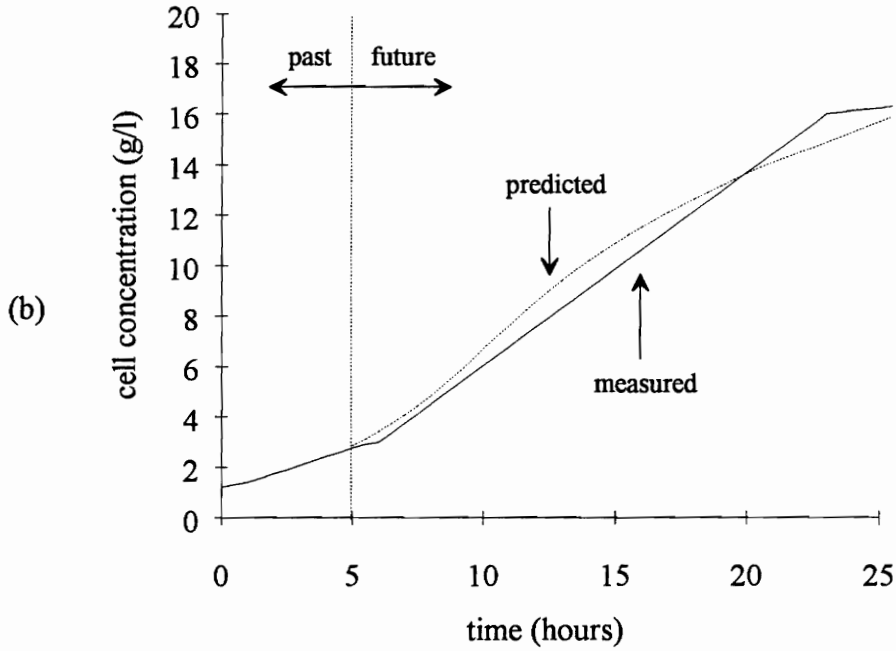
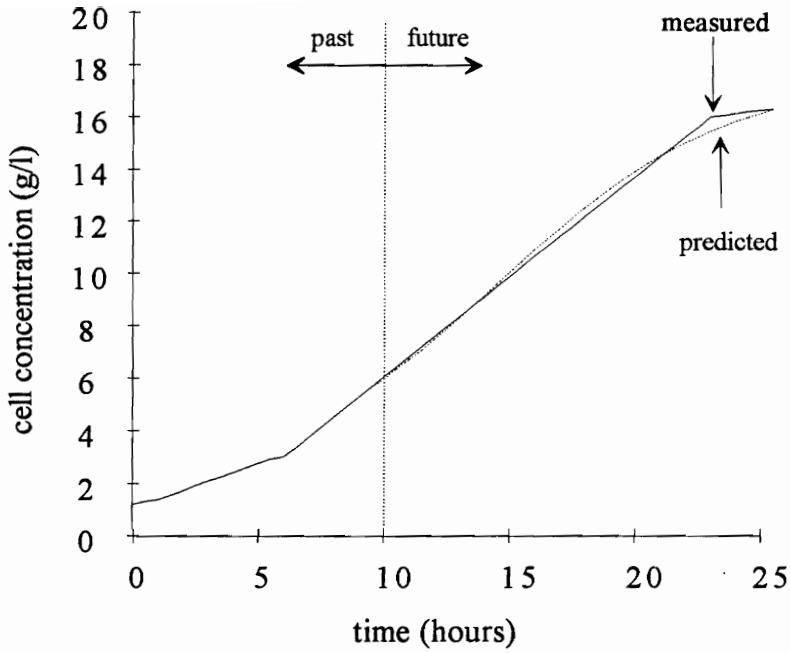


Figure 5.26a-d. The process forecasting of future cell concentrations by the fermentation-processing network for experiment 5 beginning at 0, 5, 10,and 15 hours.

(c)



(d)

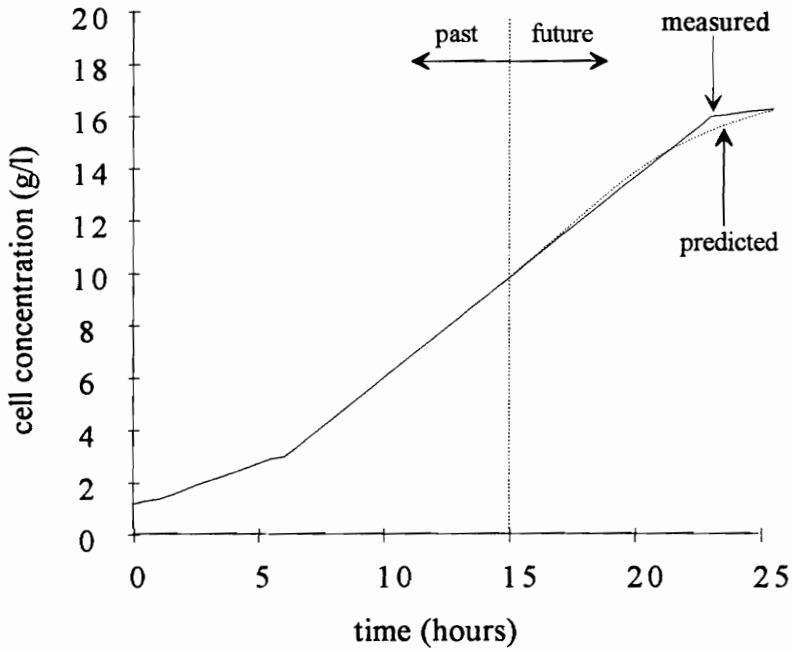


Figure 5.26a-d. The process forecasting of future cell concentrations by the fermentation-processing network for experiment 5 beginning at 0, 5, 10, and 15 hours.

We must emphasize that our broad-based experimental design strategy (Section 5.4.C.2) is chosen to model the entire feasible range of processing conditions. In order to give an accurate model for normal operating conditions and a reliable model for all outlying operating conditions, we should add a second set of training examples near the normal operating conditions of the process (refer to Figure 5.7).

c2. Forecasting the Cell Concentration with Extrapolation from Training Data

In this second example, we demonstrate one of the most common problems encountered using process-forecasting networks, that is, using a limited data set in network training and then trying to predict responses for an input vector that is not within the input boundaries of the training examples. This example illustrates why a multiple-tiered experimental design strategy is beneficial for process forecasting, by reducing troublesome situations in which the network must extrapolate outside of training regions. Specifically, we demonstrate how the fermentation-processing network performs for experimental runs that require extrapolation. This is accomplished by *eliminating* certain experimental runs, whose input variables are set at extreme operating conditions, from the training examples and testing the network under those conditions.

For this case, we first train the network with 12 of the 15 experimental runs (groups 1 to 4 of Table 5.5). Next, we use experiment 15, from the eliminated training group (group 5), to illustrate the effect of extrapolation from input training data. We have

selected this experiment because it has the largest deviation from the input training patterns and has a cell-growth profile at the upper bound of the output responses. Specifically, experiment 15 corresponding to low initial glucose concentration (50 g/l), high initial cell concentration (> 0.46 g/l), high temperature (35°C), and high aeration rate (5.0 Nm^3/hr), with pH (4.5) and agitation rate (400 rpm) at their nominal values. Therefore, the network must extrapolate all output responses from the corresponding input pattern.

For this example, we arbitrarily set the initial forecasting to begin at 12 hours and proceed to the completion of the cell-growth phase at 37 hours. Figure 5.27 shows the forecasting of future cell concentrations for experiment 15, in conjunction with predictions utilizing 5-point-average and 10-point-average error-correction procedures (see Section 5.3.C). The two error-correction procedures can improve the prediction capability for operating conditions unfamiliar to the network. These procedures are important when the network is not trained with a sufficiently large database or when the data are collected without a good experimental design strategy.

With no error correction, the fermentation-processing network predicts minimal increase in cell growth for the given set of conditions and glucose concentrations. The network underestimates the final cell concentration by 6.6 grams/liter, with final cell concentrations at 22.3 and 15.7 grams/liter for the measured and predicted values, respectively. This large error is expected based on similar differences in the predicted and measured cell concentrations just prior to predictions starting at 12 hours.

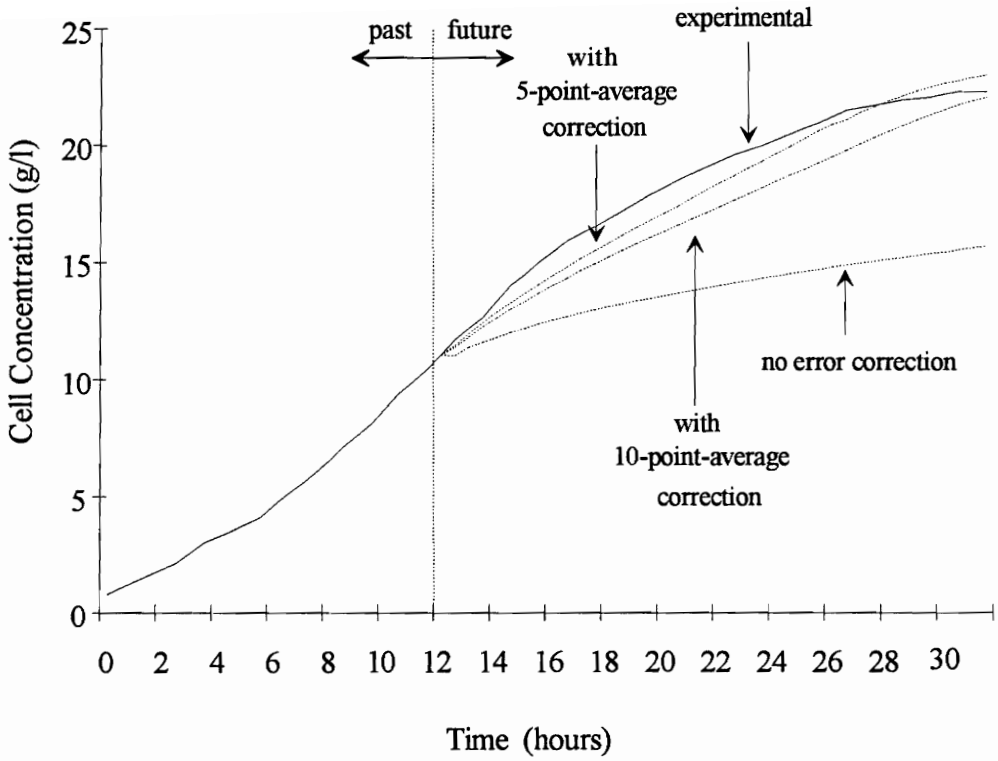


Figure 5.27. The process forecasting of future cell concentrations from the fermentation-processing network, trained with only the first twelve experimental runs, for experiment 15 with 5-point-average and 10-point-average error corrections, and with no error correction.

The 5-point-average and the 10-point-average error corrections, e_{avg}, are -0.45 and -0.35 grams/liter, respectively. Including the error term significantly improves the network's prediction capability in this example. The adjusted predictions still contain errors because the prediction error is not constant throughout the process (refer to Figure 5.28), as the difference between the 5-point-average and 10-point-average errors indicates. Better results require a nonlinear error-correction procedure.

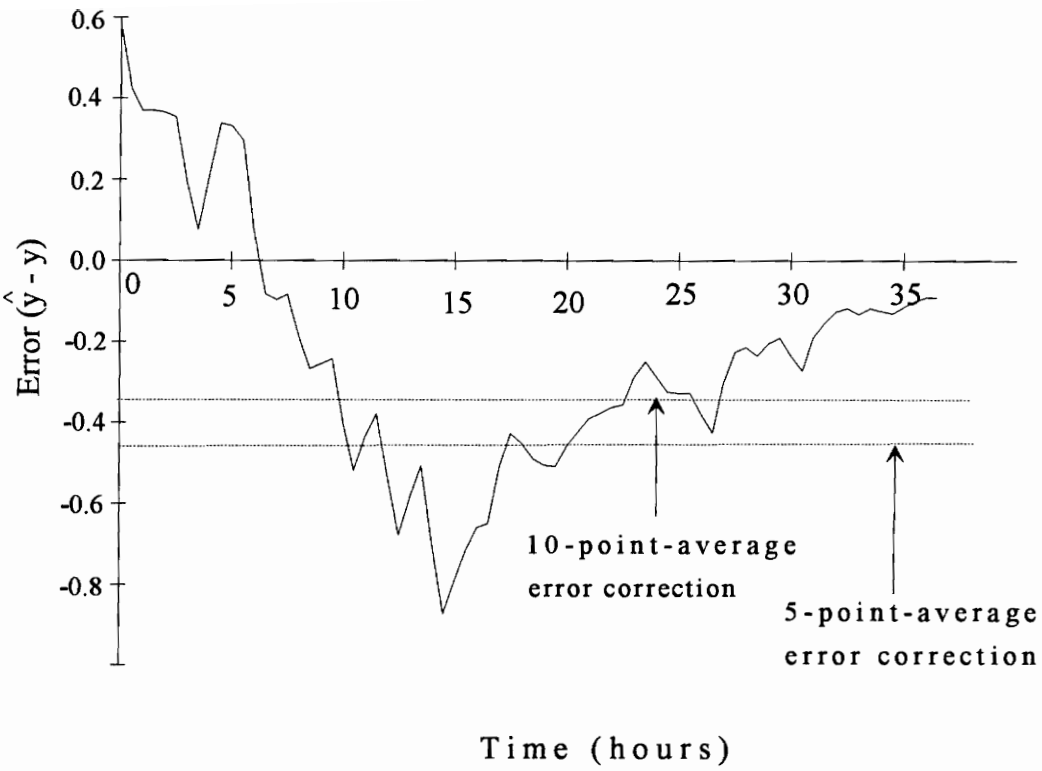


Figure 5.28. The prediction error, $\hat{y} - y$, for experiment 15 with the network trained using only experimental groups 1 to 4 of Table 5.5.

E. Conclusions

- Three neural networks in series (Figure 5.9) including the data-compression network (Figure 5.10), the cell-growth-phase classification network (Figure 5.17), and the fermentation-processing network (Figure 5.21) are effective in forecasting future cell concentrations in a batch fermentation process.

- The data-compression network (Figure 5.10) is capable of smoothing a noisy signal retrieved from an on-line cell counter. The data-compression attributes of the neural network reduce the amount of random noise in the input signal and also filter the gross-error in the data resulting from sensor malfunctions, drift, and bias.
- The cell-growth-phase classification network (Figure 5.17) successfully categorizes the four phases of the fermentation process (induction, growth, stationary and death phases). Therefore, it can activate the fermentation-processing network when the fermentation is in the growth phase.
- The fermentation-processing network (Figure 5.21) performs well in forecasting future cell concentrations over a wide range of operating conditions, including temperature (25 to 35 ° C), pH (3.5 to 5.5), agitation rate (200 to 600 rpm), aeration rate (0 to 5.0 Nm^3/hour), and glucose concentration (0 to 100 grams/liter).
- A second set of training examples should be added near the desired operating conditions of the process since the experimental runs used for network training are designed to model the entire feasible operating range.

5.5 Illustrative Case Study : Tennessee Eastman Plantwide Control Problem

In this section, we first give an introduction to the Tennessee Eastman plantwide control problem that has been widely studied in analyzing different types of control strategies for

complex chemical processes (Downs and Vogel, 1993). We define preliminary controller settings, used by TUTSIM process control simulation software (TUTSIM Products, 1994), to generate a training database for neural network to model the process. We then describe how to use a neural network as a supervisory controller replacing multiple cascade controllers. Finally, we develop a neural network model of the reactor, a key unit operation of the Tennessee Eastman problem.

A. Introduction

The Tennessee Eastman plantwide control problem was developed to study various process control strategies for complex chemical processes (Downs and Vogel, 1993). We use this problem to demonstrate how to use neural networks for process forecasting, adaptive process control, and optimization of a complex chemical process that consists of multiple unit operations.

This process consists of five major unit operations, including a reactor, a product condenser, a vapor-liquid separator, a recycle compressor, and a product stripper (Figure 5.29). This process has four reactants, A, C, D, and E, which produce two products, G and H, through irreversible exothermic reactions. In addition, there is an inert component B and a byproduct F that are also present in the process.

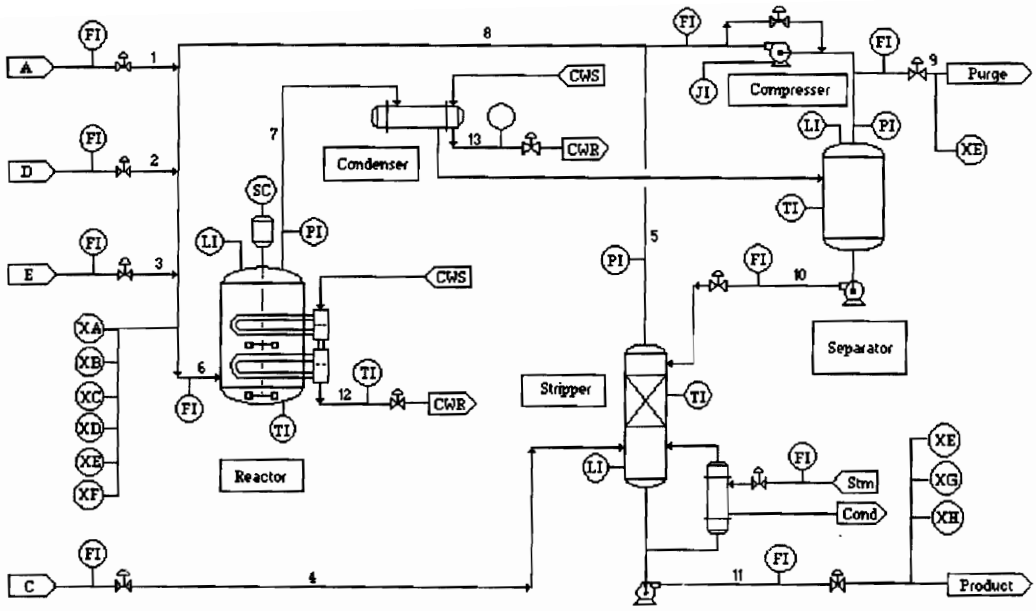


Figure 5.29. Schematic diagram of the Tennessee Eastman plantwide control problem.

The reactions are given as :

$A(g) + C(g) + D(g) \rightarrow G(liq)$	Product 1
$A(g) + C(g) + E(g) \rightarrow H(liq)$	Product 2
$A(g) + E(g) \rightarrow F(liq)$	Byproduct
$3D(g) \rightarrow 2F(liq)$	Byproduct

Essentially pure components A, D, and E (streams 1 to 3) are fed directly into the recycle stream (stream 8) which enters the reactor. Component C (stream 4) is fed into the bottom of the stripper in a nearly 50/50 (mass ratio) mixture with component A. Products F, G and H are formed as liquids, but the latent heats from the exothermic reaction cause them to vaporize. The reactor is supplied with a water-cooling system and an adjustable agitator. Vaporized products leave the reactor with the unreacted gas components and enter a shell-and-tube condenser. Next, the two-phase overhead stream from the reactor (stream 7) enters a vapor/liquid separator that separates the product-rich liquid from the unreacted gas components. The vapor phase (stream 8) enters a compressor and is recycled back into the reactor feed stream. The condensed liquid phase (stream 10), goes through a product stripping column in conjunction with C-feed (stream 4), to remove unreacted components from the product stream. The two products, G and H, exit the stripper (stream 11) where they are further refined downstream. The base-case processing mode gives a product stream with a 50/50 G/H mass ratio, i.e., 7038 kg/hr G and 7037 kg/hr.

There are twelve manipulated variables, XMV(1) to XMV(12), within the process which can be adjusted for controlling the process. Table 5.16 shows the twelve manipulated variables that affect 41 process measured variables, XMEAS(1) to XMEAS(41). Table 5.17 lists 22 variables that are measured continuously and Table 5.18 gives 19 variables that are measured at the given sampling times. Table 5.19 presents the

20 process disturbances, IDV(1) to IDV(20), that can be introduced during process operations.

Table 5.16. Process manipulated variables.

Variable name	Variable number	Base-case value (%)	Low limit	High limit	Units
D feed flow (stream 2)	XMV(1)	63.053	0	5811	kg/hr
E feed flow (stream 3)	XMV(2)	53.980	0	8354	kg/hr
A feed flow (stream 1)	XMV(3)	24.644	0	1.017	kscmh
A and C feed flow (stream 4)	XMV(4)	61.302	0	15.25	kscmh
Compressor recycle valve	XMV(5)	22.210	0	100	%
Purge valve (stream 9)	XMV(6)	40.064	0	100	%
Separator liquid flow (stream 10)	XMV(7)	38.100	0	65.71	m^3/hr
Stripper liquid flow (stream 11)	XMV(8)	46.534	0	49.1	m^3/hr
Stripper steam valve	XMV(9)	47.446	0	100	%
Reactor cooling-water flow (stream 12)	XMV(10)	41.106	0	227.1	m^3/hr
Condensor cooling-water flow (stream 13)	XMV(11)	18.114	0	272.6	m^3/hr
Agitator speed	XMV(12)	50.000	150	250	rpm

Table 5.17. Continuously measured process variables.

Variable name	Variable number	Base-case value	Units
A feed (stream 1)	XMEAS(1)	0.2505	kscmh
D feed (stream 2)	XMEAS(2)	3664	kg/hr
E feed (stream 3)	XMEAS(3)	4509.3	kg/hr
A and C feed (stream 4)	XMEAS(4)	9.3477	kscmh
Recycle flow (stream 8)	XMEAS(5)	26.902	kscmh
Reactor feed rate (stream 6)	XMEAS(6)	42.339	kscmh
Reactor pressure	XMEAS(7)	2705	kPa gauge
Reactor level	XMEAS(8)	75	%
Reactor temperature	XMEAS(9)	120.4	°C
Purge rate (stream 9)	XMEAS(10)	0.3371	kscmh
Product separator temperature	XMEAS(11)	80.109	°C
Product separator level	XMEAS(12)	50	%
Product separator pressure	XMEAS(13)	2633.7	kPa gauge
Product separator underflow (stream 10)	XMEAS(14)	25.16	m^3/hr
Stripper level	XMEAS(15)	50	%
Stripper pressure	XMEAS(16)	3102.2	kPa gauge
Stripper underflow	XMEAS(17)	22.949	m^3/hr
Stripper temperature	XMEAS(18)	65.731	°C
Stripper steam flow	XMEAS(19)	230.31	kg/hr
Compressor work	XMEAS(20)	341.43	kW
Reactor cooling-water outlet temperature	XMEAS(21)	94.599	°C
Separator cooling-water outlet temperature	XMEAS(22)	77.297	°C

Table 5.18. Sampled process variables.

Variable name	Sample time	Variable number	Base-case value	Units
A : reactor feed (stream 6)		XMEAS(23)	32.188	mole %
B : reactor feed (stream 6)		XMEAS(24)	8.8933	mole %
C : reactor feed (stream 6)	0.1 hour	XMEAS(25)	26.383	mole %
D : reactor feed (stream 6)		XMEAS(26)	6.882	mole %
E : reactor feed (stream 6)		XMEAS(27)	18.776	mole %
F : reactor feed (stream 6)		XMEAS(28)	1.6567	mole %
A : purge gas (stream 9)		XMEAS(29)	32.958	mole %
B : purge gas (stream 9)		XMEAS(30)	13.823	mole %
C : purge gas (stream 9)		XMEAS(31)	23.978	mole %
D : purge gas (stream 9)	0.1 hour	XMEAS(32)	1.2565	mole %
E : purge gas (stream 9)		XMEAS(33)	18.579	mole %
F : purge gas (stream 9)		XMEAS(34)	2.2633	mole %
G : purge gas (stream 9)		XMEAS(35)	4.8436	mole %
H : purge gas (stream 9)		XMEAS(36)	2.2986	mole %
D : product (stream 11)		XMEAS(37)	0.0179	mole %
E : product (stream 11)		XMEAS(38)	0.8357	mole %
F : product (stream 11)	0.25 hour	XMEAS(39)	0.0986	mole %
G : product (stream 11)		XMEAS(40)	53.724	mole %
H : product (stream 11)		XMEAS(41)	43.828	mole %

Table 5.19. Process disturbances.

Variable number	Process variable	Type
IDV (1)	A/C feed ratio, B composition constant (stream 4)	step
IDV (2)	B composition, A/C feed ratio constant (stream 4)	step
IDV (3)	D feed temperature (stream 2)	step
IDV (4)	reactor cooling-water inlet temperature	step
IDV (5)	condenser cooling-water inlet temperature	step
IDV (6)	A feed loss (stream 1)	step
IDV (7)	C header pressure loss - reduced availability (stream 4)	step
IDV (8)	A, B, C feed compositions (stream 4)	random variation
IDV (9)	D feed temperature (stream 2)	random variation
IDV (10)	C feed temperature (stream 4)	random variation
IDV (11)	reactor cooling-water inlet temperature	random variation
IDV (12)	condenser cooling-water inlet temperature	random variation
IDV (13)	reaction kinetics	slow drift
IDV (14)	reactor cooling-water valve	sticking
IDV (15)	condenser cooling-water valve	sticking
IDV (16)	unknown	unknown
IDV (17)	unknown	unknown
IDV (18)	unknown	unknown
IDV (19)	unknown	unknown
IDV (20)	unknown	unknown

B. Development of Preliminary Control Strategy and Database Generation

The first step in developing a neural network model of the Tennessee Eastman problem requires the generation of a database that can be used for network training. We apply a PC-based process simulation program TUTSIM (TUTSIM Products, 1994) to model the Tennessee Eastman problem. Our TUTSIM simulation model for the original Tennessee Eastman problem without incorporating various control loops was developed by Karl E. Rony, formerly with the Siemen Industrial Automation and presently at WonderWare Inc., both at Johnson City, TN. This model is presently being used in process measurements and control courses at Virginia Tech by chemical engineering professor Dr. Peter R. Rony (1994), who kindly assisted us in obtaining this TUTSIM simulation model.

In order to develop a useful training database, we must initially have a preliminary control strategy that keeps the process operating during minor process disturbances. Although we focus our attention on the modeling and control of the reactor in this example, it is important to generate a global control scheme. Note that the reactor is unstable with respect to temperature and pressure variations from their setpoints. Consequently, incorporating proper control systems is a very critical step.

There is an increasing number of reported control studies of the Tennessee Eastman problem, especially at the annual meetings of the American Institute of Chemical Engineers held in November over the last few years. An incomplete list of reported studies includes Banerjee and Arkun (1993), Desai and Rivera (1993), Lyman (1992),

Lyman and Geogakis (1995), McAvoy and Ye (1994), Palarajjhala et al. (1994), Ricker (1993), and Ricker and Lee (1995a,b). We adopt the key features of two effective control strategies, developed by Lyman (1992) and McAvoy and Ye (1994), as a basis for our preliminary control flowsheet.

Figure 5.30 shows our preliminary control flowsheet with all attached controllers, "IC" representing the inner-loop controllers (fast-acting) and "OC" representing the outer-loop cascade controllers (slow-acting: adjusting setpoints). Tables 5.20 lists the setpoint, proportional (K), and integral (τ_i) controller settings for the inner- and outer-loop controllers, based on the control parameters of Lyman (1992) and McAvoy and Ye (1994). The setpoints of the inner-loop controllers have their corresponding outer-loop controllers (when applicable) listed in the table. Appendix 5.B lists the TUTSIM implementation of the control scheme that we add to Karl Rony's original TUTSIM simulation model of the Tennessee Eastman problem. Appendix 5.C shows the TUTSIM schematic block diagrams for the eleven manipulated variables (Table 5.16).

The USR blocks in the simulation model is a special block developed by Texas Instruments Application Productivity Tool (APT) group to link TUTSIM to a Texas Instrument controller. A copy of Karl Rony's USRBLK.EXE is required to run the simulation. We quote the following more detailed description by Rony and Rony (1991):

> Tutsim Products Inc. markets special software, the "User-Defined Block Option," that provides almost unlimited extensibility to the block-diagram

functions of TUTSIM. This "Option" allows a user to create a new type of TUTSIM block, called a USR block, that may add functions such as memory access to I/O hardware boards or, alternatively, may act as windows and linkages to complex code sections that are not limited to simple math functions. With USR blocks, different displays, recording devices, or I/O devices may be addressed and incorporated into TUTSIM model simulation.

To create a new TUTSIM block diagram function, a user completes a C or FORTRAN source-code skeleton that is then compiled into a file, USERBLK.EXE, that executes in parallel with the main TUTSIM executable file TUTEXEC.EXE on an IBM-class personal computer, preferably one equipped with a math coprocessor. During 1990-91, the Texas Instruments Industrial Systems Division in Johnson City developed special versions of USERBLK.EXE that could directly communicate with the memory locations within TI545 programmable logic controller.

We refer our readers to Rony et al. (1991) and Rony and Rony (1990) for further information on the use of the USR block with the TUTSIM simulation program.

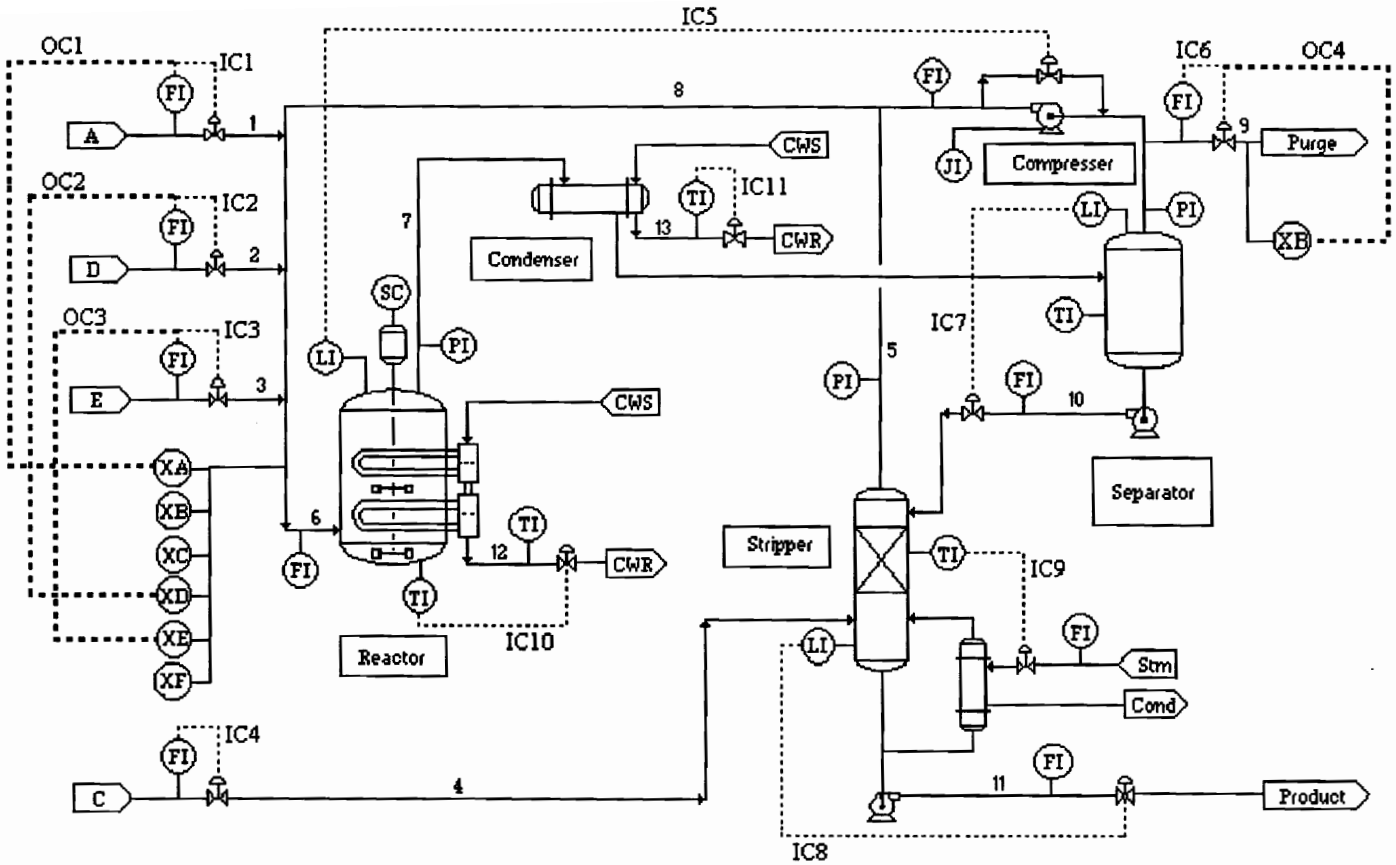


Figure 5.30. Preliminary control flowsheet for Tennessee Eastman problem.

Table 5.20. Inner- and outer-loop controller settings for the preliminary control flowsheet.

Inner-loop controllers (IC)

Variable	Controller	Setpoint	K	τ_i, sec
A-feed, kscmh	IC1	0.25052 (OC1)	200	6
D-feed, kg/hr	IC2	3664 (OC2)	0.002	18
E-feed, kg/hr	IC3	4509 (OC3)	0.002	18
C-feed, kscmh	IC4	9.347	0.1	18
reactor level, %	IC5	75	0.2	200
purge rate, kscmh	IC6	0.33712 (OC4)	100	18
separator level, %	IC7	50	-0.5	100
stripper level, %	IC8	50	-1	150
stripper temperature, °C	IC9	65.731	20	200
reactor temperature, °C	IC10	120.4	-20	200
condenser cooling water temperature, °C	IC11	77.297	-.36.6	60

Outer-loop cascade controllers (OC)

Variable	Controller	Setpoint	K	τ_i, sec
A - feed, mol %	OC1	32.188	0.001	----
D - feed, mol %	OC2	6.882	8	----
E - feed, mol %	OC3	18.776	1.5	----
B - purge, mol %	OC4	13.823	-0.005	----

To develop a training database with a wide range of feed compositions into the reactor, we introduce both positive and negative setpoint changes in each feed-stream setpoints (A-feed, D-feed, and E-feed flowrates), in addition to one run at base-case

operating conditions listed in Table 5.17. The three cascade controllers (OC1, OC2, and OC3) then bring the process back to steady-state conditions, with respect to feed compositions. The seven TUTSIM simulation runs for generating the database include:

Run 1: standard operating conditions

A-feed flowrate : 0.25052 kscmh

D-feed flowrate : 3664.0 kg/hr

E-feed flowrate : 4509.3 kg/hr

Run 2: step change in A-feed setpoint from 0.25052 to 0.10 kscmh

Run 3: step change in A-feed setpoint from 0.25052 to 0.40 kscmh

Run 4: step change in D-feed setpoint from 3664.0 to 3400.0 kg/hr

Run 5: step change in D-feed setpoint from 3664.0 to 4000.0 kg/hr

Run 6: step change in E-feed setpoint from 4509.3 to 4100.0 kg/hr

Run 7: step change in E-feed setpoint from 4509.3 to 4900.0 kg/hr

Note that the database should include many additional simulation runs to train the network for practical applications, although the preceding runs are sufficient for our purpose of demonstrating how to develop a process forecasting network for the Tennessee Eastman problem. In addition, the database should contain training examples that have larger feed-flowrate setpoint changes than the outer-loop cascade controllers yield. This will enable the network to extrapolate predictions when large process disturbances are induced.

We run each simulation for 30240 seconds (504 minutes) and record data for the database every 360 seconds (6 minutes), that corresponds to the sampling of the reactor feed analysis, giving 85 data records for each simulation. Every database for network training, used later in this example, has the training examples listed in the same order as described above (runs 1 to 7).

Figure 5.31 illustrates how the step change in feed-flowrate setpoint affects the reactor processing variables using run 1 (step change in A-feed flowrate setpoint from 0.25052 to 0.10 kscmh). In addition, this figure shows the range of operating conditions that occur throughout the processing time for a step change in A-feed flowrate. Specifically, Figure 5.31a displays how the control system forces the A-feed flowrate to adjust itself to a higher value close to the base-case value of 0.25052 kscmh through the outer-loop cascade controller OC1 for adjusting A-feed composition to 32.188%. This figure also illustrates how the reactor pressure varies with changing A-feed flowrate. Figure 5.31b shows the small variations of D-feed and E-feed flowrates to control their respective compositions through outer-loop cascade controllers OC2 and OC3, respectively. Figure 5.31c depicts the changes in feed compositions (mole % A, C, D) with processing time, with %A-feed and %E-feed eventually approaching their setpoint values of 32.188% and 18.776%, respectively, for outer-loop cascade controllers OC1 and OC3. Figure 5.31d gives a similar plot for B-, D-, and F-feed compositions. For this simulation, the reactor temperature is well-controlled at its setpoint of 120.4 °C.

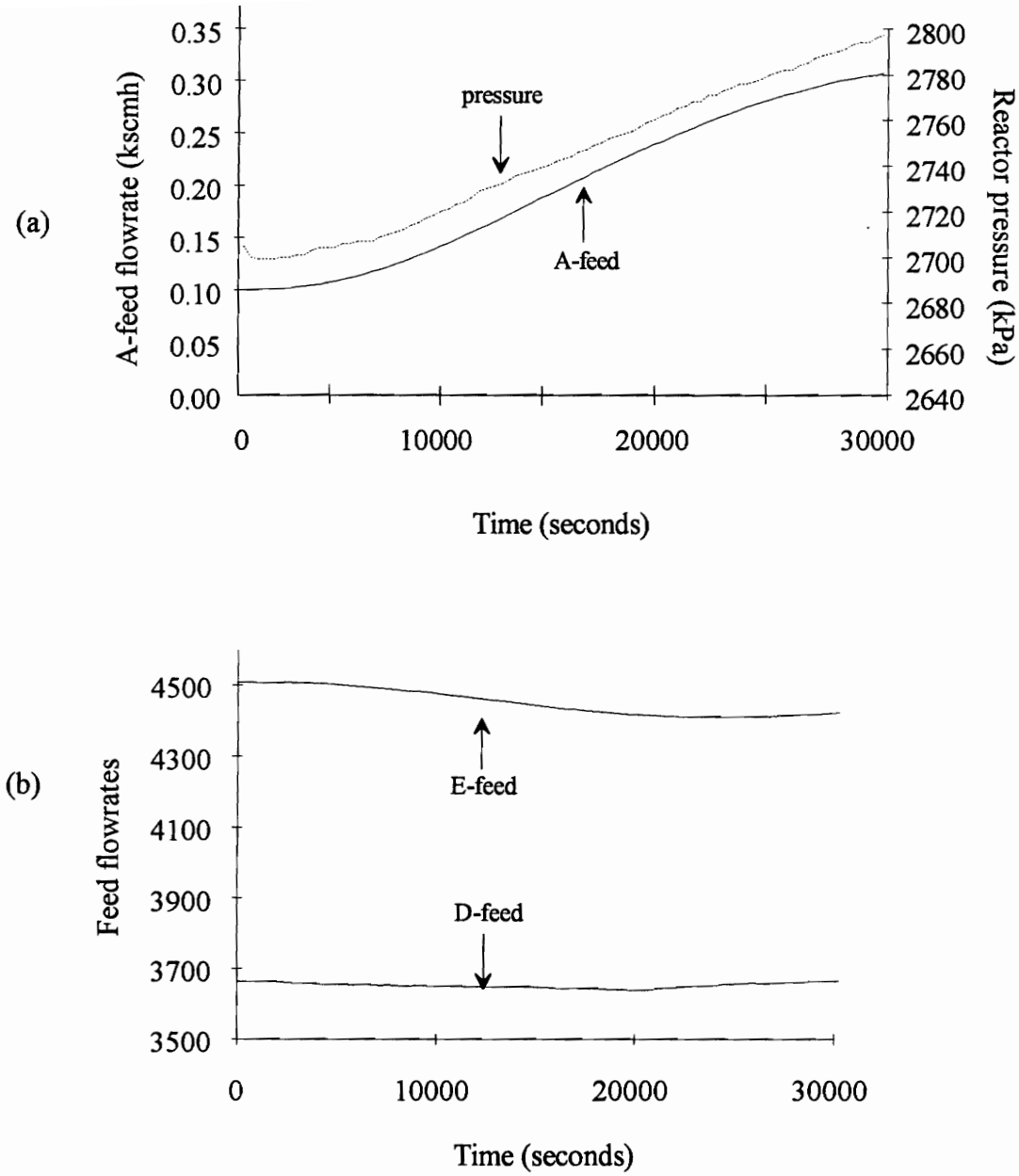
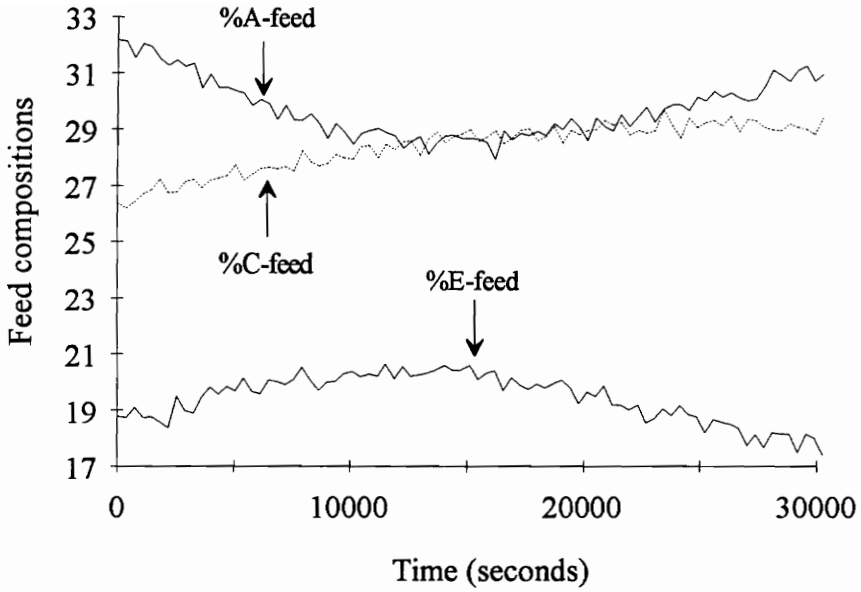


Figure 5.31a-d. Reactor processing simulation for run 1, step change in A-feed setpoint from 0.25052 to 0.10 kscmh: (a) A-feed flowrate and pressure; (b) D-feed and E-feed flowrates; (c) %A, %C, and %E feed; and (d) %B, %D, and %F feed.

(c)



(d)

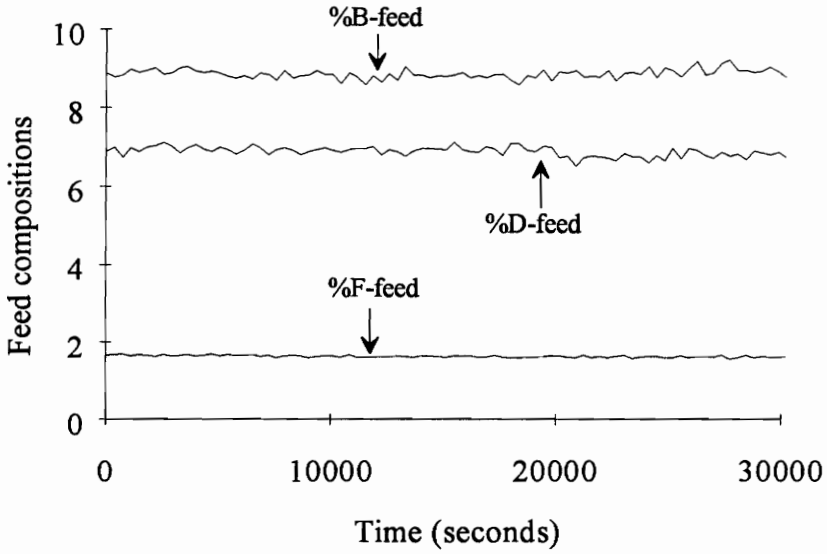


Figure 5.31a-d. Reactor processing simulation for run 1, step change in A-feed setpoint from 0.25052 to 0.10 kscmh: (a) A-feed flowrate and pressure; (b) D-feed and E-feed flowrates; (c) %A, %C, and %E feed; and (d) %B, %D, and %F feed.

C. Replacing Three Feed-Flowrate Outer-Loop Cascade Controllers with One Supervisory Controller

Now that we have developed a preliminary control flowsheet for the problem, we will show how to replace three feed-flowrate, outer-loop controllers with one supervisory controller. Briefly, supervisory control involves resetting the setpoints of a local controller according to some computer or predictive algorithm (Seborg et al., 1989; pp. 474-501). Figure 5.32 shows the process control flowsheet with a supervisory controller replacing the three cascade controllers. In this flowsheet, instead of having the A-feed composition determine the setpoint for A-feed flowrate using an outer-loop cascade PID controller (the same for D-feed and E-feed flowrates), we can utilize a neural network predictor to determine the optimal setpoints for the three flowrates from the six compositions (%A, %B, ..., %F) and the present setpoints of the three feed flowrates. In essence, we use a neural network predictor, which involves multiple inputs and multiple outputs (MIMO), to replace three conventional, single-input/single-output (SISO) predictors (i.e., three feed-flowrate, outer-loop controllers) to determine the optimal controller settings. This MIMO approach is advantageous, since in a complex chemical reacting process like the Tennessee Eastman problem, each component flowrate does affect the compositions of the other components, and some of the three SISO cascade controllers may have undesirable impact on other parts of the process.

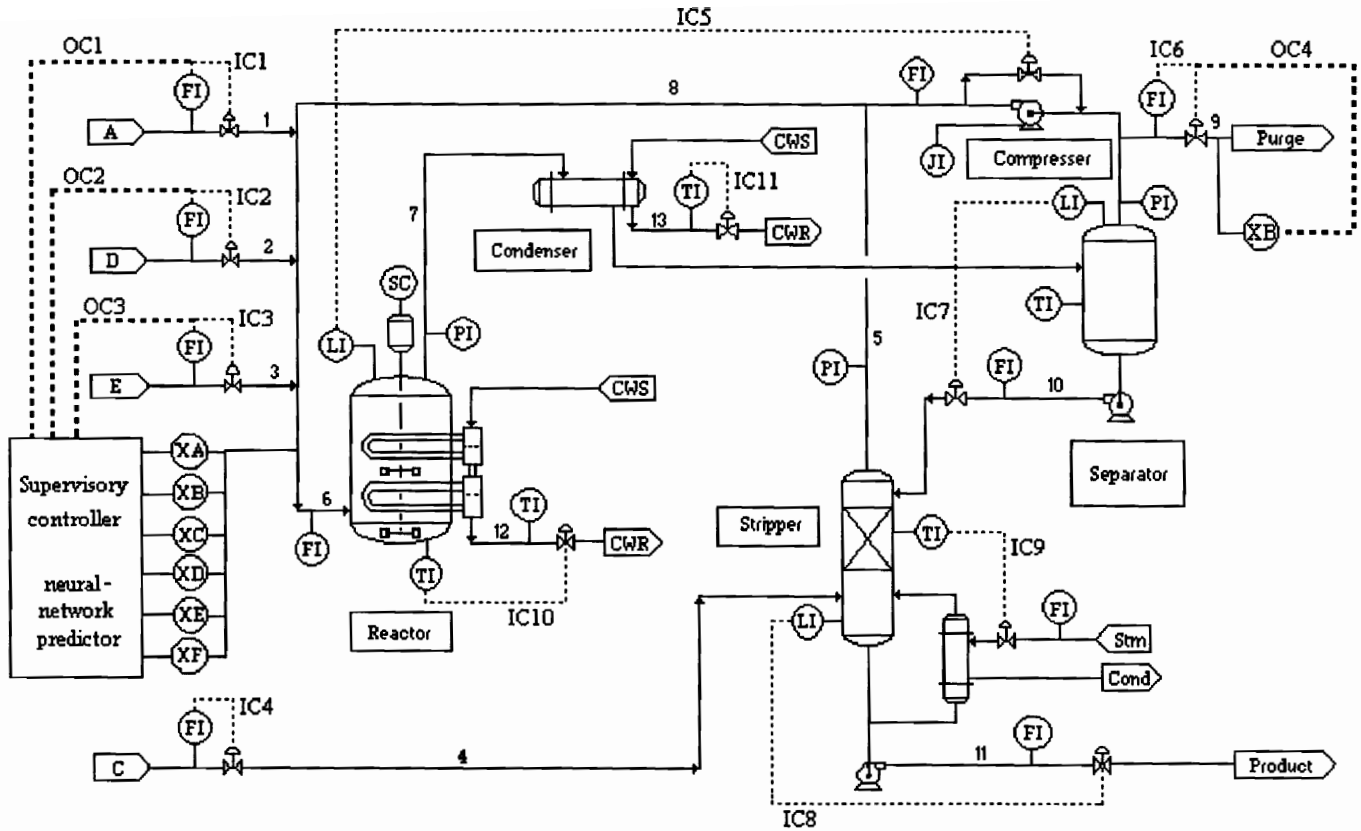


Figure 5.32. Tennessee Eastman process control flowsheet with a neural network predictor as a supervisory controller replacing three feed-flowrate, outer-loop cascade controllers.

D. Neural Network Modeling of the Reactor

In this section, we first develop a neural network predictor for feed compositions based on present feed compositions and feed flowrates, that can be used by a supervisory controller. Next, we develop an additional neural network for predicting pressure from reactor-feed compositions. Finally, we show how to link these neural networks to unit operations of the Tennessee Eastman plantwide control problem.

1. Feed-Composition Network

a. Network Architecture

We use the standard architecture of the recurrent network for process forecasting (described in Section 5.3) for the feed-composition network (Figure 5.33), except that we do not use the recurrent loops. These loops are not required for this type of process-control example because we are only interested in determining the flowrate setpoints for one time step ahead. For example, after measuring the feed compositions, we want to determine the best setpoints for the four flowrates to produce the specified feed compositions (setpoints), six minutes in the future.

This network has two hidden layers with 30 nodes in layer 1 and 15 in layer 2. The three feed flowrates (A-, D-, and E-feed) and recycle flowrate are manipulated

variables, **u**(t), and the six feed compositions (%A, %B, ..., %F) are measured variables, **y**(t). The time interval between network predictions is $\Delta t = 6$ minutes, which corresponds to that of the process measurements. Since we have fast-acting inner-loop controllers and use a sufficiently large Δt, we do not require more than the present feed compositions in the input layer (i.e., moving-window size of one time step). Note that this is not always the case for other problems and the moving-window size should be selected with care.

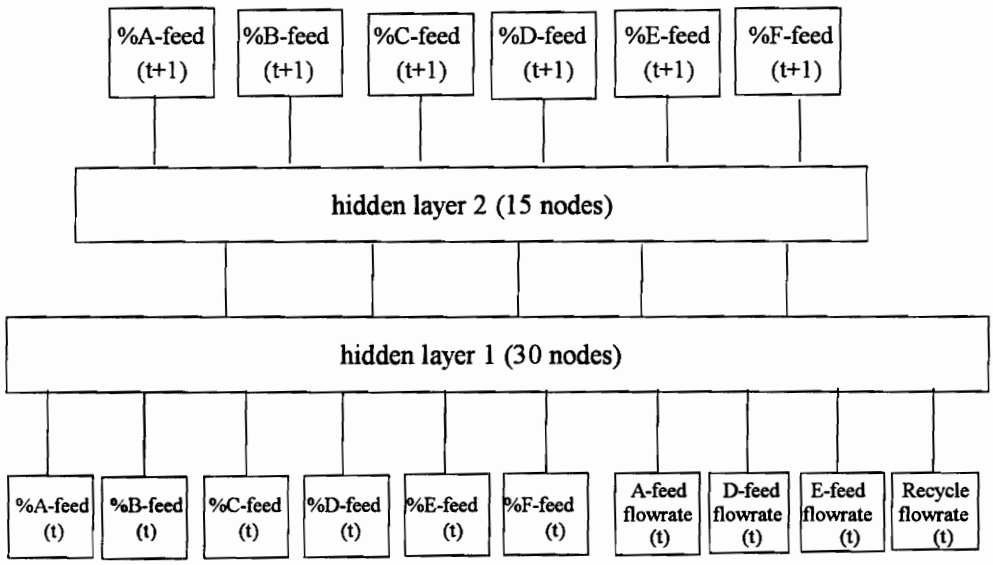


Figure 5.33. The architecture of the feed-composition network.

b. Network Training and Results

To train the network, we use the seven TUTSIM simulation runs described in Section 5.5.B consisting of 588 training examples, 84 for each run, one less than the 85 data points in the simulation due to the one-step time delay in the network. Table 5.21 lists the format of the six data files *acomp.nna*, *bcomp.nna*, *ccomp.nna*, *dcomp.nna*, *ecomp.nna*, and *fcomp.nna* used to train the feed-composition network.

Table 5.22 summarizes the primary specifications used in training the feed-composition network. We use a standard backpropagation network with the hyperbolic tangent transfer function and the delta learning rule.

Figure 5.34a-f show the scatter plots of the predicted versus actual feed compositions using the feed-composition network. All six networks are trained for 50,000 iterations. The six scatter plots demonstrate good prediction capability for %A-feed, %C-feed, and %E-feed, with more errors for %B-feed, %D-feed, and %E-feed. This is primarily due to fluctuations in recycle compositions caused by downstream effects from the setpoint changes in feed flowrates. Their is also a significant quantity of random noise within all measured and manipulated variables that must be filtered. The feed-composition network (Figure 5.33) can easily be expanded to include the recycle composition (see Figure 5.35) to improve the prediction capability of the network.

Table 5.21. The format of the six data files *acomp.nna*, *bcomp.nna*, *ccomp.nna*, *dcomp.nna*, *ecomp.nna*, and *fcomp.nna* used to train the feed-composition network.

Column number	Variable type	Normalized variable	Normalization factor	file name
1	input	%A-feed (t)	100%	
2	input	%B-feed (t)	100%	
3	input	%C-feed (t)	100%	
4	input	%D-feed (t)	100%	
5	input	%E-feed (t)	100%	
6	input	%F-feed (t)	100%	
7	input	A-feed flowrate (t)	1 kscmh	
8	input	D-feed flowrate (t)	10,000 kg/hr	
9	input	E-feed flowrate (t)	10,000 kg/hr	
10	input	recycle flowrate (t)	100 kscmh	
11a	input	%A-feed (t + 1)	100%	*acomp.nna*
11b	input	%B-feed (t + 1)	100%	*bcomp.nna*
11c	input	%C-feed (t + 1)	100%	*ccomp.nna*
11d	input	%D-feed (t + 1)	100%	*dcomp.nna*
11e	input	%E-feed (t + 1)	100%	*ecomp.nna*
11f	input	%F-feed (t + 1)	100%	*fcomp.nna*

Table 5.22. The specifications of the feed-composition network.

Network type	backpropagation
Training file name	*acomp.nna, bcomp.nna, ccomp.nna, dcomp.nna, ecomp.nna, fcomp.nna*
Transfer function (input layer)	linear
Transfer function (hidden layers)	tanh
Transfer function (output layer)	tanh
Learning rule	delta rule
Summation	sum
Error	standard
Network weight distribution	normal distribution : 3 σ limits of [-1, 1]

input layer

Training iteration	5,000
Noise	0
Learning rate	0.9
Momentum coefficient	0.6
Error tolerance	0

hidden layer 1

Training iteration	10,000	30,000	70,000
Noise	0	0	0
Learning rate	0.3	0.15	0.04
Momentum coefficient	0.4	0.2	0.05
Error tolerance	0.1	0.1	0.1

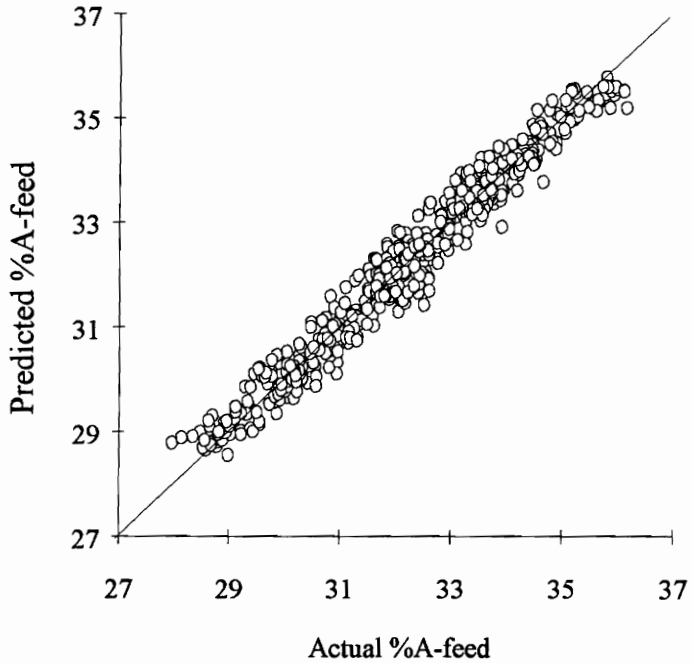
hidden layer 2

Training iteration	10,000	30,000	70,000
Noise	0	0	0
Learning rate	0.25	0.13	0.03
Momentum coefficient	0.4	0.2	0.05
Error tolerance	0.1	0.1	0.1

output layer

Training iteration	10,000	30,000	70,000
Noise	0	0	0
Learning rate	0.15	0.08	0.02
Momentum coefficient	0.4	0.2	0.05
Error tolerance	0.1	0.1	0.1

(a)



(b)

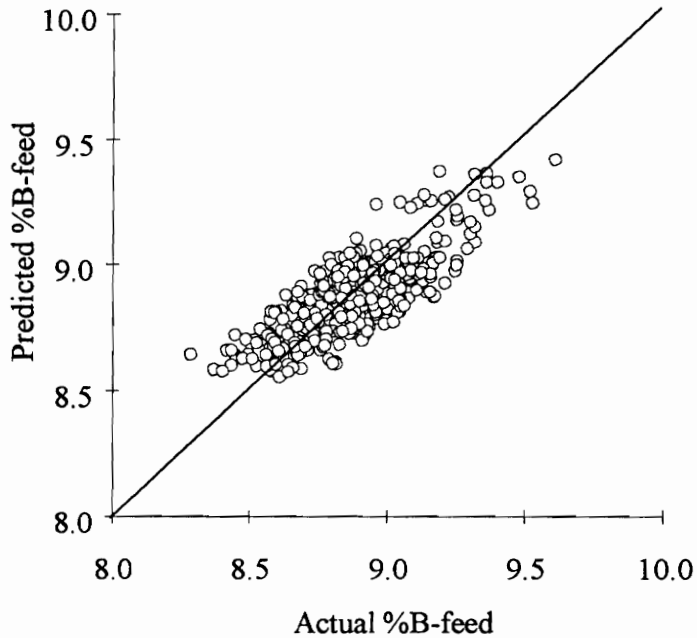
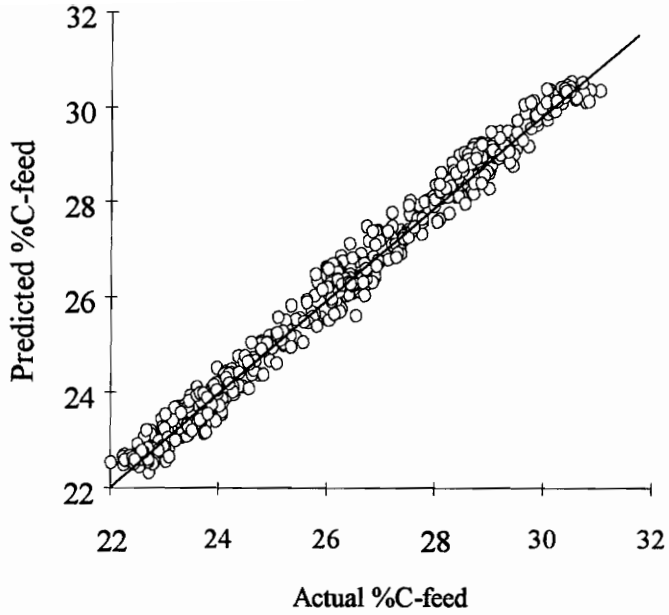


Figure 5.34a-f. Scatter plots of the predicted versus actual feed compositions using the feed-composition network. (a) %A-feed, (b) %B-feed, (c) %C-feed, (d) %D-feed, (e) %E-feed, and (f) %F-feed.

(c)



(d)

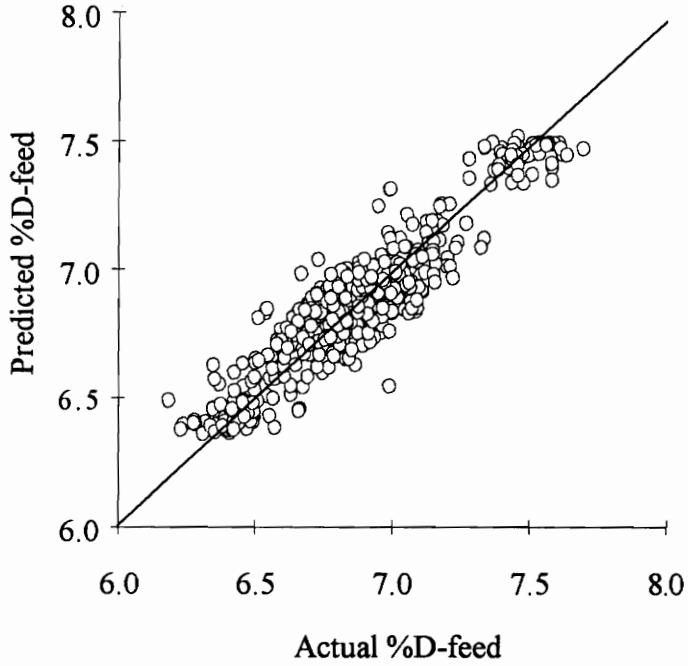
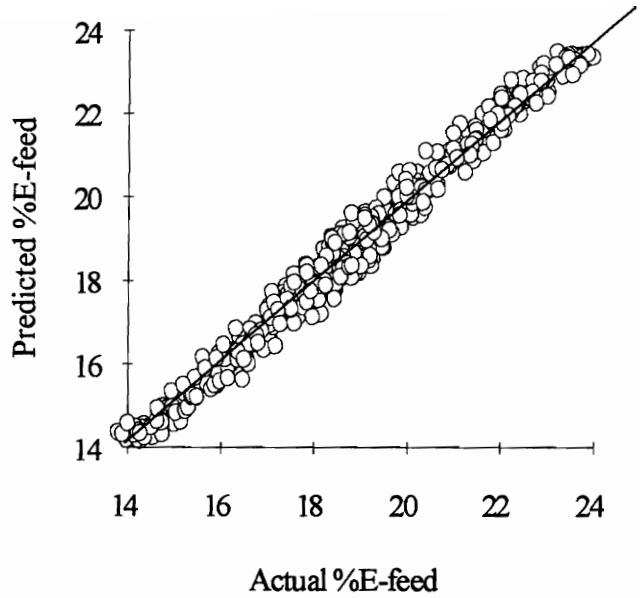


Figure 5.34a-f. Scatter plots of the predicted versus actual feed compositions using the feed-composition network. (a) %A-feed, (b) %B-feed, (c) %C-feed, (d) %D-feed, (e) %E-feed, and (f) %F-feed.

(e)



(f)

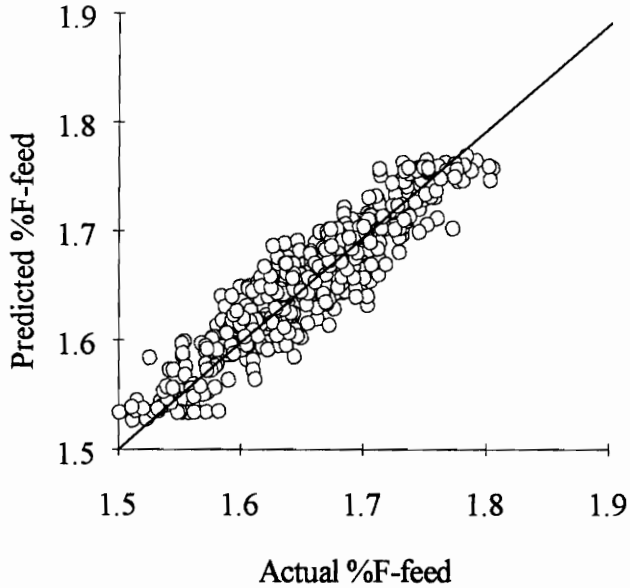


Figure 5.34a-f. Scatter plots of the predicted versus actual feed compositions using the feed-composition network. (a) %A-feed, (b) %B-feed, (c) %C-feed, (d) %D-feed, (e) %E-feed, and (f) %F-feed.

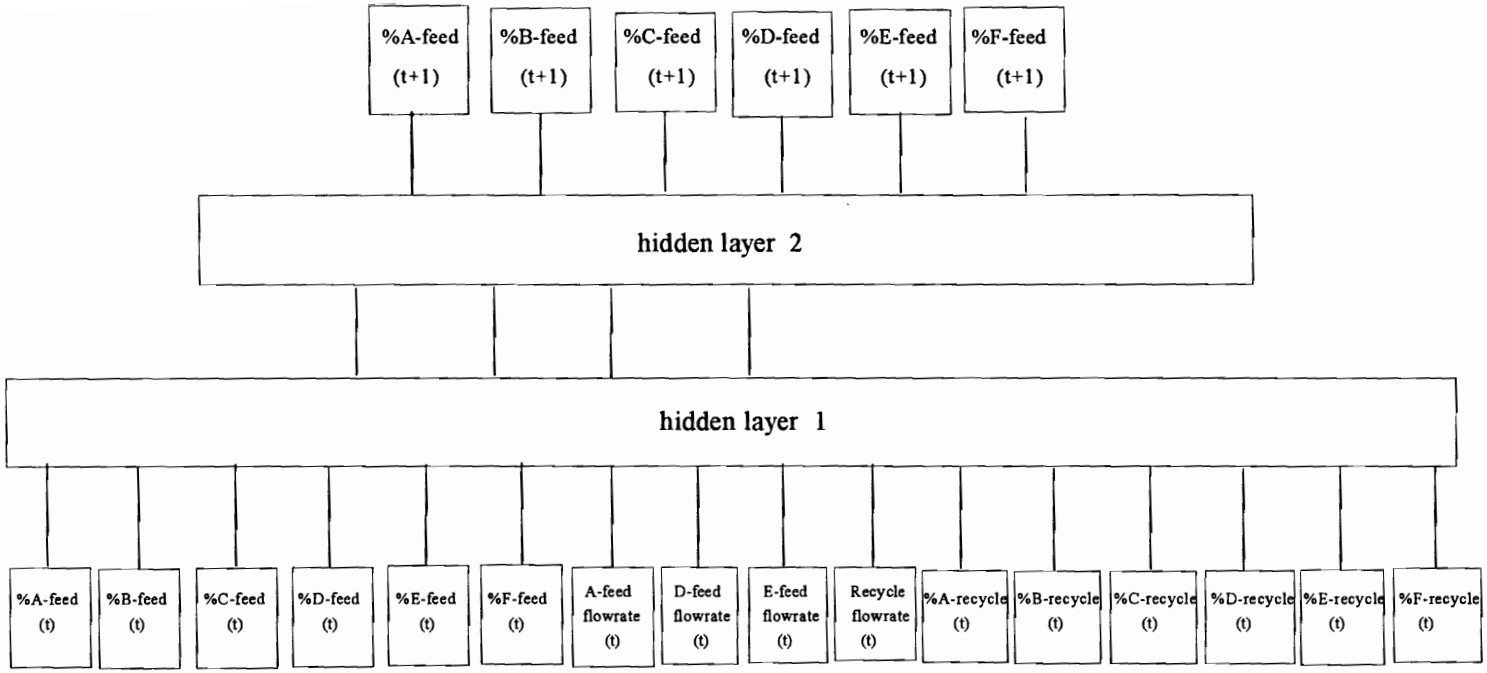


Figure 5.35. The feed-composition network (Figure 5.33) expanded to include the recycle composition.

c. Illustrative Predictions of the Feed-Composition Network

To demonstrate how to use the feed-composition network to identify feed flowrate setpoints, we give a simple example where we introduce a 0.25% decrease in the D-feed composition and a 0.25% increase in the E-feed composition from their steady-state values. The other four feed compositions are set at their steady-state values, giving the following feed compositions: %A-feed - 32.19 %; %B-feed - 8.89 %; %C-feed - 26.40 %; %D-feed - 6.63 %; %E-feed - 19.03 %; and %F-feed - 1.66 %. We then use the network to generate a matrix of predicted D-feed and E-feed compositions (i.e., inner-loop controller setpoints). Our goal is to identify the optimal D-feed and E-feed flowrate setpoints that will drive the disturbed feed compositions to their desired levels (%D-feed to 6.882% and %E-feed to 18.776%). The A-feed and recycle flowrate setpoints are kept at their steady-state values during this study.

Table 5.23 gives the predicted D-feed and E-feed compositions over a wide range of feed-flowrate setpoints: D-feed from 3500 to 5500 kg/hr and E-feed from 2500 to 4500 kg/hr. The desired A-feed compositions (%D-feed to 6.88% and %E-feed to 18.78%) lie within the range of 3500 to 4000 kg/hr for D-feed flowrate setpoint and 4000 to 4500 kg/hr for E-feed flowrate setpoint. Table 5.24 further shows the feed compositions over this desired range.

Table 5.23. Matrices of predicted D-feed and E-feed compositions as a function of D-feed and E-feed flowrate setpoints.

%D-feed

		E-feed flowrate (kg/hr)				
		2500	3000	3500	4000	4500
D-feed flowrate (kg/hr)	3500	7.29	7.18	7.00	6.75	6.60
	4000	7.49	7.45	7.39	7.31	7.26
	4500	7.56	7.54	7.51	7.50	7.52
	5000	7.58	7.57	7.55	7.56	7.60
	5500	7.59	7.58	7.57	7.58	7.63

%E-feed

		E-feed flowrate (kg/hr)				
		2500	3000	3500	4000	4500
D-feed flowrate (kg/hr)	3500	17.51	17.67	17.98	18.45	18.96
	4000	17.44	17.57	17.75	18.02	18.38
	4500	17.41	17.52	17.65	17.83	18.06
	5000	17.30	17.41	17.54	17.68	17.86
	5500	17.20	17.36	17.51	17.62	17.75

Table 5.24. Matrices of predicted D-feed and E-feed compositions as a function of D-feed and E-feed flowrate setpoints.

%D-feed

		E-feed flowrate (kg/hr)					
		4000	4100	4200	4300	4400	4500
D-feed flowrate (kg/hr)	3500	6.75	6.70	6.66	6.63	6.61	6.59
	3600	6.88	6.84	6.80	6.76	6.73	6.71
	3700	7.02	6.99	6.95	6.91	6.88	6.85
	3800	7.15	7.12	7.08	7.06	7.03	7.00
	3900	7.24	7.22	7.20	7.18	7.16	7.14
	4000	7.32	7.31	7.29	7.28	7.27	7.26

%E-feed

		E-feed flowrate (kg/hr)					
		4000	4100	4200	4300	4400	4500
D-feed flowrate (kg/hr)	3500	18.46	18.56	18.65	18.76	18.86	18.96
	3600	18.34	18.44	18.53	18.62	18.72	18.82
	3700	18.23	18.32	18.42	18.51	18.60	18.70
	3800	18.15	18.23	18.31	18.41	18.49	18.58
	3900	18.08	18.16	18.23	18.31	18.39	18.48
	4000	18.03	18.09	18.17	18.23	18.31	18.39

The E-feed flowrate setpoint gives the desired E-feed composition in the upper range between 4400 to 4500 kg/hr. The D-feed flowrate setpoint of approximately 3700 kg/hr yields the desired D-feed composition for the E-feed setpoint range (4400 to 4500 kg/hr), required to obtain the desired E-feed composition. To obtain more precise setpoints, an algorithm is required to define how each setpoint is determined. As will be discussed in Section 5.6.D on model-based control, there are much more sophisticated methods for using neural networks with supervisory controllers, although our use of response-surface matrices does give a simple representation.

2. Reactor-Pressure Network

a. Network Architecture

Now that we have developed the feed-composition network, we turn our attention to modeling the reactor pressure as a function the six inlet feed compositions (%A-feed, %B-feed, %C-feed, %D-feed, %E-feed, and %F-feed). The reactor-pressure network predicts the pressure at time t+1 using the output of the feed-composition network (Figure 5.33). Figure 5.36 shows the architecture of the reactor-pressure network having 30 nodes in hidden layer 1 and 15 nodes in hidden layer 2.

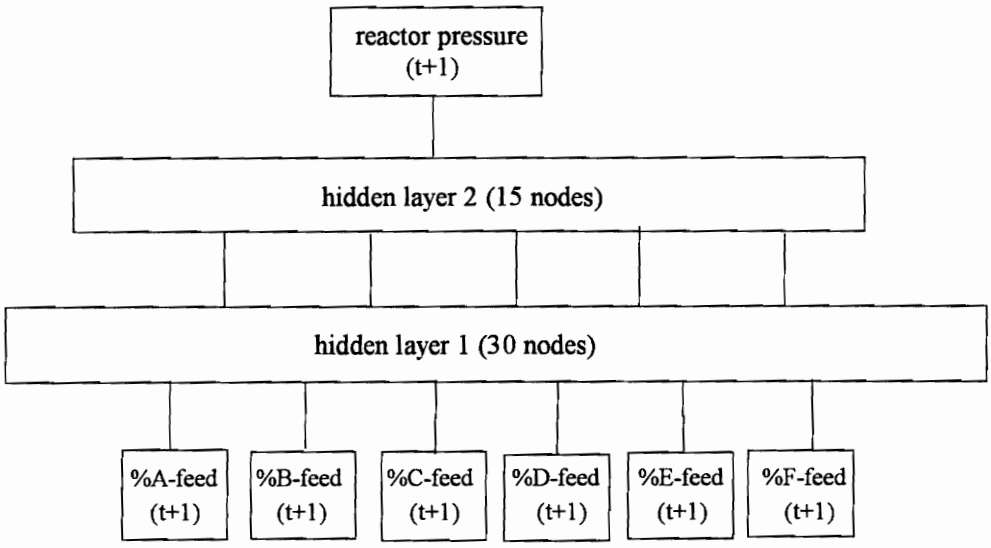


Figure 5.36. Reactor-pressure network architecture.

b. Network Training and Results

To train the reactor-pressure network, we use the same seven TUTSIM simulation runs described in Section 5.5.B. There are 85 training examples for each simulation and a total of 595 examples. Table 5.25 lists the format of the data file *rctprs.nna* used to train the network.

Table 5.26 shows the primary specifications used in training the reactor-pressure network.

Table 5.25. The format of data file *rctprs.nna* used to train the reactor-pressure network.

Column number	Variable type	Normalized variable	Normalization factor	file name
1	input	%A-feed (t + 1)	100%	
2	input	%B-feed (t + 1)	100%	
3	input	%C-feed (t + 1)	100%	
4	input	%D-feed (t + 1)	100%	
5	input	%E-feed (t + 1)	100%	
6	input	%F-feed (t + 1)	100%	
7	output	reactor pressure (t+1)	10,000 kPa	*rctprs.nna*

Table 5.26. The specifications of the reactor-pressure network.

Network type	backpropagation
Training file name	*rctprs.nna*
Transfer function (input layer)	linear
Transfer function (hidden layers)	tanh
Transfer function (output layer)	tanh
Learning rule	delta rule
Summation	sum
Error	standard
Network weight distribution	normal distribution : 3 σ limits of [-1, 1]

input layer	
Training iteration	5,000
Noise	0
Learning rate	0.9
Momentum coefficient	0.6
Error tolerance	0

hidden layer 1			
Training iteration	10,000	30,000	70,000
Noise	0	0	0
Learning rate	0.3	0.15	0.04
Momentum coefficient	0.4	0.2	0.05
Error tolerance	0.1	0.1	0.1

hidden layer 2			
Training iteration	10,000	30,000	70,000
Noise	0	0	0
Learning rate	0.25	0.13	0.03
Momentum coefficient	0.4	0.2	0.05
Error tolerance	0.1	0.1	0.1

output layer			
Training iteration	10,000	30,000	70,000
Noise	0	0	0
Learning rate	0.15	0.08	0.02
Momentum coefficient	0.4	0.2	0.05
Error tolerance	0.1	0.1	0.1

Figure 5.37 shows the scatter plot for the prediction of pressure by the reactor-pressure network trained for 50,000 iterations. As shown, the network has good prediction capability over the range of 2600 to 3000 kPa.

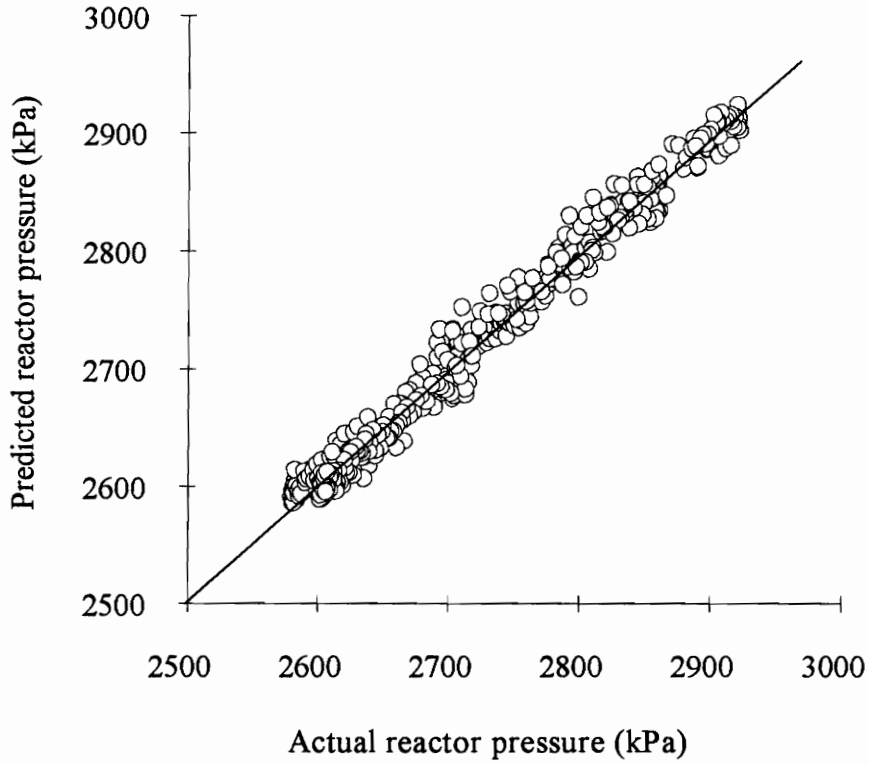


Figure 5.37. A scatter plot of predicted versus actual pressure using the reactor-pressure network.

3. Attachment of the Feed-Composition Network to the Reactor-Pressure Network

Now that we have developed both the feed-composition network and the reactor-pressure network, we illustrate how to link them into a hybrid network. Figure 5.38 shows the architecture of this hybrid network. This type of attachment of networks can be used throughout the process to link various unit operations and allow for global predictions.

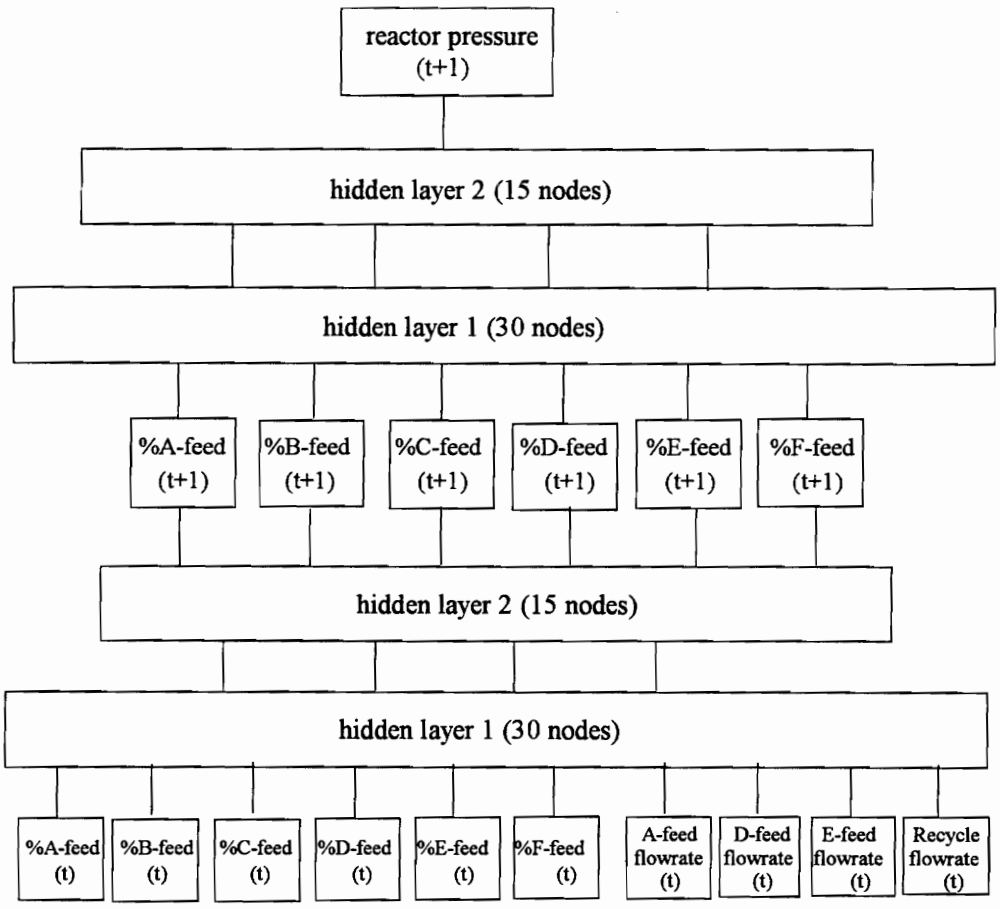


Figure 5.38. A hybrid network linking the feed-composition network to the reactor-pressure network.

E. Conclusions

- The Tennessee Eastman plantwide control problem is used to study neural network control strategies for complex chemical processes (Downs and Vogel, 1993). This process consists of five major unit operations, including a reactor, a product condenser, a vapor-liquid separator, a recycle compressor, and a product stripper.
- TUTSIM process control simulation software (TUTSIM Products, 1994) in conjunction with Karl Rony's ( ) USRBLK.EXE for the Tennessee Eastman simulation is used to generate a training database (7 simulation runs) to model the process.
- A neural network predictor (feed-composition network) in conjunction with a supervisory controller is able to replace three outer-loop feed-composition cascade controllers.
- The feed-composition network predicts the six feed compositions at time t+1 from the compositions at time t and the feed flowrates. We show how to use this network to identify the feed-flowrate setpoints to drive disturbed feed compositions to the desired feed composition levels.
- The reactor-pressure network predicts the pressure at time t+1 using the output of the feed-composition network. The reactor-pressure is then attached to the feed-composition network to demonstrate how to link neural network predictors in multiple-unit-operation processes.

5.6 Neural Networks for Process Control

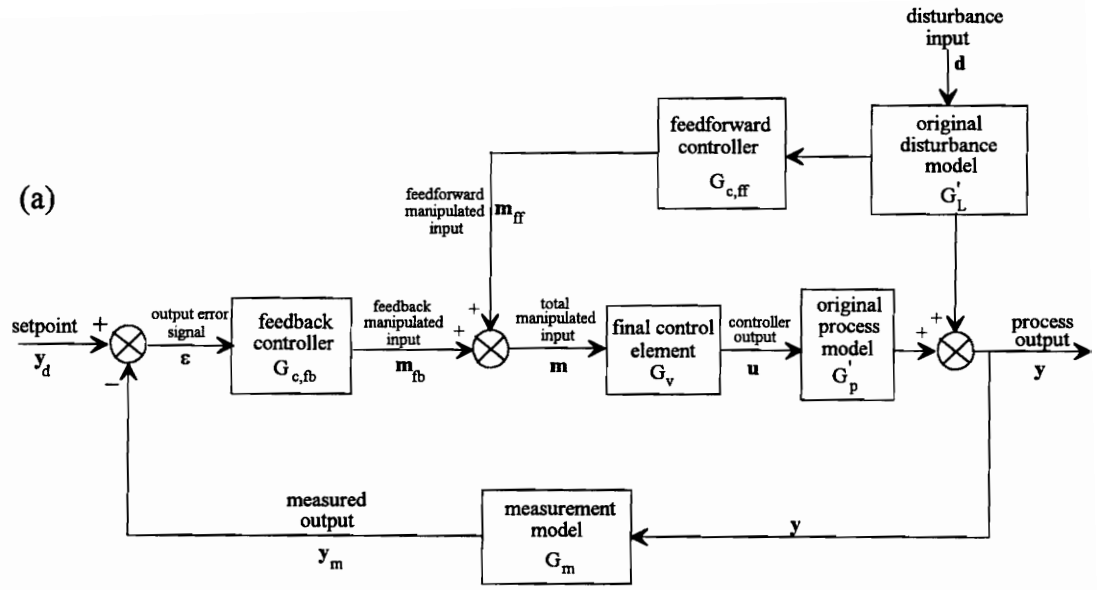
A. Overview of Process Control Applications

Process control has been by far the most popular area of neural network applications in bioprocessing and chemical engineering. Let us illustrate the role of neural networks in process control by considering a simplified block diagram of a process control system, Figure 5.39a, and its equivalent, Figure 5.39b. For convenience, we shall refer to the equivalent process and disturbance models incorporating the effects of the final control element and measurement dynamics simply as *the process and disturbance models*, that is:

$$\begin{bmatrix} \text{process} \\ \text{model} \\ G_p \end{bmatrix} = \begin{bmatrix} \text{final control} \\ \text{element} \\ G_v \end{bmatrix} \bullet \begin{bmatrix} \text{original} \\ \text{process model} \\ G_p' \end{bmatrix} \bullet \begin{bmatrix} \text{measurement} \\ \text{dynamics} \\ G_m \end{bmatrix} \tag{5.15}$$

and

$$\begin{bmatrix} \text{disturbance} \\ \text{model} \\ G_L \end{bmatrix} = \begin{bmatrix} \text{original} \\ \text{disturbance model} \\ G_L' \end{bmatrix} \bullet \begin{bmatrix} \text{measurement} \\ \text{dynamics} \\ G_m \end{bmatrix} \tag{5.16}$$



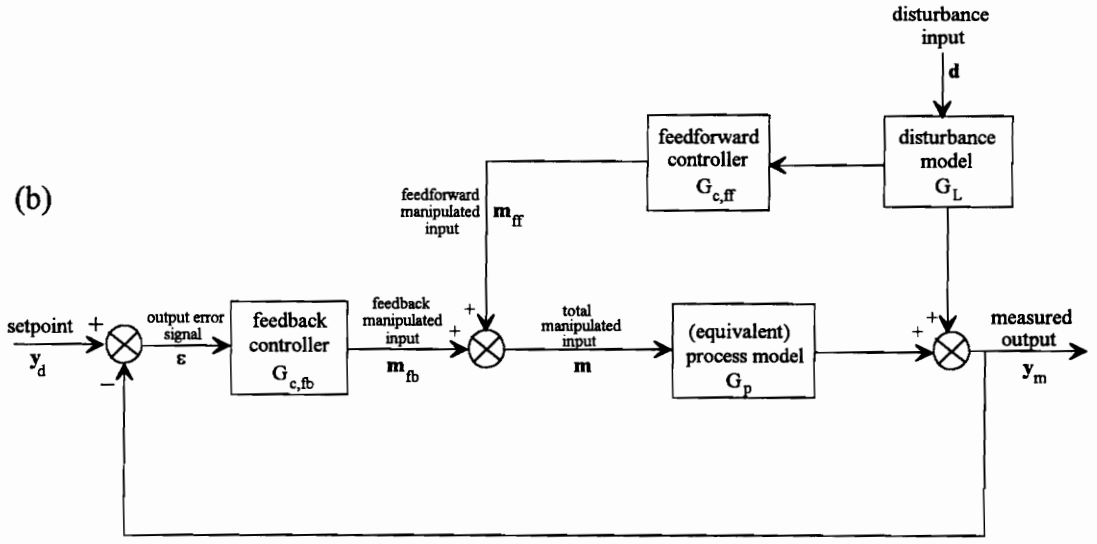


Figure 5.39. (a) A simplified block diagram for a process control system. (b) An equivalent block diagram with $G_p = G_v G_p' G_m$ and $G_L = G_L' G_m$.

Neural networks are applicable to process control to represent any of the model and controller blocks in Figure 5.39b, specifically, the process model G_p, disturbance model G_L, and feedback controller $G_{c,ff}$. Narendra and Mukopadhyag (1992) suggest that neural networks hold much promise for effectively handling three difficult problems in process control, namely, *complexity, nonlinearity, and uncertainty* (see also Section 1.5). In particular, the large variety of available neural network architectures permits us to deal with a wide range of process control problems. A well-structured and perhaps a large neural network can successfully tackle a complex process control problem. Because neural networks are nonlinear computing and modeling tools, they can apply to nonlinear process control problems with ease. Lastly, when compared to empirical curve-fitted models, neural networks are relatively less sensitive to noise and incomplete information, and can thus deal with process control problems with uncertainty.

Barto (1990), Werbos (1990b), and Hunt et al. (1992), among others, review different approaches to applying neural networks to process control. We may broadly classify various approaches into three categories.

1. *Direct network control*: training a neural network as the controller and determining the controller output directly.
2. *Inverse network control*: training a neural network as an "inverse" model of a process, i.e., predicting the process inputs necessary to produce the desired process outputs (setpoints).

3. *Indirect network control*: training a neural network to serve as a model of a process (as in the case of *model-based control*), or to determine the setpoints of a local controller (as in the case of *supervisory control*, illustrated in Section 5.5.C), thus indirectly affecting the process being controlled.

We can incorporate a neural network into a process control system in either an *adaptive* or a *nonadaptive* manner. In an adaptive scheme, we may update the network parameters (e.g., weight factors) through further training of the network over time to account for the drifts in the process conditions. A network is nonadaptive if after the network is trained for its task, the network parameters remain constant over time.

In the following, we give an overview of the preceding approaches to applying neural networks to process control.

B. Direct Network Control

1. Direct Network Control Scheme

Figure 5.40 illustrates a direct network controller, $G_{c,nn}$, being trained to mimic an existing feedback controller, $G_{c,fb}$. Given an output error signal $\boldsymbol{\varepsilon}$, that is, the difference between the desired output (setpoint) $\mathbf{y}_d$ and the measured output $\mathbf{y}_m$, the feedback controller produces the controller action $\mathbf{u}$ to minimize the output error. The neural controller $G_{c,nn}$

attempts to emulate the feedback controller $G_{c,fb}$, and it utilizes the same output error signal ε as an input. In addition, the neural controller incorporates, as an input, the network error signal ε_{nn}, which is the difference between the feedback controller output $\mathbf{u}$ and the neural controller output $\mathbf{u}_{nn}$, and is depicted as the arrow passing diagonally through the neural controller in Figure 5.40.

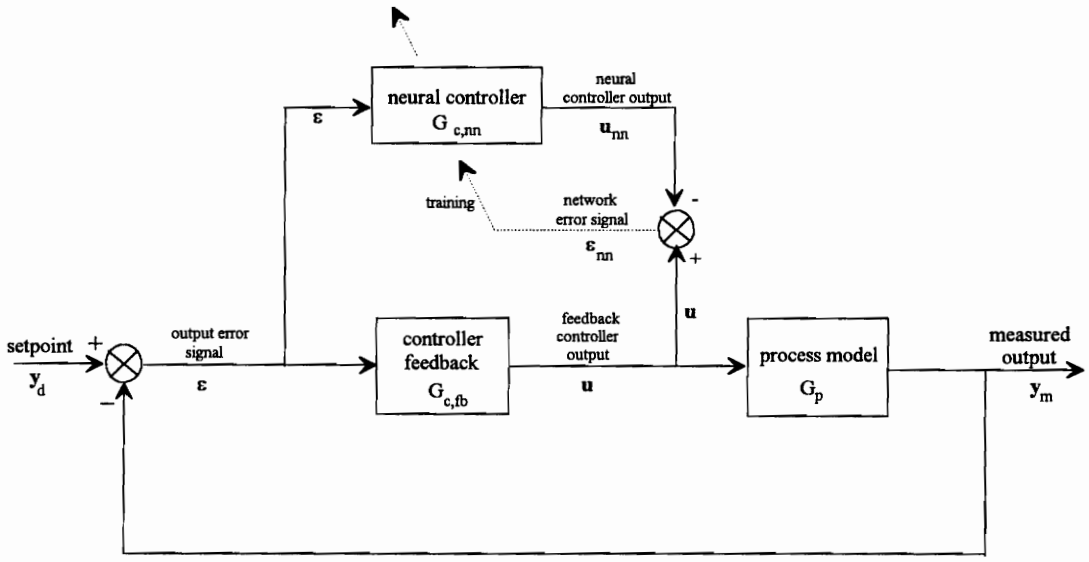


Figure 5.40. A direct network controller, $G_{c,nn}$, being trained to mimic an existing feedback controller, $G_{c,fb}$.

We can apply the direct network control scheme shown in Figure 5.40 in an online adaptive control system. We train the neural controller initially with the input-output response data from the existing feedback controller. This initial training ensures that the

neural controller will be able to provide accurate control output signals and process output responses. We then bring the neural controller online and further refine its network parameters with continuing training using new output error signals and network error signals.

To implement the direct network control scheme, a number of publications suggest the use of the current setpoint(s), the past output(s) and/or the past input(s) to the process as the network input(s). The output of the network is the controller action for the next time step. See, for example, Psaltis et al. (1988), Miller et al. (1990), Ichikawa and Sawa (1992), and Kooi and Khorasani (1992).

2. An Example of Commercial Implementation: The Intelligent Arc Furnace (IAF) Controller

We illustrate the development and implementation of an adaptive, direct network control scheme for optimizing the electrode positions in electric arc furnaces for steelmaking. Specifically, we describe the *Intelligent Arc Furnace (IAF) Controller* developed by the Neural Applications Corporation, Davenport, Iowa and implemented for controlling a large electric arc furnace (80 ton/hour, 16.6 feet in diameter, utilizing a 30-MVA transformer) at North Star Steel, Iowa (Staib and Staib, 1992; Staib, 1993). This commercial development and implementation of neural networks has revolutionized steelmaking, resulting in savings of millions of dollars per year per furnace, and was

named by the National Society of Professional Engineers as *one of the top six engineering achievements in the U.S. for 1992* (Hall, 1992).

Electric arc furnaces use electric power to melt steel. The amount of power delivered to the scrap metal is controlled by positioning three large electrodes. Figure 5.41 shows a schematic diagram of an electric arc furnace, indicating the positions of three large electrodes, called phases A to C.

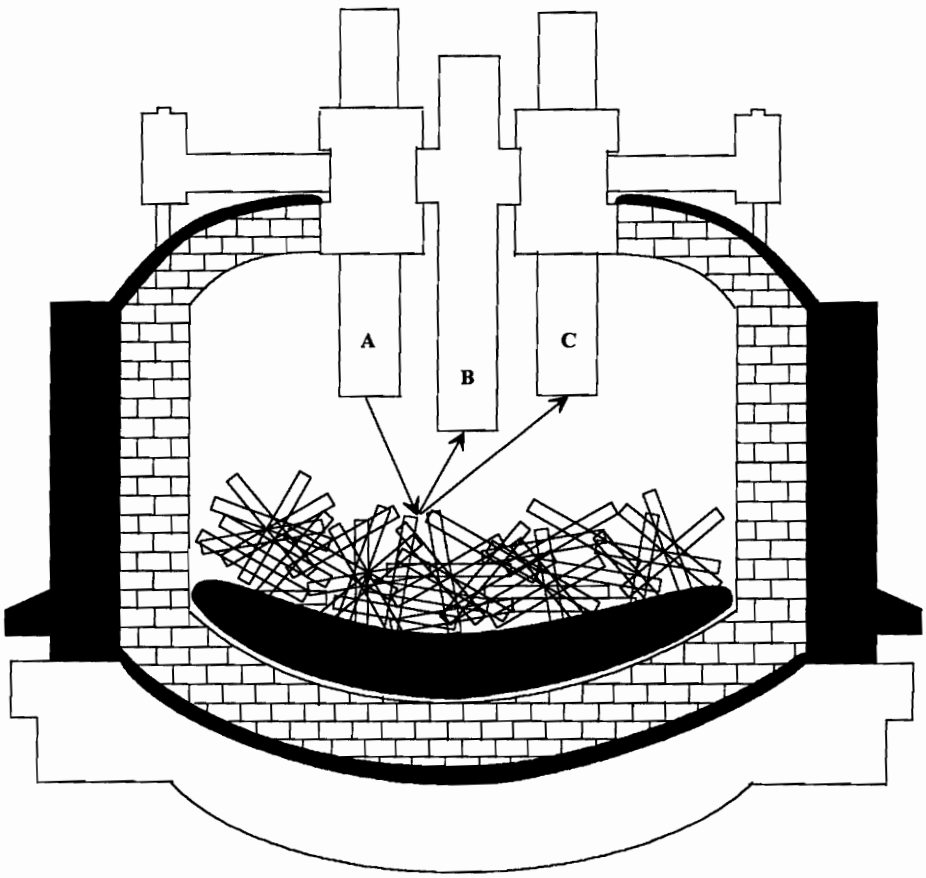


Figure 5.41. A schematic diagram of an electric arc furnace indicating the positions of three large electrodes, called phases A to C. Arrows show the direction of phase-A current flowing through phases B and C (Staib, 1993).

Considering only the current flow for phases A, we see that current flows from electrode A through the scrap metal and returns through electrodes (phases) B and C, as indicated by the arrows in the figure. Reducing the phase-A current will therefore lower the currents through phases B and C. Thus, to control effectively to a setpoint, a controller must account for the interactions among the three electrodes (phases). This controller requirement is called "three-phase aware" (Staib, 1993).

Because the electric arc furnace is a time-varying, complex nonlinear dynamic system, mathematical representations of the interrelationships among the three electrodes (phases) are practically impossible to derive, but a neural network can readily learn and continuously update a good furnace model from operating data. Previously, North Star Steel utilized a state-of-the-art closed-loop control computer, a modern electrode regulator, to control each of the three electrodes (phases) independently. This is illustrated in Figure 5.42a. In August, 1990, Neural Applications and North Star Steel initiated the research and development of a neural network controller, accounting for the complex interactions among the three electrodes, as illustrated in Figure 5.42b.

Figure 5.43a-c show the details of the development of the IAF controller. Specifically, Figure 5.43a illustrates the training of a neural controller emulator $G_{c,nn}$ to mimic the existing furnace controller (electrode regulator) $G_{c,fb}$. This initial step ensures that the neural controller initially minimize the network error signal, that is, the difference between the existing controller output and the neural controller output. Next, Figure 5.43b displays the training of a neural furnace emulator to model the actual furnace

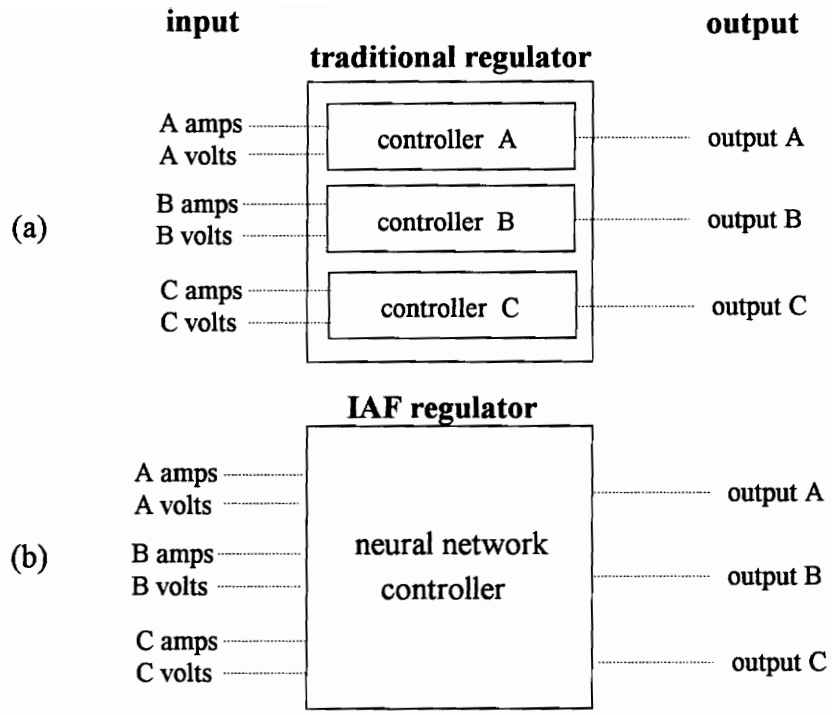


Figure 5.42. (a) A traditional regulator for controlling each of the three electrodes (phases) independently in an electric arc furnace; (b) An intelligent arc furnace (IAF) controller accounting for the complex interactions among the three electrodes (Staib, 1993).

dynamics. The goal is to minimize the network error signal, that is, the difference between the measured furnace output and the neural furnace emulator output, at a future time step. Finally, Figure 5.43c shows how to combine the preceding two neural networks into an integrated neural controller/furnace system. The resulting system is able to learn how to regulate the electrodes to achieve a set of furnace control setpoints.

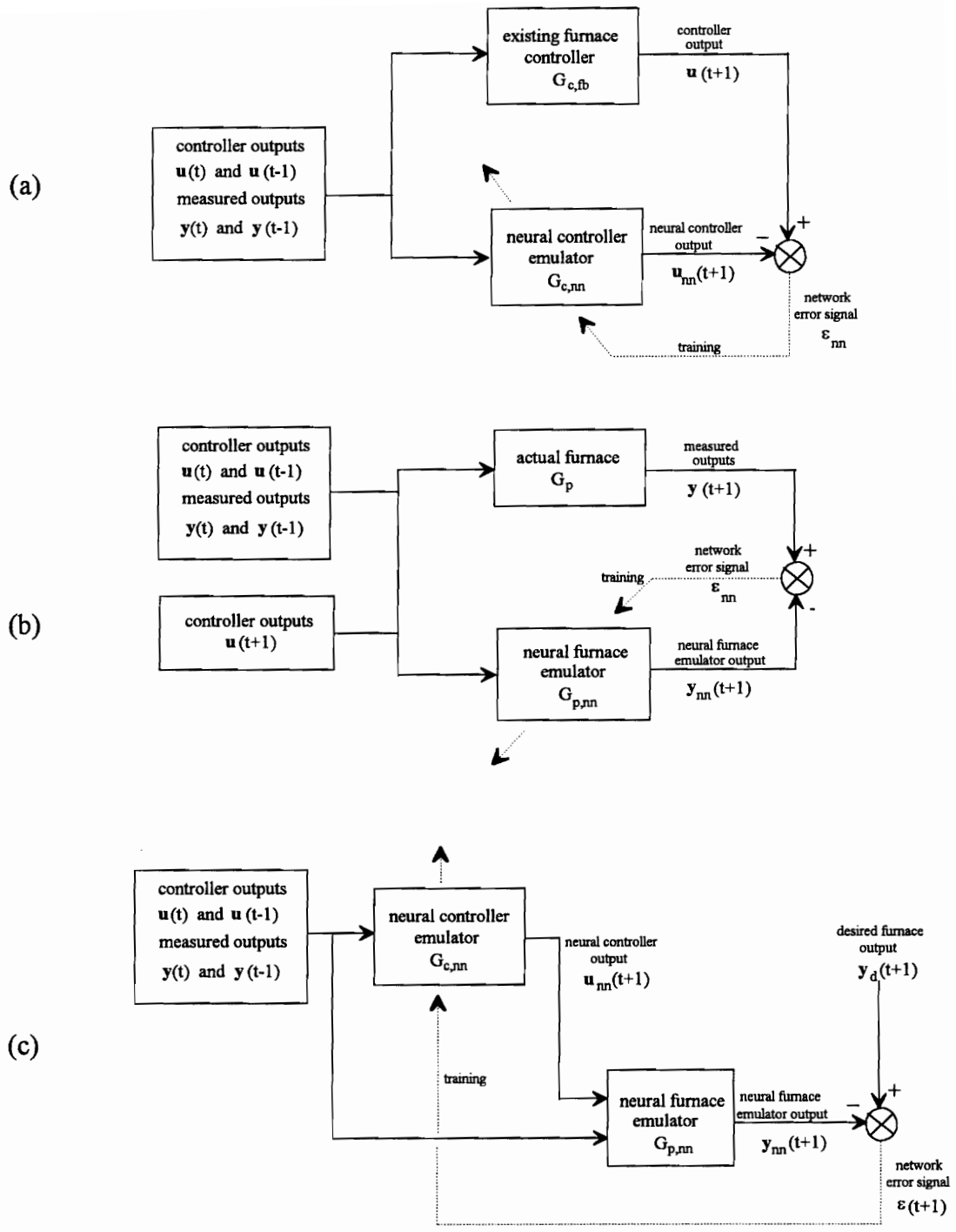


Figure 5.43a-c. The development of the Intelligent Arc Furnace Controller™. (a) the neural controller emulator, $G_{c,nn}$; (b) the neural furnace emulator, $G_{p,nn}$; (c) the integrated neural controller/furnace system (Staib and Staib, 1992).

Staib and Staib (1992) report that both the neural controller emulator and neural furnace emulator illustrated in Figure 5.43 consist of over 200 fully-interconnected nodes or processing elements. Incidentally, this neural network architecture is *larger* than any of the networks that we have used in various case studies in this text and than those networks in the reported studies in bioprocessing and chemical engineering literature. In addition, weight factors for both networks are updated every 15 seconds, or 150 data representations, thus allowing the networks to quickly recognize and adopt to changes in furnace operating conditions. Both networks are able to quantitatively model the complex interactions among the three electrodes after only 10 to 20 minutes of online training. The implementation of the IAF controller at North Star Steel and other steelmaking operations results in increased productivity, while reducing furnace and electrode wear, lowering power consumption and achieving an unprecedented smooth melt. Staib (1993) reports that overall savings from implementing the IAF controller have ranged from many thousands of dollars to tens of millions of dollars per furnace per year, and claims that "*the IAF controller has shown that neural networks are not only ready for the real world, but also that the world is in need of them.*"

Finally, it is worthwhile to mention that the IAF controller also incorporates unique expert-system safeguards (see Section 6.1.B for expert systems). Because of the enormous amount of power used by an electric arc furnace, it would not be safe to trust that the neural networks will always produce a "reasonable" output. Therefore, the IAF controller is integrated with an expert system that monitors the furnace conditions and

only allows neural network control during "normal" operation. When the furnace is not normal, such as when scrap has caved-in, creating a short circuit by touching an electrode, a rule-based expert-system regulator is invoked. Staib and Staib (1992) report that in all commercial applications thus far, the IAF neural networks have proven surprisingly robust and stable. Control is rarely switched to the rule-based mode, and the neural networks have never created any safety problems.

C. Inverse Network Control

1. Inverse and Forward Modeling by Neural Networks

Figure 5.44 illustrates a neural network being trained as an inverse model of a process, that is, to determine the process inputs necessary to produce the desired process outputs (setpoints). Specifically, the inputs to the network include: (1) the past and present process inputs, $\mathbf{u}(t-1)$ and $\mathbf{u}(t)$; (2) the past and present measured outputs, $\mathbf{y}(t-1)$ and $\mathbf{y}(t)$; and (3) the desired future process outputs, $\mathbf{y}_d(t+1)$. The network will predict the future process inputs, $\mathbf{u}_{nn}(t+1)$, necessary to produce the desired future outputs, $\mathbf{y}_d(t+1)$. Here, t-1 and t-2 mean $t-\Delta t$ and $t+\Delta t$, respectively, and Δt is the sampling time for output measurements. In essence, the inverse model corresponds to the following functional relationship, denoted by $g[\cdot]$ (Psaltis et al., 1988; Hunt et al., 1992):

neural inverse model $G^{-1}_{p,nn}$

$$\mathbf{u}_{nn}(t+1) = g\left[\mathbf{u}(t), \mathbf{u}(t-1), \mathbf{y}_d(t+1), \mathbf{y}(t), \mathbf{y}(t-1)\right] \tag{5.17}$$

The subscript nn refers to a neural network, and the subscript d indicates a desired output value (setpoint).

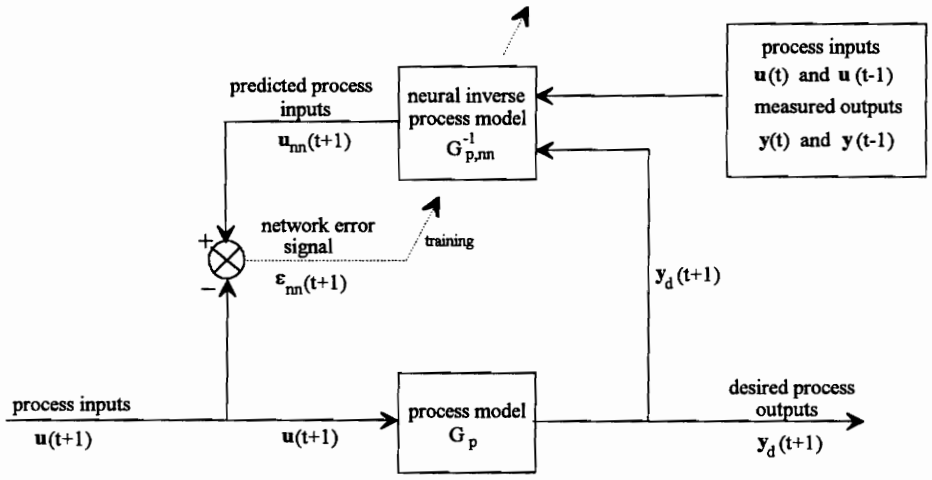


Figure 5.44. A neural network, $G^{-1}_{p,nn}$, being trained as an inverse model of a process with a model G_p.

By contrast, given the future process inputs, $\mathbf{u}(t+1)$, the original process model predicts the future process outputs, $\mathbf{y}(t+1)$, according to the following functional relationship, represented by f[•]:

process model G_p

$$\mathbf{y}(t+1) = f\,[\mathbf{u}(t+1), \mathbf{u}(t), \mathbf{u}(t-1), \mathbf{y}(t), \mathbf{y}(t+1)] \tag{5.18}$$

For convenience, we call equation 5.18 *the forward model*, and equation 5.17 *the inverse model*.

2. <u>An Illustrative Example of Inverse and Forward Modeling: Neutralization Operation in a Continuous Stirred-Tank Reactor</u>

We consider an example of the modeling the dynamic behavior of a neutralization operation in a continuous stirred-tank reactor (CSTR), shown in Figure 5.45. This operation involves two inlet streams, acetic acid CH_3COOH (flowrate F_1) and sodium hydroxide, NaOH (flowrate F_2). The resulting pH value of the outlet stream (flowrate $F_1 + F_2$) is governed by the acid-base equilibria and electroneutrality relationship, and the specific equations and parameters appear in Bhat and McAvoy (1990). By using the pH-F_2 response data generated by the governing equations, we can train neural networks as the forward and the inverse models of the neutralization operation:

forward model $G_{p,nn}$

$$pH_{nn}(t+1) = f\left[F_2(t+1), F_2(t), F_2(t-1), pH(t), pH(t-1)\right] \tag{5.19}$$

neural inverse model $G^{-1}_{p,nn}$

$$F_{2,nn}(t+1) = g\left[F_2(t), F_2(t-1), pH_d(t+1), pH(t), pH(t-1)\right] \tag{5.20}$$

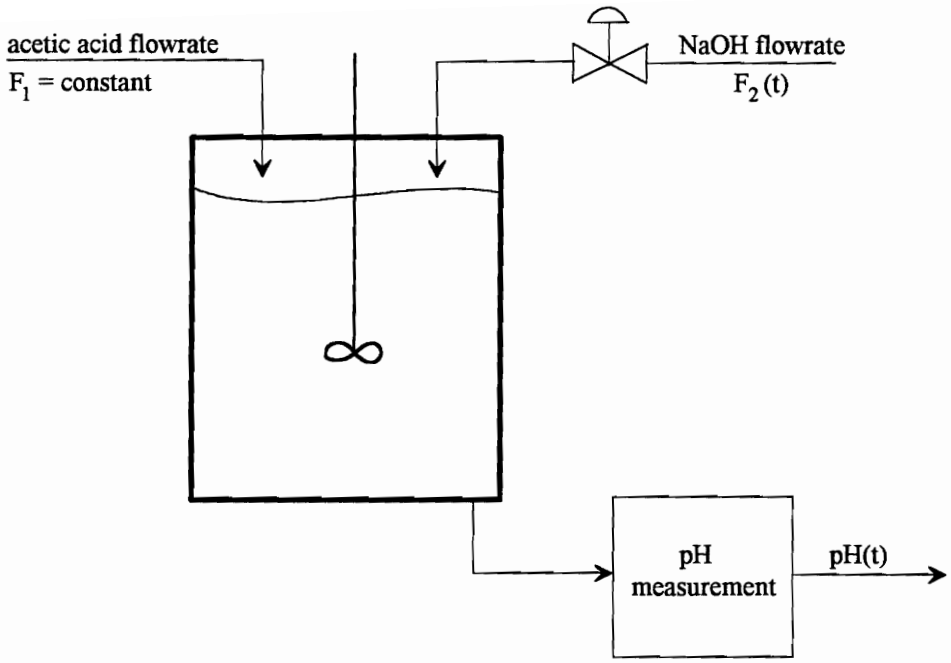


Figure 5.45. A neutralization operation in a continuous stirred-tank reactor (CSTR) (Bhat and McAvoy, 1990).

Figure 5.46 illustrates the known network inputs and unknown network output in the development of the forward and inverse models of the neutralization operation. We can readily use the recurrent network architecture of Figure 5.5 to implement both the forward and inverse modeling according to equations 5.19 and 5.20.

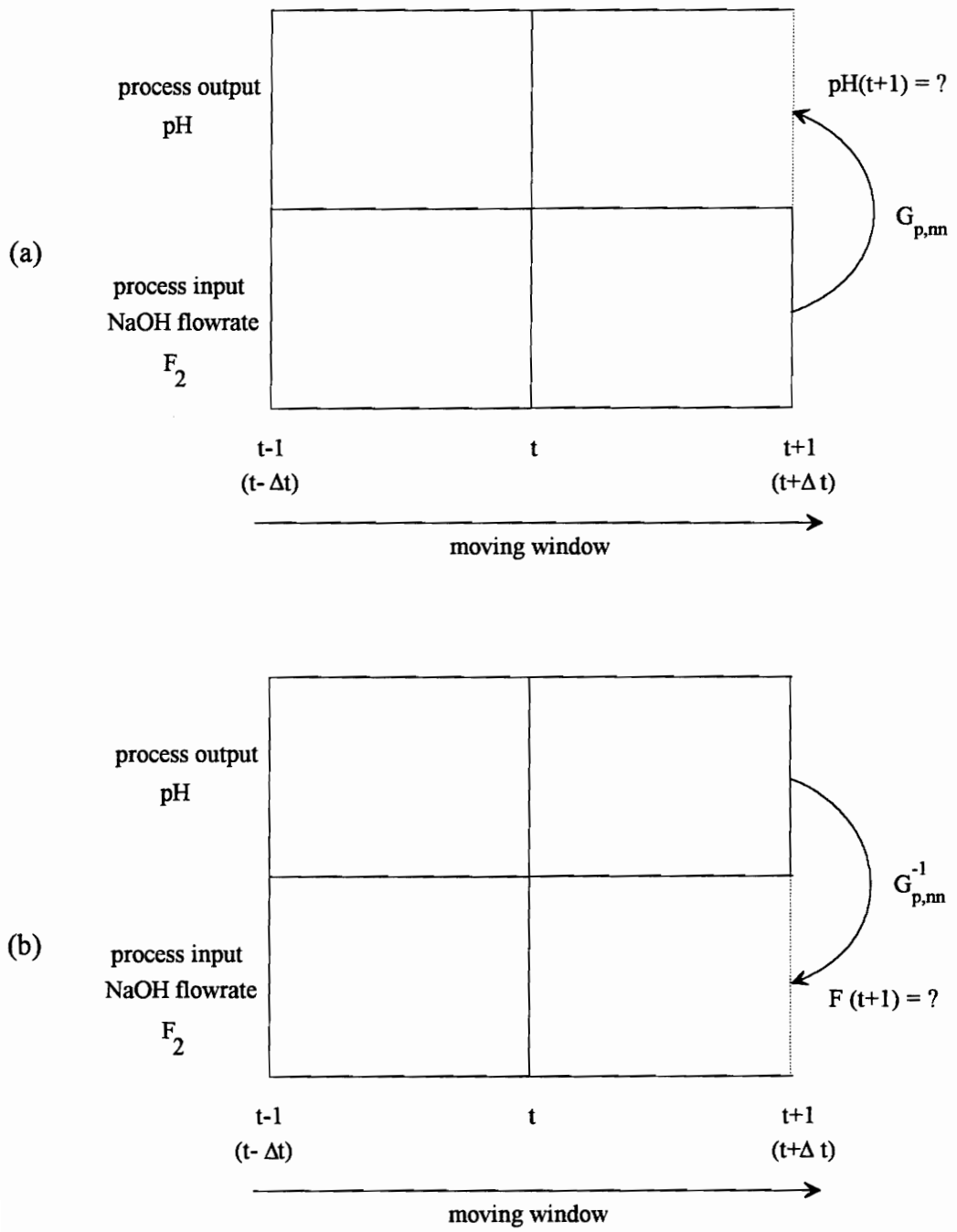


Figure 5.46. An illustration of the known network inputs (solid line) and unknown network output (dashed line) in the development of (a) the forward model, $G_{p,nn}$; and (2) the inverse model, $G^{-1}_{p,nn}$, of a neutralization operation.

3. Inverse Network Control Schemes for the Neutralization Operation

A number of investigators have described the schemes to implement the neural inverse model, $G_{p,nn}^{-1}$, of equation 5.20, in process control. See, for example, Barto (1990), Bhat and McAvoy (1990), Hosogi (1990), Hunt et al. (1992), Psichogios and Ungar (1991), Nahas et al. (1992), and Yuan and Guo (1994). We illustrate below two of the schemes using the neutralization example.

Figure 5.47 proposes a scheme to use a neural inverse model, $G_{p,nn}^{-1}$, as a feedforward controller based on the pH setpoint. As shown, the inverse network controller is parallel to a feedback controller, $G_{c,fb}^{-1}$. This control scheme follows the work of Hosogi (1990), but the practical aspects of its implementation need to be investigated further.

Perhaps the most popular application of the neural inverse model, $G_{p,nn}^{-1}$, in process control is to incorporate it as part of the so-called *internal model control (IMC)*, a model-based control scheme that we shall discuss in more detail in Section 5.6.D.2. Figure 5.48 shows a *simplified* IMC scheme employing both the neural forward and inverse models, $G_{p,nn}$ and $G_{p,nn}^{-1}$, for the neutralization operation (Bhat and McAvoy, 1990). This control scheme is called "simplified," since it ignores the effects of disturbance variables and process deadtime that are often present in process control. We include this scheme here to merely illustrate an implementation of the inverse network control.

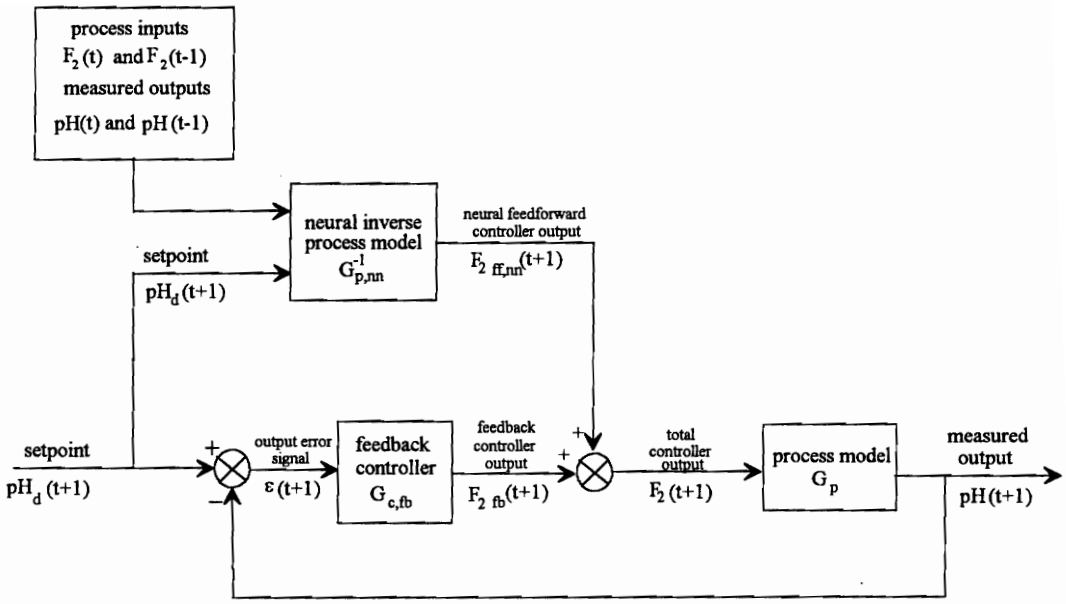


Figure 5.47. A neural network model, $G^{-1}_{p,nn}$, being used as a feedforward controller based on the setpoint in parallel with a feedback controller, $G_{c,fb}$, for the neutralization operation.

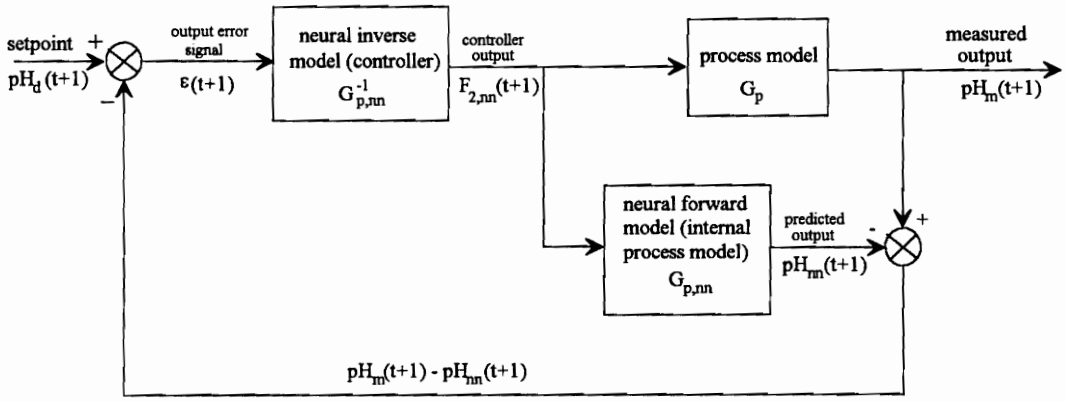


Figure 5.48. A simplified internal model control (IMC) scheme for the neutralization operation, employing both neural forward and inverse models (Bhat and McAvoy, 1990).

We must point out a limitation of the inverse network control schemes in process applications involving a dominant lag and/or a significant dead time. In general, mapping the process inverse (i.e., developing an inverse model of a process) from operating data for such situations is brittle and often results in errors. This follows because the inverse of a deadtime (requiring a time advance) or a lag (requiring pure derivative action) is not attainable with physical components in a control system. Therefore, some academicians (e.g., Stephanopoulos and Han, 1994) and practitioners (e.g., Shinskey, 1994) do not favor using control schemes with inverse process models, unless effective provisions accounting for process deadtime and dominant lag are incorporated. We shall discuss this aspect further in Section 5.6.D.2.

D. Model-Based Control

1. Model-Based Approaches to Process Control

In model-based control, we use a process model explicitly to aid in controller design. After specifying explicitly and precisely the desired output response from a process, we apply a process model to derive the required controller to obtain the prespecified output response. The process model may be identified by a classical technique, such as the impulse-response, step-response, or frequency-response method described in control texts

(e.g., Ogunnaike and Ray, 1994, Chapter 13), or by a new approach, such as the neural networks approach covered in this book.

Ogunnaike and Ray (1994) divide the model-based approaches to process control into two categories.

(1) *The direct synthesis approach*: the which the described output response is specified in the form of a reference trajectory, and the process model is used directly to synthesize the required controller to cause the process output to follow this trajectory *exactly*. Three of the specific techniques under this category for which neural networks are employed as process models are:

- *Direct synthesis control (DSC)*: given the models (e.g., transfer functions) of all the components (except the controller) of *a feedback loop*, synthesize the controller required to produce a specified closed-loop response.
- *Internal model control (IMC)*: given a model of the process *open-loop dynamics* in parallel with the process itself, synthesize a feedback controller required to produce a specified closed-loop response.
- *Generic model control (GMC)*: given a *steady-state, approximate model* of a process and a reference trajectory defined by first-order dynamics, synthesize the controller required to follow the trajectory according to an algebraic-integral equation of the output's deviation from its setpoint.

(2) *The optimization approach*: the desired output response is specified in the form of an objective function (which may or may not involve an output trajectory explicitly), and the process model is used to derive the required controller to *minimize* (or *maximize*) this objective function. It is also possible to incorporate some known operating constraints in this optimization objective. There are two important techniques under this category that utilize neural networks as process models:

- *Model-predictive control (MPC)*: given an unsteady-state model of the process and an objective function of the process involving economic data and operational constraints, predict explicitly the future process outputs and compute the appropriate corrective control action required to minimize the output deviation from its setpoint.
- *Optimal control (OC)*: Given an unsteady-state model of the process and an algebraic-integral objective function of the manipulated and controlled variables, and process disturbances, find the control policy to minimize (or maximize) the objective function subjected to operating constraints.

The literature on some of the model-based control techniques, such as IMC, MPC, and OC, is difficult to understand without a good foundation in control theory and applied mathematics. A full treatment of these techniques is beyond the scope of this text. We list, however, in Table 5.27 some suggested general textbook references for detailed

descriptions of these techniques together with representative papers that *employ neural networks as the required process models* described above to aid in controller design.

Table 5.27. Applications of neural networks to model-based control.

Model-based control approaches	Textbook references	Neural network papers
1. The direct synthesis approach		
• Direct synthesis control (DSC)	Ogunnaike and Ray (1994), pp. 648-64. Seborg et al. (1989), pp. 273-78, 622-32. Shinskey (1994), pp. 109-13.	Scott and Ray (1993)
• Internal model control (IMC)	Coughanower (1991), pp. 271-79. Luyben (1990), pp. 404-08, 609-11. Ogunnaike and Ray (1994), pp. 665-71, 1071. Seborg et al. (1989), pp. 278-82, 633-34. Shinskey (1994), pp. 102-137.	Bhat and McAvoy (1989) Hunt and Sbarbaso (1992) Nahas et al. (1991) Piovoso et al. (1992) Psichogios and Ungar (1991) Scott and Ray (1993a,b; 1994)
• Generic model control (GMC)	Ogunnaike and Ray (1994), pp. 671-74, 1018.	Piovoso et al. (1992) Ramchandran and Rhinehart (1994)
2. The optimization approach		
• Model-predictive control (MPC)	Luyben (1990), pp. 281-88, 606-09. Ogunnaike and Ray (1994), pp. 991-1014. Seborg et al. (1989), pp. 649-69. Shinskey (1994), pp. 101, 116-20.	Blum et al. (1992) Chen and Weigand (1994) Deshpande et al. (1995) Donat et al. (1991) Eaton et al. (1994) Graettinger et al. (1994a,b) Hernandez and Arkun (1990, 1992) Himmelblau (1991) Lee and Park (1992) Montague et al. (1994) Pottmann and Seborg (1992) Psichogios and Ungar (1991) Samad and Su (1994) Schenker and Agarwal (1993, 1994) Song and Park (1993) Su et al. (1992)
• Optimal control (OC)	Ray (1981), pp. 84-118.	Roat et al. (1990) Roat (1991)

2. Internal Model Control

Figures 5.49a-b compare the block diagrams of the classical feedback control and the internal model control (IMC). We follow Shinskey (1994) to briefly explain the features of this IMC structure, as well as the deficiency of the simplified neural-network-based IMC scheme presented previously in Figure 5.48. First, Figure 5.49b shows that the controller output **u** passes through the process G_p and its internal model G_m in parallel. The setpoint $\mathbf{y}_d$ is not compared with the measured output $\mathbf{y}_m$, but with the model error, $\mathbf{y}_m$-$\mathbf{y}_p$, which in fact represents a predicted value of the disturbance input, $\mathbf{d}_p$. The difference resulting from this comparison is therefore not the output error signal $\boldsymbol{\varepsilon}$ (= $\mathbf{y}_d$ - $\mathbf{y}_m$), which must be driven to zero, but the output error signal plus the model output, $\boldsymbol{\varepsilon}$ + $\mathbf{y}_p$ (= $\mathbf{y}_d$ - $\mathbf{y}_m$ + $\mathbf{y}_p$), which would only approach zero in the absence of disturbance input.

Secondly, the internal model controller in Figure5.49b combines a signal filter with a feedback controller. Let us illustrate the reason for this combination by considering the simplest practical case of a first-order process with deadtime:

$$G_p(s) = \frac{y(s)}{u(s)} = \frac{K_p e^{-\tau_d s}}{1 + \tau_1 s} \tag{5.21}$$

where K_p is the process steady-state gain, τ_d is the deadtime, τ_1 is its time constant, and s is the Laplace operator. The internal process model $G_m(s)$ would contain these *same* elements. Ideally, the feedback controller in the IMC structure would represent an inverse

model of the process, allowing us to determine the process inputs (controller outputs) necessary to produce the desired process outputs (Section 5.5.C.1). Unfortunately, inverting the process model, equation 5.21, is not attainable by physical components, since deadtime is not invertable (requiring a time advance), as discussed in Section 5.6.C.3, in conjunction with Figure 5.48. One way to overcome this problem is to specify the feedback controller $G_{c,fb}(s)$ as the following inverse form:

$$G_{c,fb}(s) = \frac{1}{G_m^*(s)} = \frac{1 + \tau_1 s}{K_p} \tag{5.22}$$

Here, the asterisk indicates that deadtime has been omitted from the internal process model $G_m(s)$, which has the same form as equation 5.22, i.e.,

$$G_m^*(s) = \frac{K_p}{1 + \tau_1 s} \tag{5.23}$$

or

$$\left[G_m^*(s)\right]^{-1} = \frac{1}{G_m^*(s)} = \frac{1 + \tau_1 s}{K_p} \tag{5.24}$$

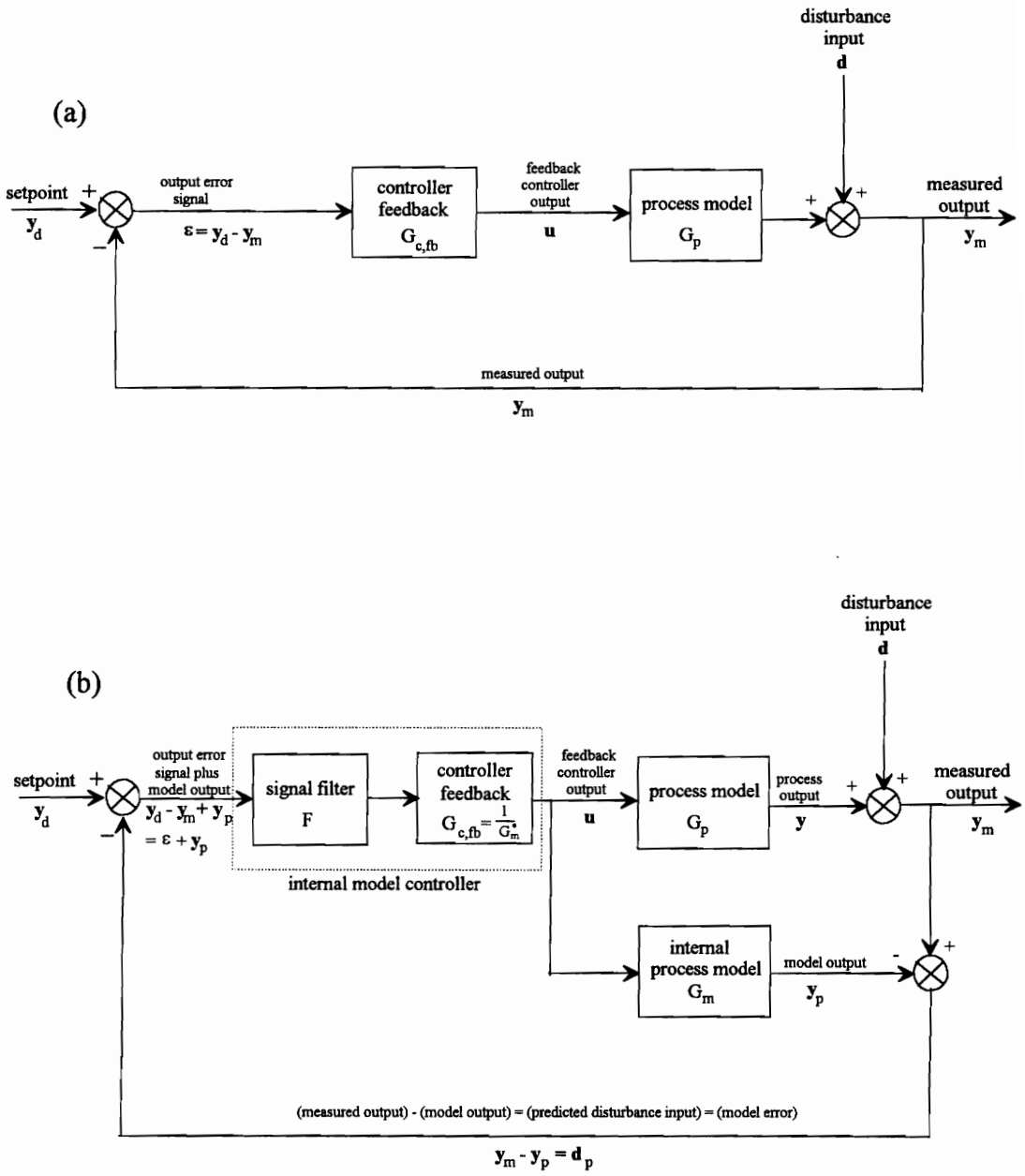


Figure 5.49. Block diagram of (a) the classical feedback control, and (b) the internal model control. Process model, G_p, incorporates the effects of the final control element and measurement dynamics, according to equation 5.15.

$\left[G_m^*(s)\right]^{-1}$ represents the transfer function for the feedback controller in the IMC structure for a process with $G_p(s)$ given by equation 5.21, and it corresponds to a phase lead type of transfer function. To implement this lead element, it is customary in process control to add a physical component with a phase-lag type of transfer function. A simple addition takes the role of a single filter with a transfer function (Figure 5.50):

$$F(s) = \frac{1}{1 + \tau_f s} \tag{5.25}$$

where τ_f is the filter time constant.

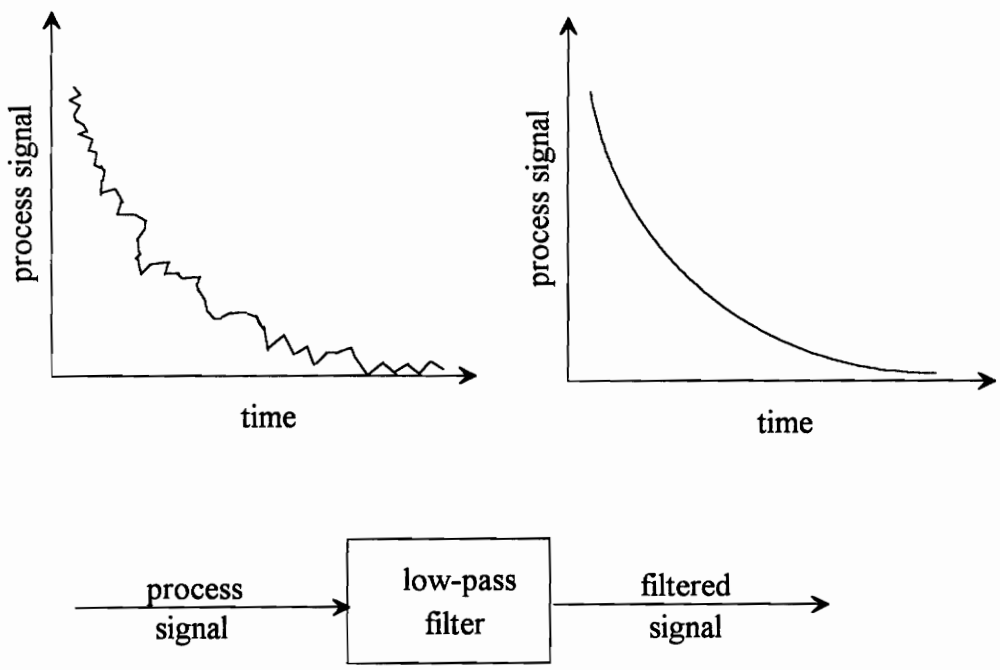


Figure 5.50. An illustration of a simple signal filter called a low-pass filter (Ogunnaike and Ray, 1994).

The internal model controller in the IMC structure of Figure 5.49b for a process with $G_p(s)$ given by equation 5.21 is then

$$F(s)\cdot G_{c,fb}(s) = F(s)\cdot\left[G_m^*(s)\right]^{-1} = \frac{1+\tau_1 s}{K_p(1+\tau_f s)} \tag{5.26}$$

In this example, we have retained the process parameters (i.e., process steady-state gain K_p and its time constant τ_1) in the feedback controller $\left[G_m^*(s)\right]^{-1}$, which is in fact the exact inverse of the process model. In practice, we try to develop an inverse process model by matching its parameters as closely as possible to the process. A typical example is the neural inverse model (controller) $G_{p,nn}^{-1}$ incorporated in the simplified IMC structure of Figure 5.48. In addition, we note in equation 5.25 that the filter time constant τ_f is the only tunable (adjustable) parameter in the internal model controller and it does not have a corresponding process component. This filter parameter is to be adjusted to increase the "robustness" of the controller. A robust controller is one that maintains satisfactory control despite changes in the dynamics of the process. We see that the lack of a filter represents a serious limitation in the simplified IMC structure of Figure 5.48

Despite the addition of a signal filter to improve the robustness of the controller, the IMC scheme of Figure 5.49b employing neural networks still has two deficiencies. Specifically, because the internal process model and the feedback controller use separate neural network models, the controller does not invert the steady-state gain of the model and offset cannot be eliminated. Moreover, this control scheme does not provide a tuning

parameter that can be adjusted to account for process-model mismatch (Hunt and Sbarbaro, 1991; Psichogios and Ungar, 1991).

In view of the preceding deficiencies, Nahas et al. (1992) suggest that a control component be incorporated to compensate for the effect of any time delays (lags) present in the process. They specifically apply the well-known Smith predictor for time-delay compensation (Ogunnaike and Ray, 1994, pp. 604-608; Seborg et al., 1989, pp. 419-424; Shinskey, 1994, pp. 113-116). In order to ensure an offset-free control performance, they also recommend modifying the approach to developing the forward and inverse neural models. In their approach, the predicted process output is based on the past model outputs, instead of past process outputs. For example, the neural forward model for the neutralization operation, equation 5.19, becomes

$$pH_{nn}(t+1) = f\left[F_2(t+1), F_2(t), F_2(t-1), pH_{nn}(t), pH_{nn}(t-1)\right] \tag{5.27}$$

Here, both $pH_{nn}(t)$ and $pH_{nn}(t-1)$ represent the predicted process outputs from the neural network model. Nahas et al. (1992) demonstrate that their nonlinear IMC scheme based on neural networks outperforms the conventional PID control for the neutralization operation and for a continuous stirred-tank reactor.

Further discussion of the use of neural networks in the IMC schemes is beyond the scope of this text. We conclude this discussion by noting that control practitioners, in general, tend to have a less favorable view of the IMC schemes when compared to a

properly-tuned PID controller. The interested reader may refer to Shinskey (1994, pp. 102-138), and Coughanower (1994, pp. 271-280).

3. Model-Predictive Control

As discussed previously, a thorough understanding of the model-predictive control (MPC) requires a good background in control theory and applied mathematics. Three texts are available that cover principally the subject of MPC: Prett and Garcia (1988), Morari and Zafiriou (1989), and Morari et al. (1995). Deshpande et al. (1995) have recently published a tutorial article on MPC.

In this section, we will only illustrate the basic idea of one MPC scheme based on neural networks, as applied to the neutralization operation of Section 5.6.C.2 (Donat et al., 1991), and introduce a commercial implementation, called NeuCOP, the Neural Control and Optimization Package, developed jointly by Texaco and NeuralWare, Inc. (Graettinger et al., 1994a,b).

A MPC scheme consists of *three operational steps: prediction, correction, and control-move determination*. The prediction step uses a model of the process to determine the future values of the process outputs. The correction step compares the measured value of the process output to the value predicted by the model, and corrects the predictions to account for the difference between two values. The control-move determination step performs the optimization, or more generally, the model inversion, to

calculate the future control moves needed to minimize or eliminate deviations of process output variables from reference (target) trajectories.

To better understand these operational steps, we consider the following *four basic elements* that are shared in common by *all* MPC schemes and are depicted in Figure 5.51 (Ogunnaike and Ray, 1994). We note that what differentiates one specific scheme from the other is the strategy and philosophy underlying how each element is actually implemented.

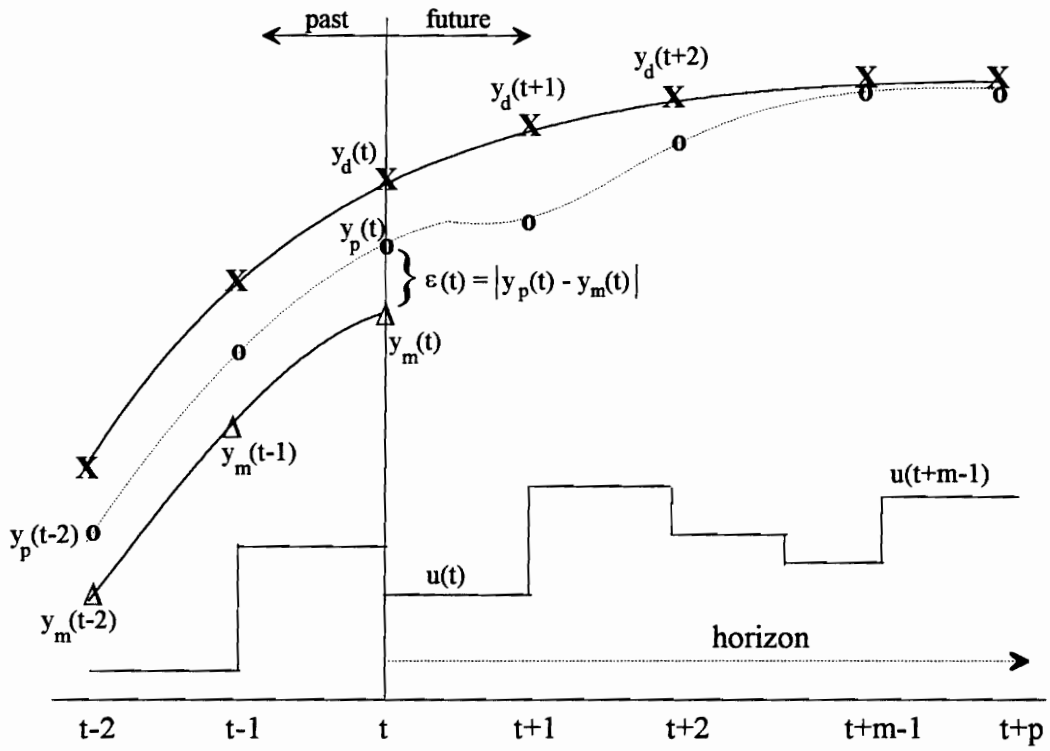


Figure 5.51. An illustration of the four basic elements of a model-predictive control scheme: (1) reference-trajectory specification - setpoint $y_d(t)$; (2) process-output prediction - predicted output $y_p(t)$; (3) process-output measurement - measured output $y_m(t)$; (4) error-prediction update - output error $\varepsilon(t) = | y_p(t) - y_m(t) |$ (Ogunnaike and Ray, 1994).

- *Reference trajectory specification*: This element refers to a desired reference (target) trajectory for the process output ($y_d(t)$, $y_d(t+1)$, $y_d(t+2)$, ...). It can simply be a step to a new setpoint value, or more commonly, a desired reference trajectory that is less abrupt than a step change. For the neutralization operation of Section 5.6.C.2, $y_d(t)$ corresponds to the pH setpoint, $pH_d(t)$.
- *Process output specification*: This element refers to the prediction of the process output over a predetermined, extended time horizon beginning with the current time as the prediction origin, by using an appropriate process model, *in the absence of further control action*. In Figure 5.51, this element means predicting $y_p(t+1)$, $y_p(t+2)$, ..., $y_p(t+i)$ for i sampling times ($t = i\Delta t$) into the future based on all actual past control actions u(t), u(t-1), ..., u(t-j) for j sampling times ($t = j\Delta t$) into the past. In MPC terminology, *the prediction horizon* refers to how far ahead we should predict process outputs, i.e., i sampling times into the future. For the neutralization operation, we may use a neural forward model, $G_{p,nn}$ of equation 5.19, to predict future pH values, $pH_{nn}(t+1)$, $pH_{nn}(t+2)$, ..., $pH_{nn}(t+i)$.
- *Control-action sequence determination*: This element uses the same process model to calculate a sequence of control moves, u(t), u(t+1), ..., u(t+m-1), that will minimize the predicted deviation of the process output from the reference (target) trajectory over the prediction horizon, or minimize some economic and operational objective function, subjected to prespecified operational constraints.

In MPC terminology, *the control horizon* refers to how far ahead we should try to control, i.e., (m-1) sampling times into the future. For the neutralization operation, the control-action sequence is $F_2(t)$, $F_2(t+1)$, ..., $F_2(t+m-1)$.

- *Error-prediction update*: This element refers to the deviation between the process measurement $y_m(t)$ and the model prediction $y_p(t)$, i.e., $\varepsilon(t) = y_m(t) - y_p(t)$ or $|y_m(t) - y_p(t)|$. For the neutralization operation, $\varepsilon(t) = |pH_m(t) - pH_{nn}(t)|$

Figure 5.52 illustrates a simple MPC scheme for the neutralization operation (Donat et al., 1991). We use the neural forward model of equation 5.19 as the process model.

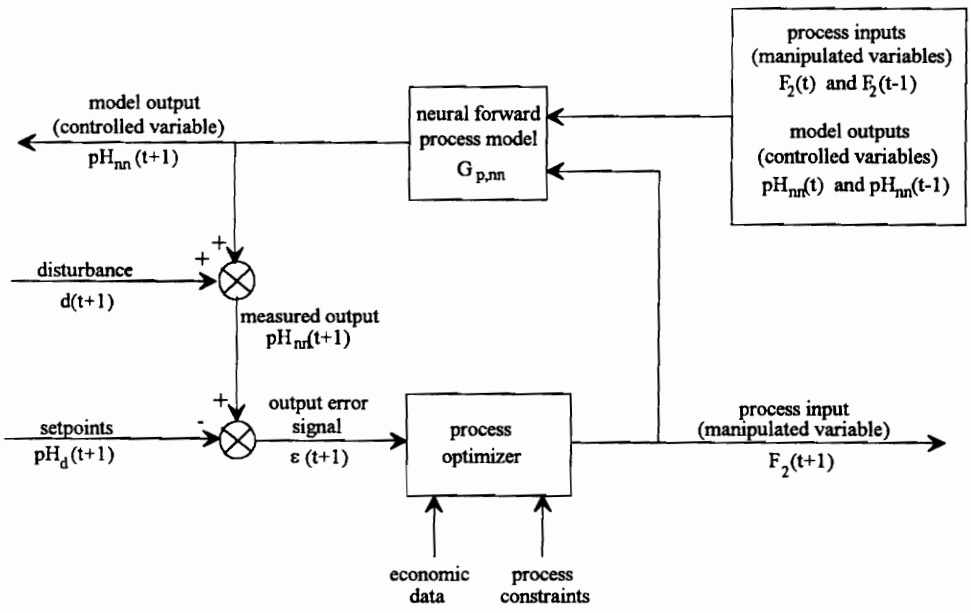


Figure 5.52. A model-predictive control scheme based on a neural forward process model and a process optimizer for the neutralization operation (Donat et al., 1991).

Our control objective is to determine the optimal NaOH flowrate F_2 as a function of time that will minimize the following square-error between the predicted pH values based on the neural process model, $pH_{nn}(t)$, and its setpoint value, $pH_d(t)$, over a prediction horizon of P (i.e., P sampling times into the future).

$$F = \sum_{k=1}^{P} \left[pH_{nn}(k) - pH_d(k) \right] \tag{5.28}$$

Here, pH(k) means the pH value at k sampling times, $k\Delta t$.

We assume the following operational constraints for the manipulated variable $F_2(t)$ and the controlled variable pH(t):

$$pH_{min} \leq pH \leq pH_{max} \tag{5.29}$$

$$F_{2,min} \leq F_2 \leq F_{2,max} \tag{5.30}$$

where min and max refer to the minimum and maximum allowable values, respectively.

We can efficiently solve the optimization (minimization) problem of equation 5.28 together with constraints, equations 5.29 and 5.30, by a so-called *feasible sequential quadratic programming (FSQP) code* developed by Professor A. L. Tits of the Department of Electrical Engineering, University of Maryland (Zhou and Tits, 1993). This optimization results in a sequence of future manipulated variable values, $F_2(t+1)$, $F_2(t+2)$, ... over the control horizon.

The basic ideas of the preceding MPC scheme based on neural network models have now been implemented commercially as NeuCOP, the Neural Control and Optimization Package, by Texaco and NeuralWare, Inc. (Graettinger et al., 1994a,b; NeuralWare, 1994). Figures 5.53a-b show the three components of NeuCOP, namely, Modeler, Optimizer, and Controller:

- *Modeler*: captures the unique nonlinear characteristics of each individual process unit in a "virtual process" simulator. Tune control loops, diagnose difficult operating conditions, search out new operating points - without disturbing the on-line process. Using historical data, create models of how the process actually operates, including seasonal and other disturbance variables.
- *Optimizer*: identifies economically optimum steady-state operating points and the best path to reach them. Explore the impact of changes in energy and raw material costs, and of the revenue from different product characteristics. Simultaneously maximize product quality, environmental objectives, and profit. Verify the dynamics of the NeuCOP Controller off-line. Integrate economics directly into the steady-state optimization process.
- *Controller*: combines economics with process dynamics to address the key control objectives of on-line economic optimization, that is, reduced process variability, and self-maintenance and tuning. Key features include: (1) advanced nonlinear multivariable neural modeling; (2) state-of-the-art model-predictive control; (3) real-time economic-based steady-state optimizer; (4) real-time

constraint-based path optimizer; (5) real-time automatic data acquisition; (6) self-maintaining and self-tuning automatic model builder; and (7) user-interface plots of historical and predicted process parameters including minute-by-minute unit profit.

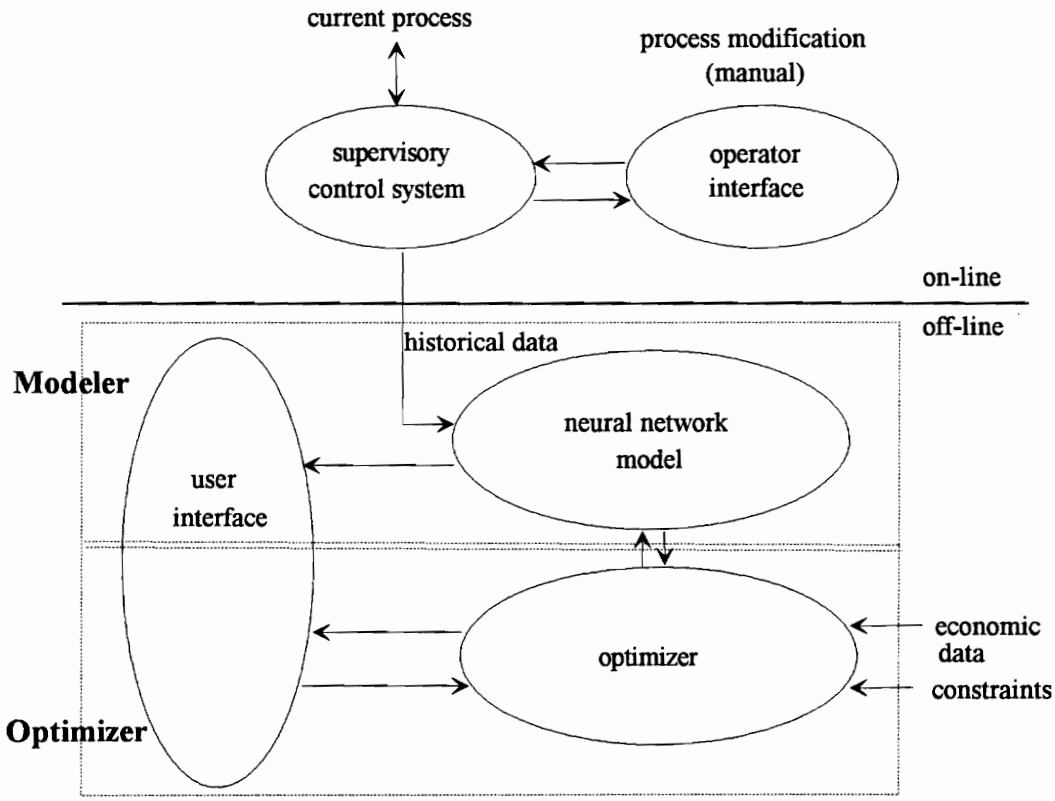


Figure 5.53a. An illustration of the components of NeuCOP, the Neural Control and Optimization Package: (a) NeuCOP Modeler and Optimizer; and (b) NeuCOP Controller (NeuralWare, 1994).

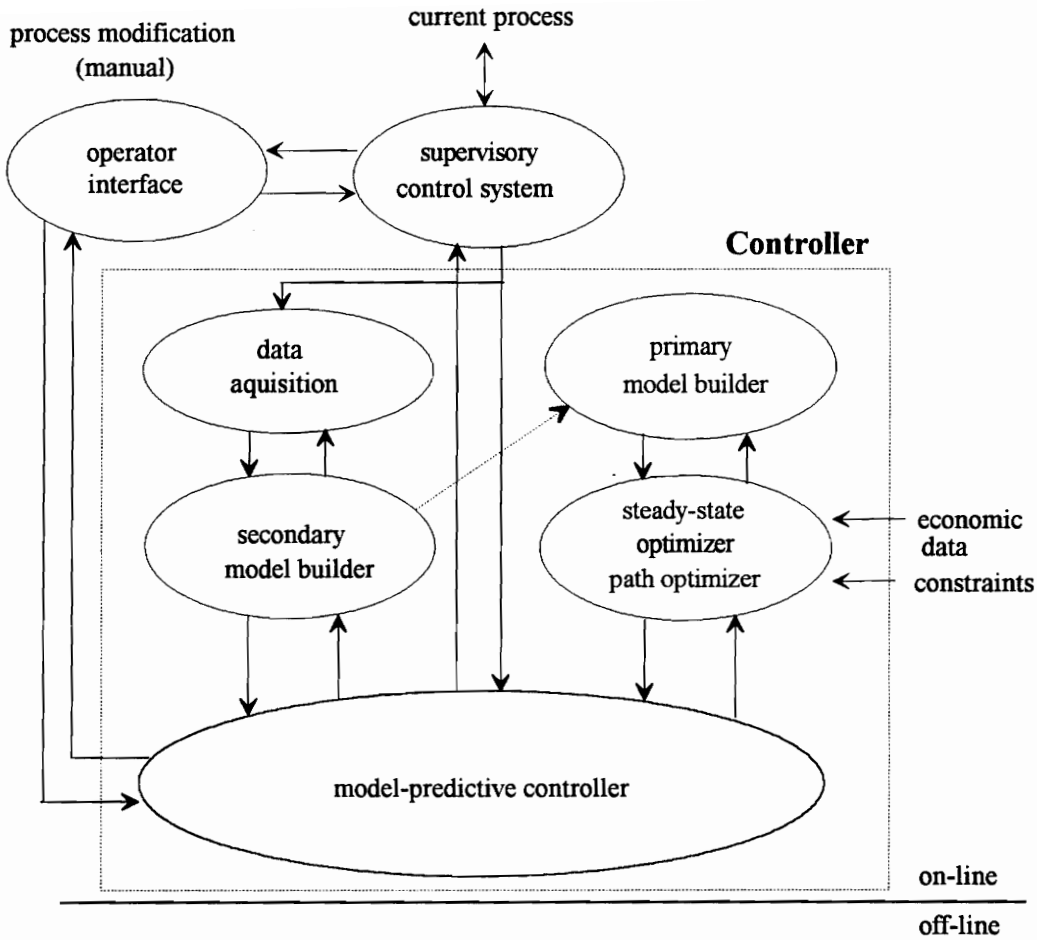


Figure 5.53b. An illustration of the components of NeuCOP, the Neural Control and Optimization Package: (a) NeuCOP Modeler and Optimizer; and (b) NeuCOP Controller (NeuralWare, 1994).

NeuCOP has been successfully applied to the predictive control and economic optimization of distillation columns at the Texaco Puget Plant, Texas (Graettinger et al., 1994a,b).

Finally, we conclude this section by noting that MPC schemes have been used in process control for over a decade. The most widely known MPC scheme, dynamic matrix control (DMC), uses as the process model a matrix of step-response coefficients relating process outputs to control actions. Other variations of MPC include, for example, quadratic dynamic matrix control (QDMC), a constrained version of DMC using quadratic programming; model-algorithmic control (MAC) that uses an impulse-response model of the process; linear dynamic matrix control (LDMC), also a constrained version of DMC solved with linear programming techniques. A detailed discussion of these MPC schemes and other extensions is beyond the scope of this text. We refer the reader to a broad coverage and an in-depth discussion of MPC in Chapter 27of the recent text by Ogannaike and Ray (1994), and to an industrial assessment of the strengths and limitations of MPC by Shinskey (1994, pp. 113-119).

E. Neural-Fuzzy Control

This section presents illustrative examples of three types of process control applications which have both neural network and fuzzy-logic components. The first application has a neural network predict the future responses of a time-dependent system, compute the difference between the response and a targeted value, and use this difference with other important variables in a fuzzy-logic control algorithm. The second application has the neural network determine a signal's temporal pattern type (e.g., increasing, decreasing,

constant, oscillating, etc.), and use a fuzzy-logic control algorithm to determine the appropriate process adjustments. The final application uses a neural-fuzzy network which has a fuzzy-logic system embedded in the neural network structure.

We note that all three illustrative examples represent the incorporation of fuzzy reasoning to the direct synthesis approach to model-based control discussed in Section 5.6.D.1. Specifically, the neural-fuzzy network models are used to determine the control action for the process output to follow the desired setpoint exactly. We also note that these fuzzy-neural systems are particularly effective in keeping the manipulated variables within the prespecified lower and upper limits because of the bounded nature of the membership functions.

1. Neural-Fuzzy Control of a Batch Fermentation Process Using Predictions of Future Operating Conditions

This section presents a neural-fuzzy control scheme to the batch fermentation process described in Section 5.4. The objective of this controller is to have the fermentation follow a prespecified cell-growth profile $c_{ref}(t)$ (designated by the designer) by adjusting the fermentation temperature (ΔT). The fermentation-processing network (Figure 5.21) predicts future cell concentrations for the operating conditions being held at the present level. A fuzzy controller is used to determine temperature modifications (ΔT's) required to achieve the desired cell concentration at a future time (e.g., t+4 or 2 hours).

Figure 5.54 shows the desired cell-growth profile of the fermentation, along with the actual cell concentrations c(t) at time 0,.., t, and the predicted cell concentrations $c_{nn}(t)$ at time t+1,.., t+4. The two concentration deviations are defined as $\Delta c(t) = c(t) - c_{ref}(t)$ and $\Delta c_{nn}(t+4) = c_{nn}(t+4) - c_{ref}(t+4)$.

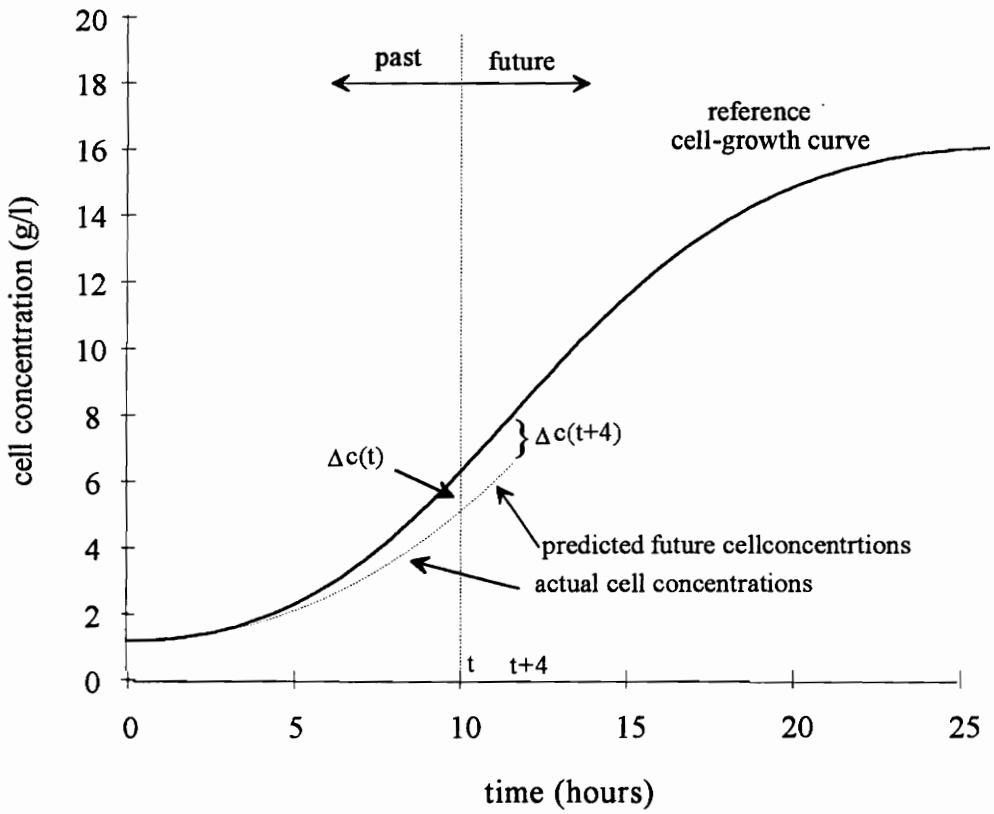


Figure 5.54. The desired cell-growth profile of the fermentation, along with the actual cell concentrations c(t) at time 0,.., t, and the predicted cell concentrations $c_{nn}(t)$ at time t+1,.., t+4.

The three variables used by the fuzzy-logic controller ($\Delta c(t)$, $\Delta c_{nn}(t+4)$, and ΔT) must be converted into fuzzy-logic variables (see Section 2.6.D.2). Figure 5.55 shows the standard form of the membership function used for conversion of these variables, where each variable is divided into five membership groups: NL - negative large, NM - negative middle, ZE - zero, PM - positive middle, and PL - positive large.

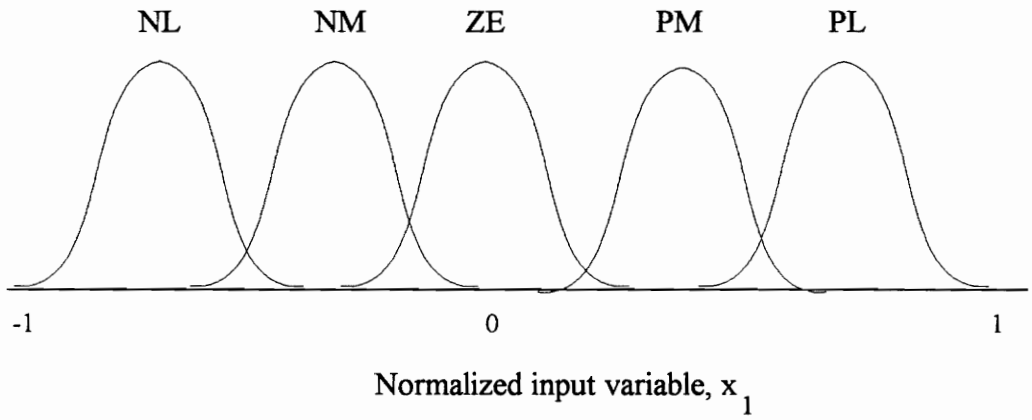


Figure 5.55. The standard form of the membership function used for the three variables ($\Delta c(t)$, $\Delta c_{nn}(t+4)$, and ΔT) which are divided into five membership groups: NL - negative large, NM - negative middle, ZE - zero, PM - positive middle, and PL - positive large.

The fuzzy-logic controller is based on a set of IF-THEN rules, which is typically presented in a matrix form. The columns and rows represent, respectively, the membership groups of the controlled variables (e.g., cell concentration), and the interior of the matrix (intersections of a row and a column) shows the resulting control action taken

by the manipulated variable (e.g., temperature) from a conditional AND statement. Table 5.28 lists the rules for determining $\Delta T(t)$ from $\Delta c(t)$ and $\Delta c_{nn}(t+4)$. For example, if $\Delta C(t)$ is "negative middle" and $\Delta c_{nn}(t+4)$ is "positive large," then T(t) is "positive large".

Table 5.28. The fuzzy rules for determining ΔT from $\Delta c(t)$ and $\Delta c(t+4)$. NL - negative large, NM - negative middle, NS - negative small, ZE - zero, PS - positive small, PM - positive middle, and PL - positive large.

$\Delta T(t)$		$\Delta c(t+4)$				
		NL	NM	ZE	PM	PL
$\Delta c(t)$	NL	PL	PL	PM	ZE	ZE
	NM	PL	PM	PM	ZE	ZE
	ZE	PM	PM	ZE	NM	NM
	PM	ZE	ZE	NM	NM	NL
	PL	NM	NM	NM	NL	NL

Figure 5.56 shows the overall architecture of the neural-fuzzy network for controlling the cell-growth profile in a batch fermentation process. This hybrid system incorporates the best features of both neural networks and fuzzy-logic system, while avoiding some of the limitations associated with each independently. The neural network is used as a predictor of future cell concentrations, but is not required to make controller adjustments which can often lead to extrapolation problems. The power of the fuzzy-logic

controller is also enhanced by adding future responses of the process into its set of rules.

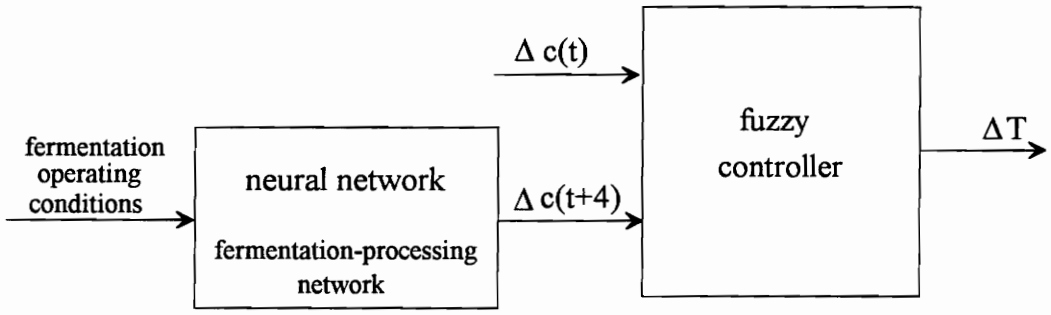


Figure 5.56. The architecture of a neural-fuzzy network for controlling the cell-growth profile in a batch fermentation process.

2. Neural-Fuzzy Control of a Bioreactor System Using Temporal Pattern Recognition

Shi and Shimizu (1992a,b) developed a neural-fuzzy algorithm to control the glucose concentration for the fed-batch fermentation of saccharomyces cerevisiae, an important host strain for recombinant DNA. The control of the glucose concentration is critical to this process because high glucose concentration leads to the production of ethanol, which in turn inhibits the production of the desired genetic material. Since online glucose sensors are seldomly available, Shi and Shimuzu (1992a,b) use the dissolved oxygen content and ethanol concentration to control the glucose concentration.

In this fermentation, the glucose will follow a certain set of trends based on the dissolved oxygen content and ethanol concentration:

- excess glucose causes ethanol concentration to increase.
- glucose depletion causes ethanol concentration to decrease.
- glucose depletion causes the substrate to decrease, which will lead to the oscillation of the dissolved oxygen content.

Therefore, the glucose concentration can be indirectly controlled by maintaining a constant ethanol concentration and keeping the dissolved oxygen content from oscillating. A fuzzy set is assigned for both these variables to determine their present status: (1) ethanol concentration ("increasing", "decreasing", or "nearly constant"); and (2) dissolved oxygen content ("oscillating" or "non-oscillating").

Figure 5.57 shows the overall schematic diagram of the neural-fuzzy controller described in this section (Shi and Shimizu, 1992a). In this diagram, there are two neural networks used to identify the temporal pattern type of the dissolved oxygen content and the ethanol concentration. The glucose concentration is controlled based on its nominal flowrate $F_{sp}(t)$ which is predetermined by a mass balance. Deviations in normal operating and environmental conditions will require a flowrate correction term, $\Delta F(t)$, to obtain the desired glucose concentration. This correction term is used to control the glucose concentration based on the ethanol concentration and dissolved oxygen content.

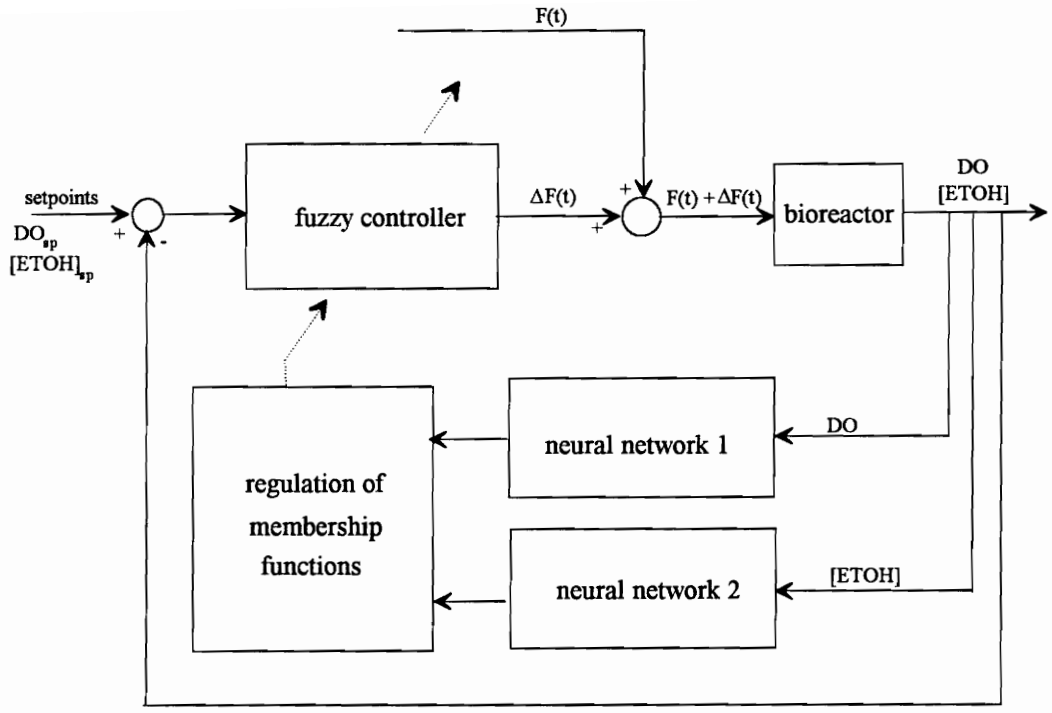


Figure 5.57. The architecture of the neural-fuzzy controller (Shi and Shimizu, 1992a). [ETOH] - ethanol concentration and DO - dissolved oxygen content.

a. Temporal-Pattern Recognition Neural Networks

Two temporal-pattern-recognition networks are employed to determine the proper category for ethanol concentration and dissolved oxygen content. To do this, we overlay the ethanol concentration and dissolved-oxygen-content signals on a two-dimensional (M*N grids) and assign a value of "1" to a quadrant were the signal is present and a "0" where it is not (Figure 5.58). Note that a signal partially passing through a block can be given a proportional value between 0 and 1. To input this signal into the network, we

assign each block (having a value ranging from 0 to 1) of the M*N temporal map to an input node of the neural network.

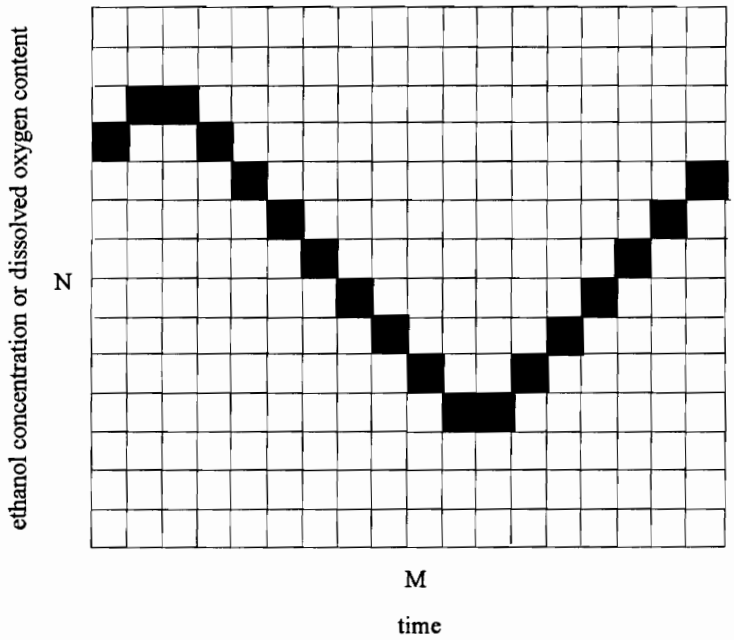


Figure 5.58. The representation of the ethanol concentration and dissolved oxygen content in a temporal map format.

Shi and Shimizu (1992a,b) trained the dissolved-oxygen-content network with 12 "oscillating" patterns and 30 "non-oscillating" patterns, and ethanol-concentration network with 100 "increasing", "decreasing", and "nearly constant" patterns. Variable random noise is added into all of these base signals, yielding a total of 1440 training patterns for dissolved oxygen content and 1500 patterns for ethanol concentration. The network achieved 98.7% prediction accuracy for dissolved oxygen content and 90.2% for ethanol concentration.

b. Fuzzy-Control Algorithm

In the development of a fuzzy-logic control algorithm to determine the flowrate correction term to achieve the desired glucose concentration, ΔF(t), we first define three membership functions for the value of dissolved oxygen content (S - small; M - medium; and B- big), the value of ethanol concentration (S - small; M - medium; and B- big), and ΔF(t) (NB - negative big; NS - negative small; ZE - zero; PS - positive small; PM - positive medium; and PB - positive big). See Figure 5.59 for a graphical representation of the membership functions. The ΔF membership function is bounded by limits of f_{min} and f_{max}.

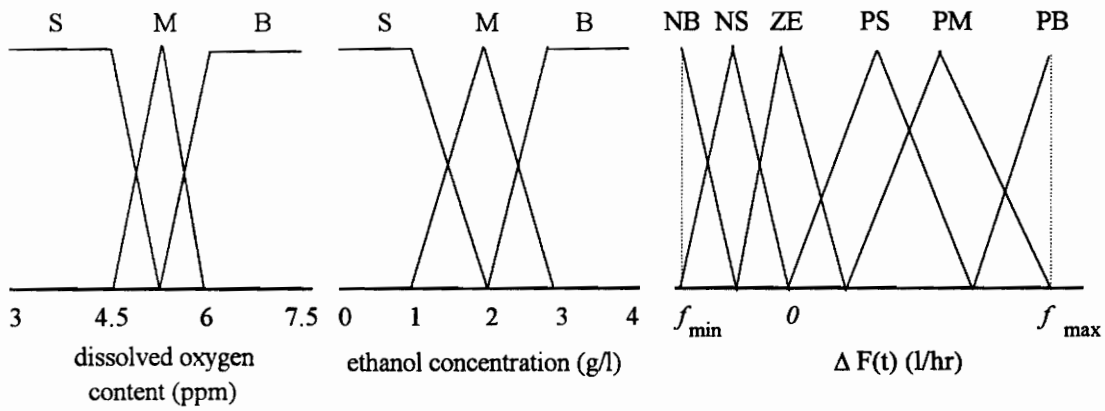


Figure 5.59. A graphical representation of the membership functions for dissolved oxygen content, ethanol concentration, and ΔF(t). S - small, M - medium, B- big, NB - negative big, NS - negative small, ZE - zero, PS - positive small, PM - positive medium, and PB - positive big.

Now that the control variables are in a fuzzy representation, the fuzzy rules are put in place. Table 5.29 lists the production rules for setting ΔF(t) as a function of the ethanol concentration and dissolved oxygen content. For example, a IF-THEN rule states that if ethanol concentration is "medium" and dissolved oxygen content is "low", then ΔF(t) is negative small.

Table 5.29. The fuzzy-logic production rules for setting ΔF(t).

ΔF(t)		dissolved oxygen content		
		S	M	B
ethanol concentration	S	ZE	PS	PB
	M	NS	ZE	PM
	B	NB	NS	PS

Shi and Shimizu (1992a) made online adjustments of f_{min} and f_{max} using the following equations:

$$f_{min}(k+1) = (1 + \delta a) f_{min}(k) \tag{5.31}$$

$$f_{max}(k+1) = (1 + \delta b) f_{max}(k) \tag{5.32}$$

where k is the sampling instant and δ is the step size (= 0.03). Table 5.30 lists the values of a and b (-1, 0, or 1) as a function of the ethanol signal pattern ("increasing", "decreasing", or "near constant") and dissolved oxygen content signal pattern ("oscillating" or "non-oscillating").

Table 5.30. The fuzzy-logic production rules for setting the constants a and b for calculating f_{min} and f_{max}.

		dissolved oxygen content signal type			
		oscillating		non-oscillating	
		a	b	a	b
ethanol concentration signal type	increasing	1	-1	-1	0
	near constant	1	0	0	0
	decreasing	1	1	0	0

Shi and Shimizu (1992a) found that this neural-fuzzy controller would suppress oscillation of the dissolved oxygen content, maintain low levels of ethanol produced, and increase cell productivity by about 15% over conventional fuzzy controllers.

3. Neural-Fuzzy Control of High-Cell-Density Cultivation of Recombinant Escherichia coli Using a Fuzzy Neural Network

Ye et al. (1994) developed a five-layer fuzzy neural network for the control of fed-batch cultivation of recombinant Escherichia coli JM103 harboring plasmid pUR 2921. The major objective of the fuzzy network is to significantly improve the expression of plasmid-encoded protein (β-galactosidase) by optimizing and controlling the culture conditions. The high expression of the plasmid-encoded protein and high cell density improves the process's productivity (final cell concentration).

Many advanced control strategies have been attempted to control the glucose concentration and the fed-batch cultivation of recombinant E. coli, but have had little success due to:

- lack of accurate mathematical models.
- complex interactions between cells and environment.
- lack of reliable online sensors.
- slow response of the process

To overcome these problems, Ye et al. (1994) apply a fuzzy network to control the glucose feeding rate using deviations in pH value and specific cell growth rate from their setpoints. The specific growth rate is the most important variable in this process, influencing both cell growth and the expression of plasmid-encoded protein. The pH of the culture broth is an indirect measure of the glucose concentration: (1) a decrease in the pH reflects an abundance of glucose which leads to the accumulation organic acids (e.g., acetic acid) during glycolysis; and (2) an increase in the pH reflects a lack of glucose caused by the production of NH_4^+. Therefore, pH increases require an increase of glucose flowrate and pH decreases require a reduction in glucose flowrate. The glucose feed rate is therefore constrained by lower and upper limits: (1) low glucose feed rates result in a decrease in specific growth rate from its targeted value; and (2) high glucose feed rates lead to the production of organic acids which inhibits of cell growth and leads to overexpression of the cloned gene product.

a. Structure of Fuzzy Neural Network

The primary objective in developing a fuzzy neural network is to use the neural network structure to process fuzzy reasoning. The fuzzy network identifies the fuzzy rules and tunes the membership functions by modifying weight factors.

Figure 5.60 shows a typical structure of a four-layer fuzzy network (Ye et al., 1994) that has two input variables. In this architecture, there is an input layer, three hidden layers, and one output layer. The three hidden layers act as a membership function in hidden layer 1, a measure of the rule strength in hidden layer 2, and a normalized measure of the rule strength in hidden layer 3.

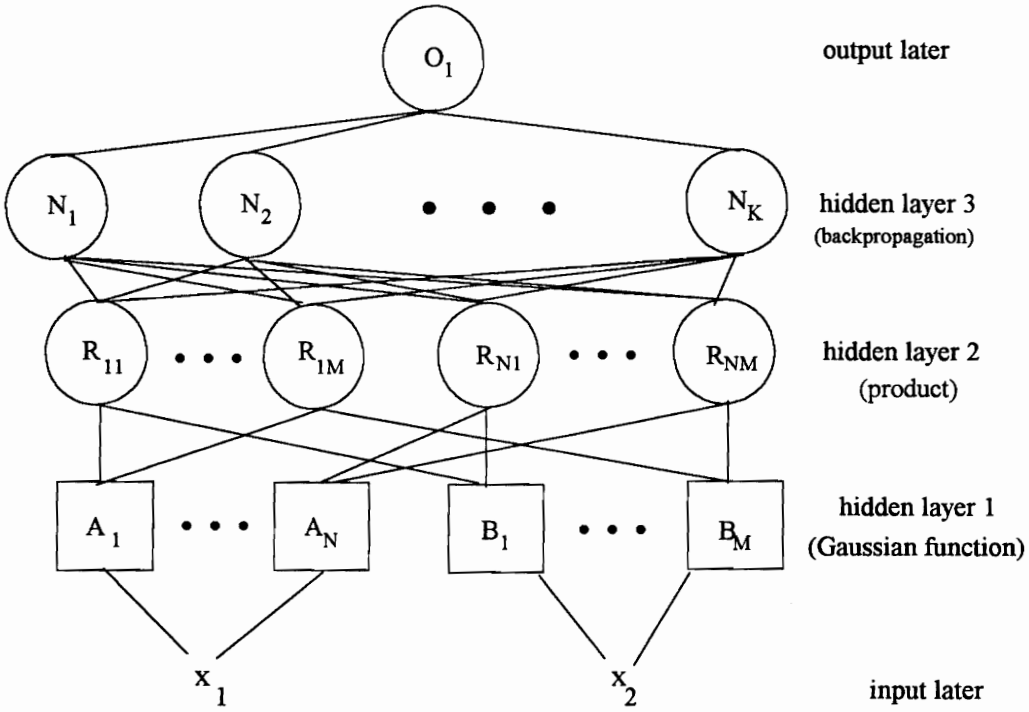


Figure 5.60. A typical structure of a four-layer fuzzy network (Ye et al. , 1994) that has two normalized input variables, x_1 and x_2.

Layer 1: represents a radially symmetric (Gaussian) membership functions of A_i (i = 1 to N) and B_j, (j = 1 to M) for the normalized input variables x_1 and x_2, respectively. Therefore, M and N are the numbers of fuzzy membership groups for their respective input variable. The outputs of the membership functions from layer 1, $O^1_{A_i}$ and $O^1_{B_j}$, are given by:

$$O^1_{A_i} = \exp\left[-\left(\frac{x_1 - \bar{x}_1}{\sigma_{ai}}\right)^2\right] \tag{5.33}$$

$$O^1_{B_j} = \exp\left[-\left(\frac{x_2 - \bar{x}_2}{\sigma_{bj}}\right)^2\right] \tag{5.34}$$

where $\bar{x}$ is the center and σ is the standard deviation of a Gaussian membership function. These two adjustable parameters can be modified to alter the shape and position of the Gaussian membership functions. This is similar to the principles of the radial-basis-function network (Section 3.2) that sets these parameters based on the K-mean clustering algorithm.

As an illustration, Figure 5.61 shows a common representation of the membership function stored in hidden layer 1. In this figure, there are five membership groups (N = 5) which correspond to nodes in hidden layer 1, A_1 to A_5 (NL - negative large, NM - negative middle, ZE - zero, PM - positive middle, and PL - positive large). The input variable x_1 is normalized using the zero-mean-normalization method (Section 2.3.B).

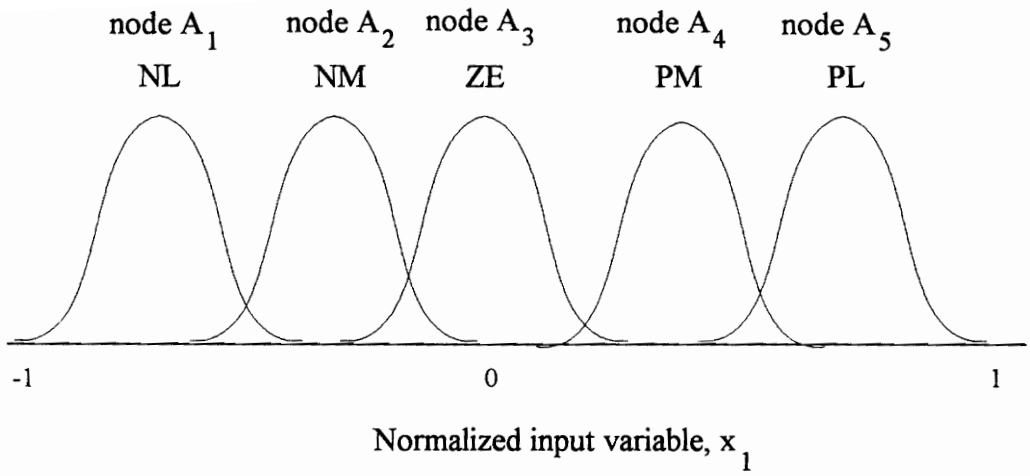


Figure 5.61. A common representation of the Gaussian membership function stored in hidden layer 1 of a fuzzy network.

Layer 2: gives a measure of the firing strength of a rule for the fuzzy-logic operation "AND," performed by taking the dot product of the outputs of the membership functions (hidden layer 1), $O^1_{A_i}$ and $O^1_{B_j}$.

$$O^2_l = O^1_{A_i} \cdot O^1_{B_j} \quad (l = 1 \text{ to } M*N) \tag{5.35}$$

For example, the output of the first node (R_{11}) in hidden layer reflects whether the rule of A_1 (x_1 is negative large) and B_1 (x_2 is negative large) should be applied.

Layer 3: operates the same as a standard backpropagation network, having weight factors (w_{ij}) and a sigmoid transfer function. The weight factors are adjusted to map the firing strength of the fuzzy rules onto the desired output variable.

Layer 4: operates the same as a standard backpropagation network, as in layer 3.

b. Implementation of a Fuzzy Network

Ye et al. (1994) developed their fuzzy network model for the addition of glucose using the following feeding rate equations:

$$\bar{F} = \frac{\mu_{sp} \cdot X \cdot}{Y_{x/s} \cdot S_f} \tag{5.36}$$

$$F = \bar{F} \cdot (1+\lambda) \tag{5.37}$$

where μ_{sp} is the specific growth rate (hr^{-1}), $\bar{F}$ is the feeding rate or volumetric flowrate of glucose (l/hr), S_f is the glucose concentration in the feed (g/l), V is the volume of the culture broth (l), X is biomass concentration in the culture broth (g/l), and $Y_{x/s}$ is the yield coefficient (g-cell/g-glucose), and F is the modified glucose feeding rate (g/l). Since the yield coefficient ($Y_{x/s}$) varies significantly with time for this process, a compensation factor (λ) is included to adjust the glucose flowrate.

The output of the fuzzy network is the compensation factor λ, which is used to adjust the glucose flowrate, according to equation 5.37. The two inputs to the network

are the deviations in pH (ΔpH) and specific cell growth rate ($\Delta\mu$) from their setpoints (pH_{sp} and μ_{sp}).

$$\Delta\mu(k) = \mu(k) - \mu_{sp}(k) \tag{5.38}$$

$$\Delta pH(k) = pH(k) - pH_{sp}(k) \tag{5.39}$$

where k is a time step. The setpoints are $pH_{sp} = 7.1$, and $\mu_{sp}(k) = 0.31\ hr^{-1}$ before the induction period and $\mu_{sp}(k) = 0.18\ hr^{-1}$ after the induction of the plasmid-encoded β-galactosidase using 3 mM of IPTG.

Figure 5.62 shows the fuzzy network structure for controlling the glucose flowrate (by determining the compensation factor for glucose feeding rate) from measured pH and specific growth rate.

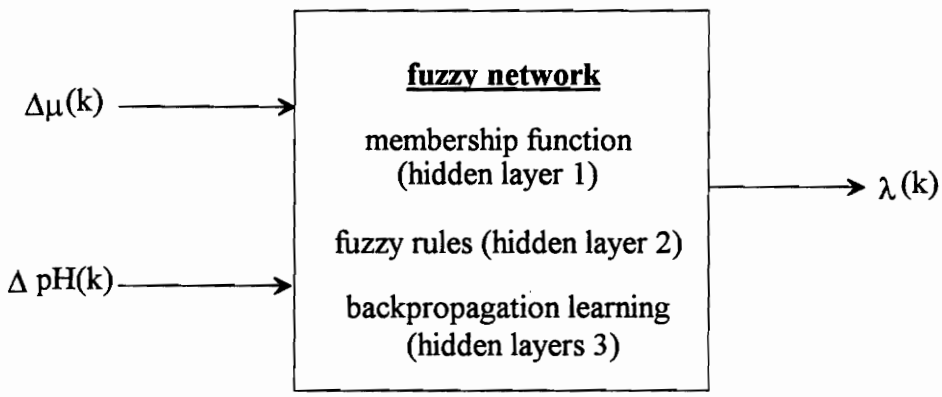


Figure 5.62. Fuzzy network structure for controlling the glucose flowrate from measured pH and specific growth rate.

Table 5.31 lists the rules for determining the compensation factor $\lambda(k)$ from $\Delta\mu(k)$ and $\Delta pH(k)$. For example, if $\Delta\mu(k)$ is "negative middle" and $\Delta pH(k)$ is "positive large," then $\lambda(k)$ is "positive large".

Table 5.31. The fuzzy rules for determining the compensation factor $\lambda(k)$ from $\Delta\mu(k)$ and $\Delta pH(k)$ (Ye et al., 1994). NL - negative large, NM - negative middle, NS - negative small, ZE - zero, PS - positive small, PM - positive middle, and PL - positive large.

$\lambda(k)$		$\Delta pH(k)$				
		NL	NM	ZE	PM	PL
$\Delta\mu(k)$	NL	NL	PS	PM	PL	PLL
	NM	NM	PSS	PS	PM	PL
	ZE	ZE	ZE	ZE	PS	PM
	PM	NS	ZE	ZE	ZE	PM
	PL	NL	NM	NS	ZE	PS

Ye et al. (1994) found that neural-fuzzy control improved the fed-batch cultivation of recombinant E. coli. from 30 DCW/l to 84 DCW/l, compared to the conventional exponential feeding of glucose. Although they significantly improved the final cell density, the relative activity of β-galactosidase was not high due to high residual glucose concentration after induction.

In order to overcome the low relative activity, they created two fuzzy networks, one for the period before induction and one for after induction. In training these two

networks, they found the membership functions (hidden layer 2) for ΔpH to change only slightly between the induction phases, but noticed significant changes for $\Delta\mu(k)$ between the phases. The addition of the second network for control after induction improved the sensitivity of $\Delta\mu$ fuzzy inferences, so that more cautious glucose feedrate control procedures were employed. The relative activity of β-galactosidase was increased four fold by adding the second fuzzy network, while cultivation of recombinant E. coli. was 50 DCW/l, representing an improvement over conventional techniques. Note that the setpoint $\Delta\mu_{sp}$ was lowered from 0.18 to 0.1 g/l in this case.

5.7 Chapter Summary

- Neural networks provide a method for developing dynamic models accounting for instabilities, nonlinear response characteristics and unsteady-state operating conditions that often occur in chemical and biological time-dependent systems.
- Data-compression networks reduce the amount of random noise in the input signal, while filtering gross errors in data resulting from sensor-malfunctions, drift, and bias. The neural network approach simplifies noise reduction by replacing the standard procedures of data rectification, gross-error detection, failure identification, and sensor-value replacement estimation with a single feedforward network.

The data-compression network is an autoassociative network composed of a input layer **In(m)**, a hidden layer **B(n)**, and a output layer **Out(m)**. The network maps

the noisy input vector of m elements onto a smaller intermediate vector, known as a bottleneck vector having n elements (where $m > n$). The compression ratio of the network is the ratio of elements in the input vector (m) to elements in the bottleneck vector (n).

- Recurrent neural networks for modeling time-dependent systems are composed of four sections: (1) the input layer that has both a manipulated variable vector **u**(t, t-1, t-2, ...) and a measured variable vector **y**(t, t-1, t-2, ...) for forecasting future measured variables; (2) the hidden layers that map the input layer on the output layer; (3) the output layer that contains the future predictions of measured variables, **y**(t+1, t+2, ...); and (4) recurrent loops that recycle network predictions to the input layer to make additional predictions.
- The first case study develops a recurrent network for process forecasting and modeling of a fermentation process, batch fermentation of saccharomyces cerevisiae and ethanol. The time-dependent network includes three types of networks: (1) an autoassociative backpropagation network for data compression and filtering that smoothes a continuous cell-concentration signal; (2) a radial-basis-function classification network that identifies the four phases of the fermentation process (induction, growth, stationary and death phases); and (3) a recurrent time-dependent network attached to the growth-phase output signal that forecasts future cell concentrations over a wide range of operating conditions, including temperature (25 to 35 ° C), pH (3.5 to 5.5), agitation rate (200 to

600 rpm), aeration rate (0 to 5.0 Nm^3/hour), and glucose concentration (0 to 100 grams/liter).

- The second case study develops a neural network to model the reactor of the Tennessee Eastman plantwide control problem. A feed-composition network is used with a supervisory controller to replace three outer-loop feed-composition cascade controllers, identifying the feed-flowrate setpoints to drive disturbed feed compositions to the desired feed composition levels. A reactor-pressure network is also developed to predict reactor pressure at time t+1 using the output of the feed-composition network.
- Direct network control uses a trained neural network as the controller and determines the controller output directly. A well-known successful commercial implementation of the direct network control is the Intelligent Arc Furnace (IAF) Controller.
- Inverse network control uses a trained neural network as an "inverse" model of a process, i.e., predicting the process inputs necessary to produce the desired process outputs (setpoints).
- Indirect network control uses a trained neural network to serve as a model of a process (as in the case of *model-based control*), or in determining the setpoints of a local controller (as in the case of *supervisory control*), thus indirectly affecting the process being controlled. Two important model-based control schemes using neural networks are *the internal model control (IMC)* and *the model-predictive control (MPC)*. In particular, Texaco, Inc. and others have reported the successful commercial implementation of neural networks for MPC.

- Neural-fuzzy control incorporates both the principles of neural networks and fuzz-logic control, and includes typically three main types of structures: (1) a neural network that predicts the future response of a time-dependent system, computes the difference between the response and a targeted value, and uses this difference with other important variables in a fuzzy-logic control algorithm; (2) a neural network that determines the temporal pattern type of a signal (e.g., increasing, decreasing, constant, oscillating, etc.), and uses a fuzzy-logic control algorithm to determine the appropriate process adjustments; and (3) a neural-fuzzy network that has a fuzzy-logic system embedded in the neural network structure is used to determine the controller adjustments.

Neural-fuzzy control appears to hold much promise in bioprocessing applications

Nomenclature

B(n)	:	n-dimensional bottleneck vector.
[C]	:	concentration of cells.
d	:	disturbance input.
[E]	:	concentration of ethanol.
$[E]^*$	:	limiting ethanol concentration at which no cells are present.
e(t)	:	error vector calculated from the difference between the predicted responses and the actual measured variables, $e_k(t) = \hat{y}_k(t) - y_k(t)$ (k = 1 to n).

e_{avg}	:	average error between predicted and measured variables.
$f(\,)$	:	transfer function for the processing element or functional relationship of the dynamic process model.
$G_{c,ff}$	:	feedforward controller.
$G_{c,fb}$	:	feedback controller.
$G_{c,nn}$	:	neural controller.
G_L	:	disturbance model.
G_L'	:	disturbance model.
G_m	:	measurement dynamics.
G_p	:	process model.
$G_{p,nn}$	:	a neural network process emulator.
G_p'	:	original process model.
G_v	:	final control element
[G]	:	concentration of glucose.
g[•]	:	functional relationship of the inverse process model.
f[•]	:	functional relationship of the process model.
In(m)	:	m-dimensional input-signal vector.
$\mathbf{In_e(m)}$	:	error component of the m-dimensional input-signal vector.
$\mathbf{In_n(m)}$	:	noise component of the m-dimensional input-signal vector.
$\mathbf{In_p(m)}$	:	noise-free/error-free (pure) component of the m-dimensional input-signal vector.

k	:	Monod reaction rate constant.
K	:	proportional gain in a controller.
K_M	:	Monod constant.
m	:	total manipulated input.
m_{fb}	:	feedback manipulated input.
m_{ff}	:	feedforward manipulated input.
m	:	number of incoming data points (size of input vector) entering a data-compression (autoassociative) network.
m/n	:	compression ratio of a data-compression (autoassociative) network having *m* nodes in the input and output layers and *n* nodes in the bottleneck (hidden) layer.
n	:	number of nodes in the bottleneck (hidden) layer in a data-compression (autoassociative) network.
n	:	constant in the poison-limited Monod model.
Out(m)	:	n-dimensional output-signal vector.
p_k	:	number of time intervals used in the process model for the measured variables at past time intervals (k = 1 to n).
r_c	:	production rate of yeast cells.
r_i	:	number of time intervals used in the process model for the manipulated variables (i = 1 to m).

s_j	:	number of time intervals used in the process model for the measured variables (j = 1 to n).
u	:	controller output.
u_{nn}	:	neural controller output.
$\mathbf{u}(t)$	:	m-dimensional vector of the manipulated variables $u_i(t)$ (i = 1 to m).
y	:	process output.
$\mathbf{y}(t)$	:	n-dimensional vector of the measured variables $y_i(t)$ (k = 1 to n).
y_d	:	desired output (setpoint).
y_m	:	measured output.
y_{nn}	:	neural network emulator output.
$\hat{\mathbf{y}}(t)$	:	n-dimensional vector of predicted responses of the measured variables, $\hat{y}_k(t)$ (k = 1 to n).
$\hat{y}_{k,adj}(t)$	:	adjusted prediction of measured varible $y_k(t)$.
ε	:	output error signal.
ε_{nn}	:	network error signal.
τ_i	:	integral time in a controller.
τ_{ui}	:	total time span for the input window for the manipulated variables.
τ_{yj}	:	total time span for the input window for the measured variables.
$\tau_{\hat{y}k}$	:	total time span for the output window for the predicted variables.

Practice Problems

(5.1) Create a data-compression network to reduce the noise in an input signal having 480 time steps (NeuralWare, 1993). Figure 5.P1a shows the pure (noise-free) input signal to be used in network training (file: *noise.nna*) and Figures 5.P1b-d shows the same signal with increasing quantities of noise added (files: *noisea.nna* - ±0.1 noise added; *noiseb.nna* - ±0.15 noise added; and *noisec.nna* - ±0.2 noise added).

(a) Using the pure input signal (file: *noise.nna*), train an autoassociative network (input vector equals the output vector) that has 20 nodes in the input and output layers and 10 nodes in the bottleneck (hidden) layer (compression ratio of 2:1). Filter the noise from the three noisy signals. Compare the filtered signals to the pure signal.

(b) Determine the optimal compression ratio, for a network with 20 nodes in the input layer, by generating a plot of RMS error versus compression ratio (see Section 2.6.A and Figure 2.35). For example, vary the number of nodes in the bottleneck layer from 4 (compression ratio of 5:1) to 15 (compression ratio of 4:3).

(c) Repeat part b for 5 and 10 nodes in the input and output layers. Compare the noise filtering capabilities of these networks with respect to the input vector size and compression ratio. .

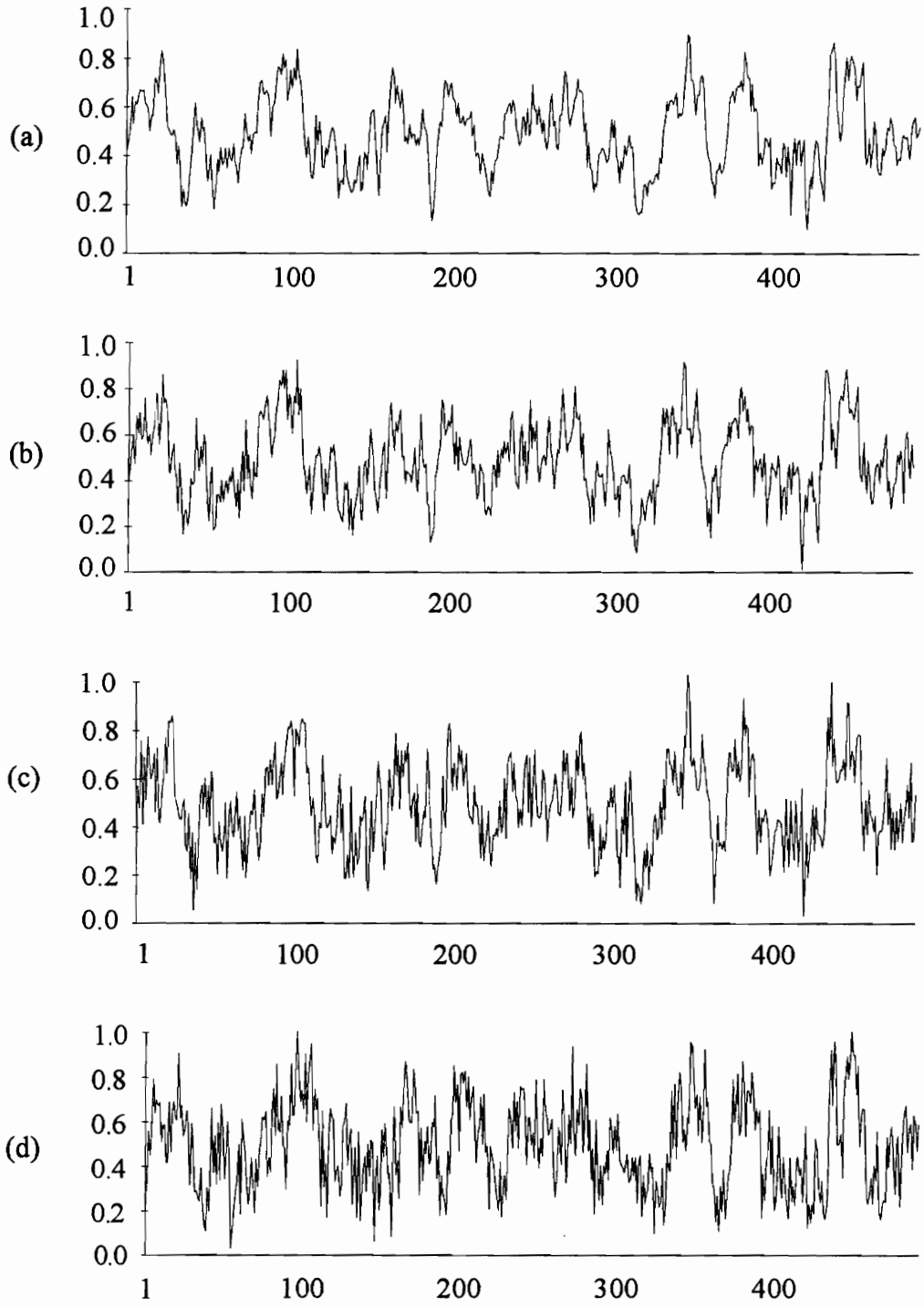


Figure 5.P1. (a) pure signal (file: *noise.nna*) used in training the data-compression network; (b) ±0.1 noise added (file: *noisea.nna*); and (c) ±0.15 noise added (file: *noiseb.nna*); (d) ±0.2 noise added (file: *noisec.nna*).

(5.2) Develop a recurrent network for predicting relative concentration versus time and temperature for pyrolysis of oil shale (Bates and Watts, 1988). Table 5.P1 lists the experimental data used for generating a data base. To obtain a continuous signal for network training, we smoothed the raw experimental data (Figure 5.P2) and extracted data (file: *oilshl.nna*) in one-minute time intervals. The training data file *oilshl.nna* has seven input and two output columns, arranged in the following order: (1) [bitumen](t-2); (2) [bitumen](t-1); (3) [bitumen](t); (4) [oil](t-3); (5) [oil](t-1); (6) [oil](t); (7) temperature; (8) [bitumen](t+1); and (9) [oil](t+1).

(a) Train and test a time-dependent recurrent network for several one- and two-hidden-layer configurations using file *oilshl.nna*. What is the optimal hidden-layer configuration for recall of training data only?

(b) For the optimal hidden-layer configuration, use the network to reproduce the six concentration-versus-time curves.

(c) To test for generalization, remove the data corresponding to temperature 723 K from the training database (file: *oiltrn.nna*) and train a new network. Present the excluded data (file: *oiltst.nna*) to the network and generate a concentration-versus-time curve for 723 K. How well does it compare to predictions made when the experimental run is included in the training set?

Table 5.P1. Experimental data used for generating a data base for predicting relative concentration versus time and temperature for pyrolysis of oil shale (Bates and Watts, 1988).

T = 673 K			T = 698 K		
time (min)	[bitumen]	[oil]	time (min)	[bitumen]	[oil]
5.0	0.0	0.0	5.0	6.5	0.0
7.0	2.2	0.0	7.0	14.4	1.4
10.0	11.5	0.7	10.0	18.0	10.8
15.0	13.7	7.2	12.5	16.5	14.4
20.0	15.1	11.5	15.0	29.5	21.6
25.0	17.3	15.8	17.5	23.7	30.2
30.0	17.3	20.9	20.0	36.7	33.1
40.0	20.1	26.6	25.0	27.3	40.3
50.0	20.1	32.4	30.0	16.5	47.5
60.0	22.3	38.1	40.0	7.2	55.4
80.0	20.9	43.2	50.0	3.6	56.8
100.0	11.5	49.6	60.0	2.2	59.7
120.0	6.5	51.8			
150.0	3.6	54.7			

T = 723 K			T = 748 K		
time (min)	[bitumen]	[oil]	time (min)	[bitumen]	[oil]
5.0	8.6	0.0	3.0	0.7	0.0
7.5	15.8	2.9	4.5	17.3	2.9
8.0	25.9	16.5	5.0	23.0	17.3
9.0	25.2	24.4	5.5	24.4	20.9
10.0	26.6	29.5	6.0	23.0	25.9
11.0	33.8	35.2	6.5	33.1	29.5
12.5	25.9	39.5	7.0	31.6	33.8
15.0	20.1	45.3	8.0	20.9	45.3
17.5	12.9	43.1	9.0	10.1	53.2
17.5	9.3	54.6	10.0	4.3	58.2
20.0	3.6	59.7	12.5	0.7	57.5
20.0	2.2	53.9	15.0	0.7	61.1

T = 773 K			T = 798 K		
time (min)	[bitumen]	[oil]	time (min)	[bitumen]	[oil]
3.0	6.5	0.0	3.0	25.2	20.9
4.0	24.4	23.0	3.25	33.1	25.2
4.5	26.6	32.4	3.5	21.6	17.3
5.0	25.9	37.4	4.0	20.9	36.7
5.5	17.3	45.3	5.0	4.3	56.8
6.0	21.6	45.3	7.0	0.0	61.8
6.5	1.4	57.5			
10.0	0.0	60.4			

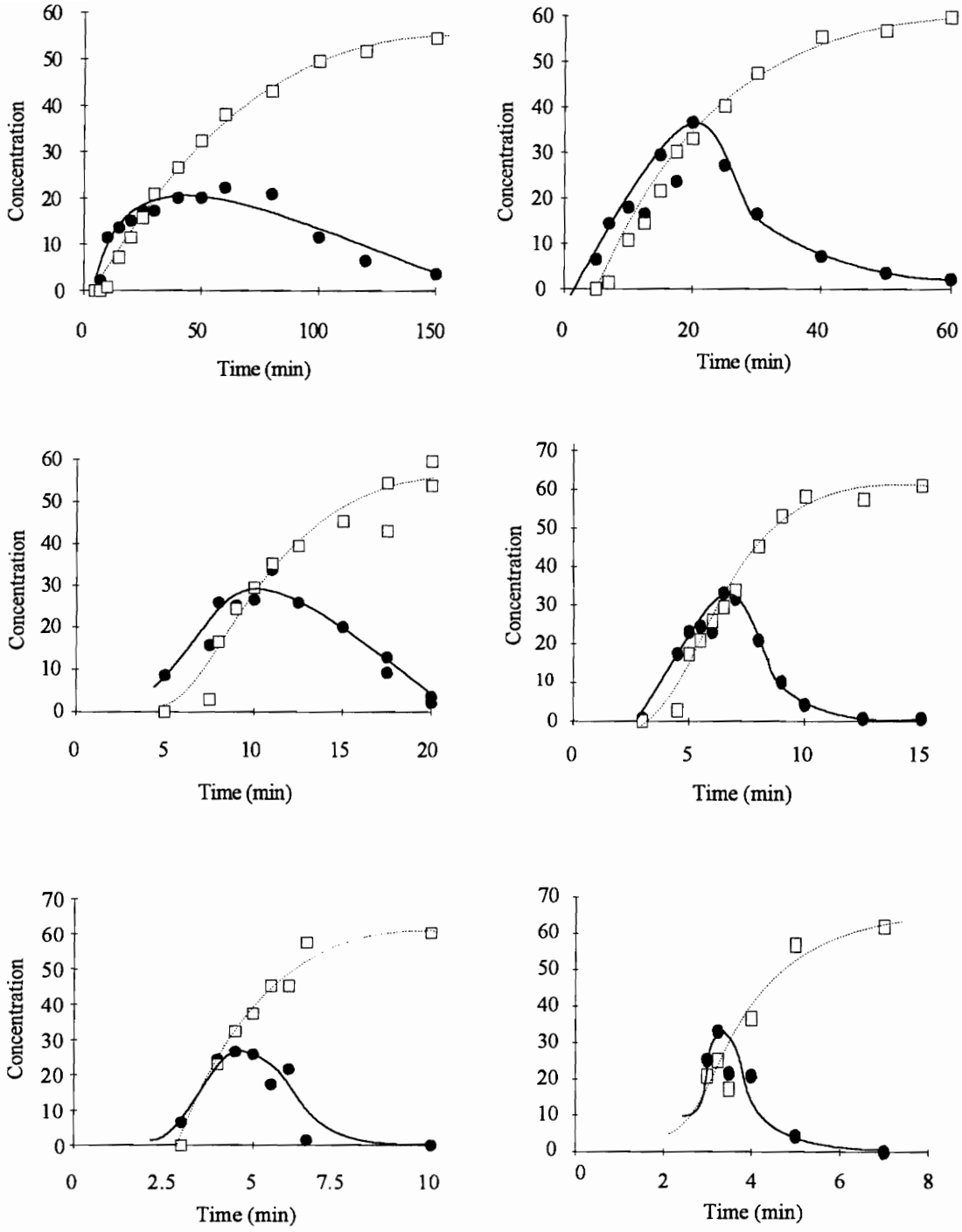


Figure 5.P2. The smoothed concentration-time data (file: *oilshl.nna*) for pyrolysis of oil shale, used to train the recurrent network. (•) - bitumen (□) - oil.

(5.3) Develop a backpropagation network to predict the reactor pressure (Tennessee Eastman plantwide control problem of Section 5.5) at the next time step from the six feed compositions (%A, %B, %C, %D, %E, and %F) and the four feed flowrates (A, D, E, and recycle) at the present time step. In other words, replace the two linked networks (Figure 5.38), namely, the feed-composition network (Figure 5.33) and reactor-pressure network (Figure 5.36), with a single network.

(a) Train and test the network using file *rctprsb.nna* for several one- and two-hidden-layer configurations, and identify the optimal configuration.

(b) Generate a scatter plot of predicted versus actual values, and compare the results to those in Figure 5.37.

(5.4) Develop a neural network to control a process using the *Adaptive Heuristic Critic* (AHC), introduced by Barto et. al. (1983). We adopt this problem from Quantrille and Liu (1991). The AHC is a three-layer *feedforward* neural network that uses supervised learning with reinforcement in a unique topology. The output from the first layer feeds into *both* the second and third layers, and is adjusted using weight factors v_{ij} and w_{jk}, respectively. Figure 5.P3 shows the AHC topology. Note that the number of nodes in layer 2 must equal the number in layer 3.

The learning procedure for AHC is a type of reinforcement learning called *reinforcement adaptation*, and differs from standard error-correction learning.

Each node *does not* receive an individual, explicit estimate of its error as done in backpropagation examples described in this text. Instead, in AHC, all components receive the same single error estimation, and corrections in weight factors follow from there. As seen in Figure 5.P3, the output from layer 2 goes directly into layer 3 with no weight factors. Thus, the nodes in layer 2 act as critics that provide a heuristically derived correction value to each node in layer 3.

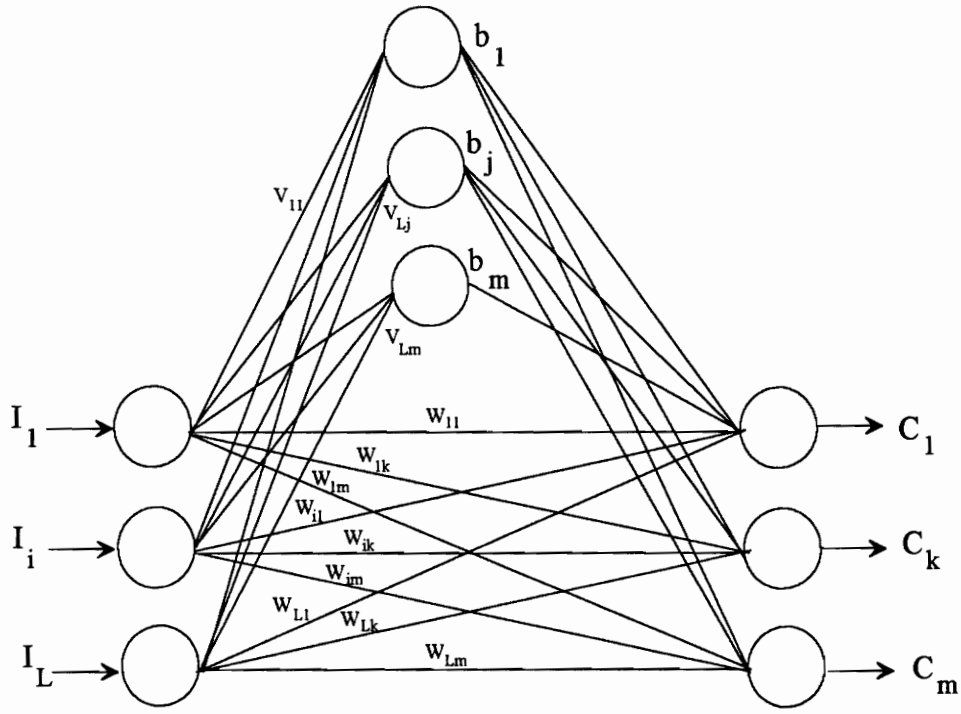


Figure 5.P3. Topology of the Adaptive Heuristic Critic (AHC). Reprinted with permission from *IEEE Trans. on Systems, Man, and Cybernetics*, SMC-13, "Neuron-Like Adaptive Elements That Can Solve Difficult Learning Control Problems," Barto, A., R. Sutton, and C. W. Anderson, copyright 1983, IEEE.

(a) If your neural network software package does not have the capability to make the connections of Figure 5.P3, write a computer program that follows the training and recall steps described below.

Training:

Step 1: Assign random values between [0, +1] to all v_{ij} and w_{ik} weight factors.

For m = 1, 2, ..., M input patterns, repeat steps 2-10 for each input pattern.

Step 2: Calculate the output from the first layer according to the equation:

$$a_i^m = I_i^m$$

where I_i^m is the input and a_i^m is the output, both for the ith node on the input layer for training session m. Note that the first layer has no thresholds or sigmoid functions; it simply passes the input directly into the network.

Step 3: Calculate the output from the third layer using the equation:

$$c_k(t) = f\left(\sum_{i=1}^{m} a_i^m w_{ik}(t) + \gamma(t)\right)$$

where $c_k(t)$ is the output from the kth node on the output layer at time t; $\gamma(t)$ is a random value in the interval [0,1]; and $f()$, instead of being a sigmoid function, is a bipolar step function, defined as:

$$f(x) = \begin{Bmatrix} +1 \text{ if } x > 0 \\ -1 \text{ if } x \leq 0 \end{Bmatrix}$$

Step 4: Calculate the *prediction*, $p_j(t)$, for the jth node on layer two according to the equation:

$$p_j(t) = \sum_{i=1}^{n} v_{ij}\, a_i^m$$

where v_{ij} is the weight factor between the ith node in layer one and the jth node in layer two.

Step 5: Calculate the *internal reinforcement*, $b_j(t)$, according to the equation:

$$b_j(t) = r(t) + \mu\, p_j(t) - p_j(t-1)$$

where μ is a positive constant ($0 < \mu < 1$) controlling the internal reinforcement, and $r(t)$ is the external reinforcement. The value of $r(t)$ depends on the neural network's performance. If a "fail" situation occurs, where the controlled variable falls outside of a desired range, then $r(t) = -1$. If the controlled variable is within desired range, then $r(t) = 0$.

Step 6: Calculate the *eligibility*, $e_{ik}(t)$, of the change in weight factor w_{ik} at time t, according to the equation:

$$e_{ik}(t+1) = \delta\, e_{ik}(t) + (1-\delta)\, a_i^m\, c_k^m$$

where δ is a positive constant ($0 < \delta < 1$) controlling the eligibility's decay rate.

Step 7: Adjust the w_{ik}'s, the weight factors for layers one to three, according to the equation:

$$\Delta w_{ik} = \alpha\, b_k(t)\, e_{ik}(t)$$

where α is a positive constant controlling the learning rate; Δw_{ik} is the change in the weight factor connecting the ith node in layer one and the kth node in layer three; $e_{ik}(t)$ is the *eligibility* of the change Δw_{ik} at time t; and $b_k(t)$, called the *internal reinforcement*, is the output from the kth node in layer two at time t.

Step 8: Calculate x_i, the weighted average of the ith node's value in layer one, by the equation:

$$x_i(t+1) = \varepsilon\, x_i(t) + (1 - \varepsilon)\, a_i^m$$

where ε is a positive constant controlling the decay rate ($0 < \varepsilon < 1$).

Step 9: Adjust the weight factors v_{ij} according to the equation:

$$\Delta v_{ij} = \beta\, b_j(t)\, x_i(t)$$

Step 10: Increment t and return to step 2; continue to iterate through steps 2-9 until no poor performers exist (i.e., r(t) = 1 for a sufficiently long period of time).

Recall:

The AHC provides constantly updated values of the outputs from the network. Thus, we can perform recall easily using the equation:

$$c_k(t) = f\left(\sum_{i=1}^{m} a_i^m w_{ik}(t) + \gamma(t)\right)$$

The continuous updating and easy recall enables on-line training of the AHC.

We will now use the AHC to control the temperature of water in a vessel. Note that the AHC uses an "on-off" type of control since the output is either +1 (turn the heat on) or -1 (turn the heat off). One benefit of the AHC is that we can start the controller up "naively"; the controller will take in process data on-line, and update its weights to properly control the process. *The system requires no prior tuning*; just plug it in and turn it on.

We use the AHC to control the temperature of the water exiting the tank shown in Figure 5.P4.

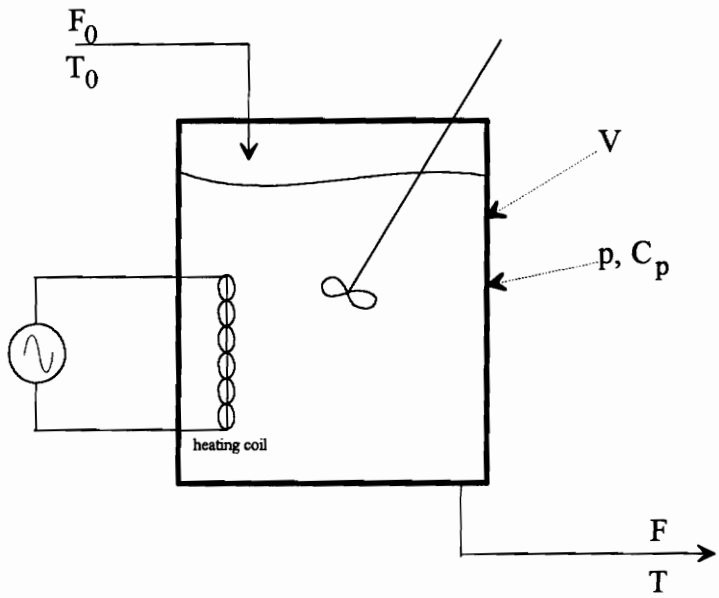


Figure 5.P4. On-off control of temperature in a tank.

The flow rate into the tank is fixed at 2 m^3/s, and does not change as a function of time. The temperature into the tank *does* change as a function of time, and falls from a maximum value of 35 °C to 15 °C in a pulsating fashion:

$$T(t) = 35 + 15 \sin\left(\frac{\pi t}{1000}\right) \qquad \text{if } \sin\left(\frac{\pi t}{1000}\right) < 0$$

$$T(t) = 35 \qquad \text{if } \sin\left(\frac{\pi t}{1000}\right) \geq 0$$

(b) Develop the energy balance for the tank, and determine dT/dt as a function of F (tank exit flowrate, m^3/s), V (tank volume, m^3), T_o (tank inlet temperature, °C), T (tank exit temperature, °C), Q (heating-coil duty, watts), ρ (density), and C_p (heat capacity). Develop the numerical equation for T(t), using Euler's method.

(c) Using the equations for T(t) and dT/dt as a model of the process, use the AHC developed in part (a) as a controller. The AHC has two inputs: T and dT/dt. It also has one output, c_1. If $c_1 = 1$, we turn the heat on. If $c_1 = -1$, we turn the heat off. To help, use the following values for the AHC parameters:

$\alpha = 1.1$ $\beta = 0.0005$ $\delta = 0.9$ $\eta = 0.95$ $\varepsilon = 0.99$

Recall that $\gamma(t)$ is a "noise" function that generates random values on the interval [0,1]. Make $\gamma(t)$ a Gaussian density function with a mean value of 0.5 and σ = 0.125. Start with a Δt of 0.5 second in the simulation, though you may find it valuable to experiment with the time increment. How well does the controller control the process? What do you suggest to improve performance? What does control theory suggest about on-off control for a first-order process?

Appendix 5.A.

Apparatus and Materials Used in the Fermentation Experiments

Apparatus:

Accu-Chek III Blood Glucose Monitor: Cat. No. 766. Boehringer Mannheim Corporation., 9115 Hague Rd., Indianapolis, IN 46250. Measures glucose concentration.

Bausch and Lomb Spectronic 20: Cat Nos. 33-31-7 and 33-31-72. Correlates cell concentration (off-line) to transmittance..

Biostat Fermentation Vessel: U.B.Braun Diessel Biotech GMBH. PO Box 120 d-3508 Melsungen, Federal Republic of Germany. BAE-U Rev. 0589. Model u50 fermenter. Volume - 75 liters.

Camile 3000 Systems Hardware Version 11.0: Dow Chemical Company. Used for data acquisition and control of fermentor.

Corning Ion Analyzer M250: Ciba Corning Diagnostics Limited Sudbury, Suffolk Cola 6XD, England. Corning Science Products Corning, New York 14831. Serial No. 4808. Measures pH in fermentor.

Max. Cell Mass Measuring System: Cerex Corp. 10078 Tyler Court, Ijamsville, Md. 21754. OMO32393. On-line measuring device for cell concentration.

Oxygen Probe: Ingold. Measures dissolved oxygen content.

Materials:

Antifoam B Silicone Emulsion: J.T. Baker Inc. Phillipsburg, NJ 08865.

D-(t)-Glucose: (Dextrose: Corn Sugar), Anhydrous. ACS Reagent [50-99-7] Molecular formula: $C_6H_{12}O_6$ Fw: 108.2. Specific Rotation: 53.0 at 25C. distributed by Sigma Chemical Company. St. Louis, Mo. 63178.

Deionized Water: 20 liters.

Dextrin : Distributed by Fisher Biotech of Fisher Scientific. Fair Lawn, NJ 07410.

Malt Extract: Distributed by Sigma Chemical Co., P.O. Box 14508. St. Louis, Mo. 63178.

D-(+)-Maltose: $C_{12}H_{22}O_{11} \cdot H_2O$ Distributed by Fisher Biotech of Fisher Scientific. Fair Lawn, NJ 07410.

Peptone: Bacteriological technical. Distributed by Sigma Chemical Company. St. Louis, Mo. Lot# 041176.

Yeast: YSC-2 Saccharomyces Cerevisiae (Baker's Yeast) Type II. Fast Dried to yield 90% active, viable 10% will autolyze in aqueous buffer at 37 C.

Appendix 5.B.

Tutsim Program Listing for the Tennessee Eastman plantwide control process (based upon program by P. R. Rony)

```
1 USR   74 ;          XMEAS(1) : A FEED
          P1=  1.0000
          P2=  1.0000
          P3=  0.2505200
2 USR   74 ;          XMEAS(2) : D FEED
          P1=  1.0000
          P2=  2.0000
          P3=  3.664E+03
3 USR   74 ;          XMEAS(3) : E FEED
          P1=  1.0000
          P2=  3.0000
          P3=  4.509E+03
4 USR   74 ;          XMEAS(4) : C FEED
          P1=  1.0000
          P2=  4.0000
          P3=  9.3477
5 USR   74 ;          XMEAS(5) : RECYCLE FLOW
          P1=  1.0000
          P2=  5.0000
          P3=  26.9020
6 USR   74 ;          XMEAS(6) : REACTOR FEED
          P1=  1.0000
          P2=  6.0000
          P3=  42.3390
7 USR   74 ;          XMEAS(7) : REACTOR PRESSURE
          P1=  1.0000
          P2=  7.0000
          P3=  2.705E+03
8 USR   74 ;          XMEAS(8) : REACTOR LEVEL
          P1=  1.0000
          P2=  8.0000
          P3=  75.0000
9 USR   74 ;          XMEAS(9) : REACTOR TEMP
          P1=  1.0000
          P2=  9.0000
          P3=  120.4000
10 USR  74 ;          XMEAS(10) : PURGE RATE
          P1=  1.0000
          P2=  10.0000
          P3=  0.3371200
11 USR  74 ;          XMEAS(11) : SEPARATOR TEMP
          P1=  1.0000
          P2=  11.0000
          P3=  80.1090
12 USR  74 ;          XMEAS(12) : SEPARATOR LEVEL
          P1=  1.0000
          P2=  12.0000
          P3=  50.0000
13 USR  74 ;          XMEAS(13) : SEPARATOR PRESSURE
          P1=  1.0000
          P2=  13.0000
          P3=  2.633E+03
14 USR  74 ;          XMEAS(14) : SEPARATOR UNDERFLOW
          P1=  1.0000
          P2=  14.0000
          P3=  25.1600
15 USR  74 ;          XMEAS(15) : STRIPPER LEVEL
          P1=  1.0000
          P2=  15.0000
          P3=  50.0000
16 USR  74 ;          XMEAS(16) : STRIPPER PRESSURE
          P1=  1.0000
          P2=  16.0000
          P3=  3.102E+03
17 USR  74 ;          XMEAS(17) : PRODUCT FLOW
          P1=  1.0000
          P2=  17.0000
          P3=  22.9490
18 USR  74 ;          XMEAS(18) : STRIPPER TEMP
          P1=  1.0000
          P2=  18.0000
          P3=  65.7310
19 USR  74 ;          XMEAS(19) : STRIPPER STEAM FLOW
          P1=  1.0000
          P2=  19.0000
          P3=  230.3100
20 USR  74 ;          XMEAS(20) : COMPRESSOR WORK
          P1=  1.0000
          P2=  20.0000
          P3=  341.4300
21 USR  74 ;          XMEAS(21) : REACTOR COOLING WATER
          P1=  1.0000
          P2=  21.0000
          P3=  94.5990
22 USR  74 ;          XMEAS(22) : SEPARATOR COOL WATER TEMP
          P1=  1.0000
          P2=  22.0000
          P3=  77.2970
23 USR  74 ;          A-COMP FEED
          P1=  1.0000
          P2=  23.0000
          P3=  32.1880
24 USR  74 ;          B-COMP FEED
          P1=  1.0000
          P2=  24.0000
          P3=  8.8933
25 USR  74 ;          C-COMP FEED
          P1=  1.0000
          P2=  25.0000
          P3=  26.3830
26 USR  74 ;  D-COMP FEED
          P1=  1.0000
          P2=  26.0000
          P3=  6.8820
27 USR  74 ;          E-COMP FEED
          P1=  1.0000
          P2=  27.0000
          P3=  18.7760
28 USR  74 ;          F-COMP FEED
          P1=  1.0000
          P2=  28.0000
          P3=  1.6567
29 USR  74 ;          A-COMP PURGE
          P1=  1.0000
          P2=  29.0000
          P3=  32.9580
30 USR  74 ;          B-COMP PURGE
          P1=  1.0000
          P2=  30.0000
          P3=  13.8230
31 USR  74 ;          C-COMP PURGE
          P1=  1.0000
          P2=  31.0000
          P3=  23.9780
32 USR  74 ;          D-COMP PURGE
          P1=  1.0000
          P2=  32.0000
          P3=  1.2565
33 USR  74 ;          E-COMP PURGE
          P1=  1.0000
          P2=  33.0000
          P3=  18.5790
34 USR  74 ;          F-COMP PURGE
          P1=  1.0000
          P2=  34.0000
          P3=  2.2633
35 USR  74 ;          G-COMP PURGE
          P1=  1.0000
          P2=  35.0000
          P3=  4.8436
36 USR  74 ;          H-COMP PURGE
          P1=  1.0000
          P2=  36.0000
          P3=  2.2986
```

Appendix 5.B.

Tutsim Program Listing for the Tennessee Eastman plantwide control process (based upon program by P. R. Rony)

```
37 USR   74 ;          D-COMP PRODUCT
            P1=  1.0000
            P2=  37.0000
            P3=  0.0178700
38 USR   74 ;          E-COMP PRODUCT
            P1=  1.0000
            P2=  38.0000
            P3=  0.8357000
39 USR   74 ;          F-COMP PRODUCT
            P1=  1.0000
            P2=  39.0000
            P3=  0.0985800
40 USR   74 ;          G-COMP PRODUCT
            P1=  1.0000
            P2=  40.0000
            P3=  53.7240
41 USR   74 ;          H-COMP PRODUCT
            P1=  1.0000
            P2=  41.0000
            P3=  43.8280
42 USR   87  101 ;     XMV(1) : D FEED VALVE
            P1=  2.0000
            P2=  1.0000
            P3=  63.0530
43 USR   88  102 ;     XMV(2) : E FEED VALVE
            P1=  2.0000
            P2=  2.0000
            P3=  53.9800
44 USR   89  103 ;     XMV(3) : A FEED VALVE
            P1=  2.0000
            P2=  3.0000
            P3=  24.6440
45 USR   90  104 ;     XMV(4) : C FEED VALVE
            P1=  2.0000
            P2=  4.0000
            P3=  61.3020
46 USR   91  105 ;     XMV(5) : RECYCLE VALVE
            P1=  2.0000
            P2=  5.0000
            P3=  22.2100
47 USR   92  106 ;     XMV(6) : PURGE VALVE
            P1=  2.0000
            P2=  6.0000
            P3=  40.0640
48 USR   93  107 ;     XMV(7) : SEPARATOR LIQUID FLOW -
            P1=  2.0000
            P2=  7.0000
            P3=  38.1000
49 USR   94  108 ;     XMV(8) : PRODUCT FLOW - STREAM 1
            P1=  2.0000
            P2=  8.0000
            P3=  46.5340
50 USR   95  109 ;     XMV(9) : STRIPPER STEAM VALVE
            P1=  2.0000
            P2=  9.0000
            P3=  47.4460
51 USR   96  110 ;     XMV(10) : REACTOR COOLING WATER
            P1=  2.0000
            P2=  10.0000
            P3=  41.1060
52 USR   97  111 ;     XMV(11) : CONDENSOR COOLING WATE
            P1=  2.0000
            P2=  11.0000
            P3=  18.1140
53 USR   98 ;          XMV(12) : AGITATOR SPEED
            P1=  2.0000
            P2=  12.0000
            P3=  50.0000
74 CON ;
            P1=  0.0000
87 PLS ;
            P1=  0.0000
            P2=  1.000E+06
            P3=  63.0530
88 PLS ;
            P1=  0.0000
            P2=  1.000E+06
            P3=  53.9800
89 PLS ;
            P1=  0.0000
            P2=  1.000E+06
            P3=  24.6440
90 PLS ;
            P1=  0.0000
            P2=  1.000E+06
            P3=  61.3020
91 PLS ;
            P1=  0.0000
            P2=  1.000E+06
            P3=  22.2100
92 PLS ;
            P1=  0.0000
            P2=  1.000E+06
            P3=  40.0640
93 PLS ;
            P1=  0.0000
            P2=  1.000E+06
            P3=  38.1000
94 PLS ;
            P1=  0.0000
            P2=  1.000E+06
            P3=  46.5340
95 PLS ;
            P1=  0.0000
            P2=  1.000E+06
            P3=  47.4460
96 PLS ;
            P1=  0.0000
            P2=  1.000E+06
            P3=  41.1060
97 PLS ;
            P1=  0.0000
            P2=  1.000E+06
            P3=  18.1140
98 PLS ;
            P1=  0.0000
            P2=  1.000E+06
            P3=  50.0000
101 PID  113  114   -2 ;   XMV(1) : D FEED FLOW
            P1=  0.0020000
            P2=  18.0000
            P3=  0.0000
            P4=  0.0000
            P5=  0.0000
            P6=  0.0000
102 PID  115  116   -3 ;   XMV(2) : E FEED FLOW
            P1=  0.0020000
            P2=  18.0000
            P3=  0.0000
            P4=  0.0000
            P5=  0.0000
            P6=  0.0000
103 PID  117  118   -1 ;   XMV(3) : A FEED FLOW
            P1=  200.0000
            P2=  6.0000
            P3=  0.0000
            P4=  0.0000
            P5=  0.0000
            P6=  0.0000
104 PID  119  120   -4 ;   XMV(4) : C FEED FLOW
            P1=  0.1000000
            P2=  18.0000
            P3=  0.0000
            P4=  0.0000
            P5=  0.0000
            P6=  0.0000
```

Appendix 5.B.

Tutsim Program Listing for the Tennessee Eastman plantwide control process (based upon program by P. R. Rony)

```
105 PID   121   122   -8 ;    XMV(5):- COMPRESSOR RECYCLE VALVE
                P1=   0.2000000
                P2=  200.0000
                P3=   0.0000
                P4=   0.0000
                P5=   0.0000
                P6=   0.0000
106 PID   123   124   -10 ;    XMV(6) : PURGE VALVE
                P1=  100.0000
                P2=   18.0000
                P3=   0.0000
                P4=   0.0000
                P5=   0.0000
                P6=   0.0000
107 PID   125   126   -12 ;    XMV(7) : SEPARATOR FLOW
                P1=  -0.5000000
                P2=  100.0000
                P3=   0.0000
                P4=   0.0000
                P5=   0.0000
                P6=   0.0000
108 PID   127   128   -15 ;    XMV(8) : STRIPPER FLOW
                P1=  -1.0000
                P2=  150.0000
                P3=   0.0000
                P4=   0.0000
                P5=   0.0000
                P6=   0.0000
109 PID   129   130   -18 ;    XMV(9) : STRIPPER TEMPERATURE
                P1=   20.0000
                P2=  200.0000
                P3=   0.0000
                P4=   0.0000
                P5=   0.0000
                P6=   0.0000
110 PID   131   132   -9 ;    XMV(10) : REACTOR COOL TEMP
                P1=  -20.0000
                P2=  200.0000
                P3=   0.0000
                P4=   0.0000
                P5=   0.0000
                P6=   0.0000
111 PID   133   134   -22 ;    XMV(11) : COND COOL WATER
                P1=  -36.6000
                P2=   60.0000
                P3=   0.0000
                P4=   0.0000
                P5=   0.0000
                P6=   0.0000
113 ZAC   158 ;              D FEED - SETPOINT
                P1=   1.0000
                P2=   1.0000
                P3=   3.664E+03
114 PLS ;
                P1=   0.0000
                P2=   1.000E+06
                P3=   0.0000
115 ZAC   159 ;              E FEED - SETPOINT
                P1=   1.0000
                P2=   1.0000
                P3=   4.509E+03
116 PLS
                P1=   0.0000
                P2=   1.000E+06
                P3=   0.0000
117 ZAC   161 ;              A-FEED - SETPOINT
                P1=   1.0000
                P2=   1.0000
                P3=   0.2505200
118 PLS ;
                P1=   0.0000
                P2=   1.000E+06
                P3=   0.0000
119 CON ;                    C DEED - SETPOINT
                P1=   9.3470
120 PLS ;
                P1=   0.0000
                P2=   1.000E+06
                P3=   0.0000
121 CON ;                    REACTOR LEVEL -  SETPOINT
                P1=  75.0000
122 PLS ;
                P1=   0.0000
                P2=   1.000E+06
                P3=   0.0000
123 ZAC   157 ;              PURGE RATE - SETPOINT
                P1=   1.0000
                P2=   1.0000
                P3=   0.3371200
124 PLS ;
                P1=   0.0000
                P2=   1.000E+06
                P3=   0.0000
125 CON ;                    SEPER LEVEL - SETPOINT
                P1=  50.0000
126 PLS ;
                P1=   0.0000
                P2=   1.000E+06
                P3=   0.0000
127 CON ;                    STRIPPER LEVEL - SETPOINT
                P1=  50.0000
128 PLS ;
                P1=   0.0000
                P2=   1.000E+06
                P3=   0.0000
129 CON ;                    STRIPPER TEMP - SETPOINT
                P1=  65.7310

130 PLS ;
                P1=   0.0000
                P2=   1.000E+06
                P3=   0.0000
131 CON ;                    REACTOR TEMP - SETPOINT
                P1=  120.4000
132 PLS ;
                P1=   0.0000
                P2=   1.000E+06
                P3=   0.0000
133 CON ;                    COND COOL TEMP - SETPOINT
                P1=  77.2970
134 PLS
                P1=   0.0000
                P2=  999.999E+03
                P3=   0.0000
138 PID   139   140   -26 ;    D-FEED COMP CONTROLLER
                P1=   8.0000
                P2=  999.999E+03
                P3=   0.0000
                P4=   0.0000
                P5=   0.0000
                P6=   0.0000
139 CON ;                    D-COMP FEED - SETPOINT
                P1=   6.8820
140 PLS
                P1=   0.0000
                P2=  999.980E+03
                P3=   0.0000
141 PID   142   143   -27 ;    E-FEED COMP CONTROLLER
                P1=   1.5000
                P2=  999.999E+03
                P3=   0.0000
                P4=   0.0000
                P5=   0.0000
                P6=   0.0000
142 CON ;                    E-FEED COMP - SETPOINT
                P1=  18.7760
143 PLS
                P1=   0.0000
                P2=  999.980E+03
                P3=   0.0000
```

Appendix 5.B.

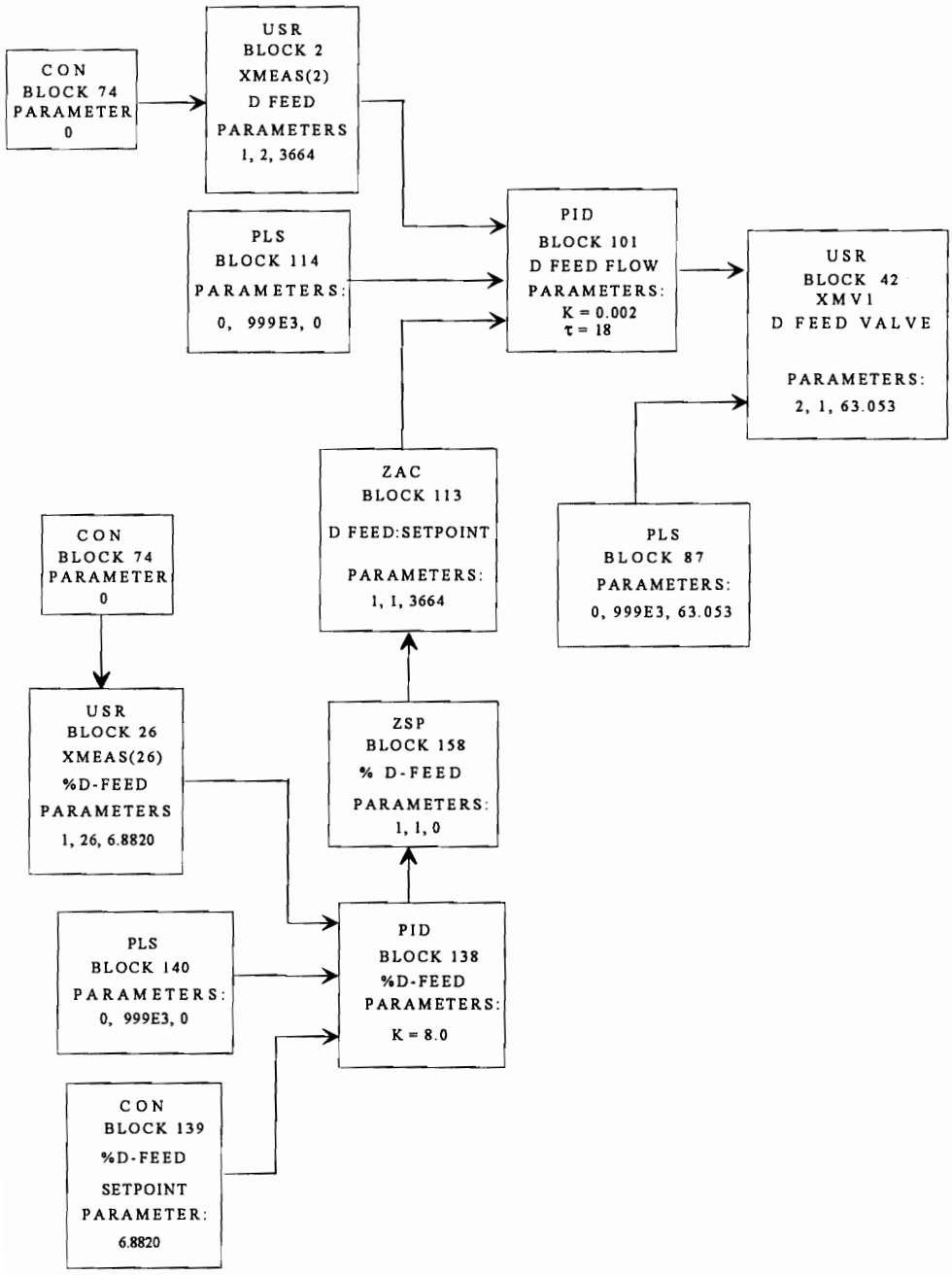
Tutsim Program Listing for the Tennessee Eastman plantwide control process (based upon program by P. R. Rony)

```
144 PID   145   146   -23 ;   A-COMP FEED CONTROLLER
                P1=   0.0010000
                P2=  999.999E+03
                P3=   0.0000
                P4=   0.0000
                P5=   0.0000
                P6=   0.0000
145 CON ;                   A-FEED COMP - SETPOINT
                P1=  32.1880
146 PLS
                P1=   0.0000
                P2=  999.980E+03
                P3=   0.0000
156 ZDT
                P1=   1.0000
                P2=  360.0000
                P3=   0.0000
157 ZSP   168 ;             B-COMP PURGE
                P1=   1.0000
                P2=   1.0000
                P3=   0.0000
158 ZSP   138 ;             D-COMP FEED
                P1=   1.0000
                P2=   1.0000
                P3=   0.0000
159 ZSP   141 ;             E-COMP FEED
                P1=   1.0000
                P2=   1.0000
                P3=   0.0000

161 ZSP   144 ;             A-COMP FEED
                P1=   1.0000
                P2=   1.0000
                P3=   0.0000
168 PID   169   170   -30 ;   B-COMP PURGE CONTROLLER
                P1=  -0.0050000
                P2=  999.999E+03
                P3=   0.0000
                P4=   0.0000
                P5=   0.0000
                P6=   0.0000
169 CON ;                   B-COMP PURGE - SETPOINT
                P1=  13.8230
170 PLS
                P1=   0.0000
                P2=  999.000E+03
                P3=   0.0000
```

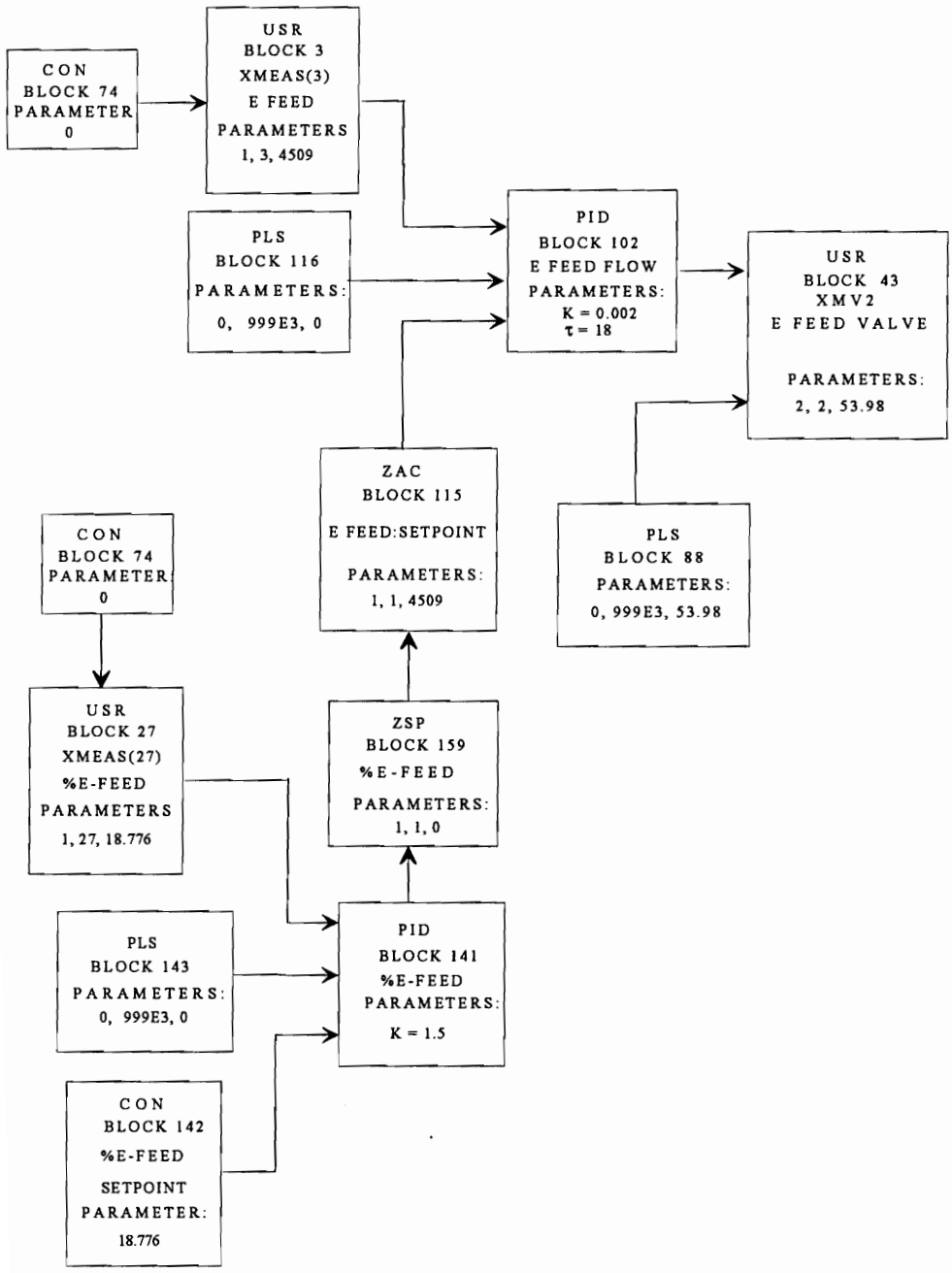
Appendix 5.C1.

Tutsim schematic diagram for the control of the D-feed valve (xmv1) for the Tennessee Eastman plantwide control process (IC2 and OC2 controllers).



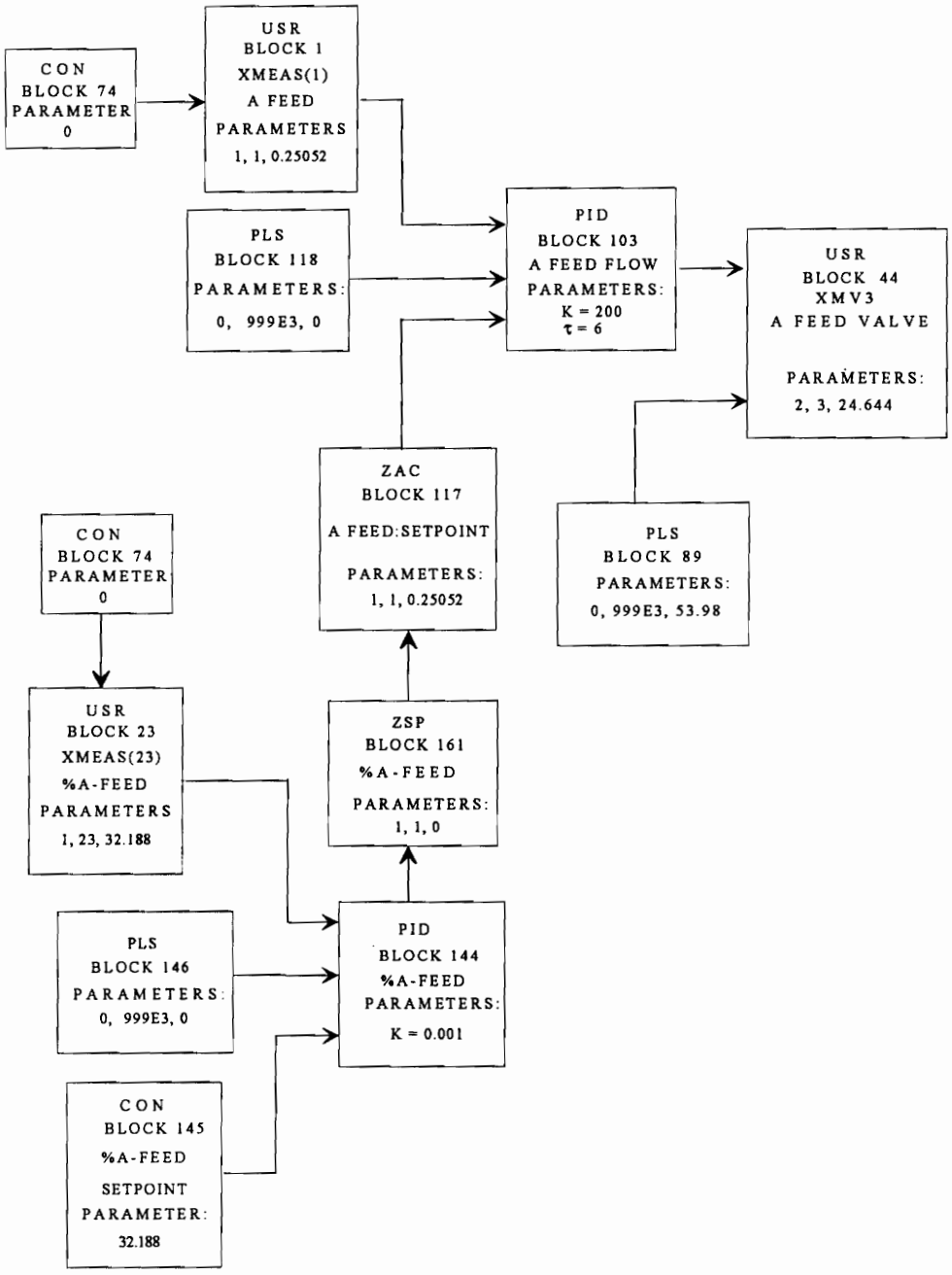
Appendix 5.C2.

Tutsim schematic diagram for the control of the E-feed valve (xmv2) for the Tennessee Eastman plantwide control process (IC3 and OC3 controllers).



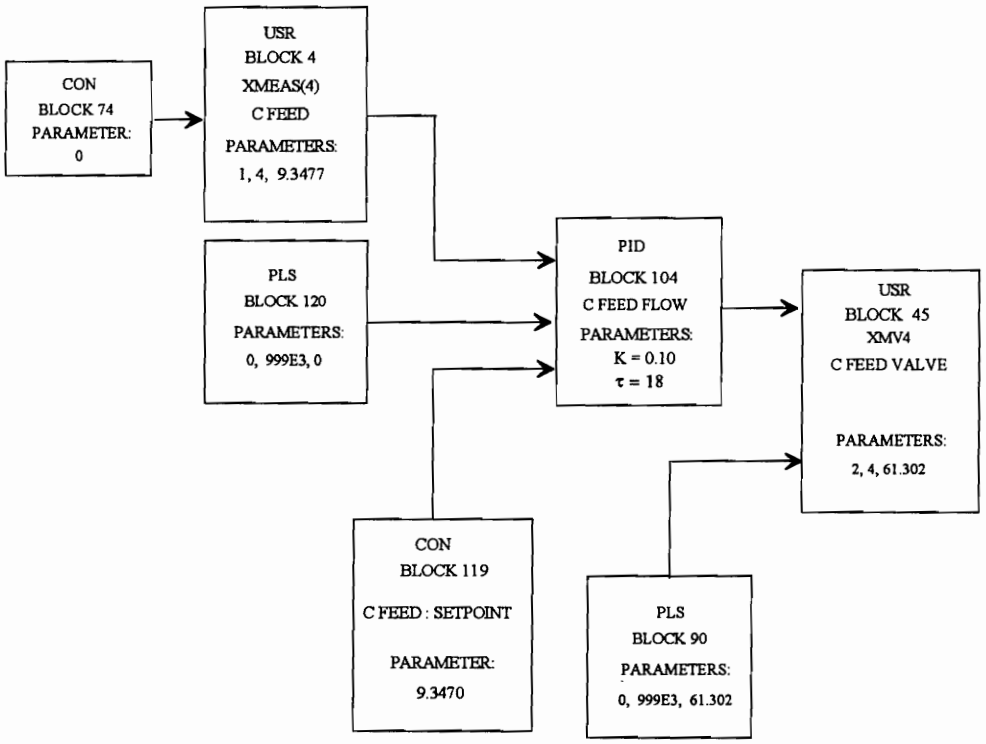
Appendix 5.C3.

Tutsim schematic diagram for the control of the A-feed valve (xmv3) for the Tennessee Eastman plantwide control process (IC1 and OC1 controllers).



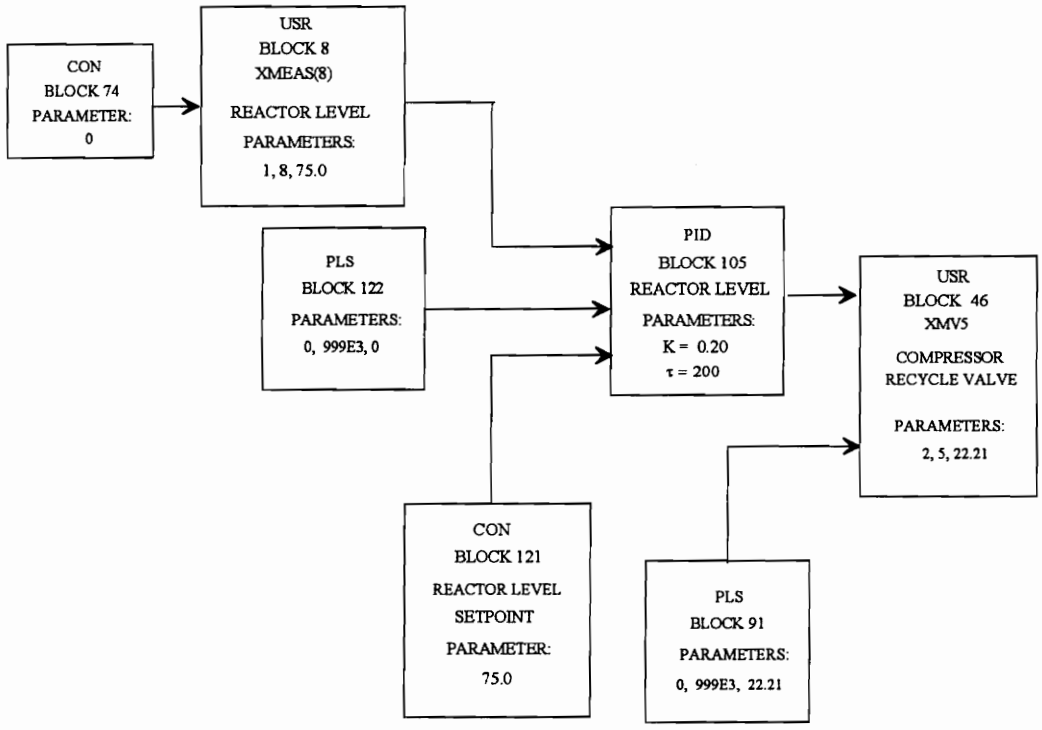
Appendix 5.C4.

Tutsim schematic diagram for the control of the C-feed valve (xmv4) for the Tennessee Eastman plantwide control process (IC4 controller).



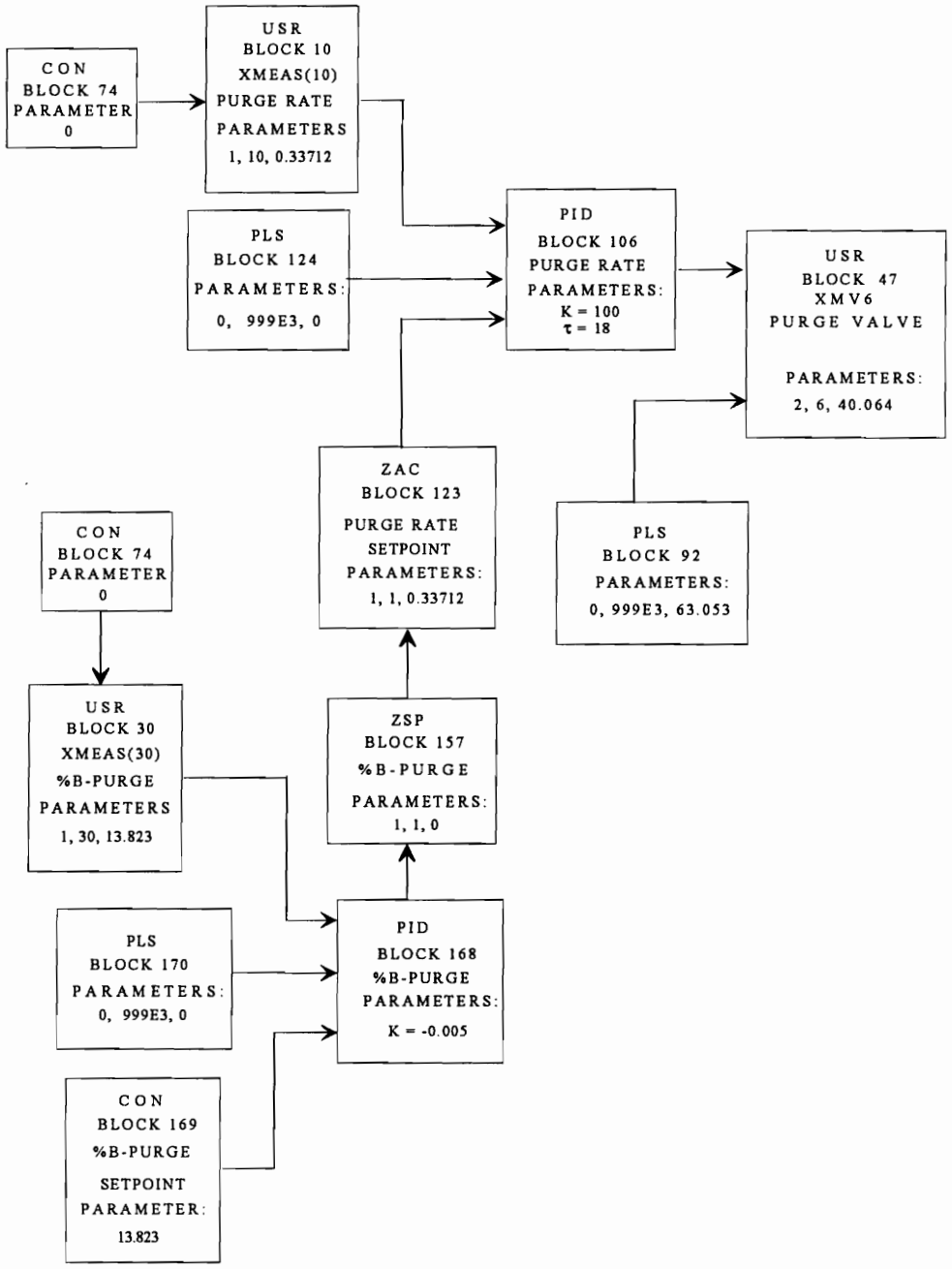
Appendix 5.C5.

Tutsim schematic diagram for the control of the compressor recycle valve (xmv5) for the Tennessee Eastman plantwide control process (IC5 controller).



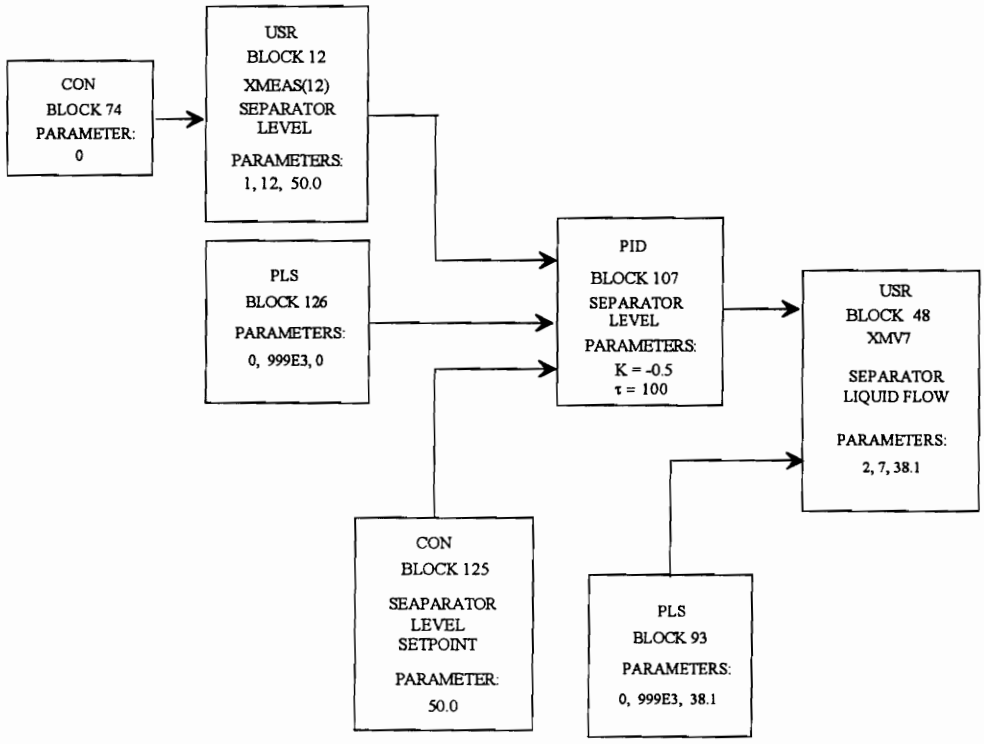
Appendix 5.C6.

Tutsim schematic diagram for the control of the purge valve (xmv6) for the Tennessee Eastman plantwide control process (IC6 and OC4 controllers).



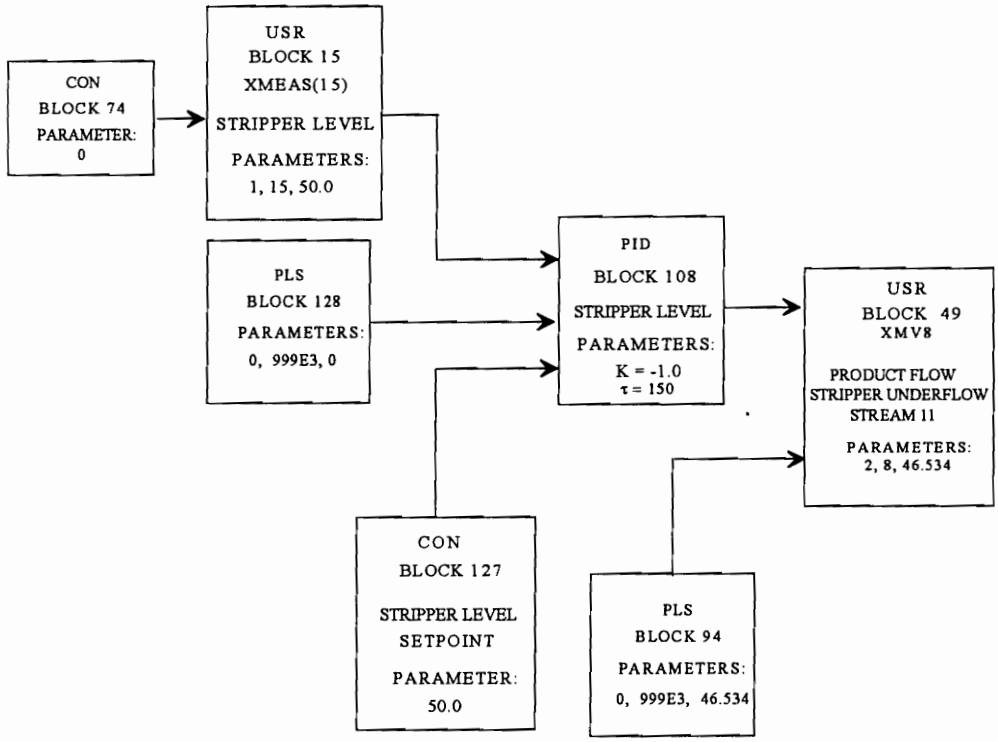
Appendix 5.C7.

Tutsim schematic diagram for the control of the separator liquid flow (xmv7), stream 10, for the Tennessee Eastman plantwide control process (IC7 controller).



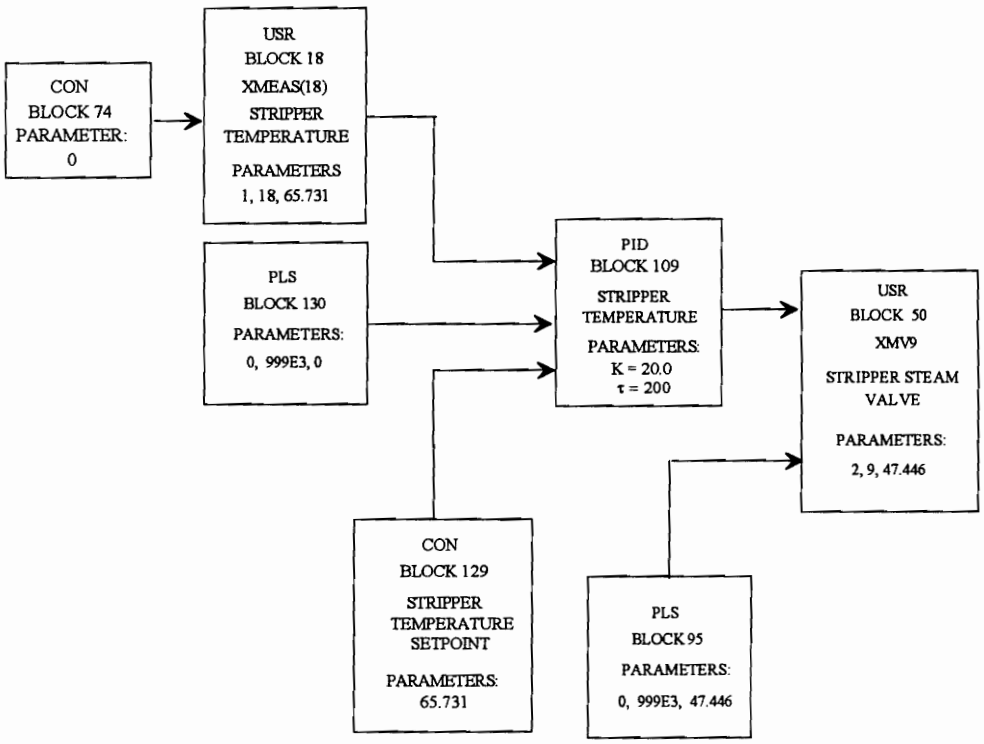
Appendix 5.C8.

Tutsim schematic diagram for the control of the stripper liquid flow (xmv8), stream 11, for the Tennessee Eastman plantwide control process (IC8 controller).



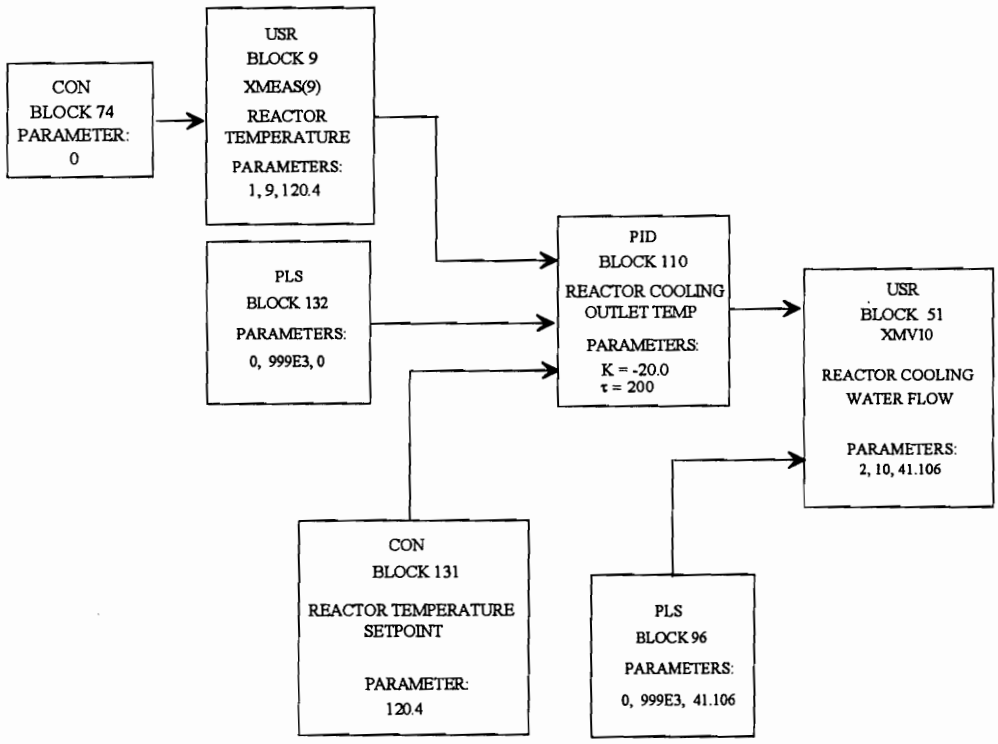
Appendix 5.C9.

Tutsim schematic diagram for the control of the stripper steam valve (xmv9) for the Tennessee Eastman plantwide control process (IC9 controller).



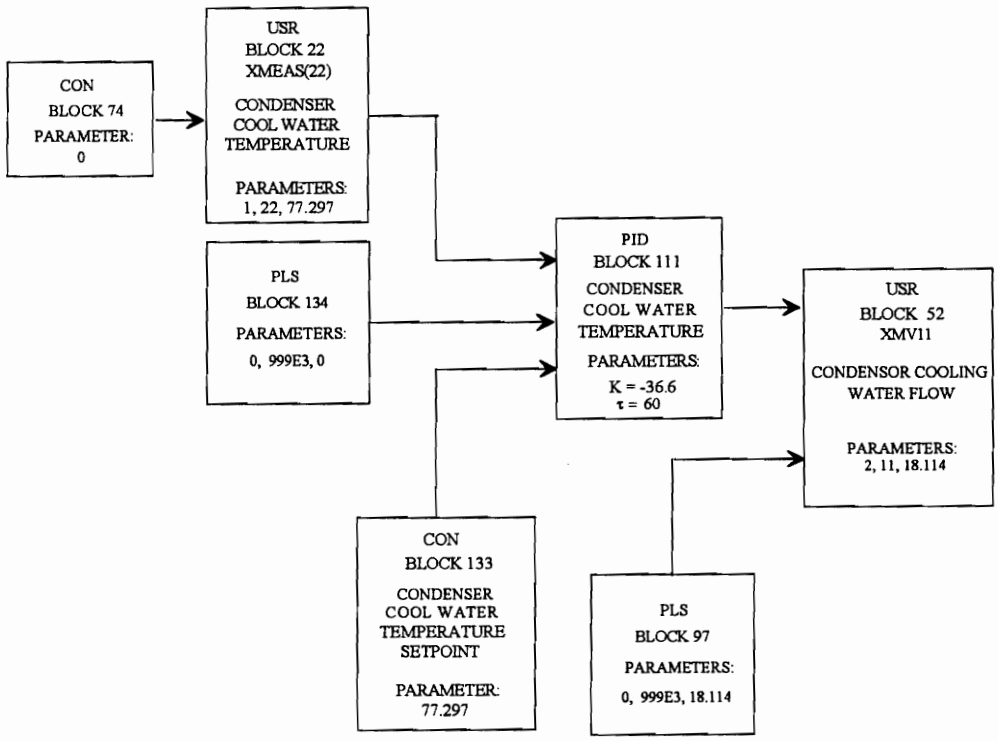
Appendix 5.C10.

Tutsim schematic diagram for the control of reactor cooling water flow (xmv10) for the Tennessee Eastman plantwide control process (IC10 controller).



Appendix 5.C11.

Tutsim schematic diagram for the control of condenser cooling water flow (xmv11) for the Tennessee Eastman plantwide control process (IC11).



References and Further Reading

Baily and Olis (1986), and Shuler and Kargi (1992) are two recommended texts for biochemical engineering fundamentals and bioprocess engineering. Ogunnaike and Ray (1994) is the most comprehensive text on the dynamics, modeling, and control of chemical processes, totaling 1,260 pages. This text gives the background on the model-based control (e.g., internal model control, IMC) in its Chapter 19, and covers the important developments on the model-predictive control (e.g.., dynamic matrix control, DMC) in its Chapter 26. Pages 271 to 444 of Arkun and Ray (1991) include seven state-of-the-art reports on model-based monitoring and control, particularly the model-predictive control (MPC). Shinskey (1995) gives an excellent industrial assessment of the strengths and limitations of using the model-based control as feedback controllers for the process industries. Miller et al. (1990) is a good edited volume on neural networks for control.

There are a number of review articles and book chapters that are relevant to the topics of the present chapter: Astrom and McAvoy (1992), and Stephanopoulos and Han on intelligent control, covering the applications of expert systems, neural networks, and fuzzy-logic systems to process control; Barto (1990), Werbos (1990b), and Hunt et al. (1992) on different approaches to applying neural networks to control problems; Morris et al. (1994) on neural networks for process modeling and control; and DiMassio et al. (1991) on neural networks for bioprocess modeling.

Some papers of particular interest are the following: Kramer (1991, 1992) on data compression using autoassociative neural networks; Staib (1993), and Staib and Staib (1992) on the commercial development and implementation of the Intelligent Arc Furnace (IAF) controller (see Section 5.6.B.2); Downs and Vogel (1993), McAvoy and Ye (1994), and Lyman and Geogakis (1995) on the Tennessee Eastman plantwide control problem (see Section 5.5); Donat et al. (1991) on the development of model-predictive control using neural networks; Graettinger et al. (1994a,b) on the commercial implementation of NeuCOP, the Neural Control and Optimization Package, at the Texaco Puget Sound Plant (Texas) for the model-predictive control and optimization of distillation columns; Krischer et al. (1993) on the use of neural networks for the model identification of a spatiotempory varying catalytic reaction; and Henson et al. (1995) on the novel habituating control strategies for chemical processes.

Adomaitis, R. A., R. M. Farber and J. L. Hudson, "Application of Neural Nets to System Identification and Bifurcation Analysis of Real World Experimental Data," Los Alamos National Laboratory Technical Report No. LA-UR-90-515, Los Alamos, ME (1990).

Anderson, C. W., "Learning to Control an Inverted Pendulum Using Neural Networks," *IEEE Control Systems Magazine*, **7**, 31, April (1989).

Anderson, C. W. and W. T. Miller, III, "A Challenging Set of Control Problems," pp. 475-510, in *Neural Networks for Control*, W. T. Miller, III, R. S. Sutton and P. J. Werbos, editors, MIT Press, Cambridge, MA (1990).

Arkun, Y. and W. H. Ray, editors, *Chemical Process Control - CPC IV*, Cache Corporation, Austin, TX and AIChE, New York (1991).

Astrom, K. and T. J. McAvoy, "Intelligent Control," *J. Process Control*, **2**, 115 (1992).

Bailey, J. E. and D. F. Ollis, *Biochemical Engineering Fundamentals*, 2nd edition, McGraw-Hill, New York (1986).

Banerjee, A. and Y. Arkun, "Control Configuration Design Applied to the Tennessee Eastman Plantwide Control Problem," *AIChE Annual Meeting*, St. Louis, MO, Nov. (1993).

Barto, A. G., "Connectionist Learning for Control: An Overview," pp. 5-56, in *Neural Networks for Control*, W. T. Miller, III, R. S. Sutton and P. J. Werbos, editors, MIT Press, Cambridge, MA (1990).

Barto, A., R. Sutton and C. W. Anderson, "Neuron-Like Adaptive Elements That Can Solve Difficult Learning Control Problems," *IEEE Transactions on Systems, Man, and Cybernetics, SMC-13*, 834 (1983).

Bates, D. M. and D. G. Watts, *Nonlinear Regression Analysis and Its Applications*, Wiley Interscience, New York (1988).

Bavarian, B., "Introduction to Neural Networks for Intelligent Control," *IEEE Control Systems Magazine*, 7, No.2, 3, April (1988).

Bekey, G. A. and P. G. Katona, "Bio-Control by Neural Networks," Summary of a Workshop Supported by the National Science Foundation, Arlington, VA, May 16-18, 1990, NSF Publication No. 91-17, Washington, D.C. (1991).

Bequette, B. W., "Nonlinear Control of Chemical Processes: A Review," *Ind Eng. Chem. Res.*, **30**, 1391 (1991).

Berenji, H. R. and P. S. Khedkar, "Adaptive Fuzzy Control with Reinforcement Learning," *Proceedings Amer. Control Conf.*, p. 1840, San Francisco, CA, June (1993).

Bhat, N. and T. J. McAvoy, "Use of Neural Nets for Dynamic Modeling and Control of Chemical Process Systems," *Comput. Chem. Eng.*, **14**, 573 (1990).

Bhat, N., P. A. Minderman, Jr., T. J. McAvoy and N. S. Wang, "Modeling Chemical Process Systems via Neural Computation," *IEEE Control Systems Magazine*, **8**, 24, April (1990).

Birky, G. J. and T. J. McAvoy, "A Neural Net to Learn the Design of Distillation Controls," Chem. Eng., Univ. of Maryland, College Park, MD (1990).

Blum, J., P. Villard, A. Leuba and D. M. Himmelblau, "Practical Issues in Applying Artificial Neural Networks for Identification in Model-Based Predictive Control," *AIChE Annual Meeting*, Miami, FL, Nov. (1992).

Bochereau, L., P. Bourgine, F. Bouyer and G. Muratet, "Using Recurrent Multilayer Neural Networks for Simulating Batch Reactors," *Proceedings of International Joint Conference on Neural Networks*, Singapore, p. 1512, November (1991).

Brandes, B. T., "An Industrial Application Using Feedforward Neural Networks for Improving Process Understanding, Modeling and Control," Paper 130h, *AIChE Annual Meeting*, Miami, FL, November (1992).

Breusegem, V. V., J. Thibault and A. Cheruy, "Adaptive Neural Models for On-Line Prediction in Fermentation," *Can. J. Chem. Eng.*, **69**, 481 (1991).

Bulsari, A. and H. Saxen, "System Identification of a Biochemical Process Using Feedforward Neural Networks," *Neural Computing*, **3**, 125 (1991).

Bulsari, A., B. Saxen and H. Saxen, "Feedforward Neural Networks for Bioreactor Control," *Proceedings of the Intern. Workshop on Artificial Neural Networks,* Stiges, Spain, June, 1993; pp. 682-689, in *Lecture Notes in Computer Sciences*, Vol. 686, Springer-Verlag, New York (1993).

Chen, C. L. and W. C. Chen, "A Novel Approach to the Design of a Self-Organizing Neural Controller," Dept. of Chem. Eng., National Taiwan Univ., Taipei, Taiwan (1991).

Chen, F. C., "Backpropagation Neural Networks for NonLinear Self-Tuning Adaptive Control," *IEEE Control Systems Magazine*, **8,** 44, April (1990).

Chen, S., S. A. Billings and P. M. Grant, "Nonlinear System Identification Using Neural Networks," *Int. J. Control*, **51**, 1191 (1990a).

Chen, S., S. A. Billings, C. F. N. Cowan and P. M. Grant, "Nonlinear System Identification Using Radial Basis Functions," *Int. J. System Sci.*, **21**, 2513 (1990b).

Chen , Q. and W. A. Weigand, "Adaptive Optimal Operation of a Bioreactor Based on a Neural Net Model," pp. 391-394, in *Modeling and Control of Biotechnical Processes 1992*, M. N. Karim and G. Stephanapoulas, editors, Pergamon Press, Oxford, United Kingdom (1992).

Chen , Q. and W. A. Weigand, "Dynamic Optimization of Nonlinear Processes by Combining Neural Net Model with UDMC," *AIChE J.*, **49**, (1994).

Chessari, C. J., G. W. Barton and J. A. Romagnoli, "A Strategy for the Nonlinear Control of Affine Systems Using Multiple Neural Networks," *Proceedings Amer. Control Conf.*, p. 1470, San Francisco, CA, June (1993).

Cleran, Y., J. Thibault, A. Cheruy and G. Corrieu, "Comparison of Prediction Performances Between Models Obtained by the Group Method of Data Handling and Neural Networks for the Alcoholic Fermentation Rate in Onology," *J. Ferm. Bioeng.*, **71**, 356 (1991).

Cooper, D. J. and A. M. Lalonde, "Process Behavior Diagnostics and Adaptive Process Control," *Comput. Chem. Eng.*, **14**, 541 (1990).

Cooper, D. J., R. F. Hinde, Jr. and L. Megan, "Pattern-Based Adaptive Process Control," *Comput. Chem. Eng.*, **14**, 1339 (1990a).

Cooper, D. J., R. F. Hinde, Jr. and L. Megan, "A Performance Feedback Neural Network for Pattern-Based Adaptive Process Control," *AIChE Annual Meeting*, Chicago, IL, November (1990b).

Cooper, D. J., L. Megan and R. F. Hinde, Jr., "Comparing Two Neural Networks for Pattern-Based Adaptive Process Control," *AIChE J.*, **38**, 41 (1992).

Davison, L. D., "Data Compression Using Straight-Line Interpolation," *IEEE Trans. Information Theory*, ***IT-14***, 390 (1968).

Desai, A. P. and D. E. Rivera, "Controller Structure Selection and System Identification for the Tennessee Eastman Challenge Problem Via Intelligent Process Control," *AIChE Annual Meeting*, St. Louis, MO, Nov. (1993).

Deshpande, P.B., J. A. Caldwell, S.S. Yerrapragada and M.A. Bholodia, "Should You Use Constrained Model Predictive Control," *Chem. Eng. Prog.*, **91**, No.3, 65, March (1995).

DiMassimo, C., M. J. Willis and G. A. Montague, "Bioprocess Model Building Using Artificial Neural Networks," *Bioprocess Eng.*, **7**, 77 (1991).

DiMassimo, C., G. A. Montague, M. J. Wills, M. T. Tham and A. J. Morris, "Towards Improved Penicillin Fermentation via Artificial Neural Networks," *Comput. Chem. Eng.*, **16**, 283 (1992).

Donat, J. S., N. Bhat and T. J. McAvoy, "Optimizing Neural-Net-Based Predictive Control," *Proceedings Amer. Control Conf.*, p. 2466, San Diego, CA, May (1990).

Donat, J. S., N. Bhat and T. J. McAvoy, "Neural-Net-Based Model-Predictive Control," *Intern. J. Control*, **54**, 1453 (1991).

Downs, J. J. and E. F. Vogel, "A Plantwide Industrial Process Control Problem," *Comput. Chem. Eng.*, **17**, 245 (1993).

Eaton, J. W., J. B. Rawlings, and L. H. Ungar, "Stability of Neural-Net-Based Model-Predictive Control," *Proceedings Amer. Control Conference*, p.2481, Baltimore, MD, June (1994).

Economou, C. G., M. Morari and B. O. Pallson, "Internal Model Control: 5. Extension to Nonlinear Systems," *Ind. Eng. Chem. Process Design and Develop.*, **25**, 403 (1986).

Endo, I. and T. Nagamune, "A Database System for Fermentation Processes," *Bioprocess Engineering*, **2**, 111 (1987).

Fogler, H. C., *Elements of Chemical Reaction Engineering*, 2nd edition, Prentice-Hall, Englewood Cliffs, NJ (1992).

Foster, W., F. Caollopy and L. Ungar, "Neural Network Forecasting of Short, Noisy Time Series," *Comput. Chem. Eng.*, **16**, 293 (1992).

Fu, P. C. and J. P. Barford, "A Hybrid Network-First Principles Approach to Structural Cell Modeling," *Comput. Chem. Eng.*, in press (1995).

Garcia, C. E. and M. Morari, "Internal Model Control: 1. A Unifying Review and Some New Results; 2. Design Procedure for Multivariable Systems; and 3. Multivariable Control Law Computation and Tuning Guidelines," *Ind. Eng. Chem. Process Design and Develop.*, **21**, 308 (1982); **24**, 472 (1985a); and **24**, 484 (1985b).

Garcia, C. E. and A. M. Morshedi, "Quadratic Programming Solution of Dynamic Matrix Control (QDMC)," *Chem Eng. Commun.*, **46**, 73 (1986).

Garcia, C. E., D. M. Prett and M. Morari, "Model-Predictive Control: Theory and Practice - A Survey," *Automatica*, **25**, 335 (1989).

Glassey, J., B. Kara, A. C. Ward, and G. Montagae, "Enhancing Fermentation Development Procedures via Artificial Neural Networks," pp. 147-152, in *Modeling and Control of Biotechnical Processes*, M. N. Karim and G. Stephanopoulos, editors, Pergaman Press, Oxford, United Kingdom (1992).

Graettinger, T. J., N. V. Bhat, K. Heckendom, and J. S. Buck, "Model-Predictive Control Using Neural Networks," *AIChE National Meeting*, Atlanta, GA, April (1994a).

Graettinger, T. J., N. V. Bhat, K. Heckendom, and J. S. Buck, "Adaptive Control with NeuCOP, the Neural Control and Optimization Package," *IEEE International Conference on Neural Networks* (1994b).

Guez, A., J. L. Eilbert and M. Kam, "Neural Network Architecture for Control," *IEEE Control Systems Magazine*, 7, No.2, 22, April (1988).

Haesloop, D. and B. R. Holt, "A Neural Network Structure for System Identification," *Proceedings Amer. Control Conf.*, p. 2460, San Diego, CA, May (1990).

Haesloop, D.G., "System Identification and Control Using Neural Networks with Combined Linear and Nonlinear Mapping Functionality," Ph.D. dissertation, Chemical Engineering, University of Washington, Seattle, WA (1991).

Hale, J. C. and H. L. Sellers, "Historical Data Recording for Process Computers," *Chem. Eng. Prog.*77, No. 11, 38, Nov. (1981).

Hall, C., "Neural Net Technology: Ready for Prime Time?," *IEEE Expert*, 7, No.6, 2 (1992).

Henson, M. A., B. A. Ogunnaike, and J. S. Schwaber, "Habituating Control Strategies for Process Control," *AIChE J.*, **41**, 604 (1995).

Hernandez, E. and Y. Arkun, "Neural Network Modeling and an Extended DMC Algorithm to Control Nonlinear Systems," *Proceedings Amer. Control Conf.*, p. 2454, San Diego, CA, May (1990).

Hernandez, E. and Y. Arkun, "Study of the Control-Relevant Properties of Backpropagation Neural Network Models of Nonlinear Dynamic Systems," *Comput. Chem. Eng.*, **16**, 227 (1992).

Himmelblau, D. M., "A Comparison of Artificial Neural Networks with Traditional Modeling for Model-Predictive Control," *AIChE Annual Meeting*, Los Angeles, CA, Nov. (1991).

Hinde, R. F., Jr. and D. J. Cooper, "Adaptive Process Control Using Pattern-Based Performance Feedback," *J. Process Control*, **1**, 228 (1991).

Hinde, R. F., Jr. and D. J. Cooper, "A Neural Network for Closed-Loop Process Excitation Diagnosis and Adaptive Process Control," *AIChE Annual Meeting*, Miami, FL, Nov. (1992).

Hinde, R. F., Jr. and D. J. Cooper, "Using Pattern Recognition in Controller Adaptation and Performance Evaluation," *Proceedings Amer. Control Conf.*, p. 74, San Francisco, CA, June (1993).

Hinde, R. F., Jr. and D. J. Cooper, "A Pattern-Based Approach to Excitation Diagnosis for Adaptive Process Control," *Chem. Eng. Sci.*, **49**, 1403 (1994).

Horikawa, S. I., T. Furuhashi, and Y. Uchikawa, "On Fuzzy Modeling Using Fuzzy Neural Networks with the Backpropagation Algorithm," *IEEE Trans. Neural Networks*, **3**, 801 (1992).

Hoskins, J. C. and D. M. Himmelblau, "Process Control Via Neural Networks and Reinforcement Learning," *Comput. Chem. Eng.*, **16**, 241 (1992).

Hosogi, S., "Manipulator Control Using Layered Neural Network Model with Self-Organizing Mechanism," *Proceedings of International Joint Conference on Neural Networks*, pp. 2-217, Lawrence Erlbaum Assosiates, Washington, D.C. (1990).

Hsiung, J. T. and D. M. Himmelblau, "Development of Control Strategies Via Artificial Neural Networks and Reinforcement Learning," *Proceedings Amer. Control Conf.*, Boston, MA, June (1991).

Hunt, K. J. and D. Sbarbaro, "Neural Networks for Nonlinear Internal Model Control," *IEEE Proceedings - D.*, **138**, 431 (1991).

Hunt, K. J., D. Sbarbaro, R. Zbikowski and P. J. Gawathrop, "Neural Networks in Control Systems: A Survey," *Automatica*, **28**, 1083 (1992).

Hofland, A. G., A. J. Morris and G. A. Montague, "Radial-Basis-Function Networks Applied to Process Control," *Proceedings Amer. Control Conf.*, p. 480, Chicago, IL, June (1990).

Hoskins, J. C. and D. M. Himmelblau, "Process Control via Artificial Neural Networks and Reinforcement Learning," *Comput. Chem. Eng.*, **16**, 241 (1992).

Hudson, J. L., M. Kube, R. A. Adomaitis, I. G. Kevrekidis, A. S. Lapedes and R. M. Farber, "Nonlinear Signal Processing and System Identification Application to Time Series from Electrochemical Reactions," *Chem. Eng. Sci.*, **45**, 2075 (1990).

Ichikawa, Y. and T. Sawa, "Neural Network Application for Direct Feedback Controllers," *IEEE Trans. Neural Networks*, **3**, 224 (1992).

Ishida, M. and J. X. Zhan, "Control of a Process with Time Delay by Policy-and-Experience-Driven Neural Networks," *J. Chem. Eng. Japan*, **25**, 763 (1992).

Ishida, M. and J. X. Zhan, "Neural-Network Control for a MIMO Process with Time-Delay," *J. Chem. Eng. Japan*, **26**, 337 (1993).

Joseph, B. and F. Wang-Hanratty, "Predictive Control of Quality in a Batch Manufacturing Process Using Artificial Neural Network Models," *Ind. Eng. Chem. Res.*, **32**, 1951 (1993).

Joshi, A. and R. L. Motard, "Process Trend Detection Using the Frazier-Jawerth Transformation and Neural Networks," *Comput. Chem. Eng.*, in press (1995).

Karjala, T. W. and D. M. Himmelblau, "Dynamic Data Rectification by Recurrent Neural Networks versus Traditional Methods," *AIChE J.*, **40**, 1865 (1994).

Karim, M. N. and S. L. Rivera, "Comparison of Feedforward and Recurrent Neural Networks for Bioprocess State Estimation," *Comput. Chem. Eng.*, **16**, S369 (1992a).

Karim, M. N. and S. L. Rivera, "Use of Recurrent Neural Networks for Bioprocess Identification in On-Line Optimization by Micro-Genetic Algorithm," *Proceedings Amer. Control Conf.*, p. 1931, Chicago, IL, (1992b).

Klimasanskas, C., "Neural Nets and Noise Filtering," *Dr. Dobb's Journal*, **14**, No.1, 32 (1989).

Kooi, S. B. L. and Khorasani, "Control of the Woodchip Refiner Using Neural Networks," *Tappi Journal*, **75**, 156 (1992).

Kraft, L. G. and D. P. Campagna, "A Comparison between CMAC Neural Network Control and Two Traditional Control Systems," *IEEE Control Systems Magazine*, **10**, No.2, 36, April (1990).

Kramer, M. A., "Nonlinear Principal Component Analysis Using Autoassociative Neural Networks," *AIChE J.*, **37**, 233 (1991).

Kramer, M. A., "Autoassociative Neural Networks," *Comput. Chem. Eng.*, **16**, 313 (1992).

Kraus, T. W. and T. J. Myron, "Self-Turning PID Controller Using Pattern-Recognition Approach," *Control Eng.*, **31**, 106 (1984).

Krischer, K., R. Rico-Martinez, H. H. Rotermund, G. Ertl and J. L. Hudson, "Model Identification of a Spatiotempory Varying Catalytic Reaction," *AIChE J.*, **39**, 89 (1993).

Kube, M. C., "Neural Network Modeling of Copper Electrodissolution," Ph.D. dissertation, Chemical Engineering, Univ. of Virginia, Charlottesville, VA (1993).

Kuespert, D. R. and T. J. McAvoy, "Human Process Operator Knowledge Extraction with Neural Networks," *AIChE Annual Meeting*, Los Angeles, CA, Nov. (1991).

Kuperstein, M. and J. Rubinstein, "Implementation of an Adaptive Neural Controller for Sensory-Motor Coordination," *IEEE Control Systems Magazine*, **8**, No.2, 25, April (1989).

Kurtanjek, Z., "Modeling and Control by Artificial Neural Networks in Biotechnology," *Comput. Chem. Eng.*, **18**, S627 (1994).

Lant, P. A., M. J. Willis, G. A. Montague, M. T. Tham and A. J. Morris, "A Comparison of Adaptive Estimation with Neural-Based Techniques for Bioprocess Applications," *Proceedings Amer. Cont. Conf.*, p. 2173, San Diego, CA, June (1990).

Lapedes, A. and R. Farber, "Nonlinear Signal Processing Using Neural Networks: Prediction and System Modeling," Los Alamos National Laboratory Technical Report No. LA-UR-87-2662, Los Alamos, ME (1987).

Latrille, E., G. Corrieu and J. Thibault, "pH Prediction and Final Fermentation Time Determination in Lactic Acid Batch Fermentations," *Proceedings of the European Symposium on Computer-Aided Process Engineering - 2*, Toulouse, France, S423, Oct. (1992).

Latrille, E., G. Corrieu, and J. Thibault, "Neural Network Models for Final Time Determination in Fermented Milk Production," *Comput. Chem. Eng.*, **18**, 1171 (1994).

Lee, M. and S. Park, "A New Scheme Combining Neural Feedforward Control with Model-Predictive Control," *AIChE J.*, **38**, 193 (1992).

Levenspiel, O., "The Monod Equation: A Re-Visit and a Generalization to Product-Inhibition Situations," *Biotech. and Bioeng.*, **22**, 1671 (1980).

Lichtenwalner, P. F., "Neural-Network-Based Control for the Fiber Placement Composite Manufacturing Process," *J. Materials Eng. and Performance*, **2**, 687 (1993).

Lin, J. L., S. H. Wong, and S. W. Yu, "Optimal Multiloop Feedback Design Using Simulated Annealing and Neural Networks," *AIChE J.*, **41**, 430 (1995).

Linden, A. and J. Kindermann, "Inversion of Multilayer Nets," *Proceedings of International Joint Conference on Neural Networks*, Washington, D.C., Vol II, p.425 (1989).

Linko, P. and Y. Zhu, "Neural Network Programming in Bioprocess Variable Estimation and State Estimation," *J. Biotech.*, **21**, 253 (1991).

Linko, P. and Y. Zhu, "Neural Network Programming in Bioprocess Estimation and Control," pp.163-166, in *Modeling and Control of Biotechnical Process*, M. N. Karim and G. Stephanopoulos, Editors, Pergamen Press, Oxford, United Kingdom (1992).

Luyben, W. L., *Process Modeling, Simulation, and Control of Chemical Processes*, 2nd edition, McGraw Hill, New York (1990).

Lyman, P. P., "Plantwide Control Structures for the Tennessee Eastman Process," M.S. thesis, Dept. of Chemical Engineering, Lehigh University, Bethlehem, PA (1992).

Lyman, P. R. and C. Geogakis, "Plantwide Control of the Tennessee Eastman Problem," *Comput. Chem. Eng.*, **19**, 321 (1995).

MacGregor, J. F., T. F. Marlin and J. V. Kresta, "Some Comments of Neural Networks and Other Empirical Modeling Techniques," p. 665, in *Chemical Process Control CPC-IV*, Y. Arkun and W. H. Ray, editors, CACHE Corporation, Austin, TX and AIChE, New York (1991).

MacMurray, J. and D. M. Himmelblau, "Identification of a Packed Distillation Column for Control Via Artificial Neural Networks," *Proceedings Amer. Control Conf.*, p. 1445, San Francisco, CA, June (1993).

Maiorelle, B. L., H. W. Blanch and C. R. Wilkes, "Byproduct Inhibition Effects on Ethanol Fermentation by Saccharomyces Cerevisiac," *Biotech. Bioeng.*, **25**, 103 (1983).

Maiorelle, B. L., H. W. Blanch and C. R. Wilkes, "Economic Consideration of Alternative Ethanol Fermentation Process," *Biotech. Bioeng.*, **26**, 1003 (1984).

Manchanda, S., M. J. Willis, M. T. Tham, C. DiMassimo, and G. A. Montaque, "An Appraisal of Nonlinear Control Philosophies for Application to a Biochemical Process." *Proceedings Amer. Control Conf.*, p.1322, San Diego, CA (1991).

Morari, M. and E. Zafiriou, *Robust Process Control*, Prentice-Hall, Engle

Miller, W. T., R. E. Sutton, and P. J. Werbos, Editors, *Neural Networks for Control*, MIT Press, Cambridge, MA (1990).

McAvoy, T. J. and N. Ye, "Base Control for the Tennessee Eastman Problem," *Comput. Chem. Eng.*, **18**, 383 (1994).

Megan, L. and D. J. Cooper, "Neural-Networks-Based Adaptive Control via Temporal Pattern Recognition," *Can. J. Chem. Eng.*, **70**, 1208 (1992).

Megan, L. M. and D. J. Cooper, "Pattern-Recognition-Based Adaptive Control of Two-Input/Two-Output Systems Using ART2-A Neural Networks," *Ind. Eng. Chem. Res.*, **33**, 1510 (1994).

Megan L. and D. J. Cooper, "A Neural Network Strategy for Disturbance Rejection and Adaptive Multi-Input/Multi-Output Control," *Comput. Chem. Eng.*, **19**, 171 (1995).

Minderman, P. A., Jr. and T. J. McAvoy, "Neural Net Modeling and Control of a Municipal Waste Water Process," *Proceedings Amer. Control Conf.*, p. 1480, San Francisco, CA, June (1993).

Montague, G. A., M. J. Willis, and A. J. Morris, "Artificial Neural Network Model-Based Control," *Proceedings Amer. Control Conference*, p.2134, Baltimore, MD, June (1994).

Morari, M. and E. Zafiriou, *Robust Process Control*, Prentice-Hall, Englewood Cliffs, NJ (1989).

Morari, M., C. E. Garcia, J. H. Lee, and D. M. Prett, *Model Predictive Control*, in press, Prentice-Hall, Englewood Cliffs, NJ.

Morris, A. J., G. A. Montague, M. T. Tham, M. Aynsley, C. Di Massino and P. Lant, "Towards Improved Process Supervision-Algorithms and Knowledge-Based Systems," p. 585, in *Chemical Process Control CPC-IV*, Y. Arkun and W. H. Ray, editors, CACHE Corporation, Austin, TX and New York (1991).

Morris, A. J., G. A. Montague and M. J. Willis, "Artificial Neural Networks: Studies in Process Modeling and Control," *Chem. Eng. Res. Develop.*, **72**, No. A1, 3 (1994).

Nahas, E. P., M. A. Henson and D. E. Seborg, "Nonlinear Internal Model Control Strategy for Neural Network Models," *Comput. Chem. Eng.*, **16**, 1039 (1992).

Naidu, S., E. Zafiriou and T. J. McAvoy, "Application of Neural Networks on the Detection of Sensor Failure During Operation of a Control System," *Proceedings Amer. Control Conf.*, p. 1336, Pittsburgh, PA, June (1989).

Naidu, S. R., E. Zafiriou, and T. J. McAvoy, "Use of Neural Networks for Sensor Failure Detection in a Control System," *IEEE Control Systems Magazine*, **10**, No.2, 49, April (1990).

Narendra, K. S., "Adaptive Control Using Neural Networks," pp. 115-142, in *Neural Networks for Control*, W. T. Miller, III, R. S. Sutton and P. J. Werbos, Editors, MIT Press, Cambridge, MA (1990).

Narendra, K. S. and S. Mukopahyay, "Intelligent Control Using Neural Networks," *IEEE Control Systems Magazine*, **12**, No.2, 11, April (1992).

Narendra, K. S. and K. Parthasarathy, "Identification and Control of Dynamic Systems Using Neural Networks," *IEEE Trans. Neural Networks*, **1**, 4 (1990).

Narendra, K. S. and K. Parthasarathy, "Gradient Methods for the Optimization of Dynamical Systems Containing Neural Networks," *IEEE Trans. Neural Networks*, **2**, 252 (1991).

NeuralWare, Inc., *Neural Computing: A Technical Handbook for Professional II/PLUS and NeuralWorks Explorer*, Pittsburgh, PA (1993).

NeuralWare, Inc., "NeuCOP: The Neural Network Control and Optimization Package," Pittsburgh, PA (1993).

NG, C. S. and G. Stephanopoulos, "A Hierarchical Approach to the Synthesis of Plantwide Control Structure," *AIChE Annual Meeting*, San Fransisco, CA, November (1994).

Nguyen, D. H. and B. Widrow, "Neural Networks for Self-Learning Control Systems," *IEEE Control Systems Magazine*, **10**, No.2, 18, April (1990).

Nikolaou, M., "Neural Network Modeling of Nonlinear Dynamic Systems," *Proceedings of Amer. Control Conf.*, p. 1460, San Francisco, CA, June (1993).

Nikolaou, M. and V. Hanagandi, "Recurrent Neural Networks in Decoupling Control of Multivariable Nonlinear Systems," *AIChE Annual Meeting*, Miami, FL, Nov. (1992).

Nikolaou, M. and V. Hanagandi, "Control of Nonlinear Dynamic Systems Modeled by Recurrent Neural Networks," *AIChE J.*, **39**, 1890 (1993).

Oishi, K., M. Tominaga, A. Kawato, Y. Abe, S. Imayasu, and A. Nanma, "Application of Fuzzy Control Theory to Sake Brewing Process," *J. Ferm. Bioeng.*, **72**, 115 (1991).

Oishi, K., M. Tominaga, A. Kawato and S.Imayasu, "Analysis of the State Characteristics of Sake Brewing with a Neural Network," *J. Ferm. Bioeng.*, **73**, 153 (1992a).

Oishi, K., M. Tominaga, A. Kawato Y. Abe, S. Imayasu, and A. Nanma, "Development of On-Line Sensoring and Computer-Aided Control System for Sake Brewing," *J. Biotechnology*, **24**, 53 (1992b).

Ogunnaike, B. A. and W. H. Ray, *Process Dynamics, Modeling, and Control*, Oxford University Press, New York, NY and Oxford, England (1994).

Palarajjhala, S., R. L. Motard and B. Joseph, "Control System Synthesis for Chemical Plants," Department of Chemical Engineering, Washington Univ., St. Louis, MO (1994).

Panier, E. R., and A. L. Tits, "On Combining Feasibility, Descent and Superlinear Convergence in Inequality Constrained Optimization," *Math. Prog.*, **59**, 261 (1993).

Parlos, A. G., A. F. Atiya and K. T. Chang, "Nonlinear Identification of Process Dynamics Using Neural Networks," *Nuclear Technology*, **97**, 79 (1992).

Passino, K. M., M. A. Sartori, and P. J. Antsaklis, "Neural Computing for Numeric-to-Symbolic Conversion in Control Systems," *IEEE Control Systems Magazine*, **9**, No.2, 44, April (1989).

Patwardhan, A. A., J. B. Rawlings, and T. F. Edgar, "Nonlinear Model-Predictive Control," *Chem. Eng. Commun.*, **87**, 123 (1990).

Pearlmutter, B. H., "Learning State Space Trajectories in Recurrent Neural Networks," *Neural Computation*, **1**, 263 (1989).

Piñeda, F. J., "Generalization of Back-Propagation to Recurrent Neural Networks," *Phys. Rev. Lett.*, **59**, 2229 (1987).

Piovoso, M., K. Kosanovich, V. Rokhlenko and A. Guez, "A Comparison of Three Nonlinear Controller Designs Applied to a Nonadiabatic First-Order Exothermic Reaction in a CSTR," *Proceedings Amer. Control Conf.*, p. 490, Chicago, IL, June (1992).

Pottmann, M. and D. E. Seborg, "A Nonlinear Predictive Control Strategy Based on Radial-Basis-Function Networks," *Proceedings of IFAC DYCORD+ Conference*, p. 309, College Park, MD, April (1992).

Pollard, J. F., M. R. Broussard, D. B. Garrison and K. Y. San, "Process Identification Using Neural Networks," *Comput. Chem. Eng.*, **16**, 253 (1992).

Prett, D. M. and C. E. Garcia, *Fundamental Process Control*, Butterworths, Stanchan, MA (1988).

Price, R. M., "Design of Plantwide Regulatory Control Systems," Ph.D. dissertation, Dept. of Chemical Engineering, Lehigh University, Bethlehem, PA (1993).

Price, R. M. and C. Geogakis, "Plantwide Regulatory Control Design Procedure Using a Tiered Framework," *Ind. Eng. Chem. Res.*, **32**, 2693 (1993).

Price, R. M., P. R. Lyman and C. Geogakis, "Throughput Manipulation in Plantwide Control Structures," *Ind. Eng. Chem. Res.*, **33**, 1197 (1994).

Procyk, T. J. and E. H. Mamdani, "A Linguistic Self-Organizing Process Controller," *Automatica*, **15**, 15 (1979).

Psaltis, D., A. Sideris, and A. A. Yamamura, "A Multilayered Neural Network Controller," *IEEE Control Systems Magazine*, **8**, No.2, 17, April (1988).

Psichogios, D. C. and L. H. Ungar, "Direct and Indirect Model-Based Control Using Artificial Neural Networks," *Ind. Eng. Chem. Res.*, **30**, 2564 (1991).

Psichogios, D. C. and L. H. Ungar, "Process Modeling Using Structural Neural Networks," *Proceedings Amer. Control Conf.*, p. 1917, Chicago, IL, June (1992a).

Psichogios, D. C. and L. H. Ungar, "A Hybrid Neural Network - First Principles Approach to Process Modeling," *AIChE J.*, **38**, 1499 (1992b).

Quantrille, T. E. and Y. A. Liu, *Artificial Intelligence in Chemical Engineering*, Academic Press, San Diego, CA (1991).

Raich, A., X. Wu and A. Cinar, "Approximate Dynamic Models for Chemical Processes: A Comparative Study of Neural Networks and Nonlinear Time Series Modeling Techniques," *AIChE Annual Meeting*, Miami, FL, Nov. (1992).

Raju, G. K. and C. L. Cooney, "Using Neural Networks for the Interpretation of Bioprocess Data," pp.425-428, in *Modeling and Control of Biotechnical Processes*, M. N. Karim and G. Stephanopoulos, Editors, Pergamon Press, Oxford, United Kingdom (1992).

Ramchandra, S. and R. R. Rhinehart, "A Very Simple Structure for Neural Network Control of Distillation," accepted for publication, *J. Process Control*, December (1994)

Rao, V. R. and W. K. Lee, "Neural Networks Approach to Automate Startup and Control of an Exothermic Batch Reactor," *Proceedings Amer. Control Conf.*, p. 2854, San Diego, CA, May (1990).

Ray, W. H., *Advanced Process control*, McGraw-Hill, New York (1981).

Ricker, N. L., "Model-Predictive Control of a Continuous, Nonlinear, Two-Phase Reactor," *J. Process Control*, **3**, 109 (1993).

Ricker, N. L., "Optimal Steady-State Operation of the Tennessee Eastman Challenge Process," *Comput. Chem. Eng.*, in press (1995).

Ricker, N. L. and J. H. Lee, "Nonlinear Model-Predictive Control of the Tennessee Eastman Challenge Process," *Comput. Chem. Eng.*, in press (1995a).

Ricker, N. L. and J. H. Lee, "Nonlinear Modeling and State Estimation for the Tennessee Eastman Challenge Process," *Comput. Chem. Eng.*, in press (1995b).

Rico-Martinez, R., K. Krischer and I. G. Kevrekidis, M. C. Kube and J. L. Hudson, "Discrete vs. Continuous-Time Nonlinear Signal Processing of Cu Electrodissolution Data," *Chem. Eng. Commu.*, **118**, 25 (1992).

Rico-Martinez, R., Y. G. Kevrekidis, M. C. Kube and J. L. Hudson, "Discrete- vs. Continuous-Time Nonlinear Signal Processing: Attractors, Transitions and Parallel Implementation Issues," *Proceedings Amer. Control Conf.*, p. 1475, San Francisco, CA, June (1993).

Rivera, S. L., "Neural Networks and Micro-Genetic Algorithms for State Estimation and Optimization of Bioprocesses," Ph.D. dissertation, Dept. of Chemical Engineering, Colorado State University, Fort Collins, CO (1992).

Rivera, S. L. and M. N. Karim, "On-Line Estimation of Bioreactors Using Recurrent Neural Networks," pp. 159-162, in *Modeling and Control of Biotechnical Processes*, M. N. Karim and G. Stephanopoulos, editors, Pergaman Press, Oxford, United Kingdom (1992).

Rivera, D. E., M. Morari, and S. Skogestad, "Internal Model Control, 4. PID Controller Design," *Ind. Eng. Process Design Dev.*, **25**, 252 (1986).

Roat, S. D., A. Farell, and C. Moore, "Applications of Neural Networks and System Cultivation to a Nonlinear Optimal Control Algorithm," *AIChE Annual Meeting*, Chicago, IL, November (1990).

Roat, S. D., "The Application of Neural Networks and System Cultivation with Nonlinear Optimal Control Algorithm for Chemical Process Industry," Ph.D. dissertation, University of Tennessee, Knoxville, TN (1991).

Rodriegues, C. C., C. L. Nasimento, Jr., and T. Yoneyama, "An Auto-Tuning Controller with Supervised Learning Using Neural Nets," *Proceedings of the Intern. Conf. on Control '91*, Vol.1, p.140, Institution of Electrical Engineers, London, March (1991).

Ruenglertpanyakul, W., K. B. Konstantinor, and T. Yoshida, "Application of Neural Networks to Variables Estimation and Stage Identification in Phenylalanine Production," pp. 429-432, in *Modeling and Control of Biotechnical Processes*, M. N. Karim and G. Stephanopoulos, editors, Pergaman Press, Oxford, United Kingdom (1992).

Rony, P. R., "ChE 3015-3016: Process Measurements and Control," Chemical Engineering, Virginia Tech, Blacksburg, VA (1994).

Rony, K. E., C. Manhard, and P. R. Rony, "Process Simulation in Batch Process Controls Education and Practice: Rapid Data Exchange Between a TI 545 Programmable Logic Controller and Real-Time TUTSIM," Pittsburgh Computer Simulation Conference, Pittsburgh, PA, April (1991).

Rony, K. E. and P. R. Rony, "The TI 500 Series Programmable Logic Controllers: Merging of TUTSIM Dynamic Simulation With APT Control System Design," Computer and Systems Technology Division Communications, American Institute of Chemical Engineers, **13**, No.2, 14, Summer (1990).

Samad, T. and T. Su, "Neural Networks as Process Controllers: Optimization Aspects," *Proceedings Amer. Control Conference*, p.2486, Baltimore, MD, June (1994).

Sanner, R. M. and J. J. E. Slotine, "Gaussian Networks for Direct Adaptive Control," *IEEE Trans. Neural Networks*, **3**, 837 (1992).

Sartori, M. A. and P. J. Antsaklis, "Implementation of Learning Control Systems Using Neural Networks," *IEEE Control Systems Magazine*, **12**, No.2, 49, April (1992).

Sato, M., "A Real Time Learning Algorithm for Recurrent Analog Neural Networks," *Biol. Cybern.*, **62**, 237 (1990a).

Schnelle, D. and J. Fletcher, "Using Neural-Based Process Modeling in Process Control," *Advances in Instrumentation and Control*, **45**, 991 (1990).

Scott, G. M., "Knowledge-Based Artificial Neural Networks for Process Modeling and Control," Ph.D. Dissertation, University of Wisconsin, Madison, WI (1993).

Scott, G. M. and W. H. Ray, "Creating Efficient Nonlinear Neural Network Process Models that Allow Model Interpretation," *J. Process Control*, **3**, 163 (1993a).

Scott, G. M. and W. H. Ray, "Experiences with Model-Based Controllers Based on Neural Network Process Models," *J. Process Control*, **3**, 179 (1993b).

Scott, G. M. and W. H. Ray, "Neural Network Process Models Based on Linear Model Structures," *Neural Computation*, **6**, 718 (1994).

Scott, G. M., J. W. Schavlik and W. H. Ray, "Refining PID Controllers Using Neural Networks," *Neural Computation*, **4**, 746 (1992).

Schenker, B. and M. Agarwal, "Robust Predictive Control Using Neural-Network Models," *AIChE Annual Meeting*, St. Louis, MO, November (1993).

Schenker, B. and M. Agarwal, "Experimental Application of Control Based on State-Feedback Neural-Network Predictors," *AIChE Annual Meeting*, San Francisco, CA, November (1994).

Seborg, D. E., T. F. Edgar, and D. A. Mellichamp, *Process Dynamics and Control,* Wiley, New York, NY (1989).

Sheppard, C. P., C. R. Gent and R. M. Ward, "A Neural-Network-Based Furnace Control System," *Proceedings Amer. Control Conf.*, p. 500, Chicago, IL, June (1992).

Shi, Z. and K. Shimizu, "Neuro-Fuzzy Control of Bioreactor Systems with Pattern Recognition," *J. Fermentation Bioeng.*, **74**, 39 (1992a).

Shi, Z. and K. Shimizu, "Fuzzy Control of Fed-Batch Fermentation with the Aid of Neural Networks," pp. 167-172, in *Modeling and Control of Biotechnical Processes*, M. N. Karim and G. Stephanopoulos, editors, Pergaman Press, Oxford, United Kingdom (1992b).

Shinsky, F. G., *Feedback Controllers for the Process Industries*, McGraw Hill, New York (1994).

Shuler, M. L. and F. Kargi, *Bioprocess Engineering*, Prentice-Hall, Englewood Cliffs, NJ (1992).

Simutis, K., I. Havlik, and A. Lubbert, "Process Monitoring and Prediction in a Production-Scale Beer Fermentation Using Fuzzy-Aided Extended Kalman Filter and Neural Network," pp. 95-98, in *Modeling and Control of Biotechnical Processes*, M. N. Karim and G. Stephanopoulos, editors, Pergaman Press, Oxford, United Kingdom (1992).

Smith, C. A. and A. B. Corripio, *Principles and Practice of Automatic Process Control*, Wiley, New York (1985).

Song, J. J. and S. Park, "Neural Model-Predictive Control of Nonlinear Chemical Processes," *J. Chem. Eng. Japan*, **26**, 347 (1993).

Staib, W. E., "The Intelligent Arc Furnace: Neural Networks Revolutionize Steelmaking," *Proceedings of World Congress on Neural Networks*, p.I-466, Portland, OR, July (1993).

Staib, W. E. and R. B. Staib, "The Intelligent Arc Furnace Controller: A Neural Network Electrode Position Optimization System for the Electrode Arc Furnace," *Proceedings of International Joint Conference on Neural Networks*, p.III-1, Baltimore, MD (1992).

Stephanopoulas, G. and C. Han, "Intelligent Systems in Process Engineering: A Review," *5th International Symposium on Process Systems Engineering*, Kyongju, Korea, May-June (1994).

Su, H. T. and T. J. McAvoy, "Identification of Chemical Processes Using Recurrent Networks," *Proceedings Amer. Control Conf.*, p. 2314, Boston, MA, June (1991).

Su, H. T. and T. J. McAvoy, "Integration of Multilayer Perceptron Networks and Linear Dynamic Models: A Hammerstein Modeling Approach," *Ind. Eng. Chem. Res.*, **32**, 1927 (1993).

Su, H. T., N. Bhat, P. Minderman and T. J. McAvoy, "Integrating Neural Networks with First Principles Models for Dynamic Modeling," *Proceedings of IFAC DYCORD Symp.*, College Park, MD, April (1992a).

Su, H. T., T. J. McAvoy and P. Werbos, "Long-Term Predictions of Chemical Processes Using Recurrent Neural Networks," *Ind. Eng. Chem. Res.*, **31**, 1338 (1992b).

Su, H. T., P. A. Minderman, Jr., T. J. McAvoy and J. Wray, "CONSCIENCE: Control and System Identification Using Elements of Neural Network Computation Engineering," *Proceedings Amer. Control Conf.*, p. 485, Chicago, IL, June (1992c).

Syu, M. J., "Neural Network Modeling of Batch Cell Growth Pattern," Ph.D. dissertation, Dept. of Chemical Engineering, Purdue University, West Lafayette, IN (1992).

Syu, M. J., and G. T. Tsao, "Neural Network Modeling of Batch Cell Growth Pattern," *Biotech. Bioeng.*, **42**, 376 (1993).

Taguchi, G., *Introduction to Quality Engineering*, Asian Productivity Organization, Tokyo, Japan and UNIPUB/Kraus International, White Plains, NY (1986).

Tang, Z., D. D. Almaida and P. A. Fishwick, "Time Series Forecasting Using Neural Networks vs. Box-Jenkins Methodology," *Simulation*, **57**, 303 (1991).

Temeng, K., D. Schnelle, H. Su and T. J. McAvoy, "Neural Model Predictive Control of an Industrial Packed-Bed Reactor," *AIChE Annual Meeting*, Miami, FL, Nov. (1992).

Thibault, J., "Feedforward Neural Networks for the Identification of Dynamic Process," *Chem. Eng. Commu.*, **105**, 109 (1992).

Thibault, J., V. V. Breusegem and A. Cheruy, "On-Line Prediction of Fermentation Variables Using Neural Networks," *Biotech. Bioeng.*, **36**, 1041 (1990).

Thompson, M. L. and M. A. Kramer, "Modeling Chemical Processes Using Prior Knowledge and Neural Networks," *AIChE J.*, **40**, 1328 (1994).

Tseng, A. Y. D., S. S. Jang, D. S. H. Wong and B. Joseph, "Predictive Control of Quality in Batch Polymerization Using Artificial Neural Network Models," *5th Intern. Symp. Process Systems Eng.*, Kyongju, Korea, May/June (1994).

TUTSIM Products, "TUTSIM (Twente University of Technology Simulation Program)," Palo Alto, CA (1994).

Ungar, L. H., "A Bioreactor Benchmark for Adaptive Network-Based Process Control," pp. 387-402, in *Neural Networks for Control*, W. T. Miller, III, R. S. Sutton and P. J. Werbos, editors, MIT Press, Cambridge, MA (1990).

Ungar, L. H., B. A. Powell and S. N. Kamens, "Adaptive Networks for Fault Diagnosis and Process Control," *Comput. Chem. Eng.*, **14**, 561 (1990).

Ungar, L. H., T. Johnson and R. D. DeVeaux, "Radial Basis Functions for Process Control," *Proceedings of Computer-Integrated Manufacturing in Process Industries*, New Brunswick, NJ (1994).

Werbos, P. J., "Generalization of Backpropagation with Application to a Recurrent Gas Market Model," *Neural Networks*, **1**, 339 (1988).

Werbos, P. J., "A Menu of Designs for Reinforcement Learning Over Time," pp. 67 to 96, in *Neural Networks for Control*, W. T. Miller, III, R. S. Sutton and P. J. Werbos, editors, MIT Press, Cambridge, MA (1990a).

Werbos, P. J., "Overview of Designs and Capabilities," pp. 59-66, in *Neural Networks for Control*, W. T. Miller, III, R. S. Sutton and P. J. Werbos, editors, MIT Press, Cambridge, MA (1990b).

Williams, R. J., "Adaptive State Representation and Estimation Using Recurrent Connectionist Networks," pp. 97-114, in *Neural Networks for Control*, W. T. Miller, III, R. S. Sutton and P. J. Werbos, editors, MIT Press, Cambridge, MA (1990).

Willis, M. J. and G. A. Montague, "Auto-Tuning PI(D) Controllers with Artificial Neural Networks," *Proceedings IFAC World Congress*, Vol.4, p.61, Sydney, Australia (1993).

Willis, M. J., G. A. Montague, A. J. Morris and M. T. Tham, "Artificial Neural Networks: A Panacea to Modeling Problems?," *Proceedings Amer. Control Conf.*, p. 2337, San Diego, CA, June (1990).

Yao, S. C. and E. Zafiriou, "Control System Sensor Failure Detection via Networks of Localized Receptive Fields," *Proceedings Amer. Control Conf.*, p. 2472, San Diego, CA, May (1990).

Ydstie, B. E., "Forecasting and Control Using Adaptive Connectionist Networks," *Comput. Chem. Eng.*, **14**, 583 (1990).

Ye, K., S. Jin, and K. Shimizu, "Fuzzy Neural Networks for the Control of High-Cell-Density Cell Cultivation of Recombinant Escherichia Coli," *J. Ferm. Bioeng.*, 77, 663 (1994).

Ye, N., T. J. McAvoy, K. A. Kosanovich and M. J. Piovoso, "Plantwide Control Using an Inferential Approach," *Proceedings Amer. Control Conf.*, p. 1900, San Francisco, CA, June (1993).

Ye, N., T. J. McAvoy, K. A. Kosanovich and M. J. Piovoso, "Optimal Averaging Level Control for the Tennessee Eastman Problem," *AIChE Annual Meeting*, St. Louis, MO, Nov. (1993).

Ye, N., T. J. McAvoy, K. A. Kosanovich and M. J. Piovoso, "An Inferential Approach Using Neural Network Models for Controlling Chemical Processes," *World Congress on Neural Networks*, San Diego, CA, June (1994).

Ye, X. and N. K. Loh, "Dynamic System Identification Using Recurrent Radial-Basis-Function Network," *Proceedings Amer. Control Conf.*, p. 2912, San Francisco, CA, June (1993).

You, Y. and M. Nikolalou, "Dynamic Processing Modeling with Recurrent Neural Networks," *AIChE J.*, **39**, 1654 (1993).

Yuan, Z. R. and X. G. Guo, "Backpropagation Networks for the Inverse Control of Discrete-Time Nonlinear Plant," *Proceedings Amer. Control Conference*, p.2938, Baltimore, MD, June (1994).

Zhou, J. L. and A. L. Tits, "Nonmonotone Line Search for Min/Max Problems," *J. Opt. Theo. Appl.*, **76**, 455 (1993).

Chapter 6

Development of Expert Networks : A Hybrid System of Expert Systems and Neural Networks

6.1 Introduction to Expert Networks
- A. An Analogy
- B. Expert Systems
- C. Expert Networks

6.2 Illustrative Case Study : Bioseparation of Proteins in Aqueous Two-Phase Systems
- A. Overview of the Bioseparation of Proteins in Aqueous Two-Phase Systems
- B. Model Development
  - 1. Prediction of Protein Partitioning
    - a. Review of Related Research
    - b. Theoretical Models
      - b1. Flory-Huggins Model
      - b2. Ogston Model
    - c. Comparison of Neural Network Models to Other Theoretical Models
      - c1. Advantages and Limitations of Neural Network Models
      - c2. An Illustrative Example Showing Effects of Protein Properties
  - 2. Overview of the Expert-System Approach
- C. Bioseparation Predictor : Protein-Partitioning Network
  - 1. Partition Coefficients of Proteins in Aqueous Two-Phase Systems
  - 2. Architecture of the Protein-Partitioning Network
  - 3. Polymer-Solution Properties
    - a. Architecture of the Polymer-Solution Network
    - b. Training of the Polymer-Solution Network
    - c. Illustrative Predictions by the Polymer-Solution Network
  - 4. Protein Properties
    - a. Independent Variables and Architecture for the Protein-Property Network
    - b. Training of the Protein-Property Network
    - c. Illustrative Predictions by the Protein-Property Network
  - 5. Ion Properties

6. Protein-Partitioning Network
a. Training of the Protein-Partitioning Network
b. Illustrative Predictions by the Protein-Partitioning Network
D. Bioseparation Sequencer : Protein-Partitioning Expert System
1. Overview of Bioseparation and Process Optimizers
2. Development of the Bioseparation and Process Optimizers and an Illustrative Application
E. Conclusions
6.3 Chapter Summary
Nomenclature
Practice Problems
References and Further Reading

This chapter introduces the basic principles of a hybrid system, called an expert network, composed of an expert system and a neural network. It describes the application of an expert network to bioseparation process design and development, taking as an example the prediction of separation performance and the synthesis of process flowsheets for separating a multicomponent protein mixture into pure-component products by liquid-liquid extraction using a system of two aqueous polymer solutions.

6.1 Introduction to Expert Networks

A. An Analogy

In defining an expert network, let us consider an analogy using the human brain as a model for the processing architecture in artificial intelligence (Figure 6.1). The left hemisphere of

the brain is a sequential processor based on logical functions: rules, concepts and calculations. The right hemisphere is a parallel processor based on gestalt functions: images, pictures and controls. An expert system simulates the qualitative reasoning functions of the left brain, while a neural network simulates the quantitative modeling capabilities of the right brain.

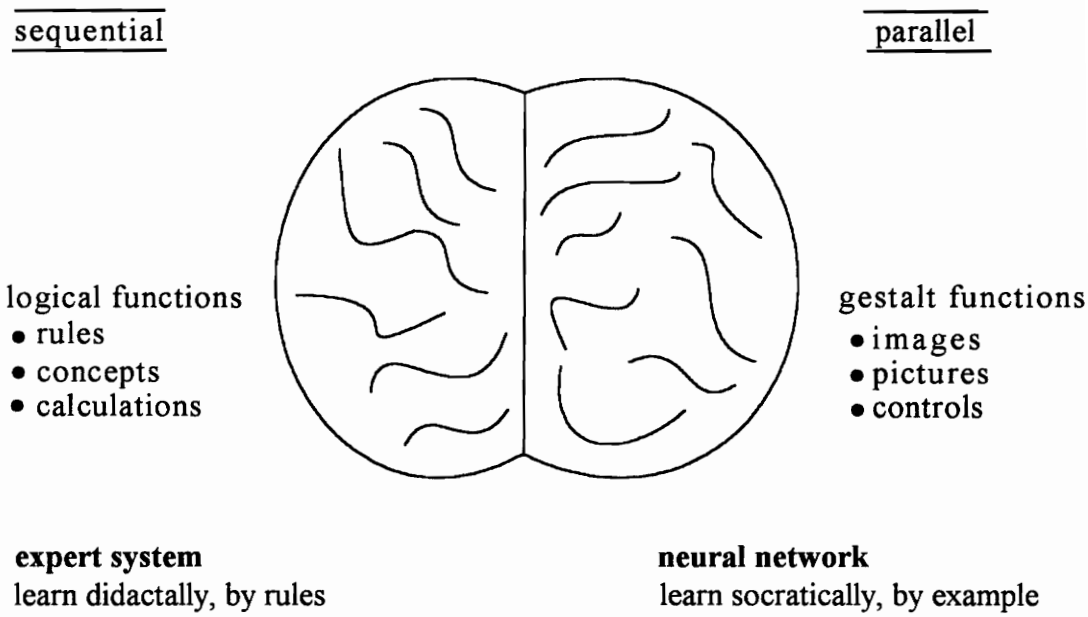


Figure 6.1. Left-brain versus right-brain processing (adapted with permission from *Expert System Strategy. Newsletters*, Vol. 3, No. 2, 1987 Harman Associates).

B. Expert Systems

Expert systems attempt to match the performance of human experts in a given field. To do so, these systems rely on in-depth, expert knowledge. The better the knowledge, the better the performance of the expert system.

Expert systems usually incorporate knowledge through relationships. The system keeps track of relations and the inferences it invokes. Therefore, the system's knowledge is explicit and accessible to the user. An expert system can explain why it needs certain information, and how it reaches certain conclusions.

Expert systems have two key advantages: (1) they can assimilate large amounts of knowledge; and (2) they never forget that knowledge. These properties distinguish expert systems from conventional computer programs. Ideally, an expert system can build its own knowledge base, although achieving that goal has been very challenging. Another goal is to enable field experts who are not programmers to expand a system's knowledge base.

A typical expert system , shown in Figure 6.2, contains: (1) a knowledge base; (2) an inference engine; and (3) a user interface. The knowledge base contains specific, in-depth information consisting of facts, rules, and heuristics about the problem at hand (Figure 6.3).

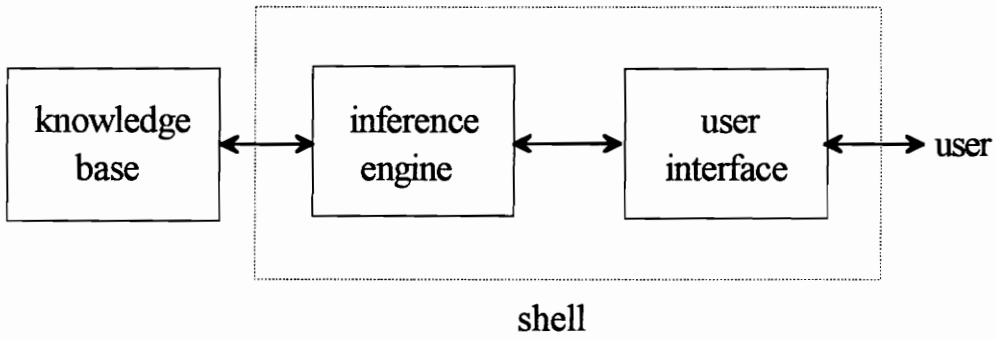


Figure 6.2. The structure of an expert system.

knowledge = facts + rules + heuristics

Figure 6.3. Knowledge in expert systems.

To utilize that knowledge, an expert system relies on its inference engine. This engine uses inference mechanisms to process the knowledge and draw conclusions. Finally, the user interface provides smooth communication between the program and the user.

An expert-system shell is simply the combination of the user interface and the inference engine, as seen in Figure 6.2. Ideally, the shell: (1) explains how the system reaches a conclusion; (2) explains why it needs certain information; and (3) adds information to the knowledge base.

Artificial Intelligence in Chemical Engineering (Quantrille and Liu, 1991) describes in detail the fundamentals and applications of expert systems. For more information, the reader may refer to the following sections of that text:

- Knowledge representation, Sections 10.2 and 11.2.
- Search or problem-solving strategies, Sections 11.1 and 11.3.
- Expert-system development, Sections 12.2 and 16.2.
- Expert-system applications in bioprocessing and chemical engineering, Sections 16.3 to 16.8.

C. Expert Networks

An expert network combines an expert system (described above) with a neural network. Although expert systems and neural networks are both important aspects of artificial intelligence, they have little else in common, as Table 6.1 reveals. Table 6.2 presents the key advantages and limitations of the two systems. When an expert system and a neural network join to form an expert network, each compensates for the other's weakness.

Figure 6.4 illustrates one approach to placing a neural network in series with an expert system. Data representing the independent variables of a given system serve as the input to a previously trained neural network. The neural network predicts the desired response variables, usually in numerical form. In order for the expert system to use the

Table 6.1. Neural networks versus expert systems (Samdani, 1990).

Neural networks	Expert systems
example-based	rule-based
domain-free	domain-specific
finds rules	needs rules
little programming	much programming
easy to maintain	difficult to maintain
fault-tolerant	not fault-tolerant
needs a database	needs a human expert
fuzzy logic	rigid logic
adaptive system	requires reprogramming

Table 6.2. Advantages and limitations of expert systems and neural networks.

	Expert systems	Neural networks
very effective	In applying a fixed set of facts, rules and heuristics (i.e., knowledge) to a domain-specific problem, typically involving only simple mathematics.	In organizing and detecting patterns from unpredictable and/or imprecise input data, in learning by examples, and in generalizing to new situations.
less efficient	In processing raw sensory data from the real, unpredictable world.	In providing in-depth solutions and full understanding of each problem as well as the reasoning behind its solution (i.e., no explanation capability).

information, the expert network must convert the numerical response values into semi-quantitative (fuzzy) categorical data (i.e., 0 = poor; 1 = average; or 2 = good) using a *classifier* . Occasionally, we can eliminate the classifier by using certain types of probabilistic classification networks (NeuralWare, 1993), that produce output already in the format required by the expert system. In either case, the fuzzy data from the neural network or classifier is then subjected to the facts, rules, and heuristics of the expert system to identify optimal operating conditions.

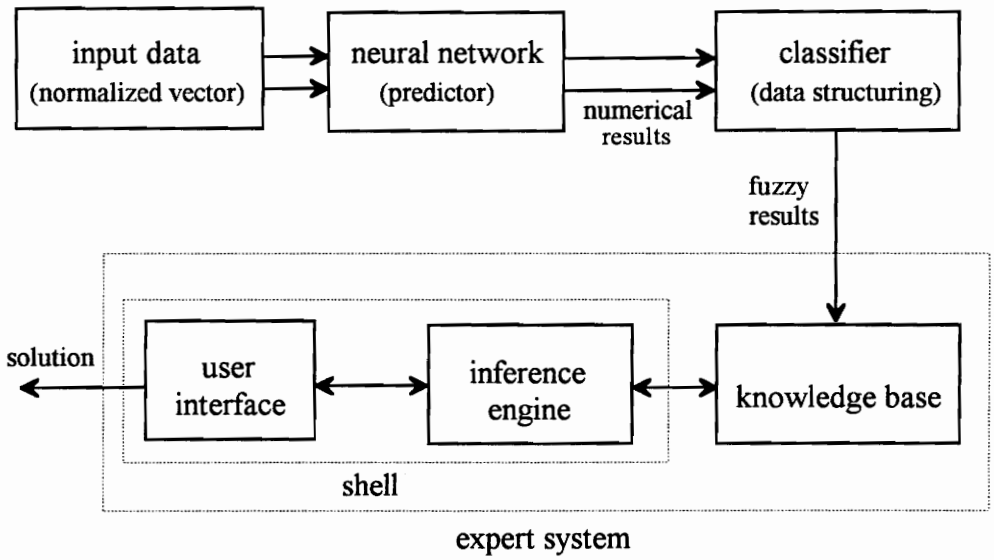


Figure 6.4. An illustration of an expert network.

For further discussion of the principles of expert networks, refer to the articles by Hendler (1989a,b), the book chapter by Caudill (1990), and the monograph by Gallant (1993). Hall (1992) summarizes a number of significant industrial applications for expert networks, including the *Intelligent Arc Furnace (IAF) Controller* (see Section 5.6.B.2). This is a hybrid system consisting of neural network models and with an expert-system component for optimizing electrode positions for more accurate heat distribution in scrap metal furnaces.

In bioprocessing and chemical engineering, applications of expert networks are being proposed and demonstrated, including an activated-sludge troubleshooting guide in wastewater treatment (Ferrada et al., 1990), continuous steel-casting diagnosis (Bulsari and Saxen, 1992a), control of fermentation processes (Gehlen et al., 1992), a chemical reactor-selection guide (Bulsari and Saxen, 1992b; Bulsari et al., 1992), PID controller tuning (Scott et al., 1992), integrated process design and control of heat-exchanger networks (Huang and Fan, 1994), process-fault diagnosis (Beacraft and Lee, 1993), integrated process monitoring, diagnosis, and control (Renegaswamy and Venkatsubramanian, 1992), and integrated process operation management (Venkatsubramanian and Kavuri, 1992; Venkatsubramanian and Stanley, 1994).

The next two sections describes the application of an expert network similar to Figure 6.4 to bioseparation process design and development.

6.2 Illustrative Case Study : Bioseparation of Proteins in Aqueous Two-Phase Systems

A. Overview of the Bioseparation of Proteins in Aqueous Two-Phase Systems

This case study describes a flexible approach to the predictive modeling and optimal design of extractive bioseparations in aqueous two-phase systems (ATPS). Our focus is protein purification, a major process in the sector of the biochemical industry whose development will benefit from a large increase in capacity. Table 6.3 compares several key commercial and developing methods for large-scale purification of proteins (Asenjo and Patrick, 1990). Liquid-liquid extraction, using ATPS, shows promise as a cost-effective, large-scale process that can achieve high selectivity and purity requirements (Michaels, 1992). One of the most well-known applications involves the protein partitioning in an ATPS with two polymers, polyethylene glycol (PEG) and Dextran, as illustrated in Figure 6.5 (Clark, 1989). Here, we represent the partition behavior of a protein by a *partition coefficient,* K_p, defined as *the ratio of the protein concentration in the PEG-rich top phase to that in the Dextran-rich bottom phase*.

Table 6.3. Key commercial and developing methods for purification of proteins (Asenjo and Patrick, 1990).

Method	Separation property	Capacity	Selectivity	Yield
Adsorption	Van der Waals forces h-bonds, polarities, dipole moments	medium to high	medium to high	medium to high
Ion-exchange chromatography	charge	medium	medium	medium
Hydrophobic-interaction chromatography	surface hydrophobicity	medium	medium	medium
Dye affinity chromatography	biological affinity	medium	high	medium
Ligand affinity chromatography	isoelectric point	medium low	very high	high
Chromatofocusing	bioactivity	low	medium	high
Gel permeation chromatography	molecular size	very low	low	high
Reverse phase liquid chromatography	hydrophilic and hydrophobic interactions	medium	high	medium
pH precipitation	charge	high	very low	medium
Ammonia/sulfate precipitation	hydrophobicity	high	very low	high
Aqueous two-phase extraction (ATPS)	phase properties	high	low	high
ATPS with biospecific ligand	phase properties and bioaffinity	high	high	high

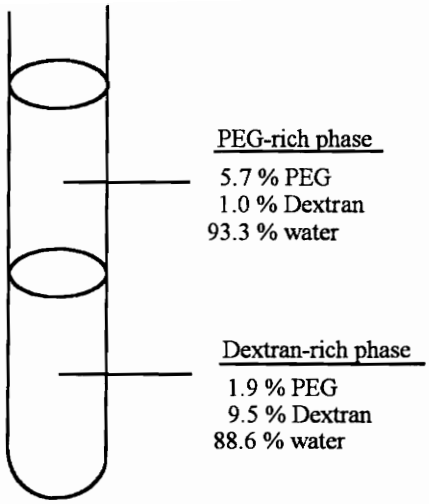


Figure 6.5. An aqueous two-phase system (ATPS) involving polyethylene glycol (PEG) and Dextran (Clark, 1989).

Dr. Alan S. Michaels, a well-known expert in biotechnology, has suggested that an important research need in protein partitioning in ATPS is the development of a predictive rationale for selecting the appropriate polymer systems (Michaels, 1992):

> "This process (i.e., two-phase aqueous extraction), which employs either aqueous mixtures of simple electrolytes and water-soluble polymers (e.g., PEG), or of incompatible water-soluble polymers (e.g., Dextran/PEG) to form two immiscible water-rich phases, has been shown to be quite effective in facilitating the separation of cells or cellular debris from soluble proteins in fermentation broths, and permitting recovery of enzymes and other useful biomacromolecular products from such mixtures. *Unfortunately, however, efforts to develop a predictive rationale for selection of appropriate multiphase-forming polymer systems useful for specific protein recovery requirements have yet to be developed.*" (italics added)

In a recent review, King (1992) states that continued research in predictive design correlations and development of novel applications and polymers will accelerate the adoption of ATPS in the biotechnology industry.

This case study represents a novel application of artificial intelligence to bioseparation process synthesis. We demonstrate how to combine the qualitative modeling skills of an expert system with the quantitative modeling capabilities of a neural network to develop an expert network for bioseparation process synthesis, particularly protein partitioning in ATPS.

Specifically, our objective is to develop an expert network for protein partitioning in ATPS that is capable of:

(1) predicting the partition coefficients (i.e., separation factors) over wide ranges of protein properties, ion properties, and polymer-solution properties;

(2) identifying the optimal separation conditions for multicomponent protein solutions to assist in experimental design and process development.

B. Model Development

The first step in developing a neural network model for a complex bioprocessing system such as protein partitioning in ATPS is to do a very extensive literature search and study all relevant theoretical models.

1. Prediction of Protein Partitioning

a. Review of Related Research

There is a large body of published literature on protein partitioning in ATPS. See, for example: review monographs (Albertsson, 1986; Walter et al., 1985); review papers (Baskir and Hatton, 1989; King, 1992); conference proceedings (Fisher and Sutherland., 1989); fundamental studies (Baskir et al., 1987; Cabezas et al., 1992; Connemann et al., 1992; Diamond, 1990; Forciniti, 1991; Haynes et al., 1993; King et al., 1988; Mahaderan and Hall, 1990, 1992; Vlachy et al., 1993); tutorial articles (Clark, 1989); recent doctoral dissertations (Diamond, 1990; Forciniti, 1991; Haynes, 1992); affinity partitioning (Ichikawa et al., 1992; Johansson and Tjerneld, 1989; Kamihira et al., 1992; Tjerneld et al., 1987); large-scale applications (Hustedt et al., 1989; Kula et al., 1982); and technology assessments (Fair, 1989; Michaels, 1992). In addition, a two-day symposium on aqueous biphasic separations at the American Chemical Society National Meeting in San Diego, CA, in April, 1994 included 35 papers from biomolecules to metal ions.

Recent fundamental research has led to four major classes of theoretical models for protein partitioning in ATPS:

- Adsorption models based on thermodynamics of adsorption of polymers from dilute solution to spherical solid particles (Baskir et al., 1987; Baskir and Hatton, 1989).

- Ogston-based models utilizing the osmotic virial equation for thermodynamics of dilute aqueous mixtures (e.g., Forciniti and Hall 1990; King et al., 1988).
- Flory-Huggins-based models applying the Flory-Huggins equation of state for polymer-solution thermodynamics (e.g., Diamond, 1990; Diamond and Hsu, 1990).
- Perturbation models based on statistical-mechanical calculations considering the protein shapes (e.g., Haynes et al., 1993; Mahaderan and Hall, 1990, 1992).

b. Theoretical Models for the Prediction of Protein Partitioning in Aqueous Two-Phase Systems.

Within this section, we present the development of the two main theoretical models used for ATPS. The primary purpose of going into such detail in the model development is to: (1) illustrate the complexity of the statistical thermodynamic models presently being used; (2) show some of the limitations associated with these types of models for experimental design and process development; (3) indicate the experimental requirements that must accompany the use of these models; and (4) identify the primary variables that should be included in the neural network architecture.

The ATPS used in the liquid-liquid extraction of proteins normally contains water, two immiscible water-soluble polymers, a salt, and proteins. The polymer concentrations for the ATPS are in the semidilute regime, 10 to 25 weight percent of polymer (Figure 6.6). Several statistical thermodynamic methods have been developed, in an attempt to

accurately predict the phase behavior of aqueous systems (reviewed by Benge, 1986). Currently, the two most popular theories used in the biochemical field are the Ogston model (King, 1988; Forciniti, 1991) and the Flory-Huggins theory (Diamond, 1990; Baskir et al., 1987).

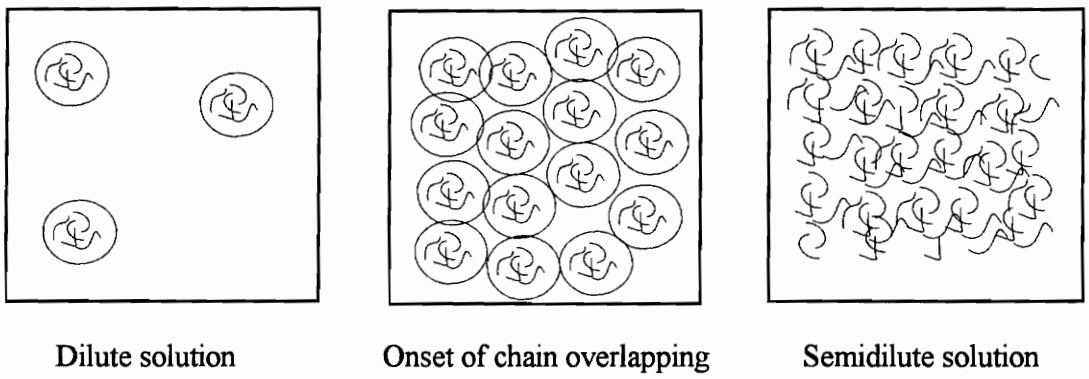


Figure 6.6. Dilute and semidilute solutions.

Both models are effective in obtaining qualitative results and predicting general trends for the aqueous system. The major problems occur with quantitative support in their development. The Flory-Huggins model has been very effective in modeling multicomponent polymer systems with organic solvents. But using an aqueous system causes deviations because of the problems associated with assuming that the free volume occupied by one unit volume of water is equivalent to that occupied by one monomer unit. The Ogston theory, on the other hand, assumes a dilute solution (Figure 6.6) inconsistent

with the semidilute region. Moreover, neither model accounts for the molecular weight distributions of Dextran. Yet, a recent study by Conneman et al. (1992) concludes that the polydispersity index (PI) *must* be included in any model that wants to accurately predict the partitioned phase behavior in ATPS. The PI is simply the ratio of the weight-average molecular weight M_w to the number-average molecular weight M_n.

b1. Flory-Huggins Theory

The Flory-Huggins theory is based on the liquid-lattice model (Figure 6.7). It assumes that the polymer and solvent molecules will arrange themselves randomly within a lattice structure, each occupying one lattice position. Each lattice is set at the free volume occupied by one monomer segment of the polymer. This theory assumes that the free volume of the solvent is equivalent to that of one monomer unit.

Flory and Huggins used the liquid-lattice model in conjunction with a statistical thermodynamic approach to calculate the configurational entropy of mixing and the enthalpy of mixing, as described in Flory (1953). The entropy of mixing, ΔS_{mix}, for a three-component mixture is:

$$\Delta S_{mix} = -R(n_1 \ln \phi_1 + n_2 \ln \phi_2 + n_s \ln \phi_s) \qquad (6.1)$$

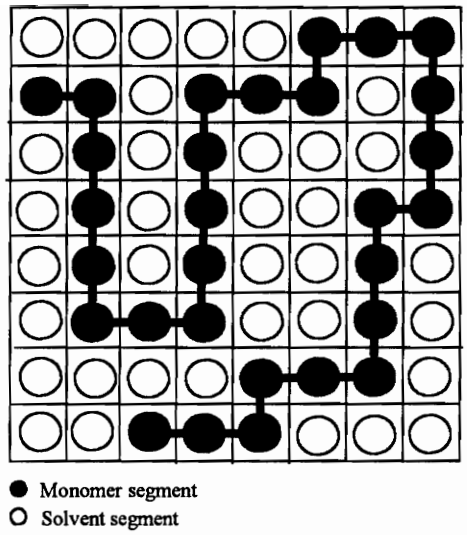


Figure 6.7. The liquid-lattice model used for the derivation of the Flory-Huggins model.

where n_i is the number of moles and ϕ_i is the volume fraction. Subscripts 1, 2, and s represent polymer 1, polymer 2, and solvent, respectively.

The enthalpy of mixing, ΔH_{mix}, is:

$$\Delta H_{mix} = RT\,(\chi_{12}\phi_1\phi_2+\chi_{1s}\phi_1\phi_s+\chi_{2s}\phi_2\phi_s)\,(m_1n_1+m_2n_2+m_sn_s) \qquad (6.2)$$

where χ_{ij} is the Flory interaction parameter and m_i is the ratio of the molar volume of the ith component to that of the reference component, normally taken as the solvent.

The Gibbs free energy of mixing, ΔG_{mix}, is related to the enthalpy and entropy of mixing by:

$$\Delta G_{mix} = \Delta H_{mix} - T \Delta S_{mix} \tag{6.3}$$

or,

$$\frac{\Delta G_{mix}}{RT} = (n_1 \ln\phi_1 + n_2 \ln\phi_2 + n_s \ln\phi_s) + (\chi_{12}\,\phi_1\phi_2 + \chi_{1s}\phi_1\phi_s + \chi_{2s}\phi_2\phi_s)\,(m_1 n_1 + m_2 n_2 + m_s n_s) \tag{6.4}$$

The chemical potential, μ_i, is related to the Gibbs free energy of mixing:

$$\Delta \frac{\mu_i}{RT} = \left(\frac{dG_{mix}}{dn_i}\right)_{T,P,n} \tag{6.5}$$

or,

$$\Delta \frac{\mu_1}{RT} = \ln \phi_1 + 1.0 - m_1 \left\{ \left(\frac{\phi_1}{m_1} + \frac{\phi_2}{m_2} + \frac{\phi_s}{m_s} \right) - (\chi_{12}\phi_2 + \chi_{1s}\phi_s) + (\chi_{12}\phi_1\phi_2 + \chi_{1s}\phi_1\phi_s + \chi_{2s}\phi_2\phi_s) \right\}$$

$$\Delta \frac{\mu_2}{RT} = \ln \phi_2 + 1.0 - m_2 \left\{ \left(\frac{\phi_1}{m_1} + \frac{\phi_2}{m_2} + \frac{\phi_s}{m_s} \right) - (\chi_{12}\phi_1 + \chi_{2s}\phi_s) + (\chi_{12}\phi_1\phi_2 + \chi_{1s}\phi_1\phi_s + \chi_{2s}\phi_2\phi_s) \right\} \tag{6.6}$$

$$\Delta \frac{\mu_3}{RT} = \ln \phi_3 + 1.0 - m_3 \left\{ \left(\frac{\phi_1}{m_1} + \frac{\phi_2}{m_2} + \frac{\phi_s}{m_s} \right) - (\chi_{1s}\phi_1 + \chi_{2s}\phi_2) + (\chi_{12}\phi_1\phi_2 + \chi_{1s}\phi_1\phi_s + \chi_{2s}\phi_2\phi_s) \right\}$$

Equilibrium in a two-phase system occurs when the chemical potential of each component is equal in both phases:

$$\Delta\ \mu_1'(\phi_1,\phi_2,\phi_s) = \Delta\ \mu_1''(\phi_1,\phi_2,\phi_s)$$

$$\Delta\ \mu_2'(\phi_1,\phi_2,\phi_s) = \Delta\ \mu_2''(\phi_1,\phi_2,\phi_s) \qquad (6.7)$$

$$\Delta\ \mu_s'(\phi_1,\phi_2,\phi_s) = \Delta\ \mu_s''(\phi_1,\phi_2,\phi_s)$$

where a single prime denotes phase 1 (bottom phase) and a double prime denotes phase 2 (top phase).

We can then use iterative numerical routines for the solution of multiple nonlinear equations to solve for the compositions of the two phases. The chemical potential relationships (6.6 and 6.7) consist of three equations and six unknowns: three volume fractions in each phase. We can reduce the six unknowns to three by using the two material-balance relationships and randomly selecting one composition.

The partition coefficient for the protein comes from expanding the system to four components with water and salt being treated as a single component (Diamond, 1990). Finally, the effect of the salt is taken into consideration using an electrostatic potential term (King, 1988).

$$\ln(K_p) = A^*\left(w'' - w'\right) + b\left(w'' - w'\right)^2 + \frac{z_b F \Delta\Psi}{RT} \qquad (6.8)$$

where

$$A^* = m_p\left\{\frac{\alpha_1}{m_1} + \frac{\alpha_2\phi}{m_2} + (\alpha_1 + \alpha_2\phi)(\chi_{1s}-1) - \alpha_1\chi_{1p} + \alpha_2\chi_{2p} + \alpha_1\chi_{1s} + \alpha_2\chi_{2s}\right\} \qquad (6.9)$$

and

$$b = m_p\left(\chi_{2s}\alpha_2^2\phi^2 - \chi_{1s}\alpha_1^2\right) \tag{6.10}$$

The variables within equations 6.8 to 6.10 are defined as:

w_i = weight fraction of species i

m_i = molar-volume ratio of species i to the reference species

χ_{ij} = Flory-Huggins interaction parameter between species i and j

$\Delta\psi$ = electrostatic potential difference between the phases

F = Faraday constant

z_b = net charge of the species

α_i = proportionality factor between volume- and weight-fraction differences between the phases for species i

ϕ = ratio of weight-fraction difference between phases for Dextran and PEG

The Flory-Huggins theory neglects a few main items. It assumes that the entropy of mixing, ΔS_{mix}, is a function of only the configurational entropy, with negligible residual contribution. But the residual entropy is affected by differences in free volume and intermolecular forces between the two components. Many molecules have preferred orientations stemming from attractive forces, and thus deviate from the assumption of a configurational entropy for random orientations.

b2. Ogston Model

Ogston (1962) proposed another approach to modeling ATPS based on the osmotic virial equation for the thermodynamics of dilute aqueous mixtures. This model is derived from the Gibbs-Duhem equation:

$$SdT - VdP + \Sigma\, n_i d\mu_i = 0 \tag{6.11}$$

where S is entropy, T is temperature, P is pressure, n_i is the number of moles of component i, and μ_i is the chemical potential of component i. At constant temperature and pressure, this equation reduces to:

$$\Sigma\, n_i d\mu_i = 0 \tag{6.12}$$

For a three-component system, where the solvent water (s) has a constant number of molecules, the chemical potentials for the two polymers (1 and 2) and the protein (p) are:

$$\begin{aligned}
\frac{\mu_1-\mu_1^0}{RT} &= \ln m_1 + C_{11}m_1 + C_{12}m_2 + C_{1p}m_p \\
\frac{\mu_2-\mu_2^0}{RT} &= \ln m_2 + C_{22}m_2 + C_{12}m_1 + C_{2p}m_p \\
\frac{\mu_p-\mu_p^0}{RT} &= \ln m_p + C_{pp}m_p + C_{1p}m_1 + C_{2p}m_2
\end{aligned} \tag{6.13}$$

where m_i is the dimensionless molality and C_{ij} is the virial coefficient.

We can determine the chemical potential of water by combining these three expressions with the Gibbs-Duhem equation.

$$\frac{\mu_s-\mu_s^0}{RT} = -\left\{m_1+m_2+m_p+\frac{C_{11}m_1^2}{2}+\frac{C_{22}m_2^2}{2}+\frac{C_{pp}m_p^2}{2}+C_{12}m_1m_2+C_{1p}m_1m_p+C_{2p}m_2m_p\right\} \quad (6.14)$$

Finally, we can calculate the partition coefficient, K_p, by equating the chemical potentials of the protein in both phases and including an electrostatic potential term to account for salt contributions.

$$\ln m_p' + C_{pp}m_p' + C_{1p}m_1' + C_{2p}m_2' + z_pF\varphi' = \ln m_p'' + C_{pp}m_p'' + C_{1p}m_1'' + C_{2p}m_2'' + z_pF\varphi'' \quad (6.15)$$

where z_p is the net charge of the protein, F is Faraday constant, ϕ is the electrostatic potential, a single prime denotes phase 1 (bottom phase), and a double prime denotes phase 2 (top phase).

The partition coefficient then becomes

$$\ln K_p = \ln\left(\frac{m_p'}{m_p''}\right) = C_{1p}\left(m_1''+m_1'\right)+C_{2p}\left(m_2''+m_2'\right)+\frac{zF\left(\phi''-\phi'\right)}{RT} \quad (6.16)$$

Working with the Dextran/PEG system, Forciniti and Hall (1990) developed an extension to the derivation by Ogston (1962) that expresses the partition coefficient in terms of the gradient of the two polymers between phases:

$$\ln K_p = M_p\left[\frac{1}{M_p}\ln\left(\frac{w_s''}{w_s'}\right) - 2C_{1p}\Delta PEG + 2C_{2p}\Delta Dextran\right] \tag{6.17}$$

where PEG is polymer 1, Dextran is polymer 2, protein is p, and water is s. The gradients for the two polymers are:

$$\Delta PEG = \left(\frac{w_1}{w_s}\right)' - \left(\frac{w_1}{w_s}\right)'' \tag{6.18}$$

$$\Delta\ DEXTRAN = \left(\frac{w_2}{w_s}\right)'' - \left(\frac{w_2}{w_s}\right)' \tag{6.19}$$

where w_i is the weight fraction of component i. The virial coefficients are defined by the shapes of the protein and polymer.

$$C_{ip} = \frac{N_a}{2M_iM_p}\left[\frac{4}{3}\pi(R_p + R_i)^3\right] \qquad \text{sphere-sphere} \tag{6.20}$$

$$C_{ip} = \frac{N_a}{2M_iM_p}\left[\frac{4}{3}\pi R_p^3 + \pi R_i^3 L\right] \qquad \text{sphere-cylinder} \tag{6.21}$$

Haynes et al. (1989, 1993) have also developed an extension to the Ogston model, which includes many additional electrostatic effects. To illustrate the complexity of these theoretical models and to understand how the interaction parameters are obtained, we will examine their first model, which combines the Guggenheim, Ogston, and Hill theories. This model is similar to the previously described Ogston-based models, but is more detailed. The model uses a similar methodology to estimate the interaction parameters, but includes a number of additional factors. Thus, Gibbs free-energy term in the model by Haynes et al. is expressed as :

$$\frac{G}{RT} = \frac{n_0\mu_0^0}{RT} + \sum_{j\neq 0}^{n} n_j\left[\ln\left(m_j\lambda_j^0\right) - 1\right] - \frac{2}{3}\alpha\sqrt{I}\,\tau\left(Hd\sqrt{I}\right)\sum_{j=0}^{n} z_j^2 n_j + \frac{1}{2}[\beta_{11}n_1m_1+\beta_{22}n_2m_2+\beta_{pp}n_pm_p] \quad (6.22)$$

$$+\beta_{12}n_1m_2+\beta_{1a}n_1m_a+\beta_{1c}n_1m_c+\beta_{1p}n_1m_p+\beta_{2a}n_2m_a+\beta_{2c}n_2m_c+\beta_{2p}n_2m_p+\beta_{ac}n_am_c+\beta_{ps}n_pm_s$$

where

$$\tau(x) = \frac{3}{x^3}\left[\ln(1+x) - x + \frac{1}{2}x^2\right] \quad (6.23)$$

and

$$\beta_{ps} = \beta_{pa}v_a + \beta_{pc}v_c \quad (6.24)$$

The variables within equation 6.22 to 6.24 are defined as:

n_0	=	number of moles of the solvent or water in the solution.
μ_0^0	=	the chemical potential of pure water.
m_j	=	the molality of jth component.
λ_j^0	=	the standard-state activity of the jth component as defined by Guggenheim.
α	=	the Debye-Huckel constant (emperical statistical-mechanical correlation for ion activity coefficients that depend on solvent and temperature).
H	=	the Debye-Huckel parameter (emperical statistical-mechanical correlation for ion activity coefficients that depend on solvent and temperature).
d	=	the mean diameter of the ions.
I	=	the ionic strength of the solution.
z_i	=	the charge of ion i.
v_i	=	the valence of ion i.
β_{ij}	=	the binary specific-interaction coefficient between the solute molecules, i and j.

This model requires sets of specific-interaction parameters, obtained as follows:

(1) *Polymer-polymer specific-interaction coefficient* : low-angle light scattering data.

(2) *Cation-anion specific-interaction coefficient* : fit Guggenheim's chemical-potential expression to water-to-water activity data for the given electrolyte solution.

(3) *Protein-Protein specific-interaction coefficient* : osmotic-pressure data for dilute aqueous solutions of protein.

(4) *Ion-polymer specific-interaction coefficient* : fit the Gibbs free-energy equation to osmotic-pressure data to ternary aqueous solutions of polymer and the salt of the ion.

(5) *Protein-salt specific-interaction coefficient* : fit the Gibbs free-energy equation to salting-out data to the protein.

(6) *Protein-polymer specific-interaction coefficient* : fit the Gibbs free-energy equation to solubility data to the protein in an aqueous solution with the specified polymer.

As these procedures indicate, predicting a protein's partitioning behavior involves a considerable amount of experimentation. As previously noted, there are thousands of proteins, and each must be tested over wide ranges of polymer-solution and ion properties before the partitioning behavior can be predicted effectively. Until a tool is developed, such as three-dimensional solubility parameters used in more generic polymer-solvent systems, the experimental aspects of the predictions cannot be avoided.

c. Comparison of Neural Network Models to Other Theoretical Models

Given the complexity and experimental requirements of these two models, we turn to a neural network model of the Dextran/PEG ATPS. In the next two sections, we will give some of the advantages and limitations of neural network models and then present an illustrative example showing some problems associated with the use of the theoretical models.

c1. Advantages and Limitations of Neural Network Models

Neural networks should be used primarily to complement theoretical models in creating an effective strategy to facilitate experimental design and process development. Theoretical models describe the behavioral patterns of a given system and show the basic trends that system will follow. In comparison, neural network models store the behavioral pattern within interior structures hidden to the user. Consequently, the user knows little about the solution's validity. However, the neural network can recognize more complex response patterns than corresponding empirical models can, especially when large set of independent variables influences the solution. The advantages and limitations of neural computing for protein partitioning in particular are as follows:

• Advantages :

(1) The neural network model recognizes patterned behavior without regard to mathematical complexity.

(2) The model allows for the inclusion of an expanded set of input variables, for example, the use of the polydispersity index for the polymers (Connemann et al., 1992) and the use of the hydrophobicity term for the protein (Albertsson, 1986), both of which are thought to have a significant impact on the protein partitioning.

(3) The model uses the protein properties predicted by amino acid compositions and protein molecular weights in conjunction with polymer-solution and ion properties to account for all binary interactions. This approach eliminates the tedious experimental measurements and data correlations required by most theoretical models to estimate the following interaction parameters: (a) protein-polymer (e.g., by fitting the Gibbs free-energy equation to solubility data for the protein in an aqueous solution with the specified polymer); (b) polymer-polymer (e.g., by using low-angle light scattering data); and (c) ion-polymer (e.g., by fitting the Gibbs free-energy equation to osmotic-pressure data for ternary aqueous solutions of polymer and the salt of the ion) (Haynes et al., 1989).

(4) The model can readily incorporate new sets of experimental data that do not follow historically predicted system behavior, thus improving the future prediction capabilities.

(5) Compartmental neural network structures easily incorporate new independent variables. For example, when the prediction of the tertiary structure for a protein becomes available, the additional input variables can be inserted into the network structure with minimal effort.

(6) The network is well suited to handling noisy, incomplete, or inconsistent data. Because each processing element affects the input-output pattern only slightly, the network filters data easily. Moreover, since each processing element operates relatively independently, the network has significant parallel processing capabilities (Quantrille and Liu, 1991, pp. 442-445).

- Limitations :

(1) The neural network model has no theoretical foundation. Therefore, the model does not provide the explicit reasoning behind its solution, and we may need insights from theoretical models to support and validate the neural network solution.

(2) Building an effective neural network model requires a broad set of training examples. Because predictions outside the boundaries defined by training

examples are often poor estimations, the training database must be diverse, complete, and well-structured.

As stated above, most theoretical models used to predict protein partitioning are complex and require adjustable interaction parameters (protein-polymer, polymer-polymer, and ion-polymer interaction parameters), to adequately describe the system. These parameters depend on many polymer-solution, protein, and ion properties. The neural network approach provides an effective and flexible means to incorporate all the properties that significantly influence the interaction parameters (see Figure 6.8). Therefore, it can predict protein partitioning between the two aqueous phases without requiring considerable experimentation and curve-fitting to obtain the interaction parameters.

We stress again that a major limitation of neural computing is the large and diverse set of examples needed to train the network. This need is especially difficult for the complex problem of protein partitioning in ATPS, where there are thousands of proteins varying greatly in functions and properties, and five other key independent variables (Mw_{PEG}, $Mw_{Dextran}$, %PEG, %Dextran, and pH). As Section 6.2.C.5 shows, we use over 1000 experimental systems to train a network, for a study that only includes 5 proteins, each with extensive data available on its partitioning behavior. Figure 6.9 shows a typical learning curve for a neural network model, with the amount of training information presently available and the threshold point where the network becomes effective for

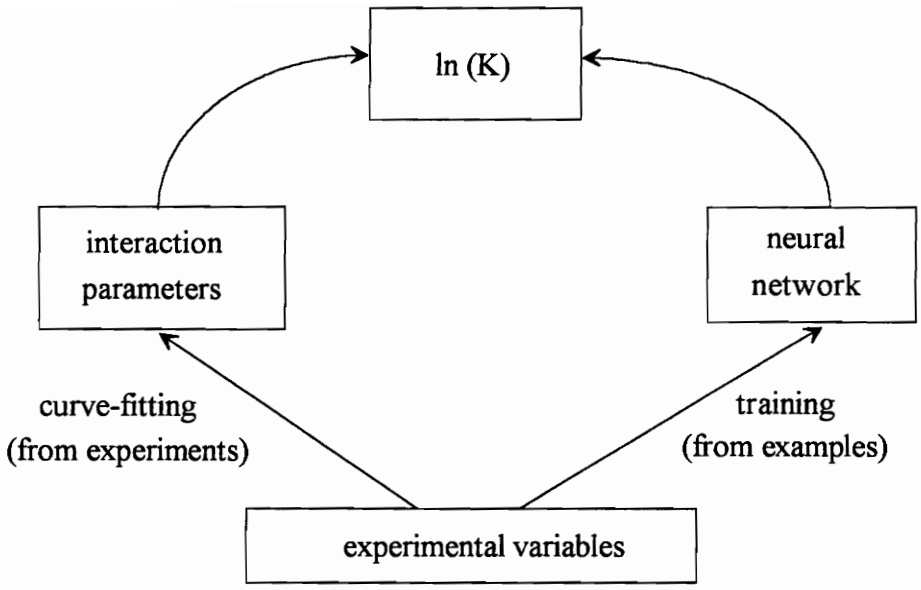


Figure 6.8. A comparison of theoretical thermodynamic models based on interaction parameters to a neural network model for the prediction of partition coefficients.

pattern recognition marked. Early on the learning curve, the network is only capable of fuzzy inferences about the protein partitioning. However, those fuzzy inferences are still effective in facilitating experimental design and process development. As the experimental data grow, the neural network predictor will become much more reliable and effective. Considering that there are thousands of proteins, this type of model can help direct experimental design and process development while all the protein interactions are being quantified.

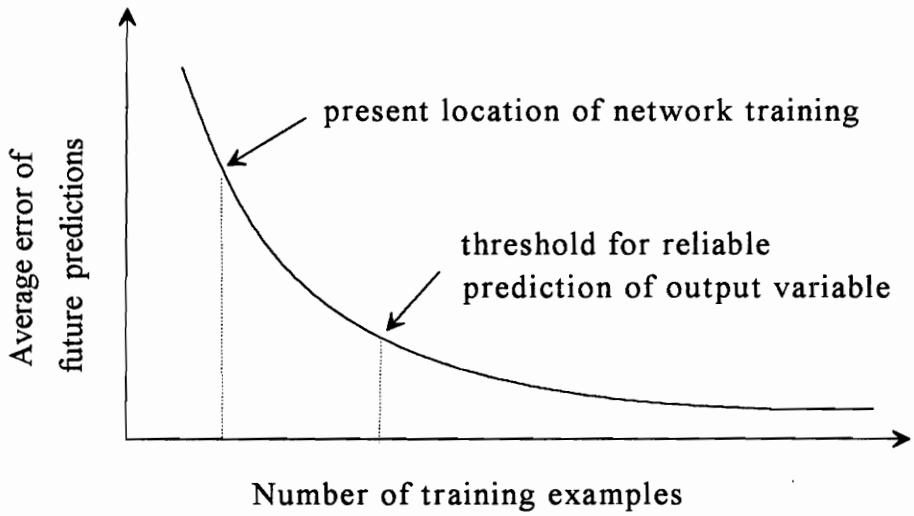


Figure 6.9. A learning curve for the neural network model.

c2. <u>An Illustrative Example Showing Effects of Protein Properties</u>

An example provided by Forciniti and Hall (1991) demonstrates how protein properties can significantly affect partitioning behavior in ATPS, and reveals some of the limitations of most theoretical models. In this case, we consider the partition coefficients for five proteins as a function of the tie-line length (Figure 6.10).

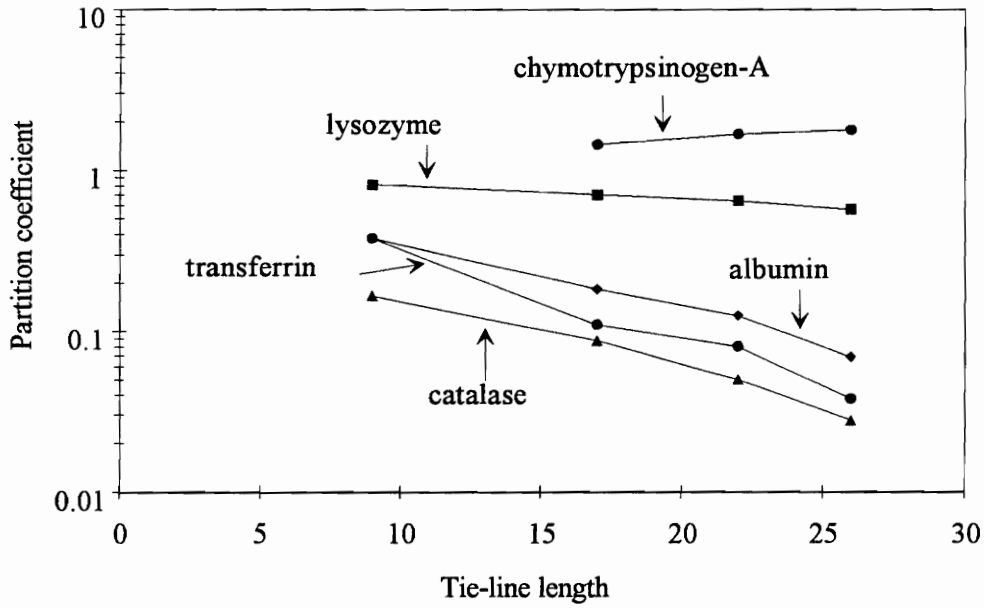


Figure 6.10. A comparison of partition coefficients for five proteins as a function of tie-line length (Forciniti, 1991).

The tie-line length is defined as:

$$\text{tie-line length} = \sqrt{(w_1'' - w_1')^2 + (w_2'' - w_2')^2} \tag{6.24}$$

w_i is the weight fraction of the ith component in a given phase. In Figure 6.11, the tie-line length is the length of the dashed line between the Dextran-rich phase (bottom) and the PEG-rich phase (top).

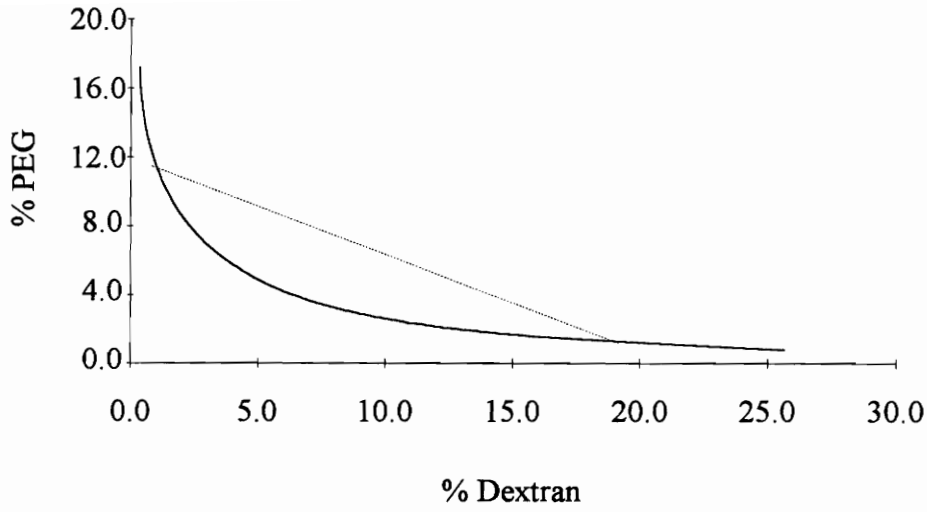


Figure 6.11. A phase diagram for the PEG/Dextran ATPS with the tie-line length represented as the dashed line between the Dextran-rich phase (bottom) and the PEG-rich phase (top).

Most thermodynamic models indicate that the partition coefficient decreases as the molecular weight of the protein increases. As Figure 6.10 and Table 6.4 show, this system follows that trend, except lysozyme and chymotrypsinogen-A are reversed. This deviation occurs because: (1) lysozyme interacts preferentially with Dextran, or chymotrypsinogen-A interacts preferentially with PEG; and (2) lysozyme self-associates to form a dimer or trimer, resulting in a higher apparent molecular weight. Both

explanations indicate that a protein's properties can play a significant role in its partitioning behavior.

Table 6.4. Comparison of basic protein properties for the five proteins used in Figure 6.10.

Protein	Molecular weight	Isoelectric point[a]
Lysozyme	13,200	10.5-11.0
Chymotrypsinogen-A	23,200	9.5
Bovine serum albumin	65,000	4.7
Transferrin	73,000	5.2
Catalase	250,000	5.6

a The isoelectric point is the pH value at which a protein has no net charge and tends to be least soluble.

Figure 6.12 shows the amino acids of each protein distributed in order of their hydrophobicity (from hydrophobic to hydrophilic). As the distribution reveals, chymotrypsinogen-A is more hydrophobic than lysozyme, which can lead to an affinity towards one phase. This type of preferential partitioning becomes much more significant as the pH of solution deviates from the isoelectric point of the protein.

This example demonstrates that a protein's physical and structural properties can have a significant impact on how it will partition between phases. Albertsson (1986, pp. 56-71) and Forciniti (1991) both describe in greater detail how protein properties influence partitioning behavior.

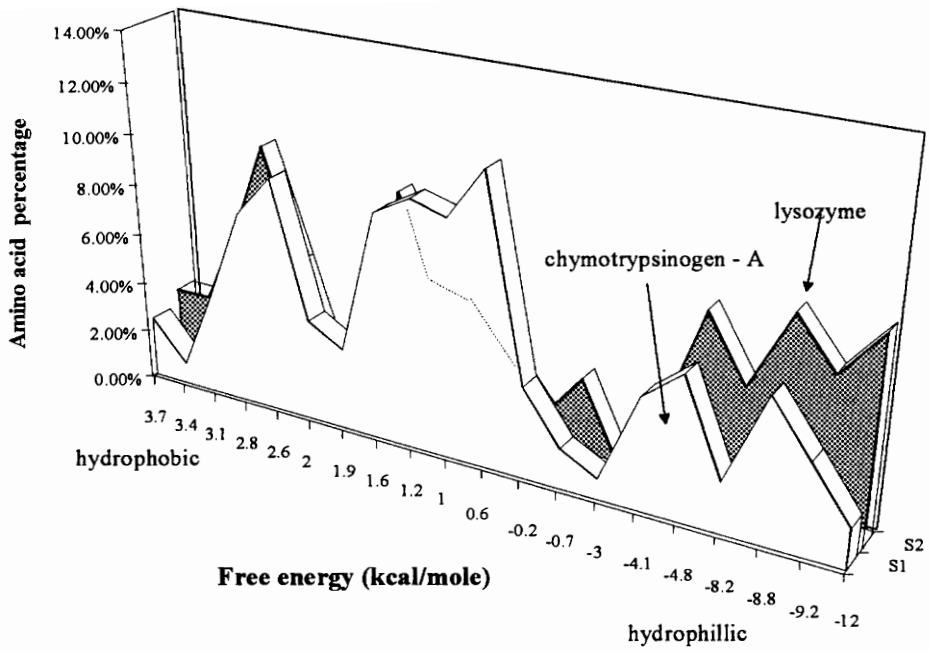


Figure 6.12. The hydrophobicity distribution for chymotrypsinogen-A and lysozyme.

2. Overview of the Expert-System Approach

Figure 6.13 shows an overview of our expert-system approach to predictive modeling and optimal design of extractive bioseparations in ATPS, which can be applied to protein

purification (Baughman and Liu, 1993, 1994a,b). The system includes three main components:

1. The *qualitative* reasoning skills of an extractive-separation expert system used as:
    - a central controller for overall processing control
    - a process optimizer for selecting the best extractor designs (Brunet and Liu, 1993)
2. The *quantitative* modeling capabilities of a protein-partitioning neural network as a bioseparation predictor for:
    - selecting the best multiphase-forming polymer/salt systems and biospecific affinity ligands
    - specifying the optimal ranges of independent variables
    - diagnosing a fault map of fault regions where independent variables correspond to destructive environments (e.g., denaturation) for protein partitioning
3. A bioseparation optimizer that applies the response-surface model (Khuri and Cornell, 1987) to identify the optimum values of key independent variables to assist in experimental design and process development.

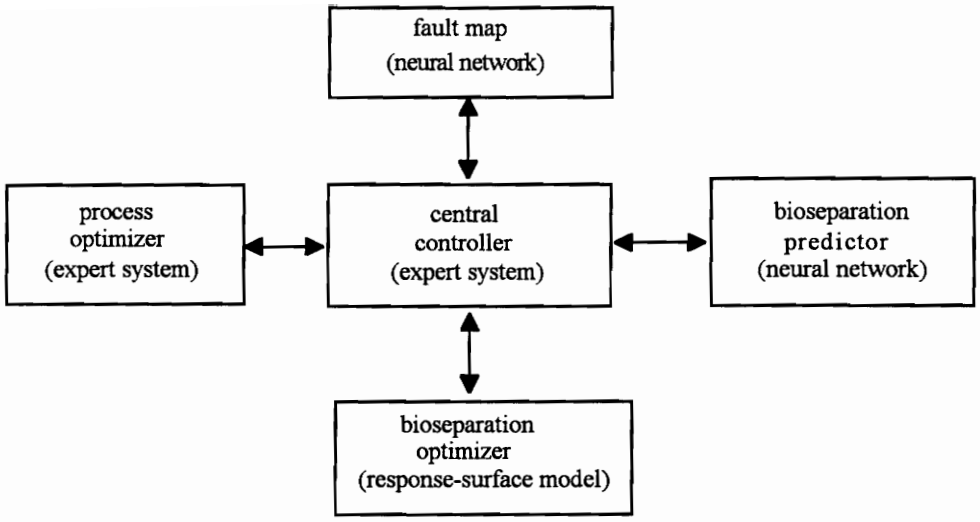


Figure 6.13. An expert-system approach to predictive modeling and optimal design of extractive bioseparations.

C. Bioseparation Predictor : Protein-Partitioning Network

With the rationale and background in place, we now turn to the development of an expert network for the PEG-Dextran system described in 6.2.A. The first step is the development of the neural network to analyze the protein partitioning behavior.

1. Partition Coefficients of Proteins in Aqueous Two-Phase Systems

We will now develop a neural network model to quantitatively predict the partition coefficients (i.e., separation factors), K_p, of proteins in ATPS over wide ranges of

independent variables, including protein properties, polymer-solution properties, and ion properties. Albertsson (1986, pp. 56-71) presents a detailed overview of the factors that determine how a protein partitions between phases, including size-dependent, electrochemical, hydrophobic, biospecific, and conformation-dependent contributions. His model represents the logarithm of the overall partition coefficient as a linear combination of the logarithm of each of these contributing factors. We use a form similar to Albertsson's equation for the logarithm of the partition coefficient, except that we include the hydrophobic and conformation-dependent properties as input variables within the neural network structure. Our overall equation to predict the partition coefficient is :

$$\ln K_p = \ln K_s + \ln K_{pH} + \ln K_t + \ln K_e + \ln K_b + \bullet\bullet\bullet\bullet\bullet \qquad (6.25)$$

where,

K_s = partition coefficient of the polymer solution at standard conditions. (temperature at 25°C and pH at the isoelectric point).

K_{pH} = partition contribution due to pH variation from the isoelectric point.

K_t = partition contribution due to the temperature variation from 25°C.

K_e = partition contribution from electrophoresis.

K_b = partition contribution from affinity binding.

The logarithmic form of equation 6.25 allows us to incorporate additional contributing terms to reflect the effects of new independent variables that result from future research.

Although, in this case study we consider only K_s and K_{pH} contributions to the overall partition coefficient K_p, the remaining contributions can be easily included. For example, a group of affinity ligand properties can be added to the input layer to predict K_b, accounting for their interactions between the protein and the ligand. Another approach could be to develop a classification network (Chapter 3) to identify how well a specific affinity ligand performs for a given set of proteins (e.g., poor, well, excellent).

2. Architecture of the Protein-Partitioning Network

The architecture of the protein-partitioning network, shown in Figure 6.14, combines the best features of both the backpropagation network and the hierarchical neural network, both discussed in Chapter 2. This architecture consists of three subnetworks, whose input nodes correspond to three groups of properties that affect protein partitioning in ATPS (Baskir and Hatton, 1989; King, 1992), namely: (1) polymer-solution properties; (2) protein properties; and (3) ion properties. Hidden layer 1 represents the interactions within each of the subnetworks, while hidden layers 2 and 3 account for higher-order interactions between the groups.

As Figure 6.14 indicates, the polymer-solution properties include the major variables from both the Ogston-based models (e.g., Forciniti, 1991; Forciniti and Hall, 1990; King et al., 1988) and the Flory-Huggins-based models (e.g., Diamond, 1990; Diamond and Hsu, 1990): temperature, polymer compositions, polymer molecular

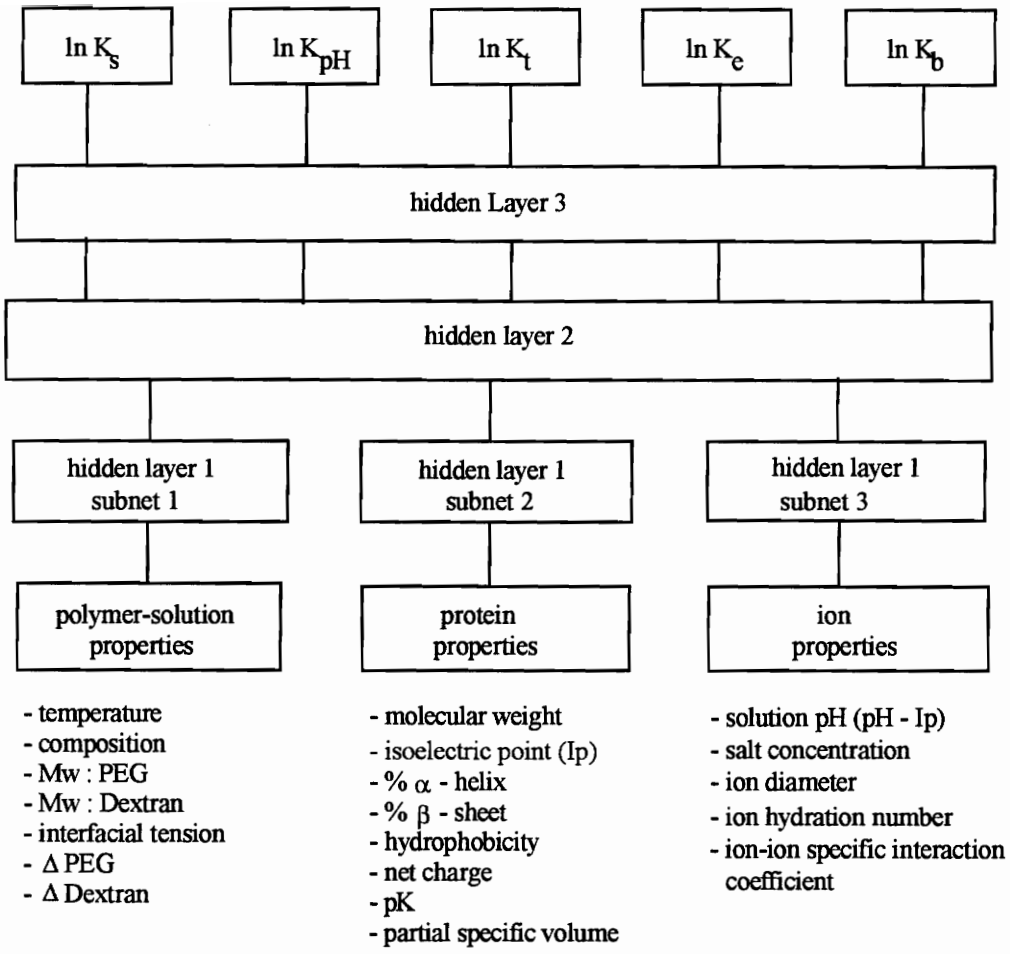


Figure 6.14. The protein-partitioning network.

weights, gradients of PEG and Dextran concentrations between the partitioned phases (i.e., ΔPEG and ΔDextran), and interfacial tension. The protein properties will be discussed in Section 6.2.C.3, Protein-Property Network. The ion properties appear in the Ogston-based models (e.g., Forciniti and Hall, 1990; King et al., 1988) as an electrostatic potential term. Here, we have selected solution pH and variation from the isoelectric

point as the two main independent variables. We have also included four salt properties: salt concentration, ion diameter, ion hydration number, and ion-specific interaction coefficient. The ion properties will be discussed further in Section 6.2.C.4.

3. Polymer-Solution Properties

The first step in developing the protein-partitioning network is to predict all polymer-solution properties that cannot be obtained without experimentation. The temperature, composition, Mw_{PEG}, and $Mw_{Dextran}$ are process operating conditions which are set by the user and directly entered into the network at specified values. The interfacial tension, ΔPEG, and ΔDextran are functions of the these properties and must be measured experimentally for a given set of process operating conditions. Therefore, we develop the polymer-solution network to predict the interfacial tension, ΔPEG, and ΔDextran, eliminating the experimental requirements.

In the following sections, we describe the development, training, and testing of the polymer-solution network.

a. Architecture of the Polymer-Solution Network

The polymer-solution network (Figure 6.15) accomplishes two tasks: (1) providing the required polymer-dependent input variables for the protein-partitioning network of Figure

6.13, particularly concentration gradients (ΔPEG and ΔDextran) and interfacial tension; and (2) generating phase diagrams for user reference, indicating quantitatively how the aqueous PEG/Dextran solution splits into two phases. As shown in Figure 6.15, four groups of properties affect how the two phases form: (1) polymer-solution interactions; (2) polymer properties; (3) polymer compositions; and (4) temperature.

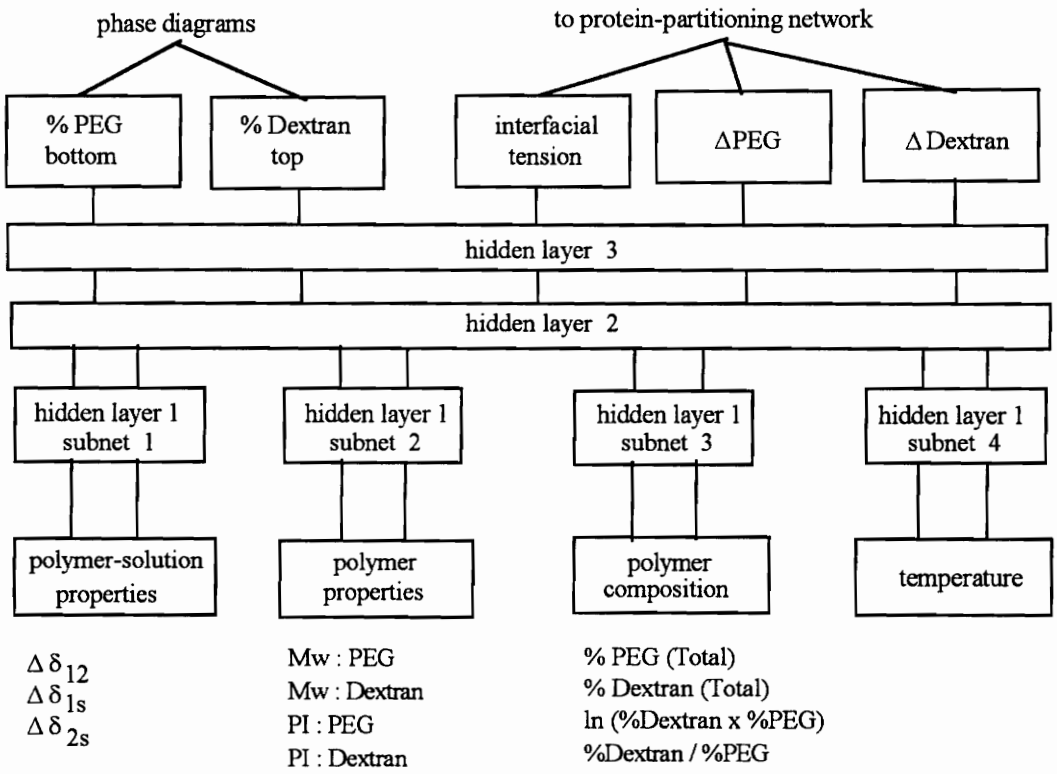


Figure 6.15. The polymer-solution network.

We can account for the polymer-solution interactions by using the solubility-parameter differences, δ_{12}, δ_{1s} and δ_{2s} (1:PEG, 2:Dextran, s:solvent). Since we are considering only the PEG/Dextran/water system, these solubility parameters are constant and the corresponding network interconnections have an essentially identical effect on the output-response patterns. Note, however, that predicting solubility parameters for glucose-based polymers, such as Dextran, by group-contribution methods has problems accurately representing variances in separations. For future work that includes multiple-phase systems, interaction properties need to be better defined (Barton, 1983).

The polymer properties in Figure 6.15 consist of not only the molecular weights (MW_{PEG} and $MW_{Dextran}$), but also the polydispersity index (PI), which is the ratio of the weight-average molecular weight M_w to the number-average molecular weight M_n. A recent study (Connemann et al., 1992) has shown that every model must include the polydispersity index to accurately predict the partitioned phase behavior in ATPS. Note that both Ogston and Flory-Huggins models fail to account for the molecular-weight distribution of Dextran.

The compositions in Figure 6.15 include the weight percents of PEG and Dextran, and ln(%PEG x %Dextran). The logarithmic term represents an interaction term for compositions that we have found very effective.

b. Training of the Polymer-Solution Neural Network

To train and test our network, we use 22 phase diagrams, including 64 training examples from Forciniti (1991) and Forciniti and Hall (1990), 32 training examples from Diamond (1990), and 20 training examples from Albertsson (1986). We divide the data into two sets, one to train the network, and the other to check for generalization. We randomly select 22 data points as the generalization set for interfacial tension, ΔPEG, and ΔDextran. We use the complete data set to train the system to predict %PEG in the bottom phase and %Dextran in the top phase, since those variables only appear in the ternary diagram and are not used in the protein-partitioning network of Figure 6.14. We present the data to the network in 13 columns, 12 input and 1 output. Table 6.5 shows the format of the data files *dex.nna*, *dextop.nna*, *peg.nna*, *pegbot.nna*, and *inttns.nna* used in the training and testing of the polymer-solution network.

Table 6.6 specifies the molecular weights and polydispersity-index (PI) values used for PEG and Dextran.

Table 6.5. The format of the files *dex.nna, dextop.nna, peg.nna, pegbot.nna* and *inttns.nna* used for training the polymer-solution network.

Column number	Variable type	Normalized variable	File name	Normalization factor
1	input	$\Delta\,\delta^2_{12}$		100
2	input	$\Delta\,\delta^2_{1s}$		10,000
3	input	$\Delta\,\delta^2_{2s}$		10,000
4	input	Mw_{PEG}		100,000
5	input	PI_{PEG}		10
6	input	$Mw_{Dextran}$		1,000,000
7	input	$PI_{Dextran}$		10
8	input	% PEG		100 %
9	input	% Dextran		100 %
10	input	%Dextran / %PEG		2
11	input	ln(%Dextran*%PEG/100)		1
12	input	$T - T_{standard}$		50 °C
13a	output	Δ Dextran	*dex.nna*	100 %
13b	output	% $Dextran_{top}$	*dextop.nna*	100 %
13c	output	Δ PEG	*peg.nna*	100 %
13d	output	% PEG_{bottom}	*pegbot.nna*	100 %
13e	output	interfacial tension	*inttns.nna*	1000 μNm^{-1}

Table 6.6. Specifications of PEG and Dextran for the illustrative predictions by the polymer-solution network.

	M_n	M_w	Polydispersity index
PEG 3400	3,400	3,400	1.00
PEG 4000	4,100	4,100	1.00
PEG 6000	5,600	5,600	1.00
PEG 8000	8,000	8,000	1.00
PEG 10000	11.400	11,400	1.00
PEG 20000	21,000	21,000	1.00
Dextran 10000 (D17)	23,000	30,000	1.30
Dextran 40000 (D19)	20,000	42,000	2.10
Dextran 70000 (D24)	72,000	135,000	1.88
Dextran 110000 (D37)	83,000	179,000	2.16

Table 6.7 lists the primary specifications used in training the polymer-solution networks. We use a backpropagation algorithm, three-hidden-layer network configuration, the hyperbolic tangent transfer function, and the delta rule to train this network.

A network configured with 20 nodes in the first layer, 12 nodes in the second layer, and 7 nodes in the third layer proves effective for training and recall of the polymer-solution networks. Figure 6.16 shows the first 50,000 of the 100,000 iterations in the training of ΔDextran output of the polymer-solution network. We leave the training of the $\%PEG_{bottom}$, $\%Dextran_{top}$, interfacial tension, and ΔPEG for the reader to develop.

Table 6.7. The specifications for the polymer-solution network.

Network type	backpropagation
Training file name	*dex.nna, dextop.nna peg.nna, pegbot.nna*, and *inttns.nna*
Transfer function (input layer)	linear
Transfer function (hidden layers)	tanh
Transfer function (output layer)	tanh
Learning rule	delta rule
Summation	sum
Error	standard
Network weight distribution	normal distribution : 3 σ limits of [-1, 1]

	input layer
Training iteration	5,000
Noise	0
Learning rate	0.9
Momentum coefficient	0.6
Error tolerance	0

	hidden layer 1		
Training iteration	10,000	30,000	70,000
Noise	0	0	0
Learning rate	0.3	0.15	0.04
Momentum coefficient	0.4	0.2	0.05
Error tolerance	0.1	0.1	0.1

	hidden layer 2		
Training iteration	10,000	30,000	70,000
Noise	0	0	0
Learning rate	0.25	0.13	0.03
Momentum coefficient	0.4	0.2	0.05
Error tolerance	0.1	0.1	0.1

	hidden layer 3		
Training iteration	10,000	30,000	70,000
Noise	0	0	0
Learning rate	0.2	0.1	0.03
Momentum coefficient	0.4	0.2	0.05
Error tolerance	0.1	0.1	0.1

	output layer		
Training iteration	10,000	30,000	70,000
Noise	0	0	0
Learning rate	0.15	0.08	0.02
Momentum coefficient	0.4	0.2	0.05
Error tolerance	0.1	0.1	0.1

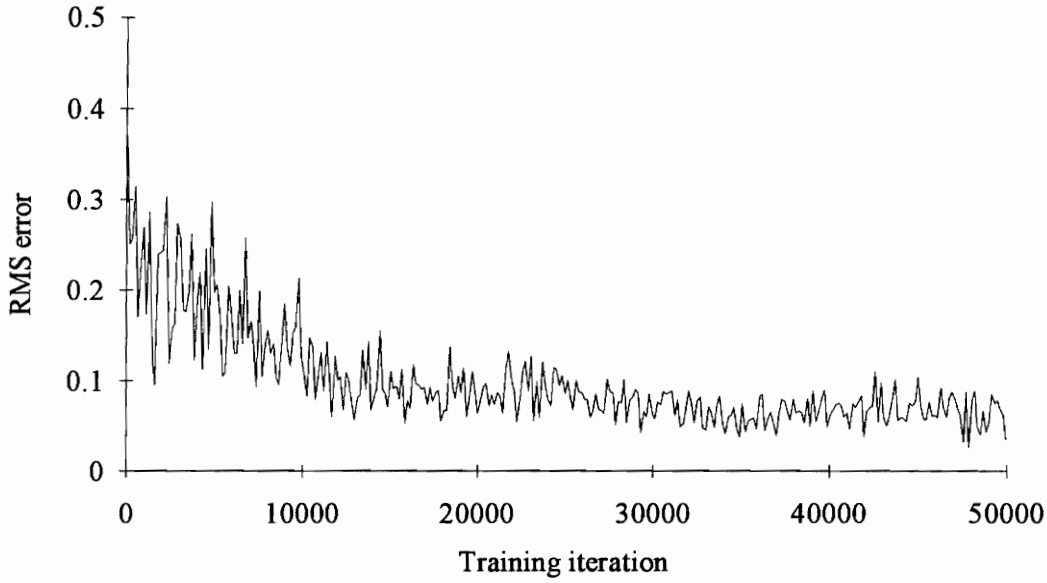


Figure 6.16. The training of the polymer-solution network for ΔDextran.

c. Illustrative Predictions by the Polymer-Solution Network

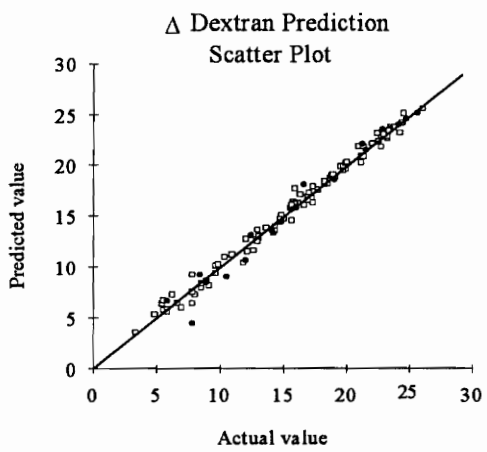
Once the network is trained, we test it against experimental data. The following tables and diagrams represent the results of those tests.

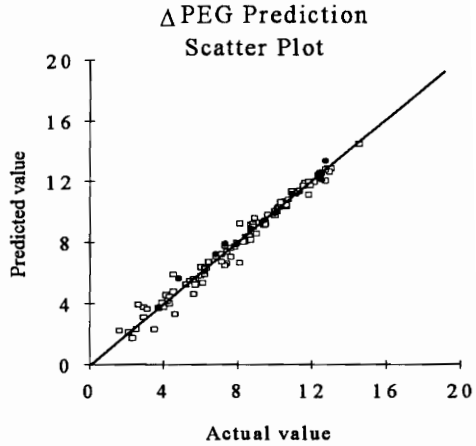
Table 6.8 gives the correlation coefficients for predictions by the ΔPEG, ΔDextran and interfacial tension subnetworks of Figure 6.15. Figure 6.17 shows the scatter plots of the predicted versus experimental values, and Figure 6.18 presents examples of comparison of predicted and experimental phase diagrams. As seen, most prediction errors in the phase diagrams appear in the low ΔPEG and ΔDextran regions near the plait

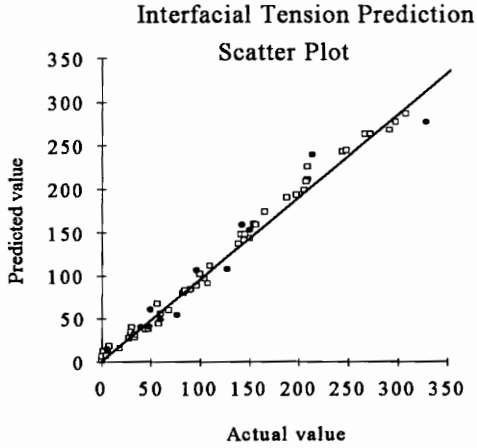
point. This is consistent with results from other techniques such as the Flory-Huggins model. For extraction applications, these regions are generally not feasible operating conditions. Therefore, the observed prediction errors have little practical significance. On the other hand, the fact that the correlation coefficients in Table 6.8 are very close to unity indicates that the polymer-solution network of Figure 6.15 *can* accurately predict the required polymer-dependent input variables for the protein-partitioning network of Figure 6.14.

Table 6.8. The correlation coefficients for illustrative predictions by the polymer-solution network.

	Data points	Correlation : Δ PEG	Correlation : Δ Dextran
Training	96	0.9902	0.9949
Generalization	25	0.9607	0.9871
Combined	121	0.9818	0.9940
	Data points	Correlation : interfacial tension	
Training	52	0.9964	
Generalization	12	0.9728	
Combined	64	0.9937	







Figures 6.17. Scatter plots of predictions by the polymer-solution network: (●) generalization (□) training results. Data Sources : Albertsson (1986), Diamond (1990), and Forciniti (1991).

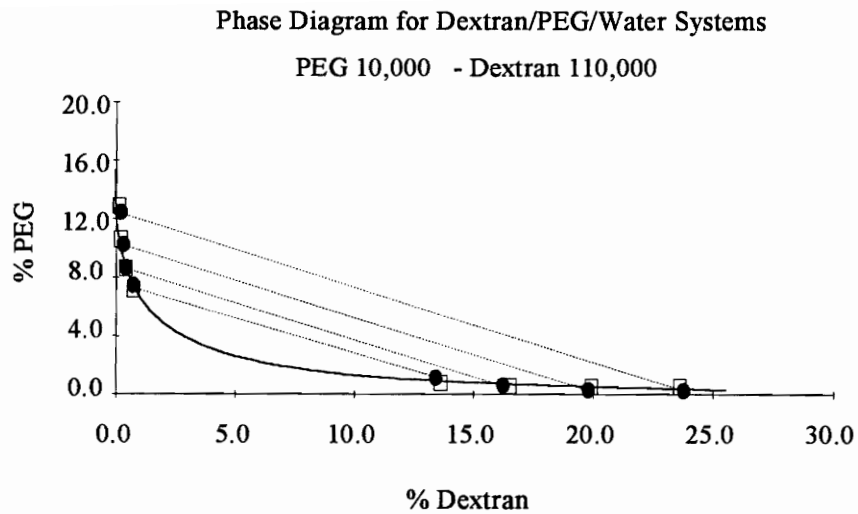


Figure 6.18a. The comparison of experimental phase diagrams with neural network predictions : (●) predicted , (□) experimental. Data sources : (a) Forciniti (1991); (b) Albertsson (1986); and (c) Diamond (1990).

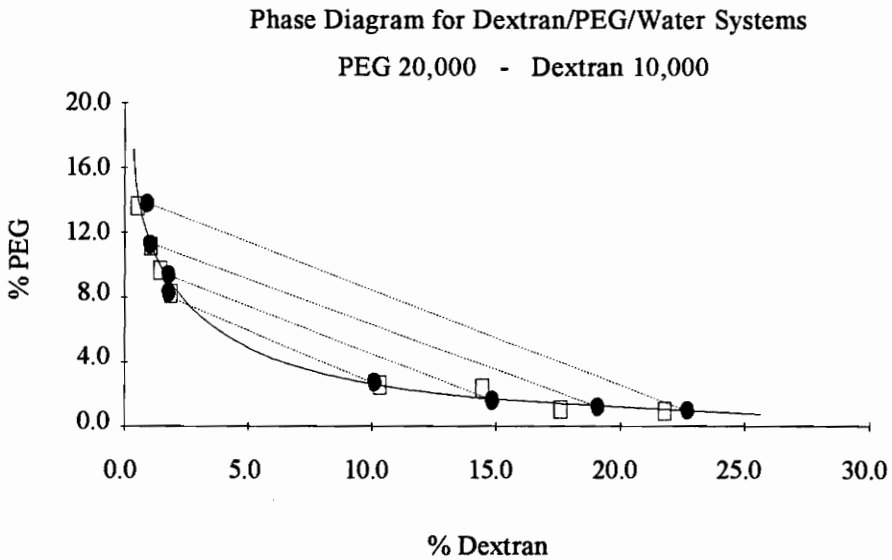


Figure 6.18b. The comparison of experimental phase diagrams with neural network predictions : (●) predicted , (□) experimental. Data sources : (a) Forciniti (1991); (b) Albertsson (1986); and (c) Diamond (1990).

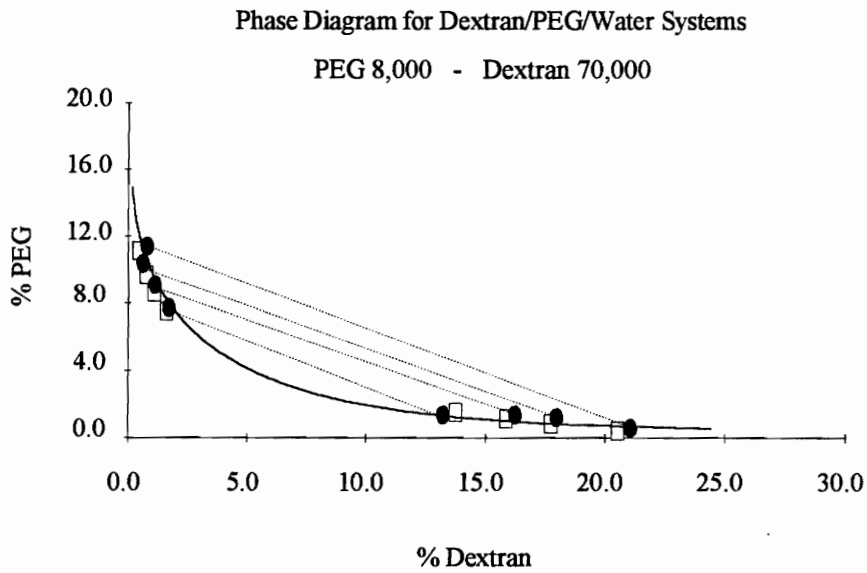


Figure 6.18c. The comparison of experimental phase diagrams with neural network predictions : (●) predicted , (□) experimental. Data sources : (a) Forciniti (1991); (b) Albertsson (1986); and (c) Diamond (1990).

4. Protein Properties

Next, we turn our attention to the second phase in developing the protein-partitioning network, namely, the generation of a protein-property data set that does not require experimentation. The molecular weight and isoelectric point are easily obtained for the desired proteins. We use the protein-property network to predict the protein's secondary structure (%α-helix and %β-sheet) for proteins that do not have their secondary structures characterized. Four characteristic factors of proteins (hydrophobicity, net charge, pK_c, and specific volume) are obtained using Kidera parameters (Kidera et al., 1985a,b).

The protein properties included in the protein-partitioning network account for the protein's affinity towards a specific phase. In this study, the top phase consists predominantly of the hydrophobic polymer, PEG; the bottom phase of the glucose-based polymer, Dextran. The differences between these two polymers in terms of both physical properties and protein interactions determine how a protein partitions between the two phases. Therefore, we need to incorporate additional protein properties, such as hydrophobicity and secondary structure, to represent the polymer-protein interactions and effectively train the network.

a. Independent Variables and Architecture for the Protein-Property Network

The molecular weight is a major protein property that we use from the Flory-Huggins (Section 6.2.B.1.b1) and Ogston (Section 6.2.B.1.b2) models. We also include the isoelectric point of the protein, which we show to affect protein partitioning in Section 6.2.B.1.c2.

As discussed in Section 3.5.B, classification of protein secondary-structure categories, the α-helix and β-sheet secondary-structure conformations have significantly different physical properties. For example, the α-helix has an overall dipole moment along the helical axis and occurs most often at the protein's surface, where one side of the α-helix is hydrophobic and the other hydrophilic. In comparison, the β-sheet has no hydrophobic/hydrophilic side at the surface of the protein; instead, hydrophobic and

hydrophilic side chains are staggered. These types of variations in the physical properties of both secondary structures, cause proteins with varying α-helix and β-sheet percentages to partition in ATPS differently.

In addition to structure, other protein properties also influence the protein's partitioning behavior. We have also adopted some of the results from the significant work of Kidera et al. (1985a,b). As seen in Figure 6.19, Kidera et al. statistically identified 188 properties of amino acids that affect protein conformation and other physical properties. They conducted two sets of cluster analysis to combine those variables having similar effects on protein properties. A principal-component analysis (Jollife, 1986) then produced the following five groups of nine characteristic properties, accounting for 68% of the variance of the original property set:

1	Bulk
2-4	Hydrophobicity
5-6	α-structure preference
7	β-sheet preference
8-9	Bend preference

Here, we describe the hydrophobicity characteristic properties (2-4) used in our protein-property subnetwork. These hydrophobicity properties fall into three independent categories. The first two groups are based on charged amino acids (arginine, asparagine,

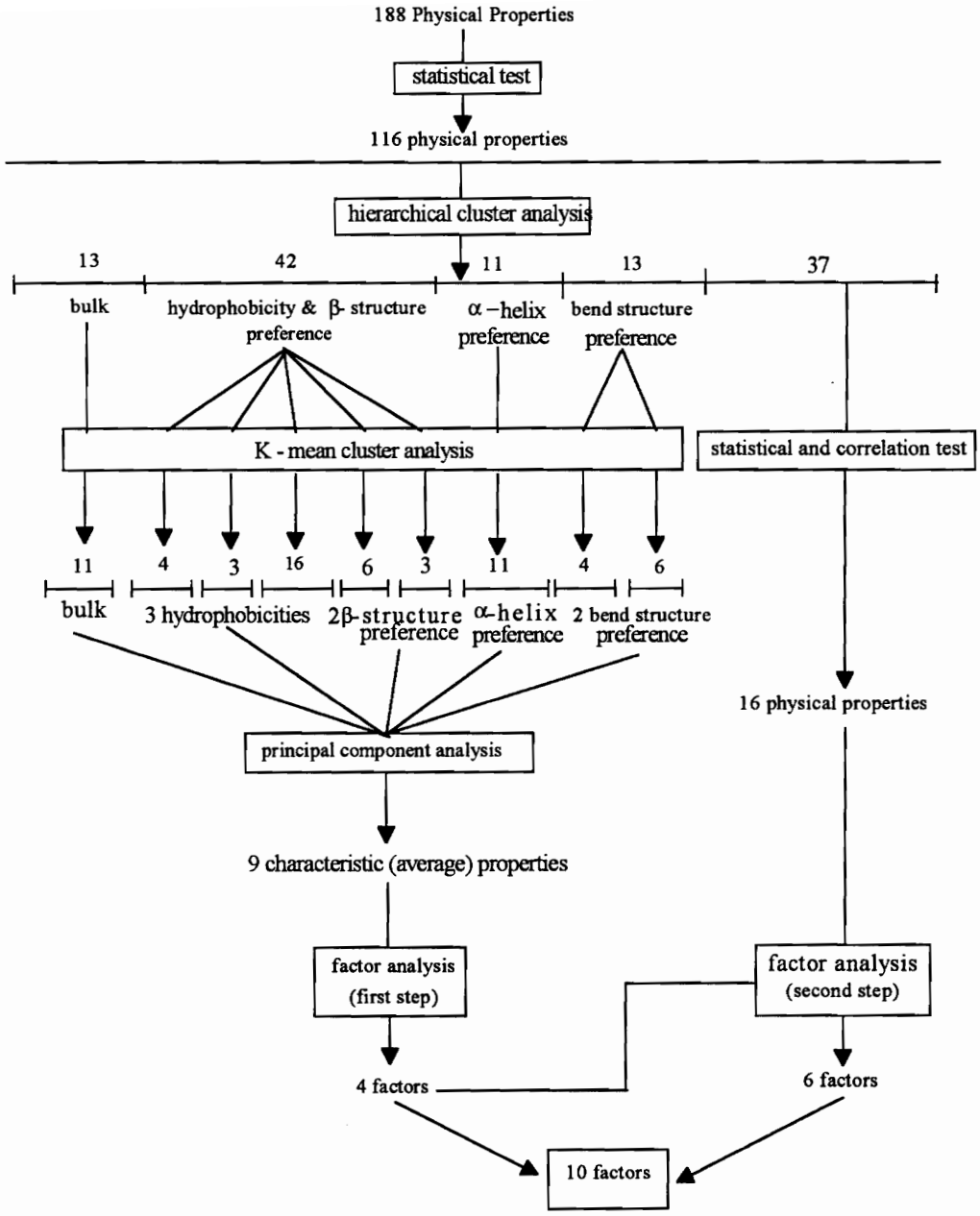


Figure 6.19. Kidera's statistical approach to identifying characteristic properties and characteristic factors for twenty naturally occurring amino acids (adopted from Kidera et al., 1985a).

glutamine, and lysine). The amino acid will either suppress or include ionizable side-chain groups. The last hydrophobicity category is defined as how deep the amino acid is buried inside the three-dimensional structure of the protein.

Table 6.9 gives values of Kidera's 9 characteristic properties for twenty naturally occurring amino acids (Kidera et al., 1985a).

As seen in Figure 6.19, an additional factor analysis generated 10 orthogonal parameters, called characteristic factors, from the 9 characteristic properties and 16 other physical properties not included in the principal component analysis (Kidera et al., 1985a). This factor analysis increased the properties accounted for to 85%. The resulting 10 characteristic factors are:

1. α-helix or bend-structure preference-related
2. bulk-related
3. β-structure preference-related
4. hydrophobicity
5. normalized frequency of double bond
6. average value of partial specific volume
7. a mixture of several physical properties
8. Normalized frequency in alpha region
9. pK_C (pH related term)
10. Surrounding hydrophobicity

Table 6.9 gives values of the 10 characteristic factors for 20 naturally occurring amino acids (Kidera et al., 1985a).

Table 6.9. Kidera's (a) nine characteristic properties and (b) ten characteristic factors for the twenty naturally occurring amino acids (Kidera et al., 1985).

Characteristic property number

	1	2	3	4	5	6	7	8	9
ALA	-1.44	-0.47	0.11	0.32	-0.51	-0.86	1.35	-1.29	-0.60
ARG	1.16	-0.57	-1.52	-1.07	-0.28	-0.13	-0.16	0.28	-0.03
ASN	-0.34	-1.25	-0.60	-0.96	-1.00	-1.19	-0.97	1.19	1.27
ASP	-0.54	-0.75	-1.74	-1.07	-1.17	-1.72	-0.06	0.74	1.39
CYS	-0.75	0.06	0.63	1.50	0.60	1.14	-0.53	1.18	-0.19
GLN	0.22	-1.24	-0.46	-1.05	0.19	-0.42	0.57	-0.14	-0.12
GLU	0.17	-0.62	-1.65	-1.03	-1.74	-1.78	1.96	-1.21	-0.27
GLY	-2.16	-1.02	-0.19	-0.03	-0.84	-0.99	-1.72	1.43	1.73
His	0.52	-0.46	-0.18	-0.13	-0.56	-0.10	0.59	-0.27	-0.27
ILE	0.21	1.37	0.97	1.52	1.91	1.27	0.06	-1.30	-1.49
LEU	0.25	1.06	1.01	1.14	0.69	0.02	0.93	-1.36	-1.14
LYS	0.68	-0.16	-1.62	-1.76	-0.86	-1.19	0.71	0.40	0.15
MET	0.44	0.20	0.72	1.00	0.45	0.24	1.39	-1.24	-1.29
PHE	1.09	1.46	1.24	1.16	0.88	0.48	0.37	-0.46	-0.75
PRO	-0.71	0.90	0.21	-0.72	-1.26	0.86	-1.72	1.03	1.98
SER	-1.21	-1.19	-0.33	-0.46	-0.54	0.22	-0.99	0.74	1.02
THR	-0.67	-0.97	0.01	-0.36	0.57	0.86	-0.68	0.11	0.14
TRP	2.08	2.06	1.55	0.67	0.61	0.42	0.23	0.83	-0.52
TYR	1.34	1.16	1.04	-0.07	1.02	1.21	-1.25	0.94	0.30
VAL	-0.34	0.42	0.77	1.38	1.84	1.66	-0.09	-1.63	-1.32

Characteristic factor number

	1	2	3	4	5	6	7	8	9	10
ALA	-1.56	-1.67	-0.97	-0.27	-0.93	-0.78	-0.20	-0.08	0.21	-0.48
ARG	0.22	1.27	1.27	1.87	-1.70	0.46	0.92	-0.39	0.23	0.93
ASN	1.14	-0.07	-0.07	0.81	0.18	0.37	-0.09	1.23	1.10	-1.73
ASP	0.58	-0.22	-0.22	0.81	-0.92	0.15	-1.52	0.47	0.76	0.70
CYS	0.12	-0.89	-0.89	-1.05	-0.71	2.41	1.52	-0.69	1.13	1.10
GLN	-0.47	0.24	0.24	1.10	1.10	0.59	0.84	-0.71	-0.03	-2.33
GLU	-1.45	0.19	0.19	1.17	-1.31	0.40	0.04	0.38	-0.35	-0.12
GLY	1.46	-1.96	-1.96	-0.16	0.10	-0.11	1.32	2.36	-1.66	0.46
His	-0.41	0.52	0.52	0.28	1.61	1.01	-1.85	0.47	1.13	1.63
ILE	-0.73	-0.16	-0.16	-0.77	-0.54	0.03	-0.83	0.51	0.66	-1.78
LEU	-1.04	0.00	0.00	-1.10	-0.55	-2.05	0.96	-0.76	0.45	0.93
LYS	-0.34	0.82	0.82	1.70	1.54	-1.62	1.15	-0.08	-0.48	0.60
MET	-1.40	0.18	0.18	-0.73	2.00	1.52	0.26	0.11	-1.27	0.27
PHE	-0.21	0.98	0.98	-1.43	0.22	-0.81	0.67	1.10	1.71	-0.44
PRO	2.06	-0.33	-0.33	-0.75	0.88	-0.45	0.30	-2.30	0.74	-0.28
SER	0.81	-1.08	-1.08	0.42	-0.21	-0.43	-1.89	-1.15	-0.97	-0.23
THR	0.26	-0.70	-0.70	0.63	-0.10	0.21	0.24	-1.15	-0.56	0.19
TRP	0.30	2.10	2.10	-1.57	-1.16	0.57	-0.48	-0.40	-2.30	-0.60
TYR	1.38	1.48	1.48	-0.56	-0.00	-0.68	-0.31	1.03	-0.05	0.53
VAL	-0.74	-0.71	2.04	-0.40	0.50	-0.81	-1.07	0.06	-0.46	0.65

Figure 6.20 shows the architecture of our protein-property network. In creating the network, we use three hydrophobicity terms from the nine characteristic properties, pK_c, and the partial specific volume from the ten characteristic factors, as independent variables that are not readily available for many proteins and cannot be easily predicted accurately. The secondary-structure percentages of α-helix and β-sheet for proteins are predicted using a backpropagation network, while the Kidera parameters are calculated as the linear combination of amino-acid compositions with their respective amino-acid weighting factors.

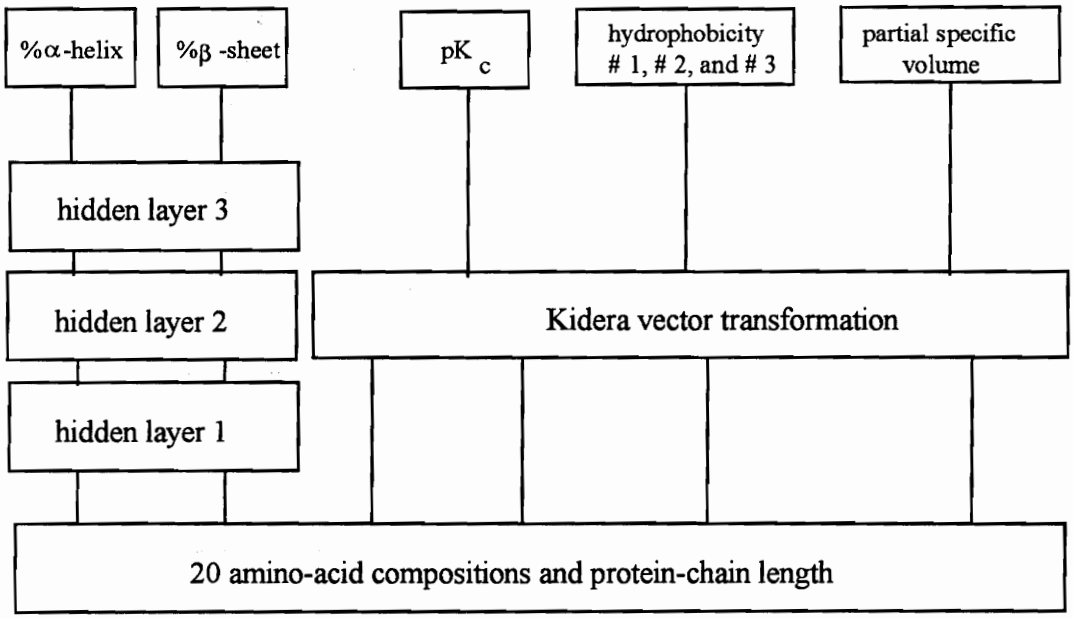


Figure 6.20. The protein-property network.

The tertiary and quaternary conformational effects on partitioning are indirectly considered in the inter-structure of the protein-partitioning network (Figure 6.14), since

many of the factors that influence these conformations are also factors in the protein-property subnetwork.

b. Training of the Protein-Property Network

Once we develop the protein-property network, we train and test it for secondary structure prediction (Figure 6.20) using 139 protein structures. Table 6.10 lists he proteins and their secondary structures used for training the network in the order they appear in the data file *ahlxtrn.nna*. The first 64 proteins are from Chou and Fasman (1974a, 1974b) and the remaining 75 are from Muskal (1991). All composition required in the training of the network for the data set from Muskal (1991) are generated from the Brookhaven Protein Data Bank, available from the Brookhaven National Laboratory.

The network uses 22 data columns consisting of 21 input variables (1 column for each amino acid and one for the number of amino acids in the protein) and 1 output variable (indicating %α-helix or %β-sheet structures). Table 6.11 shows the format of the training files *ahlxtrn.nna* and *bshttrn.nna*, used for training the protein-property network.

Table 6.12 shows the specifications used in training the protein-property networks. We use a standard backpropagation network with three hidden layers, configured with 20 nodes in the first layer, 12 nodes in the second layer, and 7 nodes in the third layer. The hyperbolic tangent transfer function and the delta learning rule are effective for this problem.

Table 6.10a. Proteins and their secondary structure used in the training of the protein-property network, file *ahlxtrn.nna* and *bshttrn.nna*.

	% α - helix	% β - sheet
Ca binding parvalbumin	56.5	4.6
Cytochrome b-562	78.6	0.0
Cytochrome c	42.7	5.8
Cytochrome c-2	43.8	3.6
Cytochrome c-550	39.6	4.5
Cytochrome c-555	36.1	0.0
Hemerythrin (Met-)	64.6	0.0
Hemerythrin (Myo-)	68.6	0.0
Hemerythrin (G. gouldi)	71.7	0.0
Hemoglobin, alpha (human)	77.3	0.0
Hemoglobin, beta (human)	76.7	0.0
Hemoglobin, alpha (horse)	77.3	0.0
Hemoglobin, beta (horse)	78.8	0.0
Hemoglobin (glycera)	76.2	0.0
Hemoglobin (lamprey)	79.1	0.0
Hemoglobin (midge larva)	83.1	0.0
Hemoglobin, gamma (human)	77.4	0.0
Myglobin (seal)	79.1	0.0
Myglobin (sperm whale)	79.1	0.0
a - Chymotrypsin	8.3	40.2
Concanavalin	2.1	57.8
Elastase	7.5	49.2
Erabutoxin B	0.0	50.0
Immunoglobulin Fab (V_H and C_H) (human)	2.3	59.1
Immunoglobulin Fab (V_H and C_L) (human)	2.4	58.7
Immunoglobulin MCG (human)	10.7	65.7
Immunoglobulin REI (human)	4.6	56.5
Penicillopepsin	8.7	41.8
Prealbumin	6.3	45.7
Protease A	8.3	51.9
Protease B	4.9	56.2

Table 6.10b. Proteins and their secondary structure used in the training of the protein-property network, file *ahlxtrn.nna* and *bshttrn.nna*.

	% α - helix	% β - sheet
Rubredoxide Dismutase	0.0	25.9
Superoxide Dismutase	4.6	50.3
Trypsin	0.0	41.3
Actinidin	28.0	14.2
Cytochrome b-5	46.2	28.0
Ferredoxin	24.1	31.5
High-protein iron protein	11.8	15.3
Insulin	49.0	23.5
Lysozyme (bacteriopage T4)	65.2	12.2
Lysozyme (chicken)	41.9	17.1
Papain	26.4	14.2
Phospholipase A-2	49.6	9.8
Ribonuclease S	25.0	44.4
Staphylococcal nuclease	25.5	28.9
Subtilisin inhibitor	17.7	33.6
Thermolysin	35.4	20.6
Trypsin inhibitor	19.0	27.6
Adenylate kinase	54.1	12.4
Alcohol dehydrogenase	28.3	30.8
Carbonic anhydrase B	18.9	27.7
Carbonic anhydrase C	20.5	26.6
Carboxypeptidase A	35.2	14.7
Carboxypeptidase B	31.4	14.7
Dihydrofolate reductase	17.6	30.8
Flavodoxin	36.2	26.8
Glyceraldehyde 3-phospate : dehydrogenase (lobster)	32.7	34.5
Glyceraldehyde 3-phospate : dehydrogenase (B. stearotherm)	31.1	26.7
Lactate dehydrogenase	40.4	24.0
Phosphoglycerate kinase	40.9	23.8
Rhodanese	41.0	14.3

Table 6.10c. Proteins and their secondary structure used in the training of the protein-property network, file *ahlxtrn.nna* and *bshttrn.nna*.

	% α - helix	% β - sheet
Subtilisin BPN	29.1	20.0
Thiordoxin	48.2	27.8
Thiose phosphate isomerase	54.0	20.2
Actinoxanthin	0.0	43.9
Azurin	11.3	25.8
Crambin	47.8	8.7
Alpha cobratoxin	5.6	22.5
2-hydroxyethylthiopapain - crystal form D	26.4	17.0
Avian pancreatic polypeptide	50.0	0.0
Bence-Jones immunoglobulin rei var. portion	2.8	47.7
Actinidin (sulfhydryl proteinase)	30.3	18.3
Alpha-lytic protease	7.1	52.5
Acid proteinase (rhizopuspepsin)	13.8	44.9
Azurin (oxidized)	16.3	33.3
Carbonic anhydrase II (carbonate dehydratase)	16.4	28.9
Carbonic anhydrase form B	15.6	30.9
Apo-lactate dehydrogenase isoenzyme c4	38.4	16.9
Bence-Jones protein (lamda, var. domain)	2.6	43.0
Adenylate kinase	54.6	12.9
Acid proteinase (penicillopepsin)	13.9	45.5
Concanavalin A	0.0	40.5
Calcium binding protein	57.3	0.0
Acid proteinase (endothiapepsin)	9.4	45.8
Calcium-binding parvalbumin	56.5	0.0
Citrate synthase	52.4	4.1
Carboxypeptidase alpha (Cox)	38.1	16.3
Aspartate carbamoyltransferase	32.7	20.2
Catalase	32.5	15.5
Cytochrome c5 (oxidized)	47.0	0.0
Cytochrome c (rice)	42.3	0.0
Cytochrome c3 (desulfovibrio desulfuricans)	24.6	0.0

Table 6.10d. Proteins and their secondary structure used in the training of the protein-property network, file *ahlxtrn.nna* and *bshttrn.nna*.

	% α - helix	% β - sheet
Ferrocytochrome C	34.0	0.0
Ferredoxin (peptococcus aerogenes)	14.8	7.4
Flavodoxin	32.0	21.8
Glucagon (pH 6 - pH 7 form)	48.3	0.0
Glutathione peroxidase	32.2	15.8
Cytochrome c (rhodospirillum molischianum)	74.8	0.0
Cytochrome c3 (desulfovibrio vulgaris miyazaki)	28.0	9.3
Cytochrome c peroxidase (baker's yeast)	50.2	5.5
Gamma cymotrypsin A	9.7	33.1
Gamma-II crystallin	7.5	44.3
Gene 5 DNA binding protein	0.0	4.6
Cu, Zn superoxide dismutase	2.0	38.4
Cytochrome b5 (oxudized)	40.0	24.7
Cytochrome c2 (reduced)	42.9	0.0
Ferredoxin (spirulina platensis)	13.3	15.3
D-glyceraldehyde-3-phosphate dehydogenase	27.2	21.0
Glutathione reductase	34.3	24.1
Cytochrome c551 (reduced)	50.0	0.0
Dihydrofolate reductase	26.4	30.8
Flavodoxin (semiquinone form)	36.2	21.0
Cytoplasmic malate dehydrogenase	42.6	18.9
Ferredoxin (azotobacter vinelandii)	34.0	13.2
Phospholipase A2 (bovine pancreas)	48.8	6.5
Oxidized high potential iron protein	22.4	10.6
Myglobin (met)	72.5	0.0
Neurotoxin B	0.0	41.9
Phospholipase A2 (porcine pancreas)	43.5	4.8
PFC fragment on an IGG1	3.6	30.6
Rhodanese	29.7	10.9
Scorpion neurotoxin (variant 3)	12.3	18.5
Prealbumin (human plasma)	7.0	51.8

Table 6.10e. Proteins and their secondary structure used in the training of the protein-property network, file *ahlxtrn.nna* and *bshttrn.nna*.

	% α - helix	% β - sheet
Subtilisin novo	21.5	13.8
Proteinase A	9.9	54.1
Staphylococcal nuclease	20.6	19.9
Streptomyces subtilisin inhibitor	15.9	24.3
Satellite tobacco necrosis virus	11.4	44.6
Taka-amylase A	26.2	14.4
Tomato bushy stunt virus	4.1	30.4
Plastocyanin (Hg2+ substituted)	10.1	35.4
Phosphoglycerate kinase	34.5	11.1
Phosphoglycerate mutase	30.0	6.5
Ribonuclease A	21.0	38.7
Rat mast cell protease II	8.0	37.1
Proteinase B (streptomyces griseus)	6.5	51.9
Thermolysin	41.5	16.5
Southern bean mosaic virus coat protein	15.1	35.2
Rubredoxin (oxidized, FE(III))	16.7	14.8

Table 6.11. The format of the file *ahlxtrn.nna* and *bshttrn.nna* used for training the protein-property network.

Column number	Variable type	Normalized variable	File name	Normalization factor
1	input	# of amino acids in the protein		1,000
2	input	% alanine		100 %
3	input	% arginine		100 %
4	input	% asparagine		100 %
5	input	% aspartic acid		100 %
6	input	% cysteine		100 %
7	input	% glutamine		100 %
8	input	% glutamic acid		100 %
9	input	% glycine		100 %
10	input	% histidine		100 %
11	input	% isoleucine		100 %
12	input	% leucine		100 %
13	input	% lysine		100 %
14	input	% methionine		100 %
15	input	% phenylalanine		100 %
16	input	% proline		100 %
17	input	% serine		100 %
18	input	% threonine		100 %
19	input	% tyrptophan		100 %
20	input	% tyrosine		100 %
21	input	% valine		100 %
22a	output	% α - helix	*ahlxtrn.nna*	100 %
22b	output	% β - sheet	*bshttrn.nna*	100 %

Table 6.12. The specifications for the protein-property network..

Network type	backpropagation
Training file name	*ahlxtrn.nna, bshttrn.nna, ahlxtst.nna,* and *bshttst.nna.*
Transfer function (input layer)	linear
Transfer function (hidden layers)	tanh
Transfer function (output layer)	tanh
Learning rule	delta rule
Summation	sum
Error	standard
Network weight distribution	normal distribution : 3 σ limits of [-1, 1]

input Layer	
Training iteration	5,000
Noise	0
Learning rate	0.9
Momentum coefficient	0.6
Error tolerance	0

hidden layer 1			
Training iteration	10,000	30,000	70,000
Noise	0	0	0
Learning rate	0.3	0.15	0.04
Momentum coefficient	0.4	0.2	0.05
Error tolerance	0.1	0.1	0.1

hidden layer 2			
Training iteration	10,000	30,000	70,000
Noise	0	0	0
Learning rate	0.25	0.13	0.03
Momentum coefficient	0.4	0.2	0.05
Error tolerance	0.1	0.1	0.1

hidden layer 3			
Training iteration	10,000	30,000	70,000
Noise	0	0	0
Learning rate	0.2	0.1	0.03
Momentum coefficient	0.4	0.2	0.05
Error tolerance	0.1	0.1	0.1

output layer			
Training iteration	10,000	30,000	70,000
Noise	0	0	0
Learning rate	0.15	0.08	0.02
Momentum coefficient	0.4	0.2	0.05
Error tolerance	0.1	0.1	0.1

Figure 6.21 shows the first 50,000 iterations in the training of %α-helix output of the protein-property network. The training for the %b-sheet is left to the reader to develop.

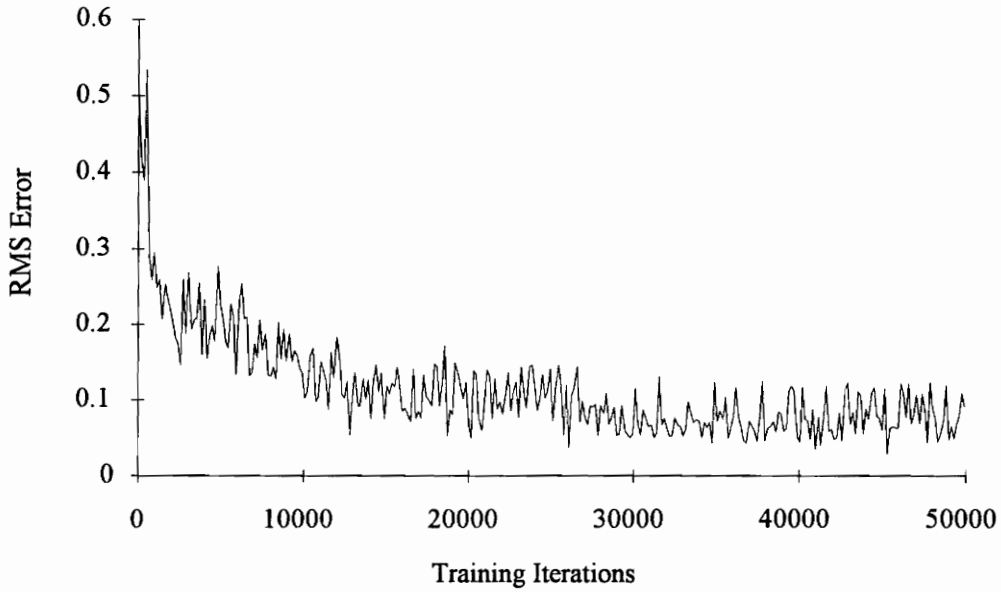


Figure 6.21. The training of the protein-property network for % α-helix.

c. Illustrative Predictions by the Protein-Property Network

Once we have trained the network, we test its predictive abilities for recall of the original data set and for generalization using a data set of 15 different protein structures, as listed in Table 6.13. This data set is from Muskal (1991).

Table 6.13. Proteins and their secondary structure used in the testing of the protein-property network, file *ahlxtst.nna* and *bshttst.nna*.

	% α - helix	% β - sheet
Tosyl-elastase	10.4	34.2
Glycolate oxidase	44.3	12.6
Apo-L-lactate dehydrogenase	41.8	19.0
Trypsin	12.1	34.5
Trypsinogen	10.2	37.8
Thaumatin I	12.1	37.2
Triose phosphate isomerase	45.7	17.0
D-galactose/D-glucose binding protein	43.0	18.4
Pea lectin	3.1	46.9
Proteinase K	26.9	21.5
Catabolite gene activator protein	32.7	13.8
Phosphofructokinase	46.1	18.5
Tyrosyl-transfer RNA synthetase	51.7	9.5
D-xylose isomerase	47.8	8.9
Trypsin inhibitor (crystal form II)	20.7	24.1

Our tests confirm the work of Muskal (1991), showing that the neural network approach predicts the secondary structure more accurately than previously developed models. For example, the probabilistic and multiple linear regression methods achieve only a 66% prediction level, while the neural network achieves greater than 80% accuracy. Figure 6.22 and Table 6.14 show the prediction capabilities of %α-helix and %β-sheet. Note that the protein-property network is still in its early stage of development and should only be used for proteins that do not have their secondary structure defined in the

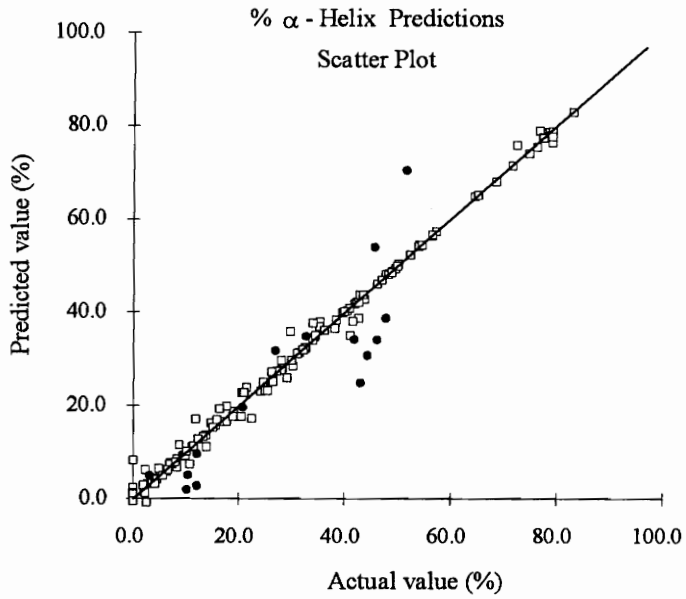
currently available protein data banks (e.g., from Brookhaven National Laboratory) or other reliable sources.

Table 6.14. The correlation coefficients for secondary-structure prediction.

	Data Points	Correlation α - helix	Correlation β - sheet
Training	139	0.9966	0.9953
Generalization	15	0.8772	0.6958
Combined	154	0.9885	0.9816

Table 6.15 gives values of the three Kidera parameters for hydrophobicity, along with the % α-helix and % β-sheet predicted by the protein-property network of Figure 6.20, for the five proteins used in the Dextran/PEG ATPS. These independent variables serve as input variables for the protein-partitioning network of Figure 6.14.

For more information, we refer the reader to a number of reported applications of neural networks to the modeling and prediction of protein structures, including Dubehak et al. (1993), Ferran and Ferrara (1991), Muskal (1991), Muskal and Kim (1992), Qian and Sejnowski (1988), Post and Sander (1993), Schneider and Wrede (1993), and Zhang et al. (1992).



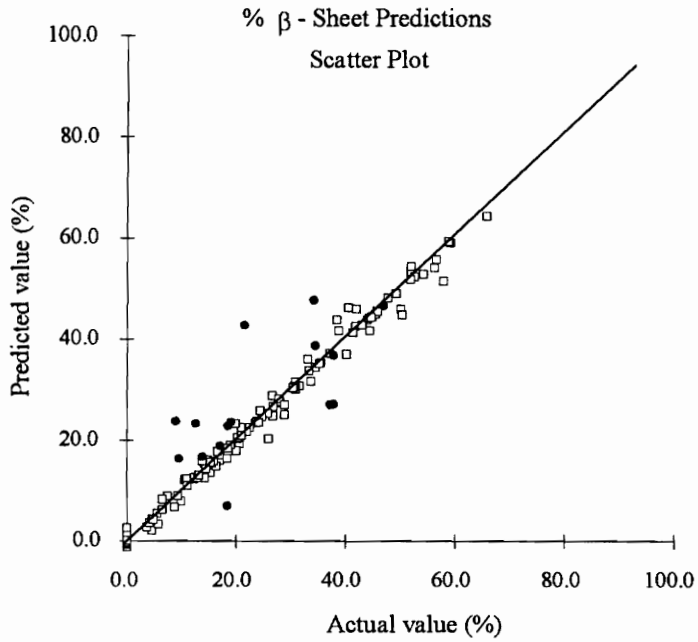


Figure 6.22. Scatter plots for secondary protein-structure predictions from the protein-property network. □ - training data, • - testing data.

Table 6.15. The specifications of property values for five proteins used in the illustrative predictions by the protein-partitioning network.

	Lysozyme	Chymotrypsinogen - A	Bovine serum albumin	Transferrin	Catalase
Molecular weight	13,200	23,200	65,000	73,000	250,000
Isoelectric point	10.5 to 11.0	9.5	4.7	5.2	5.6
% α - helix	53 %	8.0 %	47.0	74.0 %	71.0
% β - sheet	15 %	40.0 %	37.0	0.0 %	0.0
Bulk	-0.141	-0.451	-0.130	-0.250	-0.142
pK_C	-0.014	-0.186	0.121	0.036	0.113
Specific volume	-0.307	-0.274	-0.309	-0.272	-0.251
Hydrophobicity					
# 1 (charge)	-0.128	-0.220	-0.074	-0.125	-0.101
# 2 (charge)	-0.171	0.011	-0.212	-0.142	-0.133
# 3 (depth)	-0.096	0.029	-0.106	-0.059	-0.126

4. Ion Properties

The final group of properties to consider in the protein-partitioning network (Figure 6.14) is ion properties. Salts added to the ATPS partition unevenly between the two phases based on their chemical nature and their interactions with the two polymer phases. This uneven partitioning leads to an electrostatic-potential difference between the two phases, which in turn causes the proteins to partition preferably to one phase with a logarithmic correlation. The salt and ion properties used in the protein-partitioning network account for these effects.

The effect of salts on protein partitioning is presently an area of intensive research, and ion properties have been the most difficult to model in regard to their effects on protein partitioning in ATPS. Haynes et al. (1993) have developed the most complete model, based on the osmotic virial expansion, which includes many additional electrostatic effects. We use this model as the primary source for our network parameters, including only basic variables which do not require experimental measurements. As more data on the effects of ion properties and improved parameters for representing these properties become available, development of this section of the protein-partitioning network (Figure 6.14) can progress further.

The first property that we introduce to the network is the salt concentration. Albertsson (1986) shows how the addition of NaCl to an ATPS will reduce the partition coefficient of the protein. To use ion properties directly, we define the salt as a combination of an anion and a cation; e.g., NaCl is divided into Na^+ and Cl^-. Both the anion and cation properties include the hard-sphere diameter (Table 6.16), the charge, and the hydration number (Table 6.17). We also incorporate an ion-ion specific interaction coefficient (Table 6.18), β_{ij} at 25°C, included in the Haynes et al. model (1993).

Table 6.16. The anion and cation hard-sphere diameters in dilute aqueous solutions at 25°C (Haynes et al., 1993).

Cation	d_{cation} (A)	Anion	d_{anion} (A)
Na^+	2.32	Cl^-	3.62
K^+	3.04	$H_2PO_4^-$	4.44
Li^+	1.86	HPO_4^{2-}	3.82
Ca^{2+}	2.28	HSO_4^-	3.66
Mn^{2+}	1.98	SO_4^-	3.28

Table 6.17. The hydration numbers (h's) for anions and cations in water at 25 °C (Haynes, 1992).

Cation	h_{cation}	Anion	h_{anion}
Cs^+	0.0	ClO_4^-	0.3
K^+	0.6	Cl^-	0.9
Na^+	1.9	Br^-	0.9
Li^+	3.2	I^-	0.9
H^+	3.8	RSO_3^-	0.9
Ca^{2+}	4.1	F^-	1.6
Mg^{2+}	4.8	OH^-	4.0

Table 6.18. The ion-ion specific interaction coefficients, β_{ij}, in aqueous solutions at 25 °C (Haynes et al., 1993).

Electrolyte	β_{ij}	Electrolyte	β_{ij}
NaCl	0.15	KH_2PO_4	- 0.07
KCl	0.10	$NaHSO_4$	- 0.07
NaH_2PO_4	- 0.06	$KHSO_4$	- 0.08

5. Protein-Partitioning Network

a. Training of the Protein-Partitioning Network

The three property groups (polymer-solution, protein-properties, and ion properties) provide all required input variables for the protein-partitioning network of Figure 6.14. With those properties identified and other prediction networks in place, we can now turn to the training and generalization of the network, using 295 data sets for $\ln K_S$ and 922 data sets for $\ln K_{pH}$ (Forciniti, 1991). The network uses 25 data columns for input variables and 1 data column for the output variable. Table 6.19 lists the format of the data *part.nna* and *phpart.nna*, used for training the protein-partitioning network.

Table 6.20 lists the primary specifications we use to training the network. As with the other networks trained in this case study, we use a standard backpropagation network with three hidden layers. The hidden layers are configured with 35 nodes in the first hidden layer, 20 nodes in the second hidden layer, and 12 nodes in the third hidden layer. We use the hyperbolic tangent transfer function and delta rule to train the network.

Because most backpropagation algorithms are not efficient in training large data sets with numerous factors influencing the response variables. It is helpful to train the network in stages, beginning with a smaller data set and gradually introducing more training examples. In this case, we initially train the protein-partitioning network with one protein, then add proteins until the network has seen the complete data set. Note that for

each additional training example, we reset the learning rate and momentum coefficient for each layer to values consistent with those for 30,000 iterations of Table 6.20, in order to make the necessary adjustments while maintaining the integrity of the previously trained network.

Figure 6.23 shows the training of the protein-partitioning network at standard conditions, $\ln K_S$. We select lysozyme and catalase as the first two proteins to train the network, they correspond to the limiting cases (i.e., boundary conditions) of our database in terms of molecular weight: lysozyme is the lowest (13,200) and catalase is the highest (250,000). Choosing the initial training examples near the desired boundary conditions of the database improves the prediction accuracy of the resulting network. Since the backpropagation algorithm has difficulty fine-tuning a network if new training examples require large extrapolations, to include extrapolated training sets, we must essentially retrain the network using large learning rates and momentum coefficients. We use protein molecular weight as the limiting condition because it is the primary influence on protein partitioning at standard conditions. Figure 6.23 demonstrates how the network's prediction capability for new proteins, improves with the addition of each new protein, while the overall prediction capability of all the proteins declines. This result is consistent with the learning curve for neural networks shown in Figure 6.9.

Table 6.19. The format of the files *part.nna* and *phpart.nna* used for training the protein-partitioning network.

Column number	Variable type	Normalized variable	file	Normalization factor
1	input	$Mw_{protein}$		1,000,000
2	input	isoelectric point		10
3	input	% α - helix		100 %
4	input	% β - sheet		100 %
5	input	Bulk parameter		1
6	input	pK_C parameter		1
7	input	specific volume parameter		1
8	input	hydrophobicity 1 parameter		1
9	input	hydrophobicity 2 parameter		1
10	input	hydrophobicity 3 parameter		1
11	input	pH - Ip		10
12	input	Mw_{PEG}		100,000
13	input	$Mw_{Dextran}$		1,000,000
14	input	% PEG		100 %
15	input	% Dextran		100 %
16	input	ln(%Dextran*%PEG/100)		1
17	input	Δ PEG		100 %
18	input	Δ Dextran		100 %
19	input	interfacial tension		1000 μNm^{-1}
20	input	salt concentration		0.01 M
21	input	cation diameter		10 Angstrom
22	input	anion diameter		10 Angstrom
23	input	cation hydration number		10
24	input	anion hydration number		10
25	input	ion-ion specific interaction coefficient		1
26a	output	$\ln K_S$	*part.nna*	1
26b	output	$\ln K_{pH}$	*phpart.nna*	1

Table 6.20. The specifications for the protein-partitioning network.

Network type	backpropagation
Training file name	*part.nna* and *phpart.nna*
Transfer function (input layer)	linear
Transfer function (hidden layers)	tanh
Transfer function (output layer)	tanh
Learning rule	delta rule
Summation	sum
Error	standard
Network weight distribution	normal distribution : 3 σ limits of [-1, 1]

	input layer
Training iteration	5,000
Noise	0
Learning rate	0.9
Momentum coefficient	0.6
Error tolerance	0

	hidden layer 1		
Training iteration	10,000	30,000	70,000
Noise	0	0	0
Learning rate	0.3	0.15	0.04
Momentum coefficient	0.4	0.2	0.05
Error tolerance	0.1	0.1	0.1

	hidden layer 2		
Training iteration	10,000	30,000	70,000
Noise	0	0	0
Learning rate	0.25	0.13	0.03
Momentum coefficient	0.4	0.2	0.05
Error tolerance	0.1	0.1	0.1

	hidden layer 3		
Training iteration	10,000	30,000	70,000
Noise	0	0	0
Learning rate	0.2	0.1	0.03
Momentum coefficient	0.4	0.2	0.05
Error tolerance	0.1	0.1	0.1

	output layer		
Training iteration	10,000	30,000	70,000
Noise	0	0	0
Learning rate	0.15	0.08	0.02
Momentum coefficient	0.4	0.2	0.05
Error tolerance	0.1	0.1	0.1

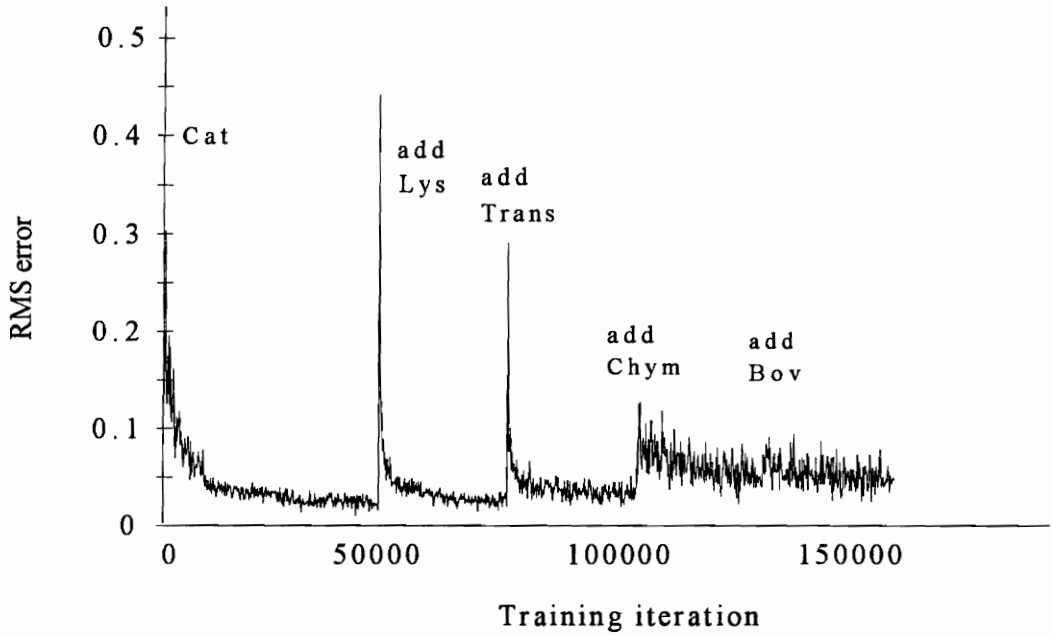


Figure 6.23. The training of protein-partitioning network at standard conditions, $\ln K_S$.

Figure 6.24 shows the training of the protein-partitioning network for deviations from the isoelectric point, $\ln K_{pH}$. This network is initially trained with two proteins, bovine serum albumin and catalase, followed by the addition of lysozyme and then chymotrypsinogen. We introduce the proteins in a different order for this training because here, in contrast to standard conditions, protein molecular weight is less significant than other protein properties. Because of complexity of the prediction, which involves multiple

input parameters together with significant interactions, we determine the most effective protein order by trial and error.

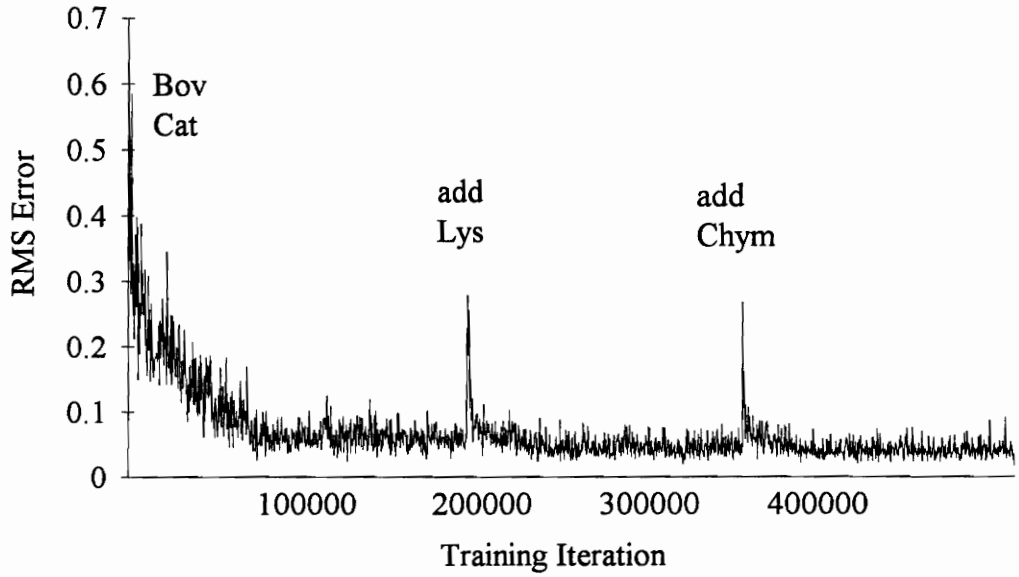
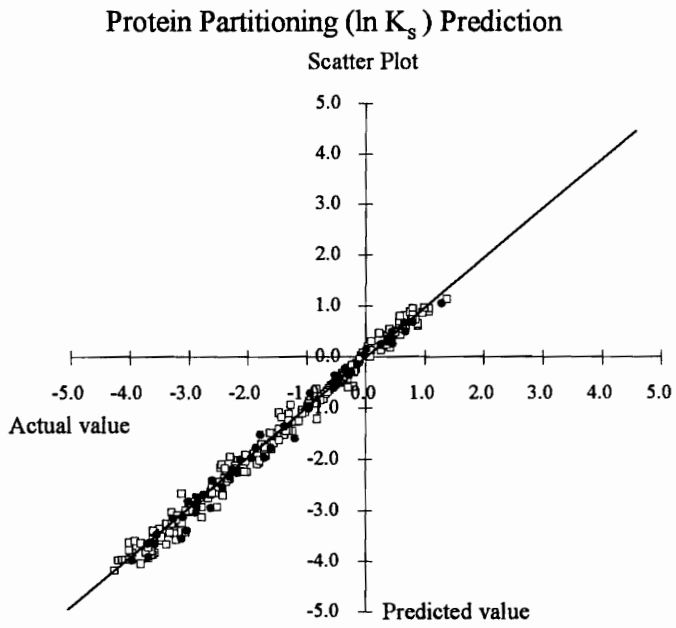


Figure 6.24. The training of protein-partitioning network for deviations from the isoelectric point, $\ln K_{pH}$.

b. Illustrative Predictions by the Protein-Partitioning Network

Finally, we turn our attention to the network testing. Figure 6.26 gives an example of the accurate predictions of the logarithm of the partition coefficient of the polymer solution at standard conditions (25°C and pH at the isoelectric point), $\ln K_s$. The network also performs well in predicting partition coefficients due to pH variation from the



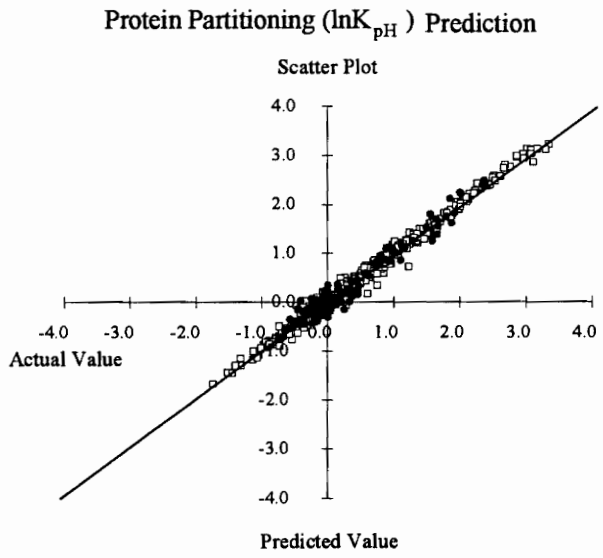


Figure 6.25. Illustrative predictions of protein-partitioning coefficients for polymer solutions at standard conditions, ln K_s , and for variations from the isoelectric point, lnK_{pH}: (●) generalization (□) training results. Data from Forciniti (1991).

isoelectric point, K_{pH}. Table 6.21 shows the correlation coefficients for predictions of both $\ln K_s$ and $\ln K_{pH}$, by the protein-partitioning network of Figure 6.14 . In both cases, the correlation coefficients are nearly unity, indicating that the network can accurately predict both $\ln K_s$ and $\ln K_{pH}$.

Table 6.21. The correlation coefficients for illustrative predictions of protein-partitioning coefficients for the polymer solutions at Standard Conditions, $\ln K_s$, and for variations from the isoelectric point, $\ln K_{pH}$ by the protein-partitioning neural network of Figure 6.14.

	Data points	Correlation $\ln K_S$	Data points	Correlation $\ln K_{pH}$
Training	235	0.994	770	0.991
Generalization	60	0.992	152	0.967
Combined	295	0.994	922	0.988

Concluding, we have found that the expanded set of protein properties used in our protein-partitioning network (Figure 6.14) has minimal effects on the prediction of partition coefficients, $\ln K_s$, when the pH is near the isoelectric point. However, the protein-property subnetwork becomes much more significant as the pH of solution deviates from the isoelectric point, i.e., in the prediction of the partition coefficient, $\ln K_{pH}$. Our initial network architecture includes only molecular weight in the protein-property subnetwork, which is consistent with the variable sets from the Ogston and Flory-Huggins models. The prediction capability for $\ln K_s$, is very similar for both the initial and

expanded protein-property subnetworks; both achieve correlation coefficients of approximately 0.99. In comparison, using the expanded set of variables in the protein-property subnetwork yields a significant increase in prediction capabilities of $\ln K_{pH}$, improving the correlation coefficient from approximately 0.92 for the initial set to 0.985.

D. Bioseparation and Process Optimizers : Optimization of Extractive Bioseparations in Aqueous Two-Phase Systems

1. Overview of Bioseparation and Process Optimizers

As Figure 6.13 shows, the three key components of our proposed approach are the bioseparation predictor (the neural network described in the previous section) , bioseparation optimizer (response-surface model), and process optimizer (expert system). This section describes how we combine the bioseparation optimizer and the process optimizer to provide a detailed overview of the entire extractive separation process. For protein systems with more than two components, the process optimizer uses heuristics for extractive separations to determine the optimal sequence for performing the desired component splits. The bioseparation optimizer provides an overview of the process within a given component split, showing a response-surface plot of partitioning behavior

over wide ranges of operating variables (Mw_{PEG}, $Mw_{Dextran}$, % PEG, % Dextran, and pH), as shown in Figure 6.26.

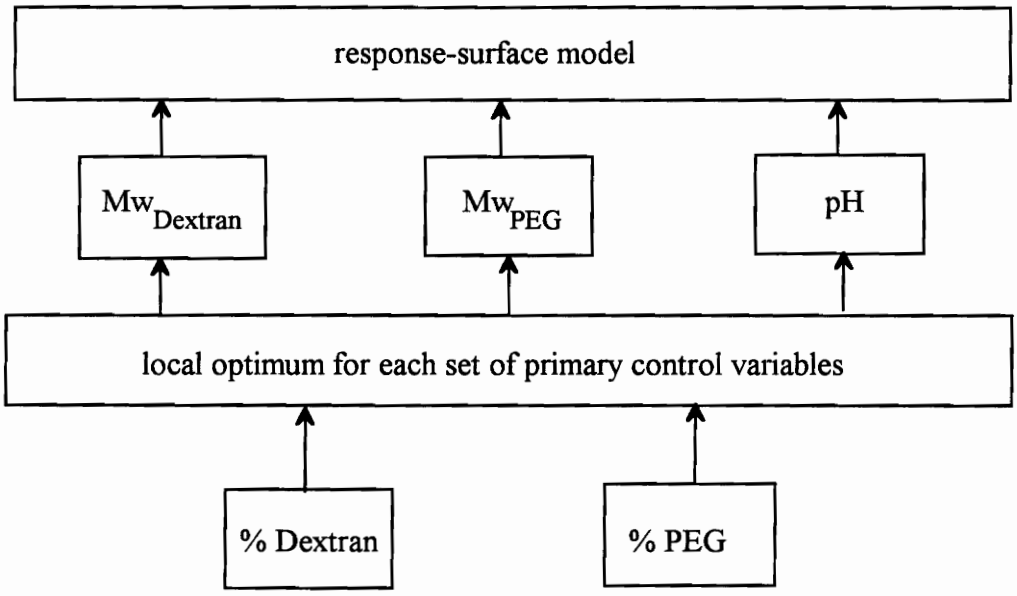


Figure 6.26. Response-surface modeling structure for determining optimal operating conditions of extractive bioseparations in ATPS of PEG and Dextran.

We use the overall architecture for the expert system shown in Figure 6.4 to develop our separation optimizers. Using predictions from the protein-partitioning network (Figure 6.14), we generate a numerical database consisting of a large number of component splits, $\ln K_p$, occurring over a wide range of operating conditions. We then convert this database to a knowledge base of separation efficiencies, applying a

classification algorithm. With the knowledge base in place, the expert system then uses its heuristics set to identify the optimal split sequences.

Previous research has identified many heuristics for separation sequencing (Liu, 1987; Wankat, 1990). For illustrative purposes, we will employ only the following heuristic for the separation sequencing of this system:

- ***Carry out easy separations first and most difficult separations last.***

We will use two rules to identify the effectiveness of a given separation:

(1) The selectivity, α_{ij} , is the ratio of partition coefficients and describes the extent to which two proteins will partition to opposite phases:

$$\alpha_{ij} = \frac{K_i}{K_j}$$

Selectivities of 1 represent infeasible splits; as the selectivity deviates from 1, the split becomes more desirable.

(2) Avoid having both components partitioning to the same phase.

The expert system applies these two rules directly to the database generated by the protein-partitioning network (Figure 6.14). In doing so, it classifies the selectivity of a component split as:

(1) very low
(2) low
(3) medium
(4) high
(5) very high

Similarly, it classifies the split's phase location as:

(1) poor : both proteins predominantly in one phase.
(2) medium : one or both proteins evenly split between phases.
(3) good : two proteins in opposite phases.

The system then determines separation efficiency based on a combination of these two parameters, and categorizes a given component split:

(1) infeasible
(2) poor separation efficiency
(3) fair separation efficiency
(4) good separation efficiency
(5) very good separation efficiency

Table 6.22 shows this categorization. The classifier is designed to err towards the infeasible class.

With all component splits classified, the expert system then rank-orders the separation efficiencies to determine the best sequences to carry out the separations.

Table 6.22. The separation-efficiency classifier from selectivity and phase classes.

Selectivity	Phase-partitioning class		
	poor	medium	good
very low	infeasible	infeasible	infeasible
poor	infeasible	infeasible	poor
medium	infeasible	fair	fair
high	poor	fair	good
very high	poor	good	very good

2. Development of the Bioseparation and Process Optimizers and an Illustrative Application

This section presents a step-by-step procedure for identifying the optimal separation conditions for multicomponent protein solutions. As an example, we shall optimize the extractive bioseparations for partitioning four proteins, chymotrypsinogen-A (Chym), lysozyme (Lys), bovine serum albumim (Bov), and catalase (Cat) in the PEG/Dextran ATPS .

step 1 : Set protein-system and polymer-solution constraints. The input parameters for protein properties that the user needs to enter include molecular weight and amino-acid compositions. Note that if experimental data for any properties such as secondary structure are available, use them in place of corresponding subnetwork predictions. All of the system parameters are internal to the expert network and are based on the boundaries used in training the neural networks. They can be changed by redefining the search space.

Input parameters:

- protein system

  (1) Lysozyme (Lys)
  (2) Chymotrypsinogen-A (Chym)
  (3) Bovine Serum Albumin (Bov)
  (4) Catalase (Cat)

System parameters:

- set molecular-weight search range

  (1) $Mw_{PEG} = \{ 4{,}000 \quad 6{,}000 \quad 10{,}000 \quad 20{,}000 \}$
  (2) $Mw_{Dextran} = \{ 10{,}000 \quad 40{,}000 \quad 110{,}000 \quad 500{,}000 \}$

- set pH search range :

  $pH = \{5, 6, 7, 8, 9\}$

- set compositions of the two polymers in the aqueous solution.

  Use the two extreme limits for % PEG and % Dextran, one at low concentrations near the plait point (i.e., low Δ PEG and Δ Dextran) and the other at high concentrations. See Figure 6.27 for a representation of where the concentration limits are chosen. Each phase diagram will have different compositions based on its actual shape and position on the ternary diagram.

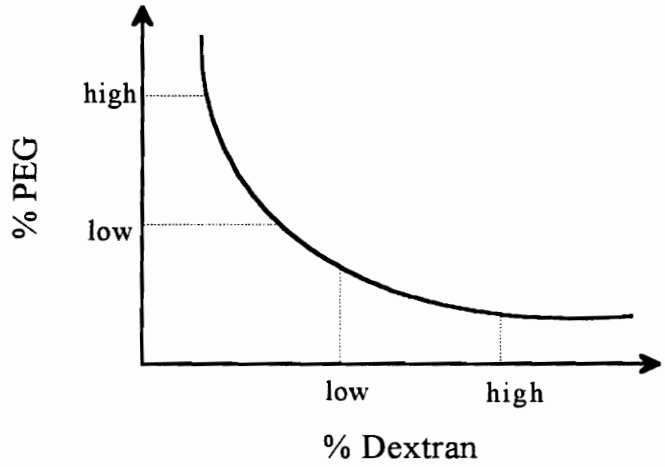


Figure 6.27. An illustration of PEG and Dextran concentration limits.

Step 2 : Initialize the search space for all partitioning possibilities occurring from the protein and polymer systems defined in step 1. To understand the magnitude of this problem, consider the system below, which has 160 experimental systems

to analyze, 640 partition coefficients to predict, and 480 component splits to consider.

- Calculate the number of experimental systems in search range :

$$\#\text{ experimental systems} = \#\,Mw_{PEG} * \#\,Mw_{Dextran} * \#\text{ pH values} * \#\text{ Compositions}$$

$$= 4 * 4 * 5 * 2 = 160 \text{ experimental systems}$$

- Calculate the number of potential separation splits in the search range :

$$\#\text{ component splits} = \#\text{ experimental systems} * [\#\text{ proteins} - 1]$$

$$= 160 * (4-1) = 480 \text{ component splits}$$

step 3 : Generate a protein-partitioning coefficient matrix which contains all of the system parameters at the levels defined in step 1.

For this case, we have the four-dimensional matrix, shown in Table 6.23, consisting of PEG and Dextran molecular weights, solution pH, and polymer concentration levels.

Table 6.23. Protein-partitioning coefficient matrix.

Low-polymer concentration

Mw : PEG		4,000					6,000					10,000					20,000				
Mw:Dextran	pH	5	6	7	8	9	5	6	7	8	9	5	6	7	8	9	5	6	7	8	9
10,000	Lys	1.46	1.09	0.93	0.80	0.61	1.04	0.92	0.87	0.80	0.65	1.19	0.94	0.89	0.90	0.73	1.43	0.98	0.74	0.61	0.50
	Chym	1.15	1.15	1.04	0.91	0.96	1.03	1.14	1.25	1.29	1.20	1.09	1.22	1.23	0.93	1.26	0.86	0.49	0.40	0.55	1.03
	Bov	0.11	0.06	0.06	0.08	0.20	0.26	0.21	0.21	0.30	0.53	0.26	0.23	0.20	0.21	0.40	0.05	0.05	0.05	0.05	0.05
	Cat	0.19	0.48	0.44	0.23	0.23	0.32	0.55	0.57	0.48	0.66	0.84	0.61	0.27	0.29	0.39	0.07	0.12	0.11	0.04	0.04
40,000	Lys	1.33	1.21	1.15	1.03	0.82	1.21	1.18	1.19	1.10	0.83	1.27	1.03	1.03	1.06	0.76	1.44	1.10	1.05	1.03	0.72
	Chym	1.23	1.23	1.24	1.19	1.25	1.07	0.87	0.78	0.85	0.93	0.84	0.88	0.65	0.49	0.89	1.16	0.98	0.65	0.67	1.12
	Bov	0.28	0.23	0.24	0.31	0.56	0.49	0.44	0.42	0.51	0.93	0.29	0.26	0.24	0.27	0.50	0.20	0.18	0.15	0.21	0.37
	Cat	0.13	0.21	0.23	0.34	0.56	0.28	0.42	0.42	0.50	0.60	0.42	0.15	0.15	0.25	0.86	0.19	0.04	0.03	0.06	0.86
110,000	Lys	1.74	1.27	1.19	1.15	0.76	1.37	1.51	1.21	0.87	1.08	1.47	1.28	1.26	0.89	0.70	1.62	1.46	1.46	1.29	0.79
	Chym	1.64	1.19	0.51	0.73	2.17	1.41	1.31	1.27	1.25	1.38	1.42	1.32	1.19	0.97	1.30	1.19	1.06	0.96	0.95	1.50
	Bov	0.28	0.21	0.17	0.21	0.66	0.68	0.52	0.42	0.60	1.74	0.31	0.22	0.18	0.26	0.63	0.17	0.12	0.09	0.19	0.34
	Cat	0.23	0.03	0.04	0.15	0.63	0.47	0.26	0.34	0.73	0.94	0.27	0.07	0.07	0.24	0.38	0.21	0.05	0.03	0.12	0.37
500,000	Lys	1.93	1.44	1.26	1.39	0.78	1.37	2.01	1.38	0.95	0.88	1.25	1.18	1.71	0.88	0.63	1.53	1.07	0.85	0.89	0.59
	Chym	1.32	1.25	0.94	0.95	0.92	2.10	1.13	0.72	0.77	1.24	1.39	0.86	0.57	0.55	0.98	1.21	1.69	1.05	0.94	1.10
	Bov	0.25	0.14	0.13	0.27	0.86	0.55	0.38	0.36	0.62	0.97	0.29	0.18	0.17	0.24	0.49	0.19	0.15	0.16	0.24	0.43
	Cat	0.29	0.02	0.09	0.43	0.73	0.23	0.08	0.20	0.46	0.84	0.24	0.03	0.09	0.26	0.41	0.18	0.02	0.04	0.16	0.32

High-polymer concentration

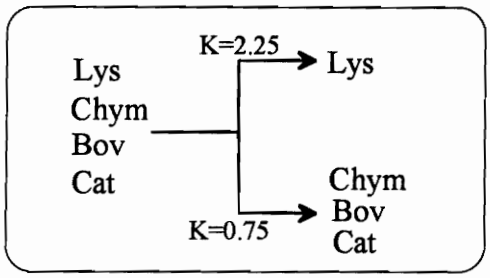
Mw : PEG		4,000					6,000					10,000					20,000				
Mw:Dextran	pH	5	6	7	8	9	5	6	7	8	9	5	6	7	8	9	5	6	7	8	9
10,000	Lys	1.93	1.37	1.08	0.76	0.45	2.00	1.29	1.01	0.86	0.58	1.70	1.01	0.67	0.53	0.45	1.86	1.36	0.91	0.53	0.27
	Chym	1.57	1.02	0.72	1.20	1.52	1.57	1.00	0.65	0.99	1.45	1.03	0.49	0.32	0.63	0.84	1.08	0.76	0.50	0.45	1.04
	Bov	0.02	0.01	0.01	0.02	0.04	0.05	0.04	0.04	0.05	0.13	0.05	0.04	0.04	0.04	0.04	0.04	0.06	0.07	0.06	0.05
	Cat	0.05	0.03	0.01	0.01	0.04	0.04	0.02	0.01	0.01	0.10	0.10	0.10	0.05	0.05	0.19	0.03	0.03	0.03	0.02	0.02
40,000	Lys	2.70	2.00	1.74	1.24	0.61	2.93	2.04	1.75	1.43	0.68	2.61	1.58	1.12	0.95	0.69	2.05	1.40	0.94	0.66	0.48
	Chym	1.88	1.15	0.92	1.80	1.98	1.84	1.43	1.30	2.20	2.16	1.82	1.73	1.16	1.32	1.35	1.31	0.74	0.37	0.35	1.14
	Bov	0.05	0.04	0.04	0.05	0.09	0.09	0.07	0.06	0.07	0.10	0.09	0.08	0.07	0.06	0.07	0.07	0.09	0.09	0.09	0.08
	Cat	0.05	0.01	0.01	0.01	0.37	0.05	0.01	0.01	0.01	0.36	0.06	0.02	0.01	0.01	0.19	0.03	0.03	0.02	0.02	0.24
110,000	Lys	2.85	1.96	1.73	1.39	0.80	2.69	1.78	1.46	1.32	0.74	3.09	1.92	1.33	1.01	0.76	2.25	1.42	0.94	0.72	0.57
	Chym	2.49	1.82	2.01	3.15	3.51	2.26	1.67	0.99	1.67	2.56	2.25	2.02	3.48	2.91	2.22	0.75	0.91	1.20	2.05	1.47
	Bov	0.08	0.06	0.05	0.05	0.10	0.14	0.11	0.09	0.09	0.15	0.10	0.08	0.06	0.05	0.07	0.05	0.04	0.03	0.02	0.02
	Cat	0.05	0.01	0.01	0.03	0.17	0.09	0.01	0.01	0.04	0.14	0.06	0.01	0.01	0.01	0.03	0.04	0.02	0.01	0.02	0.11
500,000	Lys	2.65	1.56	1.25	1.67	0.74	2.13	1.18	0.88	1.02	0.51	2.46	1.32	0.81	0.90	0.44	2.36	1.21	0.62	0.66	0.38
	Chym	1.77	1.33	1.80	1.56	1.38	2.75	2.47	3.19	1.89	1.24	1.64	1.47	1.79	1.48	1.40	0.98	0.83	0.89	1.35	1.47
	Bov	0.06	0.03	0.02	0.05	0.20	0.06	0.03	0.02	0.03	0.15	0.06	0.04	0.03	0.04	0.09	0.08	0.06	0.05	0.06	0.08
	Cat	0.05	0.01	0.02	0.07	0.20	0.09	0.01	0.03	0.07	0.18	0.04	0.01	0.01	0.03	0.03	0.03	0.01	0.01	0.03	0.05

step 4 : Identify all possible component splits from the protein-partitioning coefficient matrix, categorize all feasible experimental systems into component-split groups, and select the optimal separation within the group to represent that group for separation sequencing.

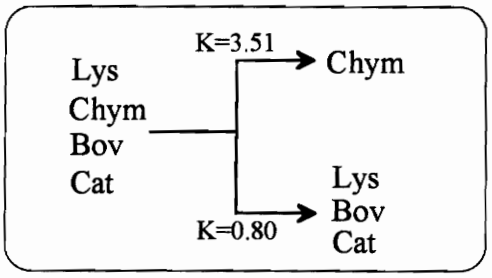
The given protein mixture has seven possible component splits, shown in Figure 6.28. Note that the infeasible component splits are only included for illustrative purposes. Figure 6.28 also shows the partition coefficients for the top and bottom phases, selectivity, and corresponding separation-efficiency class.

step 5 : Use the process optimizer to determine the optimal separation sequence of the protein mixture. As previously mentioned, for illustrative purposes, the only heuristics we use here are to carry out the easiest separations first and the most difficult ones last.

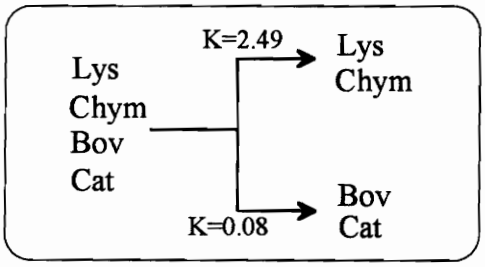
Since only component split (c) has a "good" separation efficiency, it should be carried out first, partitioning chymotrypsinogen-A and lysozyme to the top phase, and bovine serum albumin and catalase to the bottom phase. Only two possible component splits remain, one for the product of the top phase and one for the product of the bottom phase. The lysozyme/chymotrypsinogen-A splits (a) and (b) both have



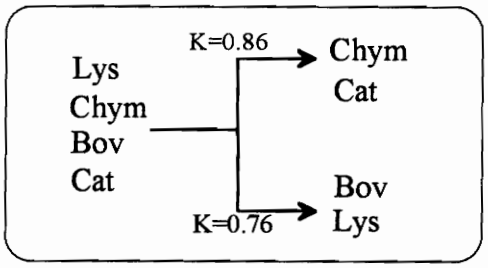
(a) $\alpha = 3.0$ (fair)



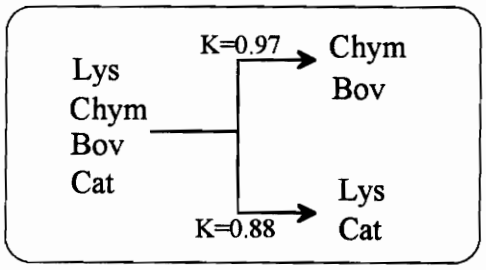
(b) $\alpha = 4.4$ (fair)



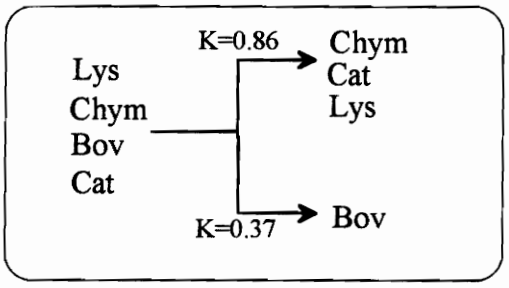
(c) $\alpha = 31.1$ (good)



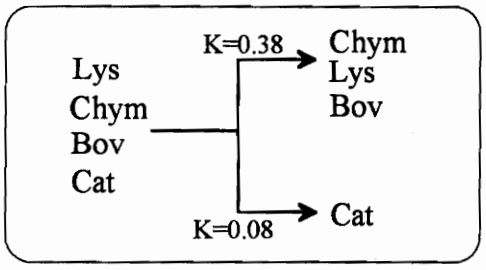
(d) $\alpha = 1.1$ (infeasible)



(e) $\alpha = 1.1$ (infeasible)



(f) $\alpha = 2.3$ (poor)



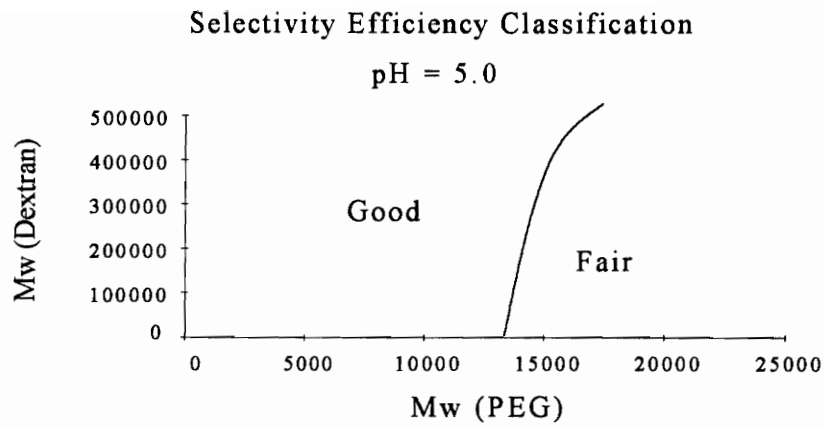
(g) $\alpha = 4.7$ (fair)

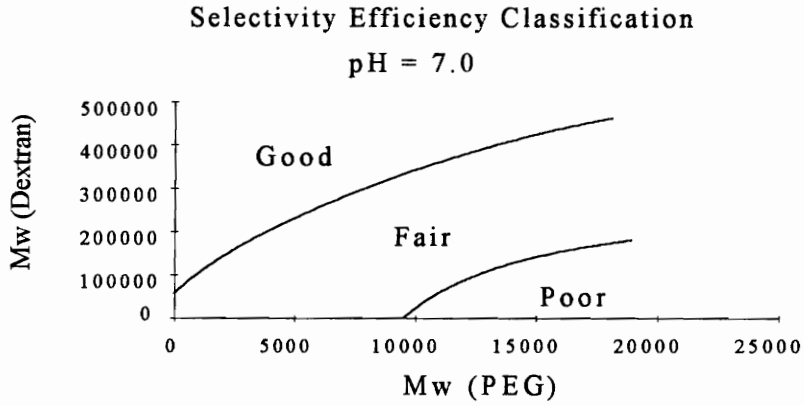
Figure 6.28. The possible component splits for the four-protein case study of chymotrypsinogen-A, lysozyme, bovine serum albumin, and catalase. $K > 1$ partitions to top-phase or PEG-rich phase and $K < 1$ partitions to bottom or Dextran-rich phase.

"fair" separation efficiencies, while each bovine serum albumin/catalase split (d, e, f, and g) has either an "infeasible" or "poor" efficiency. Therefore, this separation requires a special technique, such as bioaffinity partitioning, and will not progress to the bioseparation optimizer of step 6.

step 6 : Use the bioseparation optimizer to find the optimal operating conditions for each optimal component split identified in step 5 under each set of system parameters defined in step 1. The optimizer generates a response-surface plot for each component split, with the separation efficiency as the dependent variable. Any region where the experimental system does not yield the desired component split is classified as infeasible.

Figure 6.29 shows the response surfaces for the initial four component splits identified in step 5. Figure 6.30 shows the response surfaces for the lysozyme/chymotrypsinogen-A (top products) splits. The overall protein-partitioning flowsheet is shown in Figure 6.31, with the optimal operating conditions for each split listed. From the response-surface plots, we can easily identify the best operating conditions for the two separations: low pH, low Mw_{PEG}, high $Mw_{Dextran}$, and high-polymer-concentration for split 1 and high pH for split 2.





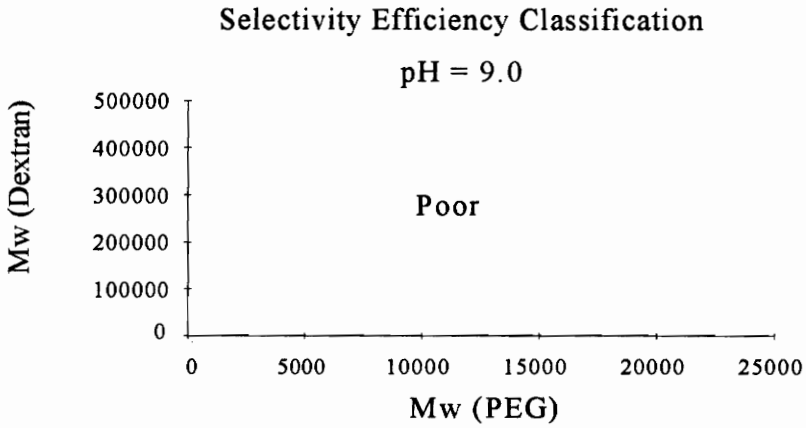


Figure 6.29. The response-surface plots generated from the bioseparation optimizer for split 1 of Figure 6.31.

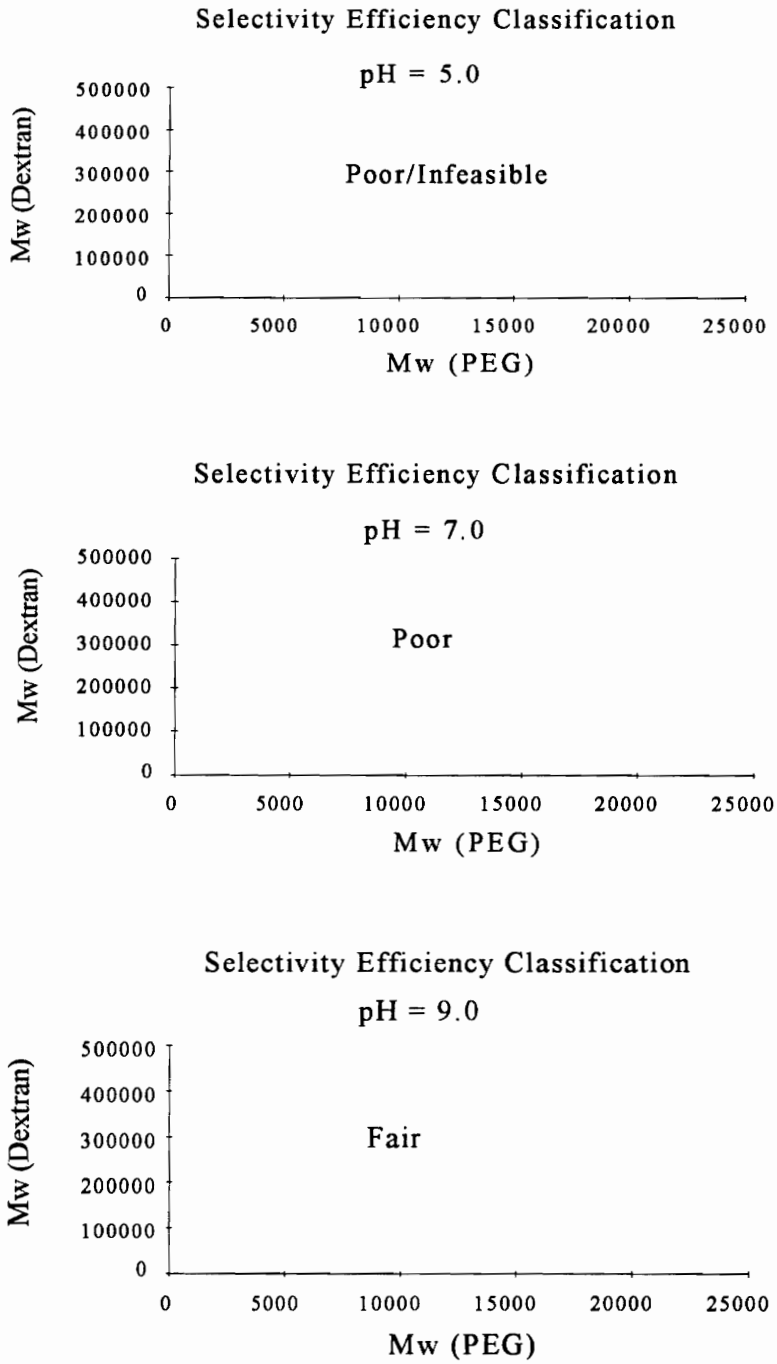


Figure 6.30. The response-surface plots generated from the bioseparation optimizer for split 2 of Figure 6.31.

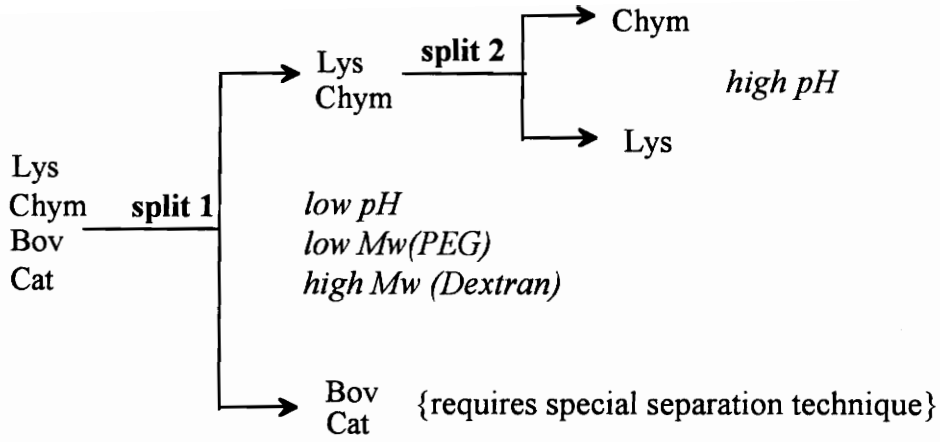


Figure 6.31. The overall protein-partitioning flowsheet.

To demonstrate using the bioseparation optimizer to quantitatively identify optimal separation conditions and thus facilitate experimental design and process development, we will now investigate the chymotrypsinogen-A/lysozyme partitioning (split 2 in Figure 6.31) further. Figure 6.32 illustrates that the predicted response-surface profiles for both proteins at the isoelectric point follow the experimental curves from Forciniti (1991) closely. Thus, our protein-partitioning network of Figure 6.14 represents a fairly accurate bioseparation predictor. Figure 6.32 also shows that proteins with dissimilar physical properties will have different response surfaces. Chymotrypsinogen-A has a linear response profile with respect to polymer molecular weights, while lysozyme has a hyperbolic response profile.

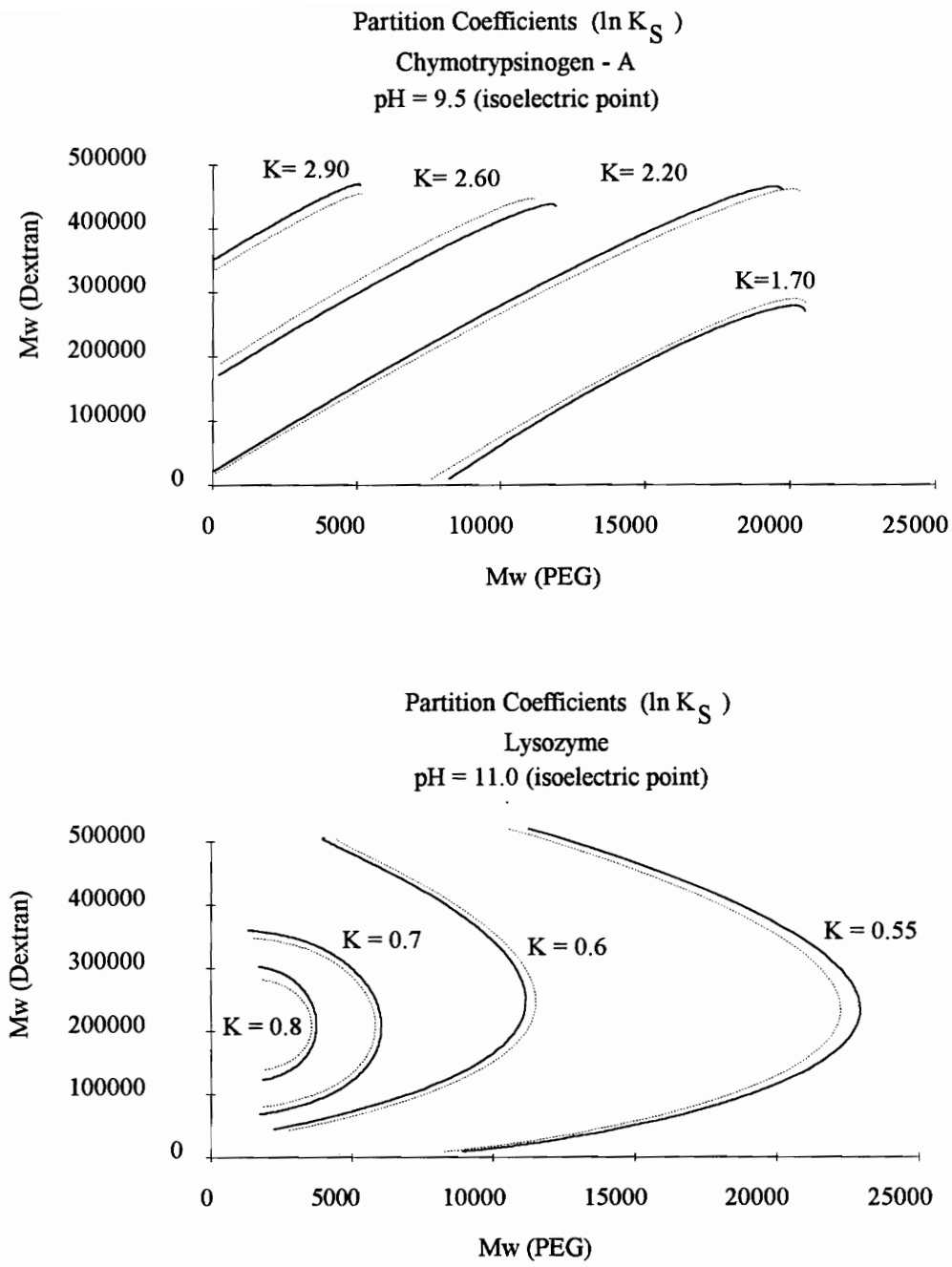
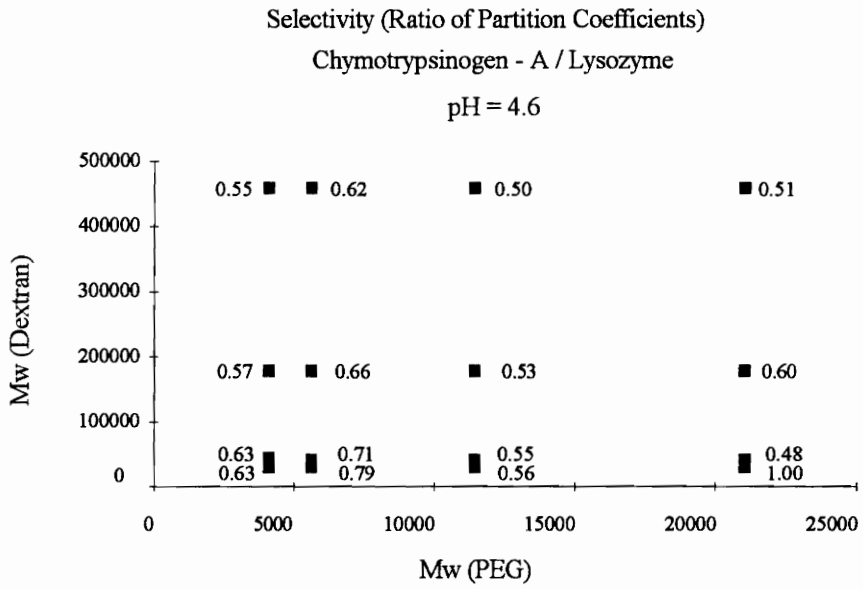


Figure 6.32. The experimental and predicted response-surface profiles of constant partition coefficients for protein partitioning in ATPS of PEG and Dextran of various molecular weights at the isoelectric point. Dashed curves represent experimental values by Forciniti (1991) and solid curves are predicted values by the protein-partitioning network.

A plot of the selectivity, or the ratio of partition coefficients in the two proteins, quantifies the sharpness of the protein split in a given ATPS. Figure 6.33 plots selectivity values for a mixture of chymotrypsinogen-A and lysozyme in an ATPS at the optimal operating conditions determined by the expert system (high pH), together with experimental values by Forciniti (1991). This response-surface plot represents the process more quantitatively fuzzy characterization of Figure 6.30. As seen in Figure 6.33, our strategy identifies the optimum ranges of PEG and Dextran molecular weights for maximizing the selectivity values within the high-pH response-surface plot.

(a)



(b)

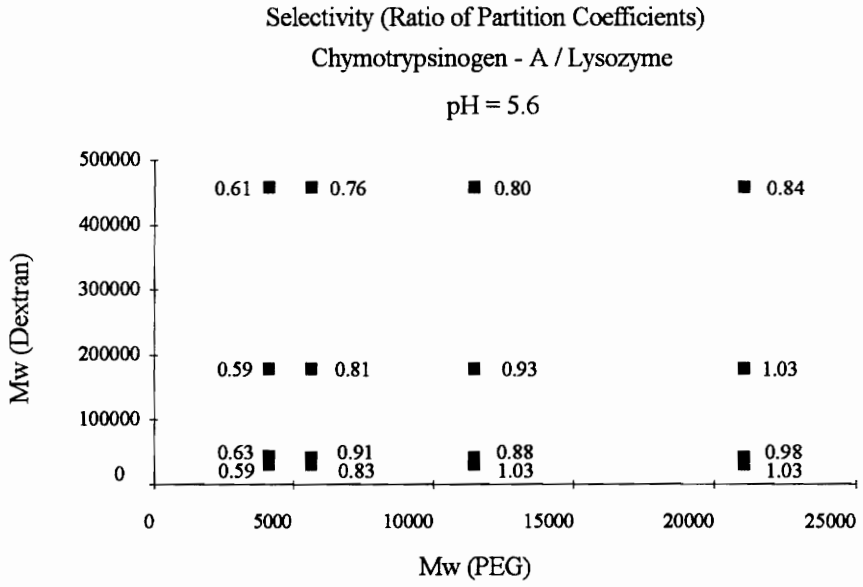
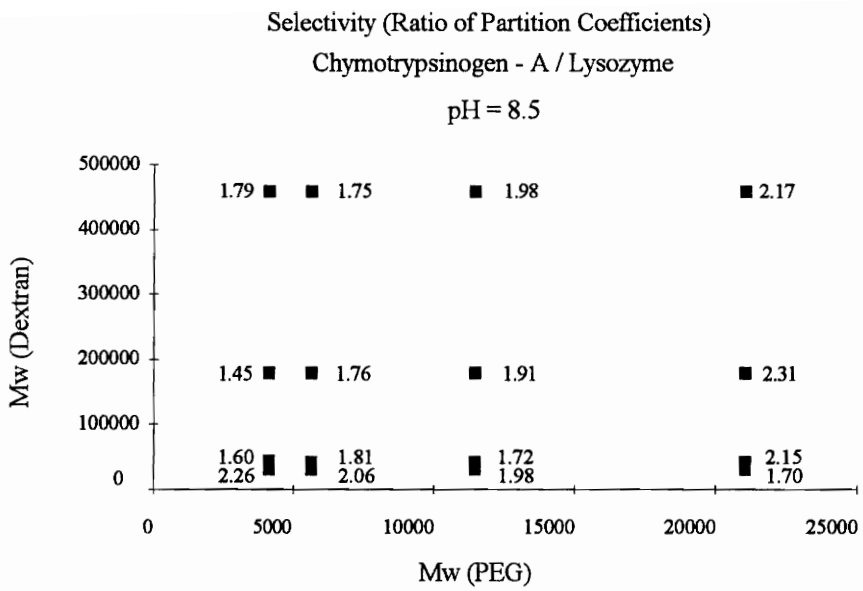


Figure 6.33a. The response-surface plots of selectivity values for partitioning of protein mixtures in ATPS of PEG and Dextran of various molecular weights at increasing pH values. Note the region of optimal operating conditions with highest selectivity values in (d).

(c)



(d)

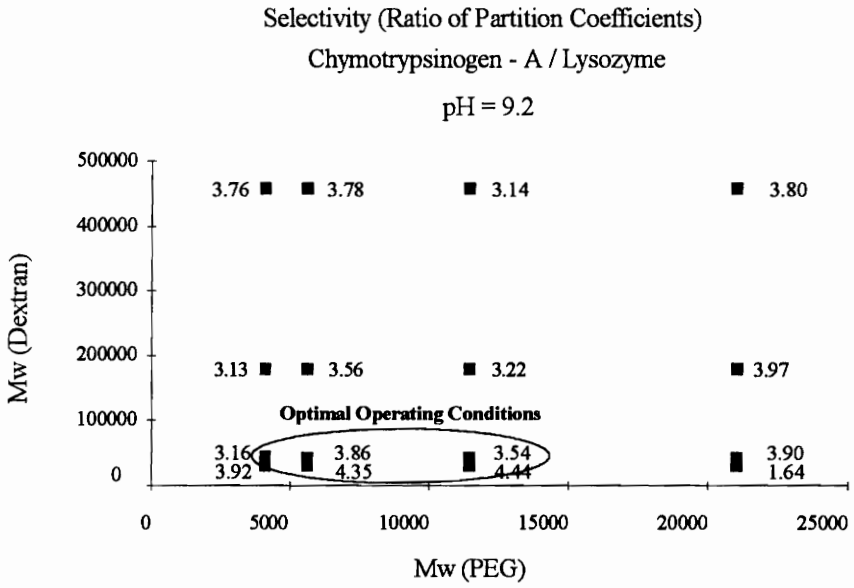


Figure 6.33b. The response-surface plots of selectivity values for partitioning of protein mixtures in ATPS of PEG and Dextran of various molecular weights at increasing pH values. Note the region of optimal operating conditions with highest selectivity values in (d).

E. Conclusions

Based on the bioseparation example considered in the Chapter, we can draw the following conclusions:

(1) The hierarchically-structured, protein-partitioning network of Figure 6.14 is very effective in developing a flexible approach to the predictive modeling and optimal design of extractive bioseparations in ATPS.

(2) Incorporating the expanded set of protein properties into our protein-partitioning network (Figure 6.14) has a minimal effect on predicting the partition coefficient, $\ln K_s$, when the pH is near the isoelectric point. However, the protein-property subnetwork significantly improves the prediction capabilities as the pH of solution deviates from the isoelectric point (i.e., $\ln K_{pH}$).

(3) The compartmentalized structure of the protein-partitioning network allows other contributing factors such as temperature, bioaffinity partitioning, and electrophoresis-induced partitioning to be added without reconstructing the entire model.

(4) The response-surface optimizer and the process optimizer provide user-friendly representations of the protein-partitioning characteristics (e.g., partition coefficient and selectivity) for a given protein mixture. Using them, we can identify the optimal operating regions with minimal experimentation.

6.3 Chapter Summary

- An expert network combines the qualitative reasoning functions of an expert system and the quantitative modeling capabilities of a neural network.
- Expert systems incorporate knowledge through facts, rules, and heuristics.
- A typical expert system consists of a knowledge base, an inference engine, and a user interface. The inference engine uses an inference mechanism to process information from the knowledge base and communicate the results to the user.
- One approach to developing an expert network places a neural network in series with a classifier and an expert system. The neural network predicts the desired response variables, which the classifier converts into categorical data. Finally, the expert system uses the fuzzy categorical data to make the required decisions.
- The illustrative case study on the bioseparation of proteins in aqueous two-phase systems shows how an expert network can be used to facilitate the preliminary design and development of biochemical processes.
    - The protein-partitioning neural network predicts the partition coefficient, lnK, over a wide range of protein properties, ion properties, and polymer-solution properties.
    - The bioseparation and process optimizer identifies the optimal separation conditions (e.g., Mw_{PEG}, $Mw_{Dextran}$, % PEG, % Dextran, and pH) for

multicomponent protein solutions to assist in experimental design and process development.

Nomenclature

C_{ij}	=	virial coefficients.
d	=	mean diameter of the ions.
F	=	Faraday constant.
H	=	Debye-Huckel parameter.
h	=	hydration numbers for anions and cations.
I	=	Ionic strength of solution.
K_b	=	partition contribution from affinity binding.
K_e	=	partition contribution from electrophoresis.
K_p	=	overall partition coefficient.
K_{pH}	=	partition contribution due to pH variation from the isoelectric point.
K_s	=	partition coefficient of the polymer solution at standard conditions. (temperature at 25°C and pH at the isoelectric point)
K_t	=	partition contribution due to the temperature variation from 25°C.
M_n	=	number-average molecular weight.
M_w	=	weight-average molecular weight.
m_i	=	ratio of the molar volume of component i to that of a reference component.

n_i	=	number of moles of species i.
P	=	pressure.
PI	=	polydispersity index, the ration of M_w to M_n.
R_i	=	radius of component i.
S	=	entropy.
T	=	temperature.
T_j	=	internal threshold value for processing element j.
V	=	volume.
v_i	=	valence of ion i.
w_i	=	weight fraction of species i.
w_{ji}	=	weighting factor from the ith input variable to the jth output variable.
x_i	=	input variable to a processing element.
y_j	=	output variable from a processing element.
z_b	=	net charge of the species.
α_i	=	proportionality factor between volume and weight fraction differences between the phases for species i
α	=	Debye-Huckel constant.
α_{ij}	=	selectivity, ratio of partition coefficients between components i and j.
β_{ij}	=	binary specific-interaction coefficient between solute molecules i and j.
χ_{ij}	=	Flory-Huggins interaction parameter between components i and j.

δ_{12} = solubility-parameter difference between PEG and Dextran.

δ_{1s} = solubility-parameter difference between PEG and water.

δ_{2s} = solubility-parameter difference between Dextran and water.

ΔDextran = gradient of Dextran between phases.

ΔG_{mix} = Gibbs free energy of mixing.

ΔH_{mix} = enthalpy of mixing.

ΔPEG = gradient of PEG between phases.

ΔS_{mix} = entropy of mixing.

$\Delta\psi$ = electrostatic potential difference between the phases.

ϕ_i = volume fraction of component i.

ϕ = ratio of weight fraction difference between phases for Dextran and PEG.

λ_j^o = standard state activity as defined by Guggenheim.

μ_i = chemical potential of component i.

Practice Problems

(6.1) Develop a neural network that can generate phase diagrams for PEG-Dextran aqueous two-phase systems using the polymer-solution network (Figure 6.15).

(a) Train and test the polymer-solution network using data files *dex.nna*, *dextop.nna*, *peg.nna*, and *pegbot.nna* (see Section 6.2.C.3.b).

(b) Create a testing data set that will obtain the required information for generating a series of phase diagrams using PEG 8000 with the four Dextran molecular weights (10,000; 40,000; 70,000; and 110,000). See Table 6.6 for polymer specifications. Note that the solubility parameters and temperature should not be altered from the training data set, and that the polymer compositions must be varied throughout the phase diagram.

(c) Generate four PEG-Dextran-water phase diagrams and compare the effect of Dextran molecular weight on the solution properties.

(d) How well does the network predict around the plate point (the point of the two-phase region where the compositions of the PEG-rich phase and the Dextran-rich phase approach each other, e.g., ΔPEG and ΔDextran are close to 0)?

(6.2) Find the best preliminary flowsheet design(s) for separating a solution of four proteins using the expert network of Section 6.2. The optimization should include both the process operating conditions (Mw_{PEG}, $Mw_{Dextran}$, pH, and polymer compositon) and separation sequence. The only data available for these four proteins are their amino-acid compositions (Table 6.P1).

Table 6.P1. The protein chain length and amino-acid compositions of four proteins to be separated.

Protein property	Protein A	Protein B	Protein C	Protein D
molecular weight	17,300	26,000	5,400	40,000
isoelectric point	6.5	10.5	4	8
# of amino acids in the protein	153	240	54	374
% alanine	9.1	7.1	12.9	7.4
% arginine	3.3	5.1	0.0	3.3
% asparagine	1.9	7.5	5.5	2.2
% aspartic acid	5.3	2.5	9.2	4.5
% cysteine	0.0	3.4	14.8	3.7
% glutamine	2.0	6.2	3.8	2.1
% glutamic acid	9.1	1.6	3.8	5.7
% glycine	7.8	10.4	7.4	10.2
% histidine	8.4	2.7	0.0	1.8
% isoleucine	5.2	4.1	11.2	6.5
% leucine	12.5	7.5	0.0	6.7
% lysine	12.5	1.2	1.8	8.0
% methionine	1.3	0.8	0.0	2.4
% phenylalanine	4.5	1.3	0.0	4.8
% proline	2.6	2.9	9.2	5.3
% serine	4.5	9.1	9.2	6.9
% threonine	3.2	7.9	0.0	6.4
% tyrptophan	1.4	3.0	0.0	0.5
% tyrosine	1.4	4.5	3.8	1.1
% valine	4.0	11.2	7.4	10.5

(a) Train and test the protein-partitioning network (Figure 6.14, and files: *part.nna* and *phpart.nna*), the polymer-solution network (Figure 6.15, and files: *dex.nna*, *peg.nna*, and *inttns.nna*), and the protein-property network (Figure 6.20, and files *ahlxtrn.nna* and *bshttrn.nna*).

(b) Use the protein-property network to predict the secondary structure of each protein: %α-helix and %β-sheet. Table 6.P1 contains the required amino acid chain length and compositions for this prediction.

(c) Use the Kidera vector transformations (Table 6.10) to obtain values of the Kidera parameters for pK_c, three hydrophobicities, and the partial specific volume, from the amino-acid compositions (Table 6.p1).

(d) Keep all ion properties at the same values as they are in file *part.nna*.

(e) Set all protein-system and polymer-solution constraints at the same levels listed in step 1, and initialize the search space similar to step 2 of Section 6.2.D.2.

(f) Follow the procedure listed in steps 3 to 6 of Section 6.2.D.2 to find the best preliminary flowsheet design(s) for separating the four proteins.

(6.3) Develop an expert network to identify the optimal operating conditions of the autoclave-curing process presented in Section 4.4.

(a) Categorize the two output response variables, final composite thickness L (cm) and the void size D_b (cm), into product quality classes (e.g., poor, satisfactory, good, and excellent). For example, to minimize void size we can assign the following categorizes: $D_b > 0.6$ cm (poor: outside of specification); $0.4 < D_b \leq 0.6$ cm (satisfactory: within specifications but close to a fault); $0.2 < D_b \leq 0.4$ cm (good: within acceptable limits); and $0.0 < D_b \leq 0.2$ cm (excellent).

(b) Modify files *thktrn.nna* and *voidtrn.nna* (Table 4.13) by replacing the numerical output values with output product quality classes, that can be later used by an expert system. Train a radial-basis-function network to predict the output product quality class for both composite thickness and void size.

(c) Generate a composite-thickness matrix similar to Figure 4.18 and a void-size matrix similar to Figure 4.20, except that the output responses are in classes that can be directly used by an expert system.

(d) Define an overall product quality (OPQ) variable which is a function of void size and composite thickness classes based on IF-THEN rules. For example, if D_b is excellent and L is excellent then OPQ is excellent.

(e) Use the rules of part d to combine the composite-thickness and void-size matrix into one OPQ matrix. Identify the optimal operating conditions based on the OPQ matrix.

References and Further Reading

Two good references on expert networks are Caudill (1990) and Gallant (1993). For applications of expert networks in bioprocessing and chemical engineering, key references include Ferrada et al. (1990), Bulsari and Saxen (1992a,b), Gehlen et al. (1992), Huang and Fan (1994), Renegaswamy and Venkatsubramanian (1992), Scott et al. (1992), and Venkatsubramanian and Stanley (1994). The popular biochemical engineering text by Baily and Ollis (1986, Chapters 2 and 11) provides an excellent discussion of protein structures and bioseparation processes. Albertsson (1986) and Walter et al. (1985) are two standard reference volumes on applications of aqueous two-phase systems (ATPS) in biotechnology, and both provide extensive discussions of the partitioning of proteins, biomolecules, and cell particles in ATPS. Cho (1989) describes the development of an expert system for protein partitioning in ATPS, and King (1992) gives an excellent review of aqueous two-phase partitioning in biotechnology.

Albertsson, P. A. *Partition of Cell Particles and Macromolecules*, 3rd edition, Wiley, New York (1986).

Asenjo, J.A. and I. Patrick, "Large-Scale Purification," Chapter 1, pp. 1-28, in *Protein Purification Applications : A Practical Approach*, IRL Press, Oxford, United Kingdom (1990).

Baily, E.J. and D.F. Ollis, *Biochemical Engineering Fundamentals*, 2nd edition, McGraw Hill, New York (1986).

Barton, A. F. M., Editor, *Handbook of Solubility Parameters and Other Cohesion Parameters*, CRC Press, Boca Raton, FL (1983).

Baskir, J. N., T. A. Hatton and V. W. Suter, "Thermodynamics of the Separation of Biomaterials in Two-Phase Aqueous Polymer Systems: Effects of Phase-Forming Polymers," *Micromolecules*, **20**, 1300 (1987).

Baskir, J. N. and T. A. Hatton, "An Overview of Theoretical Developments for Protein Partitioning in Aqueous Two-Phase Polymer Systems," pp. 217-227, in *Separations Using Aqueous Phase Systems: Applications in All Biology and Biotechnology*, D. Fisher and I. A. Sutherland, Editors, Plenum Press, New York (1989).

Baughman, D. R. and Y. A. Liu, "An Expert Network for Bioseparation Process Synthesis: Predictive Modeling of Protein Partitioning in Aqueous Two-Phase Systems," *ACS Separation Science and Technology Award Symposium on Separation Syntheses, 205th Annual Meeting*, American Chemical Society, Denver, CO, April (1993).

Baughman, D. R. and Y. A. Liu, "Optimal Design of Extractive Bioseparations in Aqueous Two-Phase Systems," *Symposium on Aqueous Biphasic Separations: Biomolecules to Metal Ions*, American Chemical Society, San Diego, CA, March (1994a).

Baughman, D. R. and Y. A. Liu, "An Expert Network for Predictive Modeling and Optimal Design of Extractive Bioseparations in Aqueous Two-Phase Systems," *Ind. Eng. Chem. Res.*, **33**, 2668, (1994b).

Becraft, W. R. and P. L. Lee, "An Integrated Neural Network/Expert System Approach to Fault Diagnosis," *Comput. Chem. Eng.*, **17**, 1001 (1993).

Benge, C. G., "A comparison of Thermodynamic Models for the Prediction of Phase Behavior in Aqueous Polymer Two-Phase Systems," M.S. Thesis, Virginia Tech, Blacksburg, VA, Aug. (1986).

Branden, C. and T. Tooze, *Introduction to Protein Structure*, Garland Publishing, Inc., New York (1991).

Brunet, J. C. and Y. A. Liu, "Studies in Chemical Process Design and Synthesis: 10. An Expert System for Solvent-Based Separation Process Synthesis," *Ind. Eng. Chem. Res.*, **32**, 315 (1993).

Bulsari, A. B. and H. Saxen, "A Fuzzy Expert System for Continuous Steel Casting Diagnosis in an Artificial Neural Network," pp. 26-27, in *Proceedings of the 75th Steelmaking, 51st Ironmaking and 10th Process Technology Conferences*, Toronto, Canada, April (1992a).

Bulsari, A. B. and H. Saxen, "Implementation of a Chemical Reactor Selection Expert System in an Artificial Neural Network," *Eng. Applic. Artif. Intell.*, **5**, 113 (1992b).

Bulsari, A. B., A. Krastawski and H. Saxen, "Implementing a Fuzzy Expert System in an Artificial Neural Network," *Comput. Chem. Eng.*, **16**, 5405 (1992).

Cabezas, H., Jr., M. Kabiri-Badr, S. M. Snyder and D. C. Szlag, "Thermodynamics of Aqueous Mixtures of Polymers and Salts," pp. 380-391, in *Proceedings of Frontiers in Biochemistry II*, Boulder, CO, June (1990), Published by American Chemical Society, Washington, DC (1992).

Caudill, M., "Expert Networks," Chapter 9, in *Neural Network PC Tools: A Practical Guide*, R. C. Eberhart and R. W. Dobbins, Editor, Academic Press, San Diego, CA (1990).

Cho, W. S., "Application of Expert System to Bioseparation: The Partition of Proteins in Aqueous Polymer System," Ph.D. Dissertation, Chemical Engineering, Univ. of Illinois at Chicago, IL (1989).

Chou, P. Y. and G. D. Fasman, "Conformational Parameters for Amino Acids in Helical, Sheet and Random Coil Regions from Proteins, *Biochem.* **13**, 211 (1974a).

Chou, P. Y. and G. D. Fasman, "Prediction of Protein Conformation," *Biochem.*, **13**, 222 (1974b).

Clark, W. M., "Thermodynamics of Protein Partitioning in Aqueous Two-Phase Systems," pp. 147-181, in *Chemical Engineering Problems in Biotechnology*, M. L. Shuler, Editor, American Institute of Chemical Engineers, New York (1989).

Connemann, M., J. Gaube, A. Pfenning and M. Stumpf, "Accurate Prediction of Ternary Liquid-Liquid Equilibria from Vapor-Liquid Equilibrium Measurements of Aqueous Polymer Solutions," *AIChE Annual Meeting*, Miami, FL, November (1992).

Cox, E., "Fuzzy Fundamentals," *IEEE Spectrum*, p. 58, October (1992).

Diamand, A. D., "Fundamental Studies of Biomolecule Partitioning in Aqueous Polymer Two-Phase Systems," Ph.D. thesis, Lehigh University, Bethlehem, PA (1990).

Diamand, A. D. and J. T. Hsu, "Protein Partitioning in PEG/dextran aqueous two-phase systems," *AIChE J.*, **36**, 1017 (1990).

Dubchak, I., S. R. Holbrook and S. H. Kim, "Prediction of Protein Folding Class from Amino Acid Composition," *Proteins: Structure Function and Genetics*, **16**, 79, May (1993).

Fair, J. R., "Commercially Attractive Bioseparation Technology," *Chem. Eng. Prog.*, **85**, No. 12, 38 (1989).

Ferrada, J. J., P. A. Grizzaffi and I. W. Osborne-Lee, "Applications of Neural Networks in Chemical Engineering - Hybrid Systems," *AIChE Annual Meeting*, Chicago, IL, November (1990).

Ferran, E. A. and P. Ferrara, "Topological Maps of Protein Sequences," *Bio. Cybernetics*, **65**, 451 (1991).

Fisher, D. and I. A. Sutherland, Editors, *Separations Using Aqueous Phase Systems: Applications to Cell Biology and Biotechnology*, Plenum Press, New York (1989).

Flory, P.J., *Principles of Polymer Chemistry*, Cornell University Press, Ithaca, NY (1953).

Forciniti, S. C., "A Theoretical and Experimental Study of Protein Partitioning in Aqueous Two-Phase Systems," Ph.D. Dissertation, North Carolina State University, Raleigh, NC (1991).

Forciniti, D. and C. K. Hall, "Theoretical Treatment of Aqueous Two-Phase Extraction by Using Virial Expansions: A Preliminary Report," *ACS Symp. Ser.*, **419**, 53 (1990).

Forciniti, D., C. K. Hall, and M. R. Kula, "Protein Partitioning at the Isoelectric Point: Influence of Polymer Molecular Weight and Concentration and Protein Size," *Biotech. and Bioeng.*, **38**, 986 (1991).

Gallant, S. J., "Connectionist Expert Systems," *Commun. ACM*, **31**, 2 (1988).

Gallant, S. I., "A Connectionist Learning Algorithm with Provable Generalization and Scaling Bounds," *Neural Networks*, **3**, 191 (1990).

Gallant, S. I., *Neural Network Learning and Expert Systems*, MIT Press, Cambridge, MA (1993).

Gallant, S. I. and Y. Hayashi, "A Neural Network Expert System with Confidence Measurements," *Third Intern. Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, Paris, France, July (1990).

Geglen, S., H. Tolle, J. Kreuzig, and P. Friedl, "Integration of Expert Systems and Neural Networks for the Control of Fermentation Processes," pp. 379-382, in *Modeling and Control of Biotechnical Processes 1992*, N. Karim and G. Stephanopoulos, Editors, Pergaman Press, Oxford, United Kingdom (1992).

Hall, C., "Neural Net Technology : Ready for Prime Time ?," *IEEE Expert*, **7**, No. 6, 2 (1992).

Haynes, C. A., "Separation of Protein Mixtures by Extraction: Thermodynamic Properties of Aqueous Two-Phase Polymer Systems Containing Salts and Proteins," Ph.D. Dissertation, University of California, Berkeley (1992).

Haynes, C. A., H. W. Blanch and J. M. Prausnitz, "Separation of Protein Mixtures by Extraction: Thermodynamic Properties of Aqueous Two-Phase Polymer Systems Containing Salts and Proteins," *Fluid Phase Equilibria*, **53**, 463 (1989).

Haynes, C. A., F. J. Benitez, H. W. Blanch and J. M. Prausnitz, "Application of Integral-Equation Theory to Aqueous Two-Phase Partitioning Systems," *AIChE J.*, **39**, 1539 (1993).

Hecht-Nielsen, R., "Hierarchical Networks," pp. 198-218, in *Neurocomputing*, Addison-Wesley, Reading, MA (1990).

Hendler, J., "Editorial: On the Need for Hybrid Systems," *Connection Science*, **1**, 227 (1989a).

Hendler, J., "Marker-passing over Microfeatures: Towards a Hybrid Symbolic/Connectionist Model," *Cognitive Science*, **13**, 79 (1989b).

Hill, T. L., "Theory of Solutions," *J. Am Chem. Soc.*, **79**, 4885 (1957).

Huang, Y. L. and L. T. Fan, "HIDEN: A Hybrid Intelligent System for Synthesizing Highly Controlled Exchanger Networks. Implementation of a Distributed Strategy for Integrating Process Design and Control," *Ind. Eng. Chem. Res.*, **33**, 1174 (1994).

Hunter, L. and D. J. States, "Bayesian Classification of Protein Structure," *IEEE Expert*, **7**, No. 4, 23 (1992).

Hustedt, H., K.-H. Kroner and N. Papamichael, "Continuous Crosscurrent Extraction in Process Scale," pp. 299-307, in *Separations Using Aqueous Phase Systems: Applications to Cell Biology and Biotechnology*, Plenum Press, New York (1989).

Ichikawa, S., M. Imai and M. Shimizu, "Solubilizing Water Involved in Protein Extraction Using Reversed Micelles," *Biotech. and Bioeng.*, **39**, 20 (1992).

Johansson, G. and F. Tjerneld, "Affinity Partition between Aqueous Phases: A Tool for Large-Scale Purification of Enzymes," *J. Biotech.*, **11**, 135 (1989).

Jolliffe, I. T., *Principle Component Analysis*, Spinger-Verlag, New York (1986).

Kamihira, M., R. Kaul and B. Mattiasson, "Purification of Recombinant Protein A by Aqueous Two-Phase Extraction Integrated with Affinity Precipitation," *Biotech. and Bioeng.*, **40**, 1381 (1992).

Khuri, A. I. and J. A. Cornell, *Response Surfaces : Designs and Analyses*, Marcel Dekker, Inc., and ASQC Quality Press, New York (1987).

Kidera, A., Y. Yonishi, M. Oka, T. Ooi and H. A. Scheraga, "Statistical Analysis of the Physical Properties of the 20 Naturally Occurring Amino Acids," *J. Protein Chem.*, **4**, 23 (1985a).

Kidera, A., Y. Konishi, T. Ooi and H. A. Scheraga, "Relation between Sequence Similarity and Structural Similarity in Proteins: Role of Important Properties of Amino Acids," *J. Protein Chem.*, **4**, 265 (1985b).

King, R. S., "Aqueous Two-Phase Partitioning in Biotechnology," Chapter 7, pp. 204-243, in *Polymer Applications in Biotechnology*, D. S. Soane, Editor, Prentice-Hall, Englewood Cliffs, NJ (1992).

King, R. S., H. W. Blanch and J. M. Prausnitz, "Molecular Thermodynamics of Aqueous Two-Phase Systems for Bioseparations," *AIChE J.*, **34**, 1585 (1988).

Kraslawski, A., T. Koiranen and L. Nystrom, "Concurrent Engineering: Robust Design in Fuzzy Environment," *Proceedings of the European Symposium on Computer-Aided Process Engineering - 2*, S477, Toulouse, France, Oct. (1992).

Kula, M. R., K.-H. Kroner and H. Hustedt, "Purification of Enzymes by Liquid-Liquid Extraction," *Adv. Biochemical Eng.*, **24**, 73 (1982).

Liu, Y. A., "Process Synthesis: Some Simple and Practical Developments," Chapter 6, pp. 147-260, in *Recent Developments in Chemical Processes and Plant Design*, Y. A. Liu, H. A. McGee, Jr. and W. R. Epperly, Editors, Wiley, New York (1987).

Mahaderan, H. and C. K. Hall, "Statistical-Mechanical Model of Protein Precipitation by Nonionic Polymer," *AIChE J.*, **36**, 1517 (1990).

Mahaderan, H. and C. K. Hall, "Theory of Precipitation of Protein Mixtures by Nonionic Polymers," *AIChE J.*, **38**, 573 (1992).

Mavrovouniotis, M. L. and S. Chang, "Hierarchical Neural Networks," *Comput. Chem. Eng.*, **16**, 347 (1992).

Michaels, A. S., "Frontiers in Bioprocessing: An Assessment," pp. 1-8, in *Proceedings of Frontiers in Biochemistry II*, Boulder, CO, June (1990), American Chemical Society, Washington, DC (1992).

Muskal, S. M., "Predicting Features of Protein Structure with Computer Simulated Neural Networks," PhD Dissertation, University of California, Berkeley (1991).

Muskal, S. M. and S. H. Kim, "Predicting Protein Secondary Structure-Content: A Tandem Neural Network Approach," *J. Mol. Biology*, **225**, 713 (1992).

NeuralWare, Inc., *Neural Computing : A Technology Handbook for Professional II/PLUS and NeuralWorks Explorer*, Building 4, Suite 227, Penn Center West, Pittsburgh, PA 15276, phone (412) 787-8222 and FAX (412) 787-8220 (1993).

Ogston, A. J., "Some Thermodynamic Relationships in Ternary Systems, with Special Reference to the Properties of Systems Containing Hyaluronic Acid and Protein," *Arch. Biochem. Biophys.*, **Suppl. 1**, 39 (1962).

Qian, N. and T. J. Sejnowski, "Predicting the Secondary Structure of Globular Proteins Using Neural Network Models," *J. Mol. Biol.*, **202**, 805 (1988).

Quantrille, T. E. and Y. A. Liu, *Artificial Intelligence in Chemical Engineering*, Academic Press, San Diego, CA (1991).

Renegaswamy, R. and V. Venkatasubramanian, "An Integrated Framework for Process Monitoring, Diagnosis, and Control Using Knowledge-Based Systems and Neural Networks," *IFAC Symposium on On-Line Fault Detection and Supervision in the Chemical Process Industries*, Newark, DE, April (1992).

Rost, B. and C. Sander, "Predicting Protein Secondary Structure by Better than 70-Percent Accuracy," *J. Mol. Bio.*, **232**, 584 (1993).

Rumelhart, D. E. and J. L. McClelland, Editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Volume I - Foundations*, MIT Press, Cambridge, MA (1986).

Scott, G. M., J. W. Shavlik and W. H. Ray, "Refining PID Controllers Using Neural Networks," *Neural Computation*, **4**, 746 (1992).

Schneider, G. and P. Wrede, "Development of Artificial Neural Filters for Pattern Recognition in Protein Sequences," *J. Mol. Evolution,* **36**, 586 (1993).

Stryer, L., *Biochemistry*, 3rd edition, W. H. Freeman and Company, New York (1988).

Tjerneld, F., G. Johansson and M. Joelsson, "Affinity Liquid-Liquid Extraction of Lactate Dehydrogenase on a Large Scale," *Biotech. and Bioeng.*, **30**, 809 (1987).

Venkatsubramanian, V. and S. Kavuri, "Intelligent Systems for Integrated Process Operations Management: Goals and Trends," *7th Intern. Conf. on Artificial Intelligence in Engineering*, Waterloo, Canada, July (1992).

Venkatsubramanian, V. and G. M. Stanley, "Integration of Process Monitoring, Diagnosis and Control : Issue and Emerging Trends," pp. 179-206, in *Foundations of Computer-Aided Process Operations*, D. W. T. Rippen, J. C. Hale and J. F. Davis, editors, CACHE Corporation, Austin, TX (1994).

Vlachy, V., H. W. Blanch and J. M. Prausnitz, "Liquid-Liquid Phase Separations in Aqueous Solutions of Globular Proteins," *AIChE J.*, **39**, 215 (1993).

Walter, H. W., D. F. Brooks and D. Fisher, Editors, *Partitioning in Aqueous Two-Phase Systems: Theory, Methods, Uses and Applications to Biotechnology*, Academic Press, Orlando, FL (1985).

Wankat, P. C., *Rate-Controlled Separations*, Section 14.5, "Heuristics," pp. 824-834, Elsevier Applied Sciences, New York, NY (1990).

Zhang, X., J. P. Mesirov and D. L. Waltz, "Hybrid System for Protein Secondary Structure Prediction," *J. Mol. Biology*, **225**, 1049 (1992).

Glossary

This glossary section has been compiled from the following references:

Caudill, M. and C. Butler, Naturally Intelligent Systems, MIT Press, Cambridge, MA (1990).

Darby, N. J. and T. E. Creighton, Protein Structure, IRL Press, Oxford, United Kingdom (1993).

Eberhart, R. C. and R. W. Dobbins, Neural Network PC Tools: A Practical Guide, Academic Press, San Diego, CA (1990).

NeuralWare, Inc., Neural Computing: A Technical Handbook for Professional II/PLUS and NeuralWorks Explorer, Pittsburgh, PA (1993).

Ollis, D. F. and J. E. Baily, Biochemical Engineering Fundamentals, 2nd edition, McGraw Hill, New York (1986).

Quantrille, T. E. and Y. A. Liu, Artificial Intelligence in Chemical Engineering, Academic Press, San Diego, CA (1991)

VerDuin, W. H., Better Products Faster: A Practical Guide to Knowledge-Based Systems for Manufacturers, Irwan Professional Publishing, Burr Ridge, IL (1995).

Accretive Associative Memory: An associative memory that responds to unfamiliar data by returning the nearest stored data item.

Activation Function: the function that determines the level of excitement or activity of a node for a given input.

Activity: The total level of activity of a neural network or of some portion of the network (such as the activity of a single layer of nodes). Activity is measured by the value of the activation function of each node in a network or layer.

Adaline: acronym for adaptive linear element, one of the first adaptive filter networks, developed by Bernard Widrow and Ted Hoff of Stanford in 1960..

Adaptive: a system that adapts or modifies its behavior to suit changing circumstances. Specifically, the neural network has the ability to modify itself during operation, through adjusting weight factors, w_{ij}.

Adaptive Control: a form of process control that attempts to adjust controller tuning parameters in response to process dynamics.

Adaptive Filter Network: a neural network that acts as a filtering system to classify input patterns into categories.

Adaptive Resonance Theory: a self-organized learning system that is one of the best models for psychological learning phenomena, used for pattern classification without an external teacher.

Adsorption Models: a statistical thermodynamic model based on the adsorption of polymers from dilute solution to spherical solid particles.

Aerobic Fermentation: fermentation taking place in the presence of oxygen.

α-Helix: the regular helical conformation of polypeptide chain that has a hydrogen bond between the NH of each residue and the carbonyl of the residue fourth along in the primary structure.

α Proteins: predominantly α-helix regions with little or no β-sheets.

α+β proteins: α-helices and β–sheets clustered in separate domains.

α/β proteins: alternating α-helices and β–strands.

AI: artificial intelligence.

Algorithm: a formal procedure that specifies a step-by-step execution path such that a correct or optimal answer results at a predefined and fixed stopping points. Importantly, if the algorithm is followed exactly, the result is guaranteed.

Anaerobic Fermentation: fermentation taking place in the absence of oxygen.

Architecture: the arrangement of nodes and interconnections of the nodes in a neural network. It also refers to how the network learns.

ART Network: a neural network built following the principles of adaptive resonance theory. A learning system introduced by Stephen Grossberg and Gail Carpenter that is truly self-organizing and serves as one of our best models to date of many psychological learning phenomena.

ART 1 Network: the first version of ART that can process only binary input data patterns.

ART 2 Network: the second version of ART that can process gray-scale input data patterns. see also gray scale.

Artificial Intelligence: the part of computer science concerned with designing intelligent computer systems, that is, systems that exhibit characteristics we associate with intelligence in human behavior. In practical terms, artificial

intelligence deals with symbolic, non-algorithmic methods of problem-solving.

Associative Memory: a memory system that stores information by associating or correlating it with other stored information.

Attentional Signal: a specific signal to a node or network used to instruct the node to treat an input stimulus in a special way. Often used as a cue for training.

Autoassociative Memory: an associative memory in which a stored data item is associated or correlated with itself.

Autonomous Learning: learning characterized by trial and error, with no teacher assistance, that occurs selectively- only important information is learned; see also unsupervised learning; self-organization.

Avalanche Network: a network using a series of outstar structures for learning or execution of patterns, especially spatiotemporal patterns. See also outstar avalanche.

Axon: the output of a biological neuron (node) over which signals are sent to other neurons.

Axon Collateral: one of many branches of an axon.

Backpropagation Learning: a type of supervised, error-correction learning in neural networks where an error on the output layer, ε, is calculated, and that error is propagated backwards through the network to determine how each individual weight factor contributes to the output error. Based on each connection

weight's contribution to the error, that weight is adjusted to minimize the total output error.

Batch Process: a production process for which a characteristic weight or volume can be associated, on the basis that the weight or volume of the product is processed as a single entity.

Basin of Attraction: an area of the energy surface of a network in which all network states will converge to a particular network state that represents some item of stored information. Similar in concept to an energy well in physics.

β proteins: predominantly β-sheets with minimal or no α-helix regions.

β-Sheet: collection of β-strands lying side-by-side, linked together in either a parallel or anti-parallel manner by hydrogen bonds between their backbone NH and carbonyl groups.

β-Strand: segment of polypeptide chain with an extended conformation that is part of a β-sheet.

Bias Function: an internal threshold value that adds a fixed amount to the nodal summation. Typically, a bias function adds 1 to the nodal summation. Bias functions are in contrast with more typical internal threshold, where we subtract the threshold value from the nodal summation.

Bidirectional Associative Memory (BAM): a crossbar or correlation matrix associative memory where the activation resonates between the two layers of nodes until a stable state is reached.

Binary: having only two possible values. In neural network technology, the value pairs (0 and 1) or (-1 and 1) are common.

Black-Box Model: a numerical model that requires no previous knowledge of the system or process. This type of model utilizes previously recorded input/output processing patterns to predict future responses to a given set of operating conditions.

Boltzman Machine: a distributed parallel processing algorithm developed by Geoffrey Hinton and Terrence Sejnowski based on statistical mechanics that uses simulated annealing to find stable solution states to problems. This system attempted to overcome some of the recall problems of crossbar networks. See also Simulated Annealing.

Boolean Model: a model where variables can take on only one of two values: 0 or 1. Usually, zero represents a logical no, or false. The one represents a logical yes, or true.

Bottleneck Layer: an autoassociative or data-compression network, with a hidden layer having less nodes than the input and output layers. The bottleneck layer is a compact representation of the input pattern.

Bottom-Up: signals proceeding from the input layer toward high levels of a system; used especially in reference to adaptive resonance networks.

Brittleness: the undesirable property of a system, characterized by being catastrophically unable to handle situations outside the narrow scope of the system's expertise.

Causal Knowledge: a type of knowledge used in fault diagnosis and characterized by in-depth, information derived from models. These models are usually qualitative, but may be semi-quantitative and based on first-principles. Causal knowledge uses a deep-knowledge approach. See deep-knowledge, evidential knowledge, and model-based knowledge.

Classification Network: a neural network that uses input values to predict a categorical output.

Classify: assign a new data point to a particular predefined group, upon the recognition of similarity between the new data point and the characteristics of existing members of the group.

Closed-Loop Control: enabling the controller of a process to automatically vary its own settings in response to measured deviations, or error, between the desired and the actual values of a process output parameter.

Cognitive Functions: the human functions of reasoning and inference.

Cognitive Tasks: tasks requiring the use of cognitive functions for performance.

Competitive Filter Associative Memory: a self-organizing system developed extensively by Teuvo Kohonen with the property of modeling the probability distribution function of the input data set.

Competitive Learning: a learning system in that nodes compete with each other for the right to adjust their weight factors.

Compression Ratio: in an autoassociative or data-compression network, the ratio of the number of nodes in the input layer (m) to that in the hidden or bottleneck layer (n). It is a measure of how much noise is being removed from the input signal.

Computer-Aided Design (CAD): computer-based tools that, at a minimum, replace the drafting task of product design. CAD continues to evolve, with increasing analytical and modeling capabilities that help designers calculate the functional requirements of a new product and asses the appearance and performance of a proposed design.

Computer-Aided Manufacturing (CAM): computer-based tools that support manufacturing tasks such as planning, scheduling, and production.

Computer-Integrated Manufacturing (CIM): The use of computers to link activities and sources of data throughout a manufacturing organization. This information management strategy is aimed at better decision-making based on faster access to a wider selection of data.

Concurrent Engineering: a management strategy that seeks to minimize needless manufacturing complexity and start-up problems. New product design is treated as a concurrent , or collaborative, task between the design and manufacturing engineers.

Confidence Interval: a measurement of data quality, based on statistical methods, that addresses the issue of uncertainty. The rang of values within which a given parameter is expected to fall with a certain probability or level of confidence

Conformation: three-dimensional arrangements in space of atoms of a molecule, differing only in rotations about covalent bonds and, possibly, disulphide bonds.

Connection Weight: see weight factor.

Connectionism: a term often applied to the use or study of neural networks.

Connectionist Architecture: a parallel computer architecture that uses many processors, each of which performs a simple computation and has a small amount of memory; each computing node is connected to many other nodes.

Connectionist Model: an artificial intelligence model of intelligent behavior that uses neural networks, i.e., a set of highly interconnected nodes that map input-output responses.

Connectivity: a measure of the relative or absolute number of connections among the nodes of a parallel computer or among the nodes of a neural network.

Constraints: in design applications, limits in the use of materials, design features, product capabilities, or costs; in manufacturing applications, limits in process capabilities, scheduling, availability of raw materials, generation of by-products, or costs.

Content Adressability: the feature of a memory system that enables it to determine the storage location of data based on the contents of the data being stored.

Content-Addressable Memory: one that has the trait of content addressability.

Continuous Process: a production process that is run on an ongoing basis.

Controlled Parameters: the process parameters that are controlled to meet process objectives.

Correlation Coefficient: a statistical measurement of the degree to which a change in one parameter is reflected in a change in another.

Correlation-Coefficient Matrix: a matrix formed by taking the outer product of two variable-column vectors.

Counterpropagation Network: a hierarchical network with the middle layer using Kohonen learning (see Competitive Filter Associative Memory) and the output layer(s) using outstars. The name is derived from the fact that fully implemented network accepts input and generates output from both sides of the hierarchy, allowing the activation to counterflow through the network.

Crisp: a fuzzy-logic term that describes a set whose limits are precisely defined.

Crossbar Network: a neural network with the physical connectivity of a telephone crossbar switch, where each of two layers of nodes is fully interconnected to each other.

CSTR: continuous stirred-tank reactor.

Data Analysis: a data processing tool that identifies duplicate or conflicting data or variables that are unchanging, linearly dependent, or weakly correlated with another variable. All of these problems should be corrected before these data are used to develop a neural-network-based or statistical-based model

Data-Association Network: Similar to a classification network, except it learns association of error-free or ideal data, then classifies or associate data that contains error or noise; e.g., learn five ideal patterns and then recognize noisy input patterns as one of the five patterns.

Data Compression: a process by which data of a certain size (number of points) are reduced to fewer number of points. Ideally, the important information encoded by the original data will still be reconvertable from the compressed version.

Data-Conceptualization Network: network that analyzes data and determines conceptual relationships; e.g., cluster data with many attributes so that grouping relationships can be inferred.

Data Filtering: remove noise from an input signal.

Dead Vectors: In a Kohonen network or counterpropagation network, these are weight vectors that point to a portion of the hyperspace that contains no input pattern vectors. These vectors are constrained by Kohonen learning rules from ever becoming the "winning" node; thus they form no part of the resulting probability distribution model.

Death Phase: in fermentation, the phase where the cells begin to die and the growth rate becomes negative as a result of the lack of nutrients or the presence of poisonous byproducts from the reaction.

Deep Knowledge: knowledge that includes the fundamental physical principles supporting and justifying macroscopic rules and heuristics. Expert systems using deep knowledge usually contain shallow knowledge (e.g., rules and heuristics), and when the problem is outside of the domain of shallow knowledge, the deep knowledge (e.g., models) is then used. Deep knowledge is an attempt to make systems more robust, since catastrophic failure is averted by accessing models found in deep knowledge.

Defuzzification: the output functionality of a fuzzy-logic system. The system accepts imprecise data as input and manipulates those data according to fuzzy rules. The results of this process are translated into a precise, or crisp, output according to one of several defuzzification processes.

Delta Rule: see generalized delta rule.

Decision Boundary: the intersection of two different decision regions.

Decision Region: a specific region within the input space which corresponds to a unique output class. All points within this region contain one and only one output class. Note that the input space can have multiple decision regions corresponding to multiple output classes.

Design of Experiments (DOE): a method to define experimental requirements. DOE will identify the range of experimental conditions required to support the development of a model valid over a given range of values. DOE is particularly useful when designing experiments involving a large number of independent variables.

Detection Error: error that occurs at the border between classes. It results from stochastic overlap of the classes and cannot be wholly avoided.

Deterministic Algorithm: an algorithm whose operation is known in advance. See also algorithm.

Differential Hebbian Learning: a version of Hebbian learning which assumes that learning occurs in the recipient node's synapses only when there is a change in that node's activity at the same time as there is a change in the stimulating node's activity.

Dimensionality-Reduction Network: an autoassociative network that reduces the size of an input vector. The goal is to minimize the number of input variables while retaining all important information from the input variables. Used as the initial section of an input-compression network.

Direct Network Control: training a neural network as the controller and determining the controller output directly.

Discriminant Classifier: identifies decision regions and surfaces from training examples without modeling each class independently as similarity classifiers do.

Distributed Memory: a memory system that stores information throughout the system rather than in a single identifiable address.

Distributed Processing: a term often used to refer to processing with neural networks; may also be used to refer to standard parallel computer systems.

Domain: the problem area of interest, apart from the computational tools use to solve problems in that area.

Domain Expert: an expert in a particular problem-solving domain. This expert would be called upon to provide analysis and problem-solving advice that would be incorporated in a knowledge-based system.

Domain Knowledge: knowledge about a specific problem area, e.g., separation process synthesis.

Drive-Reinforced Theory (DRT): a variation on differential hebbian learning that considers not just the current changes in activity in the stimulating and receiving nodes, but all the cumulative changes in each over some significant time period.

Efficiency: a rating of how quickly a program runs and how long it takes to solve a problem.

Energy Surface: a mathematical property of neural networks that corresponds to the energy wells of physics. The concept is useful in understanding the behavior of many networks

Enzymatic (Alcoholic) Fermentation: a fermentation process where alcohol is the primary product.

Episodal Associative Memory (EAM): a neural-network-based technology that captures design and manufacturing expertise as episodes. This streamlines knowledge acquisition and extends neural network's capabilities into additional application areas.

Epoch: presentation of a set of training patterns to a neural network.

Error-Correction Learning: a form of supervised learning in neural networks where weight factors are adjusted as a direct response to the output error, with the ultimate goal of minimizing or eliminating that output error. In error-correction learning, we are typically concerned with the total output error based on the vector of output error, .

Estimate: a neural-network-based functionality in which the decision-support model developed by the neural network is used to predict, or estimate, the response to a given situation.

Euclidean Distance: geometric distance between two points, given by the square root of the sum of the squares of the differences between vector components.

Euclidean Normalization: normalization of vectors to unit length (the length of a vector is the square root of the sum of the squares of the elements).

Evidential Knowledge: a type of knowledge used in fault diagnosis and characterized by evident, experiential information derived from observables. Evidential knowledge typically uses a shallow-knowledge approach. See shallow knowledge; causal knowledge.

Excitation: leading to an increased activation within a node.

Excitatory synapse: a synapse that, when stimulated by an input signal, causes an increased activation in its attached node.

Expert System: also called a knowledge-based system, an expert system is a computer program that uses high-quality, in-depth, knowledge to solve complex and advanced problems typically requiring experts.

Expert-System Shell: a software development tool that supports the development of an expert system. The shell contains a user interface and an inference engine, the logic-processing mechanism that manipulates rules within the system. The use of a shell enables the system developer to focus on acquiring knowledge and composing rules specific to the problem of interest, with minimal development required to provide generic expert-system capabilities.

Extrapolation Error: error that occurs when a new case falls beyond the range of the original training data and is misclassified.

Fail-Soft: the smoothed degradation of performance of a system as individual components of the system fail; synonymous with "graceful degradation."

Fan-Out Layer: a layer that receives an input pattern and then, by virtue of its interconnections, presents the entire input pattern to each node in the subsequent layer.

Fault Diagnosis: an application of artificial intelligence concerned with troubleshooting, i.e., determining the origins of problems based on observable information, and then recommending solutions or corrections.

Fault-Tolerant: has the property of being fail soft.

Feedback: the process whereby part of the output signal of a circuit echoes back to the input of the circuit, usually after some filtering operation.

Feedback Competition: a means of implementing competition among the nodes of a network. The output pattern is echoed back to the input layer after thresholding or some other filtering process.

Feedback Connections: interlayer connections in an neural network where signals are sent in the backward direction, i.e., from a node deeper into the network (closer to the output layer) backward to nodes more shallow in the network (closer to the input layer).

Feedforward Connections: interlayer connections in a neural network where signals are sent in the forward direction, i.e., from nodes more shallow in the network

(closer to the input layer) to nodes deeper into the network (closer to the output layer).

Feedforward Network: a network in that the signal pathway goes only one way - from the input layer to the output layer.

First Principle: a basic law of chemistry, physics, or other natural science. First principles express fundamental theory and are used as the basis for decision-support models.

Flory-Huggins-Based Models: a statistical thermodynamic model that applies the Flory-Huggins equation of state for polymer solutions.

Forced Learning: refers to the intermediate continuum of rules based on manipulation of the content of the input stimulus bring about learning.

Formulation: The recipe, or proportions of ingredients, for many products made by batch or continuous processes.

F-Ratio: a statistical term that is defined as the ratio of the mean-squares regression to the mean-squares residual. The higher the F-ratio, the more important the specific term is in the overall regression model.

Frequency Domain: the signal domain in which a signal is presented as a function of frequency.

Fully Interconnected: a neural network or layer that has the output of every node in one layer serving as an input signal to every node in the other layer, and vice versa. Sometimes called " fully connected".

Functional-Link Networks: a network that simply adds a functional-link layer to a backpropagation network. This additional layer improves the network's capability to represent complex nonlinear relations.

Fuzzifier: it converts a numeric variable to a fuzzy-logic variable.

Fuzzy Logic: an AI-based technology focused on control applications. Control objectives are stated in fuzzy, or qualitative, rules. This avoids the need for precise, or crisp, rules as required by expert systems.

Fuzzy Set: classes of objects in which the transition from membership to nonmembership is gradual rather than abrupt.

Fuzzy Rule: the type of rule characteristic of fuzzy logic and used to state control objectives. A fuzzy rule defines a qualitative relationship that may be expressed as a smoothly varying fuzzy set membership function.

Generalization Phase: the final phase of neural network development where we feed novel input data to the network. It is hoped that, with the training the network has undergone, the output response will be proper.

Generalized Delta-Rule (Delta-Rule): also called GDR, an iterative gradient-descent backpropagation training algorithm for neural networks. The delta rule possesses momentum and a bias function.

Gestalt: a pattern so unified that its properties as a whole pattern cannot be derived from the properties of its parts.

Graded Learning: learning systems that provide performance "grades" as feedback rather than specific error values; they do not require knowledge of the specific answer for a given input pattern, only the ability to rank the network's performance.

Gradient-Descent: the technique used to approach a minimal solution in a mathematically defined space by progressively moving in the direction of steepest descent (at each iteration of a multiple-step process, the move is in the direction of steepest descent at the current position in space).

Grandmother Node: an output node that acts as the sole recognizor of a particular input pattern.

Gray Scale: a range of possible discrete (noncontinuous) values for each element of the input vector to a neural network.

Growth Phase: in fermentation, the phase where most of the cell growth occurs, cells dividing at their maximum rate. The cell growth rate is proportional to the cell concentration in this phase.

Hardwired Neural Networks: neural networks with all connection weights predetermined and fixed (i.e., hardwired).

Hebbian Learning: (named after Donald Hebb), a type of learning in neural networks that adjusts the weight factors between two nodes based on a correlation between the output values of those two nodes.

Heteroassociative Memory: an associative memory in that stored data items are associated or correlated with other, different data items.

Heuristic: a rule of thumb used to guide the problem-solving pathway in a knowledge-based system. Heuristics typically limit the search space in problem domains that are not very well understood. By limiting the search space, heuristics improve expert-system efficiency and make the search more controllable.

Hidden Layer: a layer of nodes in a neural network that receives input from the input layer, performs calculations and signal processing, and passes the information to other nodes within the network.

Hierarchical Network: a network consisting of several layers with connections between layers chosen to achieve some kind of processing in stages. Such networks can have feedforward and/or feedback connections and can combine different learning paradigms at each stage.

Hinton Diagram: a compact graphic display of activations or weight factors for a layer of a network.

Hopfield Networks: named after John Hopfield of Caltech, this is another name for any one of several designs for crossbar networks.

Hydrophilic: having affinity for water.

Hydrophobic: preferring non-polar environments over an aqueous environment.

Hyper-: a prefix to indicate that the geometric form following may be of any dimensionality, including two dimensions, three dimensions, or any higher dimensions.

Hyperplane: a generalized n-dimensional construct equivalent to a plane in 3-dimensional space.

Hypersphere: a generalized n-dimensional construct equivalent to a sphere in 3-dimensional space.

IF-THEN: a common type of expert system rule describing if a situation exists, then the following question should be asked, analysis performed, or action taken.

Incremental Correlation Coefficient (R^2): reflects the fit between each variable and the overall regression model, and the contributions of different terms (i.e., linear, cross-product, and quadratic terms). A perfect fit has an incremental R^2 of 1.0, and no correlation results in an incremental R^2 close to 0.0.

Indirect Network Control: training a neural network to serve as a model of a process (as in the case of model-based control), or to determine the setpoints of a local controller (as in the case of supervisory control), thus indirectly affecting the process being controlled.

Induction Phase: in fermentation, the phase where the cells begin to adapt to their new environment and minimal reproduction occurs.

Inference Engine: The portion of the knowledge-based system that contains problem-solving methodologies and/or general problem-solving knowledge.

The inference engine is usually separate from, and acts upon the knowledge-base.

Inhibition: leading to decreased activation in the nodes.

Inhibitory (synapse): a synapse that, when stimulated by an input signal, causes a decreased activation in its attached node.

Input-Compression Network: a hierarchical network that reduces the dimensionality of an input vector, compresses the number of input variables entering a prediction network while retaining important information, prior to entering it into a prediction network. This is a method to handle systems that have a large number of input variables (e.g., > 50) that can cause training difficulties.

Input Layer: the first layer of a neural network; this layer receives information from an external source, and passes the information into the network for processing.

Input Transfer Function: see activation function.

Input Vector: the pattern presented to the input nodes of a network.

Integrated Product/Process Design (IPPD): an engineering management approach in which the design of a product and the associated process are undertaken concurrently. Product and process designs are most effectively integrated by knowledge-based and other computer-based tools that help designers identify interactions between product and process designs, and resolve these interactions in ways that best meet overall objectives. The IPPD process

identifies production problems earlier in design cycle and helps provide lower cost, more effective solutions to these problems.

Interconnect: one of the pathways over which nodes communicate in a neural network

Interlayer Connections: connections between nodes in a neural network where outputs from nodes in one layer feed into nodes in a completely different layer.

Internal Model Control: an advanced control strategy that uses a process model on-line and built right into the control loop.

Internal Threshold: a numerical value that controls the activation of the neural network. In most neural network algorithms, we subtract the internal threshold from the total nodal input. If this difference is below a certain level, the node is deactivated and has zero output.

Interpolative Associative Memory: an associative memory that responds to unfamiliar data inputs by producing a blended or interpolated response of the nearest stored data patterns.

Intralayer Connections: connections between nodes in a neural network where outputs from nodes in one layer feed into nodes in that same layer.

Inverse Network Control: training a neural network as an "inverse" model of a process, i.e., predicting the process inputs necessary to produce the desired process outputs (setpoints).

I/O: input/output, i.e., communication between the user and the computer program.

IPS: acronym for interconnect updated per second. This is becoming a standard unit of measure for the operational speed of a neural network implementation and refers to the speed at that weight factors can be updated. Sometimes referred to as COPS ("connection operations per second").

Isoelectric Point: for a protein, it is the pH at which its net charge is zero.

Iterative: a synonym for "repeated". In the context of calculation methods, it refers to a process of changing a parameter slightly, rerunning a calculation, changing that parameter again (and typically incrementally further in the same direction), rerunning the calculation again, and so on until a solution is found or an objective reached.

Kidera Parameter: nine characteristic protein properties and ten characteristic protein factors that describe protein properties solely on the basis of amino acid compositions.

K-Means Clustering Algorithm: finds a set of cluster centers and a partition of the training data into subsets. The data are partitioned such that the training points are assigned to the cluster with the nearest center.

Knowledge: the accumulation of facts, rules, and heuristics programmed into the computer in a knowledge-based system.

Knowledge Acquisition: the process of taking expert knowledge, possibly from multiple sources, and translating it into a viable knowledge representation and computer code for a knowledge-based system.

Knowledge-Based System: a computer program that uses high-quality, in-depth, knowledge to solve complex and advanced problems typically requiring experts. Knowledge-based systems typically require some form of symbolic computing, since a large portion of knowledge is inherently qualitative. Synonymous with expert system.

Knowledge Engineer: the person who designs, builds, and debugs a knowledge-based system. Usually, the knowledge engineer is someone who is very familiar with AI techniques.

Knowledge Engineering: the process of building a knowledge-based system.

Knowledge Indexing: the ability to store large amounts of information (knowledge) and access it quickly and efficiently.

Knowledge Representation: the way of characterizing and organizing the knowledge required by an expert system to solve a complex problem. For example, the knowledge can be rule-based.

Lateral Inhibition: a way of introducing competition among the nodes of a network layer, characterized by a specific pattern of lateral, or intralayer, interconnections.

Layer: an element of a neural network architecture referring to the arrangement of nodes in the network. Layers may be characterized as input, hidden and output layers, each with a specific role. The arrangement and number of layers vary among different neural network architectures. The choice of architecture and number of layers and nodes within each layer determines the capability of the network.

Learning Curve: a method to visualize how well a network performs recall and generalization. A plot of the average prediction errors for both recall of training data and generalization of test data as a function of the number of examples in the training data set. For a well-trained network, as it receives more information, the recall and generalization curves approach each other.

Learning Law: a rule for updating the weight factors of a neural network during training.

Learning Rate: a parameter that regulates the relative magnitude of weight-factor changes during learning.

Learning Schedule: a schedule which specifies how parameters associated with learning change over the course of training a network.

Learning-Vector-Quantization (LVQ) Network: an autoassociative-classification network that learns through supervised training developed by Teuvo Kohonen. This type of network is advantageous when classifying systematic patterns.

Ligand: any molecule that interacts with one or a few sites on a protein

LMS rule: the least-mean-squares training rule.

Local Receptive Field Network: a neural network with unique architecture targeted to improve performance and speed-up the training session; the local receptive field network has one layer and a radially symmetric function (such as a Gaussian density function). Nodes are pruned as a means of speeding up the training session.

Local Storage: storage that is accessible only to a given node and not to other units.

Logical Value: a value, either 0 (i.e., no, or false) or 1 (i.e., yes, or true), given to a variable.

Long-Term Memory: memory that persists over extended periods of time; memory associated with learning.

Madaline: acronym for multilayered adalines; an array of adaline adaptive filter elements.

Mapping: an association of one collection of patterns or numbers with another collection of patterns or numbers. A mathematical function $y = f(x)$ is a mapping from the legal x values to their corresponding y functional values.

Membership Function: a method used in fuzzy logic to define control objectives. Rather than requiring that a parameter either is or isn't within a precise, or crisp, set, fuzzy logic enables a user to provide a degree of membership in a function. The degree of membership then determines the degree to which a

control action is taken. The point of this approach is to accommodate errors in data and to enable easier statement of control objectives.

Memory System: a system that can store and recall information on demand.

Microbial (Cell-Growth) Fermentation: a fermentation process where cells are the primary product.

Microfeature Concept: the unique property of neural networks, where each node affects the overall input-output pattern from the network only slightly. Each node, therefore, operates independently of other nodes, and incorporates a microfeature of the total input-output response. As a result, the network does not depend heavily on the performance of a single node, and thus, can act as a filter or process noisy data.

Middle Units: nodes with no direct connection to either the input or output of the network; also called hidden units.

Model: a mathematical representation of the behavior of a system. A model may be an intuitive understanding of a cause-and-effect relationship. But more commonly, it is expressed mathematically and represents the result of analysis to determine how a system will respond to a range of inputs.

Model-Based Control: the use of a process model to aid controller design or operation. A process model can be used to design a traditional control system,

investigate control strategies off-line, or be placed in directly in a control loop and used on-line for advanced control strategies.

Model-Based Knowledge: knowledge that incorporates in-depth, quantitative information based on first-principles. Model-based knowledge is a deep-knowledge approach that is typically more quantitative than causal knowledge.

Model-Predictive Control: an advanced control strategy that uses a process model to predict response over a "long" period of time, i.e., at least as long as the open-loop response of the system.

Modular: a design concept applicable to software and hardware. A modular design is one in which the complete system is composed of a number of modules. Each module has a specific function independent of other modules. Modular design provides system-maintenance benefits because a single module can be revised or replaced without risk of unintended consequences in other modules or functional areas.

Momentum: extra weight added to the change in weight factor to speed up training in a neural network.

Monotonically Increasing Function: a function with the property that its value at large values of its variable is always larger than the function's value at smaller variable values; such a function's graph always appears to slope upward to the right.

Moving-Window Network: The moving-window network is a special hierarchical network used to model dynamic systems and unsteady-state processes. A moving window is a way to isolate subsets of a long string of time-dependent measurements, simply taking the last n time segments and using each segment as an input to a network. For training, a moving window provides a means for creating multiple training examples (patterns) from continuous raw data.

MRII: acronym for madaline rule II which is an adjustment principle for the madaline. Under this system, the change made on the madaline elements is always that which minimally disturbs the system.

Multipass Network: a hierarchical network where the network activity flows from the input layer to the output layer, and then back through the network at least once before a stable state is reached. An example is a backpropagation network during training.

Multivariate Regression: regression analysis to explore the interaction between multiple variables. Regression analysis is a statistical method that originated with studies of regressive (and dominant) traits in plants. Traditional regression analysis explores the impact on one variable of changes in the value of another. Multivariate regression analysis explores the impact on one variable of changes in the values of more than one variable.

Neocognitron: a multistage pattern recognizer/feature extractor that simulates the way visual information feeds forward in the cortex of the human brain.

Neohebbian Learning: a mathematical form of hebbian learning developed by Stephen Grossberg and Michael Cohen of Boston University.

Net: neural network

Neural-Fuzzy Networks: implements fuzzy-logic inferencing through neural networks

Neural Network: an AI-based technology that is a pattern-recognition technique. It is a quantitative empirical modeling tool characterized by a network of highly interconnected nodes that pass numerical values to each other and calculate an output based on the sum of inputs from other nodes. Neural networks are particularly useful for pattern-matching and filtering noisy or incomplete information.

Neurocomputer: a dedicated, special-purpose digital computer designed to efficiently perform the mathematical operations common in neural network simulations.

Neurode: see node.

Neuron: a biological neural cell.

Node: in neural networks, as in mammalian brains, an element that functions as a summing junction and transmitter. A neural network might consist of , say, 5 to 50 nodes, each interconnected to several others according to the particular neural network architecture employed. Each node adds together the signals

presented as input and transmits an output signal on the basis of an internal transfer function, or a relationship between input and output.

Noise: in general, that portion of a signal that is not desired because it is extraneous and interferes with the desired signal. In Taguchi method for experimental design, noise is the effect of uncontrollable process deviations.

Nondeterministic: a relation is nondeterministic if it can generate multiple solutions to the same question, and therefore, it is unclear at the outset of the goal which solution will ultimately succeed.

Nonparametric Classifier: does not have a probability distribution for the output responses, but instead predicts the output class by comparing the output class to its nearest training points.

Non-polar: having no net charge and no asymmetric charge distribution and consequently being chemically unreactive

Normalization of Vectors: adjustment of vectors so that the values of their components lie within a stipulated range, typically 0 to 1, or -1 to 1.

Numerical: information expressed as a number. The alternative is symbolic information, which is information expressed as symbols.

Numerical Modeling: problem-solving based on a strictly quantitative, algorithmic approach. Accurate values of quantitative parameters and variables is absolutely essential to the success of quantitative models.

Ogston-Based Models: a statistical thermodynamic model utilizing the osmotic virial equation for dilute aqueous mixtures.

On Center - Off Surround: another name for lateral inhibition.

Optimize: the act of finding the best possible solution, not merely an acceptable solution, to a problem. A number of optimization techniques exist, but their common goal is to enable a designer, for example, to find the one best solution and to know that the found solution is best.

Orthogonal: a characteristic of vectors in which they have no geometric projection upon each other. In normal Euclidean space, this is equivalent to being perpendicular to each other.

Orthogonal Learning: learning in which nonorthogonal patterns are orthogonalized by a network.

Output: information sent from the program to the user, usually via the screen or a disk file.

Output Layer: the layer of nodes in a neural network that receives input from other nodes in the network, calculates and output, and sends this output to an external receptor.

Output Function: the function describing the relationship between the output of a node and its internal activation level.

Outstar Avalanche: a type of neural network that is capable of performing a temporal series of recall operations. See avalanche network.

Padaline: contraction of "polynomial adaline," Don Specht's variation on the adaline that uses a polynomial surface to categorize input patterns. Also called "polynomial discriminant method" or "PDM."

Parallel Distributed Processing: another term for computing with neural networks.

Parametric Classifier: has a probability distribution associated with the output responses being classified, and is trained by predicting the best parameters for the statistical distribution.

Partition Coefficient: in an aqueous two-phase system, it is the ratio of a component's concentration in the top phase to that in the bottom phase.

Pattern Recognition: A problem-solving approach in which the situation is viewed as a collection of many pieces of information arranged in a pattern rather than as a collection of many unrelated pieces of information. Visual images represent a common pattern. The pattern recognition approach, when feasible, tends to be faster and more likely to recognize significant similarities while overlooking insignificant differences.

PE: acronym for processing element (node).

Perceptron: a neural network with feedforward interlayer connections only, and no intralayer or recurrent connections.

Piecewise Linearization: the representation of a nonlinear relationship by a series of linear relationships. When plotted, the piecewise linearization scheme

approximates a curved nonlinear relationship with a series of end-to-end linear segments.

Plastic: adaptive, capable of modification.

Polar: having a non-uniform charge distribution and being chemically reactive.

Polydispersity Index: the ratio of the weight-average molecular weight M_w to the number average molecular weight M_n.

Polypeptide Chain: a string of amino acids linked sequentially via peptide bonds

Postprocessing: a term that refers to any processing of knowledge or data after some other operations, such as the generation of those data by a particular software package. Postprocessing can accomplish analysis, transformation, or presentation of data or knowledge.

Postsynaptic: pertaining to the input signal arriving at the node after it has been modified or weighted by the synapse.

Prepreg: a composite material, a fiber mat impregnated with unreacted thermosetting resins.

Preprocessing: a term that refers to any processing of data or knowledge before it is entered into a particular software package. Preprocessing can accomplish validation, analysis, or transformation of input data or knowledge.

Presynaptic: pertaining to a signal after it leaves its source, but before it is modified or weighted by a synapse.

Primary Structure: the linear sequence of amino acid residues in a polypeptide chain

Principal Component Analysis: a method for identifying interesting but unanticipated structure (e.g., clustering) in high-dimensional data sets.

Probability Distribution Function: a functional relationship that correlates a particular input pattern with its likelihood of appearing in the input data set.

Process Forecasting: the attempt to predict the value of measured process variables in the future based on a history of noisy or seemingly chaotic data.

Process Synthesis: chemical process flowsheet development, i.e., determining appropriate unit operations and configuring these operations into a coordinated process that achieves the desired objectives.

Processing Element: a node in a neural network.

Qualitative Modeling: a form of simulation based on symbolic (i.e., qualitative) relationships between concepts. Qualitative models are used to express directionality, e.g., as the speed of the pump motor goes up, the volumetric flow rate goes up. However, qualitative models do not represent the knowledge numerically.

Quantitative Modeling: a modeling technique that represents knowledge numerically, relating information in terms of quantity.

Quaternary Structure: the 3-dimensional arrangement of multiple polypeptide chains in a folded protein; it may also be used to refer to the arrangement of individual folded domains.

Radial-Basis-Function Network: a network that has radially symmetric transfer functions in the nodes of the hidden layer. For a node to be radially symmetric, it must have a center vector and a distance measure. Used primarily for classification problems.

Randomization: the process by which a collection of vectors is randomly oriented about the unit hypersphere or about some portion of the unit hypersphere.

Real-Time: refers to events taking place concurrently with the operation of a system. A real-time system is one that can operate (accept and process input data) concurrently with the actual events and thus keep up with events as they happen.

Reasoning: the use of logic or reason to draw conclusions. Expert systems incorporate rules to enable reasoning about situations so that problems can be identified and solved.

Recall Phase: the phase of neural network development where we assess the network's performance entering a specific input into the network, calculating the output, and assessing the error from the desired input-output response.

Recall Schedule: a schedule which specifies how parameters associated with the response of a network changes over the course of recall.

Recurrent Connection: a connection in a neural network where the output from a node feeds into itself as an input.

Recurrent Network: A recurrent network combines the feedback and feedforward connections of neural networks. In other words, it is simply a neural network with loops connecting the output responses to the input layer. Also referred to as a time-lag recurrent network or an externally recurrent network.

Regression: a statistical method to explore the impact on one variable of changes in the value of another.

Reinforcement Learning: a type of supervised learning in neural networks characterized by adjustment of weight factors based on a single scaler error value. This learning is contrasted with error-correction learning, where an output error vector, is used to adjust the weight factors. Thus, reinforcement learning is "selectively supervised" and is faster and easier to use than error-correction learning. On the downside, reinforcement learning uses less precise information to adjust the weights.

Reset: in a neural network, the process of shutting off and keeping off all currently active nodes in a system without interfering with the ability of inactive nodes to become active.

Resonance: when used in reference to neural network's activity, it refers to a state of the network such that activation flowing between layers of the network is self-sustaining and generates no changes to the activity patterns of any of the layers involved. The network has thus reached a stable state, where no activity changes occur.

Robustness: the property of a system to be able to (non-catastrophically) handle situations outside the narrow scope of the system's expertise. A robust system gradually degrades in performance when it is given inconsistent, erroneous, or incomplete data.

Rule: an expert system clause that is conditionally true, i.e., the head of the clause is true if the body (condition section) can be proven true. A rule must have both head and a body. If the clause has no body, it is a fact; if it has no head, it is a question.

Rule-Based System: a knowledge representation that uses IF-THEN conditional statements (rules) to build an inference chain to ultimately solve problems.

Search: in a knowledge-based system, search is the systematic procedure used to analyze and reason through the knowledge-base and solve the problem. Typically, the pathway between the starting node and the goal node is unknown ahead of time, and the search determines the pathway.

Search Space: the state space open to the system to analyze in an attempt to find a solution to the question or problem.

Secondary Structure: local regular conformations of polypeptide chain, especially α-helices, β-strands, and reverse turns.

Selectivity: the ratio of partition coefficients and describes the extent to which two proteins will partition to opposite phases in an aqueous two-phase system.

Self-Discovery: the ability of a neural network to automatically discover relationships in data. This self-discovery capability can be used as a knowledge acquisition technique and as a way to enable a system to self-improve and adapt to changing circumstances.

Self-Organization: the process by which a neural network learns its input data without the aid of an external tutor. Self-organizing systems do not need to be told the correct response to a particular input pattern; rather, they determine their own responses to the input.

Self-Organizing system: a neural network that trains without being provided with the correct answer and that physically orders its connections during training in such a way that the resulting physical structure of the network models some aspect of the training data's organization.

Self-Tuning Controller: a controller that can change its tuning parameters on-line based on changes in process conditions or dynamics.

Sensor: a device that provides, typically, an electrical output that varies in response to a particular process parameter (e.g., temperature, pressure, etc.).

Setpoint: the target value of a process output variable. The goal of a process controller is to adjust the controlled variables of a process to compensate for changing uncontrolled variables so that the difference or error between the setpoint and process output variable is as close as possible to zero.

Shallow Knowledge: knowledge based on macroscopic rules and heuristics, with no understanding of the fundamental physical principles supporting and justifying this knowledge. Shallow-knowledge systems are brittle, i.e., their performance falls off very rapidly (sometimes catastrophically) outside of their domain of expertise. See also deep knowledge.

Shell: see expert system shell

Short-Term Memory: the short time interval during which a signal exists in a biological or artificial neural system before it decays or causes learning to occur.

Sigmoid Function: a continuous, monotonically increasing, S-shaped numerical function. Sigmoid functions typically have limiting values of [0,+1] or [-1,+1].

Signal Analysis: the analysis of a signal to extract information. Also called signal processing, this task may involve determination of the source or the propagation of a signal.

Similarity Classifier: using the training examples to develop a model for each class, and comparing future test cases to this model in order to assign a score to define similarity. Decisions are then made based on those scores. This techniques is also called a minimum-distance classifier.

Simulated Annealing: the process of slowly lowering the amount of random noise introduced into the weight factors and/or input signals of a network as training progresses.

Single-Pass Network: a hierarchical network where the activation flows from the input layer to the output layer only.

Soft Sensor: software-based sensor. An integration of process sensors with intelligent software and microprocessors to monitor a wide range of process variables.

Solubility Parameter: a measure of the polymer-solution interactions used in statistical thermodynamic models. Similar chemicals have similar solubility parameters and are more likely to go into solution together.

Spatiotemporal Pattern: a pattern characterized by changes in both space and time; a time-dependent spatial pattern.

Speech Recognition: the branch of AI that attempts to process, understand, and interpret audio communication and human speech.

Spurious Minima: extra, unwanted depression in the energy surface of a network that causes incorrect responses to input stimuli.

Spectroscopic sensors: a light-induced fluorescence technique used to estimate the compositions of biochemical species.

Squashing Function: Function whose value is always between finite limits, even when the input is unbounded.

State: a particular collection of weight factors and activations of a single node or within a neural network.

State Space: the collection of all possible situations or configurations in a problem. For example, in flowsheet development, there may be 10^{20} different flowsheet configurations that could successfully solve the problem.

Stationary Phase: in fermentation, the phase where the cell growth rate is virtually zero due to depletion of nutrients (organic feed) or crowding of cells.

Statistical Process Control (SPC) or Statistical Quality Control (SQC): The use of statistical methods to monitor and improve product quality. Measurements of product attributes such as dimensions or properties are taken on a regular basis. From these measurements, simple statistical measures such as the average value and range of these quality parameters are calculated. Operators observe trends in these calculated values and adjust process setpoints, repair equipment, or identify changes in material characteristics as required to return the measured parameter values to the desired averages and to minimize range (or variation of the values).

Statistics: the analysis of data to draw conclusions about the behavior of a system, using specialized methods such as statistical process control or statistical quality control. Statistics-based models are also used for prediction of, for example, product properties or changes in process parameters.

Stochastic Learning: a type of learning in neural networks that uses statistics, probability, and/or random processes to adjust the weight factors. Error-correction learning has a problem of getting trapped at a local minimum

of error; some stochastic methods have the ability to avoid local minima and move to the global minimum in error.

Subsymbolic Processing: the processing used by neural networks, characterized by microscopic interactions among nodes that eventually manifest themselves as macroscopic, symbolic, intelligent behavior.

Summed Input: the weighted, algebraic sum of the individual postsynaptic input signals.

Supervised Learning: a type of learning in neural networks where an external teacher controls the learning and incorporates global information and training data.

Supervisory Control: control involving resetting the setpoint of a local controller by a computer algorithm.

Symbol: a string of letters and names that represents a single concept, object, or item.

Symbolic Processing: a branch of computer science that deals with the processing of non-numerical symbols and names. Symbolic processing is contrasted with the more classical numerical computing, which deals with the processing and calculation using numbers. Also referred to as symbolic computing.

Synapse: in neural networks, the junction at the end of an interconnection that joins the output of one node (or a signal from the environment) to the input of another node. In biology, the junction between an axon collateral and a dendrite.

Taguchi Method: a statistics-based method primarily used to identify and rank factors affecting product quality. The objectives of this method are to minimize the number of experiments required to make this determination and ultimately to identify the least expensive way to achieve required quality levels.

Target: desired output of a network corresponding to a given input pattern.

Task-Oriented Approach: an approach to problem-solving in expert systems by dividing the problem up into sub-problems such that a single task, when executed, satisfies the sub-problem. Usually, the complete problem-solution is the combination of all the tasks used to solve every sub-problem.

Tertiary Structure: stable overall 3-dimensional structure of a folded polypeptide chain.

Test Set: a collection of data patterns different from the training set, but similar to it used to judge the effectiveness of a network's training.

Threshold: the activation level that the node must attain before its output rises above zero.

Threshold Function: the functional form used in neural networks to calculate nodal output. At very low input values, the threshold-function output is zero. At very high input values, the threshold-function output is one. Since threshold functions are typically sigmoid functions, they gradually deactivate the node as the input magnitude decreases.

Time Domain: the signal domain in which the signal is presented as a function of time.

Tonotopic Map: a topology-preserving map that corresponds to an ascending or descending ordered set of frequency inputs. See also topology-preserving map.

Top-Down: Signals proceeding from higher (later) levels in the network to lower (earlier) levels in the network.

Topology: the mathematical study of geometrical shapes and their properties.

Topology-Preserving Map: a network organized in such a fashion that its physical structure models some aspect of the topology of the input data patterns. The topology might be a sequence of frequencies in an auditory system, or the physical organization of a particular area, or some other feature of the data pattern.

Training Phase: the initial phase of neural network development, characterized by repeatedly presenting sets of input-output data to the network and adjusting the weight factors to minimize the output error.

Training Set: a collection of data typical of that which a neural network will see in operation, and that is used to train the network.

Transfer Function: a mathematical function that relates the input or some state of a system to the output or other state of the system. In a node, the transfer

function includes the combined effects of the input function, the activation function, and the output function.

Transition Region: the buffer between two different decision regions where we can make only fuzzy inferences about the classification.

Uncontrolled Parameters: process parameters that are not controllable, even though they may vary and may have a significant impact. Examples of uncontrolled parameters include ambient temperature and humidity. One process optimization issue is to achieve process objectives, such as, quality and cost in the presence of variations in uncontrollable process parameters.

Unidirectional Counterpropagation Network: a variation of the counterpropagation network that accepts input data only from one side of the network and responses from the other side of the network.

Unsupervised Learning: a type of learning used in neural networks where no external teacher is used and the neural network relies upon internal control and local information. Frequently, the neural network develops its own models automatically without additional information input.

Validate: in the context of developing a model, the task of determining whether the model accurately represents the relationship being modeled.

Vector Quantization: a process that replaces a probability distribution model of an input data set with a smaller, proportional representation of vectors that represent input data features.

Vigilance: the characteristic of autonomous systems by which they learn more or less detail in a particular associative category.

Wavelet Network: it resembles the radial-basis-function network, except that they use a more elaborate transfer function.

Weight Factor: an adjustable parameter in neural networks. The weight factor is frequently denoted as w_{ij}, and is the numerical value that we multiply the output from node i by to give the input to node j.

Weight Vector: the vector that has components corresponding to the weight factors on the interconnections of a node.

Weighted Sum: computed by multiplying the value of each input signal by its corresponding weight factor and then algebraically adding the resulting terms together. Mathematically, this is equivalent to computing the dot product of the weight vector and the input vector.

Zero-Mean Normalization Method: a technique that normalizes a data set between limits of -1 and +1, having the average value set at 0.

Vita

D. Richard Baughman is a Ph.D. student in the Department of Chemical Engineering, Virginia Polytechnic Institute and State University. Mr. Baughman received his B.S. degree from Virginia Tech in 1986. He worked for Haven Corporation, Baltimore, Maryland, a manufacturer of specialty coatings for the printed circuit board industry, as material and process control engineer from 1986 to 1991. He returned to Virginia Tech in Fall 1991 for graduate studies. For his outstanding academic progress at Virginia Tech, he has received the Tennessee Eastman Graduate Fellowship.

D. R. Baughman

D. Richard Baughman