Appl. Math. and Comp. Sci., 1998, Vol.8, No.4, 887–898

SYSTEM MODELLING USING NEURAL NETWORK PARAMETER FUNCTIONS

ARUN THOLUDUR[*], W. FRED RAMIREZ[*]

The development of mathematical models that accurately describe the dynamics of a complex system is a very difficult task. The use of neural networks in conjunction with prior process knowledge improves modelling performance. This can be achieved by using the method of neural network parameter function modelling. This paper presents the application of this technique to the modelling of a complex batch biotechnology system. The models developed are optimized using two different methods and the results compared.

1. Introduction

The advent of recombinant DNA technology has provided yet another method of obtaining proteins that have a variety of practical applications. This technology is based on the fact that a host organism's biosynthetic machinery can be modified by inserting a gene that codes for the protein of interest. This modified organism is then grown in huge numbers to obtain the desired amount of the protein. Like all biotechnology systems, the growth of cells and the production of proteins is a highly non-linear process. Modelling such systems poses a unique challenge in that an accurate model will have to involve many interacting microscopic phenomena. A generalized mathematical model for the production of β-galactosidase using a strain of *E. coli* has been presented by Lee and Ramirez (1992). In this paper, we summarize an alternative method of modelling this reactor system using neural networks. Indeed, the method presented is not limited to the particular system considered. This modelling method can be generalized to include many dynamic processes.

Trying to obtain an accurate model of a process is often just one step in a much bigger scheme of things. In addition to understanding the dynamics of a process, a model is used as a basis for the optimization of the process. Returning to the biotechnology system under consideration, once a method of producing proteins has been obtained, the next step would be to maximize the amount of protein that can be produced. This calls for an application of optimization techniques. In this paper, two different optimization schemes are examined and the optimization results are presented.

[*] Department of Chemical Engineering, University of Colorado, Boulder CO 80308–0424, USA, e-mail: tholudur@optimal.colorado.edu; fred.ramirez@colorado.edu.

2. Neural Networks—An Overview

A neural network is basically a computational scheme that utilizes a massive interconnection of simple processors called neurons. Many different types of neural network architectures are possible. An example of a typical neural network used in this work is illustrated in Fig. 1. There are three layers of neurons in this architecture with a set of connection weights between these layers. Each of the neurons performs a transformation of its input to an output. This transformation is also called the activation function and provides the non-linear nature of this type of neural network. The approximating capabilities of the neural network arise from the fact that the connection strengths or synaptic weights can be modified in order to shape the input-output transformation. This systematic modification of the synaptic weights is carried out using a learning algorithm.

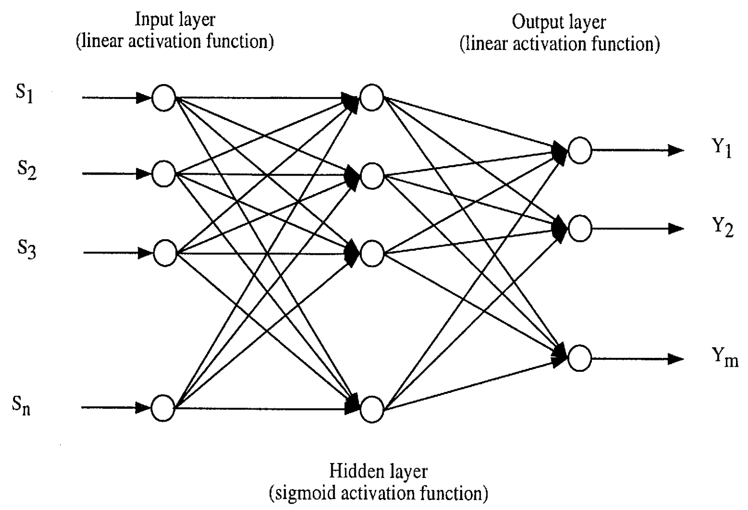
Input layer
(linear activation function)
Output layer
(linear activation function)
S1
S2
S3
Sn
Y1
Y2
Ym
Hidden layer
(sigmoid activation function)

Fig. 1. A typical feedforward neural network with one hidden layer.

3. Neural Network Training and Generalization

The adjustment of the synaptic weights of a neural network in a systematic manner is done by a learning algorithm. One of the most widely used algorithms for training feedforward neural networks is the backpropagation algorithm (McClelland and Rumelhart, 1988). This algorithm suffers from many drawbacks, including slow speed of convergence and a tendency to get stuck in local minima. Many variations of the original backpropagation algorithm are available. In this work, we have used the TRAINLM function in the MATLAB Neural Network Toolbox (1994) which is essentially an adaptation of a Levenberg-Marquardt scheme of optimization (Masters, 1995). This algorithm is very efficient in terms of speed of convergence at the expense of utilizing more memory and is often used when the number of weights to be trained is not too large.

One of the key issues in neural network training is the concept of generalization. The adaptation of synaptic weights during learning in order to mimic the input-output relationship often presents a dilemma to the user. A neural network can do a very good job of learning such a mapping, but one is often looking for a network that is able to generalize well. By this, we mean that the network should not merely "memorize" the patterns it is presented with. Rather, it should be able to decipher underlying relationships between the inputs and the outputs and should be able to provide a reasonable output when it is presented with inputs that it has not seen before. Smoothness of the function that is being approximated is usually a prerequisite for good generalization. Another factor that tends to improve the generalization capabilities of a neural network is that the input-output data that is presented for training be a representative sample of the range over which generalization is desired. In other words, a neural network which has been trained in a particular local subspace would find it very difficult to generalize in an entirely different subspace. Related issues of underfitting, overfitting and generalization in neural networks are discussed in detail in the literature (Geman *et al.*, 1992; Moody, 1992).

Some of the common methods used to obtain a neural network that is able to generalize well include early stopping and cross-validation. In this work, we have chosen to modify the method outlined by Manukian *et al.* (1994). Among the many methods we tried, this modified scheme was found to result in networks with the best generalization abilities. This method can be summarized as follows:

Define the number of individual networks to be trained, N_N. Each of these networks will be trained on different data obtained by a random permutation of the entire available training data set. Thus, we divide the entire data set (consisting of N_P input-output patterns) into N_N sets which in turn consist of N_P/N_N data points for testing and $(N_P - N_P/N_N)$ data points for training. Each of these N_N networks will be trained and tested on the corresponding training and testing data for a specified number of epochs. The final network is obtained by augmenting all the trained networks to form one network with N_N N_H hidden neurons (where N_H is the number of hidden neurons in each of the individual networks). The input-hidden layer weights (V) are just augmented while the hidden-output layer weights (W) are augmented and divided by N_N. It is assumed that the output layer of neurons employ a linear activation function. This process is demonstrated in Fig. 2 for the case where there are three individual networks ($N_N = 3$) and each of these networks has three hidden neurons ($N_H = 3$). Thus, the final network will have nine hidden neurons ($N_N N_H = 9$).

The performance of this network on a validation data set that has not been used in either training or testing is observed. Since the initial random weights have a significant impact on the final network obtained, various initial random conditions are considered. This procedure is repeated for various N_H and the network with the lowest validation error has the best generalization performance.

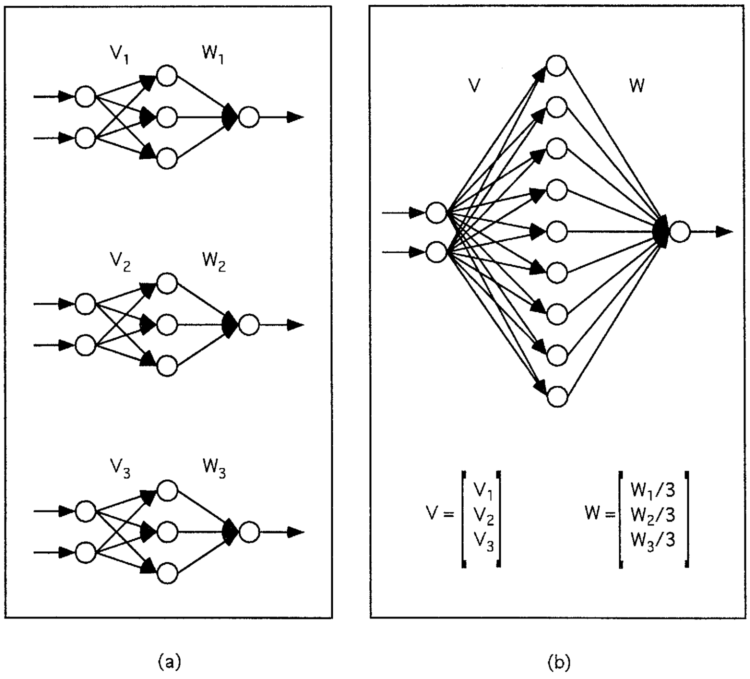
V_1
W_1
V_2
W_2
V_3
W_3
V
W
$V = \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix}$
$W = \begin{bmatrix} W_1/3 \\ W_2/3 \\ W_3/3 \end{bmatrix}$
(a)
(b)

Fig. 2. Illustration of the training methodology to improve the generalization performance of a neural network.

4. Neural Network Parameter Function Models

It has been demonstrated that neural networks are very powerful function approximators (Cybenko, 1989). In other words, using a few examples, neural networks are able to determine a functional relationship between the inputs and the outputs. This fact has been utilized in the modelling, identification and control of dynamic processes (Chen and Billings, 1992; Kurtanjek, 1994; Pollard *et al.*, 1992). Among others, Narendra and Parthasarathy (1990) provided a generalized approach to the identification of dynamic systems using neural networks. An interesting chemical engineering application was presented by Bhat and McAvoy (1990) in which they studied the estimation and control of a continuous stirred tank reactor. Many other researchers have studied neural network modelling and estimation techniques (Karim and Rivera, 1992; Yamada and Yabuta, 1993).

The typical approach to the modelling of a process has been to consider it to be a black box, and generate data using a variety of inputs and measuring the response of the system to these inputs. A neural network is then trained to mimic this input-output relationship. However, oftentimes, there exists other information about the process that can be incorporated into the models to provide a more accurate model. This has been noted by Psichogios and Ungar (1992) and Thompson and

Kramer (1994). In (Tholudur and Ramirez, 1996), we presented a parameter function modelling technique that can be used to model biotechnology processes. This utilizes readily available information in the form of material balances and uses a neural network to model just the unknown parameter functions.

A typical set of coupled differential equations that govern the dynamics of a batch process are

$$\dot{x}_1 = f_1(x_1, x_2)x_2 \tag{1}$$

$$\dot{x}_2 = f_2(x_1)(x_1 - x_2) \tag{2}$$

where x_1 and x_2 are the state variables of the system, and f_1 and f_2 are the unknown parameter functions. These types of differential equations are readily derived from some fundamental knowledge of the process and from simple conservation principles such as material and energy balances.

In the parameter function modelling, we attempt to capture the functional mappings f_1 and f_2. Process knowledge tells us that the function f_1 is dependent on both the state variables x_1 and x_2 while the function f_2 depends only upon x_1. If the states x_1 and x_2 are measurable, it is very easy to design a set of experiments that vary the initial conditions $x_1(0)$ and $x_2(0)$ and obtain the state variable measurements by sampling at periodic intervals. Then, we can rewrite the basic differential equations as follows:

$$f_1(x_1, x_2) = \frac{\dot{x}_1}{x_2} \tag{3}$$

$$f_2(x_1) = \frac{\dot{x}_2}{x_1 - x_2} \tag{4}$$

This implies that once we have the state measurements and the state derivatives at each sampling instant, we can use the above equations to obtain the values for the parameter functions at that instant. Two different neural networks can then be trained to mimic these parameter functions—the first one which has two inputs, x_1 and x_2 and one output, f_1, and the second network which has one input, x_1 and one output, f_2. Once these neural networks are trained, they can then be put back in the original differential equations and this combined neural network-differential equation model can be used for optimization.

5. Optimization of Non-Linear Models

The general optimization problem under consideration involves the constrained minimization of the scalar performance index J with respect to the initial states $\mathbf{x}(t_0)$ as given below:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) \tag{5}$$

where $\mathbf{x}$ is the state vector that describes the dynamics of the process,

$$J = \Phi\big(\mathbf{x}(t_f)\big) \tag{6}$$

defines the final state performance index to be minimized, and

$$\mathbf{x}_l \le \mathbf{x} \le \mathbf{x}_u \tag{7}$$

defines the constraints on the states, where $\mathbf{x}_l$ stands for the lower bound and $\mathbf{x}_u$ signifies the upper bound on the states. Optimization of this sort involves finding the right initial condition that minimizes the final state performance index. Many approaches are possible and in this work, we looked at two different methods—one based on the method of Luus and Jaakola (1973) and the other based on quadratic programming principles.

The Optimization Toolbox in MATLAB provides us with a constrained minimization function `constr.m` which was based on the principles of quadratic programming. This was one of the methods used for the optimization. The other method basically involves a direct search and is an iterative procedure which converges to the minimum. The algorithm is presented below:

1. Set the initial values $\mathbf{x}(t_0)$ and a range for admissible state values given by $\mathbf{r}^{(0)} = (\mathbf{x}_l + \mathbf{x}_u)/2$. Set the iteration index j to be 1.

2. Generate a specified number of random numbers for the states around the initial values such that the state constraints are not violated.

3. Integrate the system dynamic equations forward in time with each of these initial conditions and choose the one state vector that minimizes the desired performance index.

4. Reduce the allowable range by a factor ϵ, i.e. $\mathbf{r}^{(j+1)} = \epsilon\mathbf{r}^{(j)}$, set the initial state value to be the best initial condition obtained from the previous step and increment j by 1.

5. Go to Step 2 and repeat this procedure for a fixed number of iterations.

6. Case Study—Production of β-Galactosidase

Lee and Ramirez (1992) studied the production of the protein β-galactosidase using a strain of *E. coli.* They presented a dynamic model that was suitable for control purposes and applied optimal control theory to maximize the production of the protein. Their model consists of several state differential equations which directly follow from the application of conservation principles across a bioreactor. A modified differential equation set (Tholudur and Ramirez, 1997) which describes the batch dynamics is presented below:

$$\dot{x}_1 = \mu(x_2, x_4, x_5, x_6)x_1 \tag{8}$$

$$\dot{x}_2 = -Y^{-1}\mu(x_2, x_4, x_5, x_6)x_1 \tag{9}$$

$$\dot{x}_3 = R_{fp}(x_2, x_4)x_1 \tag{10}$$

$$\dot{x}_4 = 0.0 \tag{11}$$

$$\dot{x}_5 = -k_1(x_4)x_5 \tag{12}$$

$$\dot{x}_6 = k_2(x_4)(1 - x_6) \tag{13}$$

where

$$\mu = \frac{1.0x_2}{14.35 + x_2 + x_2^2/111.5}\{x_5 + x_6 R_R(x_4)\} \tag{14}$$

$$R_R = \frac{0.22}{0.22 + x_4} \tag{15}$$

$$R_{fp} = \left(\frac{0.233x_2}{14.35 + x_2 + x_2^2/111.5}\right)\left(\frac{0.0005 + x_4}{0.022 + x_4}\right) \tag{16}$$

$$k_1 = k_2 = \frac{0.09x_4}{0.034 + x_4} \tag{17}$$

The state variables are the cell density (x_1), the nutrient concentration (x_2), the foreign protein concentration (x_3), the inducer concentration (x_4), the inducer shock factor on cell growth rate (x_5), and the inducer recovery factor on the cell growth rate (x_6). The growth yield coefficient is denoted by Y and has a value of 0.51. Also, μ is the specific growth rate, R_{fp} is the foreign protein production rate and k_1 and k_2 are the shock and recovery parameters, respectively.

6.1. Modelling

The modelling process involves designing specific experiments and gathering data and postulating functional forms for each of the parameter functions and performing a curve fit to obtain the model parameters. In neural network parameter function modelling, the governing differential equations are rewritten to obtain expressions for the parameter functions as follows:

$$\mu(t, x_2, x_4) = \frac{\dot{x}_1}{x_1} \tag{18}$$

$$-Y^{-1}\mu(t, x_2, x_4) = \frac{\dot{x}_2}{x_1} \tag{19}$$

$$R_{fp}(x_2, x_4) = \frac{\dot{x}_3}{x_1} \tag{20}$$

The parameter function k_1 and k_2 cannot be modeled because they are dependent on unmeasurable states (x_5 and x_6). It is hoped that the effect of these unmeasurable states on the other parameter functions shows up in the data that is gathered and hence their effect modeled. In this simulation study, we assume that the original differential equations represent the actual experimental system that we are trying to

Table 1. Best network architecture statistics for the neural network parameter functions.

Parameter function	N_N	Best N_H	Best Validation Error
μ	3	10	5.4768477e-05
$-Y^{-1}\mu$	3	6	1.0156005e-04
R_{fp}	3	3	7.1440866e-05

model. Thus, the procedure for parameter function modelling involves generation of data using the dynamic equations and obtaining the required state derivatives and estimating the parameter functions. Three different neural networks are then trained to mimic the parameter function as given above using the procedure outlined in the Section 3. The networks for the parameter function μ and $-Y^{-1}\mu$ have three inputs, namely t, x_2, and x_4 while the function R_{fp} utilizes two inputs, namely, x_2 and x_4. Before training, the inputs to the network are normalized to lie in the range 0 to 1 in order to attach equal importance to each of the inputs. The reason for including time (t) as an input for the first two parameter functions is to compensate for the effect of the unmeasurable states and the fact that Lee (1992) had theorized the time dependence of μ.

The data for training the networks were generated by integrating the system dynamic equations with different initial conditions for the glucose concentration (x_2) and the inducer concentration (x_4), starting from $t_0 = 0$ to $t_f = 10$ with a sampling time of 1 hour. A total of 350 data points were generated with 87 data points used for the validation data set and 88 data points used for testing and 175 data points used for training the networks. The results of the training method can be summarized by Table 1 and Figs. 3–5. As can be seen, the networks have done a good job of capturing the parameter functions. These neural networks can now be put into the differential equations to provide a combined model that can be used for optimization in order to obtain the best set of initial conditions to maximize the amount of protein production.

6.2. Optimization

As mentioned earlier, two methods of constrained optimization were considered—a direct search (DS) method and a quadratic programming (QP) based method. The neural network-differential equation model was optimized for the optimum initial conditions $x_2(0)$ and $x_4(0)$. The other state variables were not varied and kept constant at $x_1(0) = 0.1$ and $x_3(0) = 0.0$. It should be noted that this combined model consists of just the first four differential equations. The aim of the optimization is to maximize the final amount of protein that can be produced while using the least amount of inducer possible and this translates to the following performance index to

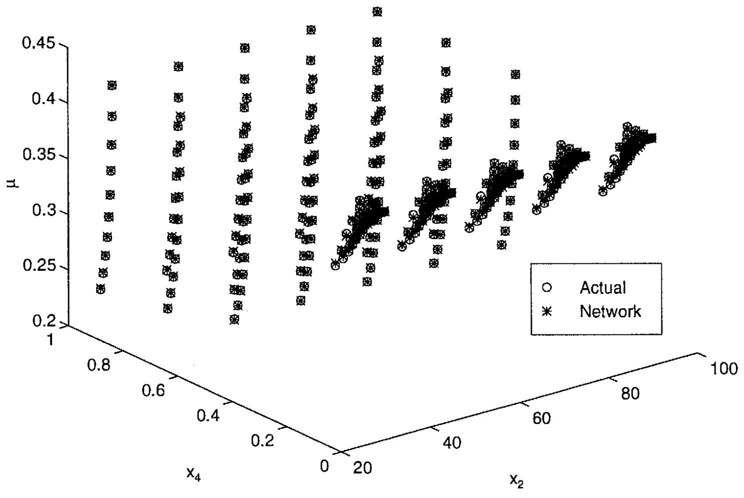

Fig. 3. Recall comparison curve for the parameter function μ.

be minimized:

$$J = -x_3(t_f)V + Qx_4(t_f)V \tag{21}$$

where V is the volume of the batch and was kept constant at 1.0 liter. The first term in the expression for the performance index corresponds to the amount of protein at the end of the batch. The second term is a penalty term to take into account the high cost of the inducer. Thus, there is a tradeoff involved in this optimization wherein we are trying to maximize the amount of protein that can be obtained by using the lowest amount of inducer possible. Lee and Ramirez considered different values for Q and we summarize the results for the cases $Q = 0.1$, $Q = 2.5$ and $Q = 5.0$ which basically refer to situations where the inducer cost is almost negligible, appreciable and prohibitive respectively.

Table 2 summarizes the optimal initial conditions predicted by the two optimization algorithms. If the neural network based model is as good as the actual model, the results of optimization on both models should be identical.

The results of optimization show that there are minor differences in the predicted optimal initial conditions. This can be attributed to two factors—approximation errors in the neural network based model and the fact that the dynamics of the unobserved states may not be truly represented in the neural network models. However, in spite of these factors, the predicted performance indices are found to be very close to the true optimum.

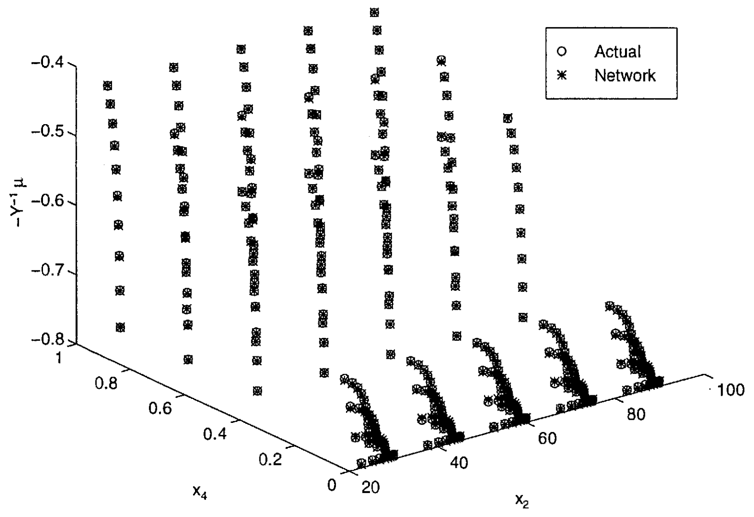

Fig. 4. Recall comparison curve for the parameter function $-Y^{-1}\mu$.

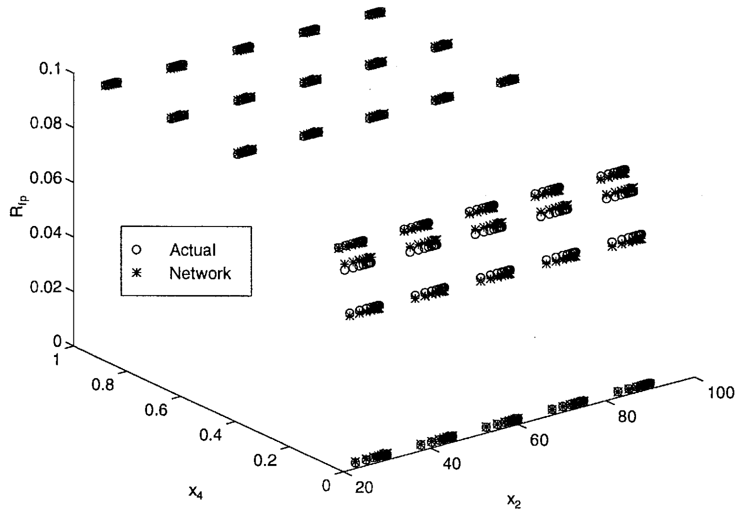

Fig. 5. Recall comparison curve for the parameter function R_{fp}.

Table 2. Comparison of the optimal results for the neural network parameter function based models.

Optimization method	Model	$x_2(0)$	$x_4(0)$	J
QP based	Actual ($Q = 0.1$)	53.4788	0.0820	−4.1784
QP based	Network ($Q = 0.1$)	58.7423	0.0827	−4.0990
DS based	Network ($Q = 0.1$)	58.6927	0.0828	−4.1002
QP based	Actual ($Q = 2.5$)	54.3116	0.0723	−3.9940
QP based	Network ($Q = 2.5$)	59.2708	0.0735	−3.9205
DS based	Network ($Q = 2.5$)	59.3438	0.0742	−3.9165
QP based	Actual ($Q = 5.0$)	54.9925	0.0651	−3.8227
QP based	Network ($Q = 5.0$)	59.6540	0.0667	−3.7551
DS based	Network ($Q = 5.0$)	59.3733	0.0662	−3.7626

7. Conclusions

This paper summarizes and demonstrates the capabilities of neural network parameter function modelling scheme to model dynamic systems. The use of conservation principles and prior knowledge about the process results in a model that is more efficient than when a black-box neural network modelling scheme is considered. These kinds of models lend themselves well to various optimization strategies. The technique is applied to the optimization of a batch bioreactor. The optimal results from both methods agree well with the true optimum. This method provides the means to model system dynamics with very simple experiments and come up with optimal operating strategies rapidly.

Acknowledgments

The authors would like to thank the Whitaker foundation and the Colorado Institute for Research in Biotechnology for financial support.

References

Bhat N. and McAvoy T.J. (1990): *Use of neural nets for dynamic modelling and control of chemical process systems.* — Comput. Chem. Eng., Vol.14, No.4–5, pp.573–583.

Chen S. and Billings S.A. (1992): *Neural networks for non-linear dynamic system modelling and identification.* — Int. J. Contr., Vol.56, No.2, pp.319–346.

Cybenko G. (1989): *Approximation by superpositions of a sigmoidal function.* — Math. Contr. Sign. Syst., Vol.2, pp.303–314.

Demuth H. and Beale M. (1994): *Neural Network Toolbox v2.0 - User's Guide.* — Natick: The Mathworks Inc.

Geman S., Bienenstock E. and Doursat R. (1992): *Neural networks and the bias/variance dilemma.* — Neural Computation, Vol.4, No.1, pp.1–58.

Karim M.N. and Rivera S.L. (1992): *Artificial neural networks in bioprocess state estimation.* — Biochem. Eng. Biotechnol., Vol.46, pp.1–33.

Kurtanjek Z. (1994): *Modelling and control by artificial neural networks in biotechnology.* — Comput. Chem. Eng., Vol.18, pp.627–631.

Lee J. (1992): *Modelling, estimation and on-line optimal control of induced foreign protein production in fed-batch reactors.* — Ph. D. Thesis, University of Colorado at Boulder, Department of Chemical Eng., USA.

Lee J. and Ramirez W.F. (1992): *Mathematical modelling of induced foreign protein production by recombinant bacteria.* — Biotech. Bioeng., Vol.39, No.6, pp.635-646.

Lee J. and Ramirez W.F. (1994): *Optimal fed-batch control of induced protein production by recombinant bacteria.* — AIChE J., Vol.40, No.5, pp.899–907.

Luus R. and Jaakola T.H.I. (1973): *Optimization by direct search and systematic reduction of the size of search region.* — AIChE J., Vol.19, No.4, pp.760–766.

Manukian N., Neuhaus J. and Wilensky G. (1994): *Multiparameter process prediction with neural networks.* — ISA Transactions, Vol.33, No.4, pp.329–338.

Masters T. (1995): *Advanced Algorithms for Neural Networks: A C++ Sourcebook.* — New York: Wiley.

McClelland J.L. and Rumelhart D.E. (1988): *Explorations in Parallel Distributed Processing.* — Cambridge: MIT Press, pp.121–137.

Moody J.E. (1992): *The effective number of parameters: an analysis of generalization and regularization in nonlinear learning systems.* — Proc. Adv. Neural Inf. Process. Syst., Vol.4, pp.847–854.

Narendra K.S. and Parthasarathy K. (1990): *Identification and control of dynamic systems using neural networks.* — IEEE Trans. Neural Networks, Vol.1, No.1, pp.4–27.

Pollard J.F., Broussard M.R., Garrison D.B. and San K.Y. (1992): *Process identification using neural networks.* — Comput. Chem. Eng., Vol.16, No.4, pp.253–270.

Psichogios D.C. and Ungar L.H. (1992): *A hybrid neural networks-first principles approach to process modelling.* — AIChE J., Vol.38, No.10, pp.1499–1511.

Tholudur A. and Ramirez W.F. (1996): *Optimization of fed-batch bioreactors using neural network parameter function models.* — Biotechnol. Prog., Vol.12, No.3, pp.302–309.

Tholudur A. and Ramirez W.F. (1997): *Obtaining smoother singular arc policies using a modified iterative dynamic programming algorithm.* — Int. J. Contr., Vol.68, No.5, pp.1115–1128.

Thompson M.L. and Kramer M.A. (1994): *Modelling chemical processes using prior knowledge and neural networks.* — AIChE J., Vol.40, No.8, pp.1328–1340.

Yamada T. and Yabuta T. (1993): *Dynamic system identification using neural networks.* — IEEE Trans. Syst. Man Cybern., Vol.23, No.1, pp.204–211.