Society of Petroleum Engineers

SPE 37557

Design of Smart Wellhead Controllers for Optimal Fluid Injection Policy and Producibility in Petroleum Reservoirs: A Neuro-Geometric Approach

Masoud Nikravesh, Lawrence Berkeley National Laboratory, BISC Program, Computer Science Division, University of California at Berkeley, SPE, Masoud Soroush, Drexel University, M. R. Johnston, CalResources, LLC., SPE, and Tad W. Patzek, University of California at Berkeley, SPE

This paper was prepared for presentation at the 1997 International Thermal Operations and Heavy Oil Symposium, 10-12 February, Bakersfield, California, U.S.A..

Abstract

In this paper, we present the next generation of "smart" controllers based on neural networks and geometric control techniques. In addition, we discuss an innovative Neural Network and Geometric Model-Based Control Strategy for developing and maintaining optimal fluid injection policy. First, the smart controller acquires data on wellhead pressures and rates continuously from the injectors . Second, a neural network model is "taught" the reservoir rate response to fluid injection pressure and vice versa. In the first case, the neural network learns how to predict the future injection rate based on injection pressure and rate. In the second case, the neural network learns how to predict the future injection pressure based on injection rate and pressure. The appropriately trained neural network can then recognize the symptoms of efficient or inefficient fluid injection around the operating point of the process. Third, feedback from the neural network models, in conjunction with geometric control, is used to design an optimal control strategy. In particular, the developed neural network-differential geometric models can be used to control and predict the behavior of individual and multiple fluid injectors.

Introduction

Oil Recovery from Petroleum Reservoirs: Fluids, such as water, carbon dioxide or steam, are injected into reservoirs to maintain pressure and displace oil. The process of fluid injection into petroleum reservoirs is known to exhibit an inherently complex, nonlinear, time varying and nonstationary behavior. Although an oil field is a complex and highly coupled system, injectors are usually controlled individually on the basis of past experience and pilots. Usually, simple single-input single-output controllers, such as Proportional-Integral-Derivative (PID) are used.

Compared with carefully-controlled and maintained pilot injectors, field injectors have a more stochastic nature and their injectivity patterns are often different. These features have made the development of first-principle models for these injectors very challenging, if not impossible. Neural networks, however, have been able to provide sufficiently accurate models of fluid injectors by using historical process data such as flow rates and pressures. Despite our incomplete knowledge, neural network models are able to predict the complex behavior of petroleum reservoirs. These models can be used to synthesize model-based controllers that are capable of providing more effective control in these processes.

In this project, we are creating an innovative Computer Assisted Operations (CAO) tool for developing and maintaining optimal injection policy for water or steam. **Figure 1** shows a simplified schematic of optimal fluid injection policy. First, the CAO system acquires continuously data from the injectors (wellhead pressures (WHPs), rates, and temperatures) and producers (WHPs, rates, etc.). Second, the pattern-by-pattern responses are inverted to "back-out" the evolution of reservoir properties such as permeability and the extent of fracturing. A simplified model of reservoir behavior that maintains first order physics is used for rapid, ideally real-time, inversion

[1]. Third, a neural network is used to "learn" the reservoir response and recognize symptoms of inefficient injection, so that control of the field improves with time [2]. Fourth, feedback from neural network is used simultaneously to decide how to adjust the coupled network of well-head controllers. This feedback is also used as input to a sophisticated reservoir simulator which is used to make long term reservoir management decisions.

In this study, the diatomaceous fields of California, which represent a $42 billion resource, have been chosen to develop and demonstrate the neural network technology for optimal fluid injection policy. However, this methodology will be applicable directly to fluid injection into tight fractured reservoirs such as the Austin Chalk and the West Texas Carbonates. Using historical data from steamdrive pilots, a neural network model has been developed to predict wellhead pressure as a function of injection rate, and vice versa. This neural network has then been used in conjunction with a mechanistic reservoir model [1, 2], to predict the extensions of hydro- and natural fractures as a function of the steam injection pressure. This approach allows predictive control.

Current paper is concerned with neural network model-based control of an oil well. A differential geometric controller is synthesized based on a neural network model of the process. The control objective is to regulate the well-head pressure by adjusting the injection rate. The performance of the model-based controller is investigated in terms of the ability of the controller to maintain the well-head pressure at a desired value over a long period of time.

Neural Network: Historically, the development of neural networks followed the philosophy of emulating the brain and the belief that if we could emulate the process of the brain, we might be able to solve many of the problems which are very difficult and seem insoluble by traditional methods.

During the last decade, a great number of neural network software packages and tools were developed. It is important to mention that the new interest in neural networks is due in part, to advances in computer technology which have made it possible to bring together a very large number of nodes and massive interconnections of simple neurons much like the human brain. However, developing a proper neural network model that is an accurate representation of the process requires a combination of art, science, and technology.

During the past several years, the successful applications of neural networks to solve complex problems have increased exponentially. In addition, in recent years, considerable attention has been devoted to the use of the neural network as an alternative approach for interpolation and extrapolation, pattern recognition, statistical and mathematical modeling. For example, back-propagation neural networks [3, 4, 5] were used to develop process models as substitutes for complicated empirical and mathematical models [2, 6]. These networks can be used as an alternative to statistical and time series analysis. Neural network analysis, unlike regression, does not require specification of structural relationships between the input and output data. However, identification using neural networks is more useful when large amounts of data are available. Once the networks are trained using sufficient information, they give excellent prediction and show excellent generalization performance. They may be trained to analyze, predict, and optimize waste management, electrochemical, reservoir, and chemical processes [7]. Self-organization maps, such as Kohonen networks [8], may be used to classify different patterns of processes. Autoassociate networks, such as Hopfield networks [9] may also be used in pattern recognition.

Among the above mentioned neural network models, multi-layer perceptron networks with a backpropagation learning algorithm are perhaps the most widely used neural networks for process modeling, identification, pattern recognition, and pattern classification. The typical network has an input layer, where data is presented to the network, an output layer, which holds the response of the network to a given input, and at least one hidden layer, which is connecting the input layer to the output layer (**Figures 2 and 3**). There is no theoretical limit on the number of hidden layers, but typically, there will be one. Each layer is fully connected to the succeeding layer with corresponding weights. The values of the weights represent the current state of knowledge of the network. These weights are adjusted to improve the network performance. They are either determined via an off-line algorithm such as the back-propagation algorithm or adjusted on-line via a learning process [10, 11, 12]. Details of the back-propagation algorithm are given in Appendix A. During the last decade, the application of neural networks with back-propagation algorithm for identification of nonlinear, time varying, and nonstationary systems has increased exponentially [13-18]. Recently, artificial neural networks have been used to model reservoir behavior under steam and water injection [2, 7, 15, 16, 19, 20], model oil and water imbibition processes [21-24], well test analysis [25, 26] and model reservoir properties [27].

Neural Network-Based Control Strategy: One of the

main difficulties in analyzing the dynamic response of industrial processes is that they are nonlinear. As a result, many are poorly predicted by linear models.

Neural networks have the potential to model very complicated nonlinear systems which are difficult to model from first principles [10, 11, 28-30]. This fact suggests that neural networks, in conjunction with a suitable control strategy such as Model-Based Control [11], Geometric Control [31-35] and Neuro-Fuzzy Control [36] can be used to control nonlinear systems.

During the past several years, the successful application of neural networks for solving complex problem has increased exponentially. Neural networks are now widely used in many nonlinear control applications [36-39].

In recent process control applications, neural networks have been used in control strategies in either directly or indirectly methods. In the direct method, the network controller is trained to learn the inverse of the process dynamics. Since the process is modeled with a separate neural network, the controller does not invert the exact process model and offset cannot be eliminated. In addition, this approach does not provide tuning parameters. Updating the process model and controller model must be done separately. The indirect method trains the neural network to predict future outputs from past and present inputs and outputs. This process model then can be used with a control algorithm to calculate the controller output.

Several backpropagation neural network control algorithms have been proposed during the past few years. Recently, neural networks have been employed in Model Predictive Control (MPC) [41-42], Internal Model Control (IMC) [43], Dynamic Matrix Control (DMC) [44], and Adaptive Control [45].

In addition to their nonlinear nature, another difficulty in many process control applications is their nonstationary nature. One of the important characteristics of each neural network model is its ability to generalize. If the network is trained based on sufficient information and the characteristics and parameters of the process do not change over time, then the network gives good prediction. It is clear that the characteristics and process parameters can change over time, and therefore, the generalization of the off-line trained network gradually deteriorates. To solve this problem, it is necessary to update the network continuously based on new information. Details of on-line training of the network are given in Appendix B.

Neuro-Geometric Controller Design for Steamflood

Model Identification: Historical data from a steam flood and the technique presented by Nikravesh *et al.* [2, 7, 15, 19, 20] are used to develop a neural network model of the injection process. For an injector, a neural network model is identified based on historical data including flow rate, well-head pressure, depth to the top of the perforations, and the length of the perforated intervals. It is assumed that the same well-operation strategy will continue into the future. If a different well-operation strategy is to be chosen in the future, then one can use the approach presented by Nikravesh *et al.* [2, 7, 15, 19, 20] and Kovseck *et al.* [46]. In this study, the process is considered as a coupled, highly nonlinear system of injectors and producers.

A neural network model with ten input nodes, five hidden nodes (with nonlinear transfer function), and one output node (with nonlinear transfer function) representing one step into the future, is developed. The resulting model has the form,

$$P(k+1)=f\{F(k-4), ..., F(k), P(k-4), ..., P(k)\} \qquad (1)$$

where P is the well-head pressure, F is the flow rate of injectant fluid, f(., ..., .) is a nonlinear and smooth function. The preceding model can easily be written is state-space form by setting

$$\begin{cases} x(k) = [F(k-4), ..., F(k-1), P(k-4), ..., P(k-1), P(k)]^T \\ u(k) = F(k) \\ y(k) = P(k) \end{cases} \qquad (2)$$

and

$$\begin{cases} x_1(k+1) = x_2(k) \\ \quad \vdots \\ x_4(k+1) = u(k) \\ x_5(k+1) = x_6(k) \\ \quad \vdots \\ x_8(k+1) = x_9(k) \\ x_9(k+1) = f\{x(k), u(k)\} \\ y(k) = x_9(k) \end{cases} \qquad (3)$$

The state space model has a relative order of one, and on-line measurements of all of the state variables of this model are available. To maintain the "accuracy" of the identified model within a desirable range, we update the model on-line (Appendix B).

Nonlinear Controller: Based on the preceding state-

space model and by using the controller synthesis results given in [33, 34], we obtain the following discrete-time nonlinear controller:

$$\begin{cases} \eta(k+1) = \alpha\,[\eta(k) - y(k)] + f\{x(k), u(k)\}, \quad \eta(0) = y(0) \\ f\{x(k), u(k)\} = \alpha\, y(k) + (1-\alpha)\,[e(k) + \eta(k)] \end{cases} \tag{4}$$

where α is an adjustable parameter such that $0 < \alpha < 1$.

The smaller the parameter α, the faster the closed-loop response. The error $e = P_{sp} - P$, where P_{sp} is the pressure set point. The controller has integral action and calculates F(k) based on x(k) and $y_{sp}(k)$. Since the equation,

$$f\{x(k), u(k)\} = \alpha\, y(k) + (1-\alpha)\,[e(k) + \eta(k)] \tag{5}$$

is nonlinear in u, then at each time instant a nonlinear algebraic equation has to be solved to obtain the value of u(k) (Appendix C). In the absence of constraints, the nonlinear controller induces the following nominal linear input-output closed-loop response:

$$y(k+1) - \alpha\, y(k) = (1-\alpha)\, y_{sp}(k) \tag{6}$$

Controller Implementation

The nonlinear controller is implemented through numerical simulations and in real-time. The regulatory performance of the nonlinear controller is compared with that of PID controller and is shown to be significantly superior to the PID controller.

Simulation Studies: The performance of the Neuro-Geometric strategy is tested on a highly nonlinear and time varying process. The process is a generic example and will be used for testing the new control strategy. The process model consists of two nonlinear ordinary differential equations [11] and is given by,

$$\frac{dx_1}{dt} = x_d\, a_1\,(x_{1f} - x_1) + a_2\, x_1 \exp\left(\frac{a_3}{x_2}\right) \phi_1(t)$$

$$\frac{dx_2}{dt} = x_d\, a_1\,(x_{2f} - x_2) + a_3\, x_1 \exp\left(\frac{a_3}{x_2}\right) \phi_1(t) + \tag{7}$$

$$a_4\, u\left[1 - \exp(a_5\, \phi_2(t))\right](x_{2f} - x_2)$$

and,

$$\phi_1(t) = \exp(a_6\, t)$$

$$\phi_2(t) = 1 - a_7\, t$$

$$y = a_8\, x_1$$

where x_1 and x_2 are state variables, u is the manipulated variable, x_d is the unmeasured disturbance, and y is the controlled variable. The model parameters (a_1 through a_8) are given in ref. [11].

Control of Process with Constant Parameters: $[a_6 = 0$ and $a_7 = 0;\ \phi_1 = \phi_2 = 1]$: The open-loop step response for a series of step changes in u (manipulated variable) is shown in **Figure 4.** It is seen that the process is highly nonlinear. A neural network model of the process has been developed. The process model has 10 input nodes (inputs include: current and four past values of x_1, and u; scaled uniformly between 0 and 1), five nodes in hidden layer with nonlinear hyperbolic tangent transfer function, and one node in the output layer (output predictions into future; scaled uniformly between 0 and 1). The model is trained via backpropagation algorithm with data generated with random changes in u. In the Neuro-Geometric Control Strategy (NGCS), equation (4) is used to find the manipulated input at each sampling time. The model predicts the controlled output.

To illustrate the performance of the NGCS, NGCS and PID are applied to the process. NGCS was tuned with a filter constant value of α=0.85. PID tuning parameters for this case study are given by Nikravesh [11]. In **Figure 5,** the setpoint tracking behaviors of NGCS and PID are compared. For setpoint changes, NGCS shows a faster response toward the setpoint than the PID control strategies. **Figure 6** shows the disturbance rejection performance of NGCS and PID controller strategies. For 20 % change in inlet flow rate as disturbance, NGCS exhibits a faster response toward setpoint when compared to PID.

Control of the Process with Time Varying Parameters: NGCS and PID control strategies will be applied to the process with time varying parameters. Two case studies are examined in which the process exhibits severe nonlinearities. In the first case study, the effect of parameter ϕ_1 on the performance of NGCS and PID

controller strategy is demonstrated. In the second case study, the effect of parameter ϕ_2 is illustrated.

Case study 1: In this study ϕ_1 is given by,

$$\phi_1(t) = (1 - 0.01\ t) \quad (8)$$

The open-loop step response for a series of step changes in u is shown in **Figure 7**. **Figure 8** shows the comparison between open-loop response of the process with time varying parameter and the process with constant parameter for +5% step changes in the manipulated variable, u. **Figures 7** and **8** show that the process does not have steady state condition and the process is nonstationary. **Figures 9** and **10** show that the setpoint tracking and disturbance rejection performance of NGCS and PID are only slightly affected by the time varying behavior of the ϕ_1. NGCS exhibits faster response time toward setpoint than PID for both setpoint tracking and disturbance rejection and is able to control the process effectively.

Case Study 2: Here we will consider the following functional form for $\phi_2(t)$;

$$\phi_2(t) = \exp (- 0.0067\ t). \quad (9)$$

The open-loop step response for a series of step changes in u is shown in **Figure 11**. **Figure 12** shows the comparison between open-loop response of the process with time varying ϕ_2 and the process with constant parameter for +5% step changes in the manipulated variable, u. **Figures 11** and **12** show that the process does not have steady state condition and the process is nonstationary. **Figures 13** and **14** show the setpoint tracking and disturbance rejection performance of NGCS and PID. Comparing **Figure 5** with **Figures 13** and **14**, one can see that the performance of PID is seriously affected due to these changes. PID shows a very large offset and very slow response with extremely poor performance. While the PID control strategy fails to control the process, NGCS shows good performance. Modification of the PID strategy to get better performance is not trivial. However, fine tuning of the NGCS controller is very easy and straightforward. For open-loop stable processes, a filter is introduced into the NGCS structure in order to ensure stability [11]. In the presence of model inaccuracies, the robustness of NGCS is improved by introducing a first order exponential filter. The value of filter constant is bounded between zero and one. Details of filter design are available in the literature.

Real-Time Controller Implementation: Nonlinear, time varying behavior is common in reservoir engineering. For example, during steam injection in the diatomite, there are several factors which cause such a behavior: (1) increase of the reservoir temperature, (2) change in the rock matrix permeability, (3) changes in the oil viscosity, (4) rock fracturing, and (5) rock dissolution. When such changes in reservoir characteristics occur, the steam injector controllers need to be retuned in order to maintain satisfactory performance. Retuning the controllers is time consuming and requires a combination of operational experience and trial-and error procedures. In addition, injection rates must be adjusted carefully, because high injection rates lead to reservoir damage, and low injection rates make oil recovery uneconomic. By using historical data for steam injectors, neural networks have been developed and used to predict the complex reservoir behavior. For instance, using historical data from the Phase III steam injection pilot in the South Belridge Diatomite (Kern County, CA), a neural network was developed to predict wellhead steam injection pressure as a function of injection rate, and vice versa. The resulting model provided an excellent input-output mapping in addition to pattern extraction, even though steam injection dynamics in this project were complex [2].

Process and Controller Models: Using historical data from the Phase III pilot, two neural network models were developed.

The first model was developed to predict the wellhead pressure as a function of current and past injection flow rate and wellhead pressure. This model can be used as process model, Equations 5 and 6, and its inverse can be used as controller. The model has 10 nodes in the input layer, 1 node in the hidden layer, and 1 node in the output layer (future wellhead pressure). As shown in **Figures 15, 16,** and **17**, the resulting models provide an excellent input-output mapping. The second model was developed to predict the steam injection rate as a function of future, current, and past wellhead pressure and past injection flow rate. This model can be used in equation 5 to invert the function f to calculate the manipulated variable, u(k). The model has 10 nodes in the input layer, 3 nodes in the hidden layer, and 1 node in the output layer (current injection rate). As shown in **Figures 18, 19,** and **20**, the resulting models provide an excellent input-output mapping.

This methodology is currently implemented at CalResources, LLC. off-line. We are working with CalResources to implement the controller on-line.

Conclusions

We presented a detailed case study in which Neuro-Geometric Control Strategy (NGCS) was applied to control an oil well injector.

For the nonlinear, time varying case, the performance of NGCS was compared to the PID control strategy. In comparison with PID control, NGCS showed significant improvement with faster response time toward the setpoint for the servo problem. The NGCS strategy was also able to reject unmodeled disturbances more effectively. NGCS showed excellent performance in controlling the processes in the region where the PID controller failed. It has been shown that the NGCS controller strategy is robust enough to perform well over a wide range of operating conditions.

Despite our incomplete knowledge, neural network models are able to predict the complex behavior of petroleum reservoirs. These models can be used to synthesize model-based controllers that are capable of providing more effective control in these processes. The performance of the NGCS was tested by applying the methodology off-line several steam injectors at Calresources, LLC. Currently, we are developing the computer interface to implement the NGCS on-line.

Acknowledgments

We thank CalResources LLC for providing the steam injector data. This work was supported by the Office of Laboratory Technology Applications, Office of Energy Research of the U.S. Department of Energy under contract No. DE-AC03-76FS00098 (ACTI) to the Ernest Orlando Lawrence Berkeley National Laboratory of the University of California.

6. Notation

B	bias
f	transfer function
u	manipulated input
x	state variables
y_l	output from hidden layer nodes
y_m	current feedback measurement
y^{Set}	setpoint
$\underline{W}$	network weights
Subscript	
1	input-hidden layer
2	hidden-output layer
Superscript	
Y	process model
U	controller
T	transpose

Greek Letters

Δ	difference
Γ	neuron transfer function

References

1. Kovscek, A.R., Johnston, R.M., and Patzek, T.W., "Evaluation of Rock/Fracture Interaction During Steam Injection Through Vertical Hydro-fractures," SPE 29622, presented at SPE 65th Western Regional Meeting, Bakersfield, CA, March 1995.
2. Nikravesh, M., Kovscek, A. R., Patzek, T. W., and Johnston, R. M., "Prediction of Formation Damage During Fluid Injection into Fractured, Low Permeability Reservoirs via Neural Networks," SPE 31103, presented at SPE Formation Damage Symposium, Lafayette, LA, 1996.
3. Charalambous, C., " Conjugate Gradient Algorithm for Efficient Training of Artificial Neural Networks," presented at *IEEE Proc. G.*, 1992.
4. Rumelhart, D.E., Hinton, G.E., and Williams, R.J., "Learning Internal Representations by Error Propagation," in *Parallel Data Processing*, D. Rumelhart and J. McClelland, Editor, MIT Press, Cambridge, MA, 1986, 318-362.
5. Widrow, B. and Lehr, M.A., "30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation," *Proceedings of the IEEE* (Sept. 1990), **78**(9), 1414-1442.
6. Chen, S., Billings, S.A., and Grant, P.M., "Non-linear System Identification Using Neural Networks," *Int. J. of Control* **52** (6), 1990, 1191-1214.
7. Nikravesh, M., Kovscek, A.R., Patzek, T.W., and Soroush, M., "Identification and Control of Industrial-Scale Processes via Neural Networks," presented at Chemical Process Control V, Tahoe City, CA, Jan. 1996.
8. Kohonen, T., *Self-Organization and Associate Memory, 2nd Ed.*, Vol., Springer-Verlag, Berlin, 1987.
9. Hopfield, J.J., "Neural networks and physical systems with emergent collective properties." *Proc. Nat. Acad. Sci. USA*, **79**, 1982, 2554-8.
10. Azimi, M. R., and R. J. Liou, "Fast Learning Process of Multilayer Neural Networks Using

Recursive Least Squares Method," *IEEE Transaction on Signal Processing*, **40**, 1992, 446-450.

11. Nikravesh, M, Dynamic Neural Network Control, *Ph.D. Dissertation*, University of South Carolina, Columbia, SC, 1994.
12. Nikravesh, M., Farell, A.E., and Stanford, T.G., "Model Identification of Nonlinear Time Variant Processes Via Artificial Neural Networks," *J. of Computers and Chemical Engineering*, **20** (11), 1996.
13. Al-Kaabi, A.U., McVay, D.A., and Lee, W.J., "Using an Expert System To Identify a Well-Test Interpretation Model," *J. Pet. Tech.* (May 1990), 654-661.
14. Bomberger, J.D., Seborg, D.E., Lightbody, G., and Irwin, G.W., " Experimental Evaluation of Neural Nonlinear Modeling," presented at Chemical Process Control V, Tahoe City, CA, Jan 1996.
15. Nikravesh, M., Dobie, C., and Patzek, T.W., "Field-Wise Waterflood Management in Low Permeability, Fractured Oil Reservoirs: Neuro-Fuzzy Approach," SPE 37523, to be presented at 1997 International Thermal Operations and Heavy oil Symposium, Bakersfield, CA, 1997.
16. Nikravesh, M., Soroush, M., Patzek, T.W., "Design of Smart Wellhead Controllers for Optimal Fluid Injection Policy and Producibility in Petroleum Reservoirs: A Neural Network Model Predictive Approach," SPE 37445, to be presented at 1997 SPE Production Operations Symposium,Oklahoma City, Oklahoma, 1997.
17. Psichogios, D.C. and Ugar, L.H., "A Hybrid Neural Network-First Principles Approach to Process Modeling," *AIChe J.* **38** (10), 1992, 1499-1511.
18. Rogers, L.L. and Dowla, F.U., "Optimization of Groundwater Remediation Using Artificial Neural Networks with Parallel Solute Transport Modeling," *Water Res. Res.*, **30** (2), 1994, 457-481.
19. Nikravesh, M., Kovseck, A.R., Murer, A.R., Patzek, T. W., "Field-Wise Waterflood Manage-ment Using Neural Network, " SPE 35721, 66th Western Regional Meeting, Alaska, 1996.
20. Nikravesh, M., Kovscek, A.R., Patzek, T.W., "Dividing Oil Fields into Regions with Similar Characteristic Behavior Using Neural Network and Fuzzy Logic Approaches," Paper NF-96103, Biennial Conference of the North American Fuzzy Information Processing Society, NAFIPS'96, Berkeley, CA, 1996.
21. Garg, A., Kovscek, A.R., Nikravesh, M., Castanier, L.M., and Patzek, T.W., "CT Scan and Neural Network Technology for Construction of Detailed Distribution of Residual Oil During Water-flooding," SPE 35737, presented at SPE Western Regional Meeting, Anchorage, AK, May 1996.
22. Nikravesh, M., Garg, A., Patzek, T.W., "Neural Network-Based Reconstruction of CT Scan Images, " Paper No 29-Chemical Physics, 32nd Western Regional Meeting and 35th Pacific Conference on Chemistry & Spectroscopy, San Francisco, CA, 1996
23. Nikravesh, M., Garg, A., Kovscek, A.R., Castanier, L. M., Patzek, T.W., "Application of Neural Network to CT Scan Imaging, " High Resolution CT work- shop, Lawrence Berkeley National Lab, May 1996.
24. Zwahlen, E., Garg, A., Nikravesh, M., Patzek, T. W., Computerized Tomography Scan and Neural Network Technology Application in Studies of One-Dimensional Imbibition in Berea Sandstone," AIChE Annual Meeting, Chicago, Illinois, 1996.
25. Juniardi, I.R. and Ershaghi, I., "Complexities of Using Neural Network in Well Test Analysis of Faulted Reservoirs," SPE 26106, presented at SPE Western Regional Meeting, Anchorage, AK, May 1993.
26. Allain, O.F. and Horne, R.N., "Use of Artificial Intelligence in Well-Test Interpretation," *J. Pet. Tech.* (March 1990), 342-349.
27. Zhou, C.D., Wu, X.L., Cheng, J.A., "Determining Reservoir Properties in Reservoir Studies Using a Fuzzy Neural Network," SPE 26430, presented at SPE 68th Annual Tech. Conf. and Exhibition, Houston, TX, October 1993.
28. Cybenko, G., "Approximation by Superposition of a Sigmoidal Function," *Math. Control Sig. System* **2**, 1989, 303-314.
29. Hecht-Nielsen, R., "Theory of Backpropagation Neural Networks," presented at IEEE Proc., Int. Conf. Neural Network, Washington, DC, 1989.
30. Jannaron, R.J., Concurrent Information Processing : *Real Time Neurocomputing for Rapidly Changing World*, book draft in review/ preparation, 1994.
31. Soroush, M. and Kravaris, C., "Discrete-Time Nonlinear Controller Synthesis by Input/Output Linearization," *AIChE J.*, **38**, 1992, 1923-1945.
32. Soroush, M. and Kravaris, C., "Feedforward/ Feedback Control of Discrete-Time Nonlinear Systems," 1992 AIChE Annual Meeting, Nov. 1-6, Miami, Paper 126e., 1992.
33. Soroush, M. and Kravaris, C., "Discrete-Time Nonlinear Controller Synthesis by Input/Output Linearization," *AIChE J.*, 38, 1992, 1923.
34. Valluri, S., Soroush, M. and Nikravesh, M., "Shortest-Prediction-Horizon Nonlinear Model

Predictive Control," *Chem. Eng. Sci.*, Submitted (1996).
35. Soroush, M., Nikravesh, M., "Shortest-Prediction-Horizon Nonlinear Model Predictive Control," 13th World Congress, International Federation of Automatic Control, IFAC, San Francisco, California, June 30-July 5, 1996.
36. Gupta, M.M., "Fuzzy Logic and Fuzzy Systems: Recent Developments and Directions," 1996 Biennial Conference of the North American Fuzzy Information Processing-NAFIPS, Berkeley, CA, 1996.
37. Langari, R. and W. Li, Analysis and Efficient Implementation of Fuzzy Logic Control Algorithms, 1996 Biennial Conference of the North American Fuzzy Information Processing-NAFIPS, Berkeley, California, *USA*, (1996), 1-4.
38. Gupta, M.M. and D.H. Rao, "Dynamic Neural Units in the Control of Linear and Nonlinear Systems, International Joint Conferences on Neural Networks," Baltimore, June 7-11, II 100-105, 1992.
39. Narendra, K.S. and K. Parthasarathy, "Identification and Control of Dynamical Systems, Using Neural Networks, *IEEE Trans. on Neural Networks*, **1** 1990, 4-27.
40. Chen, S., Billings, S. A., Grant, P.M., "Non-linear System Identification Using Neural Networks," Int. J. of Control, **51** (6), 1990, 1191-1214.
41. Garcia, C.E., Pret, D.M., and Morari, M., "Model Predictive Control - A Survey, *Automatica*, **25**, 1989, 335-348.
42. Lee M., and S. Park, S., "A New Scheme Combining Neural Feedforward Control with Model-Predictive Control," *AIChE J.*, **38** (2), 1992, 193-200.
43. Garcia, C.E. and Morari, M., "Internal Model Control. 1. A Unifying Review and Some New Results," *Ind. Eng. Chem. Process Des.*, **21**, 1982, 308-323.
44. Liu, C. C. and Chen, F. C., "Adaptive Control of Non-linear Continuous-Time Systems Using Neural Networks-General Relative Degree and MIMO Cases," Int. J. Control, **58** (2), 1993, 317-335.
45. Nikravesh, M., Farell, A. E., "Modeling and Controlling Nonlinear Chemical Processes Using Hybrid Neural Network Dynamic Matrix Control Algorithm (NNDMC)," AIChE Annual Meeting, Paper 213b8, St. Louis, Missori, Nov. 1993.
46. Kovscek, A.R., Nikravesh, M., and Patzek, T. W., "Smart Neural Network Wellhead Control Systems for Oil Fields, DOE (Patent Pending), June 14 (1996).
47. -, Neural Computing, Neural Works Professional II/Plus and Neural Works Explorer, Neural Ware Inc..
48. Shah, S. L. and Cluett, W. R., "Recursive Least Squares Based Estimation Schemes for Self-Tuning Control," *The Canadian J. of Chem. Eng.*, **69**, 1991, 89-96.
49. Ljung, L., and Soderstrom, T., "Theory and Practice of Recursive Identification," Cambridge, MA: MIT Press, 1983.
50. Ljung, L., "Analysis of a General Recursive Prediction Error Identification Algorithm," *Automatica*, **17**, 1981, 89-99.

Appendix A. The Backpropagation Network [47]

The typical Back-Propagation network always has an input layer, an output layer and at least one hidden layer (**Figure 3**). There is no theoretical limit on the number of hidden layers but typically there will be one or two. It has been shown [28, 29] that back-propagation networks with three layers can approximate any continuous nonlinear function. Some work has been done which indicates that a maximum of five layers (one input layer, three hidden, and one output layer) are required to solve arbitrarily complex pattern classification problems. In general performance of neural networks is a function of hidden layer topology. Therefore, useful generalization is affected by the number of hidden nodes and not by the number of hidden layers. Single hidden layer networks offer useful generalization and they generally train faster than multiple hidden layer networks. Each layer is fully connected to the succeeding layer. During learning, information also propagates back through the network and is used to update the connection weights.

In a neural network, the simple nonlinear elements are called nodes, neurons, or processing elements. The network consists of many of these processing elements joined and interconnected (**Figures 2 and 3**). Each connection has a corresponding weight. In neural networks, each processing element has many input paths and usually is combined by a simple summation,

$$X_j^{[s]} = f\left(\sum_i (W_{ji}^{[s]} \cdot x_i^{[s-1]})\right) = f(I_j^{[s]}) \qquad [A\text{-}1]$$

where,

$X_j^{[s]}$: current output state of jth neuron in layer [s],

$W_{ji}^{[s]}$: weight on connection joining ith neuron in layer [s-1] to jth neuron in layer s,

$I_j^{[s]}$: weighted summation of inputs to jth neuron in layer [s],

and where f is traditionally the sigmoid function and is defined by,

$$f(z) = (1.0 + e^{-z})^{-1}. \qquad [A\text{-}2]$$

Back-Propagation, The Local Error: Suppose now that the network has some global error function E. The error that back propagates through the layers is defined by,

$$e_j^{[s]} = -\partial E / \partial I_j^{[s]}. \qquad [A\text{-}3]$$

The main mechanism in a Back-Propagation Network is to forward the input through the layers to the output layer, determine the error at the output layer, and then propagate the errors back through the network from the output layer to the input layer.

Minimizing the Global Error: The objective of the learning process is to minimize the global error E of the system by modifying the weights. This can be done as follows,

$$\Delta W_{[ji]}^{[s]} = -\alpha.(\partial E / \partial W_{ji}^{[s]}), \qquad [A\text{-}4]$$

where " α" is a learning coefficient.

The Global Error Function: As the neural network learns, the information is propagated back through the network and used to update the connection weights. Learning may require showing a network many thousands of examples. The objective function for the training algorithm is usually set-up as an optimization problem and is defined as the sum of errors squared,

$$E = \frac{1}{2} \cdot \sum_k \left[\left(Y_d^{[k]} - Y_p^{[k]} \right)^2 \right], \qquad [A\text{-}5]$$

where,

$Y_d^{[k]}$: desired output,

$Y_p^{[k]}$: predicted output,

$Y_d^{[k]} - Y_p^{[k]}$: local error at output layer,

E : global error of the network.

Appendix B. Updating of the Network Recursively [12, 40, 48-50]

Consider the input-output mapping of the multilayer network is given by,

$$\underline{y}^{(Net)} = \underline{\underline{W2}}\ \underline{y}_1^{(Net)} + \underline{\theta 2}, \qquad [B\text{-}1]$$

with

$$\underline{y}_1^{(Net)} = F(\underline{\underline{W1}}\ \underline{x} + \underline{\theta 1}), \qquad [B\text{-}2]$$

and,

$\underline{\underline{W1}}$: $n_h \times n_x$; input/hidden layer weight matrix,

$\underline{\underline{W2}}$: $n_y \times n_h$; input/hidden layer weight matrix,

$\underline{\theta 1}$: $n_h \times 1$; hidden layer bias vector,

$\underline{\theta 2}$: $n_y \times 1$; output layer bias vector,

$\underline{y}^{(Net)}$: $n_y \times 1$; network outputs (predictions) vector,

$\underline{x}$: $n_x \times 1$; network inputs vector,

$\underline{y}_1^{(Net)}$: $n_h \times 1$; vector of outputs from hidden layer nodes,

n_x : number of inputs,

n_h : number of hidden layer nodes,

n_y : number of output layer nodes.

The basic least squares method produces a parameter estimation which is the result of minimization of a weighted squared error function,

$$J = \frac{1}{2N} \sum_{t=1}^{N} \lambda(t)^2 \left(\underline{y}(t) - \underline{\hat{y}}(t) \right)^2, \qquad [B\text{-}3]$$

with

$\underline{y}(t) = \underline{y}^{(observed)}(t)$: actual output, observed value for output,

$\underline{\hat{y}}(t) = \underline{y}^{(Net)}(t)$: predicted value, output from network..

A column vector, $\underline{W\Theta}$ is defined such that it includes all the elements of the matrix $\underline{W1}$, vector $\underline{\Theta 1}$, all the elements of the matrix $\underline{W2}$, and vector $\underline{\Theta 2}$,

$$\underline{W\Theta} = \begin{bmatrix} \underline{W\Theta 1} \\ \\ \underline{W\Theta 2} \end{bmatrix}, \qquad [B\text{-}4]$$

with

$$\underline{W\Theta 1} = \left[\; W1_{1,j} \; \middle| \; W1_{2,j} \; \middle| \; \cdots \middle| \; W1_{n_y,j} \; \middle| \; \underline{\Theta 1}^T \; \right]^T,$$

$j = 1, \ldots, n_x,$

$$\underline{W\Theta 2} = \left[\; W2_{1,k} \; \middle| \; W2_{2,k} \; \middle| \; \cdots \middle| \; W2_{n_h,k} \; \middle| \; \underline{\Theta 2}^T \; \right]^T,$$

$k = 1, \ldots, n_h.$

[B-5]

Therefore, to minimize the error function J, the derivative of J with respect to $\underline{W\Theta}$ should be equal to zero. The derivative of error function (Equation B-3) is defined as follows,

$$\nabla J(t) = \frac{\partial J(t)}{\partial \underline{W\Theta}(t)} = \frac{-1}{N} \sum_{i=1}^{N} \left(\underline{\underline{\Psi}}(t, \underline{W\Theta}(t)) \; \underline{\underline{\Lambda}}(t) \; \underline{\varepsilon}(t, \underline{W\Theta}(t)) \right) \qquad [B\text{-}6]$$

with

$\underline{\varepsilon}(t, \underline{W\Theta 1}(t)) = \underline{\varepsilon}(t) = \left((\underline{y}(t) - \underline{\hat{y}}(t)) \right) : n_y \times 1,$

vector of errors,

$\underline{\underline{\Psi}}(t, \underline{W\Theta}(t)) = \underline{\underline{\Psi}}(t) = \frac{\partial \underline{\hat{y}}(t)}{\partial \underline{W\Theta 1}(t)} : n_\theta \times n_y,$

matrix of gradients,

$\underline{\underline{\Lambda}}(t) : n_y \times n_y,$ diagonal matrix with $[\lambda_k(t)]^2$ element on diagonal for $k=1,\ldots, n_y,$

$n_\theta = (n_h \times n_x) + (n_y + n_h) + n_h + n_y$: number of rows in gradient matrix.

The parameters $[\lambda_k(t)]^2$ (for $k=1,\ldots, n_y$) are scalar values ($0 < [\lambda_k(t)]^2 \le 1$). In addition, each element of matrix $\underline{\underline{\Psi}}(t, \underline{W\Theta}(t))$ will be calculated as follows,

$$\hat{y}_k(t, \underline{W\Theta}) = \sum_{i=1}^{n_h} (W2_{ki}(t) \; yl_i(t) + \theta 2_i),$$

for $k=1,\ldots, n_y$

$$yl_i(t) = f\left(\sum_{j=1}^{n_x} \left(W1_{ij}(t) \; x_j(t) + \theta 1_i(t) \right) \right), \qquad [B\text{-}7]$$

with transfer function, f, defined as the hyperbolic tangent. Each element of $\underline{\underline{\Psi}}$ is given by,

$$\Psi_{ij}(t, \underline{W\Theta 1}(t)) = \frac{\partial y_j(t)}{\partial W\Theta_i(t)} =$$

$$\begin{cases} W2_{jk}(t) \; (1 + yl_k(t)) \; ((1 - yl_k(t)) \; x_i(t) & \text{if } W\Theta_i(t) = W1_{ki}(t), \; 1 \le k \le n_h, \; 1 \le i \le n_x \\ W2_{jk}(t) \; (1 + yl_k(t)) \; ((1 - yl_k(t)) & \text{if } W\Theta_i(t) = \theta 1_k(t), \; 1 \le k \le n_h \\ yl_k & \text{if } W\Theta_i(t) = W2_{jk}(t), \; 1 \le k \le n_h \\ 1 & \text{if } W\Theta_i(t) = \Theta 2_k(t), \; 1 \le k \le n_y \end{cases}. \qquad [B\text{-}8]$$

The updating routine is as follows,

$$\underline{\underline{P}}(t) = \frac{1}{\kappa_k(t)} \left\{ \underline{\underline{P}}(t-1) - \frac{\underline{\underline{P}}(t-1) \; \underline{\underline{\Psi}}(t) \; \underline{\underline{\Psi}}(t)^T \; \underline{\underline{P}}(t-1)}{\left[\kappa_k(t) \; \underline{\underline{\Lambda}}(t)^{-1} + \underline{\underline{\Psi}}(t)^T \; \underline{\underline{P}}(t-1) \; \underline{\underline{\Psi}}(t) \right]} \right\}, \qquad [B\text{-}9]$$

$$\underline{W\Theta}(t) = \underline{W\Theta}(t-1) + \underline{\underline{P}}(t) \; \underline{\underline{\Psi}}(t) \; \underline{\underline{\Lambda}}(t) \; \underline{\varepsilon}(t), \qquad [B\text{-}10]$$

$$\kappa_k(t) = \gamma_k(t-1) \; (1 - \gamma_k(t)) \Big/ \gamma_k(t), \qquad [B\text{-}11]$$

$$\Lambda^{-1}(t) = \Lambda(t-1)^{-1} + \kappa_k(t) \left[\varepsilon(t) \; \varepsilon^T(t) - \Lambda(t-1)^{-1} \right], \qquad [B\text{-}12]$$

with,

$\kappa_k(t)$: gain at time t.

Appendix C:

Neural networks can be used as a controller in either direct or indirect methods as discussed in the introduction. In the direct approach, the controller is a neural network which represents the inverse of the process model, while in the indirect approach the controller algorithm inverts the neural network process model. In the direct approach, since the controller model is trained off line, the error in the controller prediction significantly affects the controller performance. In the indirect approach, the inverse of the process model at each sampling time must be calculated. The most popular method applies an optimization routine to find the controller output.

Consider the neural network process model for input/output mapping is defined as,

$$\hat{y}(k+1) = N(\underline{uy}(k), \underline{W\Theta}(k)) = N(k), \qquad [C\text{-}1]$$

where $\underline{uy}(k)$ includes all the inputs to the network model at time (k);

$$\underline{uy}(k)^T = [y(k), y(k-1), \ldots, y(k-n_y+1),$$
$$u(k), u(k-1), \ldots, u(k-n_u+1)],$$

and $\underline{W\Theta}(k)$ includes all the weights and bias terms. The objective function (OF) for controller design is given by,

$$E(k) = \left[E(v(k), \underline{uy}(k), \underline{W\Theta}(k))\right]^2 = \left[v(k) - N(\underline{uy}, \underline{W\Theta})\right]^2, \qquad [C\text{-}2]$$

where $v(k)$ is a filtered output given by (in case of NGCS, Equation 5),

$$v(k) = h(d(k), y^{set}(k), v(k-1)). \qquad [C\text{-}3]$$

In the optimization routine, the manipulated variable, u(k) is calculated to minimize the objective function defined in equation (C-2). The back-propagation algorithm may be used, in which case the error back-propagates through the network to adjust the u(k) instead of adjusting the weights and bias terms as is done during training. Other methods such as quadratic programming also may be applied.

Another approach to finding the controller output based on the process model employs Newton's method. Several authors have employed this technique [11, 17]. The following equations are used in this method at each iteration j,

$$u(k)^{[j]} = u(k)^{[j-1]} - \frac{E(k)^{[j-1]}}{\left[\dfrac{\partial E(k)^{[j-1]}}{\partial u(k)^{[j-1]}}\right]}, \qquad [C\text{-}4]$$

with the initial guess for u(k) as follows,

$$u(k)^{[0]} = u(k-1). \qquad [C\text{-}5]$$

Details of the calculation of the Jacobian of E(k); $\left[\partial E(k)^{[j-1]} / \partial u(k)^{[j-1]}\right]$ have been presented by Nikravesh *et al.* [12].

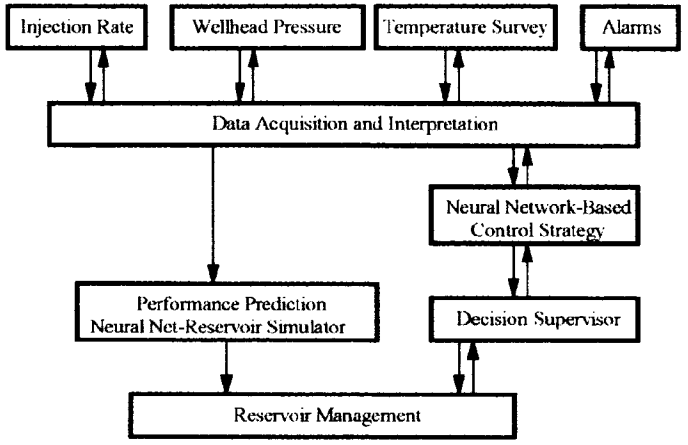

Figure 1. Simplified schematic representation of optimal injection policy.

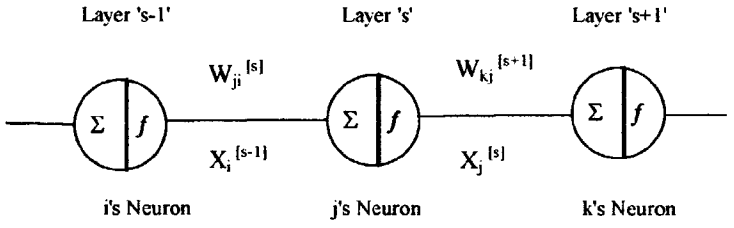

Figure 3. Schematic diagram of simple neuron model (Appendix A).

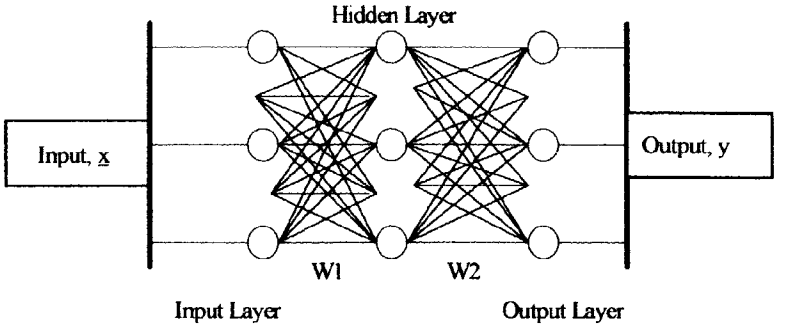

Figure 2. Typical neural network model.

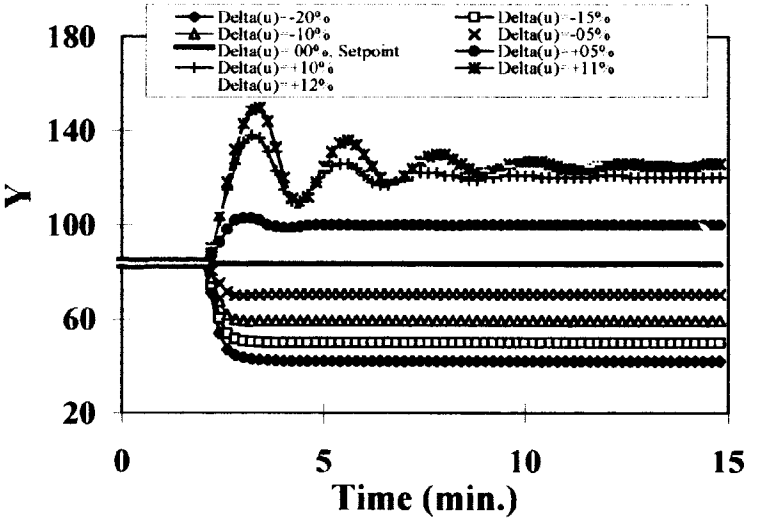

Figure 4. Open-loop response of the simulation model for step changes in manipulated variable (u).

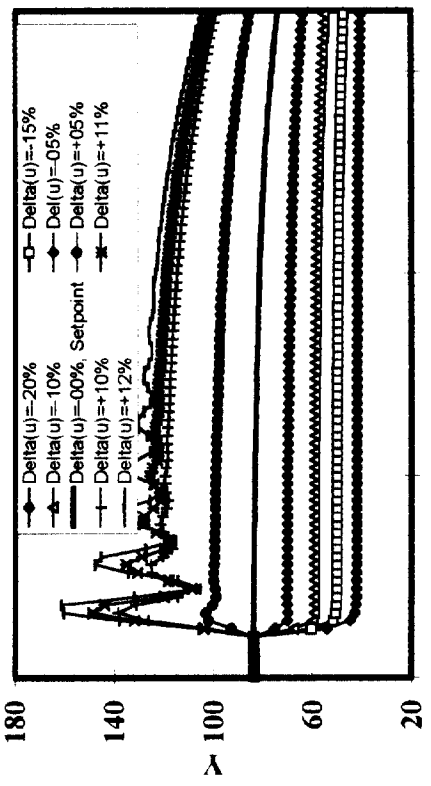

Figure 7. Open-loop response of the simulation model for step changes in manipulated variable (u), Case Study 1.

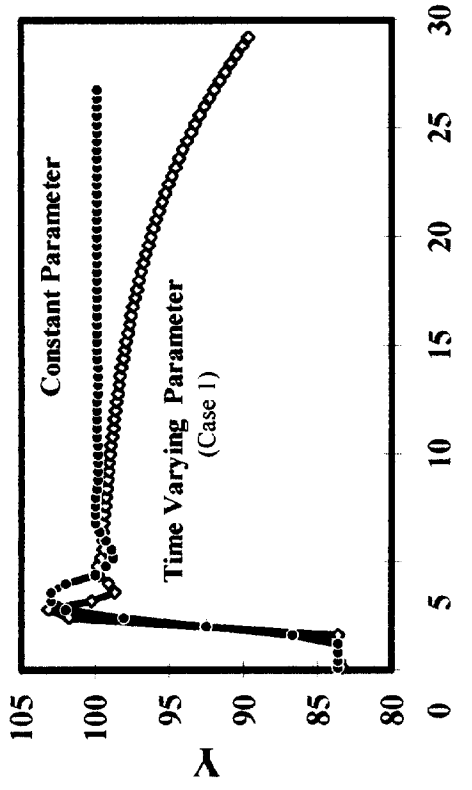

Figure 8. Comparison between open-loop response; constant parameter and Case 1; for +5% step change in manipulated variable (u).

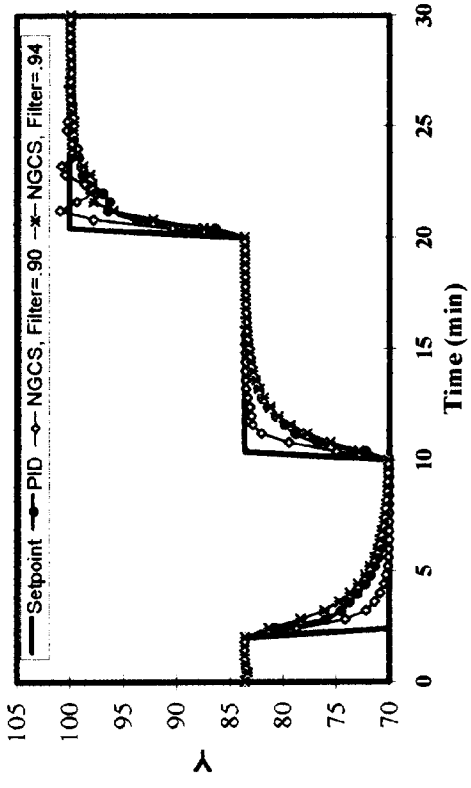

Figure 5. Setpoint tracking performance of NGCS and PID.

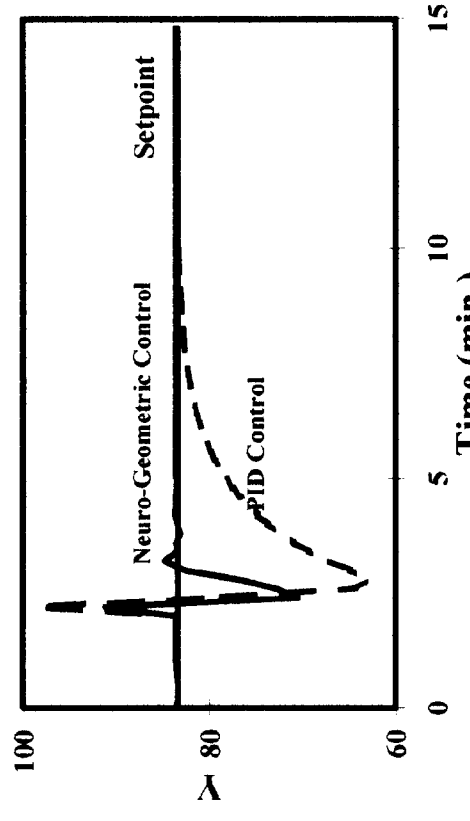

Figure 6. Disturbance rejection performance for +20% changes in the X_d.

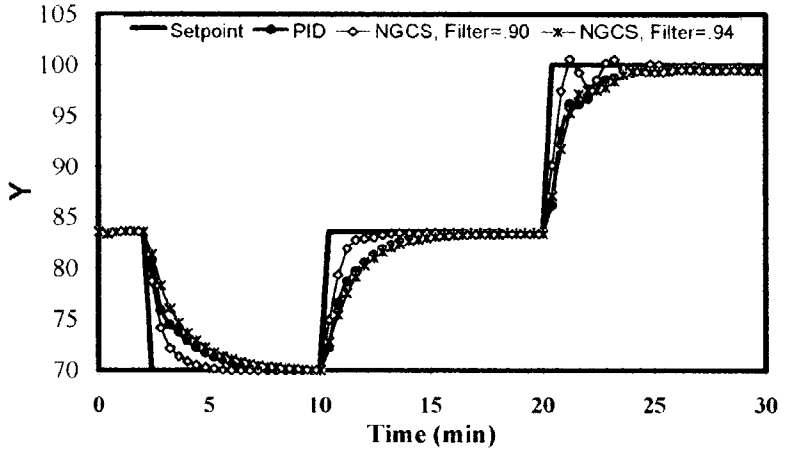

Figure 9. Setpoint tracking performance of NGCS and PID, Case Study 1.

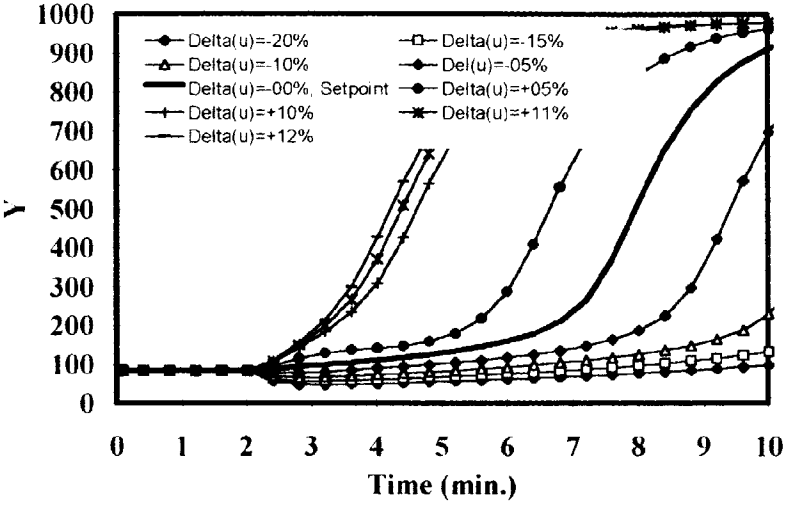

Figure 11. Open-loop response of the simulation model for step changes in manipulated variable (u), Case Study 2.

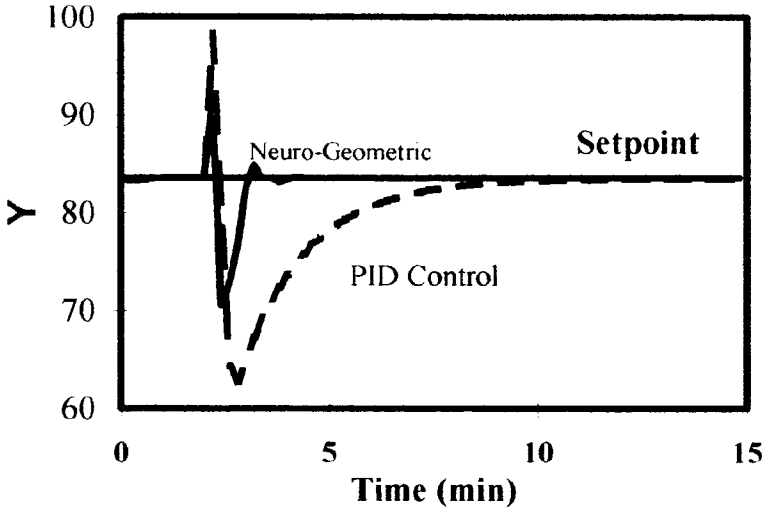

Figure 10. Disturbance rejection performance for +20% changes in the X_d, Case Study 1.

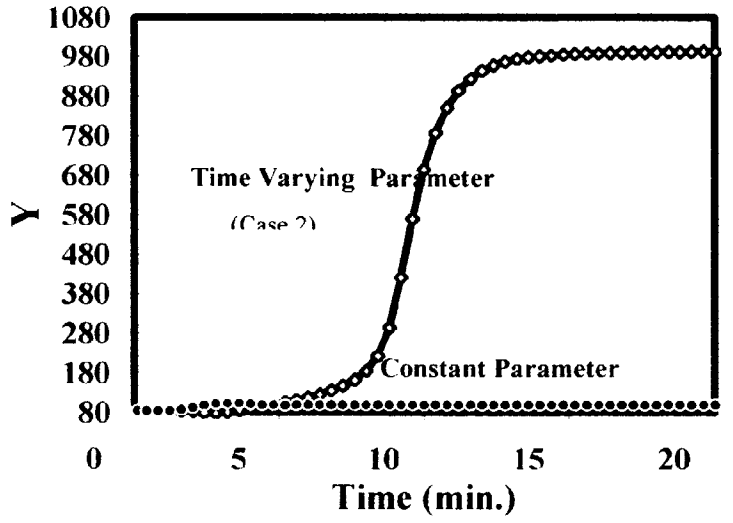

Figure 12. Comparison between open-loop responses of constant parameter and Case Study 2; +5% step change in manipulated variable (u).

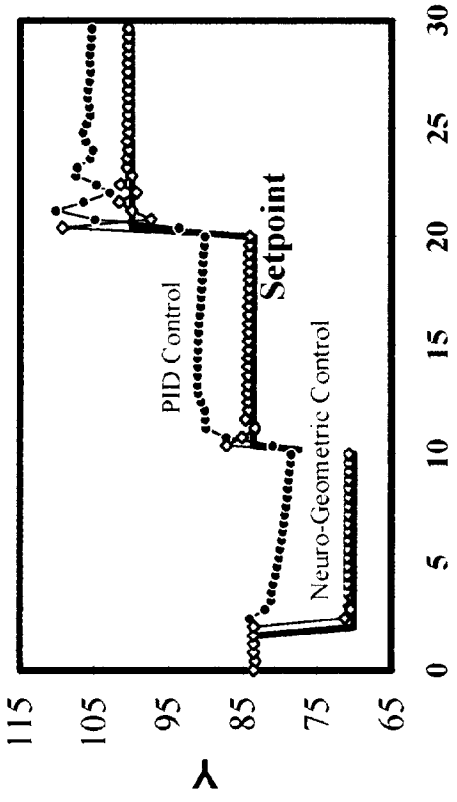

Figure 13. Setpoint tracking performance of NGCS and PID, Case Study 2.

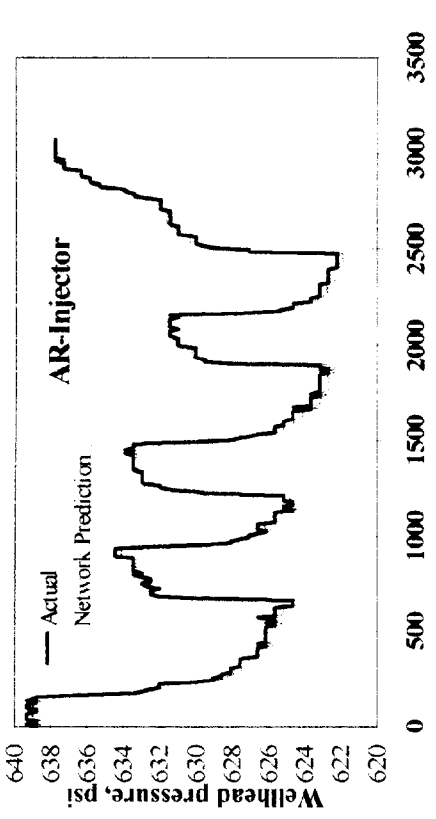

Figure 15. Neural network prediction for wellhead pressure; Injector AR.

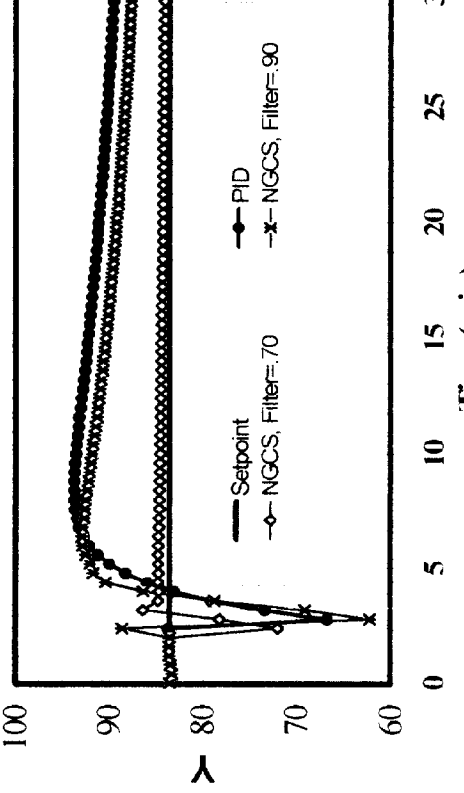

Figure 14. Disturbance rejection performance for +20% changes in the X_d, Case study 2.

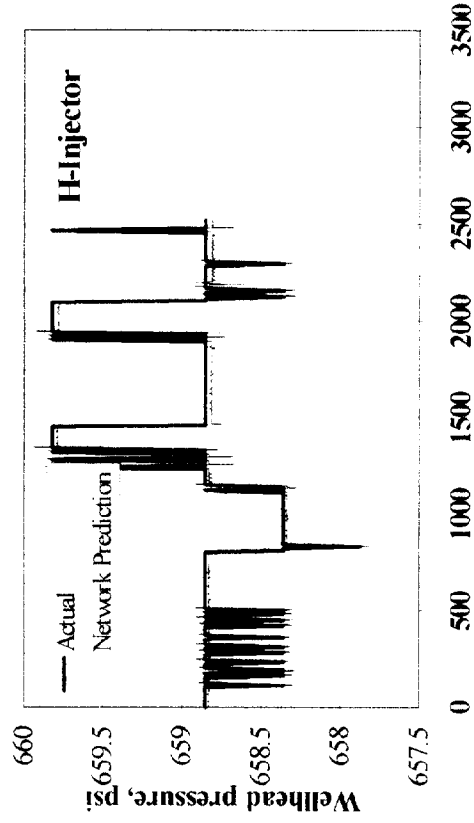

Figure 16. Neural network prediction for wellhead pressure; Injector H.

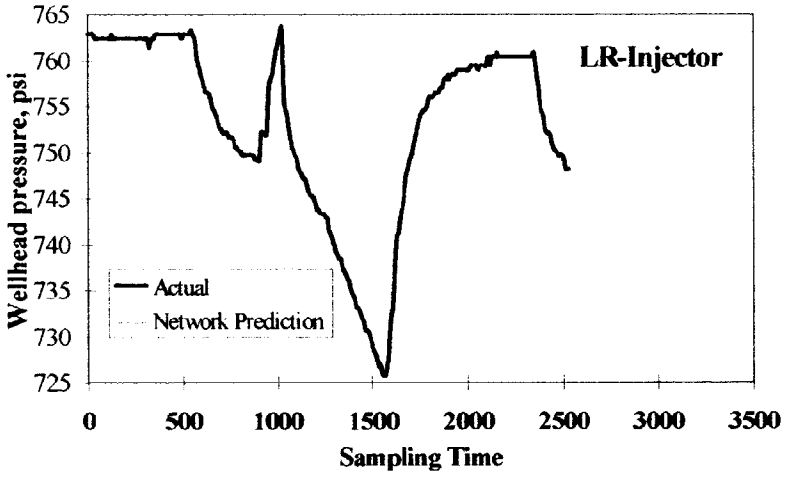

Figure 17. Neural network prediction for wellhead pressure; Injector LR.

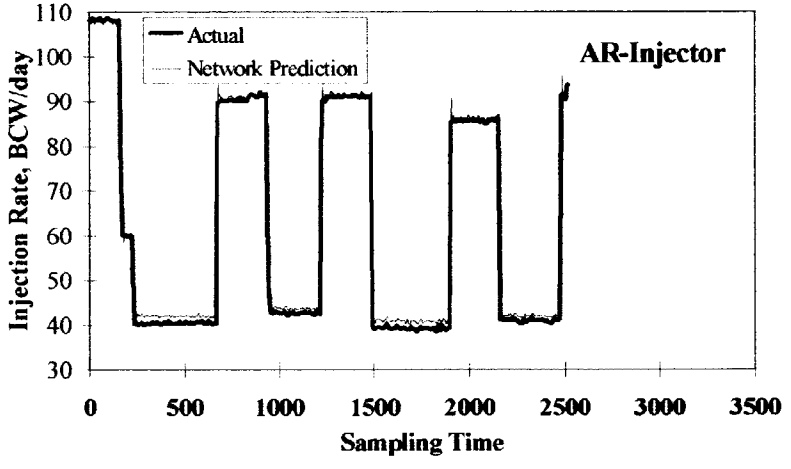

Figure 18. Neural network prediction for injection rate; Injector AR.

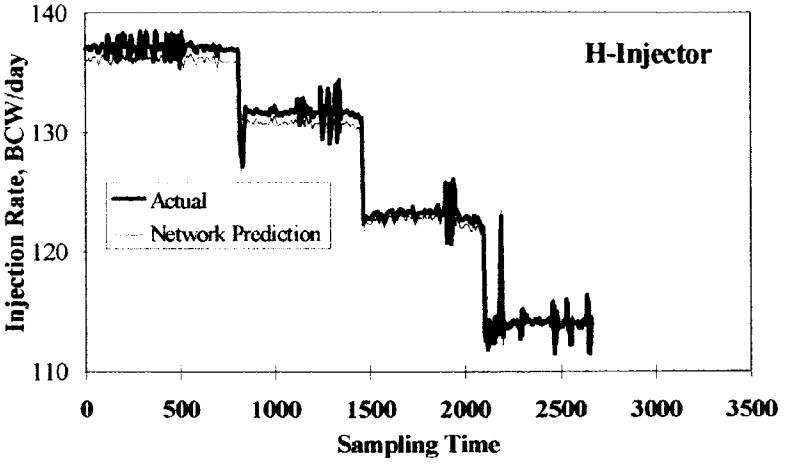

Figure 19. Neural network prediction for injection rate; Injector H.

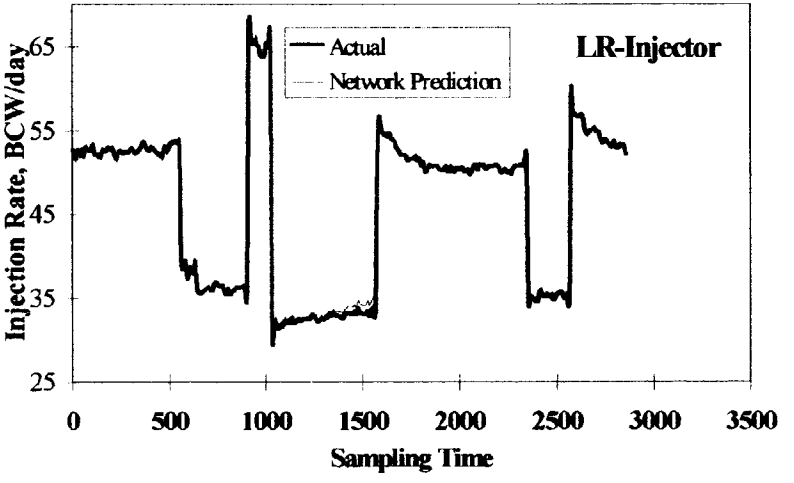

Figure 20. Neural network prediction for injection rate; Injector LR.