A Comparison of Neural Network Control Algorithms

Orlando De Jesús dorland@okstate.edu
Arjpolson Pukrittayakamee pukritt@okstate.edu
Martin T. Hagan mhagan@okstate.edu
Oklahoma State University, School of Electrical and Computer Engineering,
202 Engineering South, Stillwater, OK, 74078-5032, USA

Abstract

This paper presents a comparison of three common neural network controllers: model predictive control, NARMA-L2 control and model reference control. It describes each of the controllers and demonstrates their performance on four applications: a continuous stirred tank reactor, a robot arm, a magnetic levitation system and a simple diesel engine model. The strengths and weaknesses of each algorithm are illustrated.

1 Introduction

Neural networks have been applied successfully in the identification and control of dynamic systems ([5], [7]), but there has not been a careful comparison of the performance and design procedures of the various neurocontrollers. The objective of this paper is to compare three different neural network controllers: model predictive control [10], NARMA-L2 control [8], and model reference control [9]. These controllers are representative of the variety of common ways in which neural networks are used in control systems. We will describe each of these controllers and then apply each of them to four different systems: a continuous stirred tank reactor, a single-link robot arm, a magnetic levitation system and a simple diesel engine model. These four systems represent a variety of simple applications to which neurocontrollers can be applied. We will then compare and contrast the performances, as well as the synthesis procedures, of the three controllers on the four test problems.

There are typically two steps involved when using neural networks for control: system identification and control design. In the system identification stage, you develop a neural network model of the plant that you want to control. This network is trained offline in batch mode, using data collected from the operation of the plant. In the control design stage, you use the neural network plant model to design (or train) the controller. In each of the three control architectures described in this paper, the system identification stage is identical. The control design stage, however, is different for each architecture. (To obtain a valid comparison of the three controllers, we include a reference model between the reference signal and the controller for the predictive and NARMA-L2 controllers.)

The next three sections of this paper discuss the three controllers. This is followed by case studies, in which each of the controllers is applied to four different test problems. The final sections present some ideas about plant identification and a provide summary of results.

2 NN Predictive Control

The first stage of neural network control is the identification of the plant model, which is a multilayer network whose inputs are delayed plant inputs and outputs. Table 1 shows the plant identification parameters for the predictive controller (and the model reference controller) for the applications that will be described in a later section. The input signal for the system identification consists of a series of pulses of random amplitude and width (see Figure 9). The first two rows of the table indicate the ranges for the amplitudes and widths of the pulses. The last three rows define the architecture of the identification network.

Table 1: Parameters for Plant Identification

	CSTR	MagLev	Robot	Engine
Input range	(0, 4)	(0, 4)	(-15, 15)	(50,450)
Input interval	(5, 20)	(0.05, 5)	(0.1, 2)	(0.01, 7)
Sample time	0.2[a] 0.05[b]	0.01	0.05	0.01
Delayed inputs	2	3	2	7
Delay. outputs	2	3	2	7
Hidden layer	7	10	10	10

(a) Predictive Controller (b) Model Reference Controller

The neural network predictive controller uses the identified neural network plant model to predict future plant outputs. The predictions are used by a numerical optimization program (BFGS quasi-Newton algorithm, with a backtracking line search [4]) to determine the control signal that minimizes the following performance criterion over the specified horizon [10]:

$$J=\sum_{j=1}^{N_2}(y_r(t+j)-y_m(t+j))^2+\rho\sum_{j=1}^{N_u}(u'(t+j-1)-u'(t+j-2))^2$$

where N_2 and N_u define the horizons over which the tracking error and the control increments are evaluated. The u' variable is the tentative control signal, y_r is the desired response and y_m is the network model response. The ρ value determines the contribution that the sum of the squares of the control increments has on the performance index. Table 2 shows the values for the predictive controller parameters we will use in the applications discussed in a later section.

Table 2: Parameters for Predictive Controller

	CSTR	Maglev	Robot	Engine
N_2	7	15	9	12
N_u	2	3	2	3
ρ	2	0.01	0.005	0.1

3 NARMA-L2 Control

NARMA-L2 control transforms nonlinear system dynamics into linear dynamics by canceling the nonlinearities. The controller is simply a rearrangement of the neural network plant model, which is trained offline, in batch form. The only online computation is a forward pass through the neural network controller. The drawback of this method is that the plant must either be in companion form, or be capable of approximation by a companion form model.

To identify the system to be controlled we use the approximate NARMA-L2 model to represent the system [8]. Using the proposed equations directly can cause realization problems, because we must determine the control input $u(k)$ based on the output at the same time, $y(k)$. So, instead, we used the model

$$\begin{aligned}\hat{y}(k+d) = {} & f[y(k), y(k-1), \ldots, y(k-n+1), \\ & u(k), u(k-1), \ldots, u(k-m+1)] \\ & + g[y(k), y(k-1), \ldots, y(k-n+1), \\ & u(k), u(k-1), \ldots, u(k-m+1)] \cdot u(k+1)\end{aligned} \quad (1)$$

where $d \geq 2$. Using the NARMA-L2 model, we can define the controller

$$\begin{aligned}u(k+1) = {} & \frac{y_r(k+d) - f[Y, U]}{g[Y, U]} \\ Y = {} & [y(k), \ldots, y(k-n+1)] \\ U = {} & [u(k), u(k-1), \ldots, u(k-n+1)]\end{aligned} \quad (2)$$

which is realizable for $d \geq 2$. This controller can be implemented using the previously identified NARMA-L2 plant model. Table 3 shows the final **NARMA-L2** controller parameters for each application.

Table 3: Parameters for **NARMA-L2** Controller

	CSTR	Maglev	Robot
sample time	0.01	0.01	0.05
Delayed inputs	3	3	3
Delay. outputs	3	3	2
Hidden layer	3	10	12

4 Model Reference Control

The online computation of the model reference controller [9], as with NARMA-L2, is minimal. However, unlike NARMA-L2, the model reference architecture requires that a separate neural network controller be trained off-line, in addition to the neural network plant model. The controller training is computationally expensive, since it requires the use of dynamic backpropagation ([9], [6]). On the positive side, model reference control applies to a larger class of plant than does NARMA-L2 control. Each network has two layers, and you can select the number of neurons to use in the hidden layers. There are three sets of delayed controller inputs: reference inputs, controller outputs and plant outputs. Table 4 shows the final Model Reference controller parameters for each test

Table 4: Parameters for Model Reference Controller

	CSTR	Maglev	Robot	Engine
Del. reference	3	2	2	2
Delayed inputs	3	2	2	2
Delay. outputs	3	2	1	1
Hidden layer	8	13	13	15

5 Case Studies

5.1 Continuous Stirred Tank Reactor (CSTR)

The first application is a catalytic Continuous Stirred Tank Reactor (CSTR) [1]. The dynamic model of the system is

$$\frac{dh(t)}{dt} = w_1(t) + w_2(t) - 0.2\sqrt{h(t)}$$

$$\frac{dC_b(t)}{dt} = (C_{b1} - C_b(t))\frac{w_1(t)}{h(t)} + (C_{b2} - C_b(t))\frac{w_2(t)}{h(t)} - \frac{k_1 C_b(t)}{(1+k_2 C_b(t))^2}$$

where $h(t)$ is the liquid level, $C_b(t)$ is the product concentration at the output of the process, $w_1(t)$ is the flow rate of the concentrated feed C_{b1}, and $w_2(t)$ is the flow rate of the diluted feed C_{b2}. The input concentrations are set to $C_{b1} = 24.9 mol/cm^3$ and $C_{b2} = 0.1 mol/cm^3$. The constants associated with the rate of consumption are $k_1 = 1$ and $k_2 = 1$. The objective of the controller is to maintain the product concentration by adjusting the flow $w_2(t)$. To simplify the demonstration, we set $w_1(t) = 0.1 cm^3/s$. The level of the tank $h(t)$ is not controlled for this experiment.

For the predictive controller we noticed that the control action and system response tend to be smoother as the control horizon increases. However, this option requires more computation. We also noticed that ρ strongly affects system stability. Values of $\rho < 1$ generated a fast response. However, the system became unstable for certain transitions. For long control horizons, we noticed that lower ρ values resulted in less restrictive control action - allowing abrupt transitions. For short control horizons the control action could change faster, so we could excite unstable modes of the plant. Figure 1 shows the CSTR response for the parameters given in Table 2, which produced the best results.

To avoid large offset and control action saturation using the NARMA-L2 controller, the sampling interval was reduced to 0.01 seconds. Figure 2 shows the system response for the NARMA-L2 controller. We notice that saturation and rapid oscillation of the plant output is avoided. However, the NARMA L2 controller generally produces more oscillatory control signals than the other controllers discussed here.

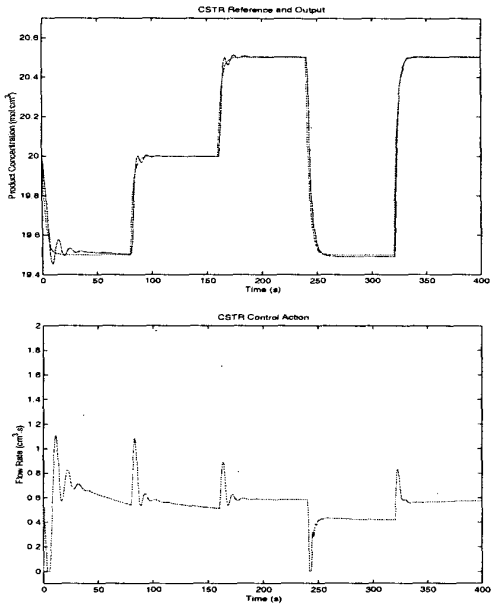

Figure 1: CSTR response and control action using the Predictive Controller.

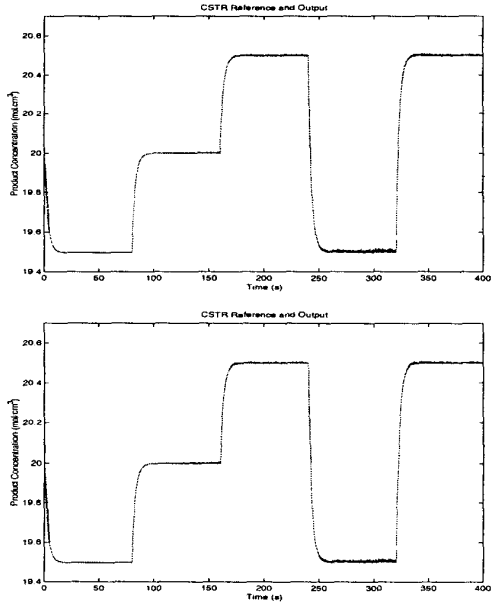

Figure 2: CSTR response and control action using the Narma-L2 Controller.

To obtain the model reference controller for the CSTR, the plant model was trained using a sample time of $ts = 0.05$. We found that normalization was critical to obtaining a good controller. Controllers trained without normalized data were unsuccessful. Another important factor for the accurate training of the controller was considering the training modifications suggested in [3]. The model reference controller requires dynamic training because of its recurrent architecture. The initial training resulted in saturation, oscillation and bad performance from the controller. As detailed in [3], the error surface of a recurrent network has spurious minima that occur in narrow valleys. These valleys can be mitigated by switching training sequences, using small random initial conditions and employing regularization[3]. Using these techniques, we were able to obtain the controller response shown in Figure 3.

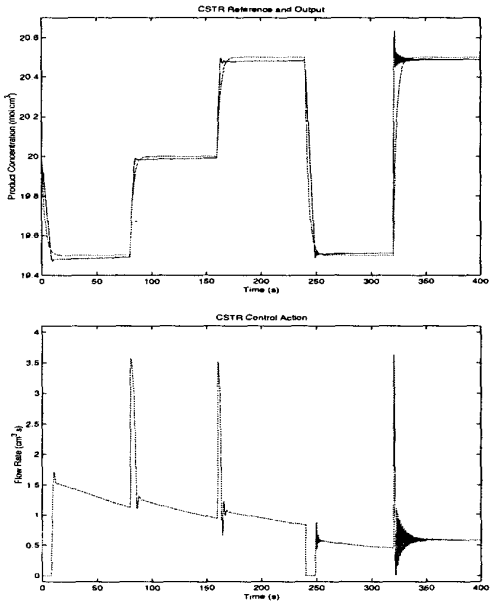

Figure 3: CSTR response and control action using the Model Reference Controller.

5.2 Magnetic Levitation System (MagLev)

In the second test problem, the objective is to control the position of a magnet suspended above an electromagnet, where the magnet is constrained so that it can only move in the vertical direction. The equation of motion is:

$$\frac{d^2y(t)}{dt^2} = -g + \frac{\alpha}{M}\frac{i^2(t)}{y(t)} - \frac{\beta}{M}\frac{dy(t)}{dt}$$

where $y(t)$ is the distance of the magnet above the electromagnet, $i(t)$ is the current flowing in the electromagnet, M is the mass of the magnet, and g is the gravitational constant. The parameter β is a viscous friction coefficient that is determined by the material in which the magnet moves, and α is a field strength constant that is determined by the number of turns of wire on the electromagnet and the strength of the magnet.

For the predictive controller, we again found that stability was strongly influenced by the selection of ρ. As we decrease ρ the control signal tends to change more abruptly, generating a noisy plant output. However, when we increase

ρ to much the control action is excessively smooth. Figure 4 shows the system response for the best predictive controller configuration. We noticed less oscillation and saturation in the control action. We also eliminated the overshoot in the system response.

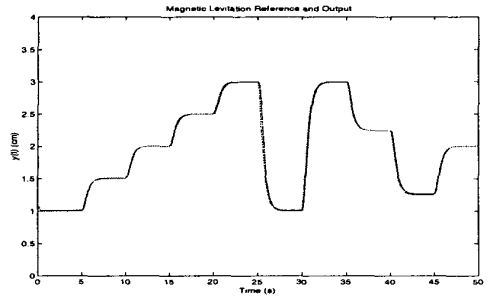

Figure 4: MagLev response using the Predictive Controller.

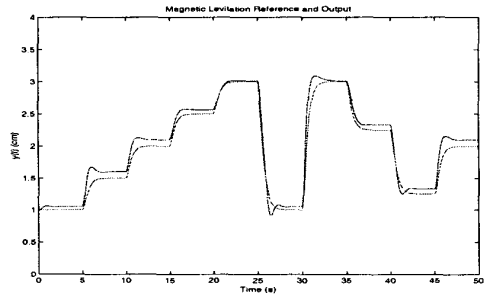

Figure 5: MagLev response using model reference control.

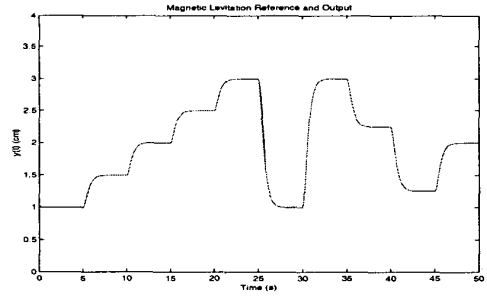

Figure 6: MagLev response using the Narma-L2 controller.

The neural network model of the plant used with the predictive controller was used to train a model reference controller. Figure 5 shows the system response and the control actions for the system after training. We see that the system has a steady state error of between 3% and 6%. Figure 6 shows the system response for the NARMA-L2 controller. The response is very similar to Figure 4. The main difference between the controllers is the control action. The NARMA-L2 controller is generally more oscillatory.

5.3 Robot Arm

The objective in this application is to control the movement of a simple, single-link robot arm, where the equation of motion for the arm is:

$$\frac{d^2\phi}{dt^2} = -10\sin\phi - 2\frac{d\phi}{dt} + u$$

where ϕ is the angle of the arm, and u is the torque supplied by the DC motor.

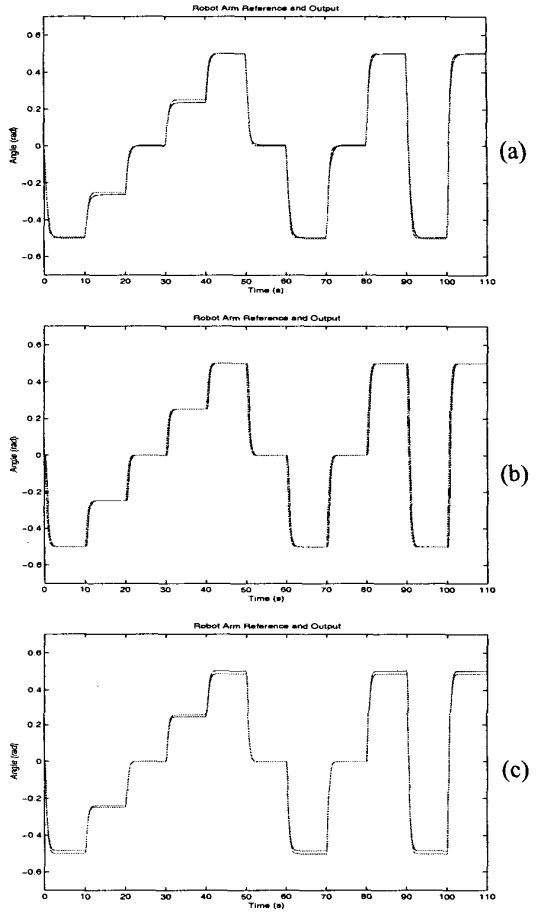

Figure 7: Robot arm response.

Figure 7a shows the response of the system using the model reference controller. The system is able to follow the reference, and the control actions are smooth. If we compare the results shown in Figure 7b for the predictive controller with the model reference controller, we can see that the predictive controller has a faster response. Figure 7c shows the robot arm response for the NARMA-L2 controller.

5.4 Simple Diesel Engine Model

The objective of this application is to control the speed of a diesel engine [2] by adjusting the fueling. The fueling has a variable time delay as a function of the engine speed. That delay can vary from 10 to 50 milliseconds and is reduced by the load applied to the engine. The effect of fueling $F_d(s)$ is

modeled by the following equation:

$$N(s) = \frac{60}{2 \cdot \pi \cdot I \cdot s} F_d(s)$$

where the engine inertia is $I = 2\ lb\text{-}ft\text{-}sec^2$, and $N(s)$ is the engine speed. A constant feedback gain of 0.2 was introduced to simulate viscous friction. The delay in the fuel system is $d(t) = 0.003 + 29.166 \cdot N_s(t)$, where $N_s(t)$ is a saturated engine speed between 500 and 2400 rpm.

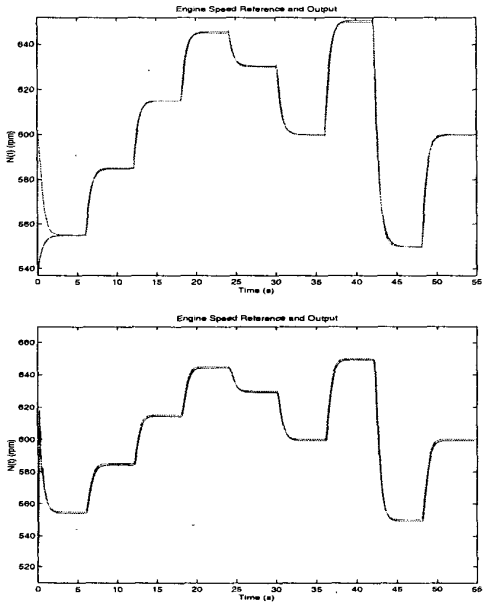

Figure 8: Engine speed using the model reference controller and Predictive Controller.

Figure 8a shows the engine speed for the model reference controller. The variable delay in the fueling is noticeable here. We would expect the engine response to follow the soft reference more closely. However, each fueling action required some time to generate a change in the engine speed.

Figure 8b shows the engine response using the predictive controller. Cost horizon values smaller than 12 resulted in no real control action. We assume that a cost horizon of 12 steps was able to cover the variable fueling delay of up to 50 milliseconds. The predictive control produces closer tracking than the model reference control. We may assume that the predictive controller is able to better "predict" the behavior of systems with variable time delay.

The NARMA-L2 controller for the diesel engine was unsuccessful. From Eq. (1) and Eq. (2) the output of the plant model is computed by adding the output of the f neural network to the product of the g network output times the plant input $u(k)$. When the plant identification was complete, the weights of the g neural network were very small. So, the response was provided only by the f network. This was due to the variable fueling delay. The plant output was generally uncorrelated to the plant input applied 10 to 40 milliseconds before. The engine model is therefore unable to create a useful NARMA-L2 controller. This resulted in a saturated control action without regulation. NARMA-L2 control will not work for systems in which the delay is not fixed and well-defined.

6 Comments on plant identification

The performances of the three controllers discussed in this paper depend on the accuracy of the plant identification. In this section we will discuss some of the procedures that can be used to improve the plant identification.

Controller performance tends to fail in either steady state operation, or transient operation, or both. When steady state performance is poor, it is useful to increase the time interval over which the input signals for the plant identification remain constant. Unfortunately, within a training data set, if we have too much data in steady state region, the training data may not be representative of typical plant behavior. This is due to the fact that the input and output signals do not adequately cover the region that is going to be controlled. This will result in poor transient performance. We need to choose the training data so that we produce adequate transient and steady state performance. The following example will illustrate the performances of three predictive controllers when we use different ranges for the pulse widths of the input signal to generate the training data.

6.1 Case I

We found that it took about 4.5 seconds for the magnetic levitation system to reach steady state. Therefore, we first specified a pulse width range of $0.01 < \tau < 5$. After training the network with the data set shown in Figure 9, the system was unstable.

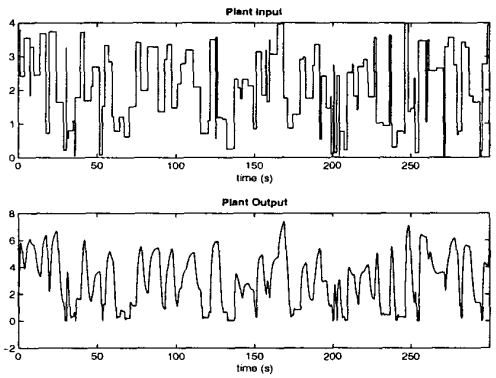

Figure 9: Training data with a long pulse width.

6.2 Case II

Based on the poor response of the initial controller, we determined that the training data did not provide significant coverage. Therefore, we changed the range of the input pulse widths to $0.01 < \tau < 1$, as shown in Figure 10. From this figure, it is clear that the training data is more dense and

provides wider coverage of the plant model input space than the first data set. After training the network using this data set, the system was stable, although it resulted in larger steady-state errors.

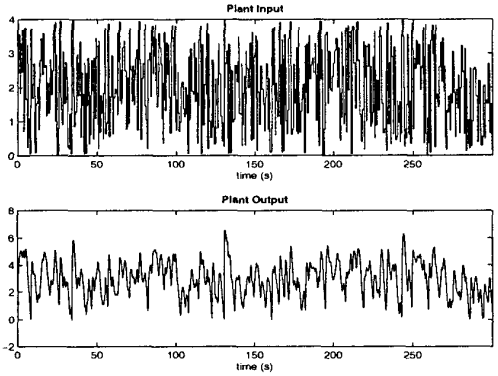

Figure 10: Training data with a short pulse width.

6.3 Case III

In the third test, we combined short pulse width and long pulse width (steady state) data. The long pulses were concentrated only on some ranges of plant outputs. For example, we chose to have steady-state data over the ranges where the tracking errors from case II were large, as shown in Figure 11. Figure 4 shows the system output of the predictive controller. Steady state errors were decreased, and the transient performance was adequate in all tested regions.

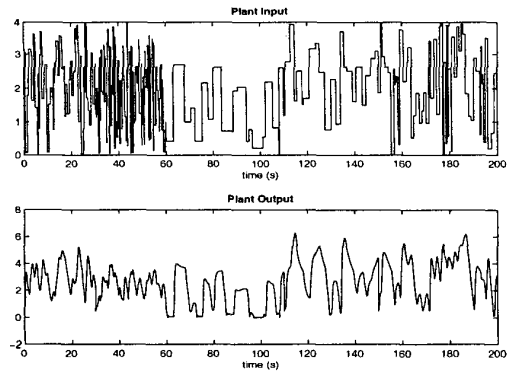

Figure 11: Training data with mixed pulse width.

7 Summary

This paper presented three representative neural network controllers and demonstrated their performance on four applications. The model predictive controller uses a neural network model to predict future plant responses to potential control signals. An optimization algorithm then computes the control signals that optimize future plant performance. The controller requires a significant amount of on-line computation, since an optimization algorithm is performed at each sample time to compute the optimal control input.

The NARMA-L2 controller requires the least computation of the three architectures described in this paper. The controller is simply a rearrangement of the neural network plant model, which is trained offline, in batch form. The only online computation is a forward pass through the neural network controller. The drawback of this method is that the plant must either be in companion form, or be capable of approximation by a companion form model.

The online computation of the model reference controller, like NARMA-L2, is minimal. However, unlike NARMA-L2, the model reference architecture requires that a separate neural network controller be trained off-line, in addition to the neural network plant model. The controller training is computationally expensive, since it requires the use of dynamic backpropagation. On the positive side, model reference control applies to a larger class of plant than does NARMA-L2 control.

8 References

[1] Brengel, D.D.; Seider, W.D., "Multistep Nonlinear Predictive Controller," Ind. Eng. Chem. Res., Vol. 28, 1989, pp. 1812-1822.

[2] De Jesús, O., "Reinforcement Learning Control Applied to Diesel Engines," *Master Thesis*, Oklahoma State University, Stillwater, 1998.

[3] De Jesús, O.; Horn, J.M.; Hagan, M.T., "Analysis of Recurrent Network Training and Suggestions for Improvements," To be publish in: *Proceedings of the INNS-IEEE International Joint Conference on Neural Networks*, Washington DC, July 2001.

[4] Dennis, J.E.; Schnabel, R.B., *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Englewood Cliffs, NJ: Prentice-Hall, 1983.

[5] Hagan, M.T.; Demuth, H.B., "Neural Networks for Control," *Proceedings of the 1999 American Control Conference*, San Diego, CA, 1999, pp. 1642-1656.

[6] Hagan, M.T.; De Jesús, O.; Schultz, R., "Training Recurrent Networks for Filtering and Control," Chapter 12 in *Recurrent Neural Networks: Design and Applications*, L. Medsker and L.C. Jain, Eds., CRC Press, 1999, pp. 311-340.

[7] Hunt, K.J.; Sbarbaro, D.; Zbikowski, R.; Gawthrop, P.J., "Neural Networks for Control System - A Survey," *Automatica*, Vol. 28, 1992, pp. 1083-1112.

[8] Narendra, K.S.; Mukhopadhyay, S., "Adaptive Control Using Neural Networks and Approximate Models," *IEEE Transactions on Neural Networks* Vol. 8, 1997, pp. 475-485.

[9] Narendra, K. S.; Parthasrathy, A. M., "Identification and control for dynamic systems using neural networks", *IEEE Transactions on Neural Networks* Vol. 1, pp. 4-27, 1990.

[10] Soloway, D.; Haley, P.J., "Neural Generalized Predictive Control," *Proceedings of the 1996 IEEE International Symposium on Intelligent Control*, 1996, pp. 277-281.