3.0 Feed-Through Neural Network

In this chapter, we are examining the use of neural networks for adaptive control in a closed-loop system in combination with traditional state-space control methods. Because most engineering situations involve building subsystems that will fit into some larger closed-loop system, the aim of this effort has been to see if overall system performance can be improved by using adaptive filters to control a plant with linear state feedback. Placing an adaptive system within a closed-loop system raises many issues of stability and implementation. In this chapter, we have looked at the effects of locating the adaptive filter inside and outside the closed-loop plant and controller. Also, we have examined the effect of applying this type of control to stable and unstable, nonlinear plants.

3.1 Methodology

Two control methods are considered because the nonlinearities of the plant are degrading performance. The first control method has the adaptive filter in the closed loop, and the error signal is used to adapt the combined dynamics of the filter and the plant to produce an overall desired response. The desired reference response is the closed loop model of the linearized plant and the estimator/regulator, as seen in Figure 3.1. The advantage to this method is that the nonlinearities of the plant can be directly influenced by the neural network. The disadvantage is that, if the weights in the neural network are randomly initialized, the gain from the unconverged neural network will probably drive an unstable plant unstable before the neural network can converge. The weights must be initialized so that the net effect of the neural network is simply to feed the control signal through to start the system in a stable state. The weights give an initial “unity gain” for the neural network, which is the starting point for the adaptation.

For the second control method, the state space controller and the plant are treated as a single unit, and the neural network is wrapped around the outside

of the closed loop, as seen in Figure 3.2. This method has the advantage that the weights for the neural network can be randomly initialized because the neural network is not in the directly-closed loop; it does not affect the initial stability of the system. The disadvantage of this method is that the system for which the neural network is trying to compensate is the closed-loop dynamics of the open-loop plant due to the dynamics of the estimator when the loop is closed through the controller.

In our simulation experiments, we use a fourth order nonlinear plant model of the form shown in Figure 3.3. The stable plant has two second-order modes, each lightly damped, with a damping coefficient of approximately .05. The natural frequency of the first filter in the series is 1 rad/sec; the natural frequency of the second filter is twice the first, 2 rad/sec. The saturation function inserted in between is a hyperbolic tangent function. For the unstable plant, the damping on the second filter is made negative. The estimator/regulator control system was designed based on the linear model, which was exactly nonlinear plant without the nonlinearity, to give the closed-loop system a response with 0.707 damping. Due to the nonlinearity, the closed-loop system does not perform as well as expected.

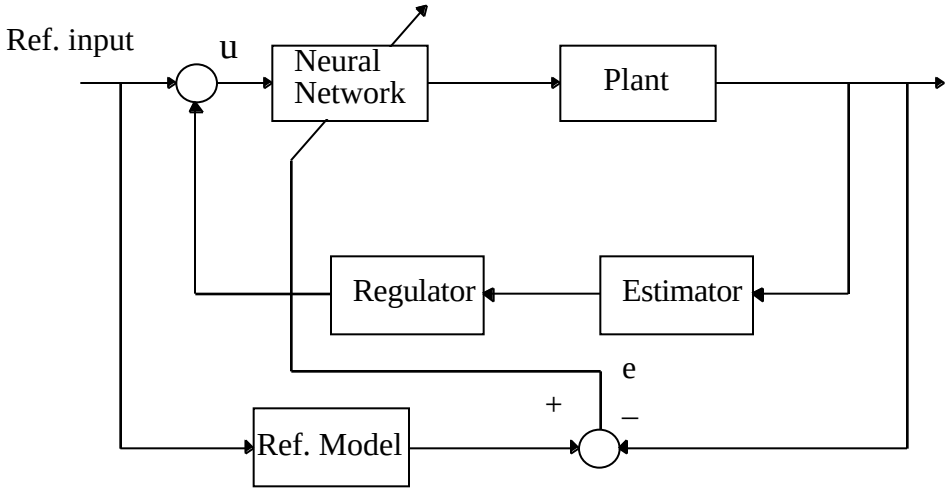

Figure 3.1 Case 1: Neural Network Inside the Closed-Loop

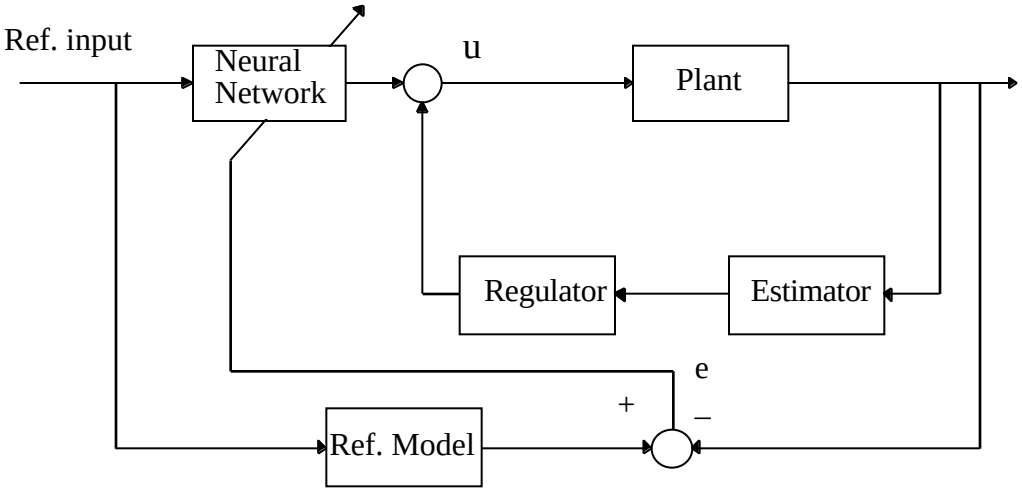

Figure 3.2 Case 2: Neural Network Outside the Closed-Loop

This plant is an approximate model of an actuator that has both saturation properties and structural flexibility. A linear model of this plant is used in the design of the regulator and estimator used to close the control loop around the plant. If the plant did not have the saturation, the response of the closed-loop system would exactly match the desired response.

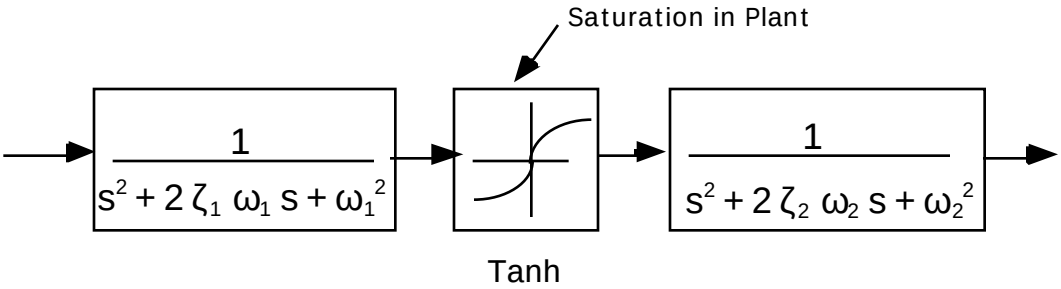

Figure 3.3 Stable, Fourth-Order Nonlinear SISO Plant

Looking at the Figures 3.1 and 3.2, it can be seen that the neural networks can be isolated from the rest of the system. When isolating them, the network breaks into an input, an output, and an error signal. The input for Case 1 is the

command input to the plant. The input for Case 2 is the reference input. The output for Case 1 is what we call “the filtered input” and is directly fed into the plant. The output for Case 2 is a “filtered reference” that is then combined with the estimated states in order to form the input for the plant. The error is always the desired plant output minus the actual plant output. The desired plant output is the reference model, and for the examples used in this chapter, the reference model is the linear model with its control system.

The neural network is a fully-connected, feed-forward network, as seen in Figure 3.4. The network has one hidden layer with one node on the hidden layer being linear and the rest of the nodes have a hyperbolic squashing function. The output layer is also linear. The number of inputs is varied for the experiments. The inputs are a tap delay of the input to the plant. A tap delay line of thirty is used as the input, which works well. A tap delay line of twenty is also tried, with a reasonable amount of success. The number of nodes in the hidden layer is set at thirty, while keeping one of the nodes linear. There is a single output node in all cases.

The adaptation of the weights of the neural network is done with the standard back propagation algorithm. In Chapter 5, two new update algorithms, developed to take advantage of a priori knowledge and the closed-loop configuration, will be derived to replace back propagation. Special consideration has to be given to the fact that one of the nodes in the hidden layer is linear. Initially, it is desired that the performance be acceptable, with no degradation in performance during training. It is not desirable, nor necessary, to step too far on each iteration. The learning rate is to be set very small. When the learning rate is small, there is a greater chance that the output error will converge to a local minima of the weight space. This is the engineering trade-off that we are willing to make in order to guarantee initial performance.

The training of the neural network uses with white noise at the reference input because white noise has a broad band frequency content. A uniform random

number generator creates a random number at each iteration. This eliminates the possibility that the neural network will memorize the solution because there is no single training set on which to perform multiple iterations. The neural network is never seeing the same training data twice. If the neural network is trained with a finite training set that is repeated, there is a risk that the neural network will memorize the training set. Checking to see if the neural network is converging, we compare the desired step response and the actual step response.

The key to the method that we implemented is the weight initialization. The 'feed through' neural network is initialized such that, on the first iteration, the current input, u(k), to the neural network is the output. The feed-through network is a fully-connected, feed-forward neural network with a single path that is linear and that initially has weights of 1.0. All weights from the inputs to the nodes of the first hidden layer are initially zero, except the weight going from the current input, u(k), to the linear node on the hidden layer, which is initialized to 1.0. The other weights past the first layer are initially random, except for the weight from the linear node on the hidden layer to the output, which is also initialized to 1.0, as seen in Figure 3.4. If the neural network structure has more than one hidden layer, then each hidden layer has a linear node and an initial weight of 1.0, connecting it with the other adjacent linear nodes.

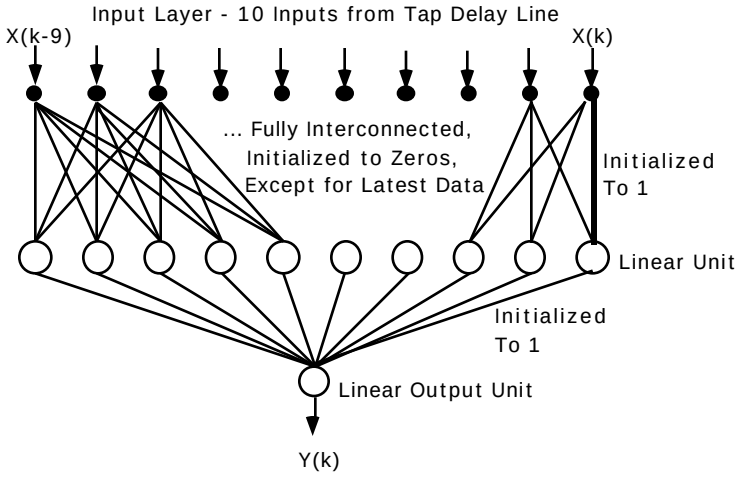

Figure 3.4 - Adaptive neural network filter structure.

A comparison is made between the feed-through neural network and a randomly-initialized neural network. For Case 1, the unstable plant diverged before the neural network had an opportunity to converge. This is not surprising, because randomly initializing the weights is essentially adding a random gain in a closed loop controller. There is no guarantee that the resulting closed-loop system will be stable. This is one of the reasons adaptive inverse control is seldom used on an unstable plant. For Case 1, the stable plant would converge when the learning rate is decreased because the neural network adds a unknown gain to the closed loop. For Case 2, the unstable plant is first stabilized by the estimator/regulator so the neural network can freely run at a higher learning rate. After approximately 300,000 iterations, the neural network converges to where there is marginal improvement in performance. It is evident that given several reinitializations, the neural network could converge to a spot that will give better performance than the case without a neural network, but that situation is not found during this set of experiments.

3.2 Results

For Case 1 and 2, step responses were plotted for both the stable and the unstable plants. Each of the following figures has the desired step response (--), the step response without the neural network (..), and the step response with the converged neural network (-). In each case, the step responses of the plant are greatly improved by the neural networks.

For Case 1, the stable plant shows remarkable improvement in steady state error and an increase in damping, as seen in Figure 3.5. The Mean Squared Error (MSE), as seen in Equation 3.1, for the step response is reduced by 99%. The unstable plant also shows improvement in steady state error and an increase in damping, as seen in Figure 3.6. The MSE for the unstable plant is reduced by 84%.

$$MSE = \sum_{k} (y_d(k) - y(k))^2 \qquad (3.1)$$

With y is the output of the plant;

y is the output of the reference model;

k is the iterative step;

MSE is the mean squared error.

For Case 2, the stable plant shows a marginal improvement, as seen in Figure 3.7. The MSE is reduced for the stable plant in Case 2 by 90%. The unstable plant shows no real improvement in performance, as seen in Figure 3.8. The MSE for the unstable plant in Case 2 is reduced by 53%.

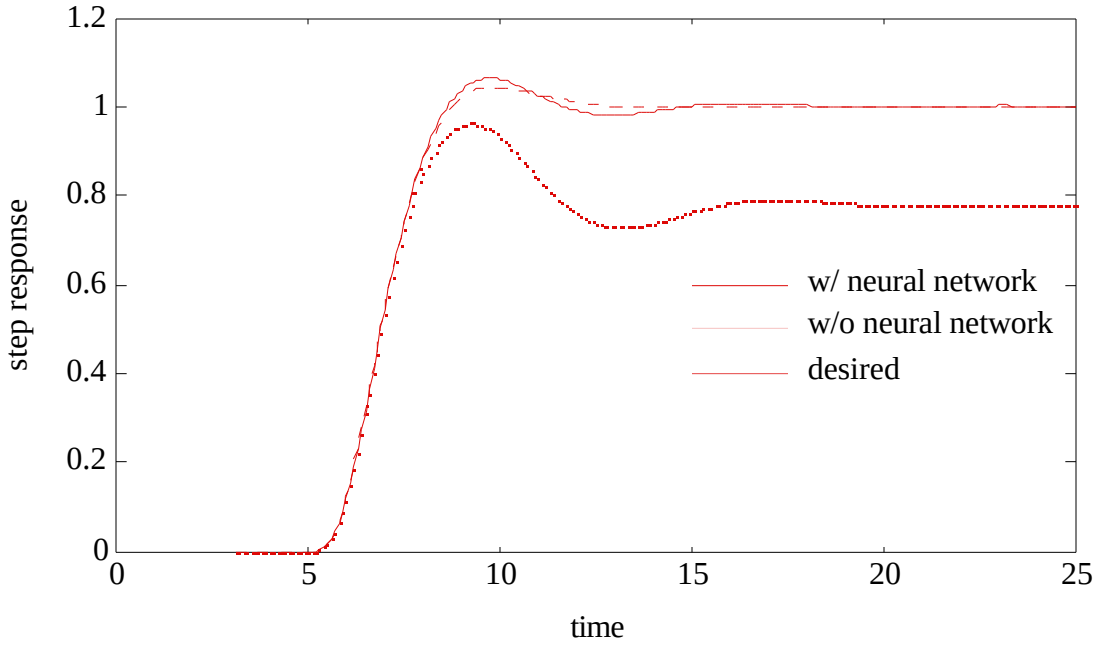

Figure 3.5 Case 1 Stable Plant Feed-Through Neural Network

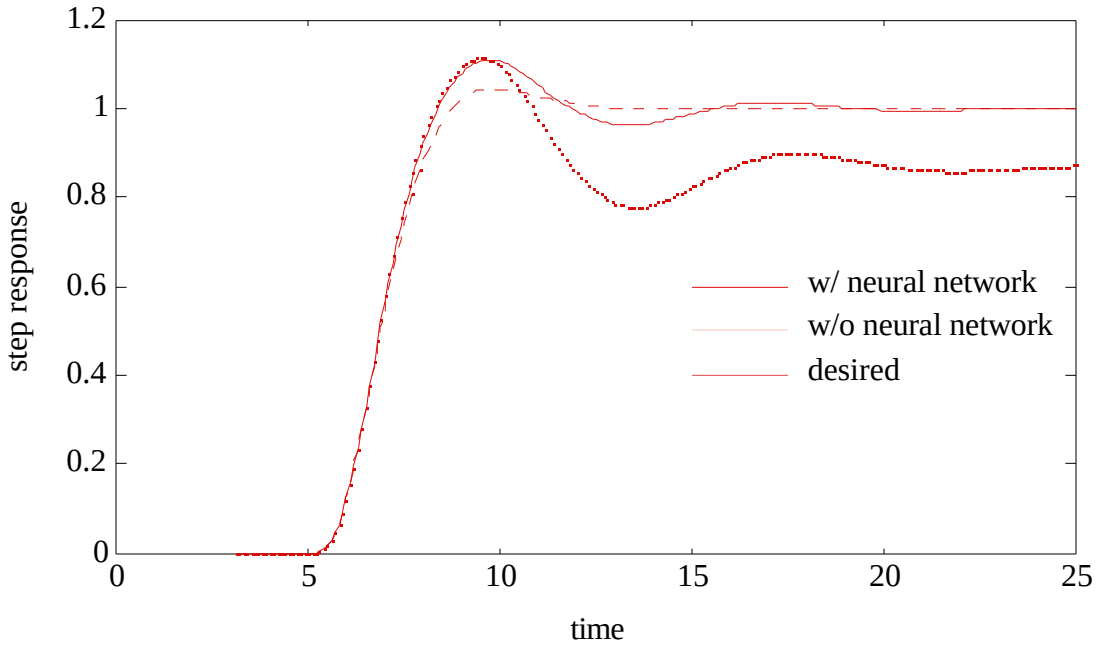

Figure 3.6 Case 1 Unstable Plant with Feed-Through Neural Network

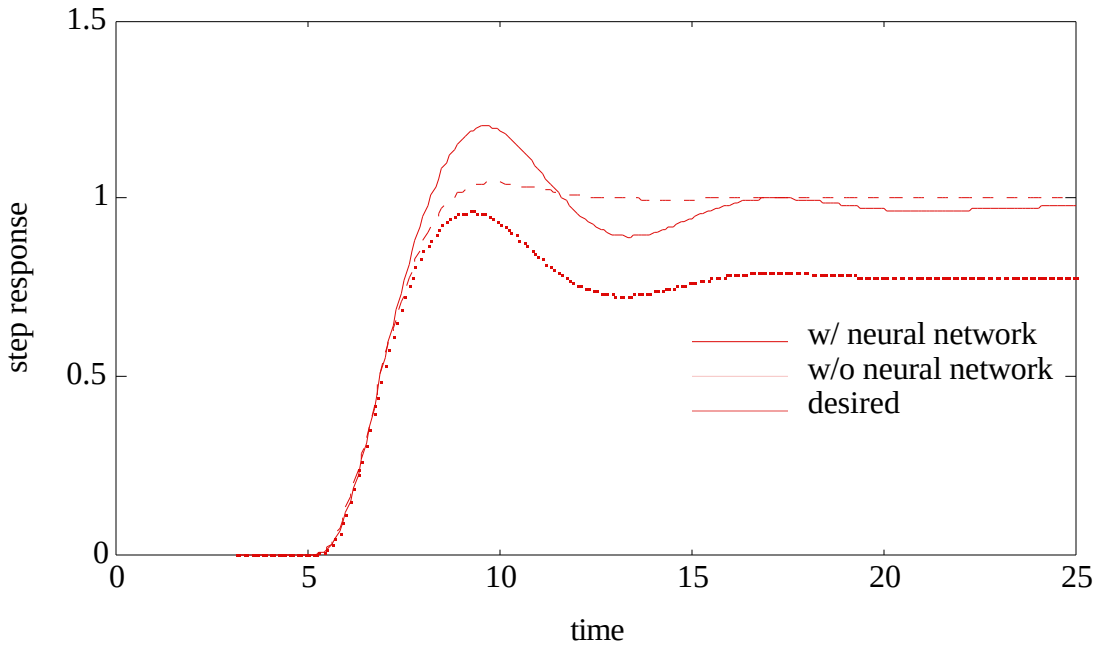

Figure 3.7 Case 2 Stable Plant with Feed-Through Neural Network

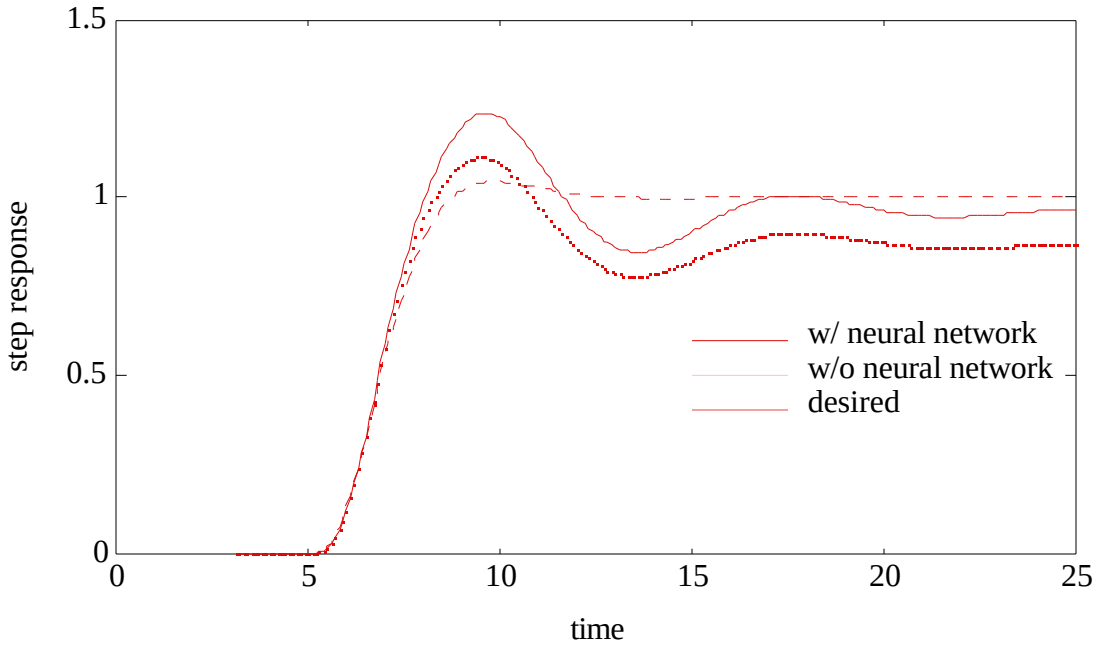

Figure 3.8 Case 2 Unstable Plant with Feed-Through Neural Network

The feed-through neural network improved the performance of the closed-loop system with no loss in performance during the convergence process. These results are what was expected of the feed-through neural network. The systems were initialized and converged a single time in order to achieve the performance seen above. The performance of the feed-through neural network inside the closed-loop is greater than the feed-through neural network outside the closed-loop. This result was also expected. They can be contrasted with the conventional method of initializing neural networks, which is to randomly initialize the neural network.

For Cases 1 and 2, randomly-initialized weights on the neural network were also tried. The desired step response (--), step response without a neural network (-.),step response with a converged neural network (-), and step response with an unconverged neural network (..) are plotted for the stable and unstable plants, as seen in Figures 3.9 through 3.11. The neural networks for each of the cases and plants is randomly initialized twenty times, and the results

shown are the best for each. For Case 1, a randomly initialized set of weights are never found to be stable for the unstable plant. For the stable plant, the neural network results in a performance that has much less damping than the original controller for the best set of randomly initialized weights. However, the steady state error has been eliminated, as seen in Figure 3.9. The MSE with the unconverged neural network is initially 2500% greater than without a neural network, but the MSE is reduced by 46% of the results without a neural network.

For Case 2, the neural network is randomly initialized with stable plant. The initial weights provide a better controller than without the neural network, and the converged weights improve performance even more, as seen in Figure 3.10. The initial MSE is reduced by 69%, and the converged controller reduces the MSE by 73%. The improvement over the unconverged weights is slight. The unstable plant also shows slight improvement with the randomly initialized weights, as seen in Figure 3.11. The MSE initially is greater by 32%, but after the convergence process, the MSE is reduced by 42% over the results without a neural network. A negative result of randomly initializing weights is that, when the input to the neural network is zero, the output of the neural network is not zero. This would result in a drift in the plant because of the constant output by the neural network to the plant. For Case 2, there is no advantage to using a neural network.

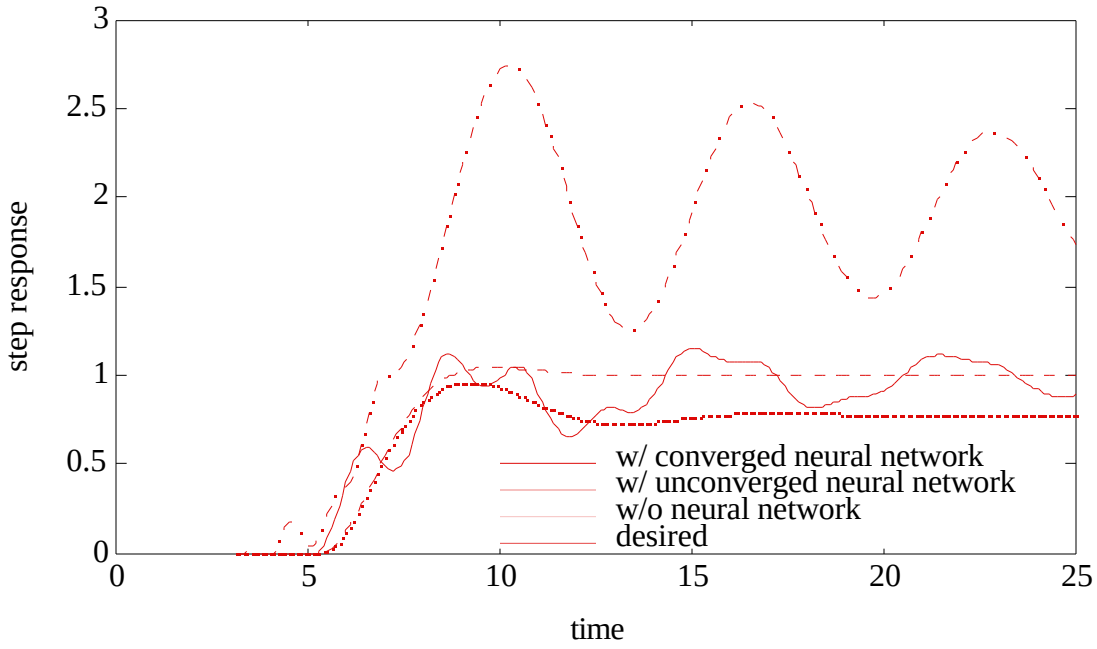

Figure 3.9 Case 1 Stable Plant with Randomly Initialized Neural Network

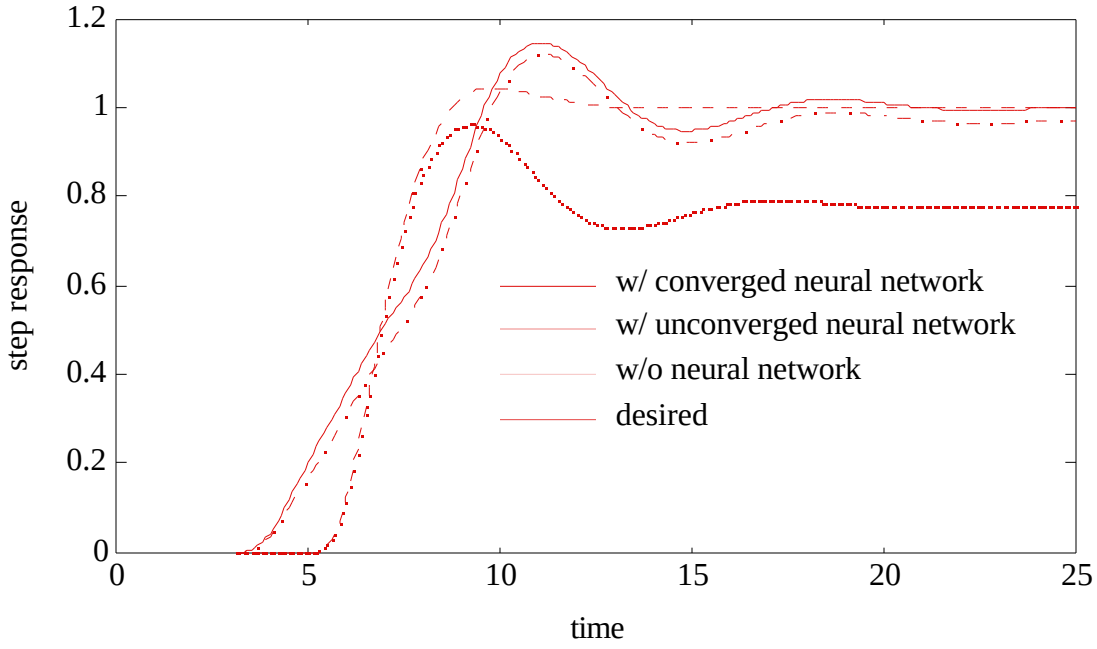

Figure 3.10 Case 2 Stable Plant with Randomly Initialized Neural Network

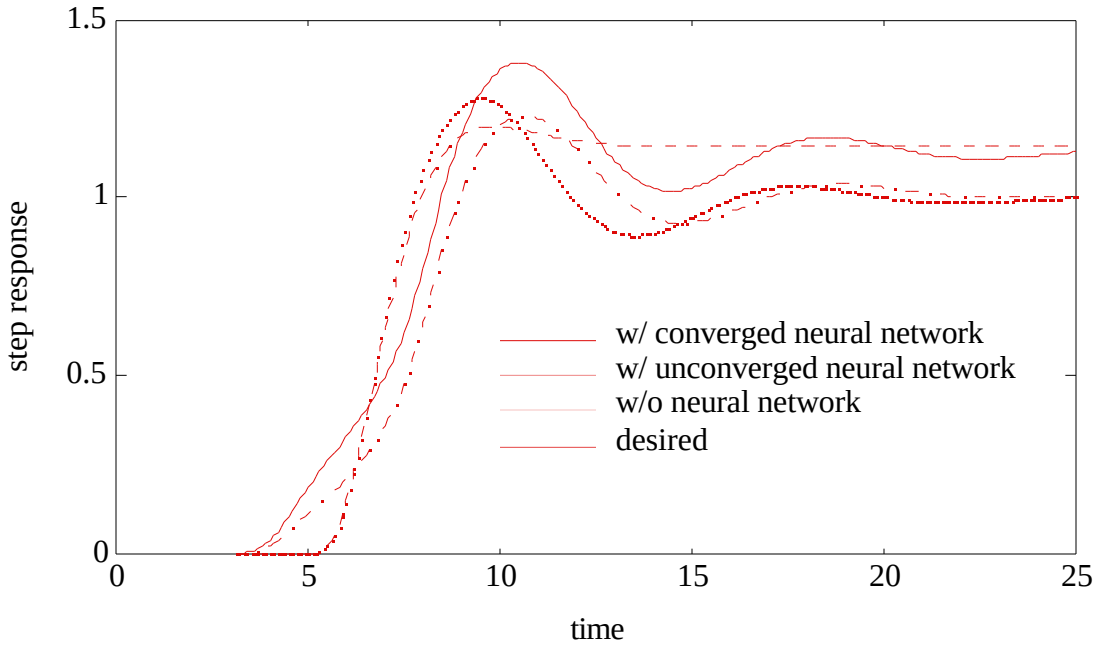

Figure 3.11 Case 2 Unstable Plant with Randomly Initialized Neural Network

3.3 Summary

There are several conclusions that can be drawn from the work done in this chapter. Our broadest goal is to demonstrate the usefulness of combining state space control and adaptive control using neural networks. We demonstrate that performance ranges from slightly to significantly better than that which is possible using a regulator and estimator designed for a linearized model of the plant; this is possible by augmenting the controller with an adaptive neural network filter. By initializing the system with the network as a feed-through where the tap delay signal for the latest value of the feed back is fed directly through linear units with unity gain, and starting the adaptation from that point to decrease the mean squared error between the actual and desired system response, the system performance in closed loop is the same as the original closed-loop system that ignored the plant's nonlinearities. The degradation in performance is due entirely to the nonlinearity of the plant. If the plant had been linear, the closed-loop performance of the system would have been identical to

the reference model. We are able to improve the closed-loop performance with no performance degradation during training.

This can be seen particularly well in Case 1; the control structure with the adaptive filter inside the loop, but the desired response is the overall desired linear closed-loop system response. Ignoring the nonlinearities and using regulator estimator control produces close to 20% error in the steady state response and less damping than the desired closed loop linear response. By adding the neural network, the steady state error is eliminated, and the damping is increased. It shows that an improvement can be made by combining the two methods of control. A summary of the MSE of the simulations run can be seen in Table 3.1 where F-T stands for feed-through neural network and R-I stands for randomly-initialized neural network.

Table 3.1 Summary of Results

Case	N/N	Plant	Initial MSE	Final MSE
1	F-T	Stable	0.0327461	0.0002855
1	F-T	Unstable	0.0115941	0.0018523
1	R-I	Stable	0.8545753	0.0177570
1	R-I	Unstable	Unstable	No Results
2	F-T	Stable	0.0327461	0.0033993
2	F-T	Unstable	0.0115941	0.0054399
2	R-I	Stable	0.0010124	0.0089960
2	R-I	Unstable	0.0153370	0.0067263

The combination of traditional control methods and neural networks is useful in controlling nonlinear, stable, and unstable, plants. The traditional state-space control methods, based on a linearized plant model, allow engineers to apply their analytical knowledge to the problem even though they might not know the exact model of the plant. The neural network adaptive control allows a controller to evolve, thus adapting to the plant, whatever it might be. By short circuiting the neural network, the traditional controller can control the plant with

the best performance possible without the use of adaptive control, the neural network can then slowly converge on a better solution. This method produces no degradation in the closed-loop performance of the system at any point during convergence. The algorithm is allowed to continuously iterate towards a better solution as the parameters of the plant, or the nonlinearities, change.

The update algorithm used on this phase of the research is back propagation, which was developed for the open-loop configuration. The weights are assumed to be independent of the input into the neural network. This is not true for the closed-loop configuration. A new update algorithm must be developed for the closed-loop configuration because the standard back propagation is not really valid inside the closed loop and does not take into account a priori information. Unfortunately, there has not been the development of an update algorithm for the linear FIR filter. In Chapter 4, the development of this algorithm should proceed the development of the new update algorithm for neural networks.