#6523

3671

ADAPTIVE CONTROL OF HVAC PROCESSES USING PREDICTIVE NEURAL NETWORKS

P.S. Curtiss, Ph.D.

J.F. Kreider, Ph.D., P.E.
Member ASHRAE

M.J. Brandemuehl, Ph.D., P.E.
Member ASHRAE

ABSTRACT

In practically all HVAC systems, the maintenance of process setpoints relies on feedback loop control. Proportional, integral, and derivative (PID) algorithms are often used in these controllers. The optimization of such loops has been the subject of many studies, most of which use techniques to first model the dynamic behavior of the process and then to use the model to arrive at a series of control constants that would effect a first-order response to changes in setpoint. This paper demonstrates how artificial neural networks (ANNs) can be used for the same purpose and compares the performance of two types of ANN controllers to a conventional PID controller.

INTRODUCTION

The advent of the personal computer has had a profound impact on the way in which building systems are controlled. Direct digital controllers and energy management systems rely on microprocessors to implement increasingly complex control strategies that can take advantage of the speed and precision of electronic controls. It follows, therefore, that stand-alone electronic controllers are used to autotune control constants or adapt to changing process conditions.

The subject of optimization of feedback loops is certainly not new, and much study has been devoted to the modeling techniques used to identify dynamic system response. Radke and Isermann (1987) and MacArthur et al. (1989) show how autoregressive moving average (ARMA) algorithms are used to arrive at parameters for linear discrete-time process models. Many permutations of these models are possible; for example, Zaheer-uddin (1990) combines an ARMA method with a energy balance on a zone. The resulting parameters can then be used to find the theoretical optimum control values, either through pole-zero cancellation or root placement in the characteristic equation, as shown by Wallenborg (1991). The computational requirement of many of these techniques precludes the use of pneumatic controllers and has instead given rise to computer-based analysis of the processes.

Meanwhile, in another realm of the computer age, the subject of artificial intelligence has been receiving considerable attention. In particular, the use of neural networks is proliferating with remarkable speed. These networks are an attempt to recreate simple biological networks by joining together "cells" (or *nodes*) in a cascaded fashion, all richly connected to each other. When a given set of cells (the inputs) are stimulated, the signals are passed through the network from node to node and finally exit the network through another set of specified nodes (the outputs). Any given node accepts input from a number of other nodes, then outputs a signal based on the sum of all the inputs. Each node is connected to other nodes through a series of weighting factors by which its output signals can be amplified or attenuated. The trick to "training" a network is to find weights such that a given set of inputs causes the network to yield the desired output. One such learning algorithm is called back-propagation, whereby the weights are adjusted to reduce the error between the actual and desired outputs of the network. Detailed descriptions of different network configurations and training techniques are given in Rumelhart and McClelland (1986) and Wasserman (1989) among many others.

Neural networks behave very much like nonlinear regressions with their ability to associate specific input and output patterns. When used as predictors with an HVAC process, for example, ANNs can provide the basis for predictive controllers. It is also possible to use a network teaching formalism, in parallel with a predictor, to actually control a system. Much of the literature on ANN controllers relies on the use of computer models to simulate a process. Nguyen and Widrow (1989) have successfully demonstrated the use of the controller/emulator method to position a trailer truck, while Anderson (1989) has used a similar technique to control inverted pendulums (e.g., balancing a broom on the palm of your hand). Helferty et al. (1989) have shown how an ANN controller would be used to maintain the balance of a one-legged hopping robot.

This paper discusses the results from a computer simulation that used ANNs for predictive control of a hot

Peter S. Curtiss is a postdoctoral fellow at the University of Colorado, Boulder, and at the Ecole des Mines de Paris. **Jan F. Kreider** is a professor and director and **Michael J. Brandemuehl** is an assistant professor in the Joint Center for Energy Management, CEAE Department, University of Colorado, Boulder.

water coil used to warm an airstream. The coil model itself is a neural network that has been trained on actual data and mimics the nonlinearities of the coil well.

PHYSICAL SYSTEM USED FOR ANALYSIS

The experiments were conducted at the Joint Center for Energy Management HVAC Systems and Controls laboratory. This facility is a full-scale representation of a commercial HVAC system with four separate zones. Loads are introduced into each zone using a series of hot and chilled-water coils and steam injectors. The control process under examination is a hot water coil in one of the zones used to generate cooling loads. A schematic of the system is shown in Figure 1, where the load is calculated from

$$\text{LOAD} = \dot{M}_{air}\, C_{p,air}\, \Delta T_{air} \tag{1}$$

where $\dot{M}_{air}$ is the mass flow rate of the air, $C_{p,air}$ is the specific heat of air, and ΔT_{air} is the temperature rise of the airstream. This load is used as the feedback signal to control the valve so as to maintain a specific load setpoint. The system is truly dynamic in the sense that the airflow

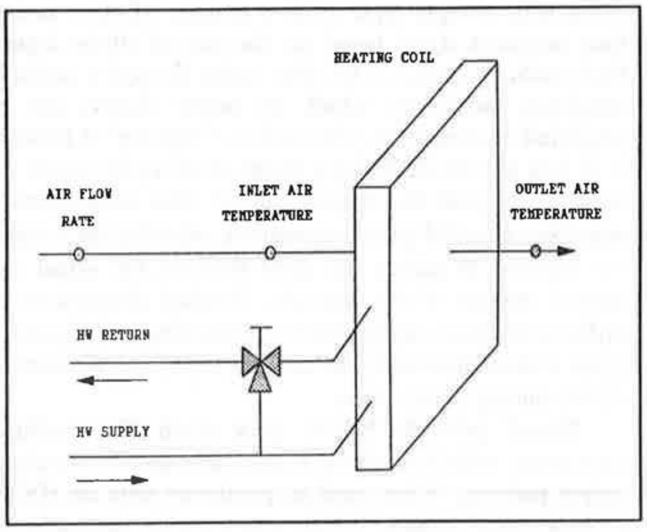

Figure 1 *Schematic of hot water coil used in experiments.*

TABLE 1
Sensor Input Ranges for Heating Coil

process input	data history	range
valve actuator control	20 time steps	0 - 100% full scale
cooling load	20 time steps	0 - 100 MBTUH
entering air temperature	10 time steps	40 - 90°F
air flow rate	10 time steps	0 - 4000 CFM
hot water temperature	10 time steps	50 - 150°F
hot water branch flow rate	10 time steps	0 - 20 GPM

rate and inlet air temperatures can vary considerably during normal operation. Additional inputs to the process include the hot water temperature and flow rate.

Normal PID control of this process has not been very successful, since the controller, feedback, and auxiliary inputs vary across a wide range of values. Different airflows or water temperatures will affect the coil time constant and hence the "speed" at which this process needs to be controlled. Furthermore, the VAV box that controls the temperature of this zone will respond to any changes in the load by varying the airflow rate and/or the temperature of the air supplied to the zone. It has been observed that for a given set of PID constants the process will be under adequate control under certain conditions and very much out of control under other conditions.

MODELING OF COIL USING A NEURAL NETWORK

The remainder of this paper will discuss how a neural network controller can be used to properly invoke several process setpoint changes. A computer simulation will model the process to ensure that the dynamic response is consistent for all the cases. The computer model is itself a neural network that has been trained using data from a wide range of coil operating conditions. The network was configured to accept a time history of each input. A list of these inputs and their effective range used in the normalization of the input is given in Table 1.

Several different network architectures were tested for their ability to correctly model this process. It was discovered that a standard, feedforward network using the delta rule back-propagation learning algorithm (Wasserman 1989; Rumelhart and McClelland 1986) provided a fast and accurate model. In this network, the input layer consisted of a time history of the valve position and the load, as well as four other channels. The output of the network was a prediction of the load at the next time step. A schematic of this network is given in Figure 2.

To check the accuracy of the model, the actual process was put through a series of setpoint step changes similar to

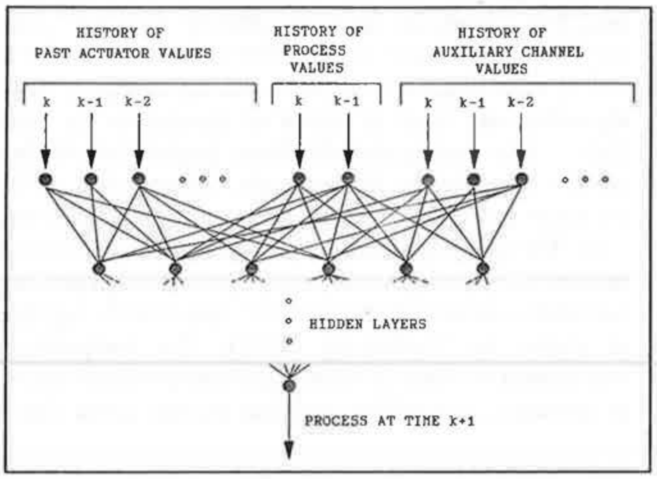
HISTORY OF PAST ACTUATOR VALUES
HISTORY OF PROCESS VALUES
HISTORY OF AUXILIARY CHANNEL VALUES
k
k-1
k-2
k
k-1
k
k-1
k-2
HIDDEN LAYERS
PROCESS AT TIME k+1

Figure 2 *Architecture of ANN predictor.*

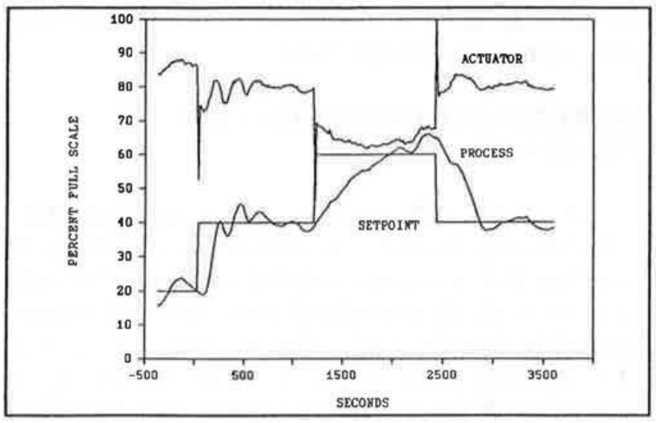

Figure 3 *Response of heating coil to changes in setpoint.*

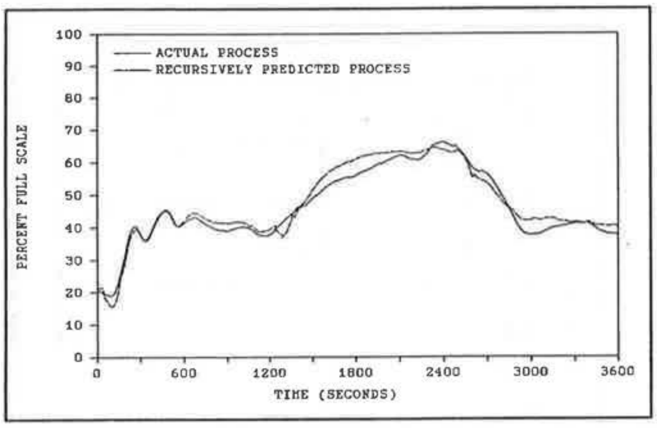

Figure 4 *Comparison of ANN prediction and actual data using recursive, highly connected network.*

those used in the controller comparisons. The process (i.e., the coil load) is initially set to 20 MBtu/h and allowed to reach stability. After 20 minutes, the setpoint is changed to 40 MBtu/h and remains there for 20 minutes, at which time the setpoint is increased to 60 MBtu/h. Finally, after 20 more minutes, the setpoint is reduced to 40 MBtu/h. The PID constants used for the actual control of this process are a gain of $K_p = 0.5$, a reset time $T_r = 200$ seconds, and a derivative time $T_d = 50$ seconds, with a scan rate of approximately 12.3 seconds. The actual response is shown in Figure 3. These control constants were chosen by trial and error as the best compromise for decent control across the range of operating conditions.

Once trained, the network was tested to see if it could match these results. The network was initialized with the appropriate values, then allowed to predict the process output at each time step. The prediction process was recursive in the sense that the output of the network was used as the input for the load at the next time step. This gave the results shown in Figure 4 and demonstrated the ability of a network to accurately predict such a process in real time. Notice, however, the depressions in the predicted process values at 100 and 1300 seconds and the slight increase at 2500 seconds. These sudden changes are related to the actuator motion at these times, which, in turn, is responding to the setpoint changes (refer to Figure 3).

These mild perturbations occur because the network is attempting to model a reverse acting process—as the actuator position increases, the process will decrease. In general, the network performs well at mapping this inverse relationship, but when there is rapid actuator motion, the network briefly allows the process to follow this motion. At first glance this would not appear to be a serious problem, but as it turns out, for use in predictive control this can be catastrophic.

In order to understand why even such a slight degree of inaccuracy is unacceptable, we need to examine how the predictive network operates. At any given time, the ANN controller will predict how the process will behave over some finite number of time steps into the future. Back propagation is then used to adjust the controller output so as to minimize the process error (i.e., the difference between the setpoint and the process output value) at this future time. If the time window is long enough so that these brief mistakes by the predicting network are passed by, then there is no problem. In the case of a shorter time window, however, any mistakes in the prediction will be propagated back through the controller and can drive the actuator into a clamped output at the wrong side. Figure 5 shows the open-loop response of the same network used to generate the results in Figure 4. Here the actuator is initially set at 85% and the system is allowed to reach stability. When the actuator position is decreased to 75%, we see the expected rise in the process value, and when the actuator is increased to 90%, the process decreases. But we can also see the short-term transient network error as it demonstrates a direct action response before correcting itself. For the initial decrease of actuator position, the network yields a low prediction for 12 time steps (approximately 150 seconds). If the controller prediction window was less than two and a half minutes into the future, then any final prediction error might tell the controller that the process is direct acting. This would lead to completely inaccurate control.

In order to compensate for these problems, another richly connected network was trained but this time with

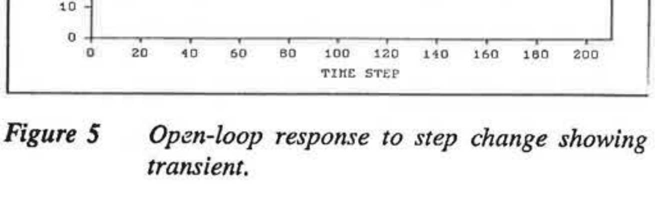

Figure 5 *Open-loop response to step change showing transient.*

selectively constrained weights. All the weights between the input nodes corresponding to the valve position and the first hidden layer were restricted to negative numbers. Thus the process model is forced to emulate a reverse-acting process. In addition, all the weights between the input nodes corresponding to the history of the process outputs and the nodes in the first hidden layer were forced to be non-negative. This approach was ultimately successful in finding a set of weights that produced a realistic process model.

Now that a model has been found that can emulate the heating coil with sufficient accuracy, we can compare different methods of controlling the process. For these comparisons, the model is used to simulate a one-hour test during which the feedback loop undergoes three different setpoint changes. These simulations closely follow the data set used for model comparison in the previous section. In all cases the model is allowed to reach steady state before any perturbations are introduced.

PERFORMANCE OF CONVENTIONAL PID CONTROLLER

A standard PID controller was initially simulated to establish some basis for comparison of the methods. The reset and derivative times were held constant at 250 and 50 seconds, while the proportional term was varied. Figures 6 through 8 show the results when the gain is 0.1, 0.5, and 1.0, respectively. All of these systems demonstrate stability, although at the lowest gain the response is too sluggish and at the higher two gains there is considerable overshoot at

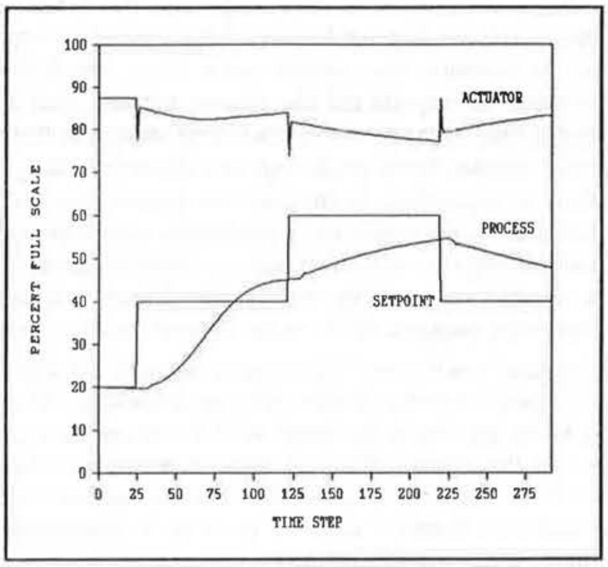

Figure 6 PID controller with gain of 0.1.

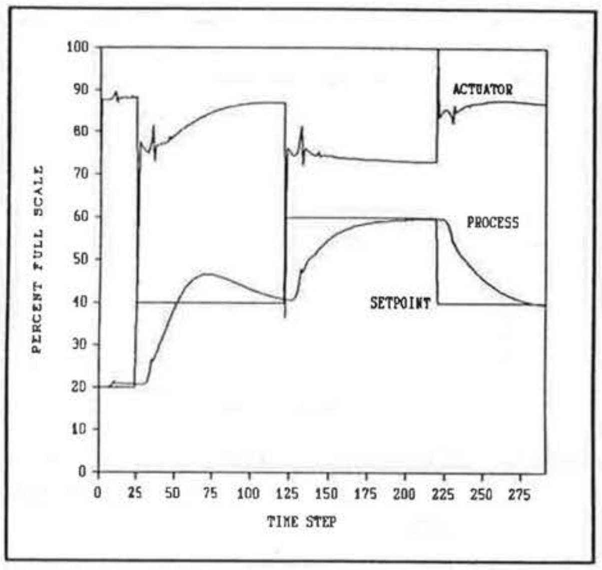

Figure 7 PID controller with gain of 0.5.

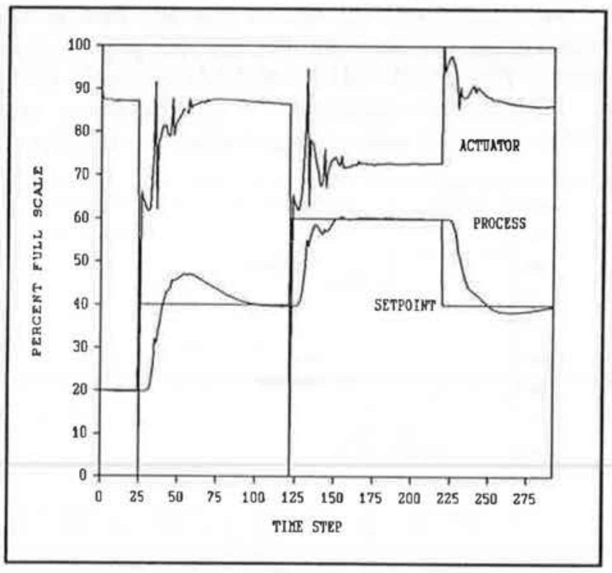

Figure 8 PID controller with gain of 1.0.

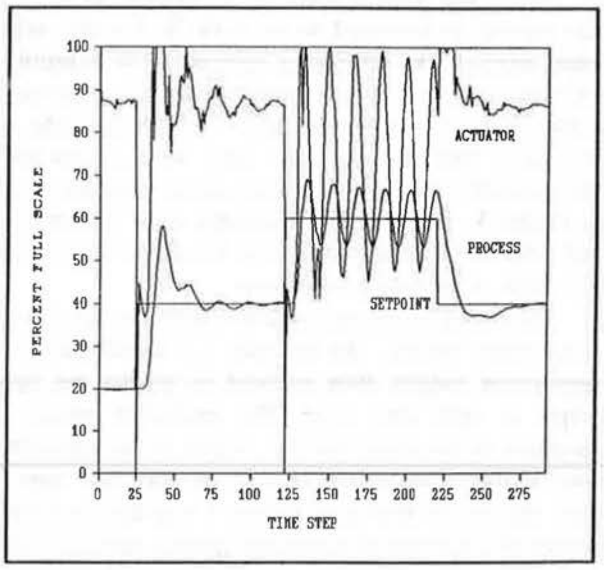

Figure 9 PID controller with gain of 2.0.

the initial setpoint change. Figure 9 shows the same PID model as used above using a gain of 2.0. Here we see some of the nonlinear behavior of the coil—at low setpoints the feedback loop is stable and converging, while at the higher setpoint the process exhibits critically damped oscillatory behavior. At gains higher than this, the system becomes oscillatory unstable.

Table 2 shows the RMS error taken over each of these tests. This value is calculated from

$$E_{\mathrm{RMS}} = \sqrt{\frac{\sum_{t=1}^{N}(\mathrm{SETPOINT}_t - \mathrm{PROCESS}_t)^2}{N}} \tag{2}$$

where N is the total number of time steps per run. The RMS error is used as a statistical basis of comparison with the ANN controllers. Notice that due to the inherent lag time in the system, it is impossible for the RMS error to ever equal zero.

TABLE 2
Error from PID Control Tests

Proportional Gain	Normalized RMS Error
0.1	10.43
0.5	7.68
1.0	6.63
2.0	7.08

ARCHITECTURE OF PREDICTIVE CONTROLLER

The basic premise behind predictive control is that the controller guesses at an appropriate output then "looks ahead" to see how this output will affect the process in the future. At each time step, the controller is readjusted and the prediction is repeated to see where the process will be at some future point. The output value that puts the process closest to the setpoint is chosen and sent to the actuator. This technique has been successfully employed by Kuperstein and Rubinstein (1989) and Liu et al. (1989) for their experiments in robot hand control.

It would be computationally intensive and time consuming for the controller to test every possible output. Indeed, since there can be a wide variety of inputs into the system and since the system may be experiencing non-steady-state conditions, the prediction of the error over the full range of actuator motion could conceivably take longer than the sampling period itself, making the controller useless for real-time applications. It is desirable, therefore, to find some way that the proper actuator position could be determined rapidly. The back-propagation algorithm developed for neural networks is a likely candidate for such a speed search, whereby the entire prediction and error

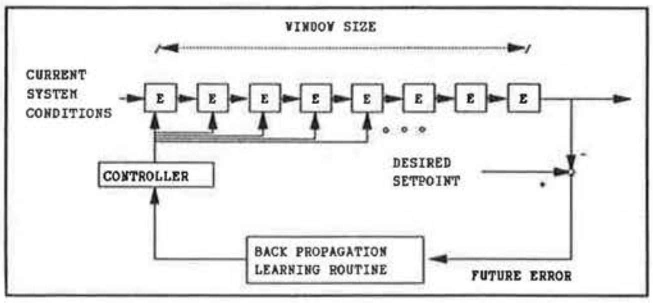

Figure 10 ANN-based predictive controller.

calculation process is treated as an activation function in a simple network. The proposed scheme thus involves a dual ANN controller: one to perform the actual prediction of the process and the other to find the correct controller output based upon the error derived by the first network. Lan (1989), Chen (1990), and Psaltis et al. (1988) have all shown that controllers with this kind of architecture function remarkably well. A typical schematic of this controller is shown in Figure 10, where the boxes labeled E represent the emulator (predicting) network. After data are received by the controller at time $k = 0$, the emulator will predict the behavior of the process for n time steps, ending at some future time $k = nT_s$, where T_s is the sampling period. The controller then takes the process value at this future time, calculates the error, then back-propagates the error through the network to modify the controller output. The new output is sent, and the controller then waits for the next set of data at time $t = k + 1$, predicts the output at $t = (n + 1)T_s$, back-propagates the error, and so on. Note that the error is not sent to each controller module as shown in the diagram but rather is returned through the emulator networks. In this sense, the entire controller can be regarded as a large ANN with many hidden layers (two per time step), which uses standard back propagation to arrive at a desired weight corresponding to the controller motion.

In the truck backer-upper problem (Nguyen and Widrow 1990), the final position of the truck was calculated for each change in the steering signal. The final position was dictated by simulating the truck movement until it hit something, for example, a loading dock. In this problem, there were well-defined bounds placed on how far ahead in the future the emulator needed to predict. In our case, however, there are no definite limits. The emulator network needs to have some constraint on how far ahead it looks, preferably some duration longer than the time constant of the entire feedback loop.

The key to this method is that the controller be able to accurately predict *on its own*, starting from some initial conditions, i.e., the network behaves like a Hopfield network where the output is fed directly back into the input over the duration of the time window used in the prediction. This is an important feature, since if the network does not predict the process accurately, there is no guarantee that the process will reach a stable control.

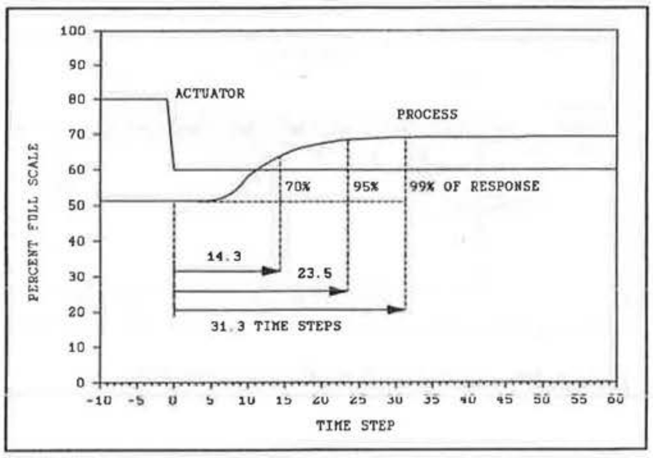

Figure 11 *Open-loop response to step change in actuator position.*

It should be possible for the ANN controller to incorporate an internal algorithm that would allow it to determine the optimal window size without user intervention. Figure 11 shows an open-loop step response of the process modeled above. The 70%, 90%, and 95% rise times are shown along with the corresponding number of time steps required to reach these values. Also clearly shown is the dead time of the process. It is left for future research to examine the correct rise time to use in order to ensure a stable and accurate response of the process to a step change. It can be seen, however, that the 70% rise time (i.e., the ln(2) time constant) requires 14.3 time steps. The neural network controllers discussed below will all be tested using a 15-time-step window.

PERFORMANCE OF LOOK-AHEAD NETWORK CONTROLLER

Any neural-network-based controller will be useful only if it can perform better than the PID controller shown above. The first kind of ANN controller involves the predictive technique discussed in the previous section. Here the network was allowed to predict where the process would be at some time in the future, and any error was back-propagated through the model in order to change the actuator output. This future predicting network is denoted by the acronym FANN. The learning rate of the controller was varied between 0.5 and 10 while the window size was kept constant at 15 time steps (roughly three minutes). Figures 12 through 15 show the results.

Figure 13 gives the results for a learning rate of one. Here we see excellent control—minimal overshoot and quick response to the setpoint changes. As the learning rate increases, the network controller deteriorates into unusable control. Notice that the feedback loop never becomes *unstable* per se, but rather that the actuator begins to oscillate excessively. This can be explained by the predictive quality of the controller: any actuator motion will cause the process to change at some point in the future. The predictor finds this value, calculates the future error, then back-propagates that error through the network, ultimately changing the actuator position by an amount proportional to the learning rate. Any mistake by the controller in positioning the actuator will cause additional future error. As the learning rate grows much larger, it becomes almost impossible for the network to find the correct value; rather, the network keeps overshooting and correcting.

The RMS error for the FANN network are given in Table 3. Notice that although the FANN controller appears to be better than the PID controller in the graphs, statistically speaking the PID controller with a gain of 1.0 is still slightly better than any of the FANN controllers using the domain of learning rates and time windows tested.

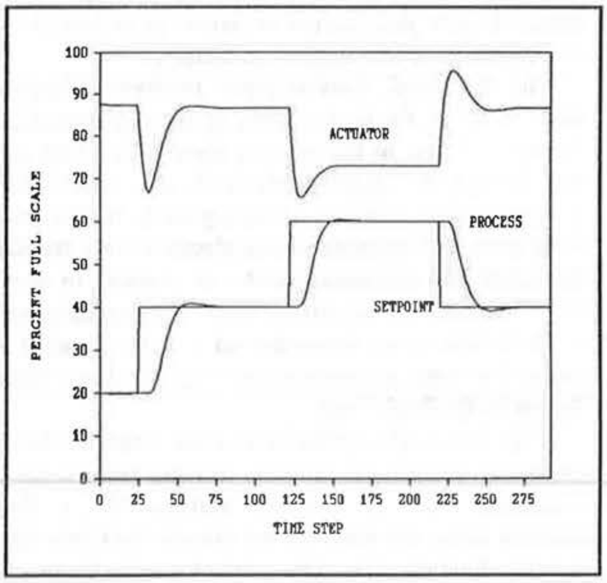

Figure 12 *FANN with learning rate of 0.5.*

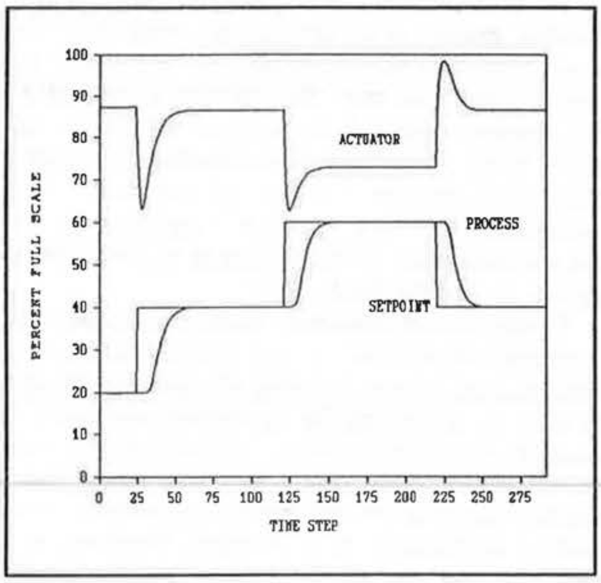

Figure 13 *FANN with learning rate of 1.*

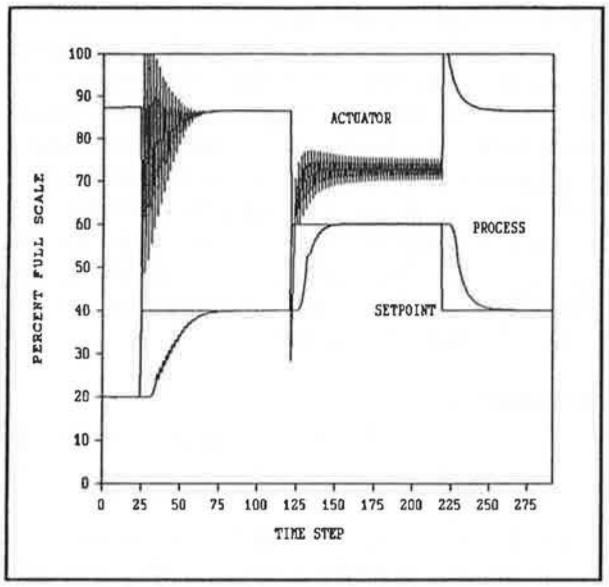

Figure 14 *FANN with learning rate of 5.*

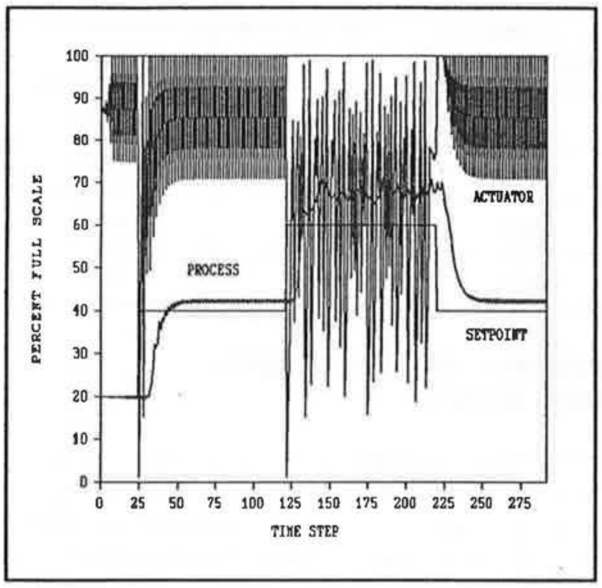

Figure 15 *FANN with learning rate of 10.*

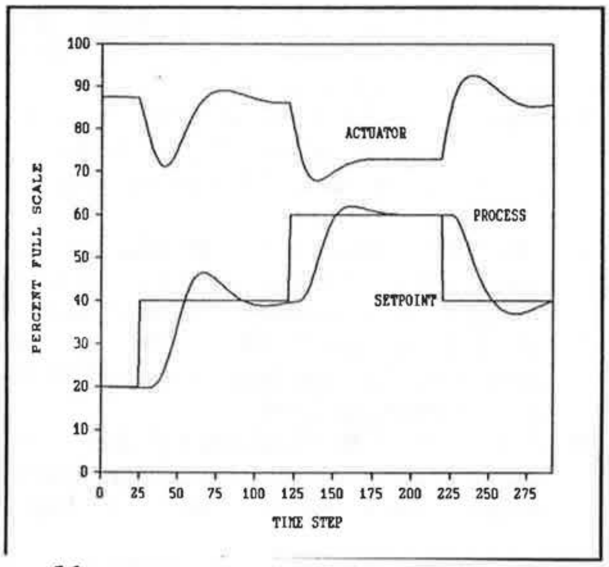

Figure 16 *IANN with learning rate of 0.5.*

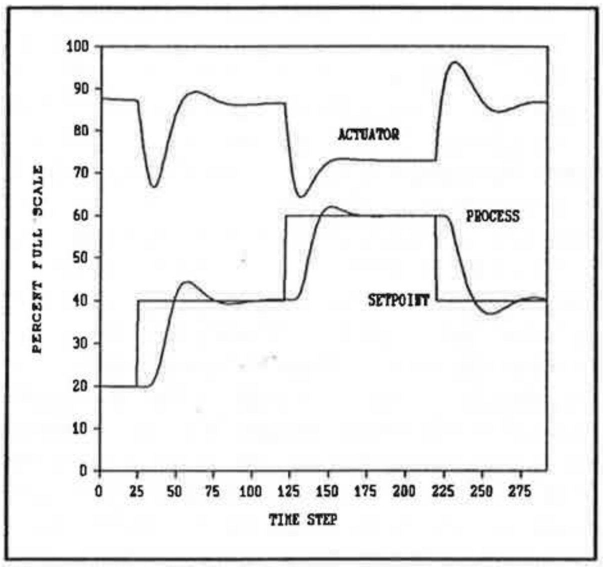

Figure 17 *IANN with learning rate of 1.*

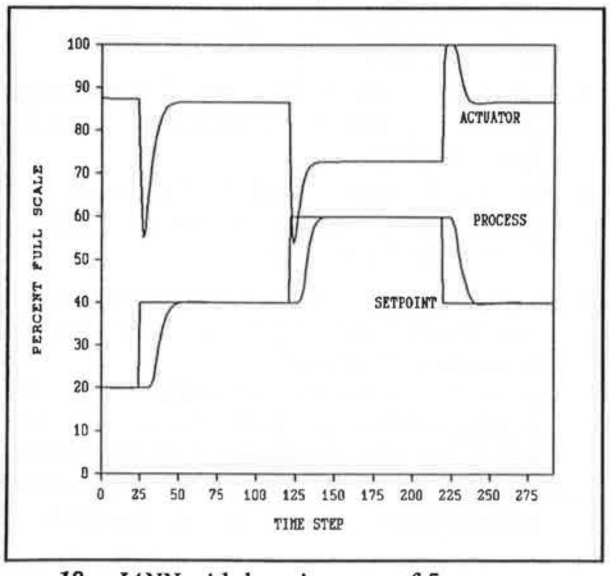

Figure 18 *IANN with learning rate of 5.*

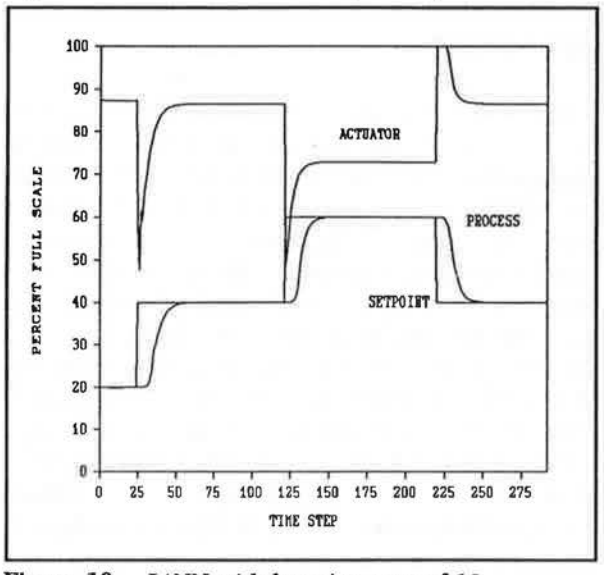

Figure 19 *IANN with learning rate of 10.*

Table 3 Normalized RMS error for ANN controllers

learning rate	FANN	IANN
0.5	7.29	8.49
1.0	6.90	7.61
5.0	6.81	6.55
10.0	8.09	6.43

PERFORMANCE OF INTEGRATING ERROR NETWORK

The FANN controller works by looking at only a single error in the process at some point in the future. Since the process is continuous and it is desirable to minimize overshoot, an integrating ANN (IANN) controller was considered. This controller works on the same principles as the FANN controller but takes the RMS process error over the window and uses that in the back propagation.

The effects of the learning rate were also studied for the IANN. Here, as with the FANN, the window is maintained at 15 time steps and the learning rate is varied between 0.5 and 10. In truth, however, it is not fair to compare the learning rates between the two methods. The reason is that in the FANN only one error was used for the comparison, while in the IANN a number of errors equal to the size of the window are used. Since any change of actuator position is directly proportional to this error, it is conceivable that an effective learning rate for a FANN network would not work for an IANN controller, and vice versa. Nonetheless, in our example, it does not appear to make much of a difference. Figures 16 through 19 show the effect of an increasing learning rate. Note that at a learning rate of 10, the IANN controller yields excellent control, whereas this same learning rate for the FANN controller made the process erratic. Results from these tests are given in Table 3.

CONCLUSIONS

The goal of this research was to see if neural networks could be used for adaptive and predictive control of a building systems process. The controller is adaptive in the sense that the output of the network used to model the process reflects the changing operating environment, and it is predictive since it examines the future effect of the current controller action. These features were demonstrated using computer simulations. As seen in Table 3, the IANN controller was able to minimize the process error better than the PID constants currently used to control this process. As this paper was being finalized, these experiments were repeated in a real-world, on-line application with success. The computer used as the ANN controller has the computational power of a slow AT-class machine, and yet was able to perform both FINN and IANN control within a 10-second scan rate.

The application of neural network controllers would seem to have promise for use in feedback loop processes. Both the future error and integrating ANN controllers have the potential to outperform the standard PID algorithm. Furthermore, the degrees of freedom have been reduced with regard to the conventional PID controller—only the window size and learning rate need to be specified as opposed to the three PID gains. It would also appear that the ANN controllers could find their own window size without additional user input.

REFERENCES

Anderson, C. 1989. Learning to control an inverted pendulum using neural networks. *IEEE Control Systems*, April, pp. 31-36.

Chen, F. 1990. Back-propagation neural networks for nonlinear self-tuning adaptive control. *IEEE Control Systems*, April, pp. 44-48.

Helferty, J.J., J.B. Collins, L.C. Wong, and M. Kam. 1989. A learning strategy for the control of a one-legged hopping robot. *Proceedings of the 1989 American Control Conference*: 896-901.

Kuperstein, M., and J. Rubinstein. 1989. Implementation of an adaptive neural network controller for sensory-motor coordination. *IEEE Control Systems*, April, pp. 25-30.

Lan, M. 1989. Adaptive control of unknown dynamical systems via neural network approach. *Proceedings of the 1989 American Control Conference*: 910-915.

Liu, H., T. Iderall, and G. Bekey. 1989. Neural network architecture for robot hand control. *IEEE Control Systems*, April, pp. 38-41.

MacArthur, J.W., E.W. Grald, and A.F. Konar. 1989. An effective approach for dynamically compensated adaptive control. *ASHRAE Transactions* 95(2): 415-423.

Nguyen, D.H., and B. Widrow. 1989. The truck backer-upper: An example of self learning in neural networks. *Proceedings of the International Joint Conference on Neural Networks* 2: 357-363.

Nguyen, D.H., and B. Widrow. 1990. Neural networks for self-learning control systems. *IEEE Control Systems*, April, pp. 18-23.

Psaltis, D., A. Sideris, and A. Yamamura. 1988. A multilayered neural network controller. *IEEE Control Systems*, April, pp. 17-21.

Radke, F., and R. Isermann. 1987. A parameter-adaptive PID-controller with stepwide parameter optimization. *Automatica* 23: 449-457.

Rumelhart, D.E., and J.L. McClelland. 1986. *Parallel distributed processing: Explorations in the microstructure of cognition*. Cambridge, MA: MIT Press.

Wallenborg, A.O. 1991. A new self-tuning controller for HVAC systems. *ASHRAE Transactions* 97(1): 19-25.

Wasserman, P.D. 1989. *Neural computing: Theory and practice*. New York: Van Nostrand Reinhold.

Zaheer-uddin, M. 1990. Combined energy balance and recursive least squares method for the identification of system parameters. *ASHRAE Transactions* 96(2): 239-244.