A Neural Network Lab Experiment

Robert Lynn Mueller
The Pennsylvania State University
New Kensington Campus

Abstract

Neural networks are becoming widely used in complex control problems. Many academic exercises approach neural network applications using only software simulations; however, simulations alone do not give students a full appreciation of the power and complexity of neural network-based controls. This paper describes a laboratory experiment that uses a temperature and airflow process simulator to demonstrate neural network control applications. The simulator is fundamentally a temperature controller in which large-scale changes in forced airflow produce significant changes in heat load.

The initial labs use PID control techniques to solve the temperature control problem and to demonstrate the problem that PID controllers have with large disturbances. The following labs address the same problem using a neural network control strategy. An actual neural network controller is built and used to perform the same temperature control as the classical PID system. Capabilities and drawbacks of neural network control are demonstrated.

Introduction

The classical PID feedback control system is shown in Figure 1. The setpoint (SP) is the system's input and the process variable (PV) is the output. The PID controller uses the error signal (SP minus the PV) to calculate a value for the manipulated variable (MV). In turn, the value of the MV then determines the value of the process variable. Unexpected changes to the process output can be caused by system disturbances, and it is the PID controller's job to adjust the MV to account for these changes.

Based on the characteristics of the process, the PID controller is tuned to achieve the desired system response. Unfortunately, when the disturbance input varies over a large range, the PID controller's ability to achieve the desired system response is greatly reduced. For these situations, a more advanced control scheme is necessary. One possible method to overcome large disturbance variations is a neural network (NN)-based control strategy.

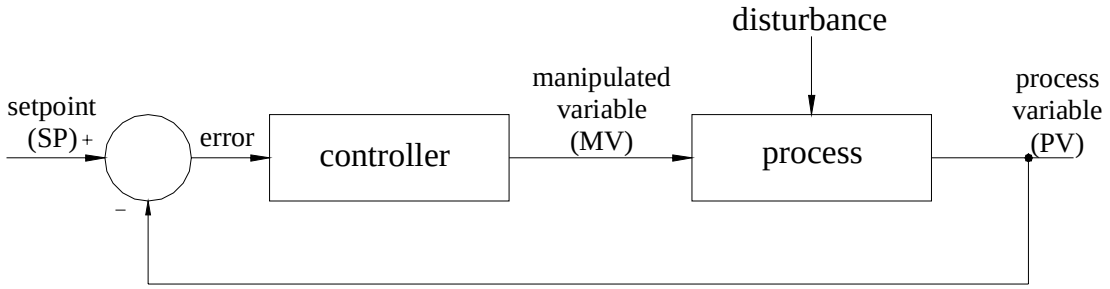


Figure 1. Classical Negative Feedback Control System

Neural Network Overview

A significant amount of literature exists regarding the operation of a NN. A detailed description of NN systems is given in [1]. A brief overview is given here to define terms.

A NN is used to predict one or more outcomes based on the NN's inputs. The basic element of a NN is the neuron as shown in Figure 2. The neuron has 1 or more inputs but produces only 1 output. The output is found by first calculating the weighted-sum of the inputs and then "squashing" that sum according to some non-linear squashing function. One popular squashing function is the sigmoid function:

$$F(x) = \frac{1}{1 + \exp(-\text{sum})}$$

The weights that are used in the sum are "learned" by the NN based on training data that it acquires both during an initial training period and after the NN goes on-line.

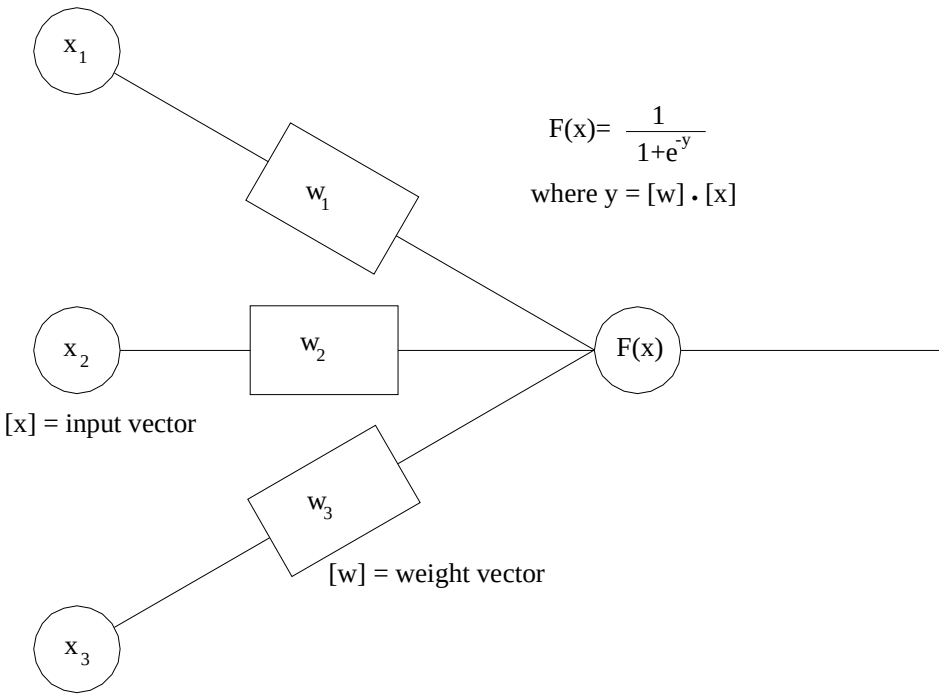


Figure 2. Typical Individual Neuron

A neural network (NN) is formed by connecting the neurons together. A typical configuration is shown in Figure 3. A NN has an input layer, one or more hidden layers, and an output layer. The number of neurons in the input layer corresponds to the number of NN inputs and the number of neurons in the output corresponds to the number of outputs to be predicted. Squashing does not take place in input neurons; the output is simply the input. The number of hidden layers and the number of neurons in each hidden layer depend on the application; in general, the NN's output prediction improves as the number of hidden layers and hidden layer neurons are increased.

Once the architecture of the NN is determined, the initial neuron weights are randomly chosen. The NN is then trained by presenting it with many sets of training data where a set consists of a value for each input and the corresponding desired output(s). From this training data, the NN calculates (learns) the input weights for each neuron so that the NN output closely matches the corresponding input for each point in the training data. This learning process is an iterative back propagation algorithm. The learning process is terminated when the NN's predicted output(s) matches the desired output(s) for each set in the training data to within some criteria. One such criterion is the sum-of-the-errors-squared.

The more sets in the training data, the better the NN is able to predict the output(s). A rule of thumb is that the number of sets should be at least 10 times the number of weights.

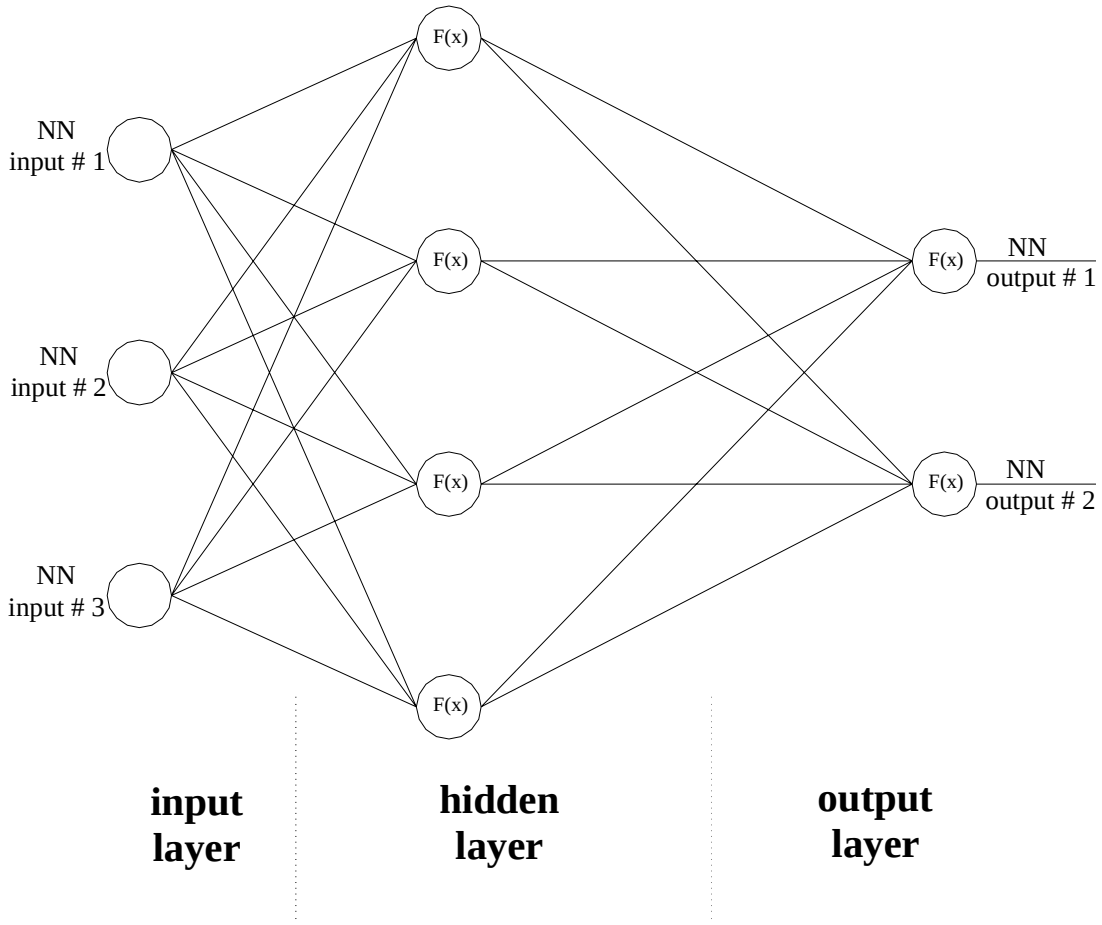


Figure 3. Typical Neural Network

Application of a Neural Network to Process Control

One possible application of a NN to process control is shown in Figure 4. This application is used for this lab experiment. While this is not the best application in terms of response time, it is very stable. The inputs to the NN are the MV and the process disturbance. Based on these inputs, the NN predicts the value of the PV. This predicted PV is then used for feedback in place of the actual PV to determine the error signal and thus the MV setting. A PI controller is needed to remove the steady state error since the NN is essentially a type 0 system.

Training data is obtained from the process every time the process reaches a steady state operating point. For training purposes, the NN compares its predicted value for the PV with the actual value of the PV and then adjusts the weights using the back-propagation algorithm.

Note that the system trains itself. For a given disturbance input and MV input, a prediction for the PV is made. Even if a bad prediction is made, a new steady state operating point is reached and a new training set is loaded into the training data.

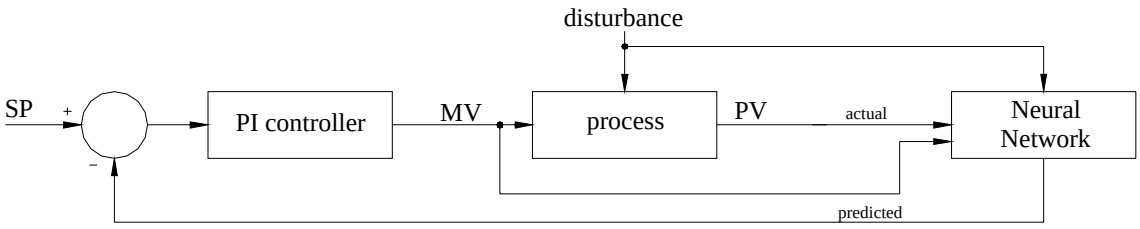


Figure 4. Block Diagram of a Possible NN Control System

Lab Equipment Descriptions

The simulation hardware is shown in Figure 5. It is based on temperature and flow simulators designed by Jim Rehg [2]. The temperature simulator consists of an enclosed box with a 23-Watt soldering iron. Varying the duty cycle of the iron's 120-Volt ac power input controls the soldering iron's heat output. This is accomplished by connecting the 4-20 mA control signal to a pulse control module (PCM) that is connected to a solid-state relay. The relay connects the 120 volts to the iron and therefore, the larger the control signal, the longer the 120 volts is applied to the iron. A type-J thermocouple connected directly to the tip of the iron measures the temperature. The box has a squirrel cage fan connected to one end and is open on the other end. A damper mounted in the inlet tube controls the airflow through the box, and a muffin fan connected at the end of the tube works as a generator to measure airflow.

The system's SP is the desired box temperature, the MV is the 4-20 mA control signal to the soldering iron, the PV is the thermocouple temperature, and the disturbance is the airflow.

The NN algorithms are implemented on a Control Micro Systems' SCADAPack programmable logic controller (PLC). It is programmable in relay ladder logic and multitasking C. It also provides up to 32 PID controllers. In its basic configuration, it provides 20 digital inputs (3 of which also function as counter/accumulator inputs), 12 relay outputs, and 8 analog inputs. Analog outputs are not included in the basic configuration and so those are added.

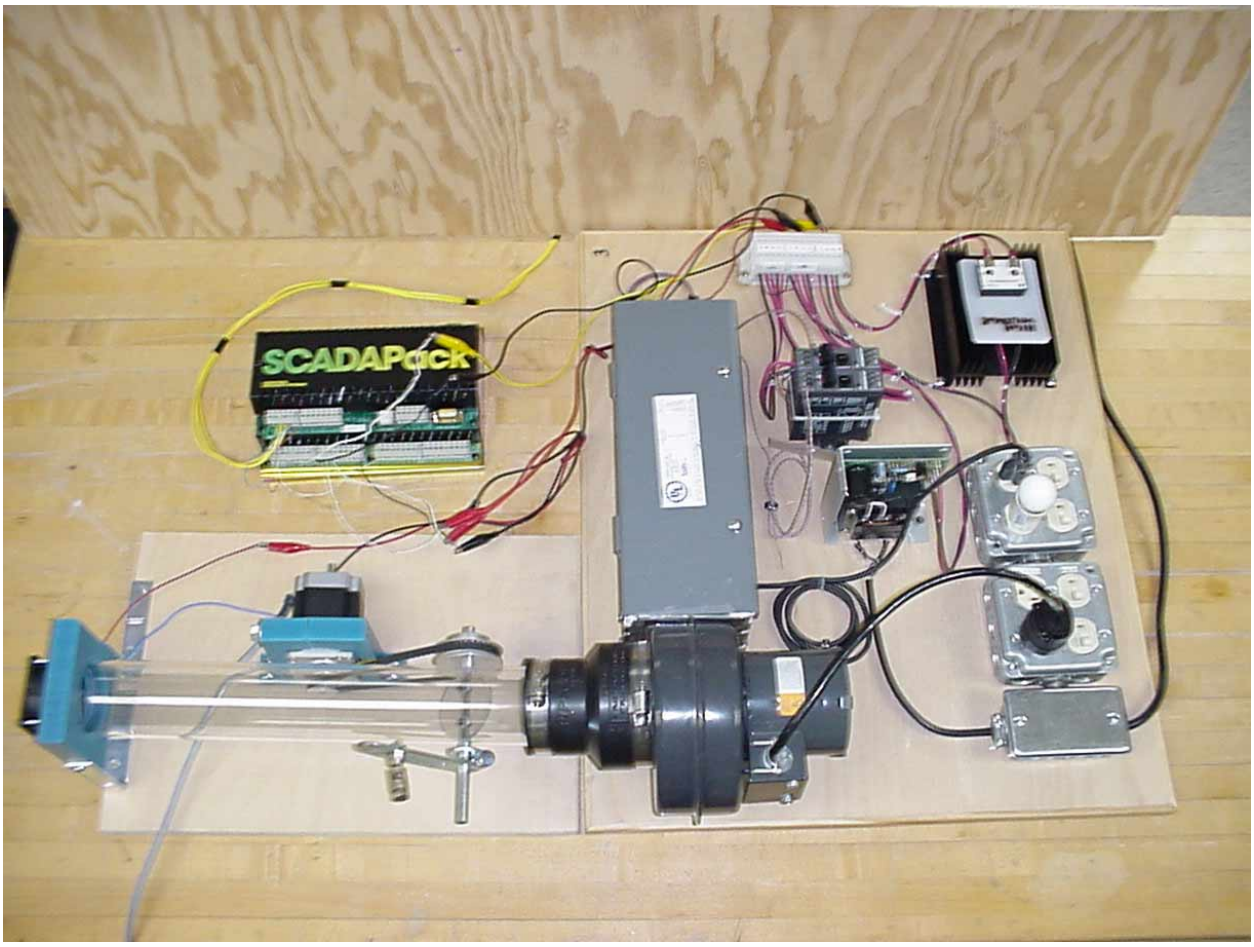


Figure 5. Simulation Hardware

Neural Network Implementation

The architecture of the NN has 2 input neurons (heater setting and airflow), one hidden layer with 4 neurons, and one output neuron (predicted temperature). Thus, there are 12 neuron weights to be determined.

The NN output and the PID control are implemented using ladder logic as a foreground task. The NN training takes place using a program coded in C running as a background task.

To keep the experiment manageable, the amount of training data is limited. The training data is based on the NN inputs and is stored in a 10 by 10 training matrix. The rows of the training data matrix correspond to heater settings that are broken down into 10% ranges of 0% to 9%, 10% to 19%, etc. All inputs are clamped to 99%. The columns of the matrix correspond to airflow and are broken down in the same manner as the heater setting. Entries in the matrix are the desired temperature for the particular heater setting and airflow inputs. Note that the 100 sets of training data are less than the desired 120.

The NN is initially "trained" by placing the system in manual. Then, each of the 100 possible combinations of heater setting and airflow is individually set and the process allowed to come to steady state. When steady state is reached, the program retrieves the temperature and stores it in the training matrix as the desired output for that set of inputs. The definition of steady state for this process is when the heater setting and airflow don't change more than 3% and the temperature doesn't change more than 3 degrees over a 30 second period.

After the initial training, the NN is put on-line to control the process. When a new steady state is reached during on-line operation, the new entry is averaged with the old entry. This demonstrates the ability of a NN to adjust for a process that might be changing with age.

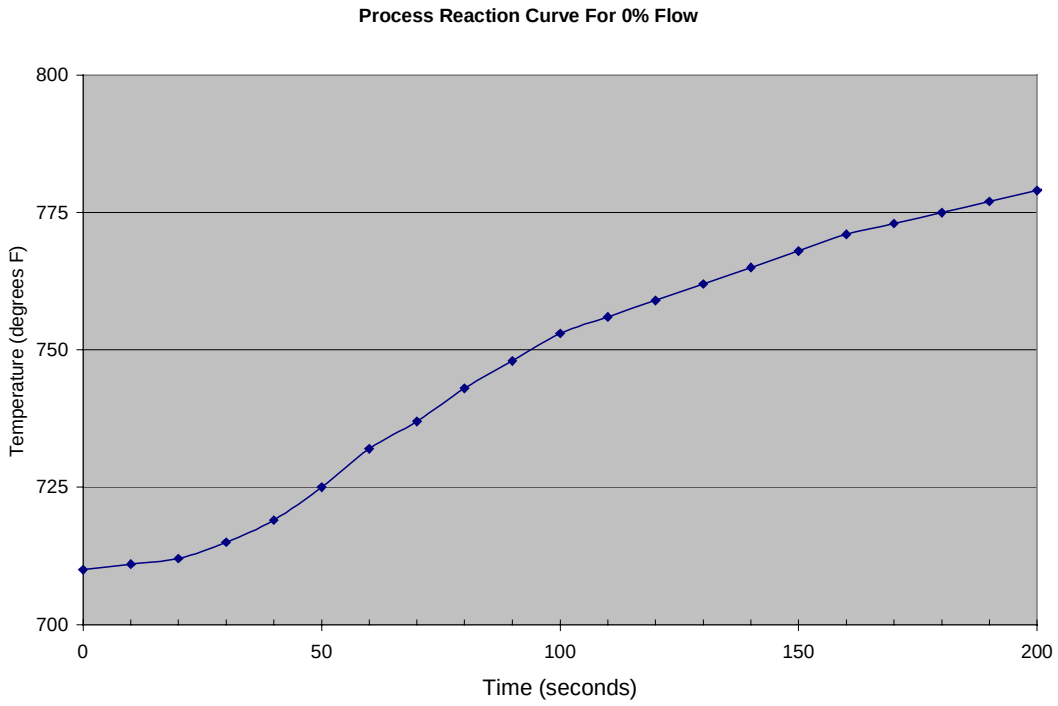
This training period points out a drawback of NN control. The initial training period took several hours to complete just to input the training data. After the training data was input, it took several days for the back-propagation algorithm to converge on a set of usable neuron weights. In practical applications, if the disturbance cannot be varied easily, the training period could take considerable time.

PID Control Results

PID parameters and results were obtained using a West model 6100 temperature controller. Process reaction curves for three different airflows were taken to demonstrate the effect of a process disturbance on the PID parameters. These curves were obtained by first letting the temperature reach steady state with the soldering iron control signal set to 40%. The control signal was then changed to 60%, and the temperature was recorded every 10 seconds until it reached steady state. Two of these curves are shown in Figure 6. The PID parameters were then calculated for each of these curves using the open-loop Ziegler and Nichols method [3]. These parameters are summarized in Table 1.

	Air Flow		
	0%	50%	100%
Kp	7.0	15	20
Ti	52 sec	40 sec	40 sec
Td	13 sec	10 sec	10 sec

Table 1. PID Parameters for Various Airflow Disturbances



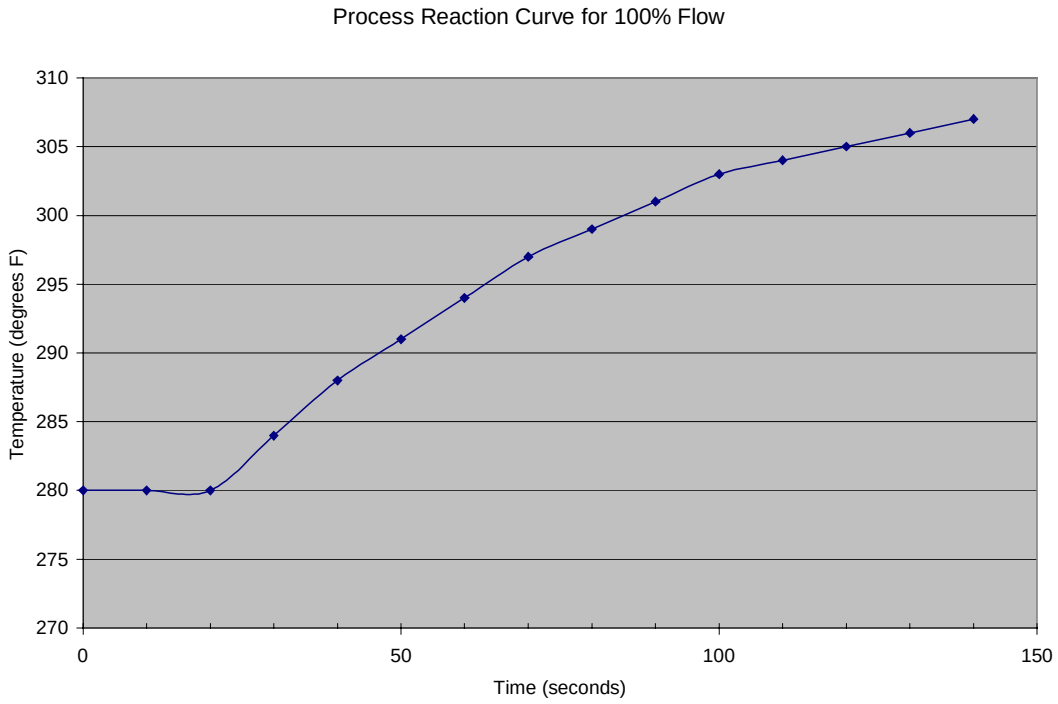


Figure 6. Process Reaction Curves for Different Disturbance Flows

To show the effects of the disturbance on the process, two step-input simulations were run using the 100% airflow PID parameters.

1. The temperature was stepped from 300 degrees to 400 degrees with the airflow at 100%.
2. The temperature was stepped from 300 degrees to 400 degrees with the airflow at 2%.

The simulation results are shown in Figure 7. It is noted that the PID controller performs as expected when operating at the designed airflow disturbance of 100%. However, when the disturbance is much lower, the response is unstable.

This result also points out that prudent PID control should be tuned for the worst case disturbance.

Neural Network Results

For this lab experiment, the PI controller varies the heater setting until the NN predicts a temperature that is equal to the setpoint.

The same two step-input simulations were run after the NN was trained. The NN results are shown in Figures 8.

It is noted from these results that the NN takes more time to reach the final value but stability problems are no longer present. It is also noted that none of the setpoints are exact because the NN is unable to predict the MV necessary to achieve the desired temperature; this is because the NN does not have enough training data.

Conclusion

This paper has described a simple lab experiment to demonstrate the application of neural networks to process control. This lab demonstrates the difficulty that PID controllers have in environments with large disturbance fluctuations. It then demonstrates the ability of a neural network to overcome these problems. It also points out the problem a neural network can have when there is insufficient training data and the drawback it has regarding lengthy training time.

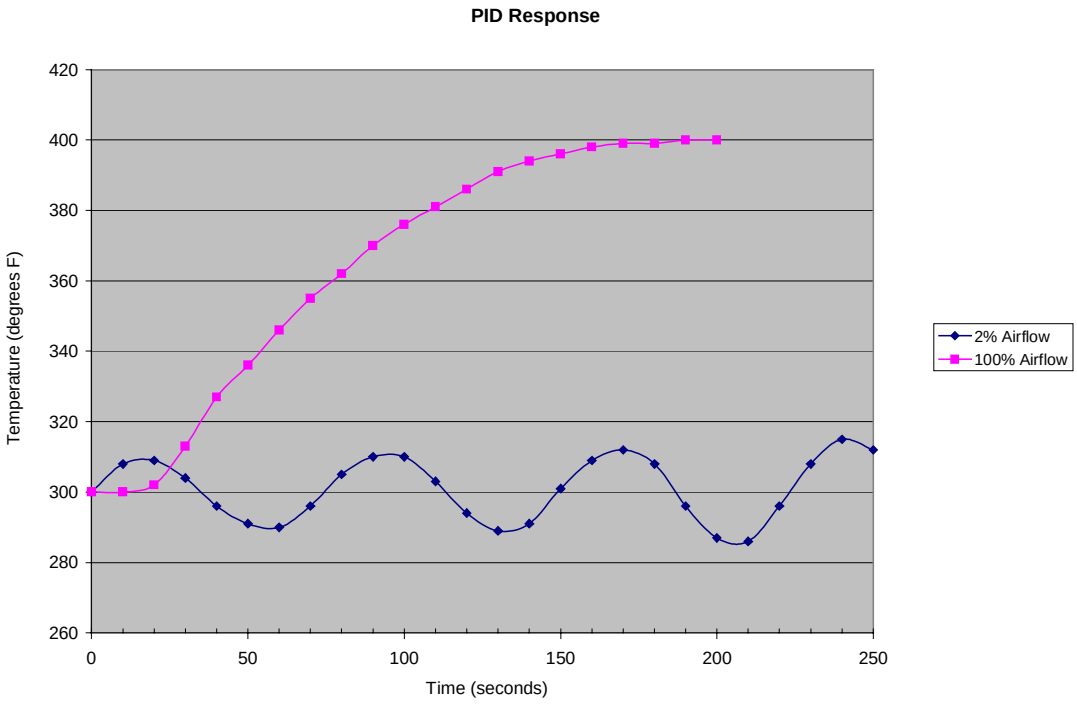


Figure 7. PID Responses Using Parameters from the 100% Airflow Process Reaction Curve

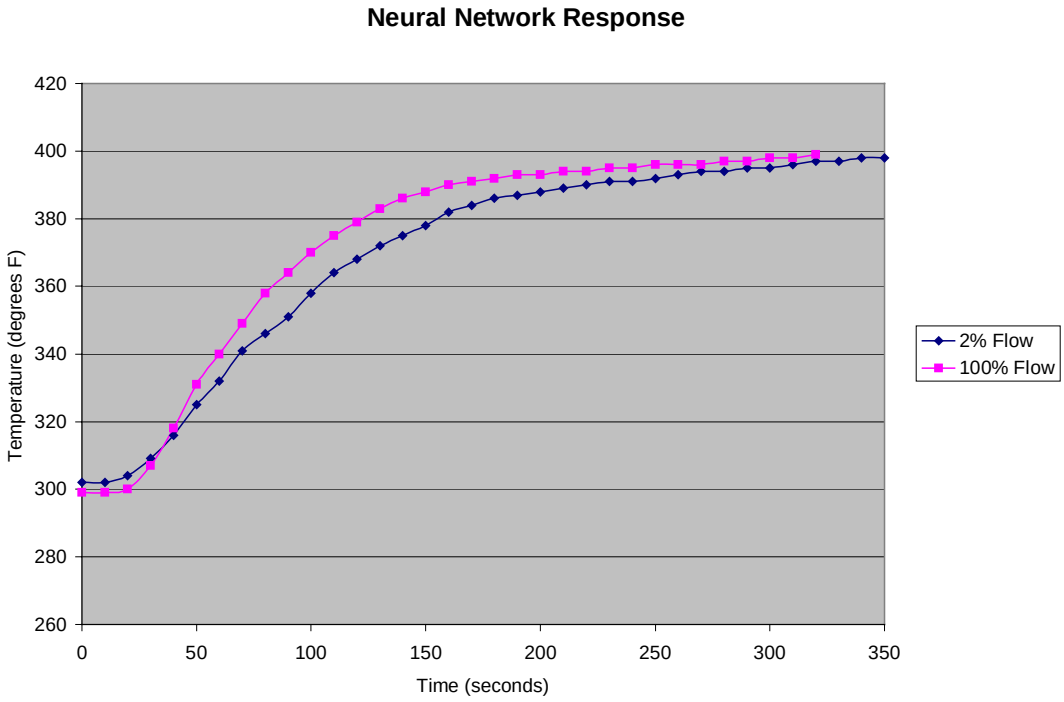


Figure 8. NN Responses to Temperature Step

Bibliography

[1] Kosko, Bart, Neural Networks and Fuzzy Systems, Prentice Hall, 1992.
[2] Rehg, James A., Low cost temperature and flow process control systems, Proceedings of the ASEE 1999 North Central Section Conference, April, 1999.
[3] Murrill, Paul W., Fundamentals of Process Control Theory, 3rd Edition, 2000.

ROBERT LYNN MUELLER
Robert Lynn Mueller is an assistant professor of Electro-Mechanical Engineering at the New Kensington Campus of the Pennsylvania State University. He received his B.S. in Electrical Engineering from Wichita State University and his Ph.D. in Electrical Engineering from the University of Pittsburgh. Dr. Mueller spent 30 years working with industrial control systems before joining Penn State and still continues to work as a consultant providing advice regarding industrial automation.