Design and Implementation of a Limited Resource PI Auto-tuning Program for First Order plus Dead Time Systems

---

A Thesis

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

---

in partial fulfillment

of the requirements for the degree

Master of Science

by

Frank Hiemstra

August

2016

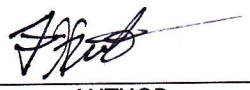
APPROVAL SHEET

The thesis

is submitted in partial fulfillment of the requirements

for the degree of

Master of Science



AUTHOR

The thesis has been read and approved by the examining committee:

Ronald D. Williams

Advisor

Gang Tao

Stephen Wilson

Accepted for the School of Engineering and Applied Science:

Craig H. Benson, Dean, School of Engineering and Applied Science

August

2016

Acknowledgments

First, I must express gratitude to my advisor, Dr. Ronald Williams. If not for him, I may have not been able to pursue a graduate degree at the University of Virginia. His intelligence of the subject has been vital to my studies, and his pursuit of excellence has spurred me to become a better researcher. He was generous with his time, and was never too busy to help me. Thank you for everything Professor.

I would also like to thank many of my fellow graduate students for their help along the way, including Thummaros Rugthum, Tamal Batabyal, Long Di and many more. Their motivation and feedback was an asset to my studies. I would like to give a special thanks to Parinya Anantachaisilp for mentoring me throughout my research.

There is no doubt I would not be the researcher and man I am today without the love and support of my family. Thank you Dad, Mom, Jessica, John Robert, and Lauren for your love and for always being there for me.

This work would not have been possible without the funding and support from Trane Corporation. The input received at our monthly meetings provided essential guidance for me in completing this work.

Abstract

One key challenges in the process control industry is designing controllers to maintain process stability. The most common controller used in this industry is the Proportional-Integral-Derivative (PID) controller. This form of control involves the tuning of gain coefficients.

Much has been done in the study of PID controllers. Traditionally, these gain coefficients have been tuned manually. Manual tuning of these coefficients is time consuming and can lead to poor performance. Therefore, systematic tuning methods were developed to achieve specific desired performance criterion for the system. Automated tuning software has been used both in research studies, and in a wide range of industrial applications. Its main feature is to calculate the gain coefficients for a PID controller.

A model-based systematic tuning approach was employed in this study. The approach involves testing the physical system by controlling an actuator. The system's response to the test is used by a plant identification method to generate an approximation of the plant. This plant approximation is then mapped to PID gain coefficients using PID tuning rules. The modeling and tuning can require a significant amount of storage in memory. This research creates a low-resource auto-tuning program, using a new plant-identification technique, and was paired

with an existing PID tuning rule to generate PID gain coefficients. The auto-tuning program was developed to obtain these gain coefficients using efficient, low-memory algorithms that do not sacrifice performance. The algorithms use significantly fewer variables than other methods. The auto-tuning program applies statistical process control by using individuals control charts, and uses a streaming algorithm to compute the corrected sum of squares for standard deviation.

This research was carried out as a case study of the Chilled Beam Water System at the University of Virginia's Rice Hall. Focus is on the interaction between a water valve, known as the Bypass Valve, and a downstream water temperature, the Chilled Beam Supply Water temperature. This system can be characterized as a first order system with dead time. Dead time, also known as time delay, is present in any system where a signal or physical variable originating in one part of the system becomes available in another part after a lapse of time. In this system, like many HVAC systems, dead time varies. This causes the need for the PID gain coefficients to be readjusted from any initial setting in response to changing dead time. Automatic tuning provides a way to readjust these gain coefficients as needed.

Both simulation and experimentation are carried out on the Chilled Beam Water System. The auto-tuning program's plant identification technique is compared with common plant identification techniques in the literature. Many PID tuning rules exist for different performance specifications, and for different plants. This thesis examines tuning rules specified for first order with dead time systems. The plant identification techniques and PID tuning rules are compared among each other by examining the generated PID gain coefficients. The PID gain coefficients

are then compared in simulation by evaluating the ability of each to recover from a disturbance.

The comparisons indicate the auto-tuning program, although using significantly fewer variables, still achieves high fidelity performance like the other plant identification methods that use more resources. Among PID tuning techniques, the Internal-Model-Control (IMC) tuning rule produced the best performance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Building automation systems control and regulate a building's heating, ventilation, air conditioning, lighting, and other systems. This largely came about from the development of networked control systems. Building automation began with direct digital control being integrated in these networked systems [1]. The goals of building automation are to improve energy efficiency, lower operation costs, and prolong life of mechanical utilities while improving comfort for the building occupants. Properly controlling the processes within the building can help achieve these goals.

In order to operate the equipment used to control the different processes within a building, direct digital controllers, known as Programmable Logic Controllers (PLCs), are used. PLCs are small embedded computing devices programmed to control actuators. Although there are many different methods of control, this thesis focuses on Proportional-Integral-Derivative (PID) control, as it is the most common form of control in building automation. Achieving the goals of building automation starts with the performance of the controllers controlling the equip-

ment in the building.

PID control has three parameters which effect the behavior of the control. These parameters are known as PID gain coefficients. The optimal values of these gain coefficients are unique to the process the controller regulates. Adjusting these gain coefficients is known as PID tuning.

Historically in HVAC systems PID gain coefficients were tuned manually by a technician, which is not only costly, but also produces mediocre results. In manual tuning, there is no criteria for process performance or robustness; rather the controller is tuned based on technician intuition. Even if a technician could perfectly set the PID gain coefficients for system, system dynamics often change with temperature, season, pressure, mechanical wear, and many other factors. The optimal PID gain coefficients for a system may then change over time. In manual tuning, a technician would need to constantly be adjusting the PID gain coefficients if the optimal gain coefficients were desired. This suggests the need for a method to simplify tuning PID gain coefficients.

This led to the development of systematic tuning, which have specific criteria for performance and robustness. Systematic tuning requires an engineer to run careful analysis on experimental data from the system to formulate PID gain coefficients. Later, PID auto-tuning programs were developed, which automatically tune the PID gain coefficients using systematic tuning techniques, but accomplish this without an engineer. These programs are then implemented onto PLCs. Auto-tuning programs are very common. Siemens, ABB, Omron, GE, Honeywell, and many other corporations all have their own auto-tuning programs on their perspective PLCs. This thesis has developed an auto-tuning program, which has

been specially designed to be low-resource.

The principal algorithm in systematic tuning is the algorithm which performs plant identification, as the plant approximation is used to determine optimal PID gain coefficients (more on this in Chapter 2). Plant identification approximates the process dynamics to fit a model. Most HVAC processes can be modeled as a First Order Plus Dead Time (FOPDT) process. The plant identification algorithm takes an input and output data for the process, and approximates FOPDT process parameters. Common methods to do this include Two-Point Identification, Method of Areas, and the Graphical Approach. To carry out these methods, each requires storage of the input and output data. This leads to a large number of variables needed to store these data points. The auto-tuning program described in this thesis does not require all of the data to be stored. This reduction in storage allows the new auto-tuning program to fit directly into the PLC.

The connectivity within building automation systems allow devices to be interconnected using a building management system. This allows for every controller to be accessed over the internet. With a program loaded onto the controller, a technician is not required to manually tune these PID gain coefficients, and therefore the gain coefficients can be tuned from the building management system.

1.1 Contributions

This thesis has developed a low-resource PID auto-tuning program for FOPDT systems. Embedded devices are often constrained by the available hard-disk and virtual memory. This program was constrained to the resources available on the Tracer MP581 Programmable Controller. The auto-tuning program uses stream-

ing algorithms to lower the number of variables needed for computation. The program applies a form of statistical process control by using individuals control charts [2] to perform change detection. To compute variance, this program applies B.P. Welford's corrected sum of squares technique, which is also a streaming algorithm [3]. These algorithms are used by the auto-tuning program to perform plant identification, which is then used to obtain PID gain coefficients. The use of these algorithms has allowed the auto-tuning program to fit within the resource constraint of the Tracer MP581 Programmable Controller.

1.2 Case Study: Rice Hall Chilled Beam Water System

The case study of this thesis was carried out on the HVAC system at Rice Hall, a building at the University of Virginia in Charlottesville, Virginia. The auto-tuning program in this thesis was implemented on TRANE's Tracer MP581 Programmable Controller. This project limits its scope to the PID control loop controlling the Chilled Beam (CB) Water System, illustrated in Figure 1.1. The CB Water System is the name of the process defining the interaction between the actuators (valves), input water temperatures, and the output water temperature, known as the CB Supply Water temperature. As shown in Figure 1.1, two valves effect the CB Supply Water makeup. First, the Mixing Valve determines the ratio of CB Return Water to CB New Water. The CB Return Water is typically warmer than the CB New Water. Second, the Bypass Valve determines how much System Supply Water should be mixed into the CB New Water. The CB New Water is

a combination of three different flows: Server Return Water, Penthouse Return Water, and System Supply Water.

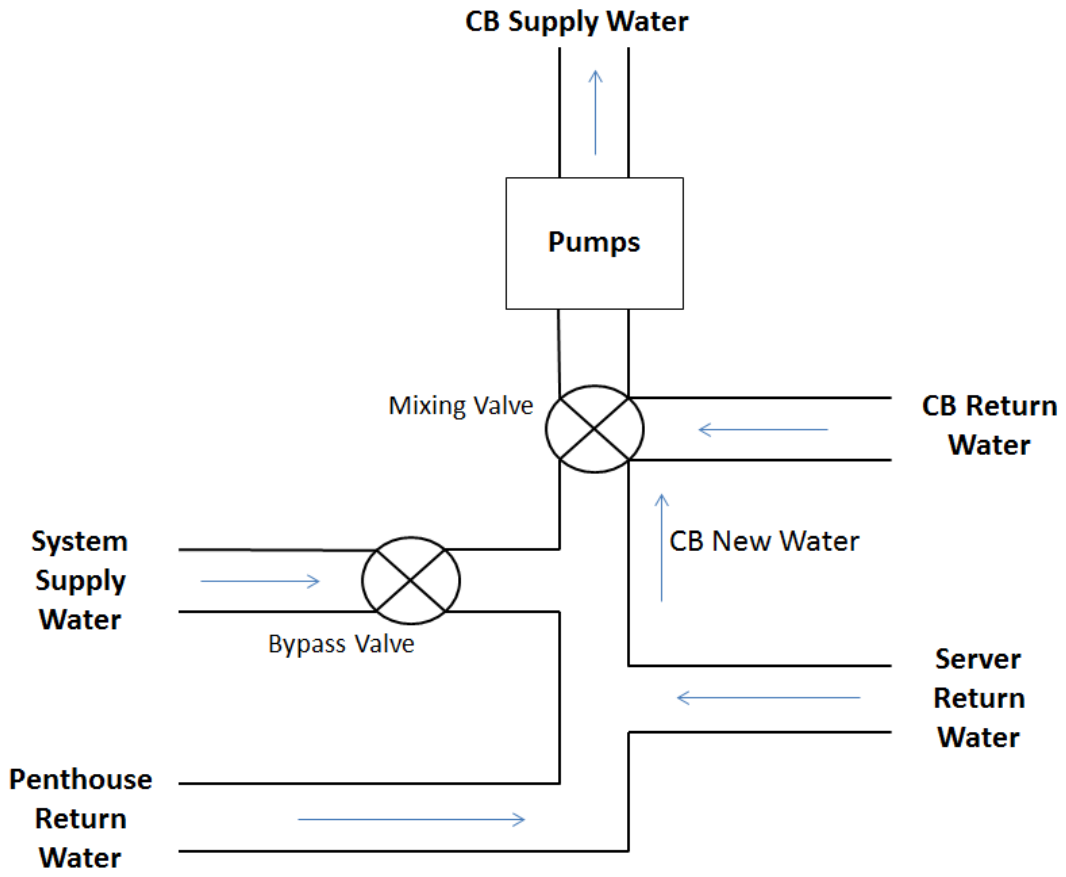


Figure 1.1: CB Water System in Rice Hall

Throughout this thesis, the Mixing Valve remained unchanged at 100% open, meaning it did not take in any CB Return Water, and only took CB New Water. The Bypass Valve is the specific piece of equipment this study is focused on. This valve is a resilient seated butterfly valve, and is manufactured by Bray International. The Bypass Valve is controlled by a PID controller that is the controller of interest in this study. The goal of this controller is to adjust the Bypass Valve position in a way that allows the CB Supply Water to maintain its desired temperature set point.

Like most physical systems, the CB Water System dynamics change over time.

The flow rate of the different water inputs that make up the CB New Water is known to change. This variable flow rate leads to variable process dead time. This change in dead time significantly impacts the CB Water System dynamics. To compensate for this variation in dead time, readjustment of PID gain coefficients is necessary. The auto-tuning program in this thesis can be used to readjust the gain coefficients.

1.3 Problem Statement

The Bypass Valve of the CB Water System examined in this project has been identified as a misbehaving valve. It has exhibited oscillatory behavior, causing the CB Supply Water temperature to also oscillate. This reduces efficiency and increases wear. After investigation, it was identified that the PID controller commanding the valve caused the oscillations. This thesis suggests tuning the gain coefficients of the PID controller can eliminate the oscillatory behavior. The objective of this thesis is to develop a robust, low-resource PID auto-tuning program for First Order Plus Dead Time (FOPDT) systems with varying dead time. This auto-tuning program will fit on a PLC and can be controlled from the building management system. A goal of the auto-tuning program algorithms is to limit the number of variables used. This program is designed to determine the PID gain coefficients for FOPDT systems, and this research is put into practice at Rice Hall on the PID controller commanding the Bypass Valve.

1.4 Thesis Organization

This paper is organized as follows. Chapter 2 describes the background of the problem and existing solutions. This includes discussing previous work done in the field of PID tuning and plant identification to put the work presented in this thesis into context. This section also provides justification for the CB Water System modeling approach.

Chapter 3 discusses equipment and techniques used to run experiments on the CB Water System. Here the methodology of the experiments is described along with a description of a tool designed to collect data from the Tracer MP581 Programmable Controller.

In Chapter 4, the auto-tuning program is discussed. This includes describing the different algorithms used in the program. Also, the actual program run on the Tracer MP581 Programmable Controller is described.

In Chapter 5, results and analysis of using the auto-tuning program on the system are described. Here the PID gain coefficients obtained from the program are compared to gain coefficients obtained from other methods. The comparison is carried out using a Simulink simulation of the CB Water System. The different PID gain coefficients are placed in the controller, and each is tested to determine its ability to respond to a disturbance. A robustness measure is also considered in the presence of varying dead time.

Finally, Chapter 6 finishes this work with concluding remarks regarding the auto-tuning program and PID tuning. Areas for possible future work are examined in the field of PID tuning.

Chapter 2

Background

A feedback controller is a device, or software package, which manages a system based on feedback. The system's measured output is fed back to the controller and used to influence the control of the system [4]. A controller controls an actuator, which is the physical device that influences the state of the process. This research utilizes PID control.

2.1 PID Control

A PID controller is a popular instrument in the process control industry, first described by Callender et al. [5]. It has been estimated that 90% of industrial controllers are PID, with the majority being PI controllers [6–8]. These controllers can be used to control any process variable, including flow, pressure, temperature, speed, and much more. PID controllers are used because of their simplicity, versatility, and applicability. Traditional PID controllers take the form described in Equation 2.1 and illustrated in Figure 2.1, where all quantities are expressed as LaPlace transforms.

$$U(s) = (K_p + K_i/s + K_d s)E(s) \tag{2.1}$$

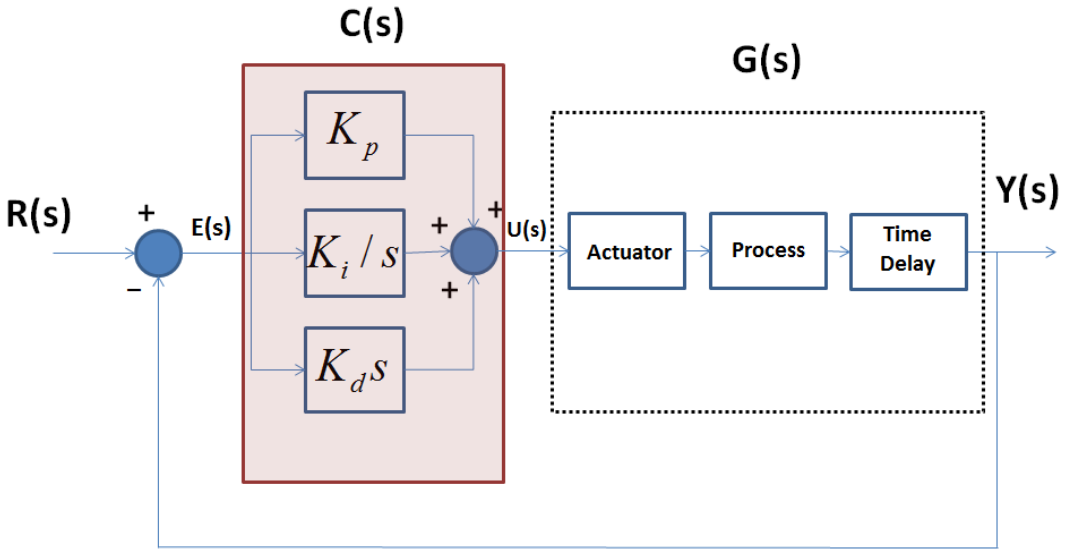


Figure 2.1: Closed-loop system with PID control

R(s) = Set Point

E(s) = Error [R(s) - Y(s)]

C(s) = PID Controller

U(s) = Control Output

G(s) = Plant

Y(s) = Process Output

PID controllers are widely used in the process control industry to control a variety of systems. Designing a PID controller requires tuning of three gain parameters. This will be reviewed in Section 2.5. Well-tuned controllers have a variety of benefits, including but not limited to energy efficiency, reduction in mechanical wear of physical systems, and stable and robust closed-loop systems

performance. Often, technicians determine the gains by trial and error. To determine the optimal gain parameters, systematic tuning procedures have been developed. This method requires an engineer to calculate the gains after studying the system. With millions of PID controllers, it is not feasible to have an engineer tune every single controller. As a result, automated tuning programs have been developed to achieve optimal gain parameters, without having an engineer calculate the gain parameters for each system. These programs are commonly referred to as auto-tuners.

2.1.1 Proportional control

In proportional-only control, the feedback control signal is linearly proportional to the error between the set point and the measurement of the process output [4]. It is described in Equation 2.2

$$u = K_p(r - y) \tag{2.2}$$

The proportional gain, K_p, is an adjusted parameter the control designer sets to achieve the desired sensitivity to error. In classical control, this proportional gain is equal to the process gain, K.

2.1.2 Integral control

On its own, a proportional only controller has a tendency to have a nonzero bias away from the set point. In order to eliminate this offset bias, the control action is also made a function of the integral of the error as shown in Equation 2.3.[9]

$$u = K_i \int_0^t e(\tau)d\tau \tag{2.3}$$

Here, K_i is the integral gain. The ability for integral control to eliminate steady state error is valuable; however, on its own integral control is weak because little control feedback takes place until the error has accumulated over time. In tandem with proportional control, which is immediately responsive to error, a controller is able to better achieve control objectives. The combination of these two control strategies is called proportional-integral (PI) control. Both the proportional and integral gains are tuned to meet the desired performance. In classical control, the integral gain is equal to the control gain divided by the reset time, $T_i i$ as shown below.

$$K_i = \frac{K}{T_i} \tag{2.4}$$

2.1.3 Derivative control

The premise of derivative control is to provide the controller with the ability to anticipate using a prediction of the output based on linear extrapolation. It is accomplished by responding to the time rate of change of the error, as is shown in Equation 2.5.

$$u = K_d \frac{de(t)}{t} \tag{2.5}$$

As with the other two control strategies, the derivative gain can be adjusted to control the sensitivity to the time rate of change of the error. The derivative gain can be calculated as described in Equation 2.6 where T_d is the derivative time.

$$K_d = KT_d \tag{2.6}$$

2.1.4 PID Control using the Velocity Model

The Velocity Model is a method of implementing discrete time PID control. This method is used in the Tracer MP581 Programmable Controller, making it the method used in this thesis. It is based on the derivative of the traditional PID controller, with the current control output, $u(n)$, equal to the input at the previous time step plus the change in the control output at the current time. This is shown in Equation 2.7 below:

$$u(n) = u(n-1) + \Delta u(n) \tag{2.7}$$

The backward difference approximations can be substituted into the velocity PID model to get equation 2.8:

$$\frac{\Delta u(n)}{T} = K_p \frac{e(n) - e(n-1)}{T} + K_i e(n) + K_d \frac{e(n) - 2e(n-1) + e(n-2)}{T^2} \tag{2.8}$$

Using the velocity model, the gain coefficients must account for the sample time, T, multiplied out to obtain $u(n)$. This makes these gains equal to:

$$\hat{K}_p = K_p \tag{2.9}$$

$$\hat{K}_i = K_i T \tag{2.10}$$

$$\hat{K}_d = \frac{K_d}{T} \tag{2.11}$$

Derivative control is not often used in systems with long dead time [9]. This includes most HVAC applications, and the CB Water System. For this reason, the derivative term is ignored by setting the derivative gain to zero.

2.2 CB Water System Plant Model

To characterize the system, the system plant model approximation must be identified. This is accomplished by applying a step to the system and observing the response. Figure 5.1 shows a step being applied to the Bypass Valve, with the CB Supply Water temperature reaction compared.

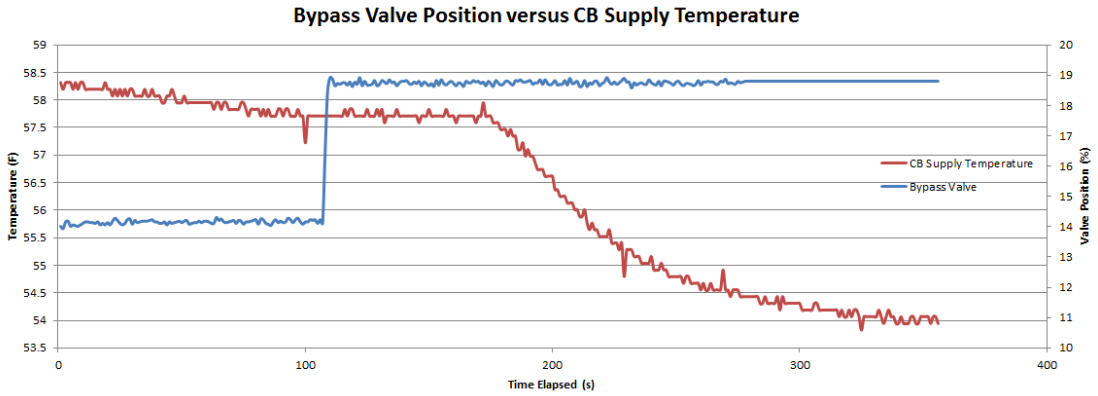


Figure 2.2: CB Supply Water temperature response to a step in the Bypass Valve position.

Figure 5.1 gives insight into how to approximate the plant model of the CB Water System. The shape of the response indicates the order of the system[1]. There is a delay between the step in the Bypass Valve and the initial change in CB Supply Water temperature due to the step. This delay is referred to as dead time. Once the CB Supply Water temperature begins to change, there are no abrupt fluctuations, and the response ends with the system settling to a new steady state value. This indicates the system is of the first order, giving the approximation of the plant model as a First Order Plus Dead Time (FOPDT) system. This is verified in simulation by finding a FOPDT model to approximate the data in Figure 5.1. This simulation is shown and discussed in greater detail in Chapter 5.

Because this research deals with a FOPDT system, the plant, $G(s)$, can be approximated in the form represented by Equation 2.12,

$$G(s) = \frac{K}{\tau s + 1} e^{-\theta s} \tag{2.12}$$

with variable represents as follows:

- Process Dead Time, θ
- Process Time Constant, τ
- Process Gain, K

Since the Bypass Valve was commissioned to Rice Hall in 2012, the valve position has operated between 0 and 20%. From 0 to 20%, the relationship between the flow through the valve and the percent of valve opening is approximately linear.

2.2.1 Changing Building Load

As shown in Figure 1.1, the CB Supply Water Temperature has multiple water sources feeding into it. Each of these water sources is a function of the load the building is commanding; therefore, the temperature values of each water source are variable. This means the building load has the potential to fluctuate. This is characterized as a process disturbance.

2.3 Dead Time

Dead time is phenomena that occurs when trasport delays exist in a process. This can occur within a system for a variety of reasons including a pure delay

mechanism caused by transport, computation time, and communication. Dead time is commonly referred to as time delay, or transport delay. For any process, as the dead time increases, the control challenge becomes greater [10]. Figure 2.3 shows a delay representation.

Figure 2.3: Dead Time in Process

In the process control industry, dead time is the time it takes the system to respond to a change in the control output. From the same data-set in Figure 5.1, dead time can be demonstrated in the CB Water System. The dead time from this data-set is labeled in Figure 2.4. The time it takes the temperature to change after a valve has moved is the system dead time.

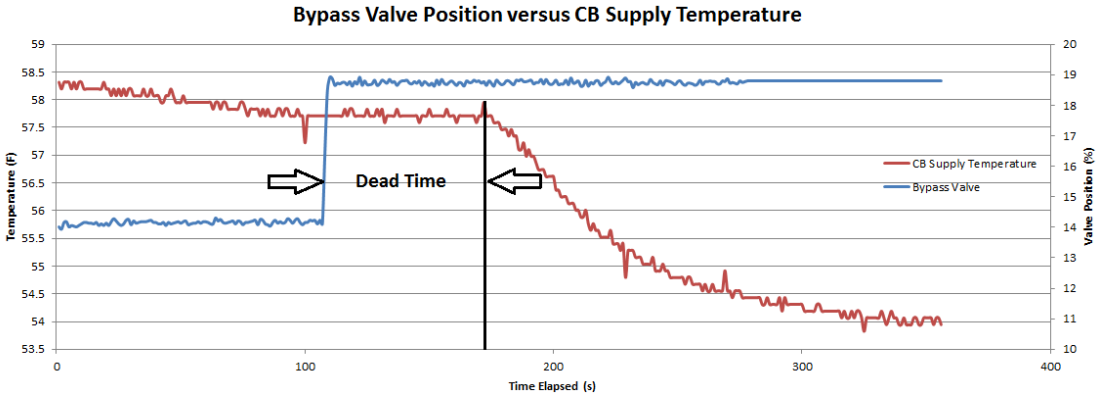


Figure 2.4: Dead Time in the CB Water System

2.3.1 Varying Dead Time in CB Water System

In the CB Water System, dead time varies. A cause of the change in dead time is the CB Supply Water flow rate. The faster the flow rate, the shorter amount of

dead time.

Throughout this thesis, steps were applied to the system at different times of the year. For example, below are two step tests. Figure 2.5 shows data from November 2015, and Figure 2.6 shows data from February 2016. In November when the step test was taken, the flow rate through the CB Supply Water pipe was 0.794GPM. When the February step test was taken, the same flow rate was 75GPM.

The data shows two very different dead times associated with the different step tests. Figure 2.5 shows a dead time of approximately 2.5 minutes, whereas Figure 2.6 shows a dead time of approximately 0.9 minutes.

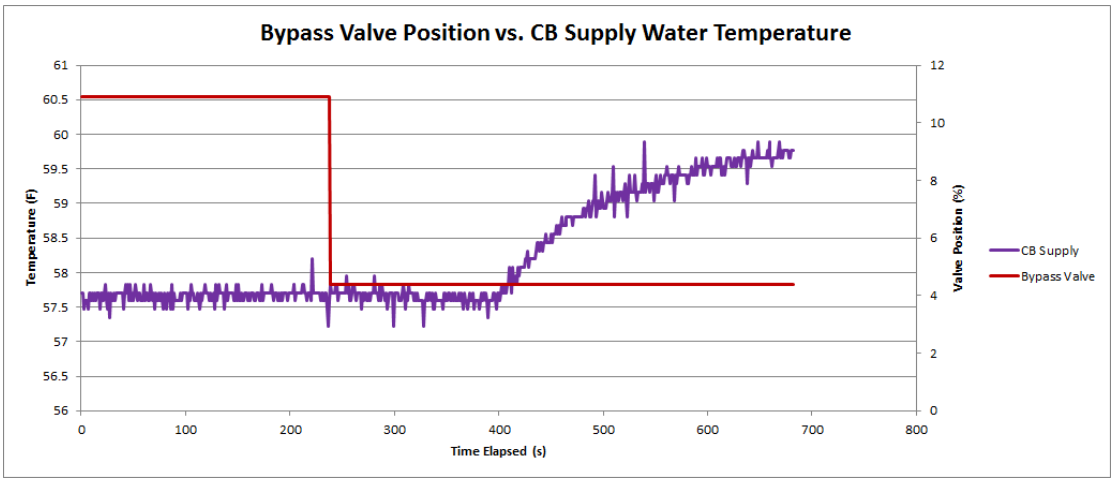


Figure 2.5: Step Test Data from November 2015

With changing plant dynamics, such as dead time, different PID gain coefficients are optimal for different plant dynamics. An auto-tuning program can be used to readjust these gain coefficients.

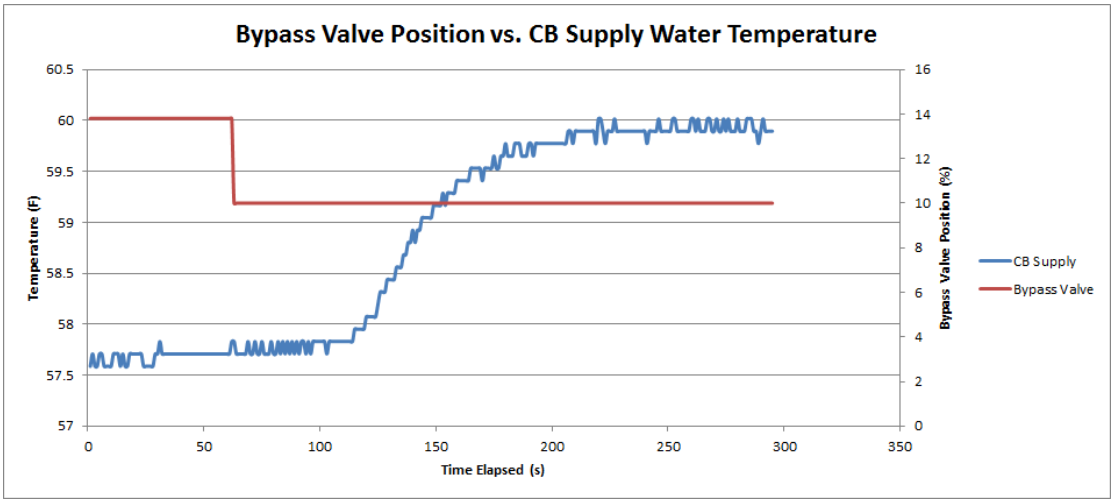


Figure 2.6: Step Test Data from February 2016

2.4 Dead Time Compensation

It is common to control processes with dead time with a dead time compensation technique, particularly when

$$\frac{\theta}{\tau} \geq 1 \tag{2.13}$$

as is true in the CB Water System. However, for this thesis dead time compensation was not chosen for a variety of reasons. The most popular dead time compensation technique involves using a Smith Predictor [11]. Smith Predictors use a model of the process and compare to the actual process to make decisions which involves prediction. This method will not work on the Tracer MP581 Programmable Controller because of the resource constraint. The Smith Predictor requires a significant amount of resources because it calculates the response of a process model. Also, Smith Predictors are most effective in set point disturbance, whereas this project deals with load disturbance [10]. For these reasons, dead time compensation is not used in this thesis.

2.5 Tuning PID Gain Coefficients

There are three ways to tune the gain coefficients of a PID controller: manual, rule-based, and systematic tuning. Manual tuning uses the knowledge of an individual for how to manually tune each PID gain coefficient until the system reaches a desired performance. Rule-based tuning utilizes a set of formulated rules to map certain error characteristics to a PID gain coefficient. Systematic tuning models an approximation of the plant to then calculate the PID gain coefficients using a set of PID tuning rules.

2.5.1 Three Tuning Approaches

Manual Tuning

This approach is most similar to a trial-and-error process, using the knowledge of the process and the effects of controller parameters. The procedure of manual tuning is described in [12], and is summarized as follows in terms of the system given in this thesis:

1. Start with the valve at a fixed position, with K_i and K_d at zero. Slightly increase K_p in small increments until some output oscillation occurs.

2. Now, add damping with K_d, which should reduce the vibration of the valve. Again, gradually increase K_d in small increments until oscillation is eliminated or very small.

3. If the temperature is not at the set point, gradually increase K_i, which will eliminate steady state error.

The above procedure has been proven successful in many systems, but is tedious, time consuming, and dangerous in many systems. It does not guarantee robustness or a specific desired performance.

Rule-Based

In rule-based methods, the approach is to wait for transients, set point changes, or load disturbances, and then make judgment based on the specific error or change in error. If these actions cause the system performance to deviate from specifications, the controller parameters are adjusted based on rules. There are many sets of rule-based methods that exist, but most follow this general guideline described in the table below:

	Speed	Stability
K	increases	reduces
T_i	reduces	increases
T_d	reduces	increases

Table 2.1: Rule-Based Guidelines

Model-based Tuning

Model-based tuning approaches are designed to tune the PID gain coefficients to achieve a desired performance criterion. These coefficients are based on information from a plant approximation. To generate this plant approximation, most model-based approaches follow the steps outlined in Figure 2.7.

In model based approaches, the first step is to run an experimental test on the

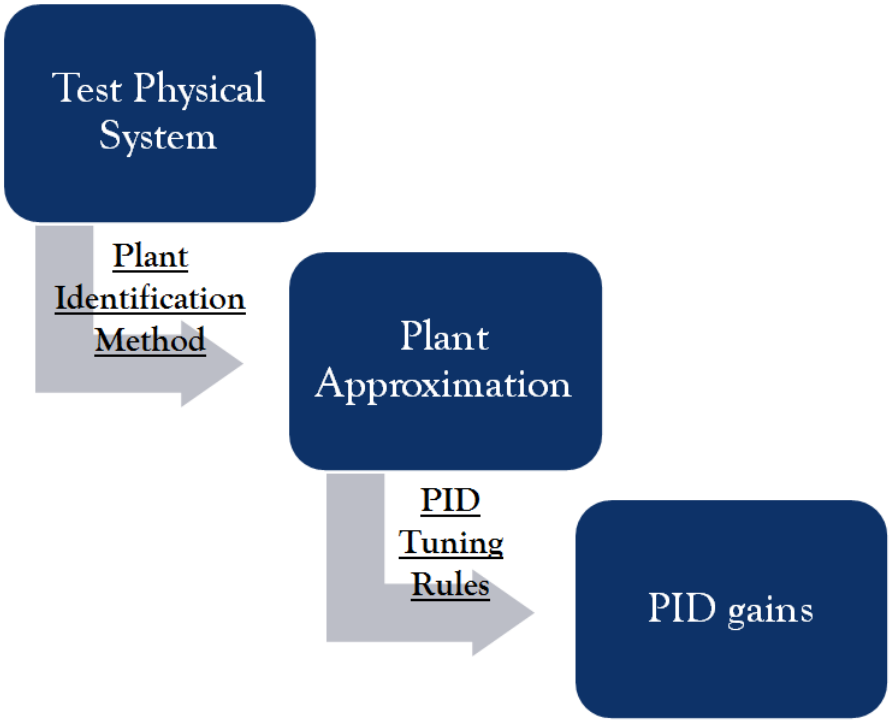


Figure 2.7: Model-based Approach

system. The type of test run is different depending on the model-based method used. The test used is able to gather input and output data, which is used for plant identification. A plant identification method, which are described in Section 2.5.4, uses this input and output data to generate an approximate plant model. This plant approximation can be translated into PID gain coefficients by using a set of PID tuning rules, which are discussed in Section 2.5.5. Sections 2.5.2 and 2.5.3 discuss two different approaches to model-based tuning.

2.5.2 Frequency Response Methods

This method uses frequency information to determine the PID gain parameters by obtaining the ultimate parameters of the system dynamics. It was first proposed by Ziegler and Nichols [13], where they connect a closed-loop proportional-only controller to the system and increase the gain until the system has a sustained

oscillation. This is known as the Continuous Cycling Method. The final gain is noted as the *ultimate gain*, K_u, and the period of oscillation is noted as the *ultimate period*, T_u. These ultimate parameters indicate when the system becomes unstable. This approach is not suitable for open-loop unstable processes.

Another method to determine this frequency information was proposed by Astrom and Hagglund [14] and is called the Relay Feedback Test. This method also determines the ultimate parameters, but in a different fashion. In this approach, an on-off relay is placed in the feedback loop, taking out the PID controller. To carry out this approach, use the following steps [1]:

1. Disconnect the PID controller and manually control the signal being sent to the actuator (the manipulated input) using the on-off relay.

2. Bring the system to a steady state below the set point.

3. Make small, yet significant increase to the manipulated input. The typical increase is 3 - 10%.

4. As soon as the output crosses the set point, swing the manipulated input in the opposite direction.

5. Repeat step 3 and 4 until sustained oscillation is observed.

Running this experiment gives us the information shown in Figure 2.8. The period of the sustained oscillation is the ultimate period, therefore, the ultimate frequency (ω_u)from this relay feedback experiment is

$$\omega_u = \frac{2\pi}{P_u} \tag{2.14}$$

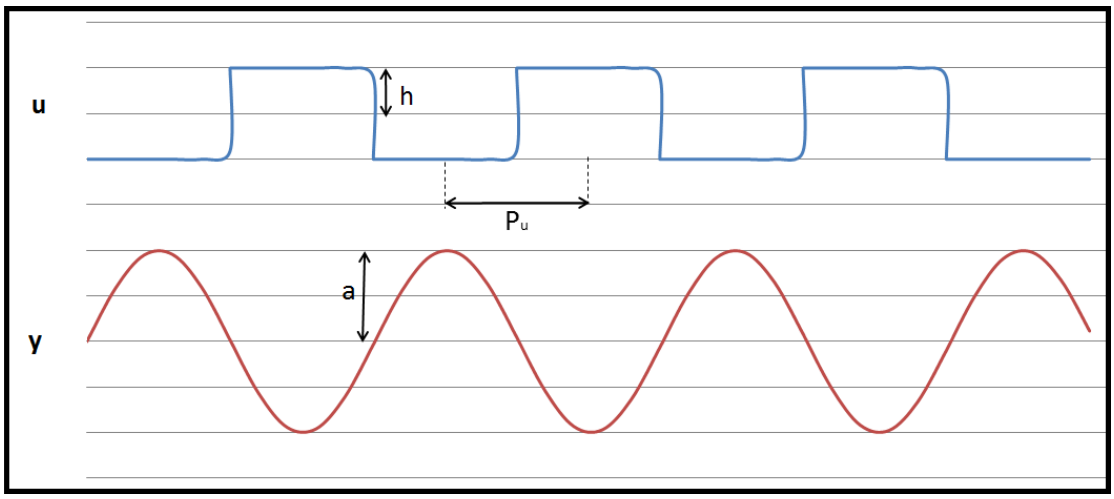


Figure 2.8: Relay Feedback Test - y: System Output, u: Manipulated Input [1]

The ultimate gain can be approximated from this relay feedback experiment described in equation 2.15 [9]

$$K_u = \frac{4h}{a\pi} \tag{2.15}$$

From the ultimate gain and frequency, a plant model can be approximated for the system using plant identification techniques. These techniques use the information from the manipulated input and system output to produce a plant approximation of the system. PID gain coefficients can be derived from the plant approximation. This method is a preferred technique in many industrial applications; however, in this project a different approach is better. In systems where the ratio in Equation 2.16

$$\frac{\theta}{\tau} > 0.28 \tag{2.16}$$

holds true [1], the Step Response Method is preferred because it will be more time efficient to execute the test experiment. For the CB Water System, Equation 2.16 holds true, and for this reason, the Step Response Method is chosen for this project.

2.5.3 Step Response Method

This method, also commonly referred as the Process Reaction-Curve method, was developed by Ziegler and Nicholas [13] to tune PID gain coefficients. This method uses an open-loop step test on the actuator to identify the plant model, and subsequently the PID gain coefficients. This method also follows the model-based approach in Figure 2.7. The general steps to running the test experiment, illustrated in Figure 5.2, are as follows:

1. Disconnect PID controller and take command of the control signal.
2. Keep the control signal constant and wait for the process variable (system output) to become stable.
3. Once the process variable is stable, increase the control signal 3 - 10%.
4. Observe system response. When the process variable stops changing and becomes stable, the experiment is over.

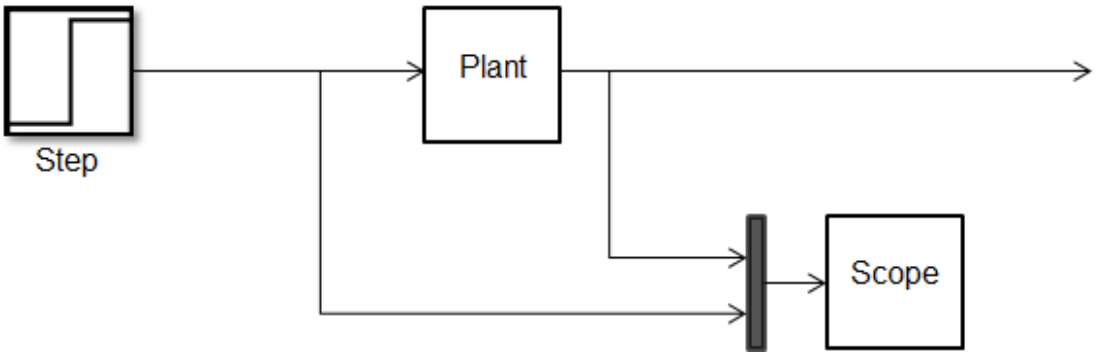


Figure 2.9: Step Test Diagram

The Step Response Method is a simple, proven approach well-suited for this application. Figure 2.10 illustrates an example of a step test being performed on

an arbitrary controller and system. Discussion on the other parts of the model based approach, plant identification and PID tuning rules, are to follow.

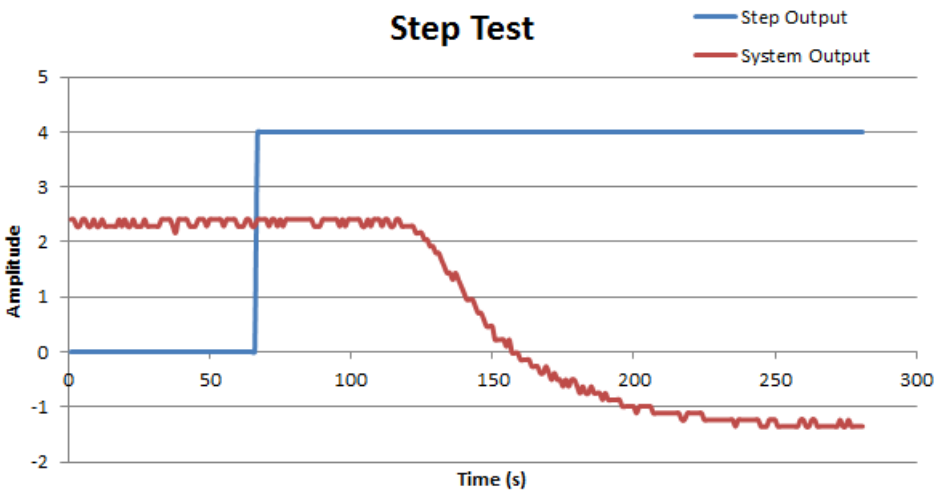


Figure 2.10: Step Test

2.5.4 Plant Identification

The system's response to a step in the control output gives valuable information about the system, as described in Section 2.5.3. There are a variety of methods to translate the system's step response into information about the system. The following are four different methods seen in the literature. This is not an exhaustive list, but the most simple and common methods. Complex methods are not viable because of the resource limitations of the physical controller.

Each system gain depends on the time when the process variable begins to change in response to the applied step and the time when the process variable stops changing. The same technique is used for the four different methods, to allow for fair comparison of the methods. The details of this technique are described in the design of the auto-tuning program later in this thesis.

Method of Areas

The Method of Areas [15] approach utilizes integration to manipulate the area under the process variable curve. Given the step response $y_s(t)$, the process gain K is computed by dividing the response total swing by the control input step amplitude A_s. The unit step response $y_{us}(t)$ is defined as $\frac{y_s(t)}{A_s}$. Equations 2.17, 2.18, and 2.19 are used in the final calculation of time constant and dead time,

$$A_0 = \int_0^{t_{end}} (K - y_{us}(t))dt \tag{2.17}$$

$$t_0 = \frac{A_0}{K} \tag{2.18}$$

$$A_1 = \int_0^{t_0} y_{us}(t)dt \tag{2.19}$$

where t_{end} is the final experiment time. The final parameter calculations are

$$\tau = \frac{eA_1}{K} \tag{2.20}$$

$$\theta = \frac{A_0 - eA_1}{K} \tag{2.21}$$

The Method of areas is a great technique for developing a FOPDT system model from the step response. It has high accuracy in the presence of noise. One key feature is this method has the ability to estimate the dead time without defining the moment that the process variable begins to change. This method is not suitable for the limited-resource auto-tuning program in this project because its computational complexity and data storage needs.

Graphical Approach

In the Graphical Approach [16], the process gain is determined by dividing the steady state output by the input set-point value. The dead time is the length of time from the application of the control step to the time when the process variable begins to change due to the step. The time constant is the length of time it takes from when the process variable begins to change to when the output reaches 63% of the final value. Even though this method does not require immense resources, it requires storage of all data points, which the low-resource PLC in this project cannot do.

Two-Point Identification

The Two-Point Identification [16] approach is similar to the Graphical Approach as it calculates the steady-state gain in the same manner. The dead time and time constant are a function of the time taken for the process output to reach 28% and 63% of the final steady state output, less the dead time. These characteristics are found solving these equations:

$$T_{63} = \theta + \tau \tag{2.22}$$

$$T_{28} = \theta + \frac{\tau}{3} \tag{2.23}$$

Like the Graphical Approach, this method requires full data storage, which disqualifies it for use in this project.

Auto-tuning Program's Approach

In the Graphical Approach, the methods for calculating process gain and dead time are ideal for a low-resource auto-tuning program, as they can be calculated as a streaming algorithm. The approach taken by the auto-tuning program uses the

process gain and dead time methods from the Graphical Approach, but calculates the time constant differently. To calculate the time constant, the initial slope of the change in the process variable is extrapolated out to the steady state value. The time intercept is the time constant, as shown in Figure 2.11. This can be implemented as a streaming algorithm which allows it for use in the auto-tuning program for this project.

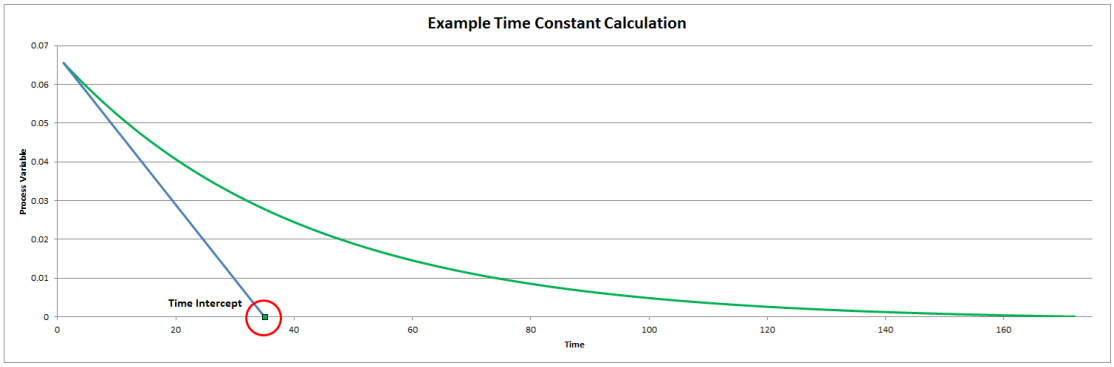


Figure 2.11: Calculating the Time Constant with Linear Extrapolation

2.5.5 PID Tuning Rules

Figure 2.7 defines PID tuning rules as the way by which the plant approximation is mapped to PID gain coefficients. Many systematic PID tuning rules were proposed based on the information from either a step response or frequency response [17] in order to reduce the time of tuning and to optimize performance of the system. PID tuning rules translate a system's process gain, time constant, and dead time into these gain coefficients for the Step Response Method. Many PID tuning rules exist in the literature, each optimizing a differnt cost function from the rest. Many PID tuning rules have been compared for chemical processes in [12, 13, 18, 19]. This thesis examines three different tuning rules designed for FOPDT systems with $\frac{\theta}{\tau} > 1$, which describes the CB Water System.

Dimensional-reduction (DR) Method[20]

Like most PI tuning rules, the controller gain, K_c, and reset time, T_i, are functions of the process dead time, time constant, and gain. This tuning method uses dimensional analysis and numerical optimization techniques to solve PI gain coefficients for FOPDT systems. This tuning rule was formed by considering the integral square error, integral absolute error, and the integral time absolute error performance creiteria. Using dimensional analysis, the tuning rule is able to reduce the number of parameters involved, which eliminates extraneous information[21]. This method is designed for systems with long dead time. Table 2.2 below describes the DR tuning rule.

Method	K_c	T_i
DR	$\frac{0.4849\tau}{\theta K} + \frac{0.3047}{K}$	$0.4262\theta + 0.9581\tau$

Table 2.2: DR Tuning Rule

Haalman's Method[22]

This tuning method was developed by Haalman in 1965. This method accurately accounts for systems with long dead time. This method suggests an ideal loop transfer function $G_I(s) = PC$ where P is the process, and C is the controller. Haalman's method chooses Equation 2.24 for $G_I(s)$

$$G_I(s) = \frac{2}{3Ls}e^{-sL} \tag{2.24}$$

which only accounts for the dead time, as process gains and poles are canceled. When applying this method to a FOPDT process, solving for the controller this leads to Equation 2.25.

$$C(s) = \frac{2(1+sT)}{3KLs} \tag{2.25}$$

This tuning rule lends itself to the following PI tuning rules described by Table 2.3 below.

Method	K_c	T_i
Haalman	$\frac{2\tau}{3K\theta}$	τ

Table 2.3: Haalman Tuning Rule

Internal Model Control (IMC) [23]

This PID tuning method is fundamental in PID tuning. The IMC-PI rule uses a zeroth-order Padè approximation, which neglects dead time. This gave forth to the PID tuning rules described in Table 2.4. This led to satisfactory results, and could not be used in systems with significant dead time.

Method	K_c	T_i
IMC	$\frac{\tau}{\lambda}$	τ

Table 2.4: IMC-PI Tuning Rule

The IMC-PI tuning rule was remedied in the "Improved PI rule" [24]. This rule is referred to as the IMC tuning rule in this thesis. This specific tuning rule is designed for systems with $\frac{\theta}{\tau}$ ratios greater than 1 by adjusting the additional parameter, λ

$$\lambda > 1.7\theta \tag{2.26}$$

This tuning rule is described as follows in Table 2.5.

Method	K_c	T_i
IMC	$\frac{\tau+0.5\theta}{K\lambda}$	$\tau + 0.5\theta$

Table 2.5: Improved-PI Tuning Rule

PID Tuning Rules Summary

The PID tuning rules are summarized in Table 2.6 below, where the proportional gain is $K_p = K_c$ and the integral gain is $K_i = \frac{K_c}{T_i}$. There exist many other PID tuning rules in the literature; the ones mentioned here are used specifically in systems similar to the CB Water System.

Method	K_c	T_i
IMC	$\frac{\tau+0.5\theta}{K\lambda}$	$\tau + 0.5\theta$
DR	$\frac{0.4849\tau}{\theta K} + \frac{0.3047}{K}$	$0.4262\theta + 0.9581\tau$
Haalman	$\frac{2\tau}{3K\theta}$	τ

Table 2.6: Summary of PID Tuning Rules

2.6 Change Detection

Change detection is determining the point in data where the data begins to change after being a steady value. This auto-tuning program uses a individuals control chart [25] to determine when the process variable deviates from steady-state and begins to change.

2.6.1 Individuals Control Chart

An individuals control chart (i-chart) is an application of Statistical Process Control (SPC). It arises from Walter A. Shewhart's [2] X-bar and R charts to detect when an assignable cause occurred in a continuous process variable. In an i-chart, a center-line is the reference point of the chart, which in this application is the mean of the observed data. Action lines are placed three i-chart standard deviations from the center-line. Points outside of these action lines are considered to be out of control, or transient. The auto-tuning program will use this to determine points where steady data begins to change. Figure 2.12 gives an example of an individuals control chart with upper and lower control limits for an output.

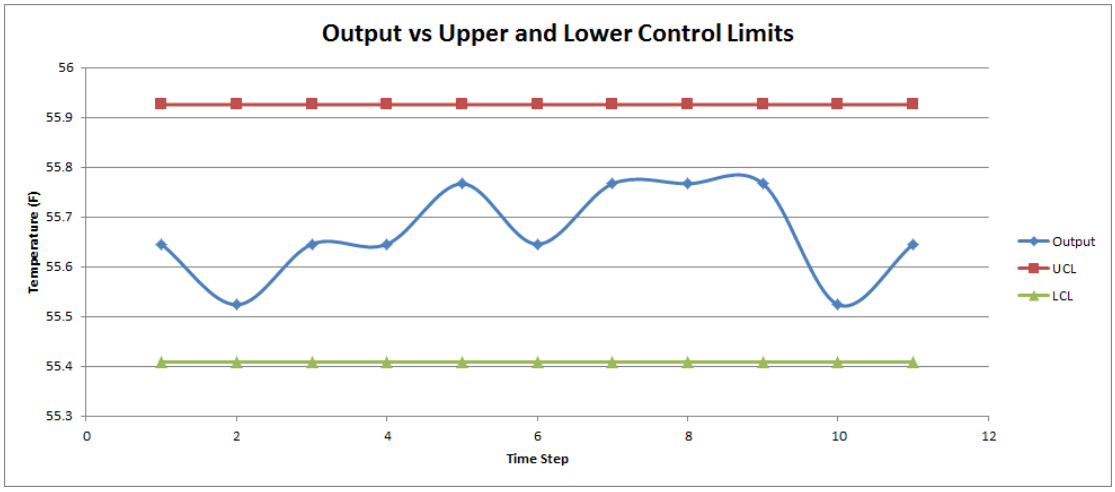


Figure 2.12: Example of individuals control chart

The i-chart standard deviation used to calculate the action lines are described by Equation 2.27

$$\sigma = \frac{\bar{R}}{d_n} \tag{2.27}$$

where $\bar{R}$ is the mean moving range, and d_n is a statistical coefficient. The moving range is the difference between adjacent data points. These measurements are

then used to determine the action lines, shown in Equation 2.28.

$$ActionLines : \bar{X} \pm \frac{3\bar{R}}{d_n} \tag{2.28}$$

For the statistical coefficient d_n, the range will always be between adjacent data points making $n = 2$. The statistical coefficient will then be $d(2) = 1.128$ [25], which makes the actions lines simply:

$$ActionLines : \bar{X} \pm 2.66\bar{R}$$

Use in Auto-tuning Program

This auto-tuning program uses the i-chart to determine when the CB Supply Water temperature has begun to change after holding a steady value. The traditional i-chart described above is offline. It first collects data points, calculates action lines for the entire data set, then determines which points are outside the action lines, as shown in Figure 2.13. The proposed auto-tuning program takes the basic structure of the i-chart and modifies it to be an online algorithm. It becomes online by calculating a new center-line and action line at each time step, which is used to determine if the CB Supply Water temperature has started to change, as shown in Figure 2.13. The auto-tuning program will discuss the use of this in greater detail in Chapter 4.

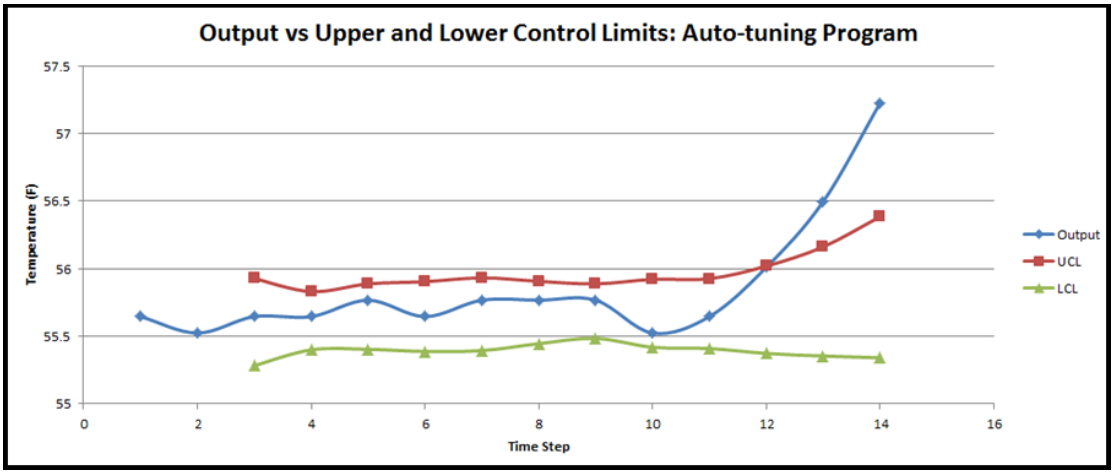


Figure 2.13: Example of auto-tuning program use of individuals control chart

2.7 Closed-Loop Performance Metrics

To evaluate the process output response to a step in the control output, three quantities are determined [4]:

1. **Rise Time** t_r

   Rise time is the amount of time it takes the process output to reach the vicinity of the new final steady-state output value due to the step in the control output. This project more specifically defines the rise time as the time it takes the process output to go from 10% of the change of the final steady-state output to 90% of the change in the final steady-state output.

2. **Settling Time** t_s

   The settling time is the amount of time it takes the process output transients to stabilize to the final steady-state output. This is quantitatively defined as the time it takes the output to be bounded by $\pm$ 1% of the final steady-state output.

3. **Overshoot** M_p

Overshoot is the maximum amount the process output overshoots its final steady-state value divided by the final steady-state value.

Chapter 3

Equipment and Techniques for Experimentation

This chapter describes the steps taken to run the auto-tuning program on the CB Water System in this case study. Figure 3.1 describes the CB Water System running under normal operation.

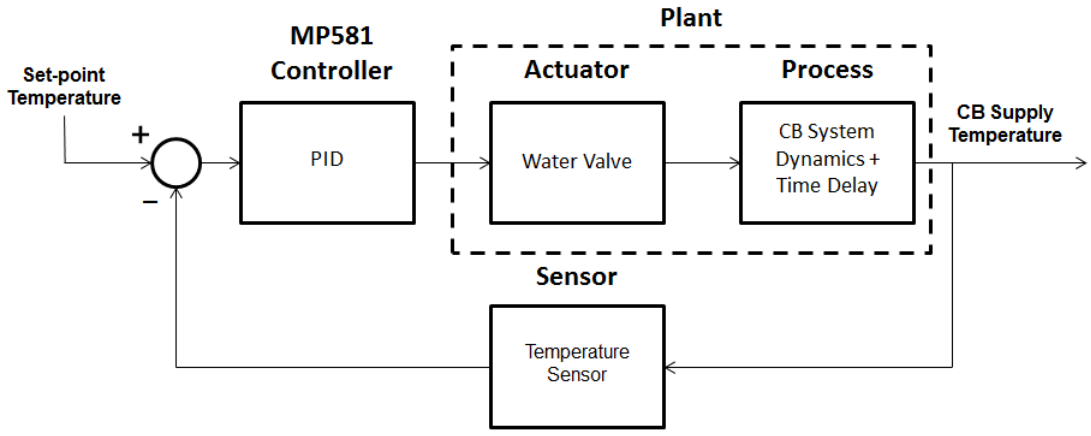


Figure 3.1: System Diagram

The set-point temperature for the CB Supply Water temperature does not change, staying constant at 58°F. PID control is housed within the Tracer MP581

Programmable Controller. The water valve of interest in this system is the Bypass Valve, which commands the CB Supply Water temperature. With the set-point staying constant, the only disturbances this system will experience are from the water sources making up the CB New Water, shown in Figure 1.1.

3.1 Tracer MP581 Programmable Controller

The Tracer MP581 Programmable Controller controls multiple control loops, including those in the CB Water System. This programmable controller runs on a Motorola MC68332 20 MHz microprocessor with 512KB RAM, 2MB Flash ROM, and 256KB EEPROM. On its own this programmable controller has twelve universal inputs, six binary outputs, and six analog outputs; however it can be connected to expansion modules to increase these numbers. The analog inputs can take the form of dry contact binary, 0-20mA, 0-10 Vdc, linear resistance, or thermistor. The analog outputs can output 0-10 Vdc or 0-20 mA.

3.2 Collecting Data with the NI myRIO

To observe system behavior, input and output data from the CB Water System must be collected. Trane has developed a web-based systems integration solution, known as the Tracer ES, which stores the data from analog inputs and outputs connected to the Tracer MP581 Programmable Controller. Data is recorded to the Tracer ES on fifteen minute intervals. To better observe the CB Water System, finer resolution data is needed. This led to the development of a device to record information. More information on the Tracer ES is provided in Section 3.4.

This project uses National Instrument's (NI) myRIO to collect data. The NI myRIO is an embedded hardware device with many real-time processing capabilities, which run LabVIEW. Wires from the NI myRIO are directly connected to the Tracer MP581 Programmable Controller terminals. The NI myRIO was programmed to collect data on one-second intervals, more than adequate for observing CB Water System behavior.

Not only is the ability to collect data necessary, the ability to output an analog signal is required. The NI myRIO has the capability to carry out this task as well. Outputting an analog signal is required to command analog outputs, such as valve position, which is necessary for this project.

The NI myRIO on its own has two analog inputs and two analog outputs. For experimentation, many more analog inputs were desired. There are two expansion ports on the NI myRIO capable of adding eight additional analog inputs and four additional analog outputs. A printed circuit board (PCB), known in this project as the Breakout Board, was built to gain access to the additional analog inputs and outputs.

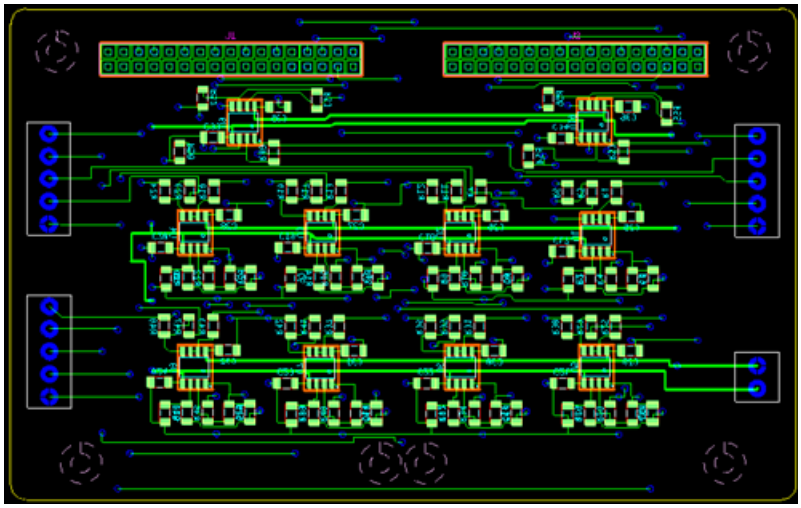


Figure 3.2: Layout of PCB connected to myRIO expansion ports

The Breakout Board not only gives access to the additional analog inputs and outputs on the expansion ports, it runs the inputs through an anti-aliasing filter, which gives higher fidelity data. After connecting this tool to the Tracer MP581 Programmable Controller, full access to reading the analog inputs, and writing to the analog outputs were achieved. The experimental setup is described in Figure 3.3. This equipment allowed for the CB Water System to be observed and tested.

Figure 3.3: Experimental Setup

3.3 Configuring the Tracer MP581 Programmable Controller

To connect and program on the Tracer MP581 Programmable Controller, the network uses the LonTalk protocol. USB-enabled computers can connect to the Tracer MP581 Programmable Controller through LonTalk using the Echelon U10 Network Interface USB device. The Trane Rover Service Tool software can then be run on a laptop, to configure the PLC. The different programs running on the PLC can be view and edited from this software. This is where PID gain coefficients can be changed, and an auto-tuning program placed in the Tracer MP581 Programmable Controller. Also, the software shows the analog input and output values in real-time.

3.4 Tracer ES

The Tracer ES is an application used by the building management system to allow users to access information from a building from any internet-capable device on the network. This has many uses for a building operator, but for a researcher this is used to obtain data from sensors within a building. One particular use of the Tracer ES in this project was to examine historical data. By looking at historical data, the Bypass Valve was identified as a misbehaving valve, and deemed to be a high-priority candidate for this research.

The Tracer ES is collecting and storing data at all times, which is a monumental advantage compared to the NI myRIO, which only records data during experimentation.

Chapter 4

Auto-tuning PID Gain Coefficients

Automatic tuning is a technique used to automate the adjustment of PID gain coefficients. It is generally referred to as on-demand tuning, one-shot tuning, or auto-tuning. In the process control industry, there can be thousands, if not millions of PID control loops in a given system. It is impossible to have an engineer systematically tune each of these loops to obtain a desired performance. It is common for technicians to manually tune these controllers, or set them to default settings. Manual-tuning consists of a technician adjusting each of the individual controller gains and seeing how the system responds. This is done until the process seems good to the technician or he or she runs out of time. The later is too often the case. Automatic tuning allows for desired performance to be achieved without an engineer calculating the gains. Automatic tuning can be built into a controller or built into an external device connected to a control loop. In this research, re-tuning is needed when the dead time changes. A diagram of

the implementation of the auto-tuning program into the control system for this case study is shown in Figure 4.1.

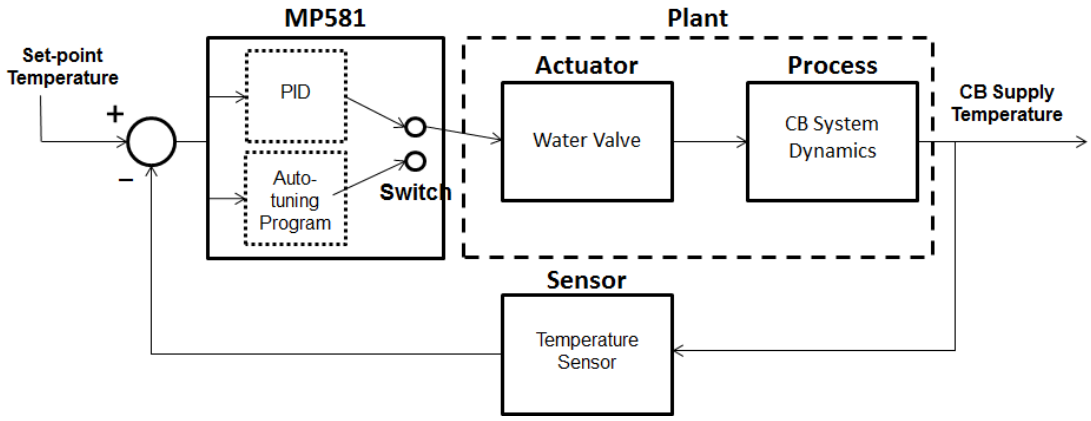


Figure 4.1: System Diagram with Auto-tuning Program

The auto-tuning program developed in this thesis was built in the Tracer Graphical Programming (TGP) language, which is installed on the Tracer MP581 Programmable Controller. This PLC has the limitation of 120 analog and binary variables, and a 60KB limit on the size of each program. Many memory and resource constraints were therefore taken into account during the design of this auto-tuning program. Most notably, the auto-tuning program takes advantage of a low-resource technique for calculating standard deviation.

4.1 Welford's Method for Calculating Corrected Sum of Squares

The auto-tuning program uses standard deviation in its calculations. This measurement represents how dispersed the data are from each other. If the sample $x_1, ...x_N$ is given, the variance, s^2, is defined as:

$$s^2 = \frac{1}{n(n-1)}(n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2) \tag{4.1}$$

where x is the data point, n is the number of points, and s is the standard deviation. To run the textbook definition of variance, it is required for the program to run two passes through the stored data. Not only does this require a significant amount of memory to store all the data, it requires a significant amount of computation to do two passes of the data.

This project uses a streaming algorithm to compute the standard deviation in one pass. Because it is a streaming algorithm, not only does it soley do one pass of the data, it also does not require the data to be stored. This algorithm was developed by B.P. Welford [3] to calculate the corrected sum of squares. The recurrence formulas are described Equations 4.2, 4.3 for this algorithm[26],

$$M_k = M_{k-1} + (x_k - M_{k-1})/k \tag{4.2}$$

$$S_k = S_{k-1} + (x_k - M_{k-1})(x_k - M_k) \tag{4.3}$$

with initial conditions $M_1 = x_1, S_1 = 0$. From this, the variance is calculated as $s^2 = \frac{S_k}{k-1}$, and then the standard deviation can be computed.

It is common in PID auto-tuning programs to run a relay or step experiment, store all the data, then do the analysis offline to find the PID gain coefficients. The Tracer MP581 Programmable Controller does not have the memory capacity to run in such a way. The auto-tuning program in this thesis performs its analysis of the step response on-line, without storing any data. This greatly reduces the memory load and allows the Tracer MP581 Programmable Controller the ability to

have an auto-tuning program. This method was implemented in the auto-tuning program as illustrated by the TGP code in Figure 4.2.

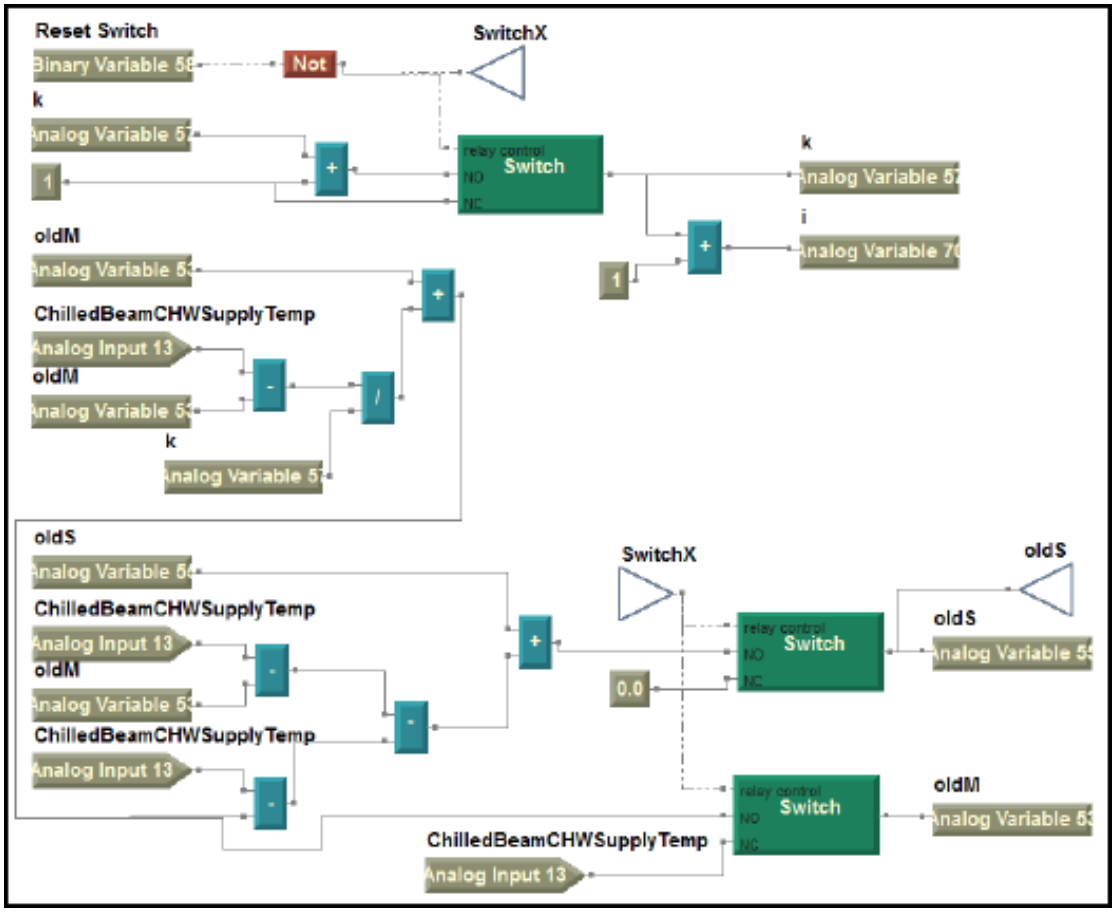


Figure 4.2: Streaming Standard Deviation in Auto-tuning Program

4.2 Current Program in Tracer MP581 Programmable Controller

The current PID control program commanding the Bypass Valve is pictured in Figure 4.3. This discrete time program operates on ten second time intervals. Like many industrial PID controllers, the current PID gain coefficients, shown in Table 4.1, have not been changed since the controller's initial commissioning. To compensate for this, the engineer designing the program added a feature to

allow the Bypass Valve ability to only open 0.05% every time step. There is no constraint on how much the valve can close per time step. This has led to poor tuning of the Bypass Valve.

Parameter	Value
Proportional	2
T_i Integral	0.5
T_d Derivative	0

Table 4.1: Current PID Gain Coefficients

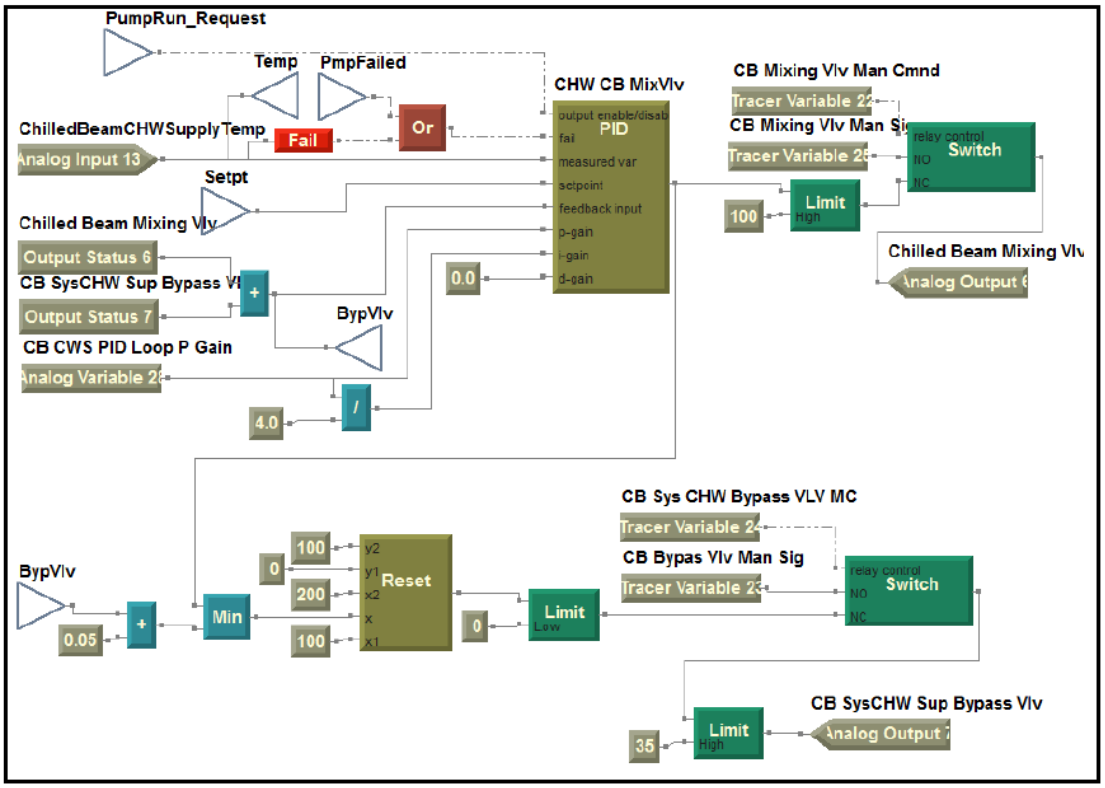


Figure 4.3: Bypass Valve Control Program

4.3 Methodology

This auto-tuning program utilizes the Step Response Method, which is a model-based systematic tuning approach, discussed in Section 2.5.3; therefore, the auto-tuning program will follow the steps outlined in Figure 2.7 to obtain PID gain coefficients. This section will describe how the auto-tuning program realizes each of these steps.

4.3.1 Bypass Valve PID Control Loop Program

The auto-tuning program is separate from the program running the Bypass Valve PID control loop. To implement the auto-tuning program, the Bypass Valve PID control loop program remains essentially intact, with the simple addition of a switch. This switch decides whether the PID controller is controlling the Bypass Valve, or the auto-tuning program. A switch, controlled by the building operator, determines what controls the Bypass Valve.

4.3.2 Testing the Physical System

A step is applied to the Bypass Valve, and the change in the CB Supply Water temperature is observed. This is done to get input and output data for the plant identification method to generate a plant approximation. This occurs by breaking the PID control loop and gives the auto-tuning program control of the Bypass Valve. The auto-tuning program holds the Bypass Valve in its position prior to the program taking control of the valve. The auto-tuning program then holds the Bypass Valve at this position for a predetermined length of time, set by the user. This time must at least exceed the expected system dead time. After the

time has expired, the auto-tuning program will step the Bypass Valve position, which is time t_0. The Bypass Valve will be stepped by a predetermined value set by the user, generally somewhere between 3 - 10% valve position. The initial valve position is u_{inital} and the stepped valve position is u_{final}. The auto-tuning program holds the Bypass Valve at u_{final} until the CB Water Supply Temperature has stopped changing due to the step in the valve position. How the auto-tuning program does this will be explained over the remainder of this chapter.

4.3.3 Plant Identification

Once the system has been stepped, the auto-tuning program begins searching for the time, t_{change}, in which the CB Supply Water temperature starts to change due to the step. This time is used by the plant identification method to obtain the plant approximation. From the time the Bypass Valve is stepped, the auto-tuning program begins analyzing the CB Supply Water temperature to determine if the temperature has started changing due to the step in the Bypass Valve. To accomplish this, upper and lower control limits are calculated, as discussed in Section 2.6. Figures 4.4, 4.5, 4.6 are parts of the auto-tuning program responsible for computing the upper and lower control limits. To compute the control limits, mean and mean-range need to be calculated. The auto-tuning program uses a streaming algorithm for computing mean, illustrated in the TGP language in Figure 4.4. This simple algorithm works by weighting the previous mean at the past time step at $\frac{n-1}{n}$, and adding that value to the current data point with a weighting of $\frac{1}{n}$. The mean-range computation is also a streaming algorithm, accomplished in the same fashion as the mean algorithm, but looking at range

values. This is shown by Figure 4.5. Figure 4.6 shows the upper and lower control limits being calculated in the TGP language.

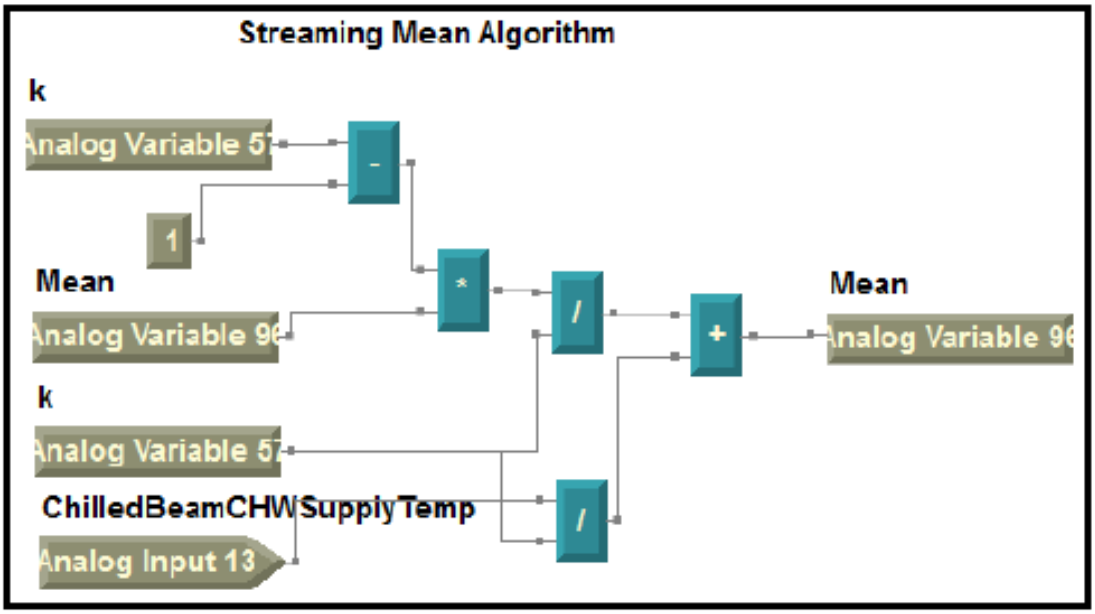


Figure 4.4: Streaming Mean Algorithm

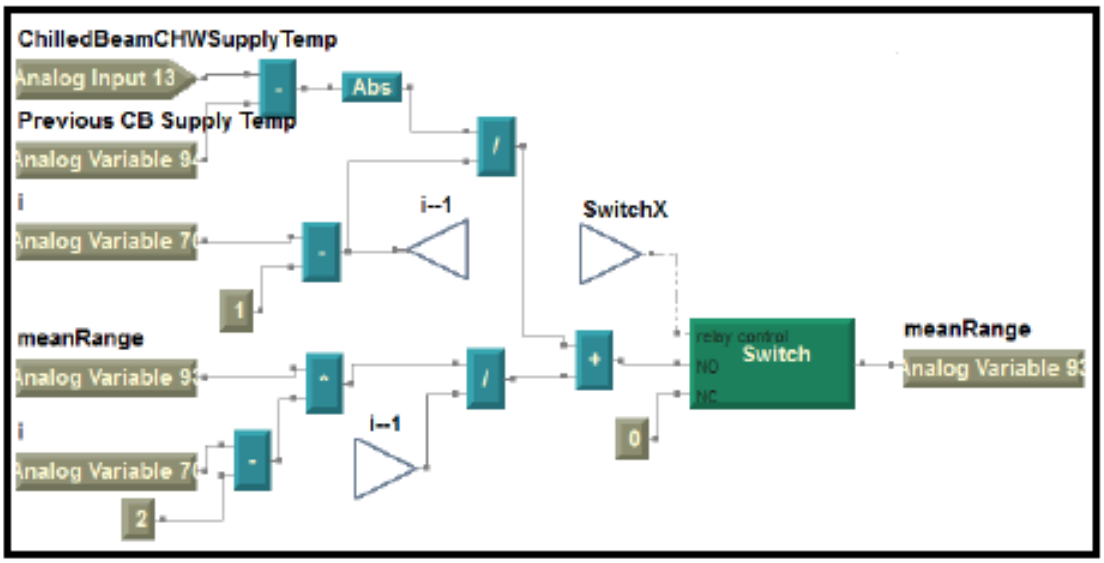


Figure 4.5: Streaming Mean-Range Algorithm

If the CB Supply Water temperature goes above the upper control limit or below the lower control limit, the auto-tuning program decides the CB Supply Water temperature has begun to change due to the step.

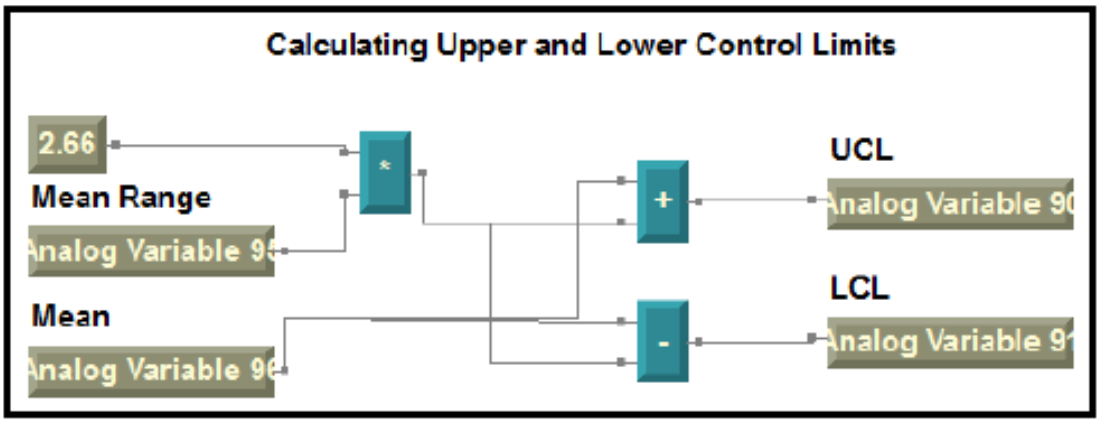


Figure 4.6: Computing Control Limits

While the auto-tuning program is looking for t_{change}, it simultaneously begins calculating a running evaluation of the standard deviation using the method described in Section 4.1 from time t_0 to the identified time t_{change}. This standard deviation, s_{ss}, is considered to be the standard deviation while the system is at steady-state.

Next, the auto-tuning program needs to identify the time the CB Supply Water temperature stops changing due to the step, time t_{final}. To do this, the auto-tuning program begins calculating a moving window for standard deviation, s_{mw}. When s_{mw} becomes less than or equal to s_{ss}, the CB Supply Water temperature has reached t_{final} and the temperature is again at steady-state.

Times t_{change} and t_{final}, along with the CB Supply Water temperatures at those times, $temp_{change}$ and $temp_{final}$, are recorded and used to identify a plant approximation of the CB Water System.

Process Dead Time Calculation

The process dead time, θ, is the time from when the Bypass Valve is stepped, to the time the CB Supply Water temperature begins to change, as shown in equation

4.4:

$$\theta = t_{change} - t_0 \tag{4.4}$$

Process Gain

The process gain is the ratio of the amount of change in CB Supply Water temperature to the amount of change in Bypass Valve position percentage, calculated by equation 4.5:

$$K = \frac{temp_{final} - temp_{change}}{u_{final} - u_{initial}} \tag{4.5}$$

Process Time Constant

The process time constant is the time at which the CB Supply Water temperature has reached 63% of its final value. Calculating this can be further described in Chapter 2 under Auto-tuning Program Approach, and is realized by solving Equation 4.6 below.

$$temp_{final} = \frac{temp_1 - temp_{change}}{\frac{1}{6}}\tau + temp_{change} \tag{4.6}$$

Here, $\frac{1}{6}$ represents the program time step, which is $\frac{1}{6}$ of a minute. The temperature $temp_1$ represents the first temperature recording after time t_{change}. Equation 4.7 shows the time constant calculation.

$$\tau = \frac{(temp_{final} - temp_{changepoint})(\frac{1}{6})}{temp_1 - temp_{change}} \tag{4.7}$$

With these three plant approximation parameters calculated, the auto-tuning program can proceed determining PID gain coefficients.

4.3.4 PID Gain Coefficients

With the plant approximation calculated, the auto-tuning program can now identify PID gain coefficients using one of the PID tuning rules described in Section 2.5.5. After the PID gain coefficients are calculated, the auto-tuning program releases control of the Bypass Valve, handing it back to the PID controller.

Chapter 5

Results and Analysis

This chapter will describe the ability of the auto-tuning program to accurately determine PID gain coefficients to control the Bypass Valve. The different plant identification techniques described in Section 2.5.4 will be carried out on the CB Water System. These plant approximations will then be used with different PID tuning rules to obtain PID gain coefficients.

5.1 Plant Approximation

Plant identification techniques are used to obtain a plant approximation. In this section, the different plant identification techniques will be compared. The comparison will be carried out comparing the plant approximations obtained from each technique. The same data-set has been applied to each technique, and is described in Figure 5.1.

The data-set shows the CB Supply Water temperature response to a 4.5% step in the Bypass Valve position. The CB Supply Water temperature begins to stabilize at about 57.5 °F, and following the step settles at about 54 °F. The plant

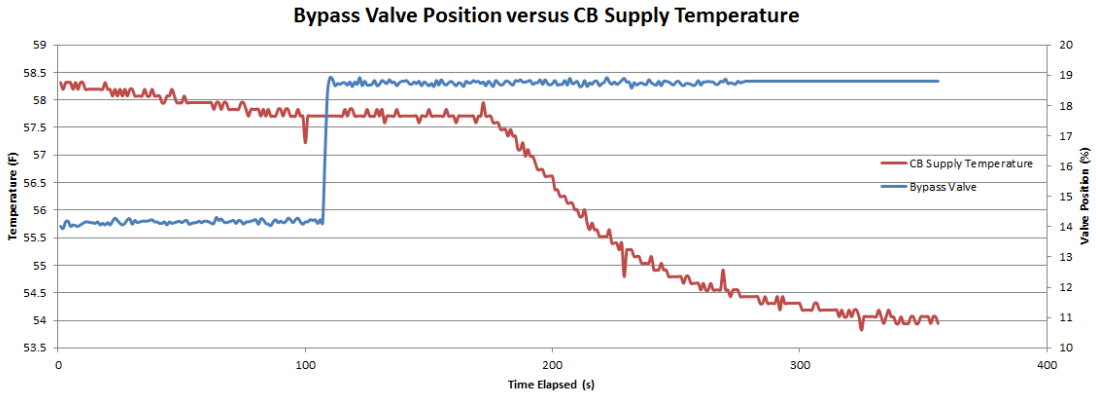


Figure 5.1: CB Water System Response to Step in the Bypass Valve

approximation obtained from each of the different plant identification techniques is summarized in Table 5.1.

Method	Gain (K)	Time Constant (min) τ	Dead Time (min) θ
Graphical Approach	0.8093	0.8000	1.1500
Auto-tuning Program	0.8093	0.9000	1.1500
Two-Point Identification	0.8093	0.6750	1.2750
Method of Areas	0.8093	0.6521	1.2485

Table 5.1: Plant Approximations from the Different Plant Identification Techniques

Each of the the plant approximations obtained by each technique is compared against one another in simulation. This is performed by first adding the plant approximation to the Simulink simulation. In the simulation an open-loop step is applied to the plant approximation, as shown in Figure 5.2. This is the same 4.5% step applied to the physical system. The plant approximation response to this step is observed. This is carried out on each plant approximation, and the responses are compared. These responses are not only compared to each other, but to the actual CB Supply Water temperature response from Figure 5.1. The

results of the Simulink simulation with comparison to the experimental data are shown in Figure 5.3.

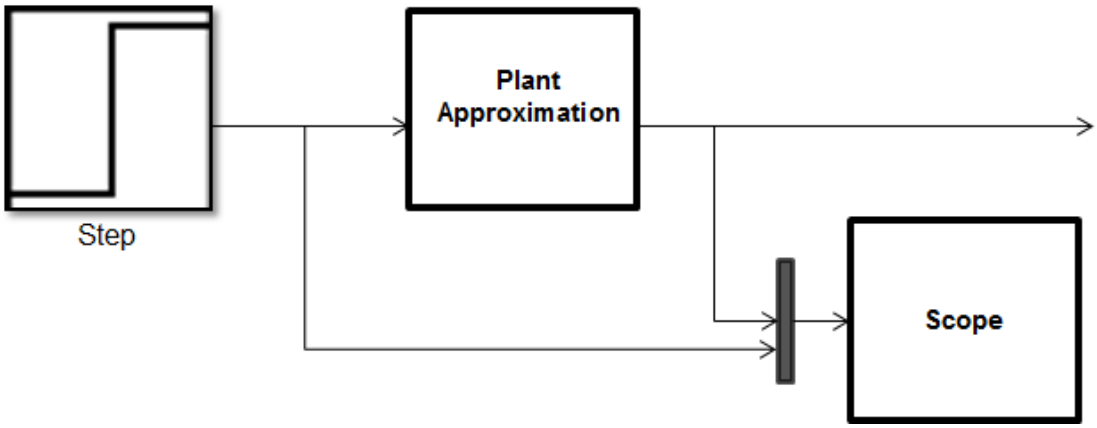


Figure 5.2: Open Loop Step Response Simulation Set-Up

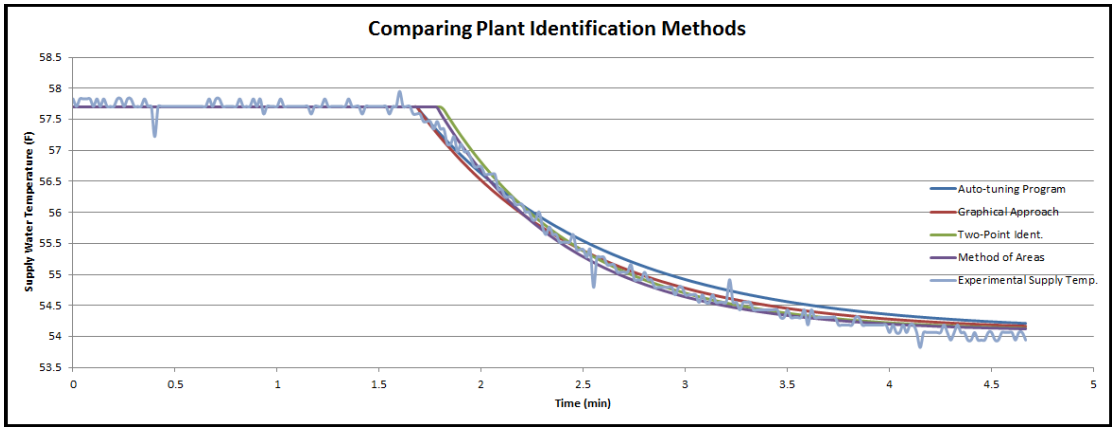


Figure 5.3: Comparison of Step Response

The results in Figure 5.3 show each of the methods describe the system with high fidelity. From visual inspection, it is difficult to tell which system is best. To compare the different plant identification methods further, the Time Domain Identification Error, ϵ , is used and explained in Equation 5.1. The error of each of the methods is summarized in Table 5.2.

$$\epsilon = \frac{1}{n}\sum_{k=0}^{n-1}[y(kT) - \hat{y}(kT)]^2 \tag{5.1}$$

Method	ϵ
Graphical Approach	0.0191
Auto-tuning Program	0.0387
Two-Point Identification	0.0161
Method of Areas	0.0144

Table 5.2: Time Domain Identification Error

From this metric a clearer picture is shown of each method's performance. The most complex method, Method of Areas, outperformed the other three methods for model fitting with a value of 0.0144 for the time domain identification error. The most error came from the Auto-tuning Program Approach, with a value of 0.0387 for the time domain identification error; however, the advantage of the Auto-tuning Program Approach is in its ability to use fewer resources. By inspection, it seems that all methods have a minimal amount of error. To examine the effects of the slight difference in error, Section 5.2 will compare the PID gain coefficients obtained from the different plant identification techniques.

5.2 Comparing Plant Identification Techniques and PID Tuning Rules

The plant approximations found in Section 5.1 will be used with the different PID tuning rules, discussed in Section 2.5.5, to calculate different PID gain coefficients. The PID gain coefficients realized are summarized below in Table 5.3.

To compare the four plant identification techniques and three PID tuning rules,

Plant Identification Method	Tuning Rule	Proportional (K_p)	Integral (K_i)
Method of Areas	Haalman	0.4303	0.2199
Method of Areas	IMC	0.4679	0.1222
Method of Areas	DR	0.6894	0.1986
Two-Point Identification	Haalman	0.4361	0.2154
Two-Point Identification	IMC	0.4711	0.1196
Two-Point Identification	DR	0.6937	0.1943
Graphical Approach	Haalman	0.5730	0.2388
Graphical Approach	IMC	0.5472	0.1327
Graphical Approach	DR	0.7933	0.2104
A-T Program	Haalman	0.6447	0.2388
A-T Program	IMC	0.5870	0.1327
A-T Program	DR	0.8454	0.2084

Table 5.3: PID Gain Coefficient Comparison

the PID gain coefficients obtained from each combination of plant identification technique and PID tuning rule will be compared, leading to twelve different PID gain coefficients. To do this comparison, it would seem logical to set the Bypass Valve PID controller to each of the different PID gain coefficients, and compare the response of the CB Supply Water temperatures to the set-point. This approach will not lead to fair comparisons, because the building load is constantly changing. There is no way of creating the same circumstances for each trial. This comparison must be done in simulation.

5.2.1 Simulation Model in Simulink

To compare the different PID gain coefficients, a simulation of the CB Water System has been created in the Simulink. This simulation is of a closed loop system containing a PID controller and plant model, as in the CB Water System. To obtain the plant model while under PID control, data from a disturbance was observed for the CB Supply Water temperature, shown in Figure 5.4. In this data sample, the CB Supply Water temperature is disturbed to 59.89°F, which is 1.89°F above the set-point, 58°F. The PID controller recovers from this disturbance as shown.

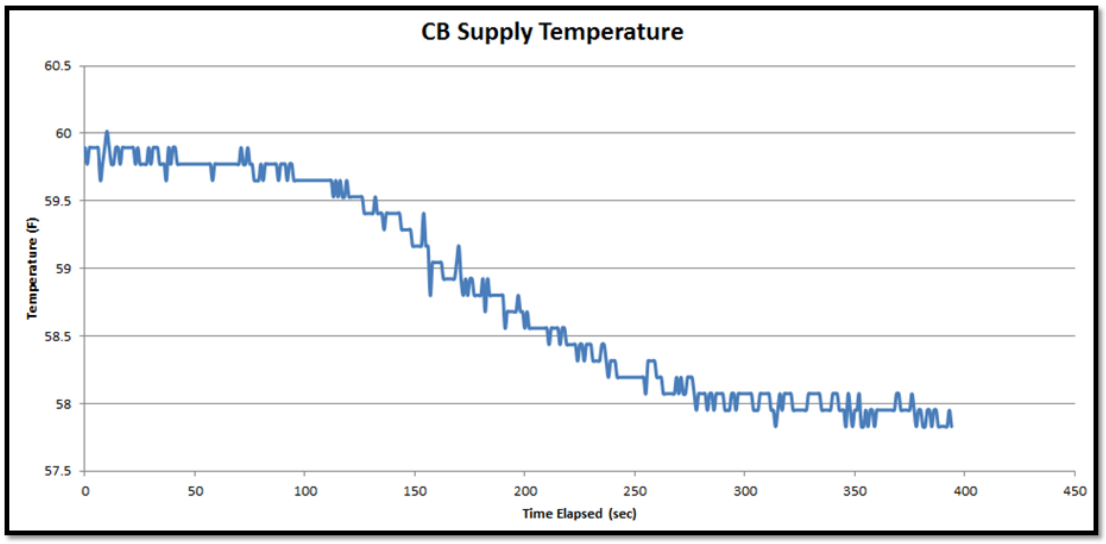


Figure 5.4: Disturbance in CB Supply Temperature

To obtain a good system model in Simulink, this disturbance was replicated with the same PID gain coefficients used by the physical system. The resulting simulation output is described in Figure 5.5. As shown, the simulation approximates the real data with high accuracy. The mean-squared error is 0.0034 for the simulation curve to the real data. Therefore, this simulation model can be used to compare different PID gain coefficients generated from the different plant

identification techniques and PID tuning rules. This simulation will be used in the rest of the chapter to compare PID gain coefficients.

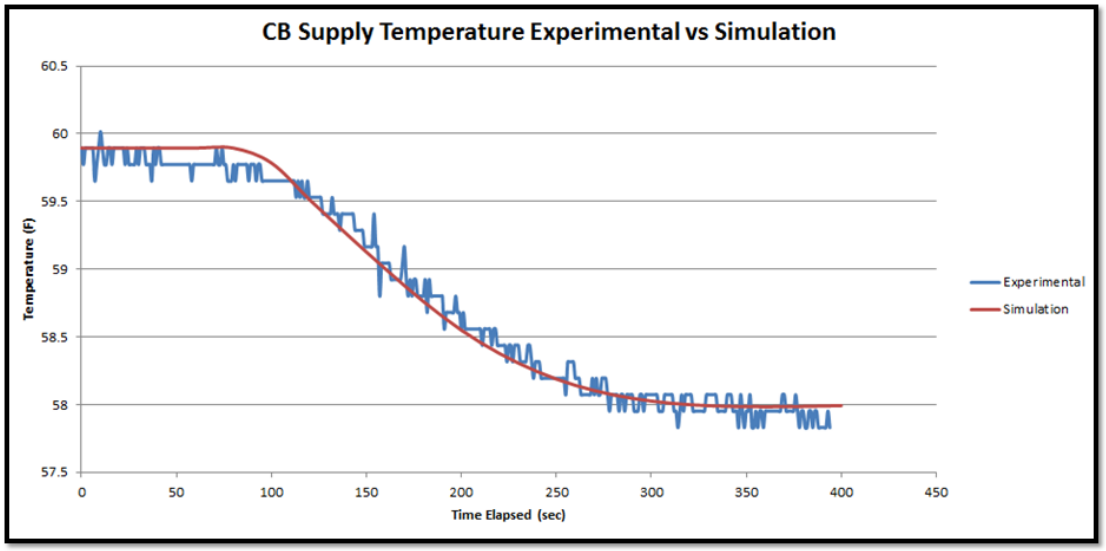


Figure 5.5: Experimental vs Simulation Disturbance Response

5.2.2 Comparing the Twelve PID Gain Coefficients

To compare PID gain coefficients, the same disturbance in Figure 5.4 is applied to each Simulink simulation, and the response of the simulated CB Supply Water temperature is observed. This simulation is done for each PID gain coefficient in Table 5.3. The results of the simulation are shown in Figures 5.6, 5.7, 5.8, 5.9.

5.3 Analysis of the Comparisons

To compare each of the results from Figures 5.6, 5.7, 5.8, 5.9, the following closed-loop response metrics are calculated: rise time (t_r), overshoot (M_p), and settling time (t_s), each described in Section 2.7. Table 5.4 examines these closed-loop response metrics for the twelve trials in Section 5.2.2.

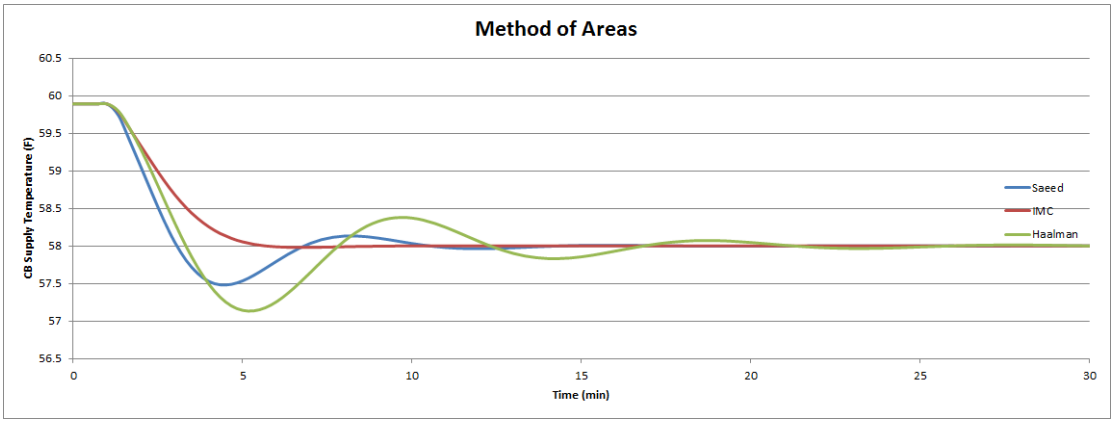


Figure 5.6: System Response to Disturbance using Method of Areas Plant Approximation with Different PID Tuning Rules

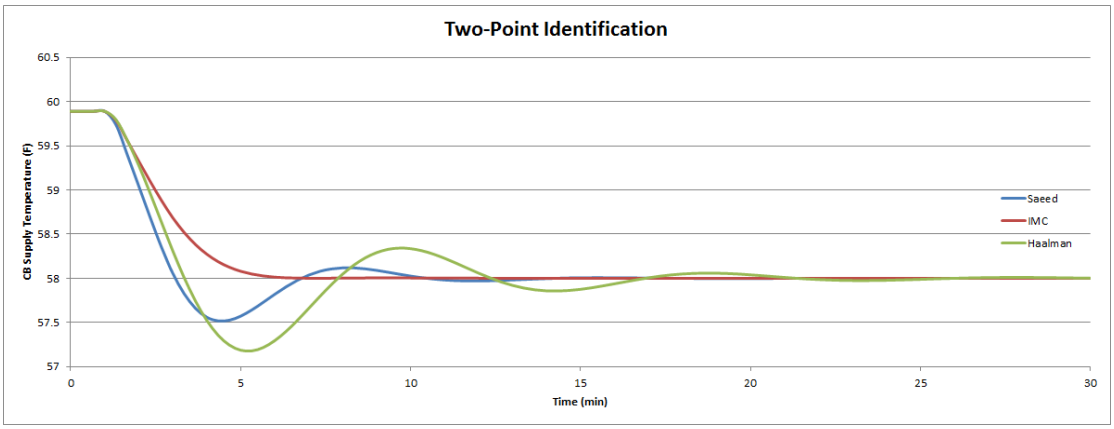


Figure 5.7: System Response to Disturbance using Two-Point Identification Plant Approximation with Different PID Tuning Rules

The results from the simulated CB Supply Water temperature response to the disturbance clearly illustrate the best-performing PID tuning rule for this system. When comparing rise time, DR and Haalman tuning rules outperform the IMC tuning rule. To achieve this rise time, both overshoot and settling time are sacrificed for these two PID tuning rules. The response that most closely resembles a critically damped system is the IMC tuning method for each of the different plant identification techniques. This tuning rule shows little to no overshoot, with the quickest settling time and a competitive rise time. Haalman's tuning rule is

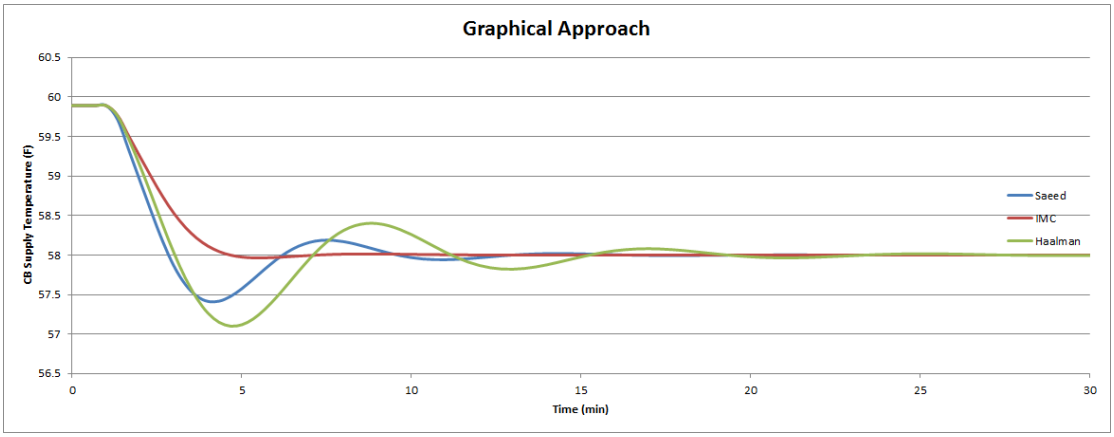


Figure 5.8: System Response to Disturbance using Graphical Approach Plant Approximation with Different PID Tuning Rules

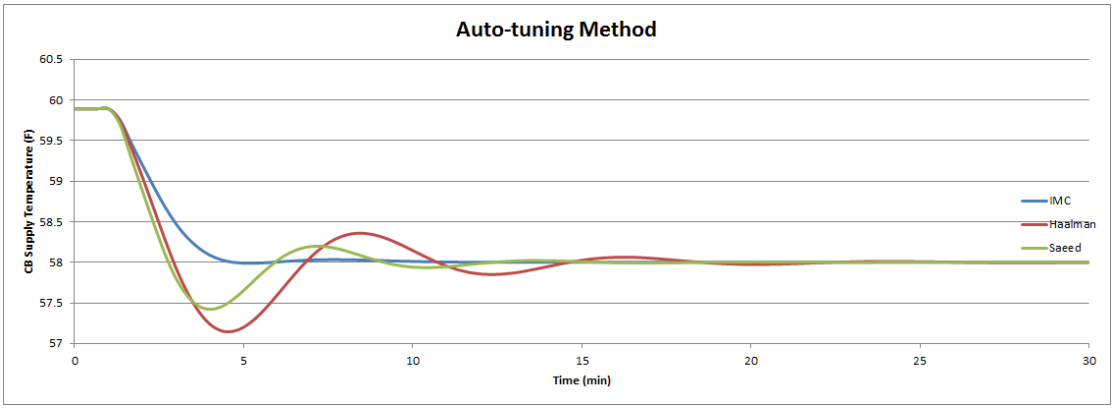


Figure 5.9: System Response to Disturbance using the Auto-tuning Program Approach Plant Approximation with Different PID Tuning Rules

furthest from critically damped, with the largest overshoot and longest settling time in each plant identification technique.

Comparing plant identification methods for the IMC tuning rule, all have similar results. The Graphical Approach and the auto-tuning program have the quickest rise times, at 2.23 and 2.29 minutes. The other two methods were close, with 2.79 and 2.88 minute rise times. The Two-Point Identification techniques has no overshoot, and is tied for the best settling time with the Graphical Approach at 6.0 minutes. No method had more than 0.028°F of overshoot.

Plant Identification	Tuning Rule	Rise Time	Overshoot	Settling Time
Method of Areas	Haalman	1.65	0.86	24.33
Method of Areas	IMC	2.79	0.018	6.67
Method of Areas	DR	1.47	0.52	10.33
Two-Point Ident.	Haalman	1.67	0.82	21
Two-Point Ident.	IMC	2.88	0.0	6.0
Two-Point Ident.	DR	1.49	0.48	10.33
Graphical Approach	Haalman	1.44	0.90	22.33
Graphical Approach	IMC	2.29	0.028	6.0
Graphical Approach	DR	1.31	0.58	10.33
Auto-tuning	Haalman	1.36	0.85	21.33
Auto-tuning	IMC	2.23	0.014	6.33
Auto-tuning	DR	1.26	0.58	14.33

Table 5.4: Closed-Loop Response Metrics

5.4 Variable Count

The Tracer MP581 Programmable Controller has a total 120 of both analog and binary variables available for the sum of all the programs loaded on the device. To compare the different plant identification techniques, the number of variables each require for computing the PID gain coefficients for the data-set in Figure 5.1 is examined. This comparison is shown in Table 5.5.

As seen, the auto-tuning program uses significantly less number of variables. This approach is able to conserve these variables because its ability to refrain from needing to store the data points. This is achieved through its use of streaming

Method	Variable Count
Graphical Approach	295
Auto-tuning Program	**27**
Two-Point Identification	296
Method of Areas	280

Table 5.5: Number of Variables used in each Plant Identification Technique

algorithms.

5.5 Robustness to Varying Dead Time

Dead time causes problems because the controller ends up waiting the amount of the dead time before it gets any feedback from the process. In many HVAC systems, the plant dynamics change because of a change in dead time. This holds true in the CB Water System, where the process dead time is a function of the flow rate of water flowing through any of the pipes in the system. In this section, the auto-tuning program is simulated with various dead times, to determine how robust the PID gain coefficients are to this change in plant dynamics.

Figures 5.10, 5.11, 5.12 show the auto-tuning program's response to the same disturbance in Section 5.2.2; however, in this section the dead time is varied. The dead time is varied between 0 minutes to 1 minute 45 seconds, in 15 second intervals. The figures also include the dead time at 1 minute 12 seconds, the actual process dead time.

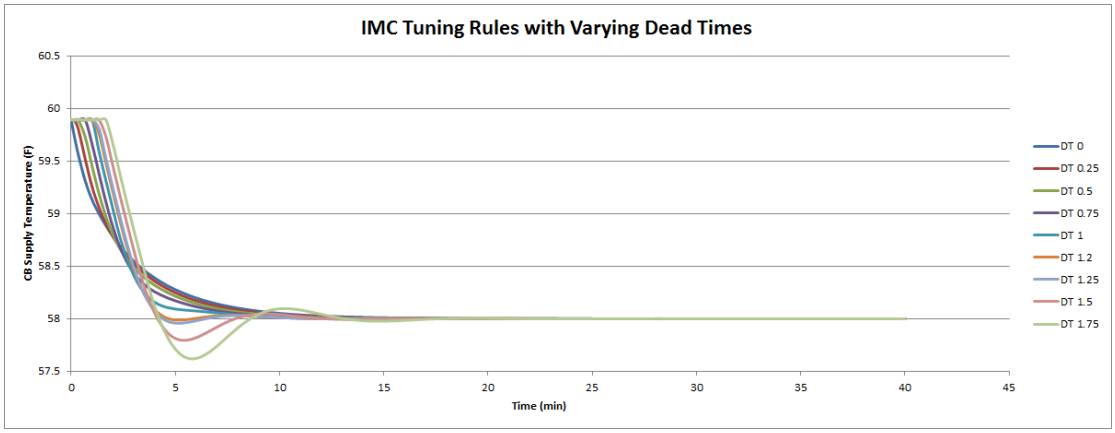


Figure 5.10: Auto-tuning Program Response to Disturbance with IMC Tuning Rules and Dead Time Varying

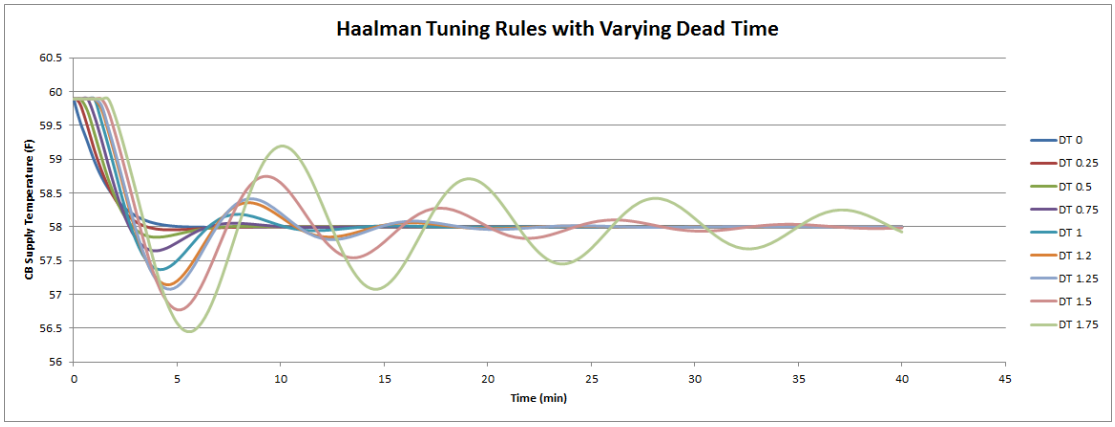


Figure 5.11: Auto-tuning Program Response to Disturbance with Haalman Tuning Rules and Dead Time Varying

5.5.1 Analysis

The dead time has a significant impact on the CB Supply Water temperature response to the disturbance. The trend of the graphs show that increasing dead time will lead to system instability. A decrease in dead time will not cause a system to go unstable, but eventually will lead to an over-damped response. The controller robustness to varying dead time is described by the following closed-loop performance metrics.

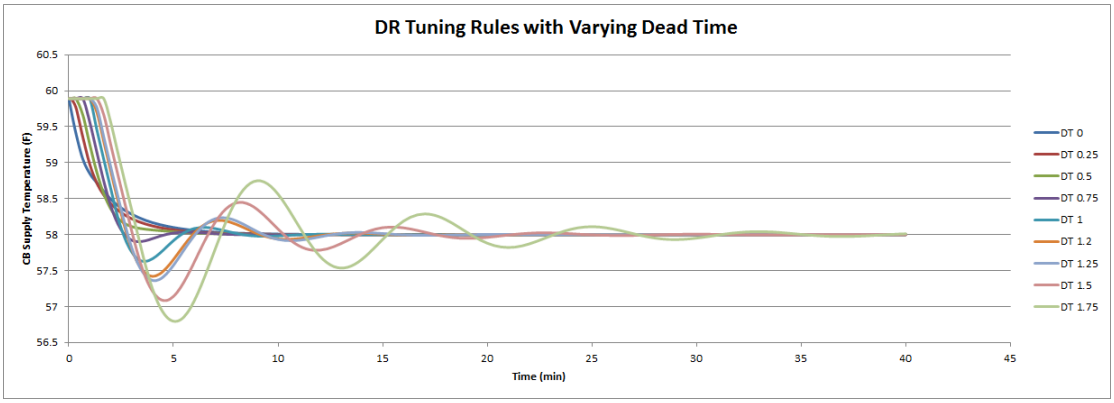


Figure 5.12: Auto-tuning Program Response to Disturbance with DR Tuning Rules and Dead Time Varying

Rise Time

Common to all trials above, as dead time becomes greater, the the rise time becomes less. For each tuning rule, the trial with dead time of one minute forty-five seconds has the shortest rise time. Most responses start over-damped, and with increasing dead time eventually becomes under-damped. For the auto-tuning program with IMC tuning rules, the response is over-damped for up to one minute of dead time. At one minute twelve seconds the system then becomes slightly under-damped, but closer to critical damping. With Haalman tuning rules, the system is under-damped for all trials, with the case of no dead time being closest to critical damping. DR tuning rules are over-damped for dead time up to forty-five seconds, and under-damped for everything above that.

Overshoot

The overshoot of the CB Water System was directly related to the dead time in the system. As dead time increases, overshoot begins to increase. The tuning rule with the least amount of overshoot was the IMC tuning rule. It did not see

any overshoot until the dead time reached 1 minute. The Haalman tuning rule performed the worst in the overshoot metric, having overshoot in every trial.

Settling Time

The variation in dead time had different effects to the different PID tuning rules. The further the dead time varied from the actual system dead time, one minute twelve seconds, the greater the settling time became during the IMC tuning rule trials. For Haalman, the shorter the dead time, the more improvement is seen in the settling time. The DR tuning rule saw the shortest settling time with a dead time of forty-five seconds, with settling time increasing as the dead time deviated further from forty-five seconds.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this work, a PID auto-tuning program was developed to automate the tuning of PID gain coefficients. The PID auto-tuning program uses the Step Response Method, a form of systematic tuning. The research has been carried out on the CB Water System at University of Virginia's Rice Hall, examining the relationship between the Bypass Valve and CB Supply Water temperature. The auto-tuning program has been designed to fit within the resource restraints of the Tracer MP581 Programmable Controller by limiting the number of variables used in the program. In performing plant identification, the program has taken advantage of individuals control charts, as well as B.P. Welford's streaming algorithm for computing the corrected sum of squares to limit the variables.

In systematic tuning, methods translate the plant approximation to PID gains using PID tuning rules. Each combination of plant identification method and PID tuning rule are compared in a Simulink simulation, where the CB Supply Water

temperature response to a disturbance is observed. The performance is measured by observing the rise time, overshoot, and settling time of the system response.

Among plant identification techniques, the auto-tuning program's approach to plant identification has shown competitive performance in comparison to common plant identification techniques. The other techniques required more data storage than practical for use on the Tracer MP581 Programmable Controller. The auto-tuning program is able to achieve comparable performance, while being able to fit within the resource constraints of the Tracer MP581 Programmable Controller.

Among the PID tuning rules evaluated, the IMC tuning rule outperformed the Haalman and DR tuning rules. Although the rise time was actual shorter for Haalman and DR, with a response time improvement of 43% and 39% for the auto-tuning program case, the effect on the settling time and overshoot metrics proved costly. The IMC tuning rule showed almost no overshoot, whereas the DR tuning rule showed 0.58°F overshoot and the Haalman tuning rule 0.85°F overshoot. The IMC tuning rule had less than half the settling time of the DR tuning rule and less than a third of the Haalman tuning rule. For its performance, the IMC tuning rule was chosen for use in the auto-tuning program.

In the CB Water System, the dead time can vary. The auto-tuning program's robustness was examined to variation of dead time. The system's original dead time is one minute twelve seconds. Using the auto-tuning program's plant identification method and IMC tuning rule, the CB Supply Water temperature response to any dead time less than the original value caused the system to have an over-damped response, whereas an increase to the dead time caused an under-damped response. The system is able to allow for some increase in overshoot before a pro-

cess instability would be reached. This shows the auto-tuning program is robust to decrease in dead time and some increase in dead time. This variation in dead time leads to the need for a method to readjust the PID gain coefficients, such as the auto-tuning program suggested in this thesis.

6.2 Future Work

This project utilizes a method for obtaining PID gain coefficients with a desired performance criteria. To accomplish this, the method must alter the state of the system by applying a step to the actuator. An area of research of interest to the author is developing an non-invasive method for tuning PID gain coefficients. This would use normal system input-output behavior to determine gain coefficients.

This has a number of very practical uses. Using building management systems, data is becoming more accessible for a wide range of systems. A method which can utilize this data to obtain PID gain coefficients would allow more controllers to have tuned PID gain coefficients, not just controllers that have auto-tuning program uploaded to them.

A modification to the systematic tuning approach may be implemented to determine PID gain coefficients. An approach may start with generating a model from the input and output data. Initially, it would need to be known if the system is linear or nonlinear and time invariant or time variant. If linear and time invariant can be proven, linear modeling approaches can be taken using common transformations. If nonlinear, the use of the Volterra series may prove to be useful. Either way, the most challenging task in this problem may be how disturbances can be modeled. In HVAC systems with random, varying degree of disturbances,

modeling the disturbance would prove to be useful.

Bibliography

[1] C.-C. Yu, *Autotuning of PID controllers: A relay feedback approach*. Springer Science & Business Media, 2006.

[2] W. A. Shewhart, *Economic control of quality of manufactured product*. ASQ Quality Press, 1931.

[3] B. Welford, "Note on a method for calculating corrected sums of squares and products," *Technometrics*, vol. 4, no. 3, pp. 419–420, 1962.

[4] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, "Feedback control of dynamics systems," *Addison-Wesley, Reading, MA*, 2006.

[5] A. Callender, D. R. Hartree, and A. Porter, "Time-lag in a control system," *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, vol. 235, no. 756, pp. 415–444, 1936.

[6] P. B. Deshpande, *Multivariable process control*. Instrument Society of America, 1989.

[7] W. Khuen Ho, T. Heng Lee, W. Xu, J. R. Zhou, and E. Beng Tay, "The direct nyquist array design of pid controllers," *Industrial Electronics, IEEE Transactions on*, vol. 47, no. 1, pp. 175–185, 2000.

[8] H. Koivo and J. Tanttu, "Tuning of pid controllers: survey of siso and mimo techniques," in *Proceedings of the IFAC Intelligent Tuning and Adaptive Control symposium*, pp. 75–80, 1991.

[9] K. J. Åström and T. Hägglund, *Advanced PID control*. ISA-The Instrumentation, Systems, and Automation Society; Research Triangle Park, NC 27709, 2006.

[10] D. J. Cooper, "Practical process control," *Electronic Textbook http://www. controlguru. com, accessed on Apr*, vol. 20, 2009.

[11] O. Smith, "Closer control of loops with dead time," *CHEMICAL ENGINEERING PROGRESS*, 1957.

[12] M. L. Luyben and W. L. Luyben, *Essentials of process control*. McGraw-Hill College, 1997.

[13] J. G. Ziegler and N. B. Nichols, "Optimum settings for automatic controllers," *trans. ASME*, vol. 64, no. 11, 1942.

[14] K. J. Åström and T. Hägglund, "Automatic tuning of simple regulators with specifications on phase and amplitude margins," *Automatica*, vol. 20, no. 5, pp. 645–651, 1984.

[15] A. Leva, C. Cox, and A. Ruano, "Hands-on pid autotuning: a guide to better utilisation," *International Federation of Automatic Control*, pp. 1–84, 2002.

[16] T. Kealy and A. O'Dwyer, "Comparison of process identification techniques in the time and frequency domains," in *International Conference on Systems Engineering*, p. 45, 2003.

[17] K. J. Åström and T. Hägglund, "Pid controllers: Theory, design, and tuning," *Instrument Society of America, Research Triangle Park, NC*, vol. 10, 1995.

[18] F. G. Shinskey, *Process control systems: application, design and tuning*. McGraw-Hill, Inc., 1990.

[19] D. Seborg, T. F. Edgar, and D. Mellichamp, *Process dynamics & control*. John Wiley & Sons, 2006.

[20] S. Tavakoli and M. Tavakoli, "Optimal tuning of pid controllers for first order plus time delay models using dimensional analysis," in *Control and Automation, 2003. ICCA'03. Proceedings. 4th International Conference on*, pp. 942–946, IEEE, 2003.

[21] E. S. Taylor *et al.*, *Dimensional analysis for engineers*. Clarendon Press, 1974.

[22] A. Haalman, "Adjusting controllers for a deadtime process," *Control Engineering*, vol. 65, pp. 71–73, 1965.

[23] D. E. Rivera, M. Morari, and S. Skogestad, "Internal model control: Pid controller design," *Industrial & engineering chemistry process design and development*, vol. 25, no. 1, pp. 252–265, 1986.

[24] M. Morari and E. Zafiriou, *Robust Process Control*. Prentice Hall, 1989.

[25] J. S. Oakland, "Chapter 7 - other types of control charts for variables," in *Statistical Process Control*, pp. 151 – 191, Oxford: Butterworth-Heinemann, sixth edition ed., 2008.

[26] D. E. Knuth, *The art of computer programming: sorting and searching*, vol. 3. Pearson Education, 1998.

Appendix A

Auto-tuning Program Code

The following is TGP code for running the auto-tuning program on the Tracer MP581 Programmable Controller.

The PI auto-tuning program will be described in the context of the Chilled Beam Water System. The actuator of interest is the CB Bypass Valve position, and the measured variable of interest is the CB Supply Water temperature. This program may be used by any linear first-order system. The auto-tuning program is broken up into four separate parts, each of which are described below.

A.1 Taking Control of the Actuator

To begin the auto-tuning program, the *Autotuner Switch* variable is switched, as shown in Figure A.1. The code in Figure A.1 is placed in the program that contains the PI controller for the CB Water System. The output of the switch box shown in green goes to the CB Bypass Valve position variable. Under normal operation, the *Autotuner Switch* is false, allowing the PI controller to control the CB Bypass Valve position. To activate the PI auto-tuning program, the *Autotuner*

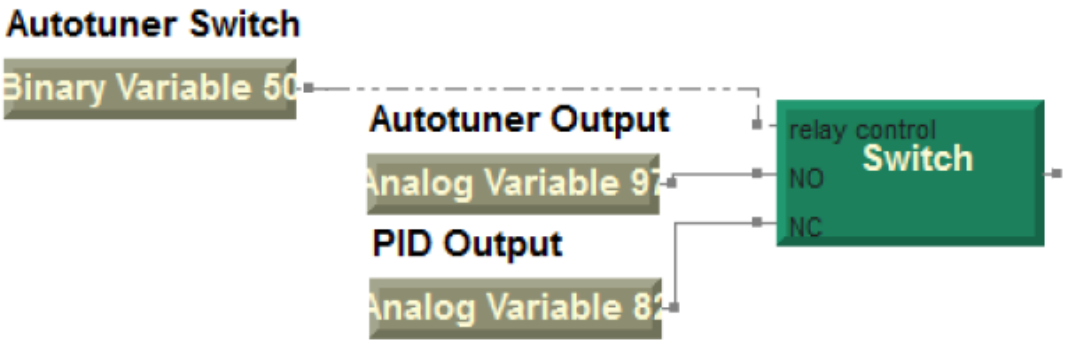


Figure A.1: Switching On the Auto-tuning Program

Switch is flipped by the user and the value becomes true.

A.2 Stepping the Actuator

Once the auto-tuning program is activated, it has complete control of the position of the actuator (in this case the CB Bypass Valve). The valve is held at the position the PI controller commanded at the last time step before the auto-tuning program has taken control of the valve. This position is held for a fixed amount of time, determined by the *Seconds before Step* variable. This value must be greater than or equal to the estimated amount of dead time in the system; with no penalty for an over-conservative estimate. This is done to ensure the CB Supply Temperature is at a steady value. After the CB Bypass Valve is held at the position for the fixed amount of time, the auto-tuning program steps the valve position either above or below its current value. The value of the step size is determined by the *Size of Step* variable. The time that the valve position is stepped is recorded and stored in the *Time of Step* variable.

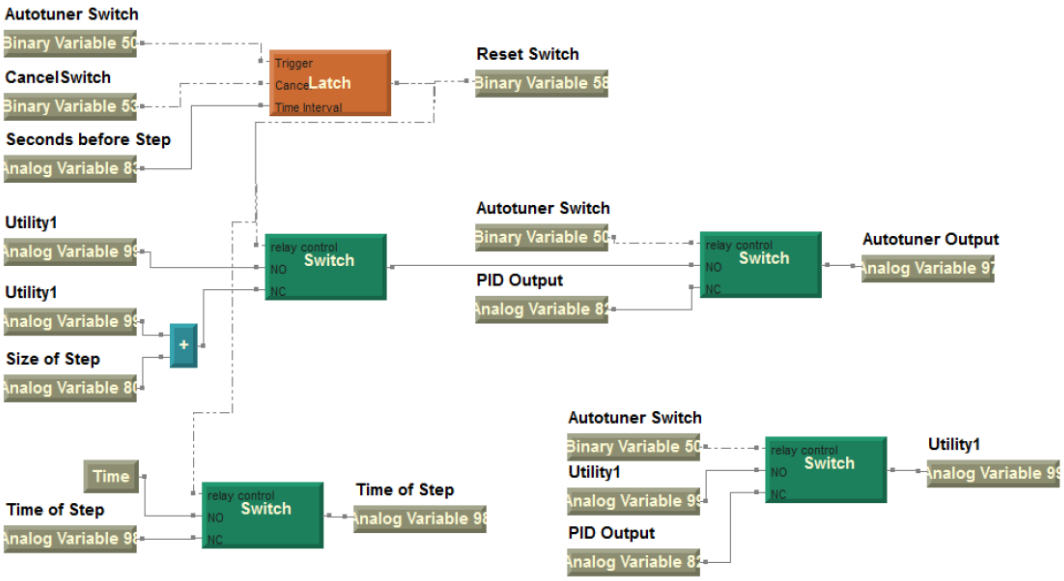


Figure A.2: Stepping the Bypass Valve

A.3 Extracting Information from Step Response

After the actuator is stepped, the auto-tuning program observes the measured variable (in this case the CB Supply Temperature) response. Figure A.3 shows example data of a measured variable's response to a step in an actuator. The auto-tuning program must decide when the measured variable begins changing due to the step in the actuator position, circled in purple in Figure A.3. It must also decide when the measured variable stops changing due to the step in the actuator position, circled in green. The following sections describe how the auto-tuning program extracts the information from the step response.

Standard Deviation Algorithm

The Tracer MP581 Programmable Controller has a fixed number of analog and binary variables available for each program to use. To conserve variables, streaming algorithms are implemented in the auto-tuning program. The program uses B.P. Welford's streaming corrected sum of squares algorithm to calculate standard

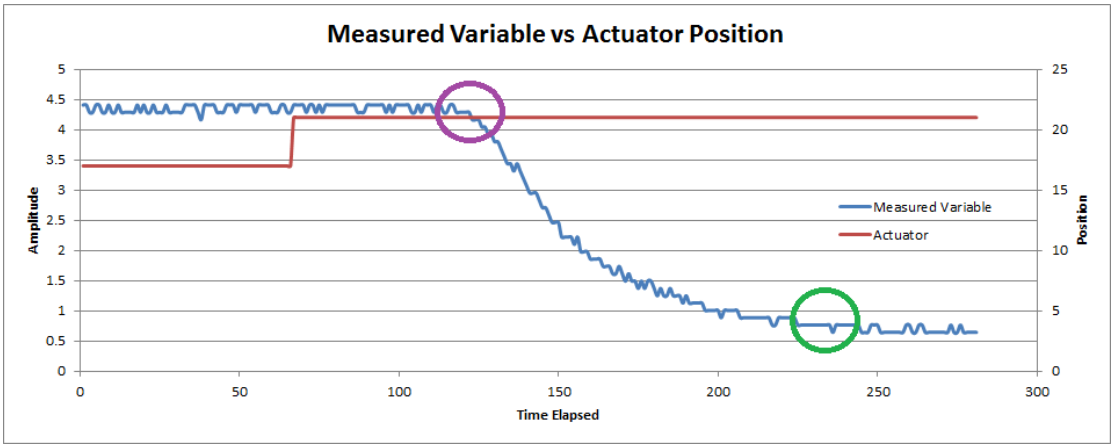


Figure A.3: Example Data from Step Response

deviation. This algorithm is demonstrated in the Tracer Graphical Programming language in Figure A.4 on the next page.

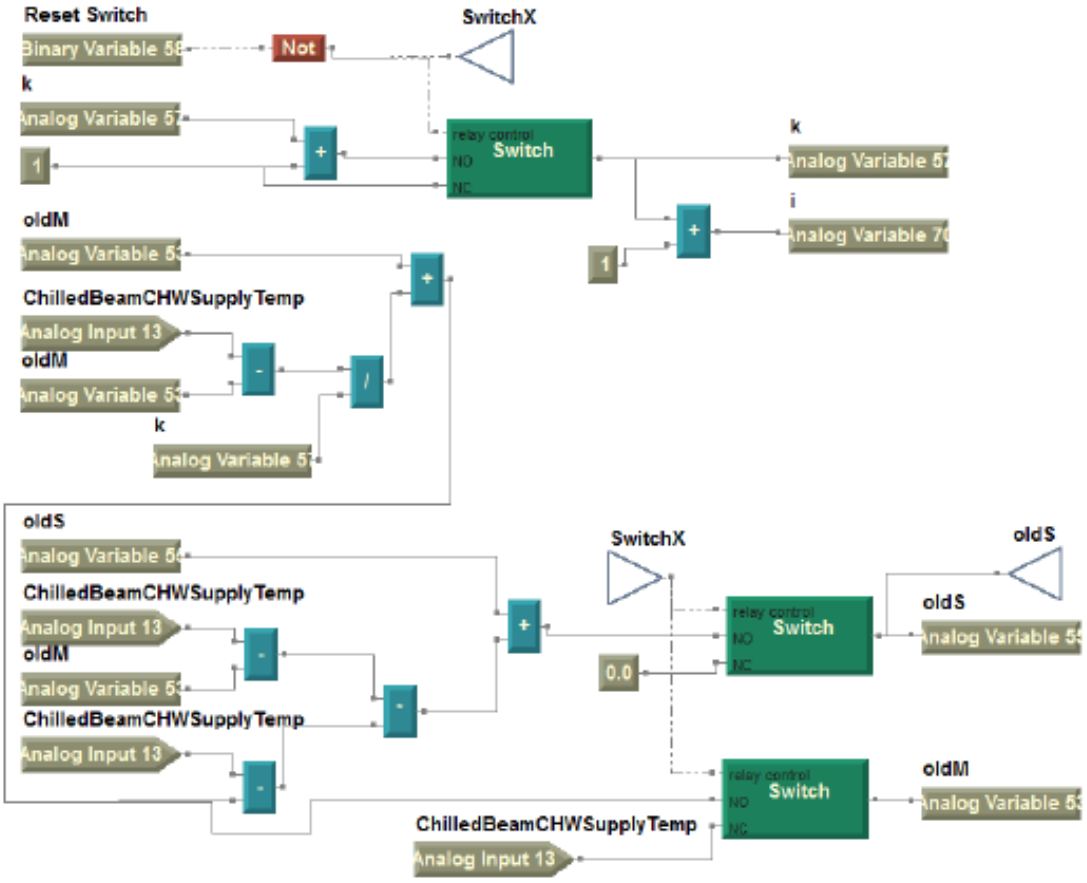


Figure A.4: Streaming Standard Deviation Algorithm

Determining Initial Change in Measured Variable

To determine the initial change in the measured variable, upper and lower control limits are assigned. When the measured variable goes above the upper control limit (*UCL)* or below the lower control limit (*LCL*), the measured variable has begun to change due to the step. When the auto-tuning program decides the measured variable has begun to change, the time and value of the measured variable are recorded in variables *Time of Change* and *TempAfterChange*. The value of the measured variable at the time step before the change is also recorded in the variable *ChangeTemp*. The standard deviation from the time the actuator position is stepped, to the time the measured variable begins to change is recorded in variable *STD Threshold*, and used to determine when the measured variable

stops changing due to the step in the actuator position.

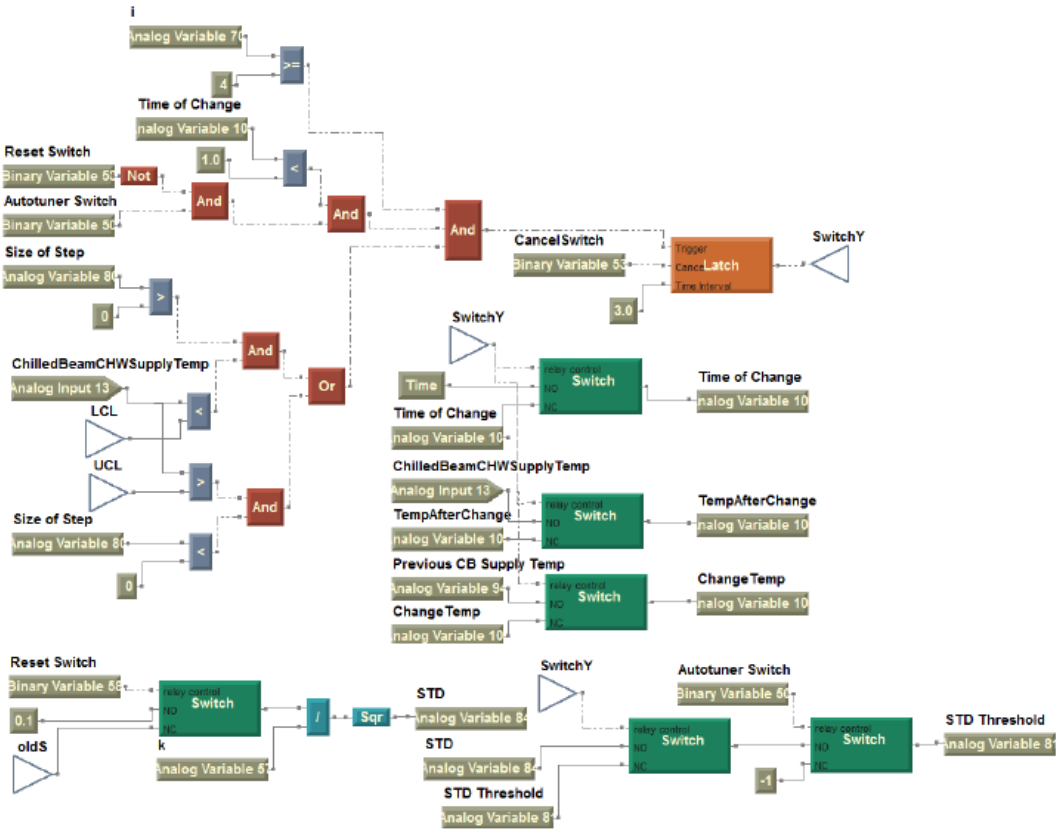


Figure A.5: Determining Change in Measured Variable

Upper and Lower Control Limits

The upper and lower control limits are defined using Walter A. Skewhart's Individuals Control Charts. These control limits are updated at every time step, using current and past measured variable data to determine each limit. The control limits are described in Figure A.6.

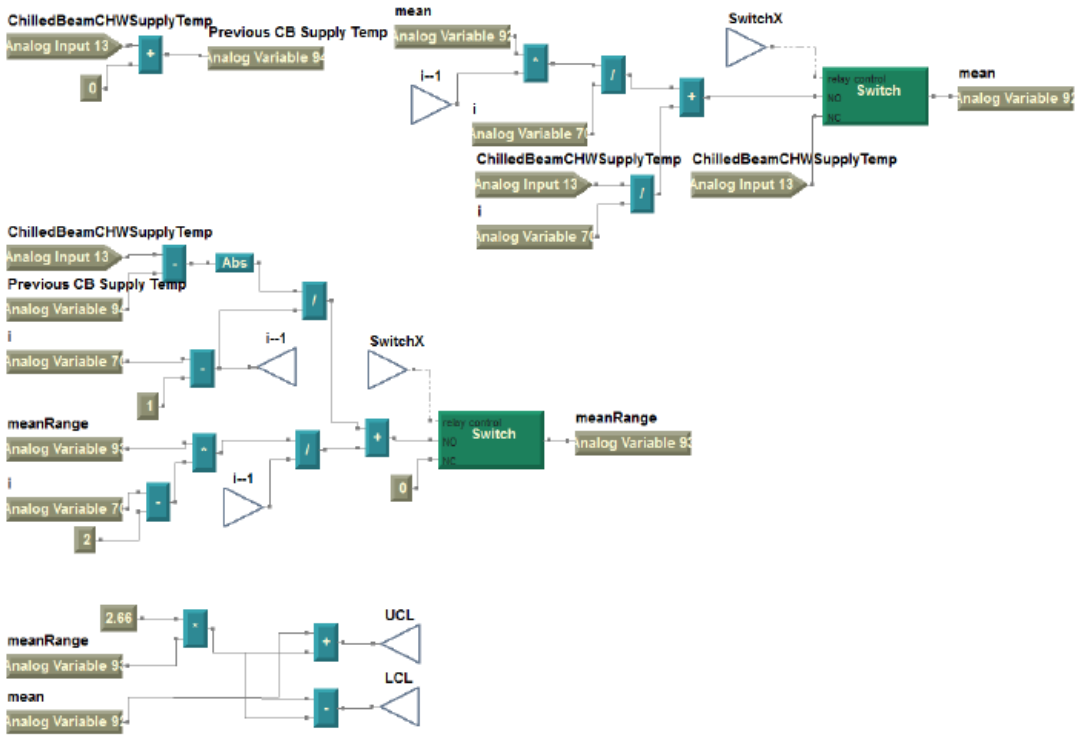


Figure A.6: Control Limit Calculations

Determining When Measured Variable Stops Changing

The next piece of information the auto-tuning program needs is the time and value at which the measured variable stops changing due to the step in the actuator. Initially, the Auto-tuing Program begins to calculate the standard deviation of a moving window starting when the measured variable beings to change due to the step. The moving window uses four data points. The standard deviation of the moving window is compared to the *STD Threshold* variable. When the standard deviation of the moving window goes below the *STD Threshold*, the measured variable has stopped changing.

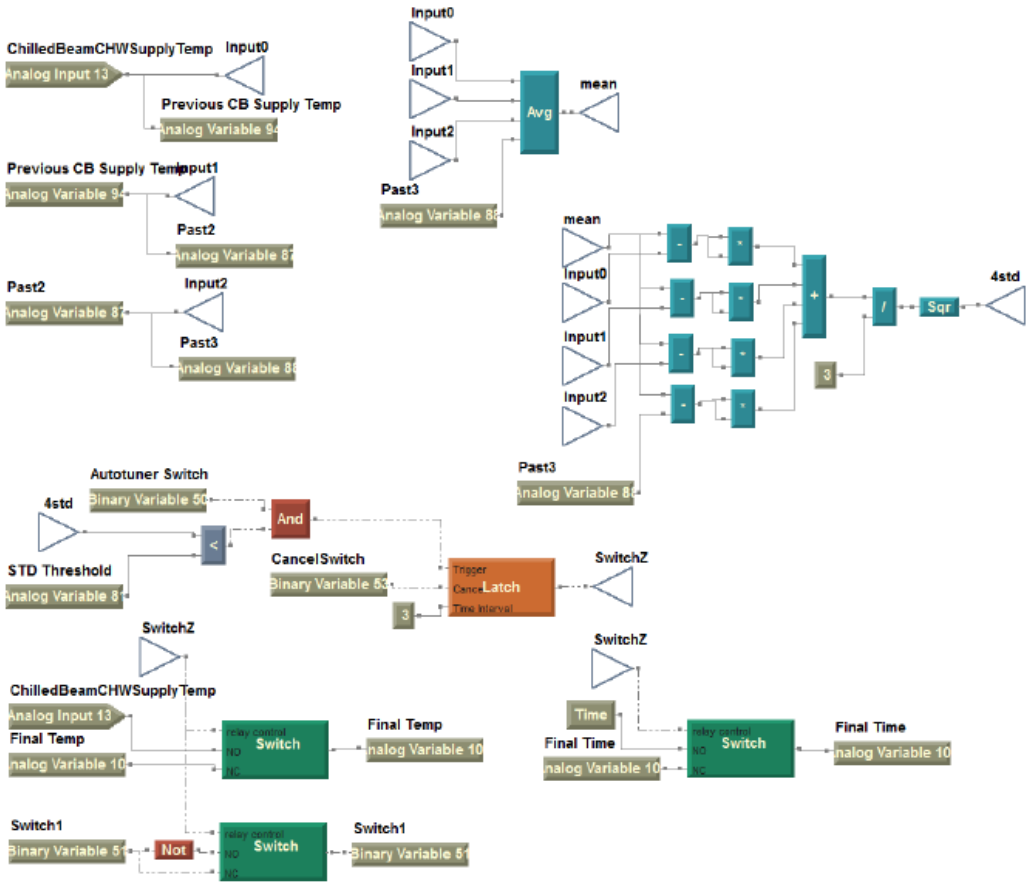


Figure A.7: Determining Stop in Measured Variable Change

A.4 Calculating PI Gain Coefficients

Using the information taken from the step response, the auto-tuning program determines the plant characteristics: dead time, gain, and time constant. These plant characteristics are then used to determine PI gain coefficients. PI Tuning Rules use these determined plant characteristics to formulate PI gain coefficients. The auto-tuning program uses a Internal Model Control (IMC) Tuning Rule developed by D. E. Rivera.

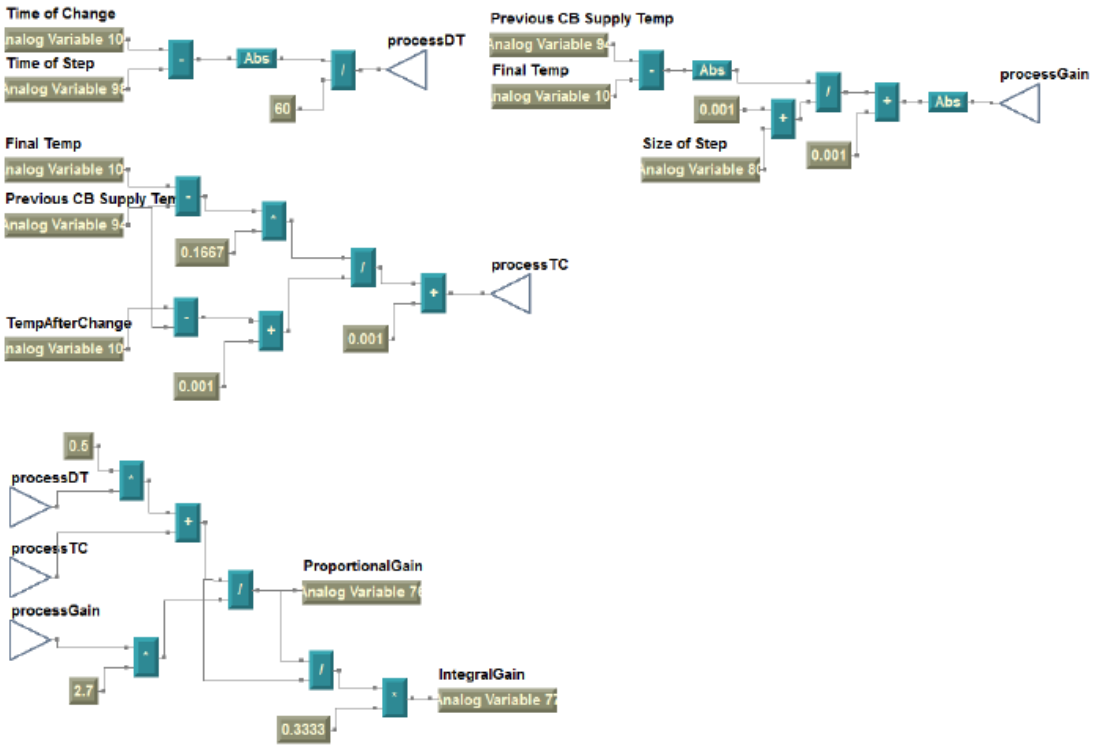


Figure A.8: Calculating PI Gain Coefficients