Literature review and simulation study of feedback control of systems with dead time

An analysis of the effects of dead time on various feedback control system techniques

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Bachelor's thesis

Electrical and Automation Engineering, Valkeakoski

Spring 2021

Ioannis Aarno Samuel Kefallinos

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Electrical and Automation Engineering Abstract

Author Ioannis Aarno Samuel Kefallinos Year 2021
Subject Review of Feedback Control Systems with dead time
Supervisors Juhani Henttonen

In this thesis various common situations in system control with different dead times and model accuracy are used to test various controller applications to further understand and write down the effects of measured and unknown dead times on different commonly used feedback control systems and their tuning methods. This testing of controllers was done in a Single-Input-Single-Output, First order plus dead time context using Matlab-Simulink to analyze the effects both with measured and unknown dead time on controller performance.

After multiple tests it could be concluded that reactive controllers were slowed from measured dead time and these were able to give non-oscillating performance under compromised conditions. Predictive controllers were able to provide a performance with less over correction, but they were extremely vulnerable to incorrect evaluations of process dynamics and dead time errors. Both controllers-types have modifications to combat their weaknesses at a cost to the overall performance.

Keywords control engineering, feedback control, controller tuning, dead time, simulation
Pages 72 pages and appendices 9 pages

Contents

Appendices

1 INTRODUCTION

Feedback controllers have seen use in a large variety of industries and have been a vital part in the function of processes. This is due to controllers providing consistent and continuous control needed for the optimal output of various machinery, avoidance of catastrophic failure, adaptability to unforeseen circumstances and other uncertainties. (Sira-Ramírez, 2014, p. 3)

In many control processes, dead time has been a constant to varying extents. As a result of this phenomenon's near omnipresence, the understanding and mitigation of the effects or complete elimination of delays present in such systems has been a topic of interest and importance in various fields. This thesis will elaborate on various subjects on the topic and research and test the various methods used for dead time compensation or control in a simulated environment.

The purpose of this thesis is to review popular feedback control techniques and evaluate them based on the results of a set of simulations exhibiting different circumstances of dead time and input disturbance in a Matlab-Simulink simulation environment. In the last chapter of this thesis the ideal technique to use based on circumstances is evaluated and a general outline on what method of feedback control should be used based on the information available and the intended performance after the general effects of dead time on each control method are made clear.

2 BACKGROUND

Before explaining the tested control methods and the effects of dead time on them, the definition of the terms and the tools used in this thesis need to be described. This is for the sake of future proofing and context for any readers that lack fluency in the field of control engineering.

2.1 Feedback control systems

Closed-loop control systems , or feedback control systems, are process control systems where a set-point value and current output are used to manage future output. The output is continuously measured and compared to another value as reference, the system calculates a discrepancy between the given value and the set point value and an error signal is relayed. With the appropriate error signal recieved, controller output is adjusted to decrease the differences between the referential value and the process output value until the difference is sufficiently lessened or zero. (McGraw-Hill, 2003)

The term includes but is not limited to: Proportional-Integral-Derivative Controllers, Smith Predictors, Model Predictive Controllers and the combinations and variations of thereof. For most control systems their values and their functions require different methods of application or tuning for in different situations. The factors affecting the decision on picking and tuning of these systems can range from the intended value, the intended behavior, the expected output values and expected disturbance. One of the factors that need to be taken into account when tuning a controller is dead time.

2.2 Dead time

Dead time for feedback control systems is the amount of time between the change in process input and the process output. Dead time can be caused by a variety of reasons like the placement or sensitivity of the sensor used to detect the output, accumulation of time lags between multiple systems, the speed of transported mass, processing of input or output

and the controller's method of manipulating the output. (McMillan & Vegas, 2010, pp. 139-141)

What makes significant unaccounted dead time undesirable is that it can result in the process producing unpredictable output or other unwanted results. A control system with that has not been prepared for the appropriate dead time can result in an overshot or undershot response due to increase in phase shift between the input and output and to larger extents result in uncontrollable oscillations. Even when taken into account for, dead time slows reactive controller response by necessity to avoid overcorrection or oscillation. The need for controllers that maintain speed and stability in large dead time environments resulted in the creation of various feedback control methods that bypass this necessity for a slower response through prediction or a separate input.

2.3 Matlab-Simulink

Matlab, formerly known as Matrix Laboratory, is a computing language initially developed by Cleve Moler and ported over to C by Moler, Jack Little and Steve Bangert (Moler, 2004).It is an advanced programming language specialized in mathematics, allowing for complex calculations and simulations of mathematical concepts. This programming language has seen use as a programming tool, a tool for education, a method of control tuning and a simulation environment. (Xue & Chen, 2013, pp. 6-7)

Simulink is a Matlab-based graphic-programming environment that is not included in the default installation of Matlab, it is an sub-application to the default Matlab programming environment. The purpose of this application is to allow a quick, easily readable and editable simulations of systems without intimate understanding and memorization of the Matlab language or commenting of code. Due to the readability of the program, quick access to various other Matlab-applications such as " PID tuner" and "MPC Designer", code generation without writing code, easy changing between controller variants, compatability with hardware and the ability to visually represent various systems for the purpose of readability in simulation environments Simulink has been a popular software for engineering and simulation. (Xue & Chen, 2013, pp. 145-146)

3 PID-CONTROLLERS

The continuous Proportional Integral Derivative controller is the oldest feedback control method that is still widely used by the time of writing this thesis. It is designed to correct output using the values of its namesake proportional, integral and derivative elements based on the error between the set point value and the output value. (Sun09p. 111) It is also the most widely used process control method, to the point of it composing the majority of process controllers used in an industrial setting (Samad, 2017)

The PID-controller controller is designed for stability and control, but it is less capable in systems with large amounts of dead time. They are described as being effective in situations where exact information regarding the process is complicated to implement or unknown.

3.1 Function of a PID-controller

The PID-controller will adjust the output using three different elements:

- The Proportional element that is proportional to the error at the instant of correction. It is supposed to be factored as the current error between the set point value and the actual value.
- The Integral element is the integral to the error up to present time. It is supposed to account for the cumulative value of error that is used for adjusting smaller frequency disturbances. It reduces or eliminates remaining error from the P-element.
- The Derivative element is the derivative of the error up to the instant of correction. It is supposed to account for future errors, preventing changes in the error signal.

The mathematical relationship between these elements of the controller and the output depend on the form of the PID-controller. While the values of each controllers type's P-, I- and D-elements can be converted between them through various equations,meaning they are interchangable, each system functions differently in their relationship between inputs and the output. (Altmann, 2005, p. 16) In all the below equations for controller structure and tunings "Kc" is the process gain,"Ti" is the integral time and "Td" is the derivative gain.

Serial PID-controller

In the currently rare serial PID-controller, also known as the classic PID-controller, all the elements are placed in a linear series. The resulting output is the product of the P-element, the I-element and the D-element, creating equation 3.1:

$$C(s) = Kc \cdot \frac{(1 + Ti \cdot s)}{Ti \cdot s} \cdot (Td \cdot s) \tag{3.1}$$

The classic PID-controller is rarely used due to it being harder to decouple elements than other variants. (O'Dwyer, 2006, p. 11)

Parallel PID-controller

In the parallel form the PID-controller, where all the elements are parallel to each-other all of the elements are summed independently together, as shown in equation 3.2:

$$C(s) = Kc + \frac{Kc}{Ti \cdot s} + Kc \cdot Td \cdot s \tag{3.2}$$

The main difference between this type and other controllers is how unlike the other methods, all elements can be decoupled with few complications. (O'Dwyer, 2006, p. 7)

Ideal PID-controller

In the “ideal” form of the PID-controllers, also known as mixed PID-controllers, the I-element and D-element are parallel to each-other before being multiplied by the P-element, with resulting output being according to equation 3.3:

$$C(s) = Kc \cdot (1 + \frac{1}{Ti \cdot s} + Td \cdot s) \tag{3.3}$$

This method is most popular due to relatively simple tuning, while still easy to decouple either I- or D- elements from the controller. In this thesis the standard PID-controller is used. (O'Dwyer, 2006, p. 7)

Constraints and Clamping

In the common event that constraints are applied, it is often suggested that an anti-windup system is used, as the Integrator element can overflow the system or it will take too much time to change to what it needs to be when limiting output (Silva, Flesch, & Rico, 2018, pp. 948-949). The anti-windup method tested in this thesis is the clamping method.

In the case of an anti-windup system using the clamping method, also known as conditional integration, software is used to stop the function of the integral element with no mathematic basis. When the output is exceeding the output limit, the integrator element is forbidden from resuming function until the integrator output and the input of the block are of opposite signs. (Mathworks, 2012)

3.2 General variants of the PID controller

The PID-controller has had different modifications for different purposes and trade-offs. Many of the modifications either added or removed elements to make tuning easier, add functionality or remove factors that would produce complications once introduced to certain conditions. The use of these variants and the default controller are not mutually exclusive and can be combined or stacked in a cascade for further disturbance rejection and better control overall (Altmann, 2005, pp. 16-17). The aforementioned combinations and cascade applications are not analyzed, as the focus of this thesis is on individual control method performance and analysis.

PI controller

The PI controller is a PID-controller with the Differential-element is removed, leaving only the Proportional and Integral elements. These are often used when the noise present in the system is large enough to cause issues in Derivative control and in dead time-dominant processes. It is also stated to work better than the default PID-controller within heavily damped systems where changes are minimal over time and is the most popular PID-controller variant where an element of the PID-controller is removed. The PI controller is also the most used PID-controller variant when it is used in conjunction with other systems such as the Smith Predictor and is usually the best suited for high noise or high Dead time-to-Time-Constant ratios. (Altmann, 2005, p. 16)

2DOF PID-controller

The two degrees of freedom PID-controller is a PID-controller that has been modified to have two degrees of freedom by modifying the Proportional and Derivative elements. The relationship between the inputs, filters or feed-forward mechanism and the output depend on whether the controller is in parallel or standard form. Equation 3.4 below explains the conversion between the controller's inputs and output in the standard form for the 2DOF-PID-controller:

$$C(s) = Kc \cdot \left((\alpha \cdot r - y) + \frac{1}{Ti \cdot s} \cdot (r - y) + Td \cdot s \cdot (\beta \cdot r - y) \right) \qquad (3.4)$$

Where the "y" represents the second input while the "r" represents the set-point value. "α" and "β" are set-point coefficients for the P- and D- elements. This variant permits set point weighting by directly or indirectly multiplying the reference value by a certain amount to control how the system responds to disturbances and its rise-time. This can be achieved by placing a filter or using a feedforward configuration to remove a part of the reference value. (Sundaravadivu, Sivakumars, & Hariprasad, 2015)

3.3 Common PID tuning rules

There is no one true tuning rule that fits all situations for PID,PI or PD controllers, certain tuning rules for different situations are used for different occasions and purposes. All of the tuning rules listed below have been made with the intent of managing and dealing with the slowed flow of information caused by dead time. In all tuning methods halving the Controller Gain is an option for stabilizing disturbance rejection capabilities, which in this thesis will be applied to the PID-controller tunings of Ziegler-Nichols and Cohen Coon tuning methods and the PI-tuning method for Ziegler Nichols.

Ziegler Nichols tuning method

The Ziegler-Nichols tuning method, also known as the ZN tuning method, is the one of the older popular tuning methods created for PID and PI controllers. The Ziegler-Nichols tuning rules are designed for situations where the time constant is longer than the dead time to a significant extent. There are two common ways of tuning a controller using the Ziegler Nichols method, using process dynamics attained from the “Ultimate Oscillation” method or doing a step test elaborated in the “Cohen Coon” tuning sub-chapter.

In the “Ultimate Oscillation” method the I-element and the D-element are tuned to 0 and the P-element is increased until the controller oscillates with stable peaks and wavelengths. Ultimate gain “Ku” is the most sustained periodic oscillation in the output gained from the P controller and the ultimate period “Tu” is the time period between the oscillations in the ultimate gain system (Smith & Corripio, 1985, pp. 304-305).Once these process values have been attained, the appropriate proportions of the PID-controllers elements are calculated using table 1 below:

Table 1: Values for Ziegler Nichols PID and PI tuning using data from the Ultimate gain test. (Smith & Corripio, 1985, p. 306).

ZN	Kc	Ti	Td
PID	0.589·Ku	$\frac{Tu}{2}$	$\frac{Tu}{8}$
PI	0.45·Ku	$\frac{Tu}{1.2}$	-

Another method for ZN-tuning is to use FOPDT model-based process dynamics in place of the ultimate gain method detailed in table 2:

Table 2: Values for Ziegler Nichols PID and PI tuning using data from the Step Test. (O'Dwyer, 2006, pp. 27,154).

ZN	Kc	Ti	Td
PID	$\frac{1.2 \cdot T}{\theta \cdot K}$	$2 \cdot \theta$	$0.5 \cdot \theta$
PI	$\frac{0.9 \cdot T}{\theta \cdot K}$	$3.33 \cdot \theta$	-

One common procedure to attain the above values from an FOPDT-model and an explanation for the model itself is detailed in later in “Cohen Coon tuning method” below.

When using ultimate oscillation values this method of tuning can harm the equipment used and the results of the tuning method using either the values itself can be considered slow to ne usable in dead time systems where dead time is more than half the time constant (Sung, Jietae, & In-Beum, 2009, p. 154).

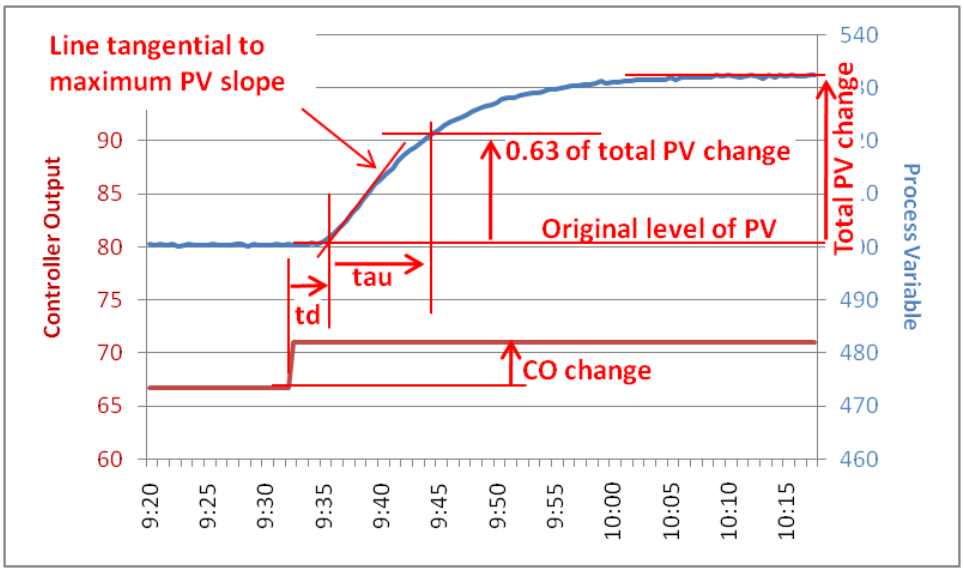
Cohen Coon tuning method

The Cohen and Coon method, abbreviated as CC-tuning, is a FOPDT-model based tuning method that uses three characteristics of the process model: process gain, dead time, and time constant. A PI controller tuned with this method is quick to respond to errors and can be expected to reach the set point value faster than other PI-tuning methods listed in this thesis.

A common first step for this method of tuning is done by performing a step test on the process to find the processes process gain, time constant and dead time. A step test is done by having the controller change its output, then measuring different statistic qualities of the process after measuring the total change in output over a length of time. The process gain of the process is estimated by dividing the change in process value with the change in controller output after controller output stops increasing. The dead time is calculated by

measuring the time difference between the changes in the controller output and the process output. The time constant is the difference between the value reaching around 63% of the total gain of the process and the end of the dead time (Altmann, 2005, p. 7).This procedure can be summarized in Figure 1:

Figure 1: Figure depicting a step test where the change in process value is the change in process output, “td” is dead time, ”tau” is the time constant and “CO Change” is the change in Controller output that will be compared to the change in Process output. (Smuts, 2010)



It is recommended to have the step test be repeated with differing controller output changes to create averages, as they are not absolute and can change slightly between step-tests due to human error or slight differences in results.

This is a common way to attain a processes First Order Plus Dead Time-model, abbreviated as the FOPDT-model. The FOPDT-model is the most popular way to mathematically express plant behavior and often is used for various controller tuning methods, modifications and simulations. The general structure of an FOPDT model should be in the form of equation 3.5:

$$Pn = \frac{K}{Ts + 1} \cdot e^{-\theta s} \tag{3.5}$$

In equation 3.5 the process dynamics of process gain “K”, time constant “T” and dead time “θ” are often used as a basis to tune the Proportional, Integral and Derivative elements of a PID-controller. In the case of the Cohen-Coon method, the P-, I- and D- elements are tuned according to table 3, in which the Cohen-Coon tuning rule values for a PI and a PID-controller are contained in:

Table 3: Values for a CC-tuning of PID and PI controller values. (O'Dwyer, 2006, pp. 28, 155)

CC	Kc	Ti	Td
PID	$\frac{1.35 \cdot (\frac{T}{\theta} + 0.0185)}{K}$	$\frac{2.5 \cdot t \cdot (T + 0.185 \cdot \theta)}{(T + 0.611 \cdot \theta)}$	$\frac{0.37 \cdot \theta \cdot (T - 0.324 \cdot \theta)}{(T + 0.129 \cdot \theta)}$
PI	$\frac{0.9 \cdot \left(\frac{T}{\theta}\right) + 0.083)}{K}$	$\frac{3.33 \cdot T \cdot (\theta + 0.093093093 \cdot T)}{\theta + 2.22 \cdot T}$	-

Skogestad’s suggestion for IMC-based tuning methods

The IMC-based tuning method, also known as Lambda-tuning, is another tuning method than can use the FOPDT-model to set values for the PI or PID-controller. In this tuning method the FOPDT-based process dynamics are used for tuning the PID-controller to emulate an Internal Model Control system. The actual controller value tuning for the PI controller using this method can be seen in Table 4 below:

Table 4: Values for Lambda tuning of PI-controller values (Smuts, 2010)

IMC	Kc	Ti
PI	$\frac{T}{K \cdot (\lambda + \theta)}$	MIN(T; (8·θ))

Where the “λ” is the closed loop time constant desired by the person tuning. In this thesis a variant of tuning this method known as the SIMC-tuning method will be tested instead. Skogestad’s suggestion for Internal Model Controller-based tuning, also known as SIMC-tuning, is a variant of the Lambda-tuning method where in place of having a user picked variable to add to the dead time as detailed in table 4 above, dead time is instead doubled:

Table 5: Values for Skogestad's suggestion for Lambda tuning for PI-controller values. (O'Dwyer, 2006, p. 48)

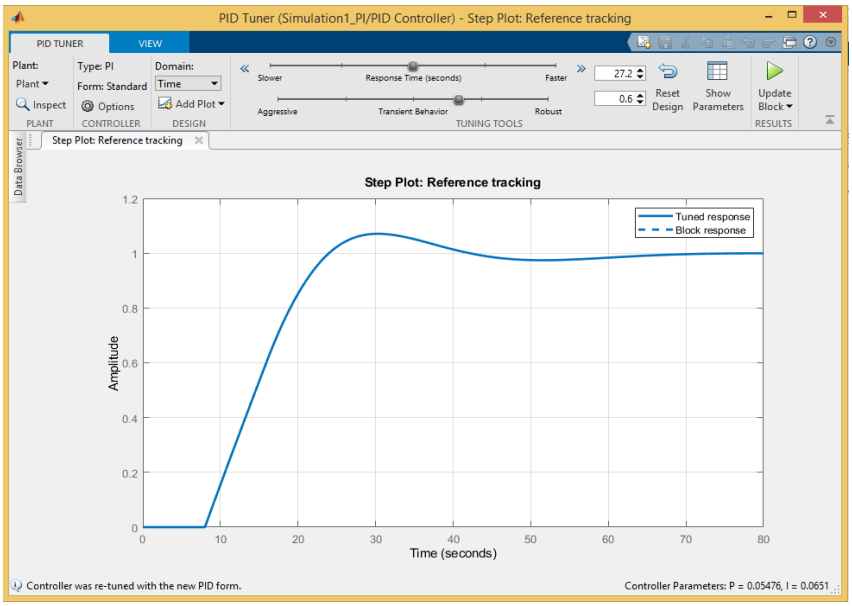
SIMC	Kc	Ti
PI	$\frac{T}{K \cdot 2 \cdot \theta}$	MIN(T; (8·θ))

Using the values in table 5 result in a more consistent method of tuning while still lowering the proportional element enough to still provide a relatively quick response. This method is said to provide a speed comparable to the Cohen-Coon method and while providing a better level of robustness with less overshoot (Smuts, 2010).

Matlab software-based tuning

The Matlab programming environment offers an option of the tuning of the PID-controller based on response time and amount of correction while offering almost real-time feedback on the output based on the input of the system if the “Control System Toolbox” is installed, as it is needed for linearization. This method works on trial and error or by through the wanted controller results, but is the simplest and often most effective method to implement with only two elements and real time results being shown in Figure 2 below:

Figure 2: "PID Tuner" program window for a PI controller in Matlab-Simulink version R2020b



When in use the controller can be tuned until the wanted form of the output is achieved. Due to the availability of this tuning method and the other tuning methods being considered good enough for the majority of industry, analysis of this tuning method will not be as throughout as it is for the named tuning methods covered in this thesis.

Tuning for 2DOF PID-controllers

The 2DOF-PID controller is tested using two tuning rules, the Hiroi-Terauchi tuning rules made in 1986 and the Taguchi-Araki tuning rules invented 14 years later (O'Dwyer, 2006, pp. 217-218). The Terauchi tuning rules are simpler with standard filters placed and an overall simpler equation for each value and tuning for Integral Time and Derivative Time are made with only dead time in mind. In contrast the Taguchi tuning rules are more complex and require more than one process dynamic values for all controller variables, including the set-point coefficients. This difference complexity is fully explained in Table 6:

Table 6: Values for the Terauchi and Taguchi tuning methods of 2DOFPID-controller values. (O'Dwyer, 2006, pp. 217-218)

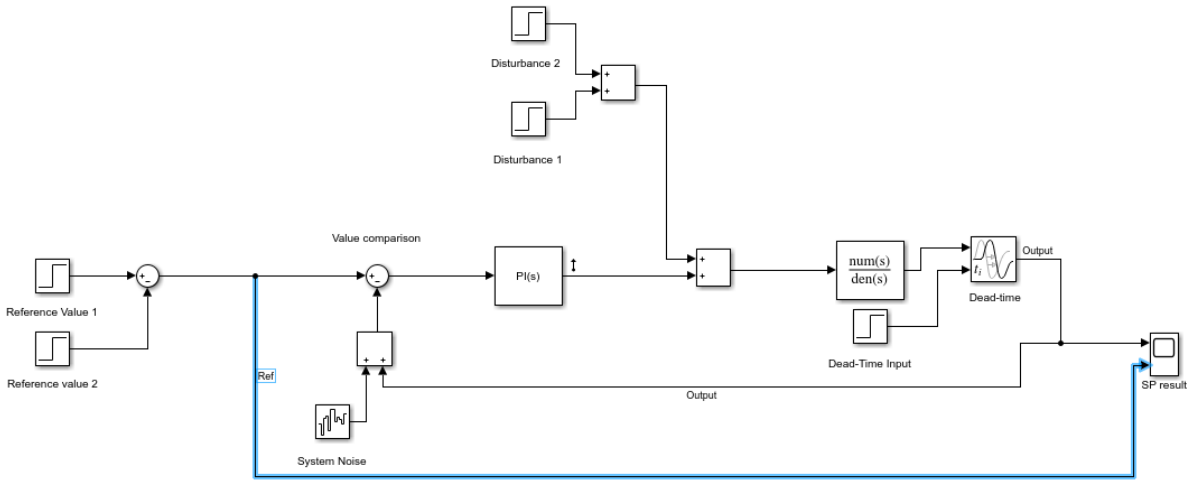
2DOF-PID	Kc	Ti	Td	α	β
Taguchi and Araki	(0.1415+(1.224/(T/θ-0.001582)))/K	T·(0.01353+2.2·(T/θ)-1.452·((T/θ)^2)+0.4824·((T/θ)^3))	(0.000278 3+0.4119·((T/θ)^2)-0.04943·(T/θ)^3)	0.6656-0.2786·(T/θ)+0.03966·((T/θ)^2)	0.6816-0.2054·(T/θ)+0.03966·((T/θ)^2)
Terauchi 0% overshoot	$\frac{0.98 \cdot T}{K \cdot \theta}$	$\theta \cdot 2.38$	$0.42 \cdot \theta$	0.6	1
Terauchi 20% overshoot	$\frac{1.2 \cdot T}{K \cdot \theta}$	$2 \cdot \theta$	$0.42 \cdot \theta$	0.6	1

As seen in table 6 above, the robustness and speed of the Terauchi-tuned controller depend on whether one opts for a 0% overshoot or a 20% overshoot variant of the tuning method. The 20% overshoot variant is best suited for slower processes and a faster disturbance rejection while the 0% overshoot variant should be used in faster processes to avoid overshoot, as seen with the increase in process gain and decrease in the Integral time.

3.4 Implementation in Matlab-Simulink

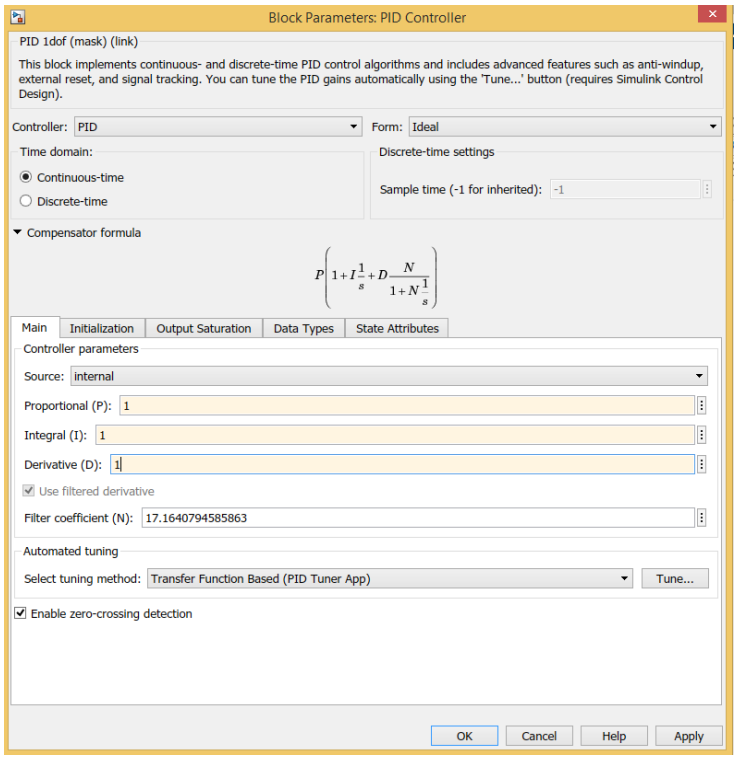
Matlab-Simulink has a library of function blocks that simulate continuous and discrete-time dynamical systems. All the PID-controller variants detailed in previous segments already exist in said library to minimize setup time and fasten the general tuning process and said PID-controllers can be dragged and dropped from the library to the Simulink simulation environment (Xue & Chen, 2013, pp. 146-149). The PID-controller testing environment for this thesis composed in Matlab-Simulink is shown in Figure 3:

Figure 3: General Template for testing PI, PID and 2DOFPID-controllers in Matlab-Simulink for this thesis.



When using these controllers in a simulation environment, the input u is the error signal that is made from the difference between the set point value and the output value “y”. Any result will have the PID-controller eliminate the error accordingly by increasing or decreasing the output according to the P-, I- and D-elements that are inputted through the block parameter window of the PID-controller that is shown in Figure 4 below:

Figure 4: PID-controller parameter window in Matlab-Simulink for version R2020b.

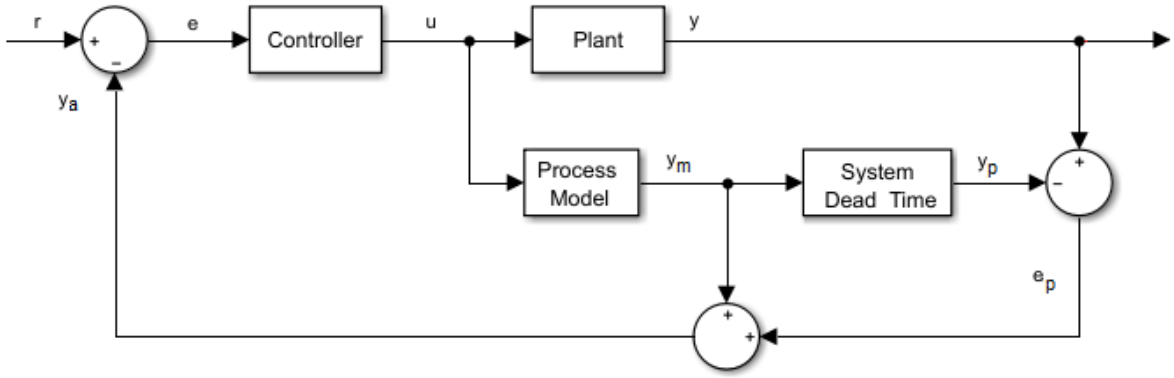


Among the settings available in the main parameter window is the “PID-tunnner” program, and the aforementioned P-, I- and D-element fields that can be set to be input externally, to enable more complex configurations to fit shifting situations. The other point of interest for this thesis in the properties window is the output saturation segment, where the upper and lower limits on output for the controller can be applied.

4 BACKGROUND FOR SMITH PREDICTORS

The Smith predictor, abbreviated as SP, is a predictor-based dead time compensation control system that predicts and controls the system according to the estimated process model output and later adjusting controls to the actual output to combat longer dead times. The system is composed by a controller part and a predictor part that uses a process model and a dead time model. The general structure of a SP is illustrated in Figure 5 below:

Figure 5: Figure depicting the general structure of the default Smith Predictor. (Mathworks, 2021)



The main controller of the SP is traditionally a PI controller with a process model “G” to estimate and directly feed the value of the output without disturbances and a dead time model, or identified system dead time, in the form of “e^(-ts)”. These two models combine in the complete prediction model process “Gn”, that in ideal circumstances should be as shown in equation 4.1 below:

$$G(s)\cdot e^{-\tau s} = G_n(s), \qquad G_n(s) \approx P_n(s) \qquad (4.1)$$

Where the prediction model should be equal to the process “P_n” or “Plant” under ideal circumstances. The controller part includes the predictor in its calculations of controlling output in advance of the dead time, either eliminating dead time completely with complete model accuracy or minimizing its effects. The Smith Predictor is able to provide quick and robust corrections to systems with large dead times, but it is vulnerable to major model miscalculations or changes in the process. (Normey-Rico & Camacho, 2007, pp. 131-133)

4.1 Mathematical function of the Smith Predictor

The Smith Predictor will constantly predict and then compare the predicted system output to the actual output of the system output through the process model and the dead time variable, then feed it back to the controller in a feedback control system. The ideal result for this overall procedure is demonstrated in equation 4.2:

$$SP(t) = \frac{C(s) \cdot P_{\mathrm{n}}(s)}{1 + C(s) \cdot G_{\mathrm{n}}(s)} \tag{4.2}$$

In which "Pn" is the plant model, "Gn" is the SPs complete prediction model and "C" is the controller used. (Karimi, 2019, pp. 212-214). In a closed-loop system, the Smith predictor has three properties that provide its function as a dead time compensator that are listed in the below segments:

Dead time Compensation

To effectively eliminate dead time, the mathematical relationship between the ideal process model and the characteristic equation for output should be similar to equation 4.3 when the error signal is 0 and there is no disturbance (Normey-Rico & Camacho, 2007, p. 132):

$$1 + C(s) \cdot G(s) = 0 \tag{4.3}$$

Prediction of result

Under the ideal circumstances of non-existent disturbances or miscalculations, the predicted input of a specific time from the predictive model should be equal to the input with dead time. This is best explained by the equations 4.4 and 4.5:

$$Y\mathrm{p}(s) = G(s) \cdot U(s) = e^{\theta s} \cdot Pn(s) \cdot U(s) = e^{\theta s} \cdot Y(s) \tag{4.4}$$

$$yp(t) = y(t + \theta) \tag{4.5}$$

In which the "Yp" and "yp" represent the feedback signal of the process model, "U" represents controller output and "Y" and "y" represent the actual output of the process. The variables "t" and "θ" stand for moments in time and dead time respectively.

If there are disturbances or other miscalculations, as there always are in practical usage, the dynamic between all values is shifted so that the disturbances are taken into account. The resulting dynamic of this shift should take the form of the equation 4.6 below:

$$yp(t) = y(t + \theta) + P_n[q(t) - q(t + \theta)] \quad (4.6)$$

In which ”q” represents the disturbance introduced into the system before entering the process. (Normey-Rico & Camacho, 2007, pp. 132-133)

Ideal Dynamic Compensation

The ideal output of the system is achieved by the acknowledgment that an ideal equivalent controller can exist and that it can be found using the equation 4.7 below for dead time compensation:

$$C'(s) = \frac{C(s)}{1 + C(s) \cdot G(s)} = G_n(s)^{-1} \quad (4.7)$$

With the ideal controller found, ideal output of the system can be estimated with uncertainties taken into account through equation 4.8 below:

$$y(t) = r(t - \theta) + P_n(t)[q(t) - q(t - \theta])] \quad (4.8)$$

Where “r” stands for reference value. The above mentioned ideal controller cannot be used in practice due to there always being minute miscalculations in the time constant, dead time or process gain. However it does show what a proper Smith Predictor and a controller for it can do in a system, providing an example to follow for performance. (Normey-Rico & Camacho, 2007, pp. 133-134)

4.2 Two Degree of Freedom Smith Predictor

The Smith predictor in some situations has to be changed to function in a situation it would otherwise fail catastrophically and to maintain desired performance, this results in modifications or combinations with other systems seeing use in place of the default Smith Predictor. One common and relatively simple modification is the Two-Degrees-of-Freedom Smith Predictor.

The Two-Degrees-of-Freedom Smith Predictor, abbreviated as 2DOF-SP, is a Smith predictor that has two-degrees-of-freedom to allow for a larger margin of error. This quality can be attained through a reference filter placed before the input of the predictor system or a feedforward function to counteract the effects of a controller deliberately tuned for a faster response. In this thesis the filter method for will be used. The equation that expresses the relationship between the input and output of the 2DOF-SP system is similar to the one by the normal Smith Predictor, but it is also affected by the filter placed before the controller models input for reference. The output of the 2DOF-SP is explained in equation 4.8 below:

$$2DOFSP(s) = \frac{C(s) \cdot P_{\mathrm{n}}(s) \cdot F(s)}{1 + C(s) \cdot G_{\mathrm{n}}(s)} \tag{4.3}$$

The filter itself is usually a transfer function that uses the de-numerator of the plant with a numerator designed to better fit the maximum expected difference of set dead time of the controller, as shown in equation 4.9:

$$F(s) = \frac{\alpha \cdot \Delta\theta + 1}{T \cdot s + 1} \tag{4.4}$$

Where ”α” is the 2DOFPID coefficient, which in this thesis is 1.7 during all simulations for the sake of consistency and maximum robustness, and “Δθ” is the expected range of changes in dead time. (Normey-Rico & Camacho, 2007, pp. 149-155)

4.3 Tuning of controller module

As with the changes brought with the Smith Predictor, the control module must be tuned accordingly due to changed conditions and the supposed elimination of dead time. This is due to the fact that the tuning rules elaborated are designed around the FOPDT model to take the delayed flow of results into account.

Matlab software-based tuning

The PI controller of the Smith Predictor is tuned through the same software as detailed before in the PID-controller sub-chapter 3.3 “Matlab software-based tuning” above.

Direct Synthesis

Direct synthesis tuning, also known as instantaneous response-tuning, is a PI and PID-controller tuning method for very low or nonexistent dead time processes such as ones which the Smith Predictor’s process model is able to provide. The controller values for this described response are depicted in table 7 below:

Table 7: Direct Synthesis method of tuning for PI controller values in systems with no dead time. (Smith & Corripio, 1985, p. 340)

DS-SP	Kc	Ti
PI	$\frac{T}{\alpha \cdot K}$	T

In which “α” is a coefficient has to be chosen, for the purpose of testing it is assumed to be 1.7 throughout all simulation environments for the sake of consistency and maximum robustness. This tuning rule is designed to respond quickly to any errors by prioritizing speed, resulting in what is usually the shortest rise time of most PI-based controllers and settle-time at ideal conditions in long dead time processes, which comes at the cost of being vulnerable to unknown dead time. (Smith & Corripio, 1985, p. 340)

Tuning for 2DOF Smith Predictor

The 2DOF-SP in this thesis will be used with an appropriately tuned PI controller to take advantage of the filter used. The controller tuning method accompanying this modification in thesis is the method suggested by Normey-Rico and Camacho detailed in table 8 below:

Table 8: PI tuning values for the controller module used in the 2DOF-SP. (Normey-Rico & Camacho, 2007, p. 157)

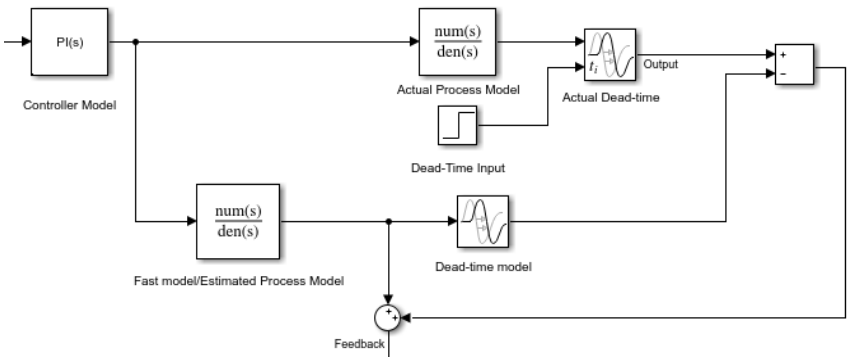
2DOF-SP	Kc	Ti
PI	$\frac{T}{K \cdot \alpha \cdot \Delta\theta}$	T

Where the "α" is the 2DOF-SP coefficient used in the filter mentioned above, for which the value is 1.7 throughout all testing environments for consistency and maximizing robustness. This is to best make use of the new resistance to changes and take into account the changes to reference value the filter introduces when tuning the controller.

4.4 Implementation of the Smith Predictor in Matlab-Simulink

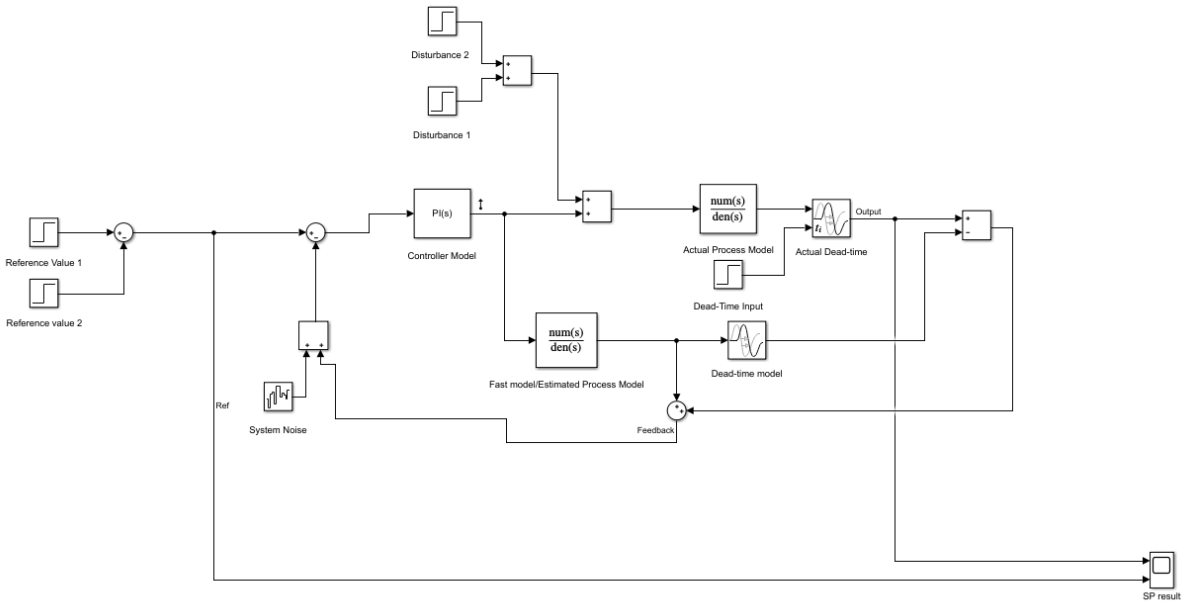
Despite there being a readily available Smith Predictor in the function block library, it lacks many features to be used as wanted, as it lacks readily available variants and their interchangeability. As a result, a Simulink model recreating a Smith Predictor was made using other functions available in the Simulink library. The general structure of the Smith Predictor used in Matlab-Simulink is depicted in Figure 6:

Figure 6: General structure of the tested Smith Predictor in Matlab Simulink.



The Smith Predictor structure depicted in Figure 6 above was added on the controller in the environment depicted in Figure 3 in sub-chapter 3.4, resulting in the testing environment shown in Figure 7 below:

Figure 7: General Template of the Smith Predictor testing environment in Matlab Simulink.



The PI controller is the controller module of the Smith Predictor, the P(s) transfer function is the actual process model or plant, transfer function G(s) is the process model, which is usually a copy of the plant in ideal conditions, and the transport delay is the dead time model of the prediction process. The controller, dead time input and the transfer functions were appropriately filled out according to the simulation environment. The filter for the 2DOF-SP was a transfer function placed at the input for the reference value before it was compared to the output.

5 BACKGROUND FOR MODEL PREDICTIVE CONTROL

The model predictive control, abbreviated as MPC, is an advanced control method heavily based on identified process models and possible knowledge of future disturbances and set-points for calculating an optimal control of a system. It was initially used for oil refineries and chemical plants in the 1980s before being implemented to other systems such as heating and climate control for buildings when capabilities of processing hardware rose while decreasing in price. As of 2011, it was the most popular advanced control technique used in a variety of industries. (Haber, 2011, pp. XII-XVI)

The simplicity in setup for general usage while providing possibilities to be tailored for each specific situation has made it a popular alternative in circumstances where there are a large number of manipulated and controller variables with shifting objectives. It can also function in simpler situations just as well, as it allowing the user to impose more constraints and its use of optimal control law also accommodates a higher level of control than many other control techniques.

5.1 Function of Model Predictive Control

The model predictive control uses three elements that can be changed to suit the situation. The system goes in cycles so that future outputs for a defined horizon are first predicted using known current and past feedback values, with future control signals being calculated using a determined criterion to fit with the trajectory of future values. The resulting control signal is then used to control the system before the next control signal is made using received feedback to repeat the cycle. (Normey-Rico & Camacho, 2007, pp. 273-275)

The consistent steps between all the Model Predictive control modules and most of their modifications are three: the prediction model, the objective function and obtaining the control law.

Prediction Model

In Matlab, the Prediction model uses a Kalman filter to predict the future behavior of a process using an estimate of the plant model. This is achieved by converting the processes input and output variables to a dimensionless form. These input and output variables are obtained by converting the given model to a linear time-invariant system state-space model. The LTI-system state-space model is then converted to discrete-time linear time-invariant system to separate objects using the controller sample time before replacing delay with discrete time poles. The general equation for the resulting prediction model after conversion are summarized in the equations 5.1 and 5.2 below:

$$x_{\mathrm{p}}(k+1) = A_{\mathrm{p}}x_{\mathrm{p}}(k) + B_{\mathrm{pu}}u(k) + B_{\mathrm{pv}}v(k) + B_{\mathrm{pd}}d(k) \tag{5.1}$$

$$y_{\mathrm{p}}(k) = C_{\mathrm{p}}x_{\mathrm{p}}(k) + D_{\mathrm{pu}}u(k) + D_{\mathrm{pv}}v(k) + D_{\mathrm{pd}}d(k) \tag{5.2}$$

In which ”x_p” is the state vector obtained after delay removal, “y_p“ is the vector for dimensionless plant output, u(k) is the dimensionless vector input for manipulated variables, v(k) and d(k) are also dimensionless vectors for measured and unmeasured disturbance. “A_p”, “B_{pu}”, “B_{pv}”, “B_{pd}”,” “C_p”, “D_{pu}”, “D_{pv}” and “D_{pd}” are matrix columns for the input and output of scale factors obtained from the constant zero-delay state-space matrices from the delay removal process. (Mathworks, 2021)

Objective function

The objective function provides a cost function to find the control law used, whilst using various values in its calculations that would have been dictated by the user. The general function for picking such a law within a SISO situation is in the form of equation 5.3:

$$J = \sum_{j=0}^{P} [(y(t+j|t) - y'(t+j)]\text{^}2 + \sum_{j=0}^{C} \lambda u \cdot [\Delta u^2(t+j)] \tag{5.3}$$

Where “y” is the predicted reference trajectory and “y’” is the predicted output. During the creation of the cost function for an MPC parameters for minimum, maximum and control horizons and coefficients must be set with a reference trajectory calculating how changes will happen in advance. Other values that must be set include the control horizon “C”, the prediction horizon “P” and maximum and minimum constraints for the picked values of u and y, as well as weighting values. (Haber, 2011, p. 7)

Obtaining Control Law

Quadratic programming for a linear quadratic estimation is used to obtain the ideal control law that will determine how the system will control the process. This is achieved by minimizing the objective function of the system and to find and use the ideal output value. This procedure can be summarized by the heavily simplified version of the cost function in equation 5.3 in the form of equation 5.4:

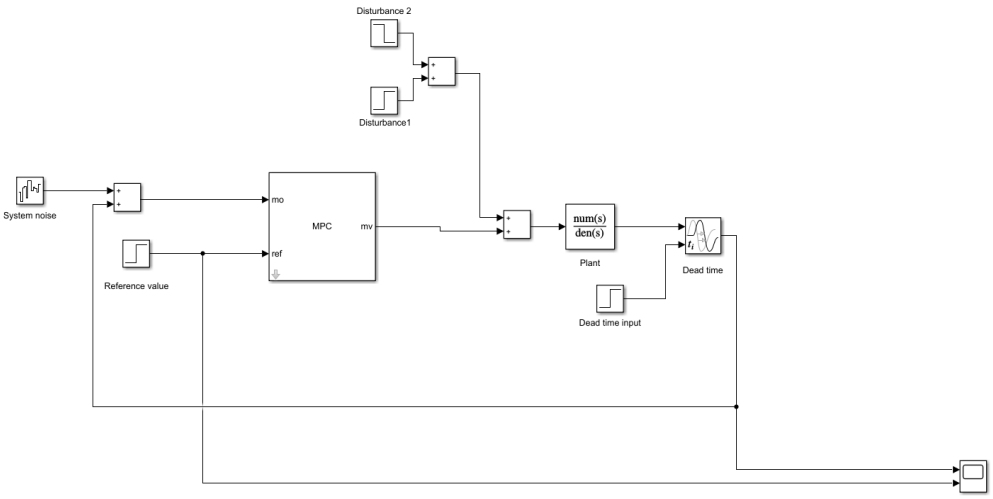
$$\min_{\Delta u(t)} J \tag{5.4}$$

This control law will dictate the output until a satisfactory new input is attained, where the MPC will then repeat the process detailed in the above descriptions for each step of the MPC. (Geyer, 2016, p. 15)

5.2 Implementation of Model Predictive Control in Simulink

Simulink has a library of Model Predictive Controllers available for use and simple implementation available for download (Xue & Chen, 2013, p. 157). The testing environment for Model Predictive Controllers in this thesis is displayed in Figure 8 below:

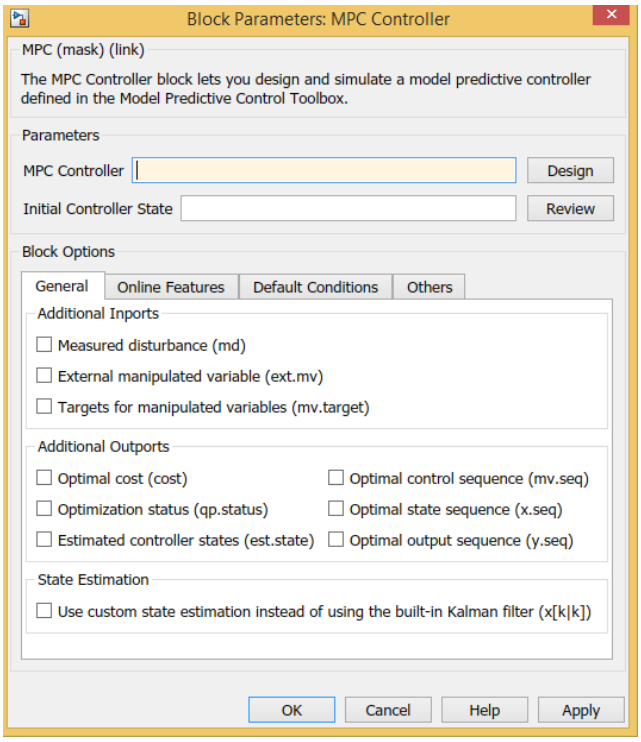
Figure 8: Simulation environment template for MPCs in Matlab-Simulink



The default inputs of the MPC were model output “mo” that was used in a feedback loop within and the reference value “Ref” of the controller. The controller default output was the manipulated variable output “mv”, which was the output of the controller.

Like the PID-controller function block, the MPC function block has a detailed parameter window to access and change various controller variables. In the block parameter window the user can add or change inputs, outputs, parameters, default conditions, weights constraints, initialization and changing state estimation methods by changing inputs as well as setting prediction, sample-time and control horizons. The MPC used in Simulink must have an MPC instance created, designed and ran in a default Matlab workspace before it can be implemented onto Simulink. The default MPC Block Parameter window in Simulink is shown in Figure 9 below:

Figure 9: MPC window in Matlab-Simulink for version R2020b.



The user can also make use of the dedicated MPC tuner program “MPC designer” for detailed and easily accessible MPC tuning. The “MPC designer” program allows for quick changes in weighting, prediction time, sampling time and it can print the resulting MPC in Matlab-script for easier storage and faster editing in later use. It can also be used to change output or input limits and to create a scenario to foresee tuning results. (Mathworks, 2021) In this thesis project the “MPC designer” software was used to fine-tune the MPC used for faster State-estimation and robustness.

6 SIMULATION

In order to actually test the viability of the control systems used in this thesis project and to ascertain their actual capabilities as controllers in an economic and repeatable manner, their performance in a realistic and general environment had to be simulated. The Matlab-Simulink software can simulate various environments mathematically within an easy-to-use and set up user interface, with a ready library of various control tools and mathematical functions.

In whatever simulation environment, the control systems evaluated have been tested to find the following information for performance alongside general behavior in a SISO context within processes that can be summarized using a FOPDT-model:

- Rise-time of the controller meeting the desired value with or without unknown dead time.
- Settling-time, as in the point where the system reaches the reference value with a deviation of at most 0.01 percent of the first digit of the referential value afterwards outside of disturbances, within systems with and without unknown dead time.
- Disturbance rejection where the time between the rise or fall of the spike and it meeting the reference value, or the closest value to it in either circumstance.

The tests for each environment were repeated with dead time different to the initial value and their relation to the time constant for cross-examination. Each environment for all controllers was retested with varying miscalculations ranging from -7% to +10% of the process model in its time constant and process gain along with band-limited white noise for analyzing controller behavior in more realistic circumstances. Dead time was different in each Simulation environment to test varying dead times in comparison to the time constant.

The performance of the control systems tested was evaluated by the length of rise-time, settle-time, disturbance rejection time and degree of overshoot. The results were generally be measured for the default conditions of the controller and their tuning methods besides halving the controller gain for ZN-PI, ZN-PID and CC-PID controllers. Modifications like limiters or the white noise models for MPCs were withheld during these measurements,

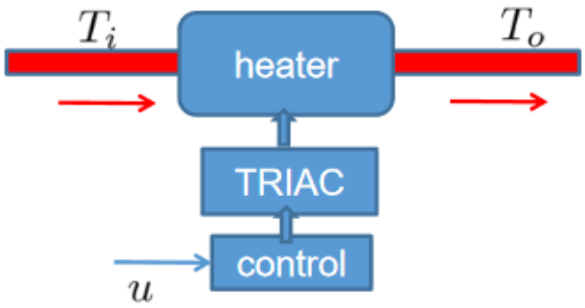
with the exception of Simulation environment 1 having output limiters applied. The effects and application of these modifications are detailed in 6.6: “Minor modifications”.

The PID-controller values for these situations formed with tuning methods were based on the book “PID tuning rules 2nd edition”, these values are included into Appendix 1 in the form of Windows Excel Sheets along with values used for the Model Predictive Controller parameters used in each Simulation.

6.1 Simulation environment 1

The first test environment for the control systems was water heater over the course of 600 seconds with an initial intended value of 14° Celsius that was changed to a value of 10 in 5 minutes. The relationship between the controller and the heater is summarized in Figure 10 below:

Figure 10: Diagram of the controller and the heater of Simulation Environment 1 (Normey-Rico J. E., 2019, p. 40)



The time constant of the process was 13.1 seconds, with a process gain of 18.7° Celsius and a dead time of 8 seconds that was increased to 10 seconds at 13.1 seconds when changing dead time, simulating a processing error slowing down the detection of temperature. The shift was done at that point in time to study the system’s ability to adapt to changing dead time while it is still reaching for the reference value. (Normey-Rico J. E., 2019, p. 40) The above dynamics can be used to create an FOPDT-model resembling with equation 6.1:

$$Pn(s) = \frac{18.7}{13.1s + 1} \cdot e^{-8s} \tag{6.1}$$

The disturbance in this test was a sudden rise in the controller input value of 0.15 at 110 seconds and a disturbance of 0.35 added at 400 seconds to the input of the process, simulating decay in measurements, pumps or the heating system. Lastly, a higher output limit of 1 and a lower output limit of 0 were placed on all the controllers for safety.

In table 9 the resulting rise-time, settling-time and disturbance rejection from tests are registered and then elaborated on individually later in this sub-chapter:

Table 9: Results for simulation environment 1 without model inaccuracy.

Test 1-1	Unchanging dead-time				Change in dead-time introduced at 13.1s			
Method/Time(s)	Rise Time	Settle time	Dist.Rej1	Dist.rej2	Rise Time	Settle tim	Dist.Rej1	Dist.rej2
ZN PI	-	-	-	-	-	-	-	-
ZN PID	48.000	107.138	81.651	88.991	39.000	109.725	75.229	77.982
CC PI	49.500	105.303	29.358	64.220	55.596	271.193	26.248	73.945
CC PID	108.055	108.055	80.734	94.294	106.972	106.972	77.872	86.239
SIMC PI	21.917	55.761	62.275	71.560	20.550	20.550	49.541	58.171
Matlab Softwate tuned PI	132.826	132.826	84.294	100.917	117.982	117.982	42.202	92.661
2DOF PID Taguchi	40.860	-	20.991	55.963	46.422	-	-	-
2DOF PID Terauchi 20%	72.000	72.000	64.110	77.700	56.100	109.000	27.100	73.945
2DOF PID Terauchi 0%	114.000	114.000	118.734	128.440	95.229	115.000	105.229	126.606
SP Matlab Software tuned PI	18.800	48.000	80.900	80.900	18.100	108.653	27.979	45.596
SP DS tuning	85.000	85.696	82.990	92.000	-	-	-	-
2DOF SP	104.200	104.200	88.700	100.297	101.000	101.000	82.700	99.300
MPC	19.000	19.000	84.10	96.28	-	-	-	-

The controllers tested in this environment were also been tested with a +3% miscalculation in process parameters for tuning, the written results were registered without noise due to it making exact measurements difficult. These measurements are contained in table 10 below:

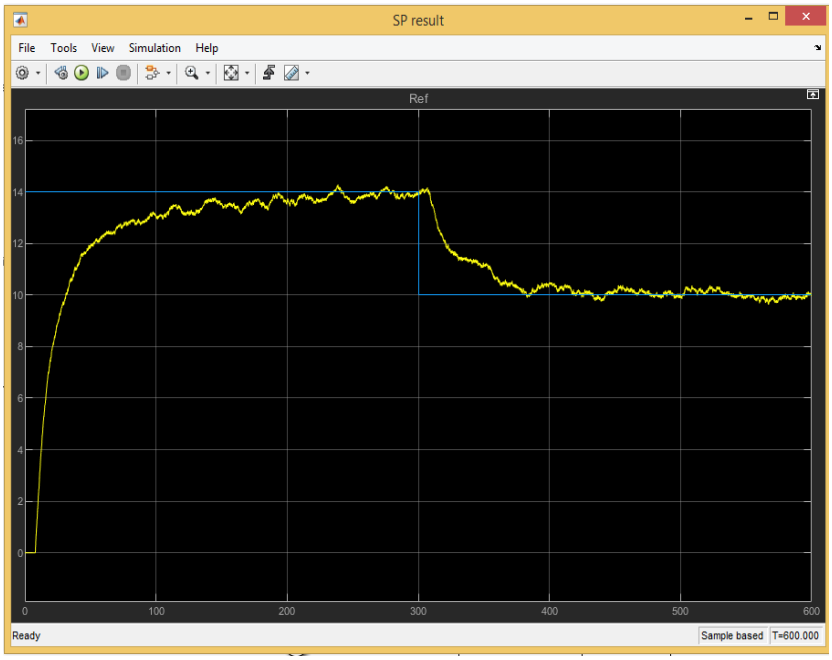
Table 10: Results for simulation environment 1 with model inaccuracy.

Test 1-D	Unchanging dead-time				Change in dead-time introduced at 1.3s			
Method/Time(s)	Rise Time	Settle time	Dist.Rej1	Dist.rej2	Rise Time	Settle time	Dist.Rej1	Dist.rej2
ZN PI	-	-	-	-	-	-	-	-
ZN PID	-	-	-	-	-	-	-	-
CC PI	49.400	125.043	26.400	78.783	23.400	-	26.200	78.500
CC PID	-	-	-	-	-	-	-	-
SIMC PI	20.920	57.217	92.870	103.304	19.920	110.000	46.800	106.000
Matlab Softwate tuned PI	111.000	128.000	111.534	128.269	75.043	115.000	39.800	130.000
2DOF PID Taguchi	129.306	129.306	106.352	122.052	122.124	122.124	105.316	119.979
2DOF PID Terauchi 20%	69.000	69.000	62.303	77.789	33.700	110.000	24.900	73.025
Terauchi 2DOF PID 0%	-	-	-	-	-	-	-	-
SP Matlab Software tuned PI	18.100	44.332	81.865	95.109	17.560	110.725	27.900	81.865
SP DS tuning	80.601	80.601	81.382	92.000	-	-	-	-
2DOF SP	100.290	100.290	85.600	96.500	98.000	99.326	79.400	99.326
MPC	17.760	65.540	82.665	94.200	-	-	-	-

PID-controllers

Both tunings of the default PID-controllers worked to an extent during nominal and disturbance-free conditions, both having generally similar performance, with both having a relatively similar settling time and disturbance rejection capabilities. However both were limited by noise and slowed down significantly, affecting their ability to address disturbances and reaching the reference value, as seen in Figure 11 below:

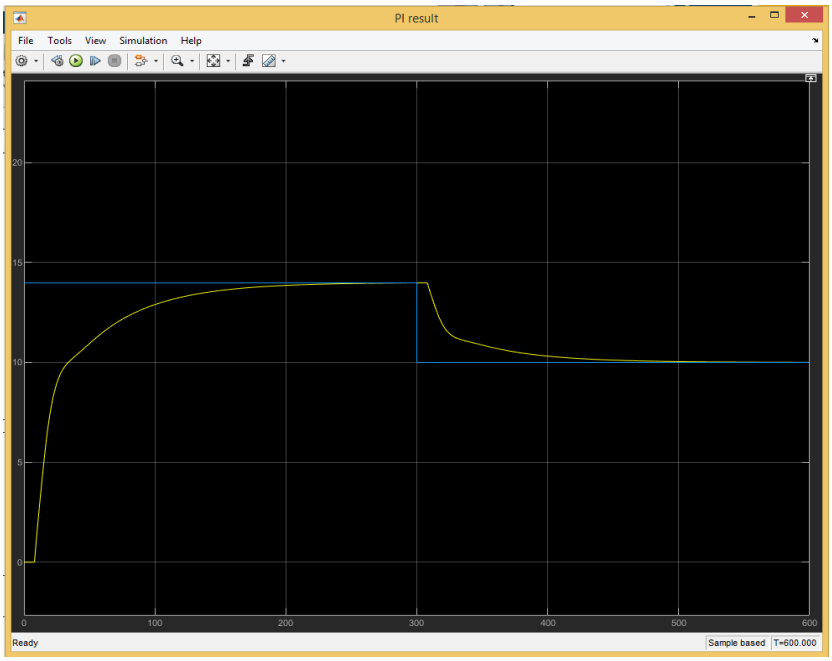
Figure 11: Simulation result of Ziegler Nichols PID-controller with disturbance in Matlab-Simulink with model miscalculation and noise.



The ZN-tuned PID-controller had a faster rise time of 48 seconds before settling at 107 seconds after an extremely small overshoot much slower disturbance rejection speed than the CC PID-controller, with a rise and settle time of 108 seconds. However the noise in the system were significantly disruptive to the PID controllers, rendering results in a more realistic setting with disturbances and measurement noise unreliable due to not reaching an acceptable value long into the simulation time given.

PI controller tunings provide more varied results depending on the tuning method and were more resistant to noise due to lacking the Derivative-element. As shown in Figure 12 below, when using the ZN method it proves itself too slow for result, making it unobservable in this situation at even ideal conditions with no miscalculation or noise:

Figure 12: Simulation results of a ZN-tuned PI controller in Simulation environment 1

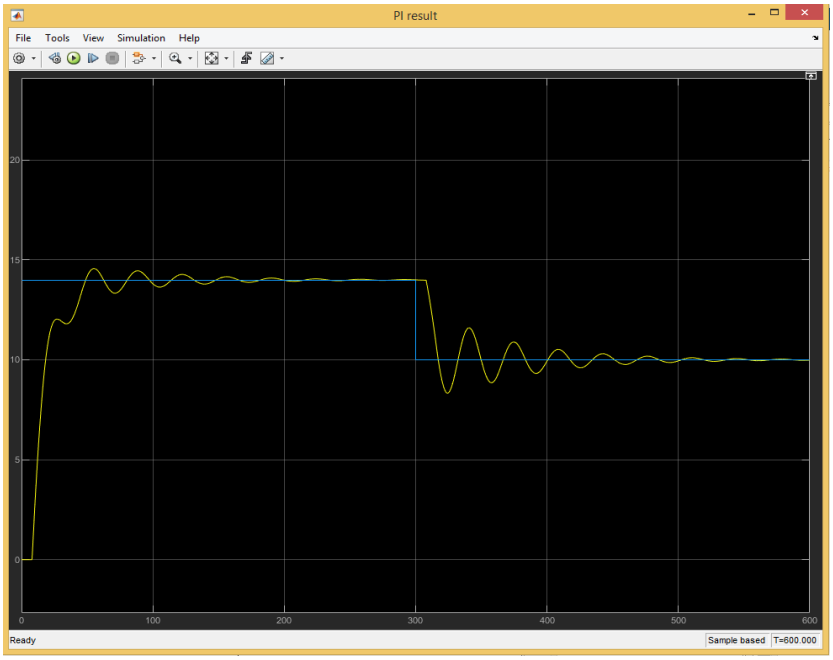


This is contrasted by the performance of the CC and the Skogestad-Suggested Lambda tuning rules that manage to reach the referential value at around 1 or 2 minutes and settling at 2 minutes.

The Cohen and Coon tuning rules provide a faster response that would usually result in multiple overshoots to meet the initial value as fast as possible and undershoots to fix the first overshoot, but due to the output limit it results in a slower rise with a dip before meeting the reference value. This makes the CC-tuned controller more stable system with lower overshoots, but it is outdone by the SIMC-tuned PI controller in its settling time and rise time. However the CC-tuned PI controller is considerably faster to address disturbances in comparison to an SIMC controller by a large margin.

Of the 2DOF PID-controllers only the Taguchi tuning rules were shown to work, but the controller settles slowly as it oscillates. The Terauchi tuning rule for 0% overshoot proves too slow even in nominal conditions, but when tuning it for a 20% overshoot it results in better performance than the Taguchi tuning rules, having a less overshoot and settling faster, despite a longer rise-time. The performance of the Taguchi tuning for this controller can be seen in Figure 13 below:

Figure 13: Simulation result of the 2DOFPID tuned with Taguchi Arai's tuning rules in Simulation environment 1

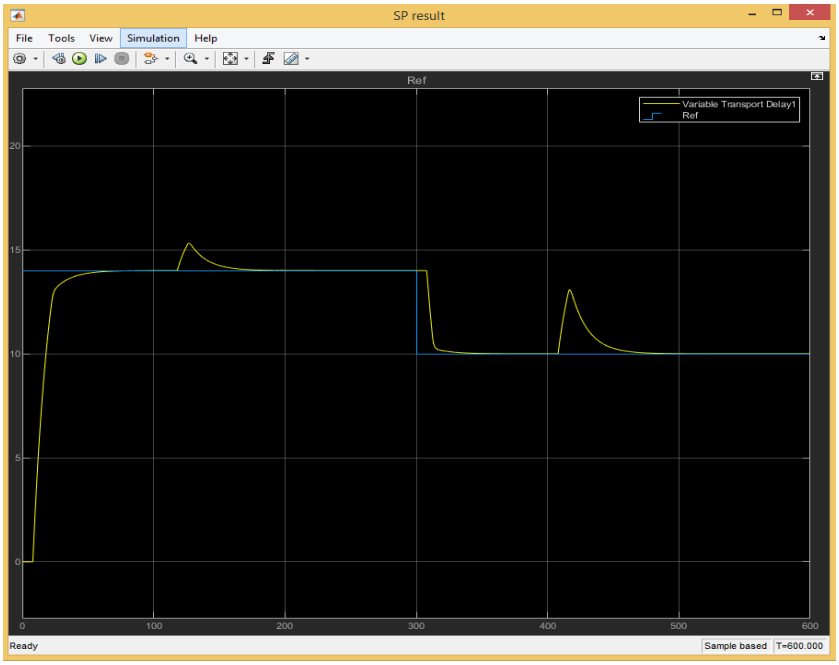


In all cases a 2 second miscalculation resulted in faster corrections, with the system reaching the referential value faster, accompanied with faster disturbance rejection and larger overshoots and more overshoots in all cases. This can be attributed to the controllers correcting values too quickly before they can slow down to not overcorrect as a result of controller values not being properly decreased to receive feedback before requiring another correction.

Smith Predictor

The Smith predictor in both software tuning and Direct Synthesis resulted in relatively slower results in comparison to other simulation, as shown in Figure 14 below:

Figure 14: Simulation result of a Smith Predictor tuned with Direct Synthesis tuning rules in Simulation environment 1



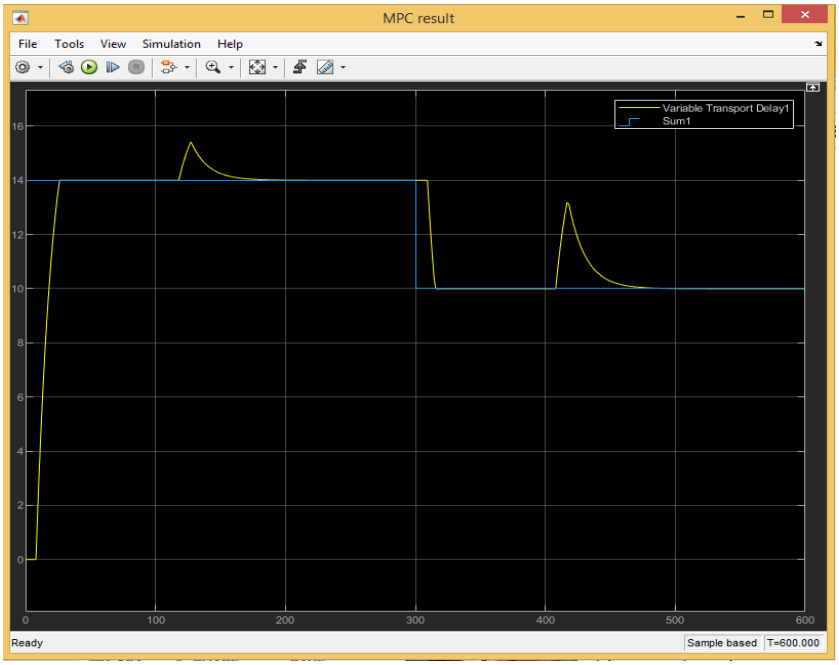
The direct synthesis tuning of the controller module results in it meeting and settling at the reference value at roughly 1 and a half minutes. It had a slightly slower disturbance rejection than the CC-tuned PI controller, but it had faster disturbance rejection than the SIMC-tuned PI controller. When using software tuning results were often superior to Direct Synthesis, but there were noticeable overshoots and undershoots when correcting disturbances and meeting the referential value. 2DOF-SP had results similar to a slower normal controller without a prediction function, resulting in slower results than the default SP, but with no overshoot.

2 second miscalculations result in slowly growing oscillations in the DS-SP, while the 2DOF-SP and Software-SP display similar results to PID-controllers, as in being less stable but with a faster response.

Model Predictive Control

The default Model Predictive had a fast response and settling time in comparison to other controllers despite limiters with a disturbance rejection time of 84 and 96 seconds, as seen in Figure 15 below:

Figure 15: Simulation result of a MPC within Simulation environment 1 without model miscalculation.

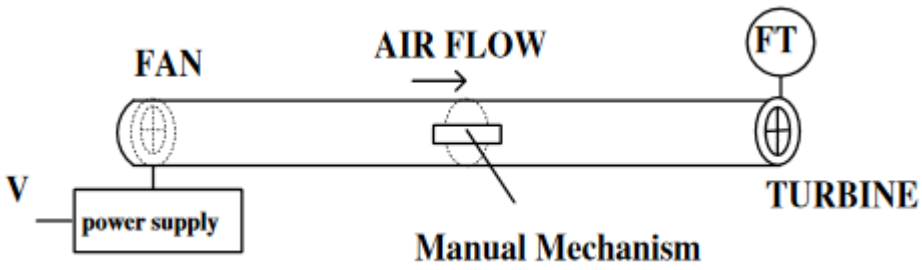


A 2 second dead time miscalculation results in increasingly large and uncontrollable oscillations in output if it is exposed to any significant change in reference value or disturbance, which it does when the point where dead time is before it can reach the reference value.

6.2 Simulation environment 2

In this simulation environment the controller of cold air in a central air conditioner system is recreated by placing a voltage controller in a windpipe that uses a fan to feed air to turn a turbine, the turbine then turns the flow of air into power that it supplies elsewhere. The power being supplied by the controller to the fan is supposed to the system is supposed to be in the range 0 to 5 volts, being the hard limit of power being provided. A drawing of the simulation environment is shown in Figure 16:

Figure 16: Concept drawing for Simulation Environment 2 (Normey-Rico & Camacho, 2007, p. 158)



The acceptable voltage of the controller is changed from the default 1.5V to 3.5V at 75 seconds over the course of 150 seconds. The process itself had a process gain of 1.02 V, a time constant of 1.7 seconds with a dead time of 8.2 seconds. (Normey-Rico & Camacho, 2007, p. 158) With the above information the process can be summarized with the below transfer function in equation 6.2:

$$Pn(s) = \frac{1.02}{1.7s + 1} \cdot e^{-8.2s} \tag{6.2}$$

The initial rise time, settling time and disturbance rejection of the system detailed above. It has to be noted that disturbance rejection is tested with a disturbance of 0.225 of the reference value at 30 seconds and a 0.525 of the initial reference value at 97 seconds. For the sake of comparing the destabilizing effect of unknown dead time with Simulation environment 1, the change in dead time for this Simulation environment is also 2 seconds.

Like in sub-chapter 6.1 “Simulation environment 1”, the results of each controller tested in this Simulation environment are detailed in table 11 below, before the actual behavior of each method is detailed in later in this sub-chapter:

Table 11: Results for simulation environment 2 without model inaccuracy. Rise Time and Settle time are without dead time being taken into account.

Test 2-1	Unchanging dead-time				Change in dead-time introduced at 1.3s			
Method/Time(s)	Rise Time	Settle time	Dist.Rej1	Dist.rej2	Rise Time	Settle time	Dist.Rej1	Dist.rej2
ZN PI	-	-	-	-	-	-	-	-
ZN PID	-	-	-	-	-	-	-	-
CC PI	13.670	52.156	24.135	24.813	11.766	-	17.668	-
CC PID	-	-	-	-	-	-	-	-
SIMC PI	22.473	56.510	33.000	33.025	18.494	-	29.288	33.245
Matlab Softwate tuned PI	12.450	60.100	22.857	22.043	10.860	-	22.531	16.670
2DOF PID	-	-	-	-	-	-	-	-
SP Matlab Software tuned P	1.227	9.889	16.881	19.800	-	-	-	-
SP DS tuned PI	13.612	13.612	21.700	24.174	-	-	-	-
2DOF SP	13.548	13.548	20.540	23.459	8.490	71.711	22.198	20.239
MPC	4.521	4.521	19.104	20.852	-	-	-	-

In this Simulation environment all the control methods have additionally been tested with a -5% model miscalculation, with the results listed in table 12 below:

Table 12: Results for Simulation environment 2 with a model inaccuracy of -5%.

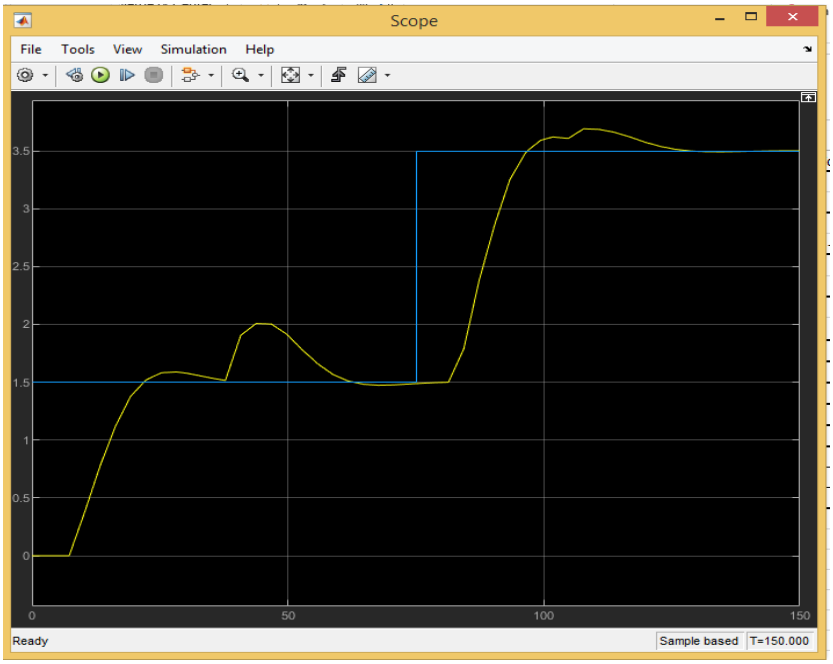
Test 2-D	Unchanging dead-time				Change in dead-time introduced at 1.3s			
Method/Time(s)	Rise Time	Settle time	Dist.Rej1	Dist.rej2	Rise Time	Settle time	Dist.Rej1	Dist.rej2
ZN PI	-	-	-	-	-	-	-	-
ZN PID	-	-	-	-	-	-	-	-
CC PI	14.120	50.157	24.700	24.100	9.720	-	20.600	21.570
CC PID	-	-	-	-	-	-	-	-
SIMC PI	11.780	-	32.735	34.200	9.720	-	27.483	30.750
Matlab Softwate tuned PI	13.090	46.310	23.600	23.360	11.160	-	21.106	20.820
2DOF PID	-	-	-	-	-	-	-	-
SP Matlab Software tuned	18.833	18.833	28.455	26.711	-	-	-	-
SP DS tuned PI	23.711	23.711	22.572	31.589	-	-	-	-
2DOF SP	1.700	19.300	27.000	30.800	8.790	70.898	21.385	20.720
MPC	20.635	20.635	24.793	27.607	-	-	-	-

The measurements in Table 12 were registered without noise taken into account due to the low scale of the environment making exact measurements difficult.

PID

The result of the default PID-controller configuration and the 2DOFPID-controller, regardless of tuning, results in immeasurable results, taking too long to reach the reference value in over half the simulation time or anything close to it for proper results or analyzable and consistent behavior regarding the effects of disturbance. As a result only the tested PI controller configurations are analyzed.

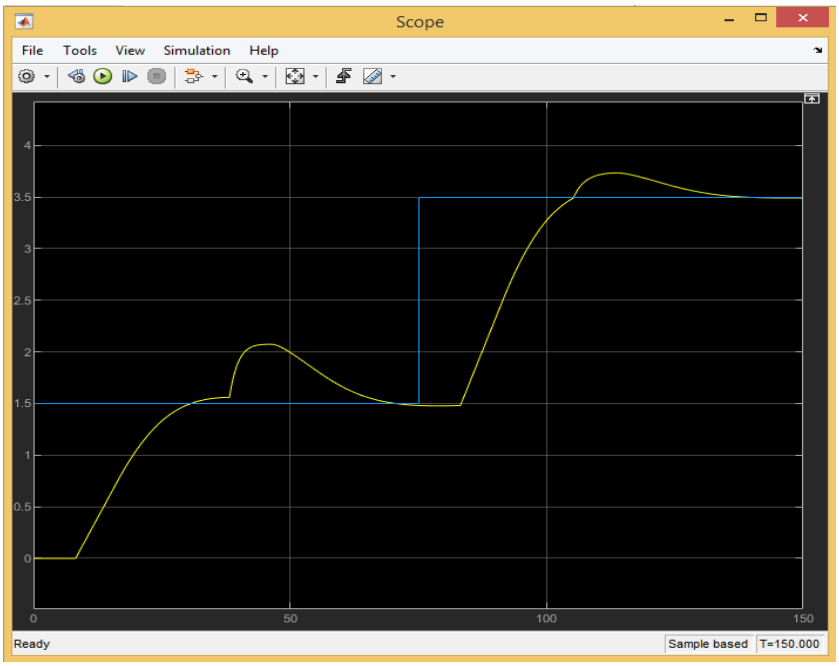
Figure 17: Simulation Result of CC- tuned PI controller in Simulation environment 1 without miscalculation.



As seen in Figure 17 above, due to a lack of a limiter and a longer relative dead time, the response of the controller when tuned with Cohen and Coon-tuning rules results in it meeting the value at 14 seconds with only one overshoot and an almost unnoticeable undershoot that was corrected and settled at 50 seconds, with a rejection response to disturbances within 25 seconds.

The SIMC-PI controller had a smaller overshoot than the CC-PI controller, but the rise time was longer than the CC-tuned PI controller with only a 4 second longer settling time and a noticeable slower response to disturbance, as seen in Figure 18 below:

Figure 18: Simulation of a PI controller using SIMC tuning rules in Simulation environment 2



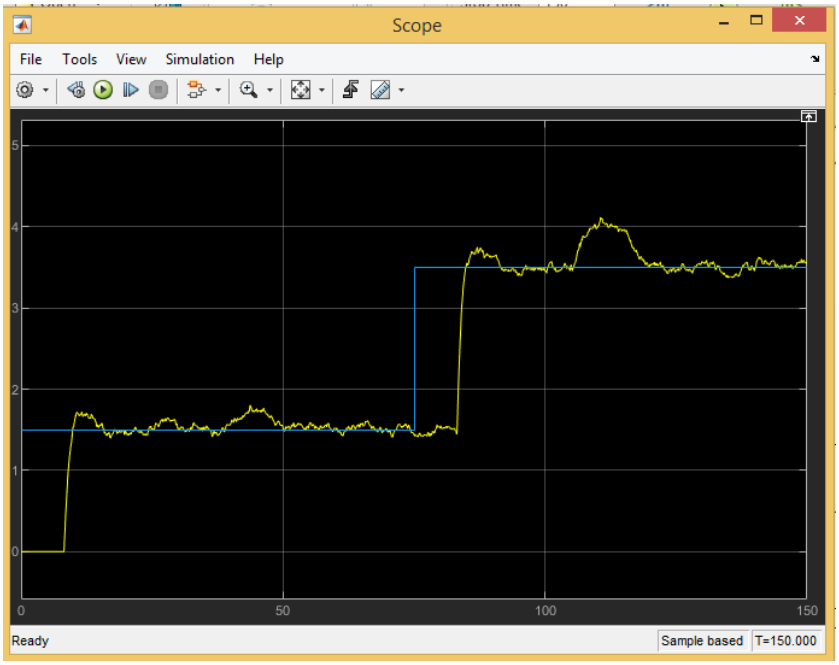
A Matlab software-tuned-PI controller attains robust solution results with a faster-rise time and disturbance rejection than other methods at the cost of a longer settling time.

Like in Simulation environment 1, a 2 second dead time change at 1.7 seconds causes most methods controllers achieve a larger overshoot, more undershoots and a longer or unreachable settling time in the simulated timeframe. The only benefits were a shorter rise time and a faster disturbance rejection with noticeable undershoots. The controllers generally succeed in not oscillating despite a larger relative error in dead time when comparing it to the time constant.

Smith Predictors

Smith Predictors provide a relatively fast, but controlled result in even non-nominal conditions in this simulation environment, as seen in Figure 19:

Figure 19: Simulation result of Smith Predictor Controller using Direct Synthesis tuning in Simulation environment 2 with a model miscalculation of -5%.



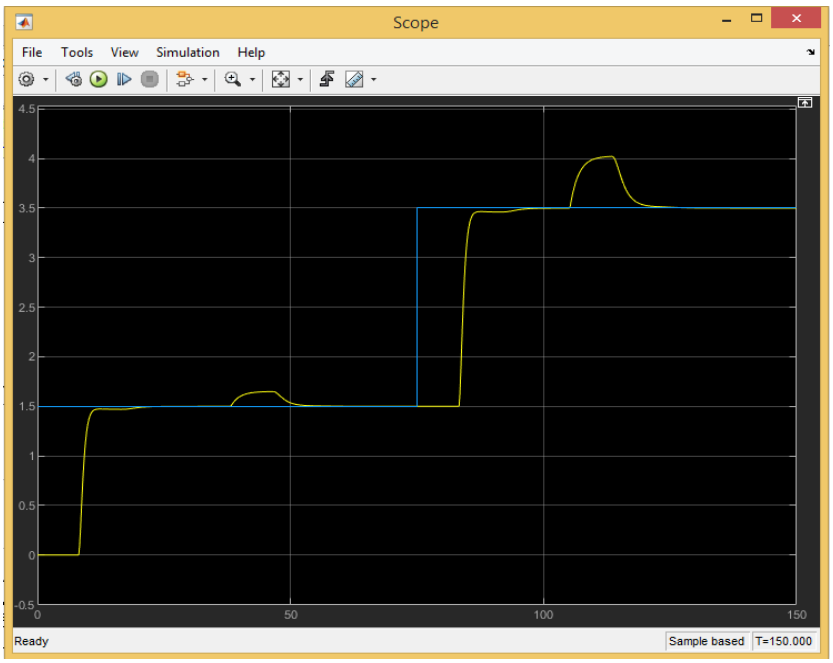
When using Direct synthesis there was no overshoot in nominal conditions while having a significantly longer rise time when exposed to a -5% process model inaccuracy, it managing to reach and maintain the referential value at 22 seconds, being much faster than the Cohen and Coon method to reach settling-time. It also had a shorter disturbance rejection period than the CC-tuned PI controller under ideal circumstances, but only by a degree of deciseconds and that speed was lost under a -5% process miscalculation. When tuned with Matlab software the controller provides superior performance to Direct Synthesis-tuning at the cost of a slightly larger overshoot, reaching the reference value at 10 seconds and maintaining it at 17-to-20 seconds. The 2DOF-SP had a similar, if better performance to the default SP-DS configuration, generally being slightly faster due to the shorter time constant resulting in a less obstructive filter and faster controller.

When exposed to a 2 second difference of dead time at the instant of 1.7 seconds, the DS-SP oscillates, the Matlab software-tuned Smith Predictor also starts to oscillate wildly and the 2DOF-SP provides only a less acceptable performance than when in ideal conditions, but it becomes slightly faster with a larger degree of overcorrection.

Model Predictive Control

Model Predictive Control had a better performance compared to the Smith Predictor tuned using Direct Synthesis, with the fastest rise time and settle time than all other methods with minimal overshoot and undershoot. It manages to deal with disturbances in 20 seconds. A -5% inaccuracy results in a noticeably longer rise- and settling- time and a slower disturbance rejection response that can still be considered superior overall to the SP configurations tested, as seen with Figure 20 below:

Figure 20: Simulation result of MPC in Simulation environment 2 with a -5% miscalculation.

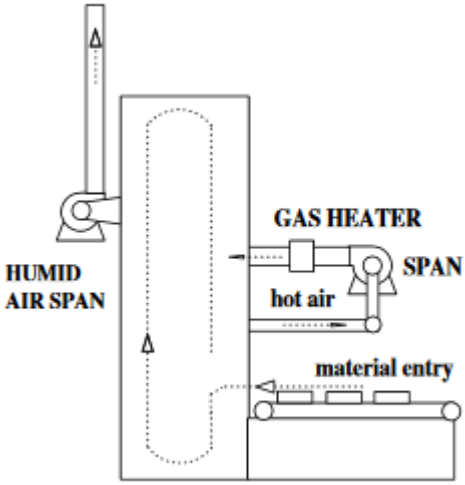


Like with Simulation environment 1, a dead time miscalculation of 2 seconds results in uncontrollable oscillations in this simulation environment. With the Simulation results for both environment 1 and 2 in mind, it can be concluded that a significant error in comparison to estimated either dead time or the time constant results in the controller creating oscillation.

6.3 Simulation environment 3

In this simulation environment a controller is managing the flow of hot air of a ceramic drier with a differing time constant and a dead time dependent on the temperature of operation, where in this case the wanted temperature is 110° Celsius. The general function of the heater is displayed in Figure 21 below:

Figure 21: Figure representing the general function of the ceramic dryer. (Normey-Rico & Camacho, 2007, p. 124)



In this simulation environment the process designed for minimum temperature is used due to the dead time of the model being over twice the time constant and the focus of this thesis being dead time. At the temperature of 110° Celsius the dead time is approximately 154 seconds, the process gain of the air-to-hot gas relationship is 0.84° Celsius and the time constant is 70.2 seconds. (Normey-Rico & Camacho, 2007, pp. 122-124). The above factors result in an FOPDT model in the form of equation 6.3:

$$Pn(s) = \frac{0.84}{70.2s + 1} \cdot e^{-154s} \tag{6.3}$$

A Disturbance step simulating issues with the heater resulting in poorer heating that decreases the heat in the air controlled by the controller, affecting control output in steps with the amounts of -10 and -20 at 110 seconds and 400 seconds respectively. The selected dead time error for this system was 38.5 seconds.

As with the previous sub-chapters 6.1 “Simulation environment 1” and 6.2 “Simulation environment 2”, results of the methods tested that are detailed in later subchapters have been written down in table 13:

Table 13: Results for Simulation environment 3 without model inaccuracy.

Test 3-1	Unchanging dead-time				Change in dead-time introduced at 70.2s			
Method/Time(s)	Rise Time	Settle time	Dist.Rej1	Dist.rej2	Rise Time	Settle time	Dist.Rej1	Dist.rej2
ZN PI	-	-	-	-	-	-	-	-
ZN PID	-	-	-	-	-	-	-	-
CC PI	190.500	1497.319	433.000	438.326	184.800	2535.078	443.165	501.635
CC PID	-	-	-	-	-	-	-	-
SIMC PI	421.060	1461.000	676.000	675.300	350.500	1570.500	619.093	621.409
Matlab Software tuned PI	279.338	1043.602	548.681	533.290	248.800	2049.706	564.897	520.393
2DOF PID	2708.545	2708.545	414.246	456.000	2107.095	2107.095	467.354	448.000
SP Matlab Software tuned PI	62.000	370.510	561.596	629.714	62.500	-	-	-
SP DS tuned PI	10.240	10.240	468.144	551.000	-	-	-	-
2DOF SP	613.000	613.000	729.000	729.000	187.200	1257.500	447.500	447.500
MPC	14.916	45.204	517.285	607.207	-	-	-	-

The degree of model inaccuracy tested for this simulation environment is a +7% miscalculation of process gain and time constant. The simulation results for this environment with model miscalculation were contained in table 14 below:

Table 14: Results for Simulation environment 3 with a model inaccuracy of +7%.

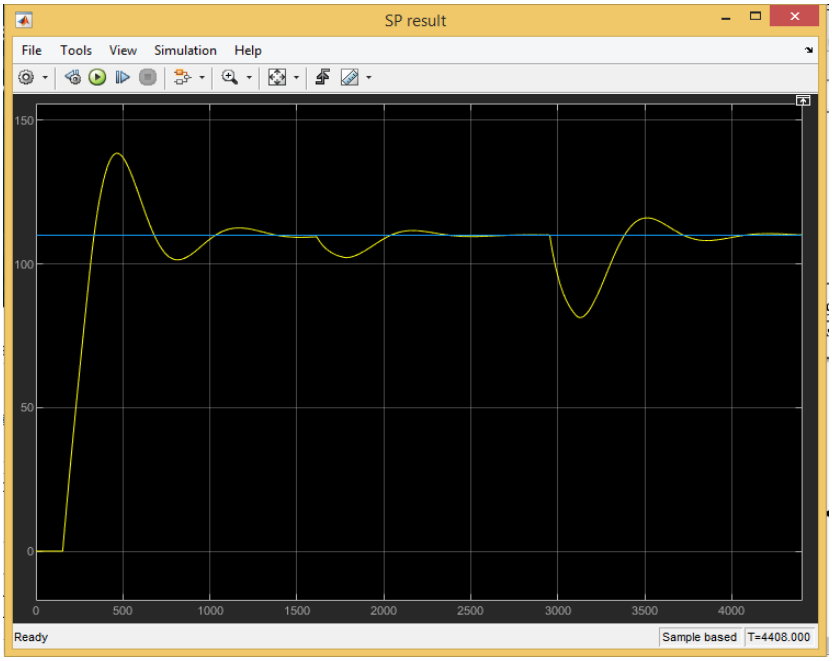
Test 3-D	Unchanging dead-time				Change in dead-time introduced at 70.2s			
Method/Time(s)	Rise Time	Settle time	Dist.Rej1	Dist.rej2	Rise Time	Settle time	Dist.Rej1	Dist.rej2
ZN PI	-	-	-	-	-	-	-	-
ZN PID	-	-	-	-	-	-	-	-
CC PI	180.400	1917.548	274.000	276.000	177.800	3107.500	231.000	265.000
CC PID	-	-	-	-	-	-	-	-
SIMC PI	360.000	1493.702	462.000	463.000	319.000	2588.990	406.683	410.000
Matlab Software tuned PI	247.800	1451.317	355.000	354.000	235.700	2546.606	336.500	319.500
2DOF PID	2426.163	2426.163	476.827	471.000	946.587	2747.933	477.500	444.500
SP Matlab Software tuned PI	59.700	698.990	227.462	216.865	-	-	-	-
SP DS tuned PI	43.500	720.183	216.865	213.894	-	-	-	-
2DOF SP	197.000	871.000	312.000	325.000	174.800	1667.500	241.000	247.000
MPC	49.699	414.508	378.969	377.839	-	-	-	-

Like in sub-chapter 6.1 “Simulation Environment 1” and sub-chapter 6.2 “Simulation environment 2”, noise was not included in result measurements listed in table 14 due to measurement difficulties, with the effects of which will stated when analyzing general controller behavior.

PID

Like with Simulation environment 2, the default PID-controllers and the ZN-tuning of PI controllers were too slow, resulting in them not reaching the referential value under any circumstance. Other PI controller tuning methods had varied results as shown in Figure 22 below:

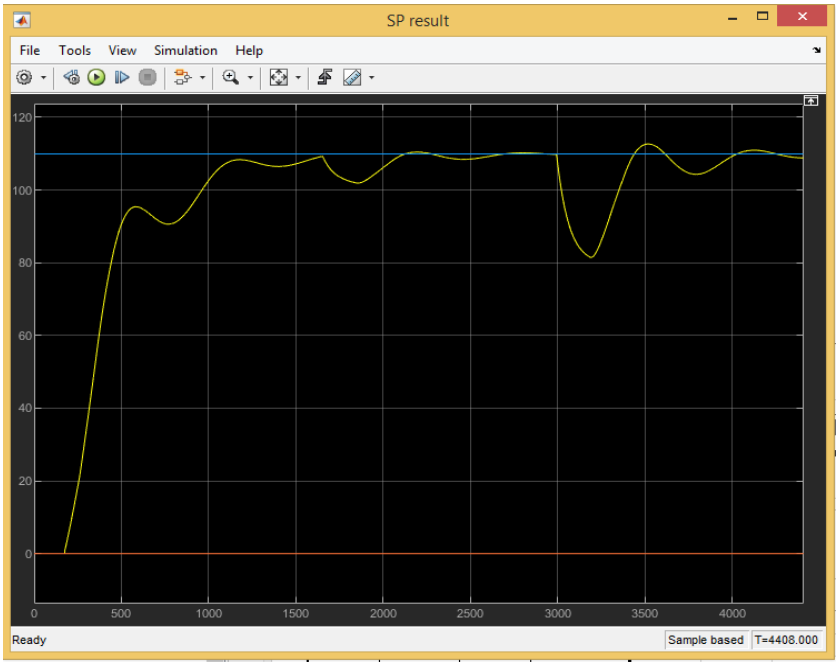
Figure 22: Simulation result of a CC- tuned PI controller in Simulation environment 3 with a model inaccuracy of +7%



The CC-tuned PI manages to meet the intended value at 190 seconds with multiple, but continuously lesser, overshoots and undershoots for 25 minutes with an average disturbance rejection of 7 minutes. Skogestad's suggestion for Lambda tuning of the controller results in it meeting the reference value at 7 minutes with an overshoot of slightly less than 10% of the reference value, settling at 24 minutes and disturbance adjustment being over 10 minutes. Software-based tuning results in the fastest rise time of the controllers of 5 and a half minutes, settling at roughly 16.5 minutes and manages to adjust to disturbances with very small overshoots in 8 and a half minutes.

Of the 2DOF PID-controller tuning rules only the Taguchi tuning rules produce a usable result, with it reaching the reference value at 46.5 minutes and having a disturbance rejection time of 428 seconds and 430 seconds. As seen in Figure 23 below, the controller was unable to reach the set point value with the negative disturbances:

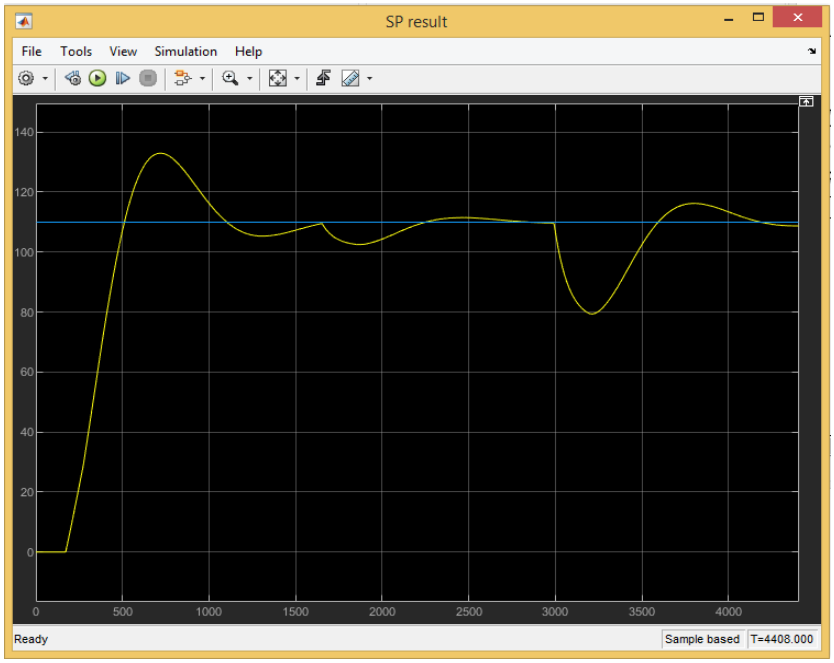
Figure 23: Simulation result of 2DOFPID tuned using Taguchi tuning rules in Simulation environment 3 without model miscalculations and a dead time error of 38.5s



When exposed to noise this tuning was significantly less usable in a similar manner as the default PID configuration, as it was unable to reach the reference value even with a +7% model miscalculation. This can be attributed to an issue PID-controllers face with the D-element becoming problematic when exposed to significant noise, slowing down controller response considerably and in extreme cases rendering them unable to deal with disturbances.

Under a 38.5 seconds of change in dead time, settling time was dramatically increased for all controllers, but rise time and disturbance rejection was fastened at the cost of multiple overshoots and undershoots as shown with the result of the SIMC tuned PI controller in Figure 24 below:

Figure 24: Simulation result of the SIMC-tuned PI controller in Simulation environment 3 with a change in dead time introduced at 70.2 seconds.

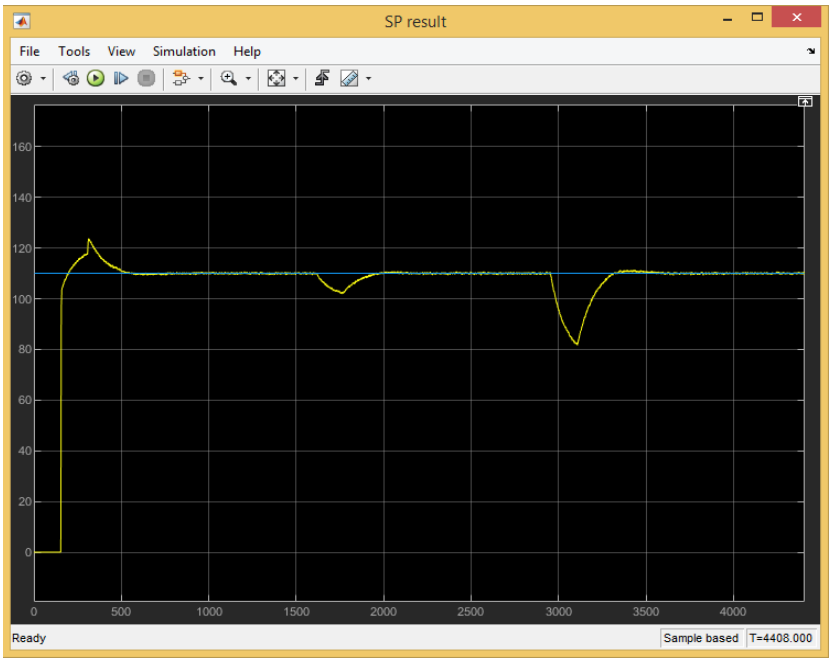


It should be noted that under such a long default dead time, a 2 second miscalculation had nearly no effect, showing that the effect of miscalculated dead time is proportional to measured dead time for reactive controllers. This is attributed to controller speed being tuned with dead time in mind as well as process parameters, as with all FOPDT-model based tuning methods the PID controller's P-,I- or D- elements were lowered for slower error correction depending on the dead time. This is to allow for the controller to course-correct before meeting the reference value or overshooting the value too much, resulting in a slower controller response to attain feedback before overcorrecting to avoid oscillation.

Smith Predictor

The default Smith Predictors generally meet the desired value at a faster rate than other controllers with a small overshoot in compromised conditions and none in nominal conditions and with the fastest responses to disturbance as seen in Figure 25 below:

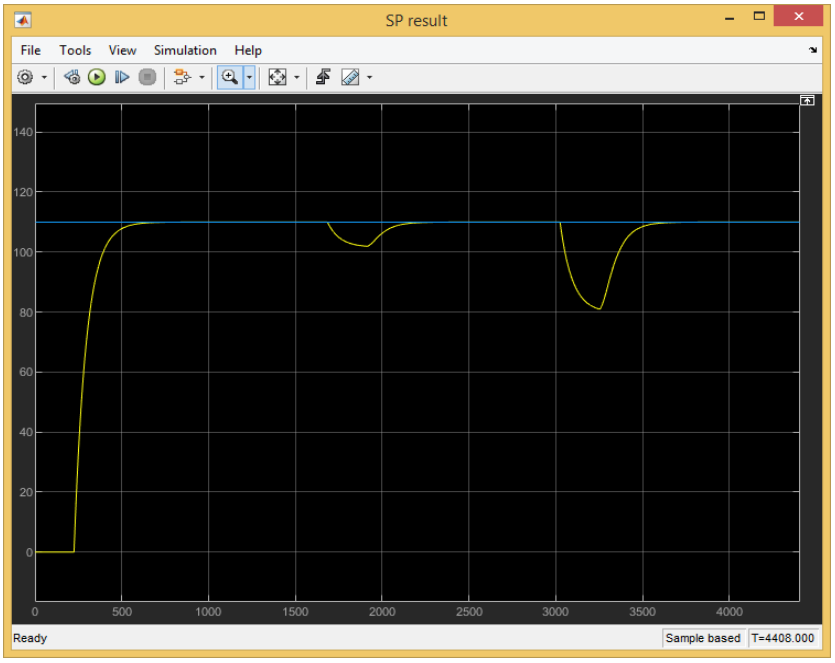
Figure 25: Simulation result of Smith Predictor PI controller tuned with Direct Synthesis tuning rules in Simulation environment 3 with a +7% miscalculation and noise.



Direct Synthesis results in the controller reaching the intended value in roughly 160 seconds and settling, adjusting to disturbance more quickly than other controllers with less overcorrection. Under non-nominal conditions there was a significant overshoot that was fully corrected at 12 minutes, as seen in Figure 25. This can be attributed to the long dead time creating a situation where model miscalculations result in an overshoot that takes longer to be corrected due to the faster controller amplifying the degree of overshoot until feedback of the actual process overshooting was received.

2DOF SP controllers trade the characteristic speed of the default SP for robustness and as a result were generally slower than the default SP configuration, in rise time, settle time and disturbance rejection as seen in Figure 26 below:

Figure 26: Simulation result of a 2DOF-SP controller in Simulation environment 3



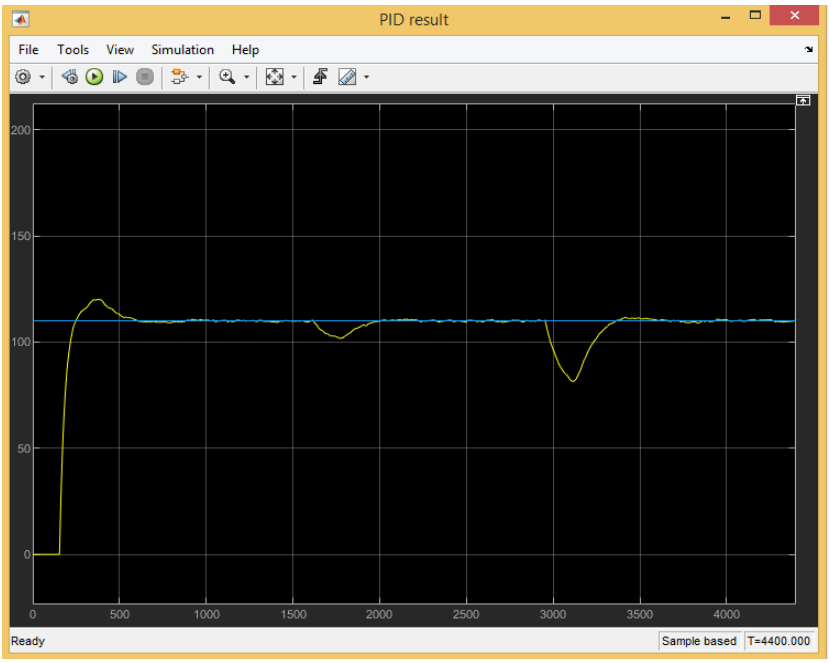
The large difference in performance between SP-DS and 2DOSP can be attributed to the 2DOF-SP being prepared for a larger dead time error than in the other Simulation environments tested. This results in both the filter and the controller being slowed due to a longer error in dead time resulting in a more stable, but significantly slower performance in default conditions.

Like with other simulation environments, the default Smith Predictor system cannot handle a large change in dead time and breaks when exposed to it. An appropriately prepared 2DOF-SP, in comparison, remains usable with a rise-time almost halved and it settling at around 1100 seconds in non-nominal conditions.

Model Predictive Control

The Model Predictive Controller configuration used in environment provides a worse response than a Smith Predictor using Direct Synthesis tuning under controlled conditions. However if there is a +7% miscalculation, the MPC manages to reach the set-point value at the same speed with a similar settling time, with a noticeably slower disturbance rejection. This can be seen in the simulation results displayed in Figure 27:

Figure 27: Simulation result of a MPC with a control horizon of 2 in Simulation environment 3 with a +7% miscalculation in the process model.



As seen in the Figure 27 above, a miscalculation of the controller value results in a 5% overshoot that takes until 10 minutes to correct with a roughly 10% overshoot. With a dead time error of 38.5 seconds, the controller oscillates uncontrollably as it did in similar tests done in other simulation environments.

6.4 Simulation environment 4

The parameters for this simulation environment were entirely artificial and self-created, this environment designed to test controller behavior at very low dead time compared to the time constant for a favorable performance for PID and 2DOF controllers. It has also been made to analyze how other controllers work in these environments for the sake of comparison with the default conditions of Simulation environment 1.

The transfer function in equation 6.4 below displays the process model, having a process gain of 30, a time constant of 35 seconds and a default dead time of 15 seconds:

$$Pn(s) = \frac{30}{35s + 1} \cdot e^{-15s} \tag{6.4}$$

This simulation was used to test the system's ability to keep a value of 45 over the course of 2000 seconds. Input disturbance was applied at the amounts of 0.1 and -0.25 process input at times 615 seconds and 1465 seconds respectively, with a dead time change of 3.5 seconds being deployed at 35 seconds. As with all the other simulation environments tested, the results of the tests made in this environment with both complete model accuracy and model inaccuracy of -10% are listed in table 15 and table 16 below, with the results being elaborated on separately in in their own later in this sub-chapter:

Table 15: Results for Simulation environment 4 without model inaccuracy. Rise Time and Settle time are without dead time being taken into account.

Test 4-1	Unchanging dead-time				Change in dead-time introduced at 35s			
Method/Time(s)	Rise Time	Settle time	Dist.Rej1	Dist.rej2	Rise Time	Settle time	Dist.Rej1	Dist.rej2
ZN PI	551.000	551.000	509.835	534.447	535.700	535.700	479.000	526.000
ZN PID	42.970	164.000	118.000	117.200	42.600	251.500	104.500	104.700
CC PI	15.192	430.500	51.700	51.700	11.691	563.963	52.950	56.120
CC PID	28.550	157.000	129.000	132.077	27.020	129.500	105.521	105.000
SIMC PI	41.100	155.935	265.000	285.000	40.500	152.500	256.239	283.066
Software tuned PI	35.070	228.900	275.000	300.000	39.500	480.456	256.239	108.559
2DOF PID Taguchi	115.200	939.071	41.004	126.000	123.550	-	-	-
2DOF PID Terauchi 0%	179.154	179.154	163.000	176.000	210.500	210.500	156.000	191.500
SP Software tuned PI	12.210	99.823	234.687	258.873	8.710	187.500	220.000	256.000
SP DS tuned PI	9.200	9.200	237.000	258.000	8.640	-	-	-
2DOF SP	319.000	319.000	251.000	271.000	313.500	313.500	244.500	265.500
MPC	5.440	19.815	244.140	264.788	-	-	-	-

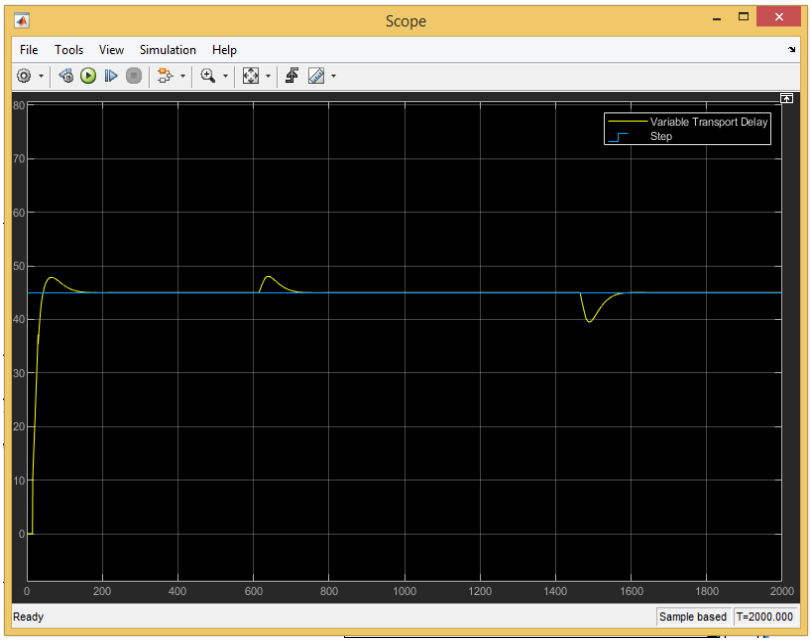
Table 16: Results for simulation environment 4 with a model inaccuracy of -10%. Rise Time and Settle time are without dead time being taken into account.

Test 4-D	Unchanging dead-time				Unstable dead-time			
Method/Time(s)	Rise Time	Settle time	Dist.Rej1	Dist.rej2	Rise Time	Settle time	Dist.Rej1	Dist.rej2
ZN PI	594.900	594.900	516.000	535.000	581.500	581.500	504.000	535.000
ZN PID	48.000	174.979	124.000	125.000	45.520	260.299	110.500	110.000
CC PI	15.400	392.098	55.209	55.000	11.900	902.168	53.500	53.709
CC PID	32.300	177.000	148.000	151.000	31.650	163.500	138.000	150.000
SIMC PI	48.400	275.188	284.000	313.152	45.900	261.500	277.116	300.626
Software tuned PI	38.770	284.000	291.000	313.990	38.600	160.500	282.500	302.500
2DOF PID Taguchi	16.455	1293.977	37.250	41.700	-	-	-	-
2DOF PID Terauchi 0%	200.000	200.000	177.000	188.000	170.500	170.500	160.500	198.600
SP Software tuned PI	12.670	213.400	243.000	263.800	9.170	159.400	241.500	259.500
SP DS tuned PI	225.084	225.084	259.739	271.399	-	-	-	-
2DOF SP	334.000	334.000	255.500	276.500	334.500	334.500	263.000	285.000
MPC	5.750	226.250	253.873	275.587	-	-	-	-

PID

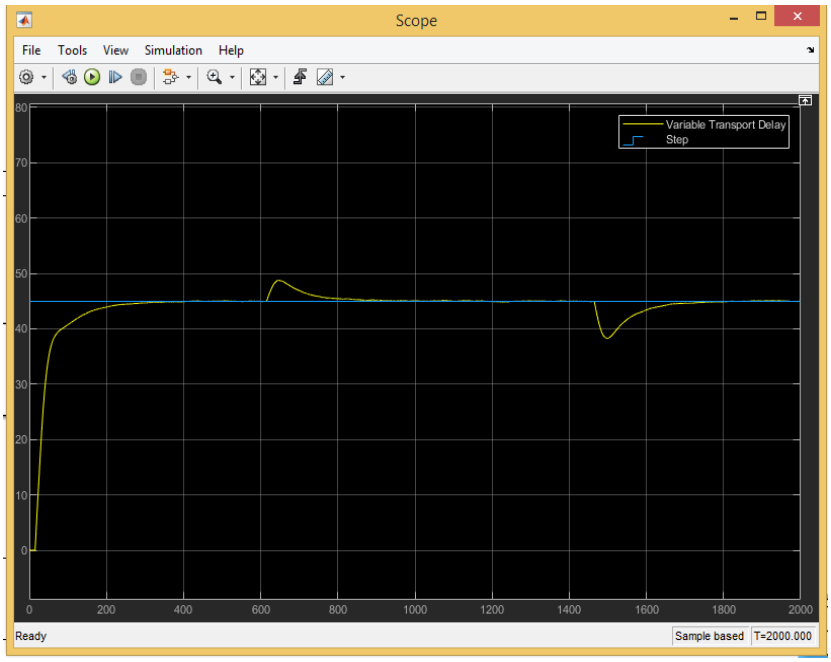
ZN- and CC-tuned PID-controllers display similar behavior and results, with CC-tuning providing a generally better response at reaching the referential value:

Figure 28: Simulation result of PID-controller tuned with Ziegler Nichol tuning rules in Simulation environment 4



The Cohen-Coon tuned PID-controller provides a slightly faster rise time of 32 seconds with a smaller overshoot, but with a slightly slower settling time of 172 seconds and a slightly slower time disturbance rejection with a 10 to 15 second difference.

Figure 29: Simulation result of Ziegler Nichols-tuned PI controller in Simulation environment 4 with no model miscalculation.



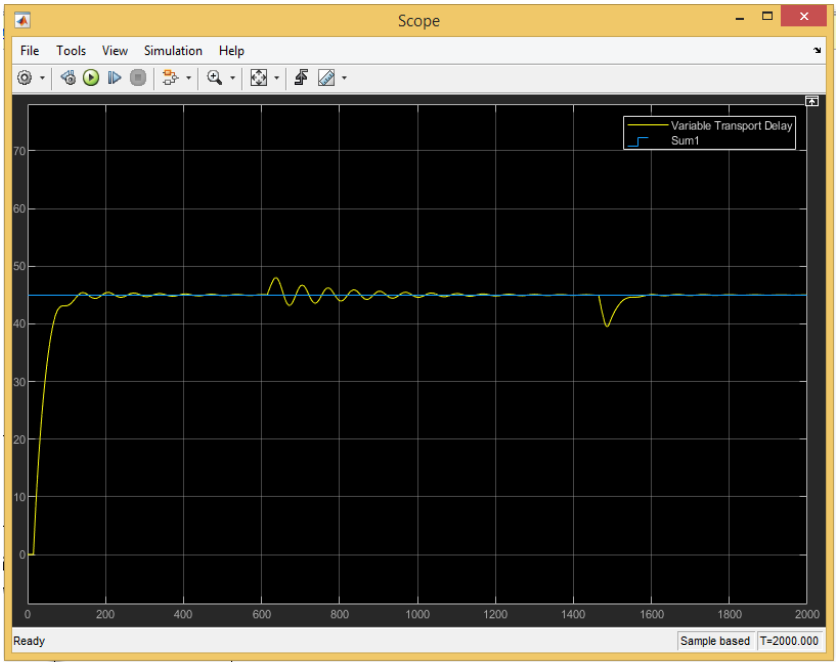
As seen in Figure 29 above, using the Ziegler-Nichols rules for tuning PI controllers the controller manages to reach the reference value at 9 minutes with minimal overshoot at either circumstance, with similar disturbance adjustment time. It should be noted that it manages to reach a 5% difference from the reference value at 5 minutes.

The CC-PI controller in this environment had more and larger relative overshoots and undershoots than in other simulations, it managing to reach 80% overshoot the reference value, which can be attributed the faster response provided by a lower dead time relative to the time constant and a lack of an output limit. It also haf the fastest disturbance rejection of 51 seconds, rise time was 15 seconds and the fastest disturbance rejection of less than a minute of all the default-PI tuning methods at the cost of the second slowest settle time, managing to correct disturbance with some slight overshoot that stabilizes in 100 seconds.

The SIMC-PI controller manages to reach the reference value at a minute and noticeably faster settle-time with a smaller overshoot than the PID-controllers, however disturbance adjustment was nearly 5 minutes. Using Matlab software to tune the PI controller provides results similar to the SIMC-PI controller, but with a larger overshoot, a slightly faster rise time, a significantly longer settling-time and a slightly longer disturbance rejection period.

The Taguchi and Terauchi-tuned 2DOF-PID-controller had a different result, as each controller have a very different way of reaching the set-point value and rejecting disturbance:

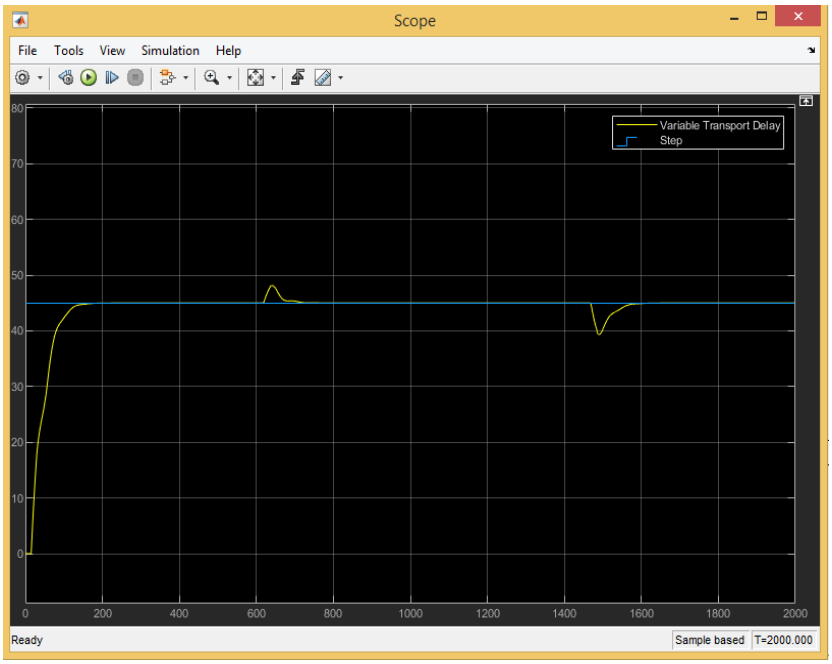
Figure 30: Simulation result of 2DOF-PID-controller tuned with Taguchi-Arai tuning rules in Simulation environment 4



As shown in Figure 30, the Taguchi 2DOFPID tuning method had a longer settling time but in exchange for a faster rise time with the fastest disturbance rejection of all methods due to its oscillatory nature with little noticeable overshoot. However its disturbance rejection had a long period of gradually decreasing oscillation comparable to the former disturbance it was rejecting in magnitude in the case of smaller disturbances. This behavior can be attributed

In contrast 0% overshoot Terauchi configuration provides a performance similar to other PID-controllers, but slower with no overshoots and a slower disturbance rejection than either of the two default PID-controller tunings, having a twice as long rise time and 25% longer disturbance adjustment times with a faster response. This is shown in Figure 31 below:

Figure 31: Simulation result of 2DOFPID-controller tuned with Terauchi tuning rules with 0% overshoot in Simulation environment 4

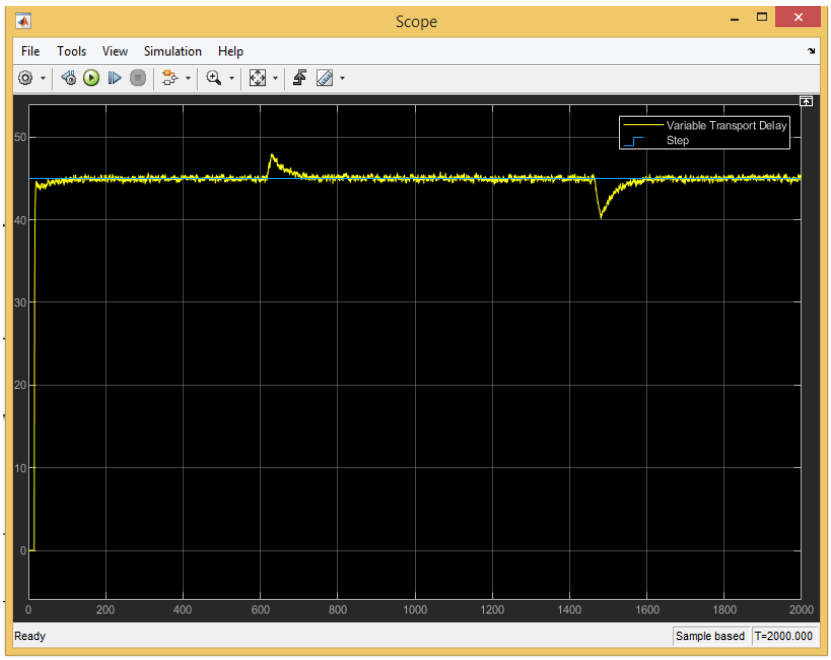


A 3.5 second difference results similar results in other simulations, only that the Taguchi-tuned 2DOFPID control oscillates consistently until interrupted by disturbance , while the 2DOF Terauchi-tuned 2DOFPID had similar performance to the PID-controllers tested, if with no overshoot when correcting due to the slower reaction. The Taguchi controller's increased oscillation can be attributed to the error in dead time increasing the controller's speed coupled with the existing oscillatory response resulting in oscillation that takes a significantly longer amount of time to stabilize, if at all.

Smith Predictor

Smith predictors generally meet their intended value at nearly half a minute and while the DS-SP meets the value with a rise time of 24.2 seconds in nominal conditions with no overshoot. Under non-ideal conditions the Smith predictor does not reach the intended value until 4 minutes, as shown in Figure 32:

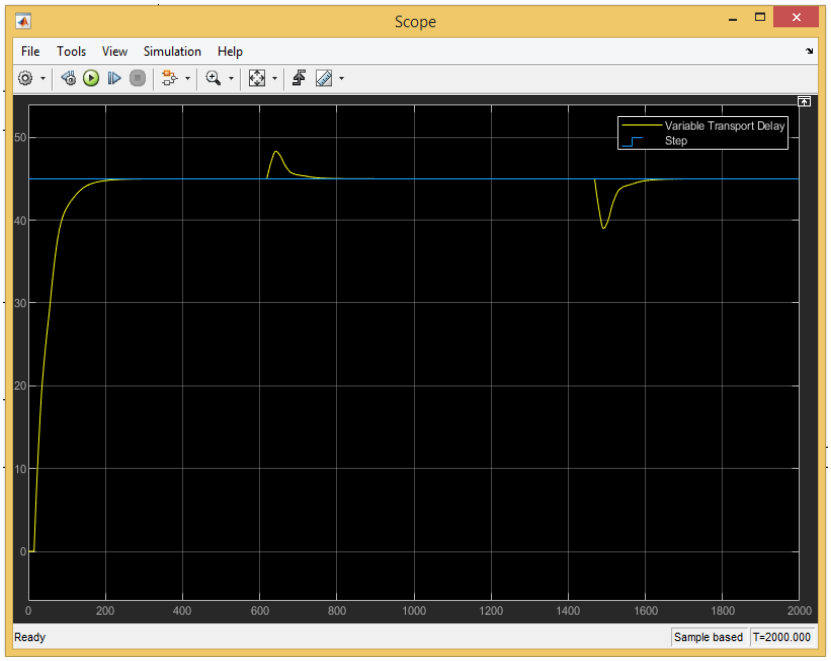
Figure 32: Simulation result in SP-PI tuned with Direct Synthesis rules in Simulation environment 4 with a -10% miscalculation



Disturbance rejection was similar to the SIMC and the Matlab software-tuned PI, but with a slightly faster adjustment that was half a minute less. It should also be noted that with process model inaccuracies there was a significant undershoot and overshoot, with the Direct Synthesis-tuned PI controller having significantly longer settling-time and rise time. This slowed performance can be attributed to the overall controller structure of the Smith Predictor settling according to the process model, before receiving a response and a low error value in addition to the -10% process model wrong estimation slowing the controller response significantly.

The 2DOF-SP was significantly slower than the default SP, with it reaching the intended value in four minutes and more with model inaccuracies, as shown in Figure 33 below:

Figure 33: Simulation result of a 2DOF-SP PI controller in Simulation environment 4 with a -10% miscalculation in process gain and time constant.

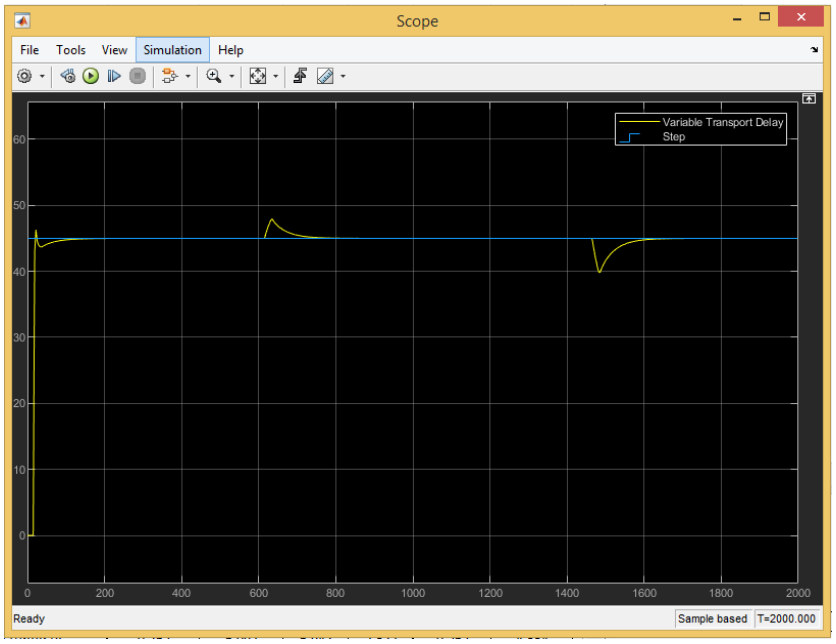


Notably, a change in dead time results in the “PID tuner” software-tuned controller having larger overshoots and undershoots rather than oscillating, which can be attributed to the nature of the controller tuning done for this simulation environment favoring a slower reaction. The Two-degree-of-freedom SPs controller configuration along with the filter was too slow for the 3.5 second change in dead time to have a significant effect as it had been prepared for it, as it had been in other Simulation Environments.

Model Predictive Control

Model Predictive Control had a faster performance than the Smith Predictors and in nominal conditions it settles at 20 seconds with a disturbance rejection of 245 to 276 seconds depending on the degree of disturbance under ideal conditions with a small overshoot. This changes when it was exposed to a -10% miscalculation in process values:

Figure 34: Simulation result of MPC in Simulation environment 4 with a control horizon of 3.



As seen in Figure 34, when exposed to less than ideal conditions with a -10% miscalculation, settling had been dramatically increased after initially reaching the reference value, this can be attributed to predicted controller values being so that there was a brief overshoot before quickly corrected back to the reference value. However due to the model miscalculation, this predicted process and the controller's attempt to fix it and disturbance significantly lengthened by the decrease in process gain.

Like in the tests done in Simulation Environment 1, 2 and 3, a 3.5 second change in dead time results in growing in oscillation, as this controller was unable to handle dead time errors of such magnitude within this Simulation environment.

6.5 Cross-examination

For the purpose of cross examination Simulations environments 2 and 3 have been tested and simulated again in nominal conditions with matching dead time-to-time-constant ratios to see if their general behavior remains consistent regardless of actual dead time, time constant and process gain.

Figure 35: Simulation results of SIMC-tuned PI controllers within Simulation environment 2 with a dead time of 5.3 (left) and Simulation environment 3 with a dead time of 225(right)

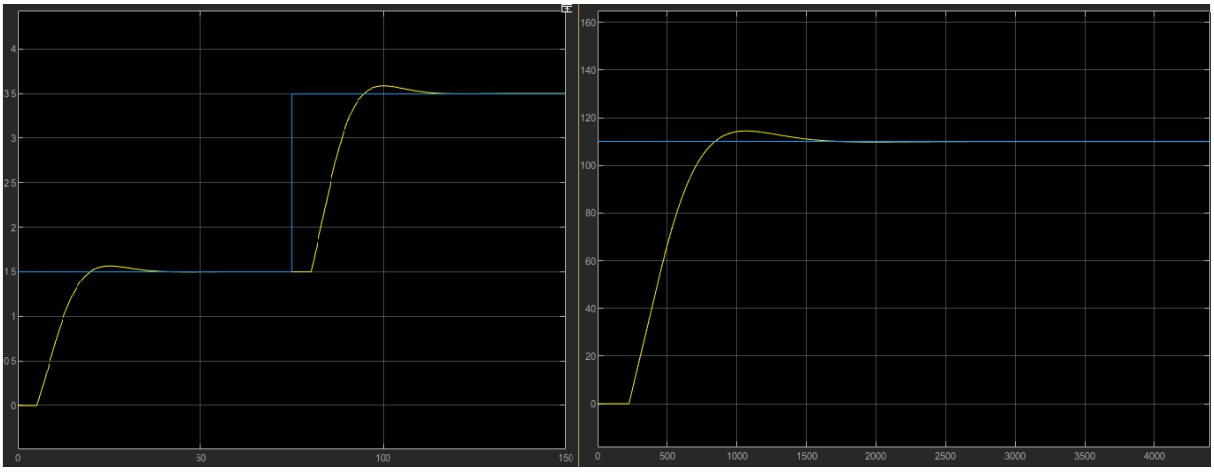
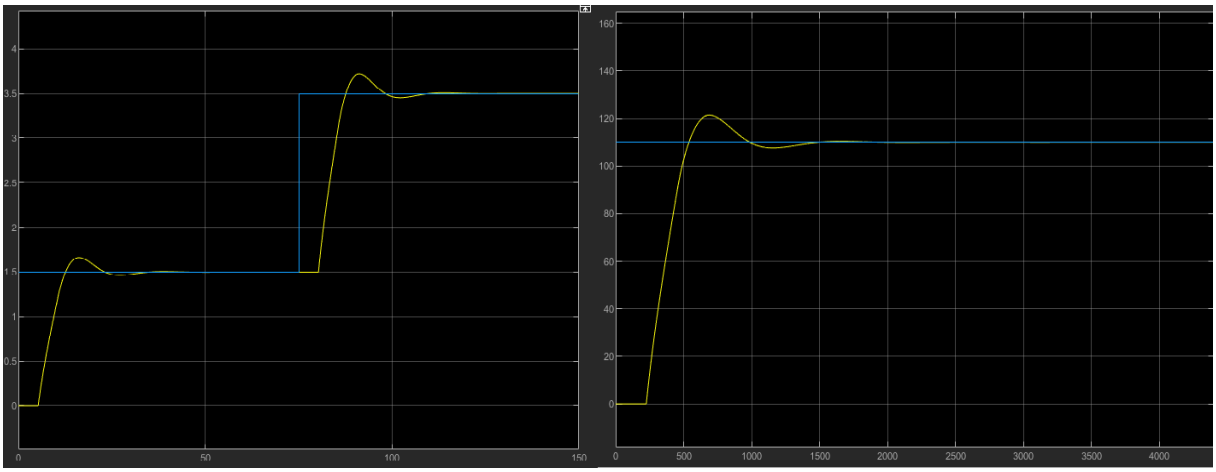


Figure 36: Simulation results of CC-tuned PI controllers within Simulation environment 2 with a dead time of 5.3 (left) and Simulation environment 3 with a dead time of 225 (right).



As shown in Figures 35 and 36 above and after multiple other measurements, general behavior such as the initial overshoots when tuned with CC-tuning rules remain similar, as does their extent from the reference value.

This can be also seen in the below table 17 and table 18, where dividing each controller's simulation results by each simulation environment's time constant, as the rise-time and settle-time had similar values for PI controllers:

Table 17: Test 2-2 results divided by the time constant. Change in dead time is 1 second.

	Test 2-2							
Results divided by time constant	Unchanging dead-time				Change in dead time introduced at 1.7s			
Method/Time(s)	Rise Time	Settle tim	Dist.Rej1	Dist.rej2	Rise Time	Settle tim	Dist.Rej1	Dist.rej2
ZN PI	-	-	-	-	-	-	-	-
ZN PID	-	-	-	-	-	-	-	-
CC PI	4.300	24.005	8.706	9.551	4.035	35.988	9.029	10.706
CC PID	-	-	-	-	-	-	-	-
SIMC PI	8.535	21.444	25.215	13.042	7.271	41.294	12.055	10.353
Matlab software tuned PI	5.700	26.294	10.176	10.249	5.053	32.264	10.012	10.249
2DOFPID Taguchi	-	-	-	-	-	-	-	-
2DOFPID Terauchi	-	-	-	-	-	-	-	-
SP Matlab Software-tuned PI	0.686	5.681	9.914	10.089	-	-	-	-
SP DS tuning	7.953	7.953	11.084	13.384	3.400	37.868	8.233	8.547
2DOF SP	8.408	8.408	11.828	13.136	3.409	21.765	7.941	8.038
MPC2	3.409	3.409	9.516	10.770	-	-	-	-

Table 18: Test 3-2 results divided by time constant. Change in dead time is 18.25 seconds.

	Test 3-2							
Results divided by time constant	Unchanging dead-time				Change in dead-time introduced at 70.2s			
Method/Time(s)	Rise Time	Settle time	Dist.Rej1	Dist.rej2	Rise Time	Settle tim	Dist.Rej1	Dist.rej2
ZN PI	-	-	-	-	-	-	-	-
ZN PID	-	-	-	-	-	-	-	-
CC PI	4.466	24.226	9.003	8.954	4.295	26.367	9.043	9.018
CC PID	-	-	-	-	-	-	-	-
SIMC PI	8.789	31.140	13.377	13.377	8.046	32.530	12.626	12.719
Matlab software tuned PI	4.402	21.671	9.018	9.018	4.281	31.027	8.945	8.873
2DOFPID Taguchi	-	-	-	-	-	-	-	-
2DOFPID Terauchi	-	-	-	-	-	-	-	-
SP Matlab Software-tuned PI	5.513	20.185	10.184	10.028	5.164	28.654	9.842	9.850
SP DS tuning	0.142	0.142	10.264	10.113	-	-	-	-
2DOF SP	7.764	7.764	9.188	10.328	3.397	17.899	7.550	7.158
MPC	0.460	1.075	8.419	9.658	-	-	-	-

The difference in resulting Smith Predictor and MPC behavior can be attributed to performance being generally based on the relative values between the time constant and the process gain, which is very large for Simulation environment 3 in comparison to Simulation environment 2. Another explanation for this variance is that generally longer processes enable better results for predictive controllers in relation to standard PID-controllers.

For further analysis, a second testing of Simulation environment 1 with a similar dead time-to-time constant to Simulation Environment 3is done and was then compared to the default Simulation Environment 3 in table 19 and table 20 below:

Table 19: Test 1-2 results divided by the time constant. Change in dead time is 5 seconds

	Test 1-2							
Results divided by time constant	Unchanging dead-time				Change in dead-time introduced at 13.1s			
Method/Time(s)	Rise Time	Settle tim	Dist.Rej1	Dist.rej2	Rise Time	Settle tim	Dist.Rej1	Dist.rej2
ZN PI	-	-	-	-	-	-	-	-
ZN PID	-	-	-	-	-	-	-	-
CC PI	2.450	19.162	5.870	5.996	2.399	20.461	6.742	6.315
CC PID	-	-	-	-	-	-	-	-
SIMC PI	5.542	13.489	9.154	9.061	4.748	12.968	8.002	7.871
Matlab software tuned PI	5.008	19.512	8.305	8.404	4.412	-	7.443	8.010
2DOFPID Taguchi	-	-	-	-	-	-	-	-
2DOFPID Terauchi	-	-	-	-	-	-	-	-
SP Matlab Software-tuned PI	1.431	5.769	5.063	8.939	1.453	19.705	5.059	5.359
SP DS tuning	4.741	4.741	8.526	9.018	-	-	-	-
2DOF SP	9.427	9.427	9.396	10.275	2.527	15.763	5.534	5.855
MPC	0.371	1.291	8.482	9.337	-	-	-	-

Table 20: Test 3-1 results divided by the time constant, change in dead time is 38.5 seconds.

	Test 3-1							
Results divided by time constant	Unchanging dead-time				Change in dead-time introduced at 70.2s			
Method/Time(s)	Rise Time	Settle time	Dist.Rej1	Dist.rej2	Rise Time	Settle time	Dist.Rej1	Dist.rej2
ZN PI	-	-	-	-	-	-	-	-
ZN PID	-	-	-	-	-	-	-	-
CC PI	2.714	21.329	6.168	6.244	2.632	36.112	6.313	7.146
CC PID	-	-	-	-	-	-	-	-
SIMC PI	5.998	20.812	9.630	9.620	4.993	22.372	8.819	8.852
Matlab Software tuned PI	3.979	14.866	7.816	7.597	3.544	29.198	8.047	7.413
2DOF PID	38.583	38.583	5.901	6.496	30.016	30.016	6.657	6.382
SP Matlab Software tuned PI	0.883	5.278	8.000	8.970	0.890	-	-	-
SP DS tuned PI	0.146	0.146	6.669	7.849	-	-	-	-
2DOF SP	8.732	8.732	10.385	10.385	2.667	17.913	6.375	6.375
MPC	0.212	0.644	7.369	8.650	-	-	-	-

As seen in Table 20 and Table 19, PI controller results remain similar under nominal conditions and the gap in performance between their Direct Synthesis-tuned Smith Predictor controllers and MPC controllers was not as large, as the difference between the time constant and the process gain was less than in the second simulation environment.

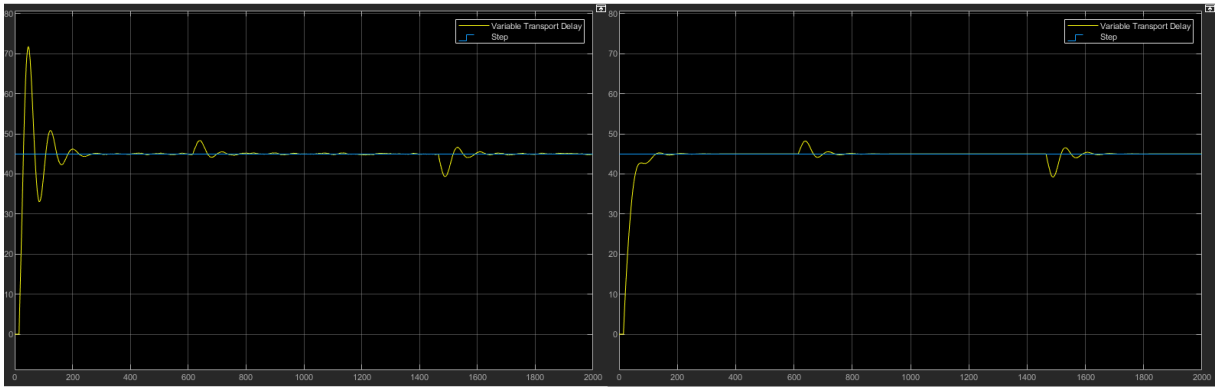
6.6 Minor tests

Various smaller tests have been done to further analyze controller behavior and how implementing various minor modifications can result in compensating for issues such as overcompensation or oscillation. Due to the focus on general behavior rather than specifics, no test results were written down.

Limiters on controllers

Output limiters can be implemented to limit the controller output spike that accompanies the response of faster controllers, as seen with controller performance in Simulation Environment 1 in comparison to other simulation environments. As the most extreme example of controller overshoots, a Cohen-Coon-tuned PI controller was used as an example on what stricter output limiters can achieve in Figure 37 below:

Figure 37: Simulation result of Cohen Coon-tuned PI controller in Simulation environment 4 without a controller output limit (left) and with an output limit of 2 (right).

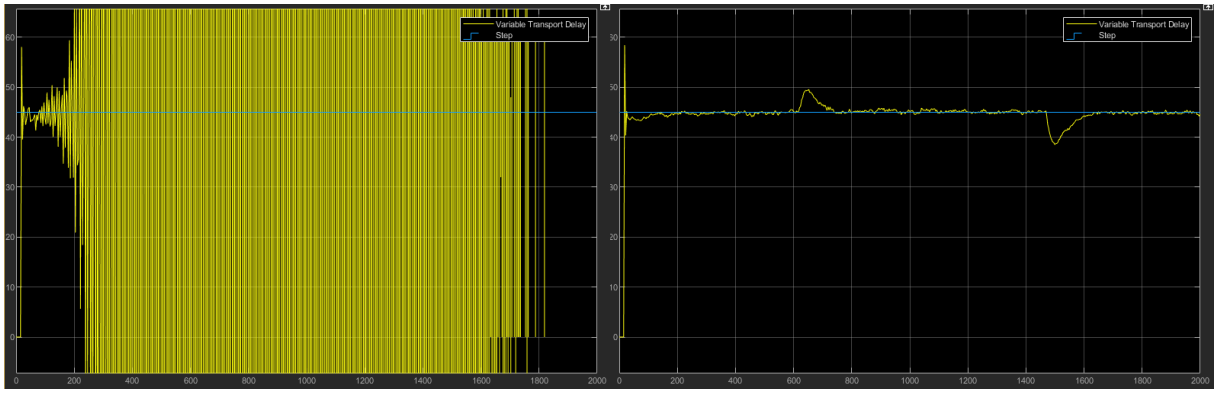


Limiting controller output can put a higher limit for the inserted value for controlling the process. This can be implemented on controller tunings that first deploy a large spike to meet the initial value faster before quickly correcting it, mitigating the negative qualities that involve massive spikes and long settling periods. However if the control system is not isolated and is expected to deal with disturbance that decreases output, a low max output limit for the controller can lead to situations where the reference value is unreachable.

Custom White Noise Model for MPC

Besides mitigating the effects of noise when implemented in Model Predictive Control, custom models for white noise can be used to deny and control oscillations caused by the misalignment of estimated and actual value caused by miscalculated dead time. The Matlab Software “MPC designer” allows the use of a custom white noise model to reduce the effects of noise on a system. Custom noise models can allow for otherwise unusable MPC configurations to be used in systems where an MPC would otherwise oscillate. The effectiveness of this model is shown in Figure 38 below, where an MPC for Simulation environment 4 using a control horizon of 3 was tested under a 3.5 second change in dead time at the time constant with and without a custom white noise model:

Figure 38: Simulation results for MPC within Simulation Environment 4 using a control horizon of 3 without (left) and with a Custom White Noise model with the magnitude of 20 (right).

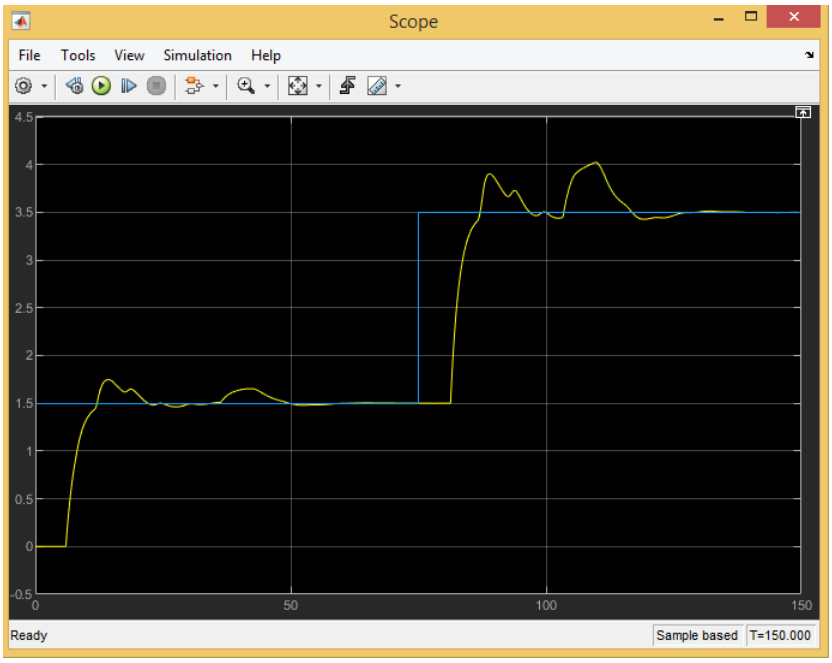


The custom white noise model does eliminate growing oscillations that tend to be present where there is a large dead time miscalculation, but response-time to errors was lengthened as a result. This can be explained as the Custom White Noise model recognizing sudden mismeasurements, such as the oscillations caused by a dead time mismeasurement, as disturbances or other errors being acknowledged as actual errors due to being a sustained change rather than a sudden one.

Unknown dead time on Predictive controllers

Unknown dead time on Predictive control models can be disastrous, as the speed at which the assigned controllers operate on can make even 2 second inaccuracies result in uncontrollable oscillation. To test how smaller dead time errors can affect the predictive controllers tested, the SP Direct Synthesis-tuned PI-controller and the MPC were tested within a modified simulation environment 2 with a different dead time miscalculation:

Figure 39: Simulation result of DS-tuned PI controller with a Smith Predictor in Simulation environment 2 with a dead time of 5.3 with a 1 second of dead time miscalculation.

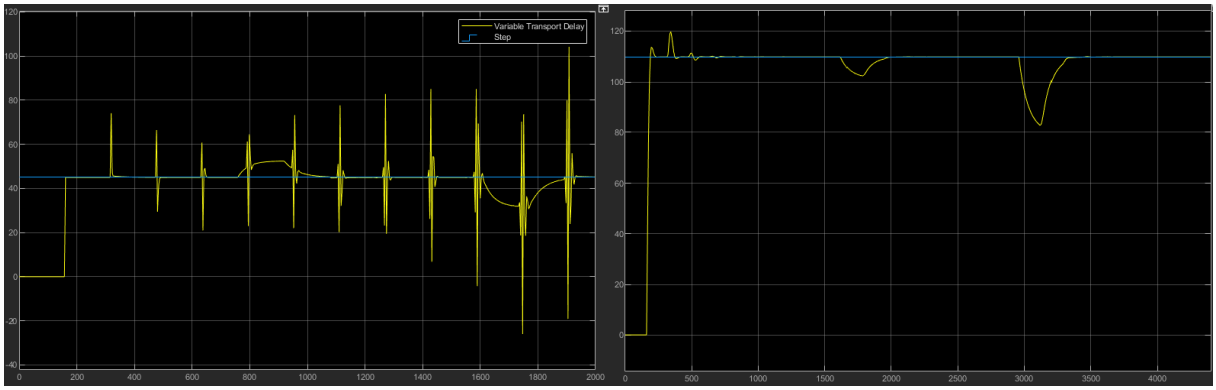


In the Figure 39 above, it can be seen that not all unmeasured dead time result in catastrophic failure, as a one second dead time mismeasurement does not seem to produce oscillations as large or growing as a 2 or 3 second dead time change. After further testing, the degree of error where the system simply oscillates before settling and the system growing in oscillation is 1.6 seconds. The effect was the same for both dead time estimation overshoot and undershoot.

The degree and speed of which unmeasured dead time effects Model Predictive controllers seems to be dependent on the error in the dead time in comparison to the time constant rather than the amount of dead time itself.

This can be seen in Figure 40 below, where individual simulation results for an MPC done with a dead time error of 4 seconds in simulation environment 4 with measured dead time of 154 and in Simulation environment 3 with the same dead time measurement:

Figure 40: Simulation Results of MPC with a control horizon of 3 within Simulation environments 3 (left) and 4 (right) with the dead time of 154 seconds and dead time error of 4 seconds.



As seen in Figure 40 above, a 4 second difference in dead time had been relatively insignificant in a process with the time constant of 70 seconds, while in a similar process with a time constant of 35 seconds it results in repeated and increasing oscillation periods.

This can be attributed to a large and continuous mismatch between the process model, the MPCs prediction model and the incorrect flow of information resulting in gradual and growing overcorrection. Due to the overcorrected and incorrect output being taken into account with other output values, the record of results is continuously corrupted creating further increasing oscillations due to the constantly incorrectly measured results brought by repeatedly mistimed comparisons. Another explanation for this behavior can simply be the aggressive correction the MPC has in comparison to other methods like with the Direct Synthesis-tuned Smith Predictor controller, however this does not explain overcorrection growing between “pulses” in the left graph of Figure 40.

7 RESULTS

The effects of a slower flow of information caused by dead time on a system are dependent on multiple factors, such as if the dead time is measured, unknown or its length compared to the time constant. For the sake of simplicity, the effects of dead time is split between whether it is measured or unknown, before a general recommendation of where and how to use each control method based on circumstances and priorities.

7.1 Measured dead time

Measured dead time will generally slow the response and general capabilities of PID-controllers, with the extent of which being dependent on the speed of the tuning method itself. Tuning methods that take dead time into account will deliberately slow their response to the dead time to match the slowed flow of feedback information to prevent uncontrollable oscillations. This is understood by the Integral element being deliberately smaller by a larger “Ti” value to facilitate a smaller gradual response and dead time being often an element that decreases the value of the P-, I- and D- elements in the different tuning methods tested in chapter 6. This is to provide a larger margin of error on all areas such as dead time, process gain estimation and time constant estimation as well as allowing for course correction.

Under ideal circumstances measured dead time is irrelevant to Model Predictive Controllers and dead time-compensators used on feedback controllers. However in a practical implementations there will always be a degree of miscalculation, as such longer dead times increase the degree of resulting overshoots, undershoots and time for correction in the case of overestimated process dynamics.

When using either reactive or Predictive controllers, measured dead time enlarges disturbances according to the time constant, leaving them to grow larger as they take longer to be corrected by the controller. This is compounded by the deliberately slowed response in non-prediction-based controllers, leading to a slowed disturbance rejection response overall.

7.2 Unknown dead time

Unknown dead time has a destabilizing effect on most tuning methods and variants of the Proportional Integral Derivative Controller, resulting in overshoots and undershoots dependent on how large it is in comparison to the measured dead time. Longer dead time than estimated result in a slower flow of information towards the feedback of the controller leading to larger and more over-adjustments, having a longer settling period for dealing with uncertainties as a result. Lower than predicted dead time brings only minor issues by comparison, with the tuned reaction being delayed enough to account for longer dead times. This results in merely a slower response than necessary rather than overcorrection or oscillation.

Model Predictive Controllers and the Default Smith Predictor with a Direct Synthesis tuning configuration are generally vulnerable to changes in dead time and result in unsettling or growing oscillations. In the case of the default Smith Predictor with a PI controller tuned with the Direct Synthesis method, it can be attributed to the prediction process correcting values as fast as possible using predicted output and comparison to actual output. A wrongly estimated rate of information, where the delayed feedback is not eliminated by being compared to the actual delayed output, bringing unwanted spikes or dips and resulting in a repeating critical failure for the prediction process until it is corrected. This failure in the prediction process creates a cycle of overcorrection and, in the case of the MPC, continuous corruption of earlier data where mis-timed information often results in growing oscillation. By contrast, the Two-Degree-Of-Freedom Smith Predictor has responses similar to unknown dead time like non-predictive controllers, but at the price of the speed and accuracy that characterizes the performance of predictive controllers.

Unlike non-predictive controllers, an overestimated dead time for MPCs and SPs is just as harmful as underestimating it due to relying in comparing the estimated feedback and the actual feedback. When using predictive controllers, accuracy is vital, with correct dead time estimation being one of the most important aspects to account for.

7.3 Recommended controllers and tunings

The default PID-controller configuration with the two most common tuning rules will only provide satisfactory results in lag-dominant situations with a very low and quickly dealt with overshoot. However it should be avoided in higher noise magnitude and frequency environments due to the derivative element being disrupted, significantly increasing the length of rise time and disturbance rejection.

Proportional-Integral controllers are more flexible and function in much larger relative Dead time situations and noise frequencies, with a wider variety of tuning rules commonly used. The Ziegler Nichols method has a performance that is overall worse than all alternatives with arguably the slowest response to error. The Cohen-Coon method has massive oscillations in exchange for its speed in reaching the set-point value and disturbance rejection, limiting controller output is recommended to prevent the massive overshoot when reaching the reference value. The Skogestad-Suggested Lambda tuning method can provide a response comparable to the Cohen Coon method with smaller overshoots and it can fit in many situations. Both PID- and PI-controllers show high tolerance in model miscalculation that would overshoot for longer periods in predictive models with minimal issues. For a larger margin of error and stable control, dead time can be deliberately increased in calculations at the cost lower speed.

Smith Predictors and Model Predictive Controllers are recommended to provide quick error correcting values for significantly delayed process situations, where the dead time is several times larger than the time constant of the process, and for processes with a long time constant. However under any circumstance they are not to be used in unstable systems without heavy modification, as changes in dead time can lead to uncontrollable oscillations. Model Predictive Controllers have a generally better performance in controlled environments and are more resilient to dead time error than the default Smith Predictor configuration. A 2DOF Smith Predictor or an MPC with a custom white noise model can be used, but the cost of speed in some circumstances could be considered an unacceptable compromise despite higher stability. If the process is unstable or the Dead time estimate is unreliable, the use of predictive controllers is not recommended, as such conditions result in oscillation or otherwise significantly compromised performance.

7.4 Conclusion

In most controller tunings for PID controllers measured dead time hinders the controller's ability to correct errors, while predictive control methods such as the Smith Predictor and the Model Predictive Control maintain their speed regardless of the amounts of dead time with an almost instant settling time. Unknown dead time has a destabilizing effect on all control methods resulting in larger overcorrections accompanied with longer settling periods and, in the case of control methods based on prediction, can result in uncontrollable oscillation.

What controller to use in what situation in relation to dead time depends on the desired output and the environment around the controller, however there are a few general controllers that provide a serviceable performance. For controlling a more unstable process is to use a Cohen Coon-tuned PI controller with stricter output limits to avoid output spikes that can be harmful to the process and lessen oscillations while being quick to error to disturbances. In more stable environments where process dynamics are accurately measured, predictive controllers are able to provide a fast rise to the set point value and a shorter disturbance rejection period with minimal overcorrection in ideal conditions.

References

Altmann, W. (2005). *Practical Process Control for Engineers and Technicians.* (S. McKay, Ed.) Elsevier Science & Technology. Retrieved from ProQuest Ebook Central, https://ebookcentral-proquest-com.ezproxy.hamk.fi/lib/hamk-ebooks/reader.action?docID=234968#

Geyer, T. (2016). *Model Predictive Control of High Power Converters and Industrial Drives.* John Wiley & Sons, Incorporated. Retrieved from ProQuest Ebook Central, https://ebookcentral-proquest-com.ezproxy.hamk.fi/lib/hamk-ebooks/reader.action?docID=4688951

Haber. (2011). Predictive Control in Process Engineering. In R. Haber, & R. U. Bars. John Wiley & Sons, Incorporated. Retrieved from ProQuest Ebook Central, https://ebookcentral-proquest-com.ezproxy.hamk.fi/lib/hamk-ebooks/detail.action?docID=1033310.

Karimi, H. R. (2019). *Stability, Control and Application of Time-Delay Systems.* (Q. Gao, Ed.) Retrieved from ProQuest Ebook Central, https://ebookcentral-proquest-com.ezproxy.hamk.fi

Mathworks. (2012). *Anti-Windup Control Using a PID Controller*. Retrieved from Mathworks: https://www.mathworks.com/help/simulink/slref/anti-windup-control-using-a-pid-controller.html?searchHighlight=anti%20windup&s_tid=srchtitle

Mathworks. (2021). *Design Controller Using MPC Designer*. Retrieved from Mathworks: https://www.mathworks.com/help/mpc/gs/introduction.html

Mathworks. (2021). *MPC modeling*. Retrieved from Mathworks: https://www.mathworks.com/help/mpc/gs/mpc-modeling.html

Mathworks. (2021). *Smith Predictor Controller*. Retrieved from Mathworks: https://www.mathworks.com/help/physmod/sps/ref/smithpredictorcontroller.html

McGraw-Hill. (2003). *McGraw-Hill Dictionary of Scientific & Technical Terms, 6th Edition.*

McMillan, G., & Vegas, H. (2010). *101 Tips for a Successful Automation Engineer.* International Society of Automation.

Moler, C. (2004). *The Origins of MATLAB*. Retrieved from Mathworks: https://www.mathworks.com/company/newsletters/articles/the-origins-of-matlab.html

Normey-Rico, J. E. (2019, January). *PID control of dead-time processes: robustness, dead-time compensation*. Retrieved from intranet.ceautomatica: intranet.ceautomatica, http://intranet.ceautomatica.es/sites/default/files/upload/13/files/Normey_PID_Control_of_Dead_Time_Processes.pdf

Normey-Rico, J. E., & Camacho, E. F. (2007). Control of Dead-Time Processes. Springer Science+Business Media.

O'Dwyer, A. (2006). *Handbook Of Pi And Pid Controller Tuning Rules(2nd edition).* Imperial College Press.

Samad, T. (2017, February). A Survey on Industry Impact and Challenges Thereof. Retrieved from IEEEXplore,https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7823045

Silva, L. R., Flesch, R. C., & Rico, .. E. (2018). Analysis of Anti-windup Techniques in PID Control of Processes with Measurement Noise. In A. o.-w. Noise. Elsevier ltd. Retrieved from ScienceDirect, https://www.sciencedirect.com/science/article/pii/S2405896318303975

Sira-Ramírez, H. (2014). Algebraic Identification and Estimation Methods in Feedback Control Systems. John Wiley & Sons, Incorporated.

Smith, C. A., & Corripio, A. B. (1985). Principles and Practice of Automatic Process Control 2nd edition.

Smuts, J. (2010, 11 22). *Optic Controls*. Retrieved from Control Notes: https://blog.opticontrols.com/archives/260

Sundaravadivu, K., Sivakumars, S., & Hariprasad, N. (2015, December). 2DOF PID controller design for a class of FOPTD models - An analysis with heuristic algorithms. St. Joseph's College of Engineering. Retrieved from Sciencedirect, https://www.sciencedirect.com/science/article/pii/S187705091500664X#!

Sung, S. w., Jietae, L., & In-Beum, L. (2009). *Process Identification and PID Control.* John Wiley & Sons, Incorporated, 2009. Retrieved from ProQuest Ebook Central, https://ebookcentral-proquest-com.ezproxy.hamk.fi/lib/hamk-ebooks/detail.action?docID=479833.

Xue, D., & Chen, Y. (2013). *System Simulation Techniques with MATLAB and Simulink.* John Wiley & Sons, Incorporated. Retrieved from ProQuest Ebook Central, https://ebookcentral-proquest-com.ezproxy.hamk.fi

Appendix 1: Controller tuning information

Controller tuning information Simulation 1

		Test 1-1		
	Std. PID tuning	Kc	Ti	Td
Ziegler Nichols	PI	0.03940508	26.64000	N/A
	PID	0.05254011	16.00000	4.00000
Cohen Coon	PI	0.08280481	11.95152	N/A
	PID	0.05977540	16.21081	2.20094
Skogestad-suggested Lambda tuning	PI	0.04378342	13.10000	N/A
2DOF PID tuning	Taguchi Araki	0.11503	12.12275	0.14263
	Hiroi Terauchi	0.085815508	19.0	3.36
Software tuned PI	PI	0.057279296	15.45527	N/A
Direct Synthesis for SP(td=0)	PI	0.70053476	13.10000	N/A
Software tuning for SP	PI	0.153674991	3.431654	N/A
2DOF Sp controller	PI	0.206039635	13.1	N/A
		Sample time	Prd.H	Ctrl. H
	MPC values	1.31	30	3
		Test 1-2		
	Std. PID tuning	Kc	Ti	Td
Ziegler Nichols	PI	0.01189587	88.24500	N/A
	PID	0.01586116	53.00000	13.25000
Cohen Coon	PI	0.02778640	19.09783	N/A
	PID	0.01851159	40.71712	2.67941
Skogestad-suggested Lambda tuning	PI	0.01321764	13.10000	N/A
2DOF PID tuning	Taguchi Araki	0.03995	32.95219	1.27665
	Hiroi Terauchi	0.031722329	53.0	11.13
Software tuning for PI	PI	0.026665086	18.50436	N/A
Direct Synthesis for SP(td=0)	PI	0.70053476	13.10000	N/A
Software tuning for SP	PI	0.055743566	4.364993	N/A
2DOF Sp controller	PI	0.082415854	13.1	N/A
		Sample time	Prd.H	Ctrl. H
	MPC values	1.31	50	2

Controller tuning information Simulation 2

		Test 2-1		
	Std. PID tuning	Kc	Ti	Td
Ziegler Nichols	PI	0.09146341	27.30600	N/A
	PID	0.12195122	16.40000	4.10000
Cohen Coon	PI	0.25616212	3.37945	N/A
	PID	0.14943777	9.82810	-1.05263
Skogestad-suggested Lambda tuning	PI	0.10162602	1.70000	N/A
2DOF PID tuning	Taguchi Araki	0.11503	12.12275	0.14263
	Hiroi Terauchi	0.085815508	19.0	3.36
Software tuned PI	PI	0.279170802	3.591187	N/A
Direct Synthesis for SP(td=0)	PI	0.98039216	1.70000	N/A
Software tuning for SP	PI	8.751199286	1.693186	N/A
2DOF Sp controller	PI	0.478240077	1.7	N/A
		Sample time	Prd.H	Ctrl. H
	MPC values	0.17	80	2
		Test 2-2		
	Std. PID tuning	Kc	Ti	Td
Ziegler Nichols	PI	0.14150943	17.64900	N/A
	PID	0.18867925	10.60000	2.65000
Cohen Coon	PI	0.35625416	2.87474	N/A
	PID	0.22450680	7.19208	-0.01415
Skogestad-suggested Lambda tuning	PI	0.15723270	1.70000	N/A
2DOF PID tuning	Taguchi Araki	0.52383	12.54145	2.50597
	Hiroi Terauchi	0.377358491	10.6	2.226
Software tuning for PI	PI	0.279170802	3.6	N/A
Direct Synthesis for SP(td=0)	PI	0.98039216	1.70000	N/A
Software tuning for SP	PI	1.797315151	0.521711	N/A
2DOF Sp controller	PI	0.924898261	1.7	N/A
		Sample time	Prd.H	Ctrl. H
	MPC values	0.17	80	2

Controller tuning information Simulation 3

		Test 3-1		
	Std. PID tuning	Kc	Ti	Td
Ziegler Nichols	PI	0.24420223	512.82000	N/A
	PID	0.32560297	308.00000	77.00000
Cohen Coon	PI	0.57733302	105.20269	N/A
	PID	0.38116941	231.26621	12.84527
Skogestad-suggested Lambda tuning	PI	0.27133581	70.20000	N/A
2DOF PID tuning	Taguchi Araki	0.83316	206.72954	1.46069
	Hiroi Terauchi	0.65120594	308.00000	64.68000
Software tuned PI	PI	0.434650069	94.7	N/A
Direct Synthesis for SP(td=0)	PI	83.57142857	70.20000	N/A
Software tuning for SP	PI	2.281449744	29.770998	N/A
2DOF Sp controller	PI	1.276874386	70.2	N/A
		Sample time	Prd.H	Ctrl. H
	MPC values	7.02	60.00	2
		Test 3-2		
	Std. PID tuning	Kc	Ti	Td
Ziegler Nichols	PI	0.16714286	749.25000	N/A
	PID	0.22285714	450.00000	112.50000
Cohen Coon	PI	0.42321429	119.87204	N/A
	PID	0.26558036	302.88462	-2.26531
Skogestad-suggested Lambda tuning	PI	0.18571429	70.20000	N/A
2DOF PID tuning	Taguchi Araki	0.62331	563.84921	2.60414
	Hiroi Terauchi	0.364	535.5	94.5
Software tuning for PI	PI	0.359401282	110.7	N/A
Direct Synthesis for SP(td=0)	PI	83.57142857	70.20000	N/A
Software tuning for SP	PI	6.847517352	41.66643	N/A
2DOF Sp controller	PI	2.657279128	70.2	
		Sample time	Prd.H	Ctrl. H
	MPC values	7.02	60	2

Controller tuning information Simulation 4

		Test 4-1		
	Std. PID tuning	Kc	Ti	Td
Ziegler Nichols	PI	0.03500000	49.95000	N/A
	PID	0.04666667	30.00000	7.50000
Cohen Coon	PI	0.07249000	26.61786	N/A
	PID	0.05291625	32.07432	4.52896
Skogestad-suggested Lambda tuning	PI	0.03888889	35.00000	N/A
2DOF PID tuning	Taguchi Araki	0.10026938	25.46833	0.07204
	Hiroi Terauchi	0.07622	35.70000	6.30000
Software tuned PI	PI	0.04260	36.24437	N/A
Direct Synthesis for SP(td=0)	PI	1.17	35.00	N/A
Software tuning for SP	PI	0.165341062	10.8	N/A
2DOF Sp controller	PI	0.196078431	35	N/A
		Sample time	Prd.H	Ctrl. H
	MPC values	3.50	60	2/3

Controller tuning information 2DOF PID

2DOF-PID Taguchi coefficients		
Test/coef.	α	β
Test 1-1	0.510253342	0.570956
Test 2-1	0.244511557	0.613594
Test3-1	0.245288412	0.42187
Test 4-1	0.55348449	0.600856
Test 1-2	0.264313624	0.42839
Test 2-2	0.182507751	0.42672
Test3-3	0.180072387	0.430688

Appendix 2: Unused simulation result information tables

Test results compared to the time constant for Simulation environment 1

Test 1-1	Unchanging dead-time				Change in dead-time introduced at 13.1s			
Method/Time(s)	Rise Time	Settle time	Dist.Rej1	Dist.rej2	Rise Time	Settle tim	Dist.Rej1	Dist.rej2
ZN PI	-	-	-	-	-	-	-	-
ZN PID	3.664	8.178	6.233	6.793	2.977	8.376	5.743	5.953
CC PI	3.779	8.038	2.241	4.902	4.244	20.702	2.004	5.645
CC PID	8.248	8.248	6.163	7.198	8.166	8.166	5.944	6.583
SIMC PI	1.673	4.257	4.754	5.463	1.569	1.569	3.782	4.441
Softwate tuned PI	10.139	10.139	6.435	7.704	9.006	9.006	3.222	7.073
2DOF PID Taguchi	3.119	-	1.602	4.272	3.544	-	-	-
2DOF PID Terauchi 20%	5.496	5.496	4.894	5.931	4.282	8.321	2.069	5.645
Terauchi 2DOF PID 0%	8.702	8.702	9.064	9.805	7.269	8.779	8.033	9.665
SP Software tuned PI	1.435	3.664	6.176	6.176	1.382	8.294	2.136	3.481
SP DS tuning	6.489	6.542	6.335	7.023	-	-	-	-
2DOF SP	7.954	7.954	6.771	7.656	7.710	7.710	6.313	7.580
MPC	1.450	1.450	6.420	7.350	-	-	-	-
Test 1-D	Unchanging dead-time				Change in dead-time introduced at 13.1s			
Method/Time(s)	Rise Time	Settle time	Dist.Rej1	Dist.rej2	Rise Time	Settle tim	Dist.Rej1	Dist.rej2
ZN PI	-	-	-	-	-	-	-	-
ZN PID	-	-	-	-	-	-	-	-
CC PI	3.771	9.545	2.015	6.014	1.786	-	2.000	5.992
CC PID	-	-	-	-	-	-	-	-
SIMC PI	1.597	4.368	7.089	7.886	1.521	8.397	3.573	8.092
Softwate tuned PI	8.473	9.771	8.514	9.792	5.728	8.779	3.038	9.924
2DOF PID Taguchi	9.871	9.871	8.118	9.317	9.322	9.322	8.039	9.159
2DOF PID Terauchi 20%	5.267	5.267	4.756	5.938	2.573	8.397	1.901	5.574
Terauchi 2DOF PID 0%	-	-	-	-	-	-	-	-
SP Software tuned PI	1.382	3.384	6.249	7.260	1.340	8.452	2.130	6.249
SP DS tuning	6.153	6.153	6.212	7.023	-	-	-	-
2DOF SP	7.656	7.656	6.534	7.366	7.481	7.582	6.061	7.582
MPC	1.356	5.003	6.310	7.191	-	-	-	-
Measured Disturbance	0.15	110.00						
Unmeasured disturbance	0.35	400.00						
Dead time	8.00	s						
Time variance step at riseup	2.00	s						

Test results compared to the time constant for Simulation environment 1

Test 2-1	Unchanging dead-time				Change in dead-time introduced at 1.3s			
Method/Time(s)	Rise Time	Settle time	Dist.Rej1	Dist.rej2	Rise Time	Settle time	Dist.Rej1	Dist.rej2
ZN PI	-	-	-	-	-	-	-	-
ZN PID	-	-	-	-	-	-	-	-
CC PI	8.041	30.680	14.197	14.596	6.921	-	10.393	-
CC PID	-	-	-	-	-	-	-	-
SIMC PI	13.219	33.241	19.412	19.426	10.879	-	17.228	19.556
Softwate tuned PI	7.324	35.353	13.445	12.966	6.388	-	13.254	9.806
2DOF PID	-	-	-	-	-	-	-	-
SP Software tuned PI	0.722	5.817	9.930	11.647	-	-	-	-
SP DS tuned PI	8.007	8.007	12.765	14.220	-	-	-	-
2DOF SP	7.969	7.969	12.082	13.799	4.994	42.183	13.058	11.905
MPC	2.659	2.659	11.238	12.266	-	-	-	-
Test 2-D	Unchanging dead-time				Change in dead-time introduced at 1.3s			
Method/Time(s)	Rise Time	Settle time	Dist.Rej1	Dist.rej2	Rise Time	Settle time	Dist.Rej1	Dist.rej2
ZN PI	-	-	-	-	-	-	-	-
ZN PID	-	-	-	-	-	-	-	-
CC PI	8.306	29.504	14.529	14.176	5.718	-	12.118	-
CC PID	-	-	-	-	-	-	-	-
SIMC PI	6.929	-	19.256	20.118	5.718	-	16.166	18.088
Software tuned PI	7.700	27.241	13.882	13.741	6.565	-	12.415	12.247
2DOF PID	-	-	-	-	-	-	-	-
SP Software tuned PI	11.078	11.078	16.738	15.712	-	-	-	-
SP DS tuned PI	13.948	13.948	13.278	18.582	-	-	-	-
2DOF SP	1.000	11.353	15.882	18.118	5.171	41.705	12.579	12.188
MPC	12.138	12.138	14.584	16.239	-	-	-	-
Measured Disturbance	0.15	30.00						
Unmeasured disturbance	0.35	97.00						
Dead time	8.20	s						
Time variance step at riseup	2.00	s						

Test results compared to the time constant for Simulation environment 3

	Test 3-1							
Results divided by Time Constant	Unchanging dead-time				Change in dead-time introduced at 70.2s			
Method/Time(s)	Rise Time	Settle time	Dist.Rej1	Dist.rej2	Rise Time	Settle time	Dist.Rej1	Dist.rej2
ZN PI	-	-	-	-	-	-	-	-
ZN PID	-	-	-	-	-	-	-	-
CC PI	2.714	21.329	6.168	6.244	2.632	36.112	6.313	7.146
CC PID	-	-	-	-	-	-	-	-
SIMC PI	5.998	20.812	9.630	9.620	4.993	22.372	8.819	8.852
Matlab Software tuned PI	3.979	14.866	7.816	7.597	3.544	29.198	8.047	7.413
2DOF PID	38.583	38.583	5.901	6.496	30.016	30.016	6.657	6.382
SP Matlab Software tuned PI	0.883	5.278	8.000	8.970	0.890	-	-	-
SP DS tuned PI	0.146	0.146	6.669	7.849	-	-	-	-
2DOF SP	8.732	8.732	10.385	10.385	2.667	17.913	6.375	6.375
MPC	0.212	0.644	7.369	8.650	-	-	-	-
Test 3-D	Unchanging dead-time				Change in dead-time introduced at 70.2s			
Method/Time(s)	Rise Time	Settle time	Dist.Rej1	Dist.rej2	Rise Time	Settle time	Dist.Rej1	Dist.rej2
ZN PI	-	-	-	-	-	-	-	-
ZN PID	-	-	-	-	-	-	-	-
CC PI	2.570	27.315	3.903	3.932	2.533	44.266	3.291	3.775
CC PID	-	-	-	-	-	-	-	-
SIMC PI	5.128	21.278	6.581	6.595	4.544	36.880	5.793	5.840
Software tuned PI	3.530	20.674	5.057	5.043	3.358	36.276	4.793	4.551
2DOF PID	34.561	34.561	6.792	6.709	13.484	39.144	6.802	6.332
SP Software tuned PI	0.850	9.957	3.240	3.089	-	-	-	-
SP DS tuned PI	0.620	10.259	3.089	3.047	-	-	-	-
2DOF SP	2.806	12.407	4.444	4.630	2.490	23.754	3.433	3.519
MPC	0.708	5.905	5.398	5.382	-	-	-	-
Measured Input Disturbance	-10	1460.00						
Unmeasured Input disturbance	-36	2800.00						
Dead time	154.00	s						
Time variance step at riseup	38.50	s						

Test results compared to the time constant for Simulation environment 4

Test 4-1	Unchanging dead-time				Change in dead-time introduced at 35s			
Method/Time(s)	Rise Time	Settle time	Dist.Rej1	Dist.rej2	Rise Time	Settle time	Dist.Rej1	Dist.rej2
ZN PI	15.743	15.743	14.567	15.270	15.306	15.306	13.686	15.029
ZN PID	1.228	4.686	3.371	3.349	1.217	7.186	2.986	2.991
CC PI	0.434	12.300	1.477	1.477	0.334	16.113	1.513	1.603
CC PID	0.816	4.486	3.686	3.774	0.772	3.700	3.015	3.000
SIMC PI	1.174	4.455	7.571	8.143	1.157	4.357	7.321	8.088
Software tuned PI	1.002	6.540	7.857	8.571	1.129	13.727	7.321	3.102
2DOF PID Taguchi	3.291	26.831	1.172	3.600	3.530	-	-	-
2DOF PID Terauchi 0%	5.119	5.119	4.657	5.029	6.014	6.014	4.457	5.471
SP Software tuned PI	0.349	2.852	6.705	7.396	0.249	5.357	6.286	7.314
SP DS tuned PI	0.263	0.263	6.771	7.371	0.247	-	-	-
2DOF SP	9.114	9.114	7.171	7.743	8.957	8.957	6.986	7.586
MPC	0.155	0.566	6.975	7.565	-	-	-	-
Test 4-D	Unchanging dead-time				Change in dead-time introduced at 35s			
Method/Time(s)	Rise Time	Settle time	Dist.Rej1	Dist.rej2	Rise Time	Settle time	Dist.Rej1	Dist.rej2
ZN PI	16.997	16.997	14.743	15.286	16.614	16.614	14.400	15.286
ZN PID	1.371	4.999	3.543	3.571	1.301	7.437	3.157	3.143
CC PI	0.440	11.203	1.577	1.571	0.340	25.776	1.529	1.535
CC PID	0.923	5.057	4.229	4.314	0.904	4.671	3.943	4.286
SIMC PI	1.383	7.863	8.114	8.947	1.311	7.471	7.918	8.589
Software tuned PI	1.108	8.114	8.314	8.971	1.103	4.586	8.071	8.643
2DOF PID Taguchi	0.470	36.971	1.064	1.191	-	-	-	-
2DOF PID Terauchi 0%	5.714	5.714	5.057	5.371	4.871	4.871	4.586	5.674
SP Software tuned PI	0.362	6.097	6.943	7.537	0.262	4.554	6.900	7.414
SP DS tuned PI	6.431	6.431	7.421	7.754	-	-	-	-
2DOF SP	9.543	9.543	7.300	7.900	9.557	16.171	16.171	16.171
MPC	0.164	6.464	7.254	7.874	-	-	-	-

Measured Disturbance	0.1	615.00
Unmeasured disturbance	-0.25	1465.00
Dead time	15.00	s
Time variance step at riseup	3.50	s

Cross examination results without being compared to the time constant

	Test 1-2							
Results	Unchanging dead-time				Change in dead-time introduced at 13.1s			
Method/Time(s)	Rise Time	Settle tim	Dist.Rej1	Dist.rej2	Rise Time	Settle tim	Dist.Rej1	Dist.rej2
ZN PI	-	-	-	-	-	-	-	-
ZN PID	-	-	-	-	-	-	-	-
CC PI	32.100	251.023	76.903	78.546	31.430	268.041	88.317	82.720
CC PID	-	-	-	-	-	-	-	-
SIMC PI	72.600	176.711	119.922	118.700	62.200	169.876	104.830	103.105
Matlab software tuned PI	65.600	255.610	108.795	110.092	57.800	-	97.500	104.931
2DOFPID Taguchi	-	-	-	-	-	-	-	-
2DOFPID Terauchi	-	-	-	-	-	-	-	-
SP Matlab Software-tuned PI	18.750	75.573	66.321	117.098	19.029	258.137	66.272	70.200
SP DS tuning	62.101	62.101	111.687	118.135	-	-	-	-
2DOF SP	123.500	123.500	123.085	134.600	33.100	206.500	72.500	76.700
MPC	4.865	16.914	111.108	122.320	-	-	-	-
	Test 2-2							
Results	Unchanging dead-time				Change in dead time introduced at 1.7s			
Method/Time(s)	Rise Time	Settle tim	Dist.Rej1	Dist.rej2	Rise Time	Settle tim	Dist.Rej1	Dist.rej2
ZN PI	-	-	-	-	-	-	-	-
ZN PID	-	-	-	-	-	-	-	-
CC PI	7.310	40.808	14.800	16.236	6.860	61.180	15.349	18.200
CC PID	-	-	-	-	-	-	-	-
SIMC PI	14.510	36.455	42.866	22.172	12.360	70.200	20.494	17.600
Matlab software tuned PI	9.690	44.700	17.300	17.423	8.590	54.848	17.020	17.423
2DOFPID Taguchi	-	-	-	-	-	-	-	-
2DOFPID Terauchi	-	-	-	-	-	-	-	-
SP Matlab Software-tuned PI	1.167	9.658	16.854	17.152	-	-	-	-
SP DS tuning	13.520	13.520	18.843	22.753	5.780	64.376	13.996	14.530
2DOF SP	14.293	14.293	20.107	22.331	5.795	37.000	13.500	13.664
MPC	5.796	5.796	16.178	18.309	-	-	-	-
	Test 3-2							
Results	Unchanging dead-time				Change in dead-time introduced at 70.2s			
Method/Time(s)	Rise Time	Settle tim	Dist.Rej1	Dist.rej2	Rise Time	Settle tim	Dist.Rej1	Dist.rej2
ZN PI	-	-	-	-	-	-	-	-
ZN PID	-	-	-	-	-	-	-	-
CC PI	313.500	1700.659	632.026	628.542	301.500	1850.984	634.800	633.094
CC PID	-	-	-	-	-	-	-	-
SIMC PI	617.000	2186.031	939.089	939.089	564.800	2283.598	886.331	892.882
Matlab software tuned PI	309.000	1521.283	633.094	633.094	300.500	2178.083	627.923	622.875
2DOFPID Taguchi	-	-	-	-	-	-	-	-
2DOFPID Terauchi	-	-	-	-	-	-	-	-
SP Matlab Software-tuned PI	387.000	1417.000	714.928	704.000	362.500	2011.500	690.923	691.500
SP DS tuning	10.000	10.000	720.538	709.942	-	-	-	-
2DOF SP	545.000	545.000	645.000	725.000	238.500	1256.500	530.000	502.500
MPC	32.300	75.500	591.042	678.000	-	-	-	-