ISSN 0280-5316
ISRN LUTFD2/TFRT--5845--SE

Temperature Control of an External Cleaning Unit

Marcus Collryd
Peter Svensson Valdt

Department of Automatic Control
Lund University
December 2009

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
December 2009

Document Number
ISRN LUTFD2/TFRT--5845--SE

Author(s)
Marcus Collryd and Peter Svensson Valdt

Supervisor
Daniel Cederström Tetra Pak, Lund
Prof. Tore Hägglund Automatic Control, Lund (Examiner)

Sponsoring organization

Title and subtitle
Temperature Control of an External Cleaning Unit (Temperaturreglering av en extern diskstation)

Abstract
The thesis work was performed at Tetra Pak where a temperature control system for an external cleaning unit was designed. The external cleaning unit is responsible for the heating and supply of cleaning fluids for an aseptic packaging and filling machine. This thesis presents a least squares estimated model predictive controller that uses an ILC inspired algorithm to predict the process behavior. The control problem countered in this thesis is a long variable time delay combined with fast process dynamics and load disturbances. Development of the prediction model was done using Matlab and Simulink, with the ability to use logged process data from the process PLC system in Simulink for verification. The model equations describe the actual physical relations in the process and are easily modified giving the ability to model several different outputs and inputs to the process. These equations were used to predict the main process output in Simulink. In the control system, the equations were modified to calculate the correct control signal and acted as a feedforward. Since the processes varies between units, a least squares estimation program was used to calculate the matching feedforward model to that specific unit. To improve the performance of the feedforward, an ILC algorithm was used to predict the process behavior during large transients. The complete control system and least squares algorithm was implemented using structured text on a Rockwell Logix PLC controller.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
61

Recipient's notes

Security classification

http://www.control.lth.se/publications/

Contents

1. Background 1

2. Process and Problem Description 2
- 2.1 The External Cleaning Unit 2
- 2.2 The Test Machine 3
- 2.3 Main Control Loop 4
- 2.4 Solution 4

3. The Simulink Model 6
- 3.1 Introduction 6
- 3.2 Water Mass Model 8
- 3.3 Energy Model 10
- 3.4 Time delays 16
- 3.5 Summary 18

4. Automatic Tuning of Model Parameters 20
- 4.1 Introduction 20
- 4.2 Automatic tuning of the Backlash 20
- 4.3 Automatic tuning of Water Mass Model 22
- 4.4 Automatic tuning of the Energy Model 24
- 4.5 Model Validation 28

5. Controlling the Process 30
- 5.1 The Control Problems 30
- 5.2 Overview of the Control System 32
- 5.3 Feedforward 33
- 5.4 Feedback 34
- 5.5 Flow Step Storage 35
- 5.6 Flow Prediction 36
- 5.7 Results and Discussion 39

6. Improvements of the ECU 41
- 6.1 Chapter Introduction 41
- 6.2 Temperature Sensor B0232 41
- 6.3 Steam Injector 41
- 6.4 Waterpressure 44
- 6.5 Filtering of Water Flow Measurement 45

7. Summary 46

A. Appendix 48
- A.1 Cleaning Sequence 48
- A.2 Process Actuators and Sensors 49

B. Adaptive Model 51
- B.1 Chapter Overview 51
- B.2 RLS Estimated Model 51
- B.3 Compensated Model 53

1. Background

Tetra Pak is a company specialized in packaging and distribution of food packages. Since the start in 1951 the packages have undergone numerous redesigns and improvements. One area in the development of new products is the aseptic technique, for example the "long life" milk. The creation of such products requires, amongst other things, that the packaging and filling environment is sterile. To ensure a sterile environment, a cleaning process is used. This process uses a controller to make sure that the cleaning fluids has the correct temperature. The main task for this master thesis was to develop a controller that keeps the temperature at the specified level, whether the machine operates in Europe or any other part of the world.

2. Process and Problem Description

This thesis is centered around the new A6 packaging and filling machine manufactured and designed by Tetra Pak. The A6 machine manufactures and fills a new antiseptic package for beverages. To ensure a clean environment inside the filling machine an external cleaning unit is used, referred to as the ECU. The main purpose of the ECU is to mix and heat the necessary cleaning fluids for the A6 machine. The main focus of this thesis lies on the temperature control of those cleaning fluids originating from the ECU.

2.1 The External Cleaning Unit

The basic function of the ECU is to quickly and accurately heat tap water used to clean the A6 machine. This heated water can be used directly to wash off pollutants and chemicals, or it can be mixed with strong cleaning chemicals before it is used inside the A6 machine. The temperature of the water effects the efficiency of the chemicals. When it is too high, the chemical injection equipment could be destroyed. When the water is used for rinsing it is more efficient if the water temperature is high enough to dissolve the pollutants and the chemicals inside the machine. On the contrary, if the temperature is too high it will burn milk products solid inside the machine. Therefore it is important that the temperature control is robust and precise during operation to guarantee an aseptic environment inside the A6 machine. In order to accomplish this, Tetra Pak designed the ECU using a large amount of components and control loops to assure its functionality and performance. The most important components are described in the following section and are visible in figure 2.1.

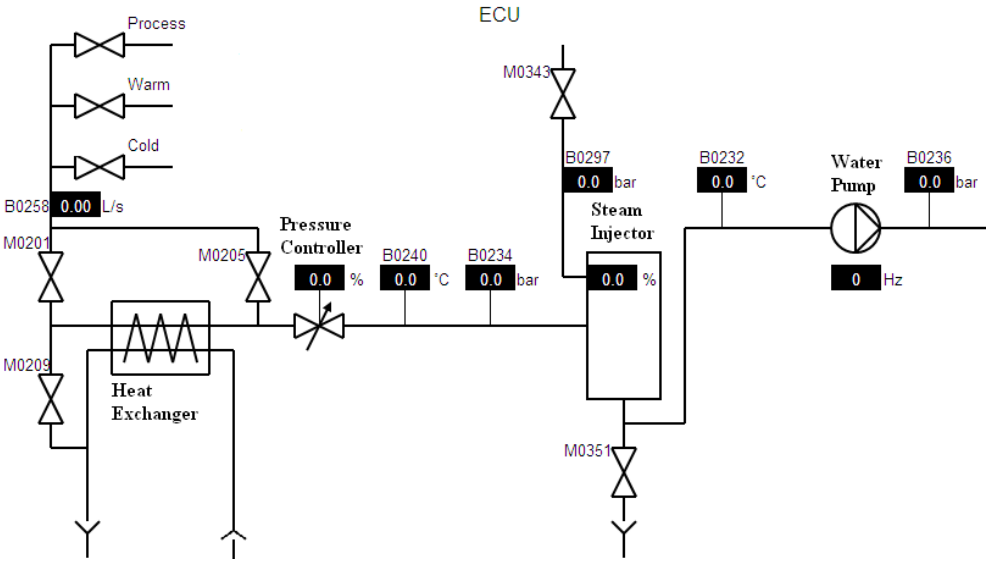

Figure 2.1 Overview of the ECU

The main component of the ECU is the steam injector used to heat the cleaning water. In order to be energy efficient, the steam injector was paired with a passive heat

exchanger using returning process water from the A6 machine to heat the incoming tap water to the ECU. To control the water pressure in the steam injector, a standard pressure controller was fitted between the heat exchanger and the steam injector. Additionally a pump was mounted after the steam injector to control the outgoing water pressure, measured by pressure sensor B0236. This pump limits the water flow thus indirectly controlling the water flow through the entire ECU by forcing the pressure controller to limit the ingoing water flow to maintain nominal water pressure around the steam injector.

The most important measurable signals are present in figure 2.1, referred to by their sensor names and placements. B0258 is the water flow (q) sensor measuring the amount of tap water being fed to the process, making the heated mass directly measurable. Sensor B0240 measures the water temperature before the steam injector (T_{in}) while the B0232 sensor measures the temperature after (T_{out}), making it possible to calculate the temperature difference caused by the steam injector. Additionally the water pressure is measured by sensor B0234, while sensor B0297 measures the steam pressure (S_p) feed to the steam injector.

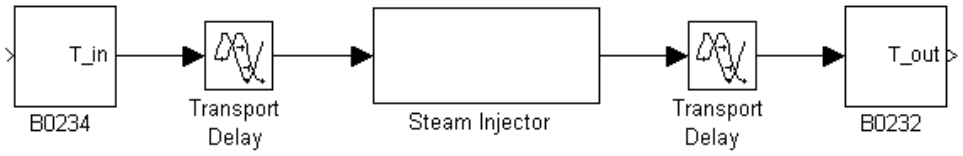

Figure 2.2 Overview of the steam injector and the adjacent temperature sensors

During the course of the thesis, the actuators manually controlled was the water pump and the steam injector. Additionally valve M0201 and M0205 were used to bypass the heat exchanger when necessary, leaving the rest of the actuators to be controlled by preprogrammed control loops made by Tetra Pak.

For more information about the steam injector and the main sensors see appendix A.2.

2.2 The Test Machine

During the thesis work, a full scale experimental A6 machine was not used to develop the ECU control program. Instead a smaller test machine was used to simulate the A6 machine during the daily work while the real A6 machine was only used to verify the results. Since the main focus lies on the control of the ECU the only important qualities of the A6 machine are the flow and pressure load it generates on the ECU.

The A6 machine is 13m long and is divided into several cleaning zones, each with their own individual cleaning nostrils and distances from the ECU. Each zone has its own individual water flow and water pressure, creating a varying load on the ECU. To emulate this behavior the same cleaning nostrils were used in the test machine but the amount was reduced. Making the test machine more compact, which resulted in lower water flow. This was compensated for by using a system of flow valves to drain additional water making the load on the ECU as similar as possible.

The test machine is roughly 2m long and situated right next to ECU making the pipe length significantly shorter than on the A6. This made the behavior during zone changes worse, increasing the spikes and drops in water temperature greatly because of large transients in the water pressure. The long piping on the A6 machines expands and dampens the large pressure spikes. These pressure transients originate from the

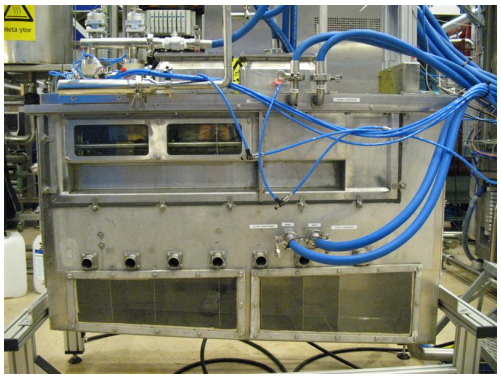

Figure 2.3 Side view of the test machine.

sudden opening and closing of valves directing the cleaning fluids to different cleaning zones.

The ECU and test machine as a unity will further be referred to as "test rig".

2.3 Main Control Loop

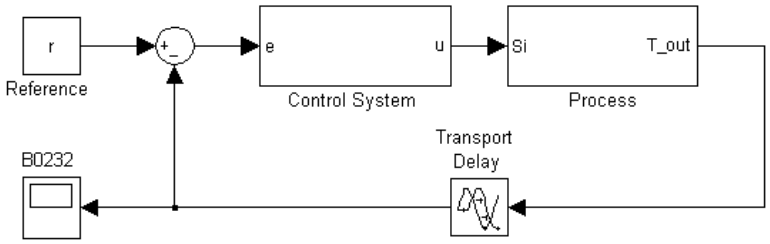

Figure 2.4 Main control loop for the ECU

At the core of the ECU lies the control loop formed by the steam injector and temperature sensor B0232, see figure 2.4. The effect of the control signal to the steam injector (Si) is directly measurable by the B0232 temperature sensor, creating a simple system to control. But since the temperature sensor can not be placed in the middle of the steam injector, there is a time delay between the control signal and the temperature sensor. The time delay puts a limit on how aggressive the control system can be without being unstable, it also makes the control system react slower to load disturbances. Time delays severely limit the performance of control systems and therefore the controlled process can not be too fast or be disturbed by rapid load disturbances. Since the controlled process contains fast dynamics during step changes and the ECU heat exchanger creates rapid load disturbances, there is an apparent need for a time delay compensation.

2.4 Solution

To compensate for the time delay a model was derived that estimates the water temperature after the steam injector. When the model was developed a mass and energy concept was used, seeing the steam injector as an energy injector while the flow meter

was used to estimate the heated mass. The reference signal to the steam injector was used to estimate the amount of heat energy transfered. By knowing the transfered energy and the calculated mass, the temperature difference caused by the steam injector was estimated. Resulting in a control loop similar to the one in figure 2.5.

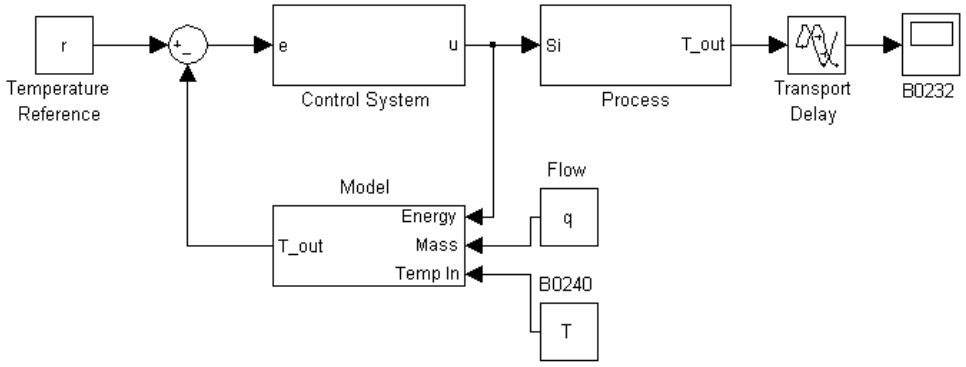

Figure 2.5 Overview of the model based control loop

The model of the system was built in Simulink using the user friendly block system to create a library of reusable ECU model components. The model parameters were calculated with the help of Matlab resulting in a respectable amount of m-files calculating the different relationships inherent to the ECU. For data collection and generation on the test rig RSLogix 5000 was used. A modern PLC development software containing everything necessary for creating and downloading software to the Logix PLC hardware controllers.

Verification of the model was first performed offline by doing dry runs on collected data from RSLogix 5000 inside Simulink, making it possible to directly see how well the model output matches the real process under the exact same circumstances . When the Simulink model produced results matching the real process, an identical discrete model was implemented using RSLogix 5000. This time verification could be done online. By using the model to predict the temperature without being in the control loop made it possible to log vast amounts of verification data. One could directly see how well the temperatures matched and if any problems occurred all the sensor data was available. Making it easy to trace the source and to improve the model. After the Simulink model had been verified online it was converted into the Feedforward model. Instead of estimating the temperature the Feedforward model estimated the control signal needed to follow the temperature reference. This effectively countered the effect of the time delay between the steam injector and the temperature sensor. In order to further improve the performance an ILC algorithm was implemented to predict and correct the behavior during zone changes.

3. The Simulink Model

3.1 Introduction

The purpose of the model is to predict the outgoing water temperature (T_{out}) from the ECU while the control system uses the steam injector to control the temperature, see figure 3.1.

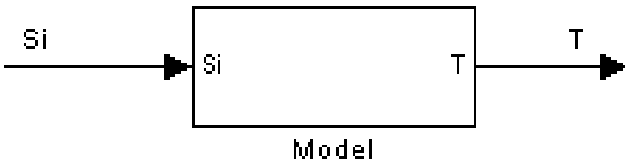

Figure 3.1 The principle for the model

However, there are several other external forces that affect the temperature, therefore a more advanced model has to be used. The temperature of the outgoing water is affected by the temperature of the incoming water as well as the flow and steam pressure. In figure 3.2, a model using S_i as the control signal to the steam injector, S_p as the steam pressure , T_{in} as the temperature of the incoming water and q as the water flow is presented.

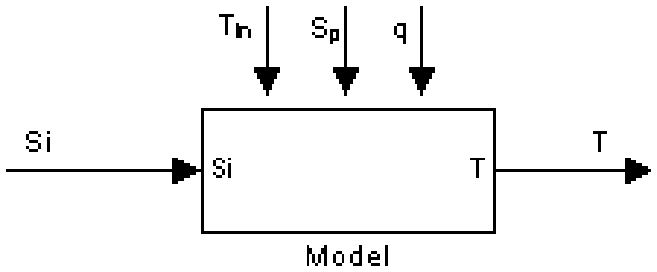

Figure 3.2 The principle for the model including the load disturbances

The starting point for derivation of a model was to divide the model into two parts (see fig 3.3). One part models the energy that is injected into the water by the steam injector and the other part models how much mass the injected energy heats up. By combining the mass and the energy it is possible to estimate the temperature difference. The first model uses the control signal (Si) and the steam pressure to calculate the amount of energy that is injected into the water. The second model uses the energy, the flow and the temperature of the incoming water to calculate the temperature of the outgoing water (see figure 3.3).

The easiest way to derive the model is to keep one of the blocks output fixed while calculating the output from the other block. Presuming that the blocks are multiplied together to form the temperature, the effect each block has on the temperature may be calculated one at the time with the exception of a scaling factor k. Physically this is true for the two blocks except for the temperature of the incoming water that directly

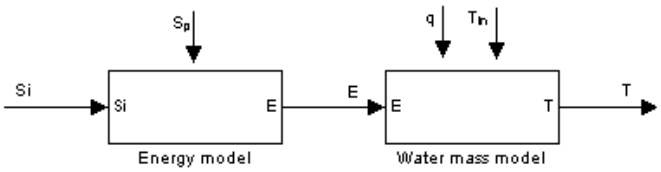

Figure 3.3 The model divided into two parts, an energy model and a water mass model

affects the temperature of outgoing water. The temperature of the incoming water is therefore separated from the water mass model (according to figure 3.4) and ΔT is then used as output instead of T_{out}.

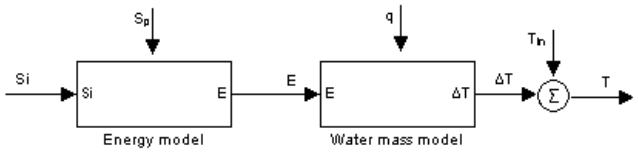

Figure 3.4 The model divided into two parts, an energy model and a water mass model, where T_{in} has been separated from the water mass model

The best way to estimate parameters in the water mass model and energy model is to keep all but one input fixed. This way, a simple relation between the input being varied and the output can be derived. Keeping Si fixed means that Sp will remain fixed as well thus the energy into the water mass model will remain the same regardless of the water flow. By varying the flow, the water mass model was calculated according to equation 3.1 where $m(q)$ is a function estimating the heated mass as a function of the water flow (q) and E is the constant energy.

$$\Delta T = \frac{E}{m(q)} \tag{3.1}$$

In order for the model to be unequivocal, the energy entering the water mass model has to be known. The water mass model was calculated using Si = 25 and therefore it was assumed that E(25) = 1 unit of energy. Derivation of the water mass model is further discussed in chapter 3.2. The energy model is more difficult to calculate then the water mass model. One reason is that when Si was changed, the flow changed as well. Therefore the temperature needed to be multiplied by the inverse effect of the water mass model, see eq. 3.2-3.3.

$$\Delta T = \frac{E(Si)}{m(q)} \tag{3.2}$$

$$E(Si) = \Delta T m(q) \tag{3.3}$$

When calculating the energy model both the flow varied and the steam pressure varied when Si changed. In contrast to the flow and Si, S_p can not be changed directly but is dependent only on Si. Another way of estimating the effect of the steam pressure was needed. Before the steam injector was changed (see chapter 6), it was linear between the interval between 10% and 90%. By using that the steam injector is linear, the effect of the steam pressure was calculated by slowly increasing Si and measuring the flow, temperature and steam pressure.

So far, all relations are static expressions and all measurements have been done when the temperature has stabilized. However, this is not the way the process works and a way of modeling the process-dynamics is required. It was proven to be enough to model the dynamics by a flow dependent low pass filter and flow dependent time delays (see page 16).

3.2 Water Mass Model

Overview

The water mass model block in the Simulink model (see figure 3.5) estimates the heated mass and calculates the temperature difference between the incoming water and the outgoing. It uses the water flow and the injected energy to calculate the temperature difference, adding the incoming water temperature gives the outgoing water temperature from the ECU.

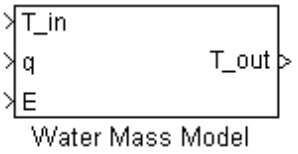

Figure 3.5 Overview of the water mass model block

Building the model

The amount of water flowing through the steam injector and the amount of steam being injected into the water influences the temperature of the outgoing water. By fixing one of them the influence of the other is singled out. The steam injector reference signal was therefore fixed at 25 percent while varying the water flow. This made it possible to measure how the water flow influences the temperature difference between the ingoing and outgoing water (see figure 3.6 for measurements).

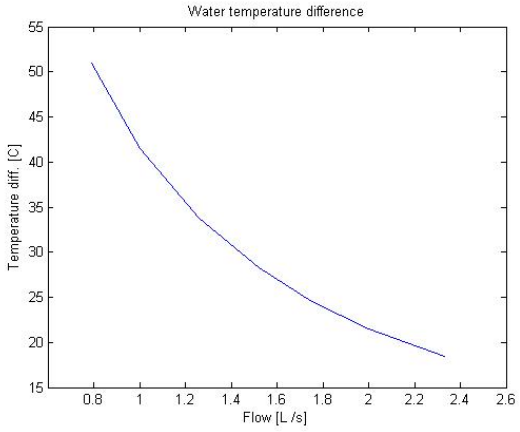

Figure 3.6 Water temperature difference as a function of the flow

It is logical that the temperature difference decreases when the flow increases, since an increase of the water flow increases the mass of water that the steam injector

has to heat. Doubling the mass results in half the temperature difference if the injected amount of energy remains the same. When plotting the temp difference as a function of the inverse of the water flow the function becomes almost linear, fitting well with the above stated reasoning (see figure 3.7).

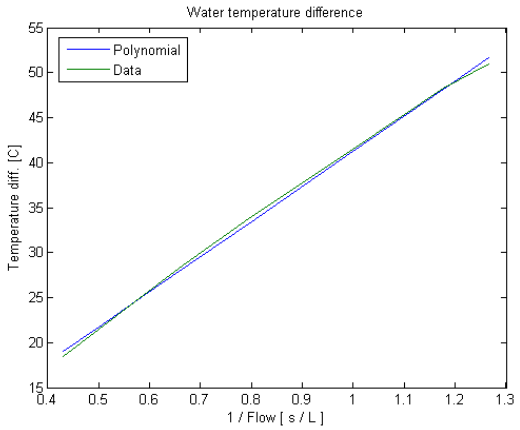

Figure 3.7 Water temperature difference as a function of the inverse of the flow

$$\Delta T = \frac{39.0164}{q} + 2.2326 \tag{3.4}$$

When matching a first degree polynomial (equation 3.4) to the curve it becomes apparent that the function is still bent, suggesting that the resulting heat transfer efficiency diminishes when the water flow decreases. This effect was caused by the heating of the metal water pipes when the water flowed from the steam injector to the temperature sensor. According to Newton the amount of heat transferred in each time instance is proportional to the temperature difference between the two objects. When there is a lower flow the water temperature is higher and the water flows slower through the pipes, resulting in the water being in contact longer with the metal pipes and also transferring more heat in each time instance.

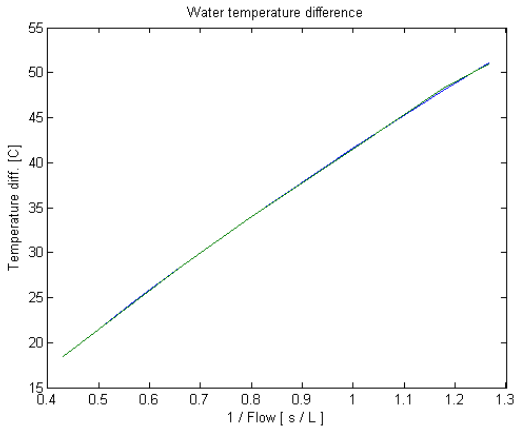

Figure 3.8 Second degree polynomial of the temperature difference

Using the polyfit function in Matlab resulted in well matching second degree polynomial, see figure 3.8 and equation 3.5.

$$\Delta T(q, Si_f) = -\frac{6.1251}{q^2} + \frac{49.4312}{q} - 1.6438 \tag{3.5}$$

Equation 3.5 only holds true for the fixed control signal (Si_f) used during the data collection.

Adding the Energy Input

To make things easier it was assumed that the steam injector was linear in its operation which it of course was not, resulting in the need to calculate an effective value of the injected energy (E) which compensates for the backlash and other non linearities inherent of the steam injector. This energy scale is closely connected to S_i and effectively linearizes the effect of the steam injector output. Adding the energy input (E) results in the finished water mass polynomial 3.6 where $E(Si_f)$ is scaled to the value one.

$$\Delta T(q,Si) = (-\frac{6.1251}{q^2} + \frac{49.4312}{q} - 1.6438)E(Si) \tag{3.6}$$

Using the fact that that the temperature difference follows the simple relation in equation 3.7 and the fact that $E(Si_f)$ is one gives equation 3.8 which estimates one divided by the heated mass (m).

$$\Delta T(q,Si) = \frac{E(Si)}{m(q)} \tag{3.7}$$

$$\frac{1}{m(q)} = -\frac{6.1251}{q^2} + \frac{49.4312}{q} - 1.6438 \tag{3.8}$$

3.3 Energy Model

Overview

The Energy Model models how the control signal to the steam injector affects the temperature by calculating the energy injected into the system. The model is separated into two blocks, Effective value calculation and Energy transformation (see figure 3.9). The Effective value calculation transforms the reference value (Si) to an effective opening (Si_e) corresponding to the actual opening and includes backlash and steam injector dynamics. The Energy calculation uses Si_e to calculate the amount of energy that is injected into the water.

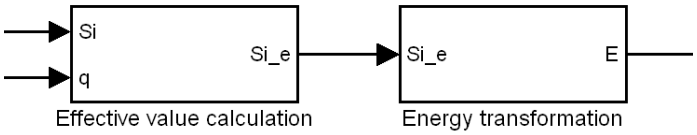

Figure 3.9 The Energy model divided into two parts

The model has the energy, E, as output but since there were no direct methods of measuring the energy, an indirect method was called for. Instead of measuring the energy, the temperature difference between then incoming and outgoing water (ΔT) was measured. The Energy model was derived by analyzing step responses where not only the input (Si) and the output (ΔT) was measured but the steam pressure and flow as well. The steam pressure affected the energy and needed to be included in the model. The flow did not affect the energy but affected the measured temperature and had to be compensated for.

Building the model

As a first step of deriving the model, it was assumed that $\Delta T = m(q_0) \cdot (k_{Si} \cdot Si + m_{Si})$ when q was kept fixed. The temperature and flow were measured when Si = 0 and then the temperature was measured when Si was increased in steps. When the data had been collected, the polyfit function in Matlab was used to estimate k and m according to equation 3.9.

$$k_{Si} \cdot Si + m_{Si} = \frac{\Delta T}{m(q_0)} \tag{3.9}$$

Figure 3.10 shows what the actual temperature and the modeled temperature became.

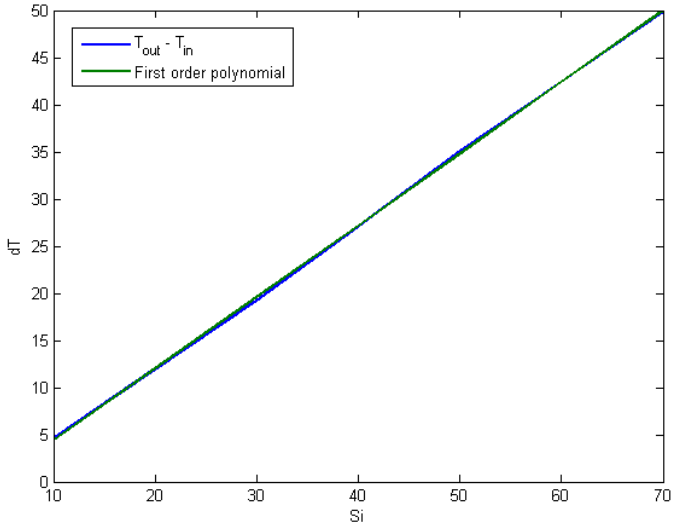

Figure 3.10 A first order polynomial fitted to the relation between Si and ΔT.

This looks like a close fit, but there are a few problems. The first problem is that the flow changes when Si is changed. This means that while the equation would model the temperature well if the flow was measured when Si = 0 and then kept the same, but it would not if the measured flow was used. This is due to the fact that the calculation assumed the flow to be q_0, for all Si, when it was not. Therefore, equation 3.9 has to be modified to include the flow according to equation 3.10.

$$k_{Si} \cdot Si + m_{Si} = \frac{\Delta T}{m(q)} \tag{3.10}$$

Notice that the equation is almost the same but for q instead of q_0 to symbolize that the flow is measured for each Si. The results of this is shown in figure 3.11.

This looks like a close fit as well, but there still are some problems. The first problem is that for large Si (Si>50%) the slope is decreasing. When analyzing the physical steam injector, it was concluded that the slope should instead be increasing. The conclusion was reached because the holes that lets the steam through are closer together for large Si than for small. To understand what happens, it is neccesary to plot $\Delta T/m(q)$ and the steam pressure in the same plot, see figure 3.12.

As seen in the figure, the steam pressure drops when Si increases. The steam pressure is supposed to be 4 bar all the times, but apparently the steam feed has problem providing the required amount. This means that while the opening slope

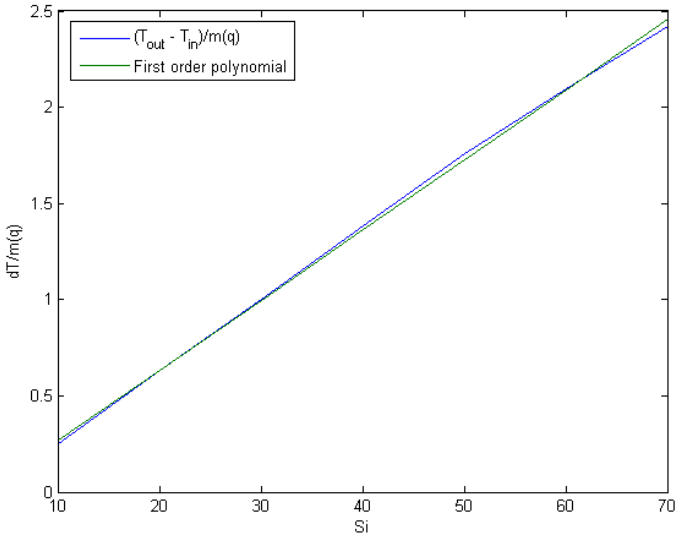

Figure 3.11 A first order polynomial fitted to the relation between Si and $\Delta T/m(q)$.

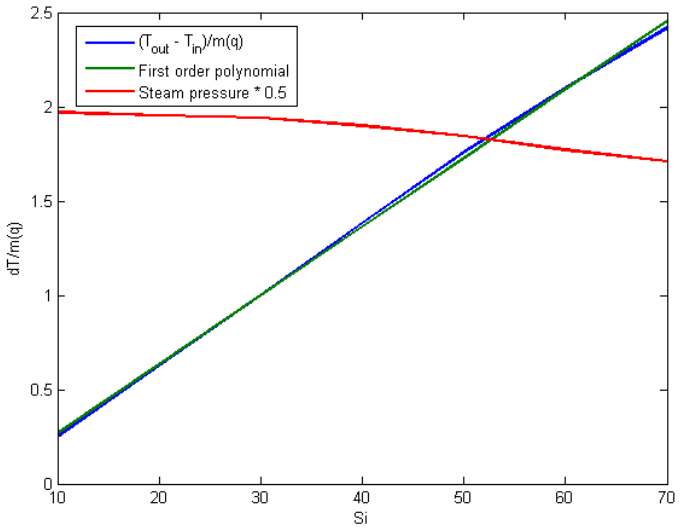

Figure 3.12 A first order polynomial fitted to the relation between Si and $\Delta T/m(q)$.

increases, the steam pressure drops to make the resulting energy slope decrease. Since the steam feed may vary from dairy to dairy, some way to compensate for the drop in steam pressure is required.

One way of doing this is simply to add a factor $\frac{S_p}{S_{pn}}$ in the equation, where S_p is the Steam pressure and S_{pn} the nominal steam pressure. For the calculations of Si to be correct, the steam pressure has to be included into the equation according to equation 3.11. As seen in figure 3.13, a first order polynomial matches quite well. This is normally a satisfactory result but since the energy estimate is multiplied with the water mass model and thus the error is also multiplied gives that a better approximation should be used. A more accurate second order polynomial of Si (see equation 3.12) is instead used to model how Si affects the energy (see figure 3.14, which due to the

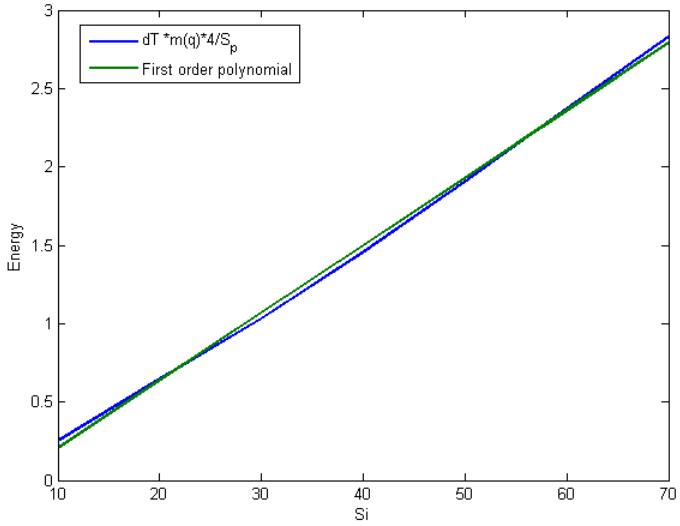

Figure 3.13 First order polynomial fitted to model the energy as a function of Si

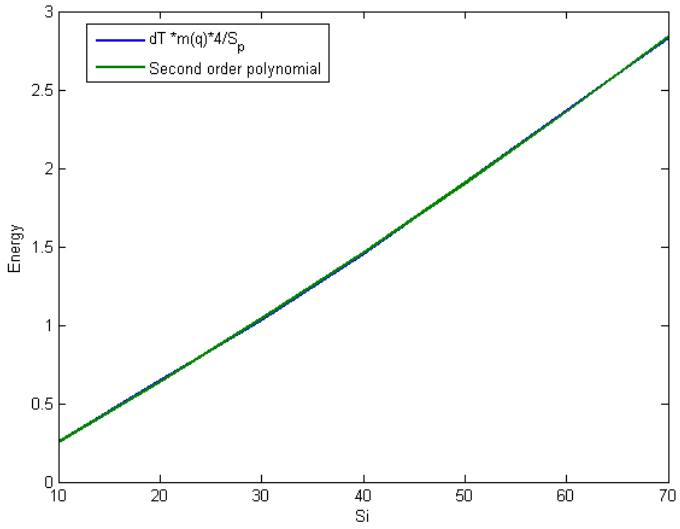

Figure 3.14 Second order polynomial fitted to model the energy as a function of Si

increasing number of holes is physically valid.

$$k_{Si} \cdot Si + m_{Si} = \frac{\Delta T}{m(q)\frac{S_p}{S_{pn}}} \tag{3.11}$$

$$k_{Si^2} \cdot Si^2 + k_{Si} \cdot Si + m_{Si} = \frac{\Delta T}{m(q)\frac{S_p}{S_{pn}}} \tag{3.12}$$

When comparing figure 3.13 and figure 3.14, it is clear that the second order polynomial is better and matches perfectly to the measured data. To verify this, a test run is done where the data collected is used as input data in simulink to estimate the temperature. A plot where Si, the real temperature and the model temperature are plotted is shown in figure 3.15.

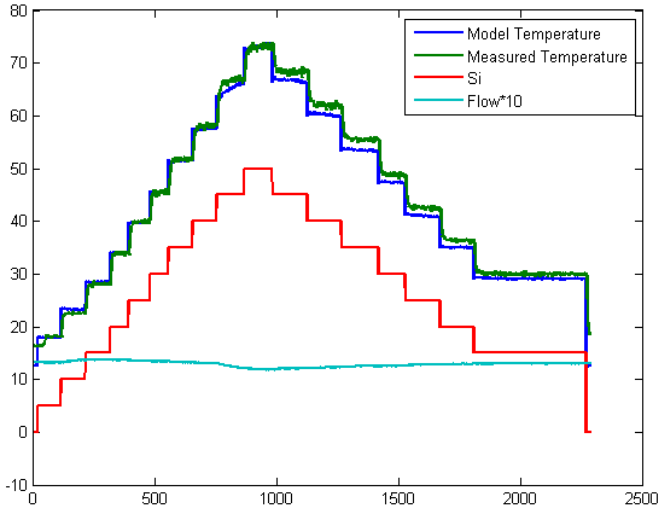

Figure 3.15 Comparison between the real temperature and the model temperature.

Studying figure 3.15, two things can be noted. The first problem is that while the model temperature matches the measured temperature very well when Si is increasing, there is a static error when Si decreases. When studying the same figure, it can be noted that the measured temperature differs for the same Si depending on whether it was reached from a higher or lower value.

This is a typical backlash phenomena which originates from the rubber seal in the steam injector tilting in the opposite direction when Si is increasing from when Si is decreasing. The easiest way to estimate the backlash is to reach the same temperature from a increasing Si and from an decreasing and compare the temperatures obtained. The simplest way of transforming the difference in temperature into Si is to linearize the temperature dependence of Si around the working point. First set $Si = 15$ from a lower value, then set $Si = 20$ and then $Si = 15$ again. Measuring the temperature in degrees Celsius at each step gives $T_{Si=15\uparrow} = 34.5$, $T_{Si=20} = 44.5$ and $T_{Si=15\downarrow} = 37.1$. The backlash is then calculated according to equation 3.13.

$$Si_{backlash} = \left(T_{Si=15\downarrow} - T_{Si=15\uparrow}\right) \cdot \left(\frac{20-15}{T_{Si=20} - T_{Si=15\uparrow}}\right) = 1.3 \tag{3.13}$$

Since the model is adjusted for rising Si, the backlash has to be implemented for when Si decreases. This is done by implementing a backlash block in simulink and then add half the backlash to the signal. This means that the effective Si is unaffected by the backlash when increasing while it is increased by the backlash when decreasing. The results from when the backlash is implemented are shown in figure 3.16

As seen in figure 3.16, the model now predicts the temperature more accurately. The second problem with the model is that a static equation is used to calculate the temperature. Since the steam injector does not instantly move from one position to another, the dynamics of the process has to be included into the model. A simple but surprisingly accurate model for the dynamics of the steam injector was a low-pass-filter. A first order butterworth filter (see equation 3.14) was chosen and modeled the dynamics well. It was implemented in the simulink model in the steaminjector block and made to fit a flow of 2.4l/s. However, when tried for other flows this simple

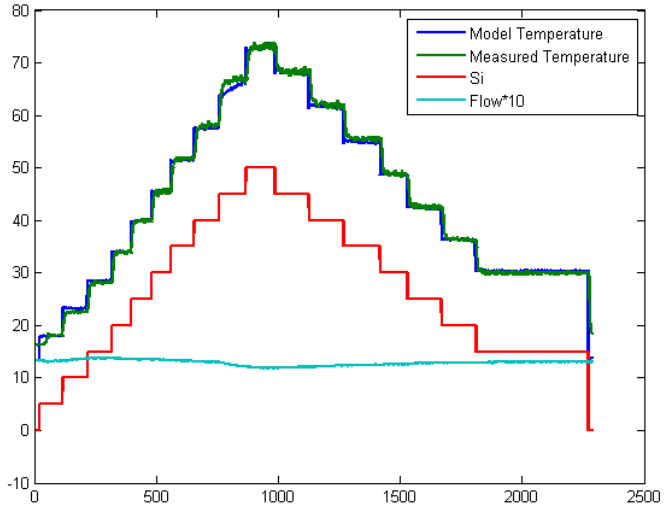

Figure 3.16 Comparison between the real temperature and the model temperature when backlash has been added.

model did not fit as well which was due to the heating of pipes etc. The filter is therefore dicretized according to the backward difference principle and implemented manually into Simulink. This way, the time constant in the butterworth filter can be flow dependent and the discrete butterworth filter can be implemented into the PLC-routine.

$$G(s) = \frac{1}{T \cdot s + 1} \tag{3.14}$$

Iteration of flow and control signal dependencies The method for deriving water mass model and energy model above uses only one calculation. When calculating the Water Mass Model, only one control signal was used. In order to improve the accuracy of the models, an iterative method is performed. As starting values, the previously derived values are used. To get better dependencies, the equations 3.15 and 3.16 are used iteratively. By using the last calculated flow parameters to calculate new control signal parameters and then using this new parameters to calculate a new flow dependence, the parameters are improved every iteration. This is done until the values have steadied themselves. However, it may for example happen that a and b may grow larger and c and d grow smaller since that will still satisfy the equations. If this happens, the parameters will not stabalize. This is solved by normalizing one coefficient, in this case, setting a to 1. b will then go toward a value as well as c, d and e.

$$\frac{a}{q^2} + \frac{b}{q} = \frac{\Delta T}{c \cdot S_i^2 + d \cdot S_i + e} \cdot \frac{S_{pn}}{S_p} \tag{3.15}$$

$$c \cdot S_i^2 + d \cdot S_i + e = \frac{\Delta T}{\frac{a}{q^2} + \frac{b}{q}} \cdot \frac{S_{pn}}{S_p} \tag{3.16}$$

Summary

In this chapter, a model to estimate how Si affects the temperature has been derived. This model, called the Energy Model, uses the steam input to estimate the amount of

energy injected into the water. In order to estimate the temperature from Si, compensation for flow and steam pressure was needed. The flow had to be compensated for since it varies when Si varies, leading to the water mass model being used during the estimation of the energy model.

The steam injector has a nonlinear curve regarding the distances between the holes. A nonlinear curve is therefore chosen to model how much energy that is injected into the water. It proved enough to use a second order polynomial to model the energy transfer from the steam injector to the water accurately.

3.4 Time delays

Overview

The time delays between the components of the ECU varies depending on the water flow. To be able to compensate for sudden changes in for instance the in temperature of the water the travel time between the sensor and the steam injector must be accurate. If the timing of the compensation is bad then a change in the control signal might make the situation worse.

Data collection and generation

With the help of the RS Logix 5000 built in logger the necessary data was collected, additionally the built in function for calculating differences between two data points was used for extracting the time data from the data logs. The relevant data was compiled into an excel file and later exported to Matlab for analysis and plotting.

To measure the travel time between the different temperature sensors and the steam injector it was needed to generate a step in the water temperature. It was decided to generate the step with the help of the steam injector or by switching between cold and warm feeding water to the process. With the available medium, sensors and actuators it was impossible to create perfect steps in the water temperature. Basic things like the response dynamics of the steam injector, heating of the material in and around the temperature sensors along with non-discrete maneuvering of the valves resulted in less than perfect step responses.

Results

Steam injector to B0232 Water temperature sensor B0232 is placed after the steam injector and thus the step was generated by the steam injector.

Plotting the data in Matlab resulted in 3.17 which suggest that the time delay is inverse proportional to the water flow. By switching the horizontal axis to one divided by the flow it is apparent that a first degree polynomial will describe the relation adequately. Using polyfit in Matlab to match a first degree polynomial to the data points results in figure 3.18.

The matching polynomial has the equation:

$$T(Si, B0232) = \frac{1.1143}{q} + 0.773 \tag{3.17}$$

B0240 to Steam injector To be able to calculate the water travel time between temperature sensor B0240 and the steam injector it was needed to generate a temperature step in the incoming water, this was accomplished by a fast switch between cold and

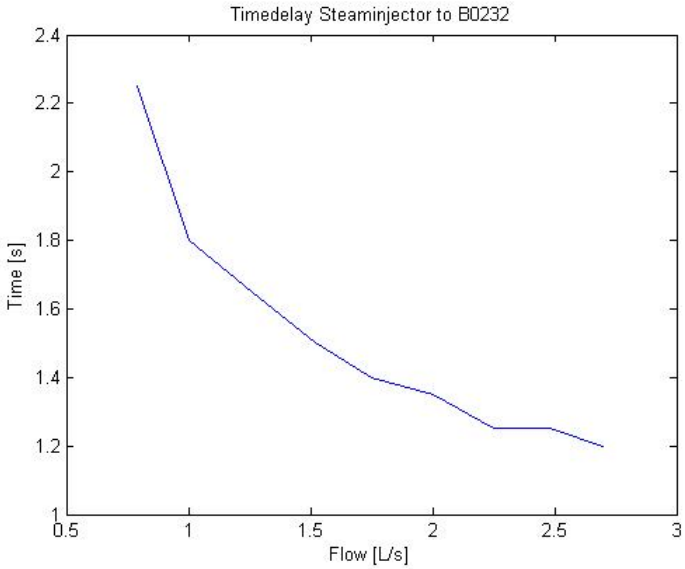

Figure 3.17 Time delay as a function of the water flow

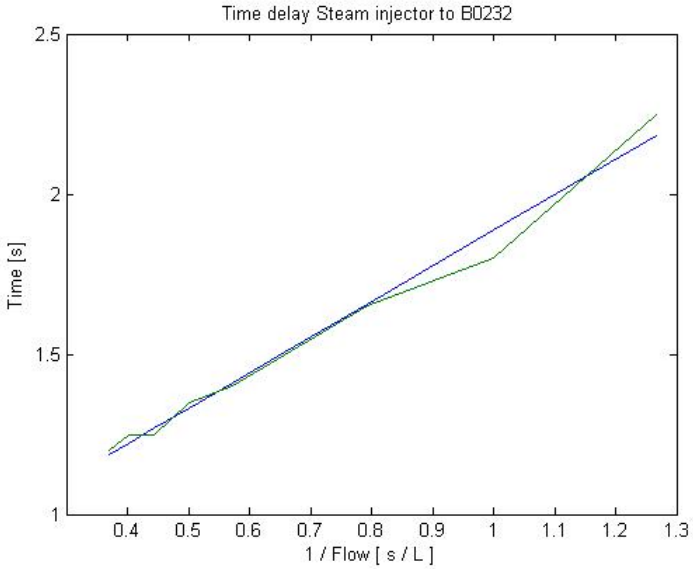

Figure 3.18 Time delay as a function of the inverse of the water flow

warm feeding water. The travel time between sensors B0240 and B0232 was measured, and by deducting the travel time between the steam injector and B0232 the travel time between B0240 and the steam injector was given.

The results in 3.19 are similar to the ones in 3.17 and the same method was used to produce a first degree polynomial describing the relationship between the time delay and one divided by the flow.

The resulting polynomial was:

$$T(B0240, B0232) = \frac{2.3721}{q} + 0.2150 \tag{3.18}$$

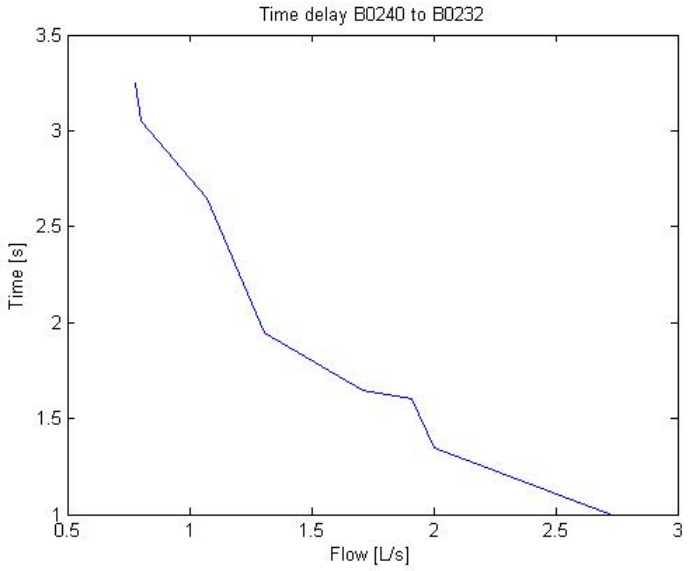

Figure 3.19 Time delay as a function of the water flow

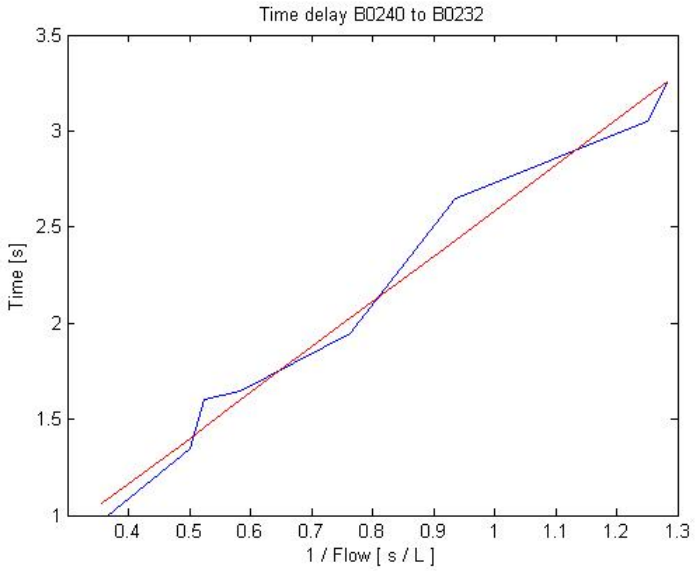

Figure 3.20 Time delay as a function of the inverse of the water flow

Finally by deducting the travel time between the steam injector and B0232 gave:

$$T(B0240, Si) = \frac{1.2578}{q} - 0.558 \tag{3.19}$$

3.5 Summary

In this chapter, a Simulink model has been derived. The model uses the available measurement signals to estimate the temperature. This was done by dividing the model into two parts, the water mass model and the energy model. Each of the parts are then derived one at a time, since the energy model could be kept constant while deriving

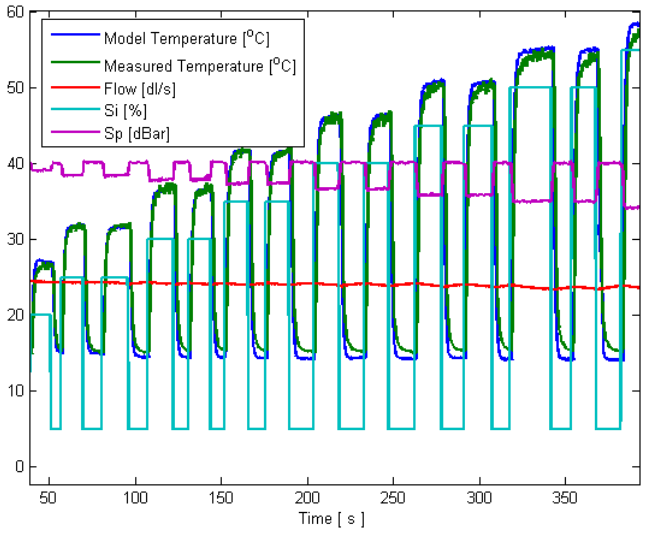

Figure 3.21 Test of the derived Simulink model

the water mass model. In the energy model the steam pressure had to be included as well as a backlash effect. The result from the model is shown in figure 3.21, where the model temperature was compared to the measured temperature. As seen in the figure, the largest temperature difference is around one degree which is well within the limits specified in chapter 5.

4. Automatic Tuning of Model Parameters

4.1 Introduction

The packaging and filling machine will be sent all over the world and therefore the conditions in which the machine will operate vary. The temperature of the incoming water as well as the flow and steam pressure are included in the model and the model compensates for variations in these. Differences in steam temperature and calibrations of the sensors may occur and some way to compensate for these are needed. This is done in the automatic tuning-sequence where the flow dependence and the steam injector dependence including the backlash is estimated.

The Main Principle

The principle for the automatic tuning is to use a sequence that starts by estimating the backlash, then the flow and the steam injector dependence. The automatic tuning uses the Least Squares method to estimate the flow and steam injector parameters while the backlash is estimated by reaching a certain steam injector level from both below and above and comparing the temperatures. In order to minimize computation spikes, as much of the calculations as possible are done when collecting the data.

4.2 Automatic tuning of the Backlash

Introduction

The backlash is an important part of the model, since it affects the precision of the model. The backlash is estimated by reaching a certain steam injector input from both a higher and lower value. If there is a temperature difference, it is due to the backlash. A simplified figure of how the backlash is estimated is shown in figure 4.1

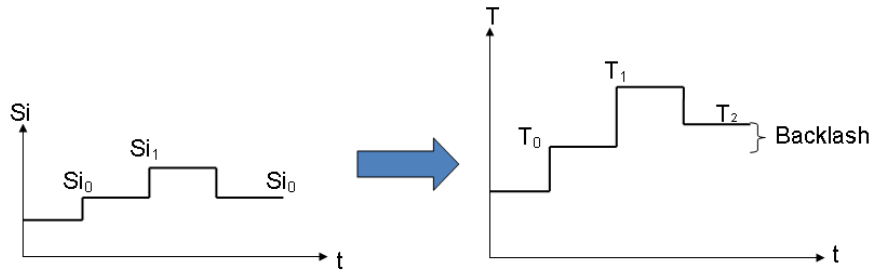

Figure 4.1 Backlash estimation principle

The backlash originates from the steam injector rubber seal. When moving up the seal tilts down and when moving down the seal tilts up. Depending on the stroke of the steam injector and how the steam injector is calibrated, the backlash varies. This means that when the steam injector is recalibrated or replaced, the backlash has

to be recalculated to be valid. The backlash may change when the rubber seal ages and therefore the backlash estimation is performed every time the automatic tuning sequence is run.

The principle

The principle for automatic tuning of the backlash is to first measure the temperature T_0 for $Si = Si_0$ when Si is reached from a lower value, then measure the temperature T_2 when the same Si is reached from a higher value. If there is no backlash, the temperature should be the same whether the input has been reached from a higher or lower value. If the temperatures are not the same, the temperature difference corresponds to the backlash. However, the temperature difference is not interesting but the difference in Si that it is equivalent to is. If $S_1 - S_0$ increases the temperature with $T_1 - T_0$, then $T_2 - T_0$ corresponds to an increase of $Si_{backlash}$ according to equation 4.1.

$$Si_{backlash} = (T_2 - T_0) \cdot \frac{Si_1 - Si_0}{T_1 - T_0} \tag{4.1}$$

When calculating the backlash it is assumed that the temperature depends linearly of Si between Si_0 and Si_1. It is also assumed that Si, including the backlash, is the only parameter changing that effects the temperature. The flow, for example, is assumed to be constant. This is almost true when Si_0 and Si_1 are close together. However, to be able to estimate the backlash, the difference between Si_0 and Si_1 must be larger than the backlash. Otherwise the full extent of the backlash will not be estimated. Another important matter is the measurement noise that inflicts uncertainty to the estimate. In order to reduce its effect, the flow should be as low as possible to make the temperature differences as large as possible. The larger the temperature increase that the backlash inflict is, the smaller the effect of the measurement noise will be and the more accurate the estimated parameters will be.

It is assumed that the backlash is the same whatever Si is. This is due to the physical interpretation of the backlash as the tilting of the rubber seal. It is reasonable to assume that the rubber seal tilts the same whether Si is 5% or 70%.

The algorithm

The principle for the backlash estimation algorithm is to use the Si in figure 4.1, where $Si_0 = 17$ and $Si_1 = 22$ from the start. When estimating the backlash, no process dynamics (such as pipe-warming/cooling) are wanted and the temperature is therefore measured when it has stabilized. For this purpose, the temperature is measured half a minute after Si has been changed. As mentioned in the principle, the flow should be as low as possible to reduce the effects of the measurement noise. In order to reduce the effect of the measurement noise even more, the temperature is low pass filtered. The mean value of four backlash estimations is then used as the backlash in the model.

Since only the steam injector should affect the temperature increase when calculating the backlash, the heat exchanger had to be disabled.

Results

When using the algorithm above, figure 4.2 was obtained. As seen in the figure, the temperatures for Si = 17 is different whether or not it has been reached from a higher value and thus there is a backlash effect. Using equation 4.1, the backlash is estimated to $Si_{backlash} = 1.3\%$.

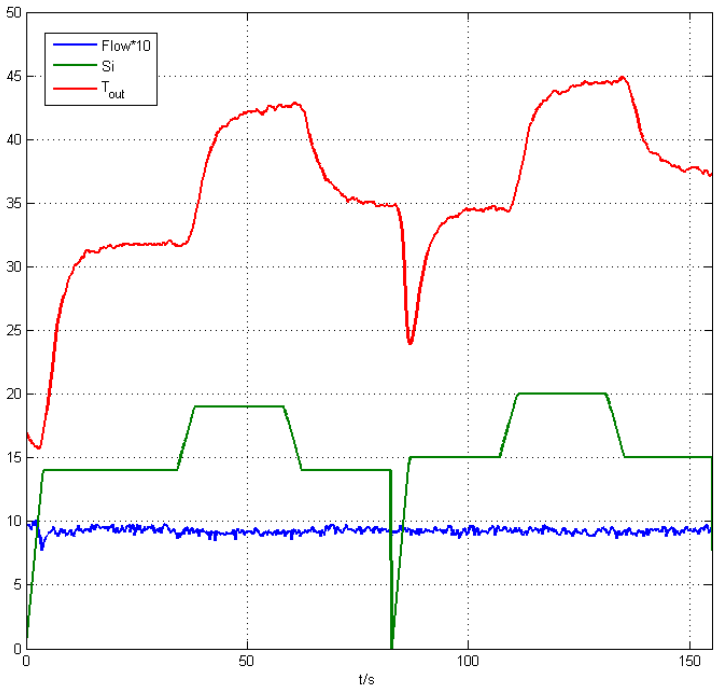

Figure 4.2 Automatic tuning of backlash. Notice that the same Si results in different temperatures

4.3 Automatic tuning of Water Mass Model

Overview

The automatic tuning algorithm for the Water Mass Model uses the same principle as was used when deriving the Simulink Model. It sets the Steam injector to a constant level and then measures the temperature difference while changing the flow in steps. It starts by setting the flow to maximum and then slowly reduces the flow in steps. In order for the measurements to be as free of noise as possible, the flow and temperature are low pass filtered. For stability reasons, the constant term of the Water Mass Model was removed since it has no physical interpretation and could therefore induce errors.

The principle

The basic idea when calculating the Water Mass Model was to use the same relation between the flow and the temperature as in the simulink model while calculating the dependence online. The least squares method is used to estimate the parameters. The principle is to keep the steam injector constant and change the flow in steps. Using the temperature difference (ΔT) as output (y) and $\frac{1}{q}$ as input u, the flow dependence was calculated according to equation 4.2 - 4.3.

$$\theta = \begin{bmatrix} k_{q^2} \\ k_q \end{bmatrix}, \phi = \begin{bmatrix} 1/q^2 \\ 1/q \end{bmatrix}, \Phi = \begin{bmatrix} \phi_1^T \\ \phi_2^T \\ \phi_3^T \\ \dots \end{bmatrix} \tag{4.2}$$

$$\hat{\theta} = \left(\Phi^T \Phi\right)^{-1} \Phi^T Y \tag{4.3}$$

By dividing the estimation of $\hat{\theta}$ into two parts according to equation 4.4-4.5, they can be calculated separately.

$$(\Phi^T\Phi)^{-1} = \left(\begin{bmatrix} \frac{1}{q_1^2} & \frac{1}{q_2^2} & \frac{1}{q_3^2} & \cdots \\ \frac{1}{q_1} & \frac{1}{q_2} & \frac{1}{q_3} & \cdots \end{bmatrix} \begin{bmatrix} \frac{1}{q_1^2} & \frac{1}{q_1} \\ \frac{1}{q_2^2} & \frac{1}{q_2} \\ \frac{1}{q_3^2} & \frac{1}{q_3} \\ \cdots & \cdots \end{bmatrix} \right)^{-1} = \begin{bmatrix} \sum \frac{1}{q_k^4} & \sum \frac{1}{q_k^3} \\ \sum \frac{1}{q_k^3} & \sum \frac{1}{q^2} \end{bmatrix}^{-1} \tag{4.4}$$

$$\Phi^T Y = \begin{bmatrix} \frac{1}{q_1^2} & \frac{1}{q_2^2} & \frac{1}{q_3^2} & \cdots \\ \frac{1}{q_1} & \frac{1}{q_2} & \frac{1}{q_3} & \cdots \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \cdots \end{bmatrix} = \begin{bmatrix} \sum \frac{y_k}{q_k^2} \\ \sum \frac{y_k}{q_k} \end{bmatrix} \tag{4.5}$$

In order to minimize the computation spikes the sums are updated every time new data are collected. By doing so, the number of computations needed to be done the last run as well as the memory usage is minimized.

Calculating the flow-parameters

When the data has been collected and summed together according to the sums in equation 4.2 - 4.3, the actual calculations of the parameters are performed. This is done as a last step of the estimation algorithm. Since no matrix-commands are available, the calculations have to be made manually. In the collection algorithm all the sums in the matrices were summed together, and therefore all that needs to be done is to take the inverse of $\Phi^T\Phi$ and multiply it with $\Phi^T Y$. The inverse of a 2by2 matrix is calculated according to equation 4.6.

$$A^{-1} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}^{-1} = \frac{1}{a_{11}a_{22} - a_{12}a_{21}} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix} \tag{4.6}$$

When calculating $\left(\Phi^T\Phi\right)^{-1}$, the following translations are done.

$$a_{11} = \sum(1/q^2) \tag{4.7}$$
$$a_{12} = a_{21} = \sum(1/q) \tag{4.8}$$
$$a_{22} = \sum(1) \tag{4.9}$$

When this is done, the inverse matrix above has to be multiplied with $\Phi^T Y$. Assume:

$$A^{-1} = \begin{bmatrix} a_{11}^* & a_{12}^* \\ a_{21}^* & a_{22}^* \end{bmatrix} \tag{4.10}$$

$$B = \begin{bmatrix} b1 \\ b2 \end{bmatrix} = \begin{bmatrix} \sum \frac{y_k}{q_k} \\ \sum y_k \end{bmatrix} \tag{4.11}$$

which gives:

$$\hat{\theta} = A^{-1}B = \begin{bmatrix} a_{11}^* b_1 + a_{12}^* b_2 \\ a_{21}^* b_1 + a_{22}^* b_2 \end{bmatrix} \tag{4.12}$$

$\hat{\theta}$ is the estimated parameters needed and the estimation is finished.

The algorithm

The automatic tuning of Water Mass Model is divided into two parts, one data collection and sum update part and one calculation part. The data collection and sum update part is a loop where a flow is set and the temperature difference is measured when the temperature and flow have stabilized. The flow is set by changing the speed of the high pressure pump and the temperature difference (ΔT) is calculated as $T_{out} - T_{in}$. Since there is a time delay between the incoming and outgoing water, it is critical that the temperature of the incoming water is constant. In order for the temperature of the incoming water to be as constant as possible, the heat exchanger was disabled. In order to reduce the effect of the measurement noise, the temperature difference was low pass filtered. When the new temperature has been measured, the sums are updated iteratively according to equation 4.13, where a_n is replaced with the elements in the sums in equation 4.2-4.3.

$$\sum_{k=1}^{n} a_k = \sum_{k=1}^{n-1} a_k + a_n \tag{4.13}$$

The loop continues to update the sums as long as the pump speed is above a predetermined value. When the pump speed has reached its limit, the data collection and sum update phase ends and the calculations are carried out. The calculations starts by calculating the matrix inverse according to equation 4.6. Then it calculates $\hat{\theta}$ according to equation 4.12, where $B = \Phi^T Y$ and $A = \left(\Phi^T \Phi\right)^{-1}$

Results

In figure 4.3 a run of the flow estimation sequence is shown. As seen in the figure, the temperature and flow needs to be low pass filtered to reduce the noise. The measurement of flow and temperature have to be performed when both have stabilized in order for the estimated model to be correct.

4.4 Automatic tuning of the Energy Model

Overview

The automatic tuning of the energy model estimates a relation between Si and temperature with the LS-method. As was the case with the flow dependence estimation, the LS-method had to be manually implemented. In contrast to when the water mass model was estimated, other factors are changed when changing the Si. This means that the temperature can not directly be used as output, instead the temperature multiplied with the inverse effect of the other factors are used. Another difference from the water mass model is that the energy model uses three variables that need to be tuned.

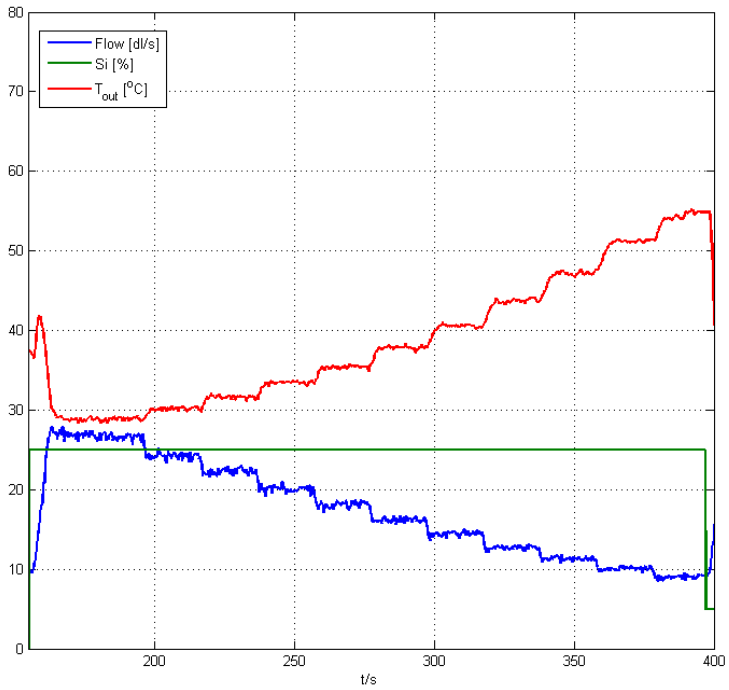

Figure 4.3 Automatic tuning of the Water Mass Model

The principle

The main principle for the automatic tuning of the energy model is the same as for the automatic tuning of the water mass model. The difference is that while the external factors are constant when estimating the water mass model, the flow (q) as well as the steam pressure (S_p) vary when the steam injector input is changed. In order to estimate the energy model, the temperature difference is multiplied with the inverse effect of the flow ($m(q)$) and steam pressure (S_p/S_{pn}, S_{pn} = nominal steam pressure) according to equation 4.14

$$y_k = E(Si) = \frac{\Delta T}{m(q)} \cdot \left(\frac{S_p}{S_{pn}}\right) \tag{4.14}$$

In order to minimize the computation spikes and the memory used during the tuning sequence, the sums are directly updated when new measurements of the flow, temperature and steam pressure are available. When the summation is finished, the inversion of the matrix $\left(\Phi^T\Phi\right)$ is multiplied with $(\Phi^T Y)$ and thereby $\hat{\theta}$ is obtained.

$$\begin{aligned}(\Phi^T\Phi)^{-1} &= \left(\begin{bmatrix} Si_1^2 & Si_2^2 & Si_3^2 & \dots \\ Si_1 & Si_2 & Si_3 & \dots \\ 1 & 1 & 1 & \dots \end{bmatrix} \begin{bmatrix} Si_1^2 & Si_1 & 1 \\ Si_2^2 & Si_2 & 1 \\ Si_3^2 & Si_3 & 1 \\ \dots & \dots & \dots \end{bmatrix}\right)^{-1} \\ &= \begin{bmatrix} \sum Si_k^4 & \sum Si_k^3 & \sum Si_k^2 \\ \sum Si_k^3 & \sum Si_k^2 & \sum Si_k \\ \sum Si_k^2 & \sum Si_k & \sum 1 \end{bmatrix}^{-1}\end{aligned} \tag{4.15}$$

$$\Phi^T Y = \begin{bmatrix} Si_1^2 & Si_2^2 & Si_3^2 & \dots \\ Si_1 & Si_2 & Si_3 & \dots \\ 1 & 1 & 1 & \dots \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \dots \end{bmatrix} = \begin{bmatrix} \sum y_k Si_k^2 \\ \sum y_k Si_k \\ \sum y_k \end{bmatrix} \tag{4.16}$$

The algorithm

The algorithm for automatic tuning of the Energy model uses the same principle as the algorithm for automatic tuning of the water mass model. The first part uses a loop that changes Si, waits a certain amount of time (until the temperature has stabilized) and then measures the temperatures, steam pressure and flow. Then it calculates y_k and adds Si_k, y_k to the appropriate sums, see equation 4.15-4.16. The temperature difference $T_{out} - T_{in}$ in y_k is heavily low pass filtered to get the average value.

The second part of the algorithm is the calculation part and it is started when Si has reached a predefined value or when the temperature has become too high.

Calculating the parameters

When the data has been collected and summed together, $\hat{\theta}$ is calculated. Since no matrix-operations are available on the PLC, the matrix-inversion and multiplication are made manually. When calculating the energy model, three parameters are estimated in contrast to when the water mass model was calculated and two parameters were estimated. Assuming the data has been summed together and using the notation in equation 4.17 - 4.21, the matrix inversion is calculated according to equation 4.22

$$a_{11} = \sum Si^4 \tag{4.17}$$
$$a_{12} = a_{21} = \sum Si^3 \tag{4.18}$$
$$a_{13} = a_{22} = a_{31} = \sum Si^2 \tag{4.19}$$
$$a_{23} = a_{32} = \sum Si \tag{4.20}$$
$$a_{33} = \sum 1 \tag{4.21}$$

$$\begin{aligned} A^{-1} &= \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}^{-1} \\ &= \frac{1}{\det(A)} \begin{bmatrix} a_{22}a_{33} - a_{32}a_{23} & a_{31}a_{23} - a_{21}a_{33} & a_{21}a_{32} - a_{31}a_{22} \\ a_{32}a_{13} - a_{12}a_{33} & a_{11}a_{33} - a_{31}a_{13} & a_{31}a_{12} - a_{11}a_{32} \\ a_{12}a_{23} - a_{22}a_{13} & a_{21}a_{13} - a_{11}a_{23} & a_{11}a_{22} - a_{21}a_{12} \end{bmatrix} \end{aligned} \tag{4.22}$$

$$\begin{aligned} \det(A) &= a_{11}\left(a_{22}a_{33} - a_{32}a_{23}\right) \\ &\quad + a_{12}\left(a_{31}a_{23} - a_{21}a_{33}\right) \\ &\quad + a_{13}\left(a_{21}a_{32} - a_{31}a_{22}\right) \end{aligned} \tag{4.23}$$

Rewriting A^{-1} according to equation 4.24 and multiplying it with B in equation 4.25 gives $\hat{\theta}$ according to equation 4.26.

$$A^{-1} = \begin{bmatrix} a_{11}^* & a_{12}^* & a_{13}^* \\ a_{21}^* & a_{22}^* & a_{23}^* \\ a_{31}^* & a_{32}^* & a_{33}^* \end{bmatrix} \tag{4.24}$$

$$B = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} \sum y_k Si_k^2 \\ \sum y_k Si_k \\ \sum y_k \end{bmatrix} \tag{4.25}$$

$$\hat{\theta} = A^{-1}B = \begin{bmatrix} a_{11}^* b_1 + a_{12}^* b_2 + a_{13}^* b_3 \\ a_{21}^* b_1 + a_{22}^* b_2 + a_{23}^* b_3 \\ a_{31}^* b_1 + a_{32}^* b_2 + a_{33}^* b_3 \end{bmatrix} \tag{4.26}$$

The calculations are implemented in PLC with a few exception. By replacing the elements in A^{-1} according to 4.17 - 4.21, the number of calculations needed are reduced since:

$$a_{12}^* = a_{21}^* \tag{4.27}$$

$$a_{13}^* = a_{22}^* = a_{31}^* \tag{4.28}$$

$$a_{23}^* = a_{32}^* \tag{4.29}$$

Implementing these on the PLC reduces the number of equations and therefore reduces the computation time.

Results

In figure 4.4, an automatic tuning sequence for the Energy model is shown. As seen in the figure, there are some measurement noise that is reduced by the low pass filter. It can also be noted that the flow varies and has to be be compensated for.

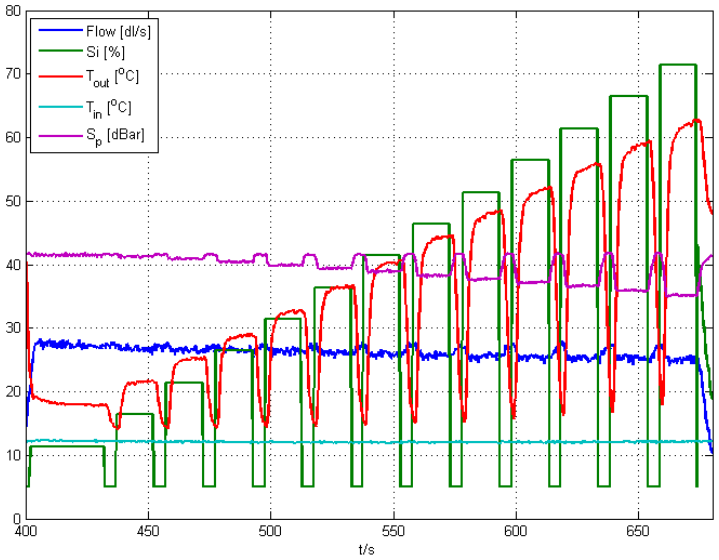

Figure 4.4 The results from an automatic tuning run for the energy model

4.5 Model Validation

Overview

In order to guarantee that the model obtained from the automatic tuning sequence will work, a validation sequence is used. The validation sequence uses the reference temperature, the temperature of the incoming water, the flow and the steam pressure to calculate the control signal to the steam injector. For the model to work, it has to predict the temperature for different control signals and flows. More correctly, the model has to estimate what control signal to give for given reference temperatures and flows. The model validation sequence verifies that this is the case with the estimated model. The sequence tries different flows and temperature references to make sure that the model outputs the correct control signal.

The principle

The model uses a feed forward-signal calculated based on the model. The feed forward uses an inverse model that instead of the measured temperature uses the reference temperature. In order to validate the model, the validation sequence uses different flows and reference temperatures to calculate the control signal. The measured temperature is then compared to the reference. If the temperature difference is within specified limits, the sequence uses a new Si and/or flow. If all flows and Steam injector inputs passes, the validation signals that the model is valid. If the temperature difference during one measurement is to large, the sequence aborts and signals that the model is invalid.

The validation algorithm

The validation sequence uses two different temperature references and four different flows. All eight combinations are tried by first setting a flow and trying the two different references. After the two different references have been tried, the flow is increased and the procedure starts all over again. This is repeated until the flow has reached the maximum flow, if the validation goes well. If the temperature differs to much, the sequence is aborted and signals that the validation process failed. If all temperature differences was within the limits, the sequence signals that the model worked as it should.

Results

In figure 4.5 a successful validation run is shown. The limits for the temperature is $\pm 3^{\circ}C$ from the reference when the temperature has stabilized. The reference used is $40^{\circ}C$ and $60^{\circ}C$ respectively for four different flows. The parameters in the test run are estimated using the automatic tuning sequence and since the model passed the validation sequence it is ready to be used as feedforward.

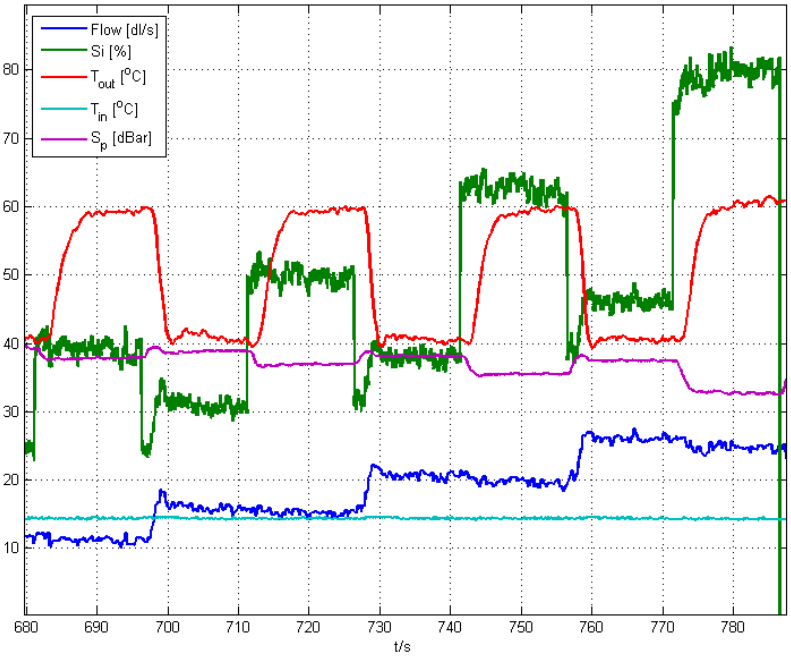

Figure 4.5 Validation run, using automatically tuned parameters, successful

5. Controlling the Process

5.1 The Control Problems

Introduction

The process contains two widely different control situations, caused by the two different cleaning modes foaming and warm rinse. During foaming the controlled process suffers from large time delays because of the low water flow, but there are no transients during zone changes and the heat exchanger is not used which results in the temperature in being constant. The lack of transients during zone changes is a result of the internal flow restriction in the ECU during foaming. A side effect of the low water flow is that the flow sensor works outside its specified interval, making the measurement signal unreliable. Additionally, the temperature reference is 40 degrees Celsius during foaming in comparison to the 55 degrees during warm rinse.

During warm rinse the time delay is smaller because of the high water flow, allowing a more aggressive feedback control. Since there are no internal flow restrictions in the ECU during warm rinse, the process suffers from fast and large transients during zone changes. Additionally, the heat exchanger is used to heat the incoming water which causes a rapid load disturbance.

The results and plots in this chapter are taken from data logged during runs on the full size prototype A6 machine called P1.

Foaming

The largest problem during foaming is the startup where the water flow fluctuates before it stabilizes at the restricted value. The unreliable readings from the flow sensor and the time-delayed temperature readings causes problems for both the feedforward and the feedback loop. See figure 5.1 for an example of a typical flow transient during startup.

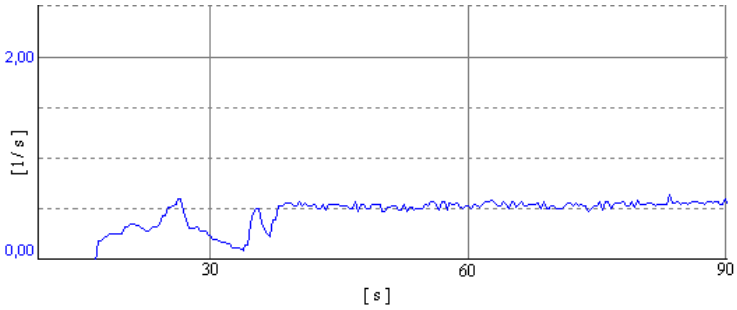

Figure 5.1 The water flow during foaming startup.

Warm Rinse

During startup of the warm rinse sequence, the water flow is quickly ramped up before stabilizing, see figure 5.2. In the figures, the water flow (q) is the blue line while the red line is the temperature of the incoming water (T_{in}). During startup, the water is quickly rushing through the machine while filling the empty pipes. The

readings from the flow sensor should be used with caution during this time since the lack of resistance might cause the water flow to be different through the steam injector and the flow sensor. When the water flow has stabilized, the next shock to the system comes from the in temperature that makes a large step as the returning cleaning water enters the heat exchanger. Since the temperature sensor (T_{in}) is placed close to the steam injector, the diffusion piston is not given enough time to move even though the control system reacts immediately to the new in temperature causing the temperature (T_{out}) to rise or fall during sudden changes in the in temperature (T_{in}).

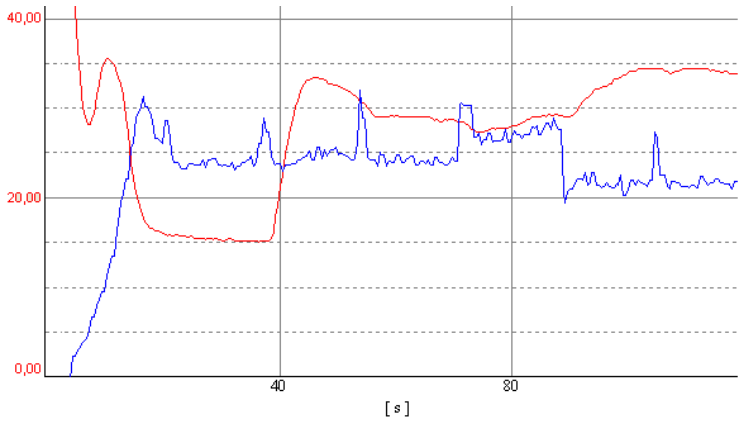

Figure 5.2 The waterflow (Blue) and in temperature (Red) during warm rinse startup.

During zone changes the flow valves are cycled, one opening and one closing. For a short while the two valves are open at the the same time causing the water flow to spike, see figure 5.3. Filling of empty pipes in the new zone might make the flow spike larger during the valve cycling. If the process goes from a high flow zone to a low flow zone then the water flow will transient quickly. First the flow spikes because of the valves opening and closing, then it quickly drops to the new zone flow. A side effect of the flow transients is that the temperature in will vary as well, since the returning flow of process water to the heat exchanger is constant while the heated water flow fluctuates.

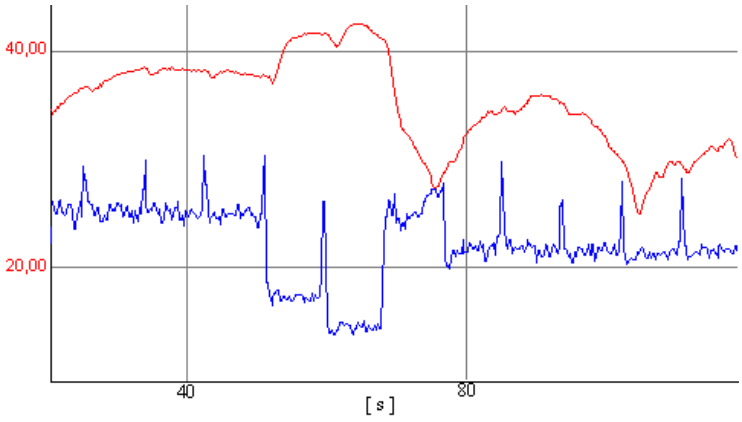

Figure 5.3 Warm rinse, normal zone changes. Water flow (Blue) Temperature in (Red)

Another variable that causes problem during temperature control is the water pressure inside the steam injector, since the relationship between the water pressure and the steam pressure controls how effective the steam injector is. The manufacturer recommends that the water pressure remains 2 bars below the steam pressure

to insure optimal effect and mixture, therefore the nominal water pressure is 2 bar while the steam pressure is 4 bar. See figure 5.4 where the steam pressure is the black line and the water flow is the blue line. During startup the water pressure is very low, causing the effect of the steam injector to be unpredictable as well as the flow measurement. Every zone change there is a pressure transient as the valves open and close, during zone changes where the flow difference is small this transient is negligible since it occurs while the water flow spikes and its effect is small in comparison. When the flow difference is big, especially when going from a high flow to a low flow zone, the pressure spike is larger. When the water pressure goes above 3 bar it starts to diminish the steam injectors out effect, which has the positive side effect that the chances for a temperature overshoot during a high to low flow zone change is reduced. If the water pressure falls then the effect of the steam injector increases, this is common during low to high flow zone changes. Unfortunately the behavior of the water pressure is not consistent between the cleaning runs, making a compensation for its effects difficult.

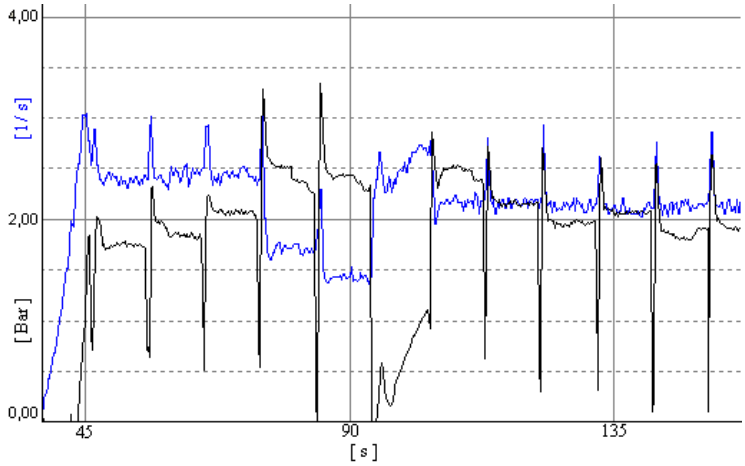

Figure 5.4 The water flow (Blue) and water pressure (Black) during warm rinse.

5.2 Overview of the Control System

To counter the temperature time delay, a feedforward model based upon the model derived in chapter 3 and 4 is used. This model uses the temperature reference, the incoming water temperature and the water flow to estimate a control signal for the steam injector. The rapid flow transients during zone changes is detected and stored using the flow step storage (FSS) subprogram. By using the stored values (q_s), the flow step prediction (FSP) subprogram predicts the water flow (q_p) immediately after a zone change. The water flow sent to the feedforward model is thereby changed before the actual transient occurs and the steam injector is given time to move the piston and adjust the amount of injected energy. This reduces the resulting temperature transient after a zone change and works due to the repetitive behavior of the flow.

The feedforward is combined with a feedback loop from the temperature out sensor (T_{out}) which consists of a slow PI controller that is strictly limited not to induce temperature oscillations when combined with the time delay.

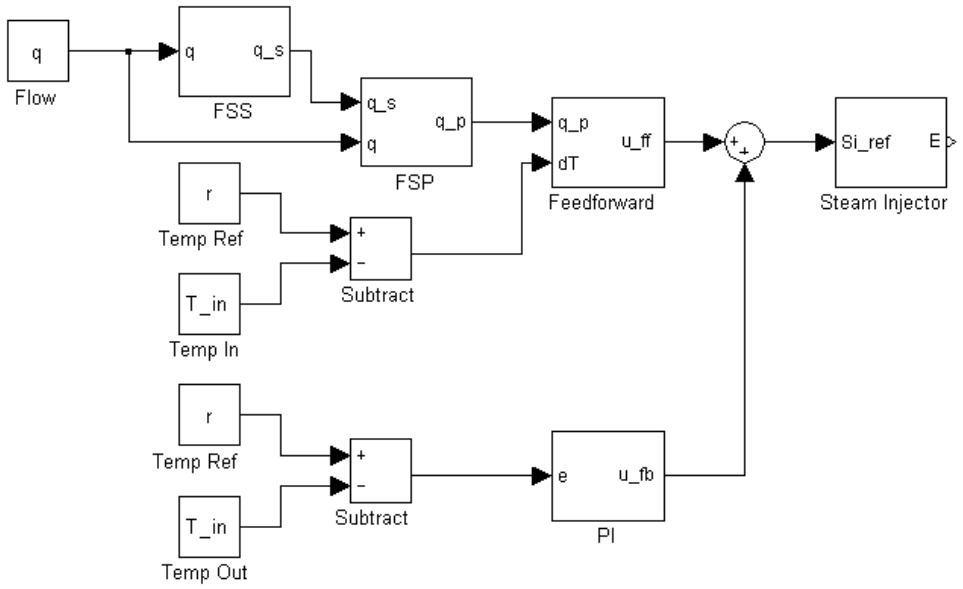

Figure 5.5 Overview of the control system.

5.3 Feedforward

The feedforward model is used to estimate the control signal needed to accomplish a desired temperature, unlike the Simulink model that estimates the out temperature using the water flow and control signal to the steam injector. To derive the feedforward model from the Simulink model, the quadratic formula is used (see equation 5.2). As an alternative to the feedforward solution there is the possibility to create a control loop using the estimated temperature from the Simulink model. This was tested but in the end the feedforward topology performed better and was calmer during flow transients. In order to be able to use the Simulink model for accurate control, the model needed to be adaptive. Appendix B covers some of the adaptive models tested, including an RLS solution.

$$x^2 + px + q = 0 \tag{5.1}$$

$$x_{1,2} = -\frac{p}{2} \pm \sqrt{\frac{p^2}{2} - q} \tag{5.2}$$

The Simulink model consists of the energy and mass polynomials, see equation 5.3. By assuming that ΔT is a user defined parameter and q is a measurement signal, the energy and mass polynomials can be rewritten as equation 5.4. After replacing E(Si) with the least square estimation polynomial and rewriting the equation the second degree polynomial 5.6 is given, making it possible to apply the quadratic formula and derive Si.

$$\Delta T(q, Si) = \frac{E(Si)}{m(q)} \tag{5.3}$$

$$E(Si) = \Delta T m \tag{5.4}$$

$$aSi^2 + bSi + c = \Delta T m \tag{5.5}$$

$$Si^2 + \frac{b}{a}Si + \frac{c - \Delta T m}{a} = 0 \tag{5.6}$$

$$Si(\Delta T, m) = -\frac{b}{2a} \pm \sqrt{\left(\frac{b}{2a}\right)^2 + \frac{\Delta T m - c}{a}} \tag{5.7}$$

After applying the quadratic formula the expression is similar to 5.2 with two different roots. One of these roots is false and needs to be discarded. Since the slope of the E(Si)-curve is increasing (see 3.14, the positive root is the correct one. As a precaution, a test to see whether a is positive or negative is implemented and the root is chosen accordingly. From here on however, the positive root will be used according to equation 5.8.

$$Si(\Delta T, m) = -\frac{b}{2a} + \sqrt{\left(\frac{b}{2a}\right)^2 + \frac{\Delta T m - c}{a}} \tag{5.8}$$

Since the feedforward relies on the water flow during its calculations it is important that the flow sensor functions as intended, which unfortunately is not the case during foaming. The flow goes below the recommended interval stated by the manufacturer causing the error to be at least 20% with no upper limit according to the manufacturers specifications. Combining this uncertainty with the startup transients in figure 5.1 makes it impossible to create a reliable feedforward model using the waterflow. Instead the control signal during stationarity is measured and stored, and later used during the next foaming run as a constant feedforward signal. This is possible due to the flowrestrictor mounted in the ECU which keeps the steadystate waterflow relatively identical during the foaming runs.

5.4 Feedback

After moving the temperature sensor (T_{out}) closer to the steam injector the time delay was decreased, making it possible to use a stronger feedback loop without inducing oscillations (see section 6.2). The main purpose of the feedback loop is to improve the behavior during sudden flow (q) or temperature (T_{in}) changes where the measurements are unreliable or slow, and to remove small offsets during stationarity by using the integration part.

The PI controller was written using structured text and uses some common but important improvements like anti windup and the possibility to reset and/or turn off the integration part at will. Additionally, the proportional part and the integral part is limited to avoid overshoots and undershoots.

The PI controller uses different constants during warm rinse and foaming because of the different length of the time delay. The biggest difference is in the proportional part where the gain is reduced to one third when switching from warm rinse to foaming. The speed of the I part is roughly the same during warm rinse and foaming. Since the I part is used mainly to support the feedforward model by bumpless transfers and removal of small offsets during stationarity it does not need to be fast during warm rinse. Since the feedforward model can not be used during foaming, the I part is more actively controlling the temperature and therefore the integration speed is the same even though the time delay is longer.

5.5 Flow Step Storage

Overview

Compensating for sudden load disturbances is difficult when the control loop contains time delays, the fastest and therefore the most difficult load disturbances to compensate for in the ECU are the steps in the water flow. These steps occur after a zone change and their size and type varies. There exist a slight time delay between the water flow change and the flow sensor reaction, offsetting the feedforward estimation and forcing the temperature control loop to depend on the temperature feedback which contains a time delay. Adding the response time of the steam injector makes the total response time for the control system quite large, which might result in severe temperature spikes or drops if the flow step is large.

The solution implemented to this problem is loosely based on Iterative Learning Control method, using the fact the process runs are repetitive. By storing data in the background about large steps in the water flow, the next time the same cleaning sequence runs it is possible to predict when the step occurs and how large the step is. This makes it possible for the control system to compensate for the step before it occurs. In order to store this data, a subprogram called Flow Step Storage (FSS) is used.

Detection and Classification

Sudden steps and spikes in the water flow occurs around the zone changes and by analyzing figure 5.3 it is apparent that there are three different types of sudden flow changes. The most common flow change is the flow spikes where the water flow is similar before and after the zone change and the flow transients. Additionally, the step up and step down type of flow transients exists where the water flow takes a large step up or down.

To detect these sudden flow fluctuations an IF - THEN construct is used that compares the difference between the current water flow and a slightly delayed version of the same signal. The IF - THEN construct activates a trigger if the difference between the signals exceeds a set value, by changing this value it is possible to set how large the flow step or spike has to be to trigger a detection. By controlling how much the comparison signal is delayed it is possible to set a rate for how fast the water flow is allowed to change before it triggers a detection. The final FSS program uses a minimum of 0.5L/s difference over a 1 second long time period as limit for when a detection occurs. After a detection has occurred, a boolean is used to activate the classification procedure.

Directly after a step is detected the time delayed water flow is stored down in a temp variable, by comparing this value with the steady state flow at the end of the current substep the flow transient is classified. Additionally, a variable is used to store the maximum water flow during a transient. This variable is used to distinguish between large spikes and normal spikes. The green dots in figure 5.6 is the time delayed water flow stored before the transient and the red dots are the flow stored at the end of the substep.

Storage

The prediction data is indexed in a two level storage array using a special data type. By using the step and substep as indices the data is well organized and easy for the prediction program to find. The step refers to which cleaning sequence is running while the substep gives the current cleaning zone in the machine. The data type stores

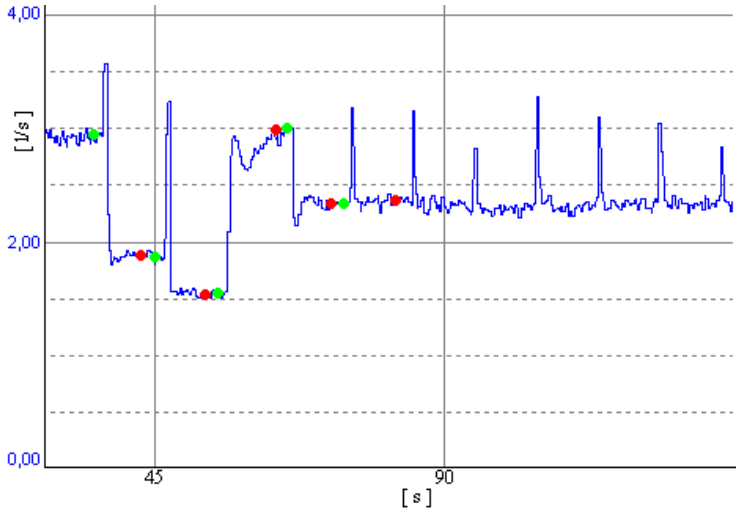

Figure 5.6 Overview of the storage points for the classification.

the classification data and the steady state flow, see the list below of the variable name and data type used to store the information. Step and substep is not stored in the data container, since they are used as indices.

Detection	BOOL	True if a transient is detected
StepUp	BOOL	True if transient is classified as step up
StepDown	BOOL	True if transient is classified as step down
Spike	BOOL	True if transient is classified as a spike
LargeSpike	BOOL	True if transient is classified as a large spike
Flow	REAL	Contains the water flow after the transient

5.6 Flow Prediction

Overview

The prediction program modifies the flow signal sent to the feedforward model by using the transient data stored by the FSS program. Correcting the flow signal makes it possible to improve the behavior of the control system during the transients. The flow prediction (FSP) checks the stored prediction data every substep change, if the detection boolean is true the correction algorithm is started. By checking which of the classification booleans that are true, different correction strategies are used by the FSP program depending on the situation. Increasing the capabilities of the feedforward control loop calms the process down, removing the oscillations that the feedback loop induces if it has to handle to large disturbances alone.

The plots in this section uses the same color coding throughout and the time axis is divided into several zones giving the reader the ability to estimate the performance. Between the vertical dotted lines it is 15 seconds and between the horizontal ones it is 5 degrees.

Spike

The most common transient is the flow spike where the water rushes to fill the empty pipes in the new zone. Since the feedforward model uses the water flow to calculate

the control signal the flow spike causes a spike in the control signal to the steam injector. Which in turn causes a temperature overshoot because of the reaction time of the steam injector and the time delayed flow measurement. Since the feedback loop is relying on the delayed temperature measurement it will increase the overshoot while compensating for the temperature dip caused by the sudden increase in water flow. See figure 5.7 for a example of this behavior. The figure uses dark green for the water temperature out (T_{out}).

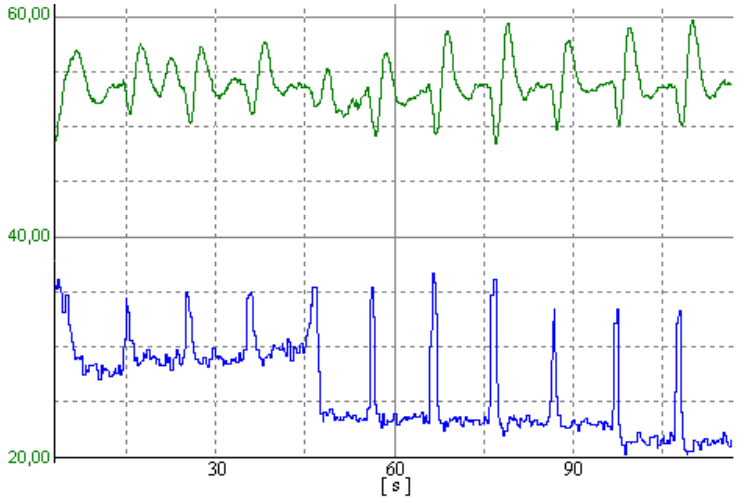

Figure 5.7 Behaviour during flow spikes without FSP.

It is apparent in figure 5.7 that the size of the temperature transient is directly proportional to the height of the water spike, therefore the LargeSpike boolean is used to single out severe water spikes that causes large temperature osciallations that goes outside the ± 5 degrees temperature limit.

To compensate for the smaller spikes the FSP program freezes the water flow measurement just before the spike and releases it after the spike goes down. This removes the overshoot induced by the feedforward model but keeps the undershoot caused by the increased water flow. See figure 5.8 for the process behavior during smaller flow spikes. In the figure orange is used for the predicted flow signal (q_p) and as before blue is used for the measured water flow (q).

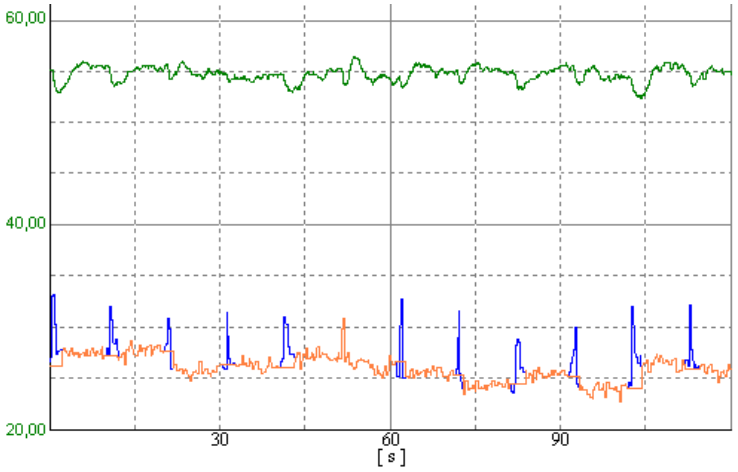

Figure 5.8 Behavior with FSP activated during small flow spikes.

Since the large flow spikes may cause the temperature to fall below the 5 degree limit, the FSP program boosts the flow just before the flow spike in order to raise the control signal and thereby reduce the temperature drop. In figure 5.9 a side by side comparison is available and the effect of the boost is shown. Focus on the water temperature directly after the flow spike for comparison, and pay special attention to how the predicted water flow (q_p) spikes just before the real spike.

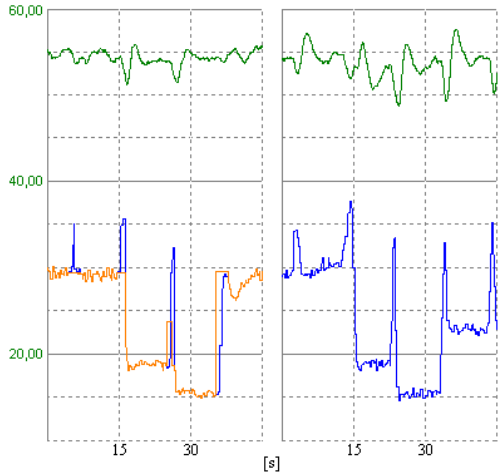

Figure 5.9 Comparison between with and without prediction.

Step Down

When the water flow takes a large step down, the expected behavior is that the water temperature will rise quickly but that was not the case on the P1 prototype. The previously mentioned spikes in the water pressure effectively strangles the steam injection during large flow steps down, see figure 5.4. Compensating the flow signal by making the flow step earlier will cause a large temperature dip when combined with pressure spike and the flow spike just before the step down. Therefore a less aggressive compensation was chosen, where a limit was introduced on the flow signal just before the step down. See figure 5.9 for an example of the limit in use. The limit basically freeze the water flow just before the flow spike and waits for the water flow to fall below the freezing point before it releases the flow measurement. Limiting the flow signal adds extra security encase the pressure spike is smaller than expected, since limiting the flow signal limits the feedforward control signal.

Step Up

When the water flow suddenly increases, the water temperature drops before the flow sensor and the feed forward loop reacts. Just as during the flow spikes, the feedback loop makes the situation worse by inducing a temperature overshoot after the flow step. Another concern during flow steps is that it sometimes is combined with a flow spike, causing the feedforward to overshoot the control signal. This top needs to be cut of not to cause problems. The compensation during step up raises the water flow to the stored value just as the process enters a new substep and holds it there for 2.5 seconds, effectively removing the temperature dip and the temperature overshoot caused by either the feedforward or the feedback loop. See figure 5.10 for overview and comparison of the behavior.

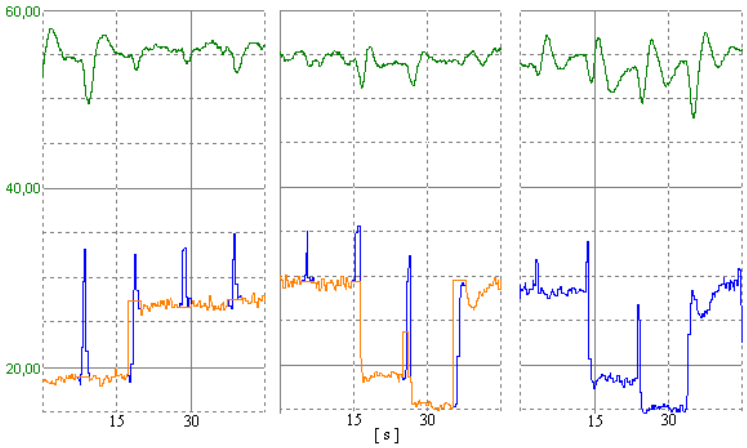

Figure 5.10 Comparsion and overview of the FSP performance.

5.7 Results and Discussion

Color Coding and Units

Between the vertical dotted lines it is 15 seconds and between the horizontal lines it is 5 degrees.

Pink	S_i (%)	Control signal steam injector
Green	T_{out} (°C)	Water temperature out
Red	T_{in} (°C)	Water temperature in
Blue	q (dl/s)	Measured water flow
Orange	q_p (dl/s)	Predicted water flow
Black	W_p (dBar)	Water pressure

Warm rinse

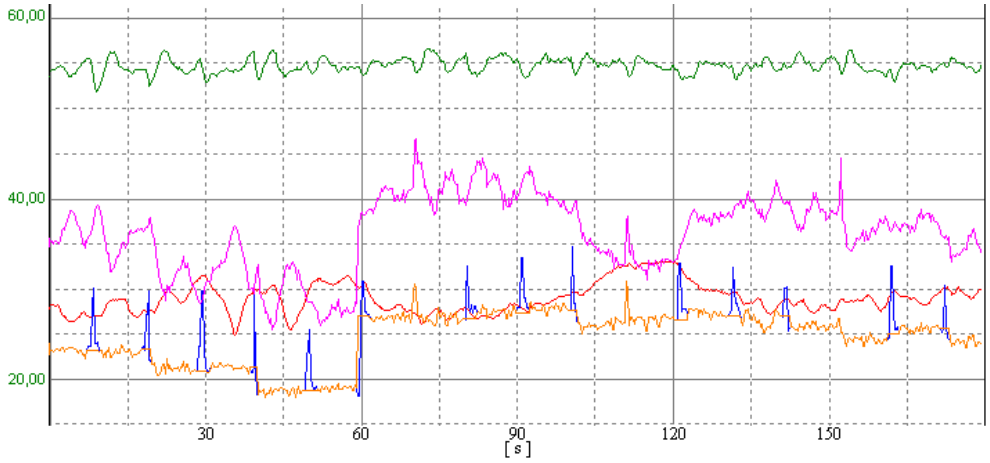

Figure 5.11 A typical warm rinse run.

The combination of the feedforward model and the prediction system works well, keeping the temperature between the $\pm$ 5 degree alarm limit at all times. Since the time limit for how long the temperature may go outside the alarm boundaries is 20

seconds, this control strategy more than fulfills the expectations.

Notice how the temperature fluctuations are kept to a minimum and there are no oscillations or instability even though the large time delay in the feedback loop in figure 5.11.

Foaming

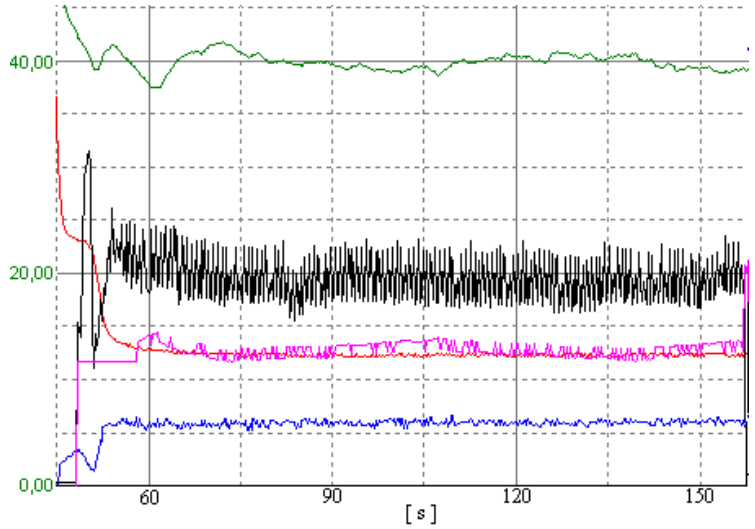

Figure 5.12 Foaming run where the water flow is close to the specified 0.67 L/s.

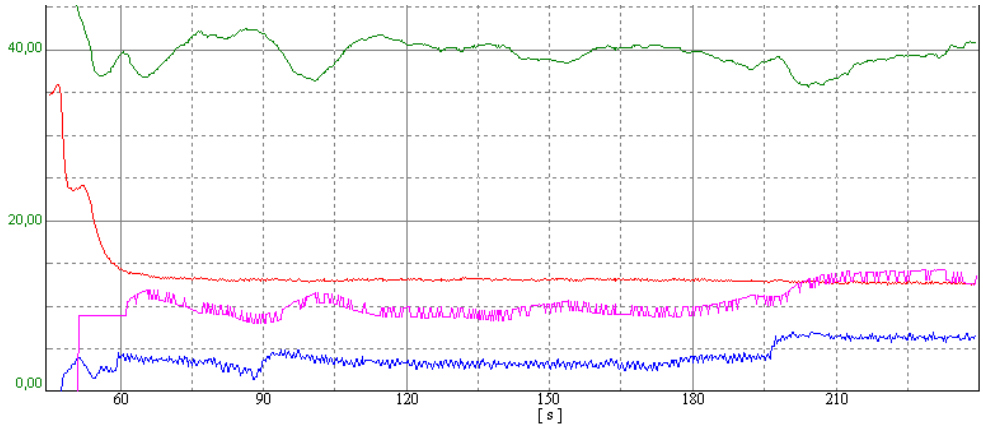

Figure 5.13 Foaming run where the water flow fluctuates below the specification.

The specified water flow during foaming is 0.67 l/s as seen in figure 5.12, where the equipment is working as intended. Then the temperature control works well with no temperature oscillations. The oscillations in the water pressure is caused by the foam injector and causes no problems for the control system.

In figure 5.13 the equipment is malfunctioning, causing the water flow to vary below the specification which makes the temperature fluctuate. It still keeps the temperature in between the alarm limits but it is not as stable as in the nominal case. This is a common problem on the P1 prototype. To improve this behavior it was suggested to clean two zones at the same time, which doubles the water flow and decreases the time delay allowing more aggressive feedback control. Another positive side effect of cleaning several zones at the same time is that the foaming cleaning steps are finished faster.

6. Improvements of the ECU

6.1 Chapter Introduction

The test rig used during the thesis work is under development resulting in that several ECU components was changed or moved because of design flaws detected during the test runs. These changes made the process easier to control and collectively improved the performance of the ECU.

6.2 Temperature Sensor B0232

Introduction

The most important feedback signal for the control system comes from the temperature sensor situated after the steam injector. This measurement signal is delayed since the temperature sensor can not be situated in the center of the steam injector. The placement of the sensor in relation to the steam injector directly controls the time delay. The original design for the ECU had the temperature sensor situated far away from the steam injector which caused a large time delay. By moving the temperature sensor closer to the steam injector the time delay was decreased while still giving accurate temperature measurements.

Placement and Results

Moving the temperature sensor too close too the steam injector would not give the water enough time to mix and thus a hot or cold zone may occur around the temperature sensor for certain flows. When preparing to move the temperature sensor several holes were drilled at different distances from the steam injector making it possible to quickly try where the closest reliable position was for the sensor. After testing different positions it was apparent that the test position closest to the steam injector provided good temperature readings, although being a bit more noisy than the original placement. For comparison between new and old placement and overview of the different test positions see figure 6.1.

Moving the temperature sensor shortened the time delay with roughly a second and made it possible to use a more aggressive feedback loop, which in turn improved the behavior during sudden load disturbances (see figure 6.2).

6.3 Steam Injector

Linear Diffuser

The original steam injector used a linear diffuser and had around one inch of stroke length when moving the piston from close to maximum injection, see figure 6.3. The short stroke length combined with mechanical play resulted in an effective deadzone of two percent. During normal operation the effective opening would be around ten

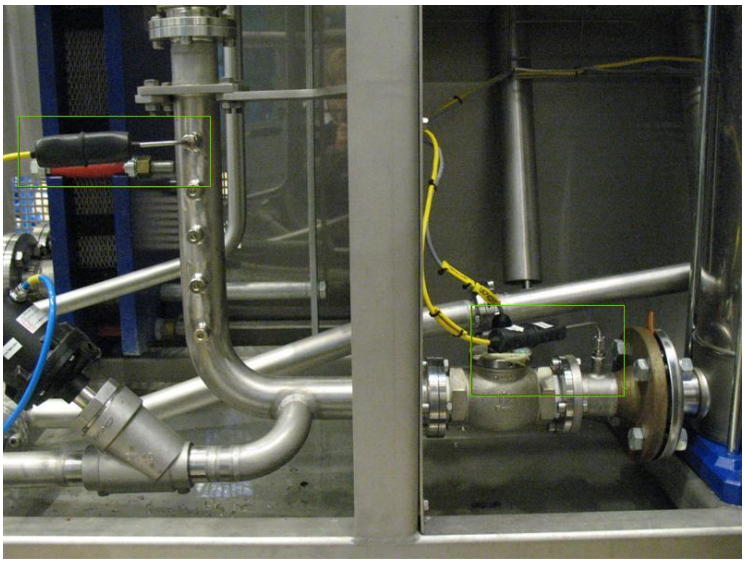

Figure 6.1 Overview of the old and new position.

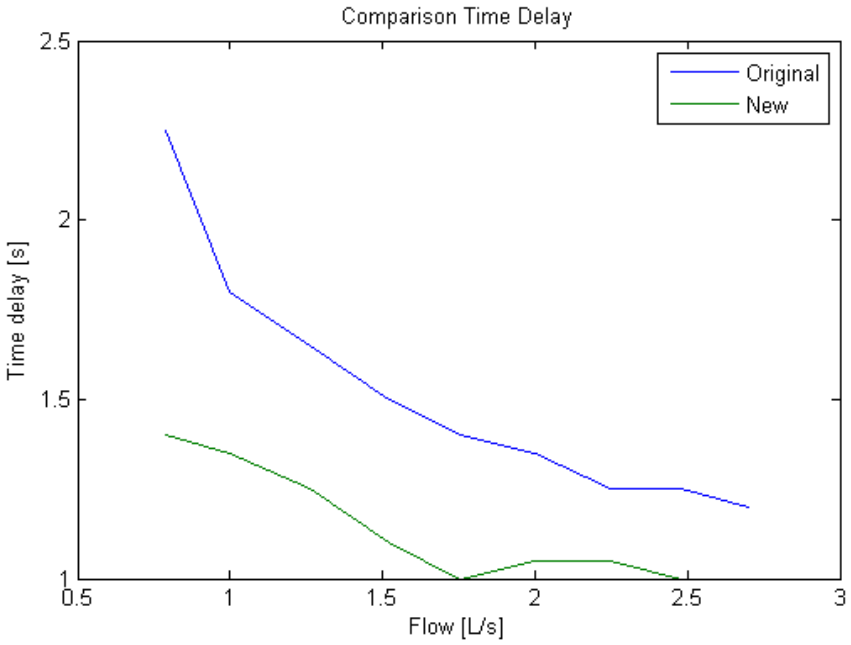

Figure 6.2 Comparison between new and old placement

percent during foaming and twenty percent during warm rinse, leading to the conclusion that the steam injector was too big and lacked the necessary precision to control the process when combined with the deadzone. The built in piston control system also had a tendency to overshoot during sudden reference changes, causing the resulting piston position to vary. This made it impossible to predict the piston position without measuring the temperature and thus made it impossible for a feedforward model to predict the output from the steam injector.

Nonlinear Diffuser

To make the process easier to control, a nonlinear diffuser was installed. By placing the diffuser holes further apart at low openings and having them gradually come

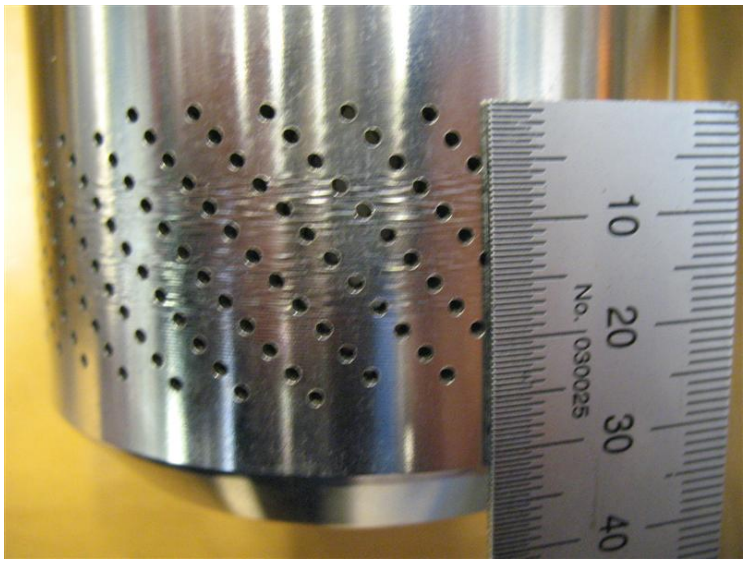

Figure 6.3 Closeup of the diffusion holes on the linear cylinder.

closer as the piston travels up a nonlinear energy injection was accomplished, see figure 6.4. This increased the precision at low openings and decreased the effect of any modeling error while still being able to quickly ramp up the temperature without the support of the heat exchanger. Additionally the stroke length of the piston was increased by fifty percent, effectively decreasing the backlash to roughly one percent while substantially increasing the precision. Changing the diffuser cylinder made the control of the process easier, especially during low water flow where a small error in the the control of the steam injector would result in a large temperature error. See figure 6.5 for a comparison between the two diffusers energy output.

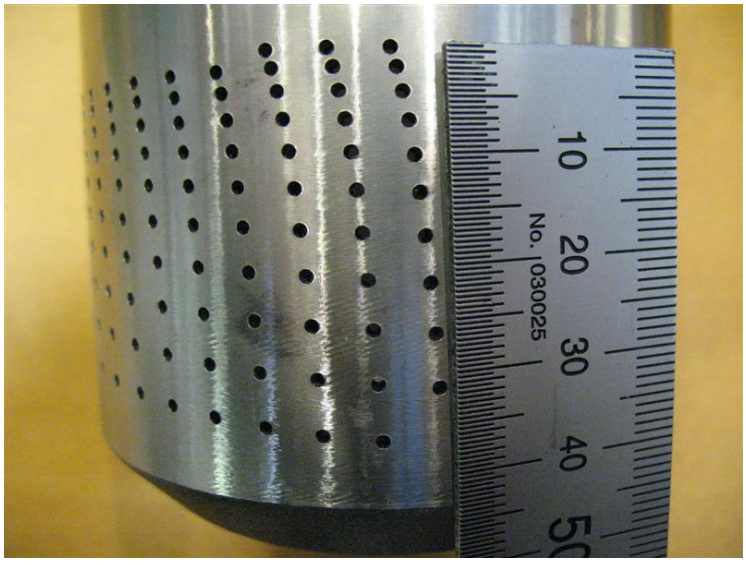

Figure 6.4 Closeup of the diffusion holes on the nonlinear cylinder.

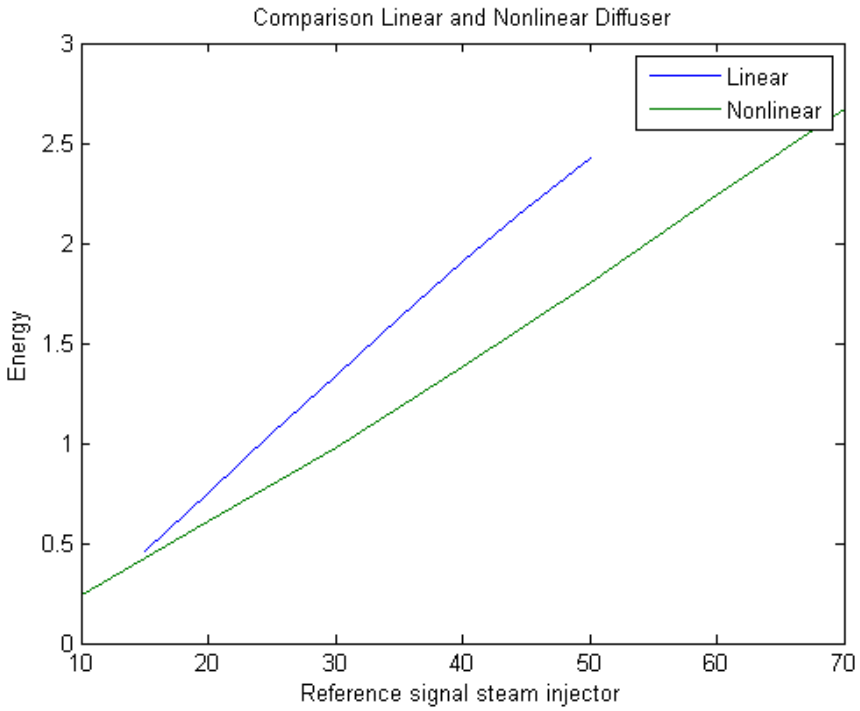

Figure 6.5 Comparison between the diffusers.

6.4 Waterpressure

During a zone change the water pressure spikes or drops if there is a large difference in flow between the new and old zone. The original water pressure control loop was too slow and did not manage to get the water pressure back to the set point before the next zone change. It had a weak proportional part and a slow integration part while running on a 200ms scan cycle instead of 50ms like the rest of the control loops, introducing an unnecessary time delay. See figure 6.6 for an overview of its performance during warm rinse. In the figure black is used for the water pressure [Bar] and blue is used for the water flow [l/s].

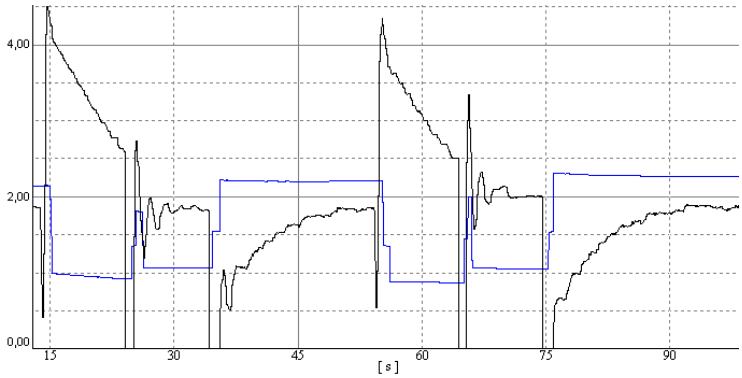

Figure 6.6 Orignal pressure controller performance during warm rinse.

The new pressure regulator uses a 50 ms scan cycle which removes the unnecessary time delay and speeds up the integration part four times, giving the performance visible in figure 6.7.

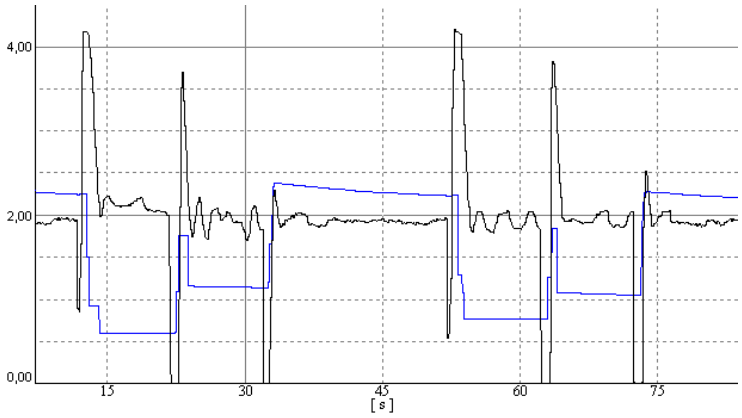

Figure 6.7 New pressure controller performance during warm rinse.

6.5 Filtering of Water Flow Measurement

The flow sensor contains a built in filter which can be set on a scale between 0 and 9, where 0 is no filtering and 9 is maximum filtering. As standard the filter is set to 9 from the factory suggesting that the sensor is not intended to be used in processes with fast dynamics. The flow measurement was behaving like a slow integrator, slowly gliding to the correct value when the flow differences was small. Although during large flow steps it would make sudden jumps to almost the correct flow. During the cleaning runs there was unexplained temperature transients and the feedforward gave the wrong temperature estimation even though it was correct during manual testing. This lead to the conclusion that the flow estimation was wrong during the cleaning runs, and so the manufacturer was contacted. Instructions were given on how to turn off the internal filter which dramatically changed the measurement signal revealing new flow transients during zone changes and speeding up the flow estimation. This removed the gliding behavior, which caused problems for the feedforward, and decreased the feedforward reaction time. See figure 6.8 and compare the flow measurement with the old flow measurement in figure 6.7.

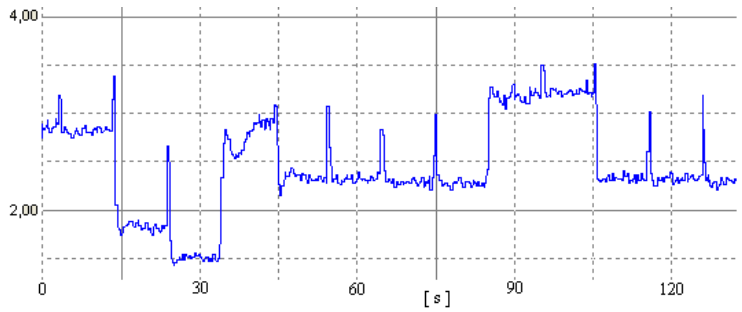

Figure 6.8 Flow measurement with no filtering.

7. Summary

This Master Thesis has been centered around temperature control of an external cleaning unit. The main problems have been long dead times and time delays combined with fast process dynamics, such as rapid flow transients and temperature changes of the incoming water. The long time delays made it impossible to use feedback for the temperature control since the conditions changed too fast in comparison to the reaction time of the delayed feedback loop. Therefore a model based feedforward was created to counter the time delay present in the temperature measurement.

The first step in the creation of the feedforward model was to create a process model in Simulink, which different control topologies later could be tested on. Different kinds of models were tested, both linear and non linear. The model that proved to match the process best was a model derived by physical relations. This model uses non linear relations between temperature, steam injector input, water flow and the steam pressure. The Simulink model served as a starting point in the development of the feedforward model. The final feedforward model uses modified equations from the Simulink model that instantly calculates the correct control signal based upon the current water flow, incoming water temperature, temperature reference and steam pressure.

Not only the conditions that are measurable and used in the model affects the temperature, but conditions such as the steam temperature and manufacturing differences of the components in the ECU affects the temperature as well. In order to compensate for these, an automatic tuning sequence was created. The model then indirectly takes the steam temperature and other such conditions into account since they affect the effectiveness of the steam injector. The calibration of the flow sensor and steam injector are also compensated for in the tuned model, making the feedforward model more accurate.

The feedforward model compensated for the time delayed temperature measurement, and while doing so it exposed two other control problems. The reaction time of the steam injector and the inaccurate flow measurements. The later problem was corrected by turning off the built in filter in the flow sensor which improved the accuracy of the flow measurements and slightly reduced the time delay. Since the changes in the water flow are sudden the reaction time of the steam injector combined with the time delayed flow measurement caused problems for the temperature control. The only way of improving the performance was to use some kind of prediction. Since the largest problems were caused by the sudden flow transients, it was decided to predict their behavior. Since the behavior of the transients was repetitive between the cleaning runs, it was possible to use the sequence data to index the transients.

The repetitiveness of the process and the possibility to use the step and substep data as a sort of time axis made it apparent that a ILC method would work well for prediction of the water flow. By storing data about the flow transients and categorizing them it was possible to predict the flow signal used for feedforward calculations the next time the same step and substep was cleaned. This gave the steam injector more time to move the piston and removed the influence of the time delayed flow measurement, which improved the behavior during flow transients.

Since the feedforward model is dependent on the water flow and the water flow is below the specification of the flow sensor, the feedforward model can not be used during foaming. Instead the steady state control signal from the last foaming run was used as a constant feedforward signal. This was possible because the ECU contains an internal flow restrictor that made the flow similar during the foaming runs. Since

it was not possible to use the flow measurement for the feedforward or improve the feedback loop because of the time delay it was decided by Tetra Pak to improve the process behavior during foaming instead.

The final control system fulfilled the specifications and, more importantly, the expectations of Tetra Pak. The controller will therefore be used in the final product and sent with the A6 machine all over the world.

A. Appendix

A.1 Cleaning Sequence

Overview

The cleaning sequence uses different steps during the cleaning. The steps perform one task each, such as warm rinse, cold rinse, foaming etcetera. Each step contains substeps that cleans one zone each. The steps and substeps are then performed in a predefined order which will clean the machine. Each step uses their own number of substeps and the substeps do not correspond to the same zones in all steps.

Warm Rinse

Warm rinse is step where warm water ($55°C$) is used to rinse the zones. The warm rinse step uses a high flow which is limited by the amount of water each zone can output. This means that the flow vary depending on which zone is active and that flow spikes are induced during zone changes. The reason there are flow spikes are that, when switching zones, two valves are open at the same time. Having two zones open at the same time increases the flow resulting in a spike. Depending on whether the pipe after the newly opened valve are filled with water or not, the size of the spikes varies. If it is not filled with water, the flow spikes while filling the pipe.

Cold Rinse

Cold rinse is a similar step to warm rinse with the difference that cold water is used instead of warm water. This means that no temperature control is used during cold rinse.

Foaming

In contrast to cold and warm rinse, foaming is not limited by how much water each zone can output. Instead a flow limiter inside the ECU controls the water flow. The flow during foaming is thereby kept substantially lower than during warm rinse. Since the flow is limited by the flow limiter, there will not be any flow spikes and the flow remains almost constant. The reference temperature is $40°C$ during foaming .

Soaking

In order for the foam to be effective, a soaking step is used. The step is basically a waiting step where the foam is given time to dissolve milk residue. Soaking follows after a foaming step and is often followed by a warm rinse step.

Draining

After a warm rinse step has been performed, the pipes are filled with warm water. In the draining step the pipes are emptied of this water in order for the foaming water temperature not to be affected by the higher warm rinse temperature.

Cool Down

Cool down is a waiting step that follows after a draining step. It basically waits for the pipes to cool down in order for the foaming step to start properly.

A.2 Process Actuators and Sensors

Steam Injector

The steam injector controls the fluid temperature by injecting steam directly into the water. By using a diffusion cylinder with a set of holes and a piston to control how many of the holes that are exposed to the steam pressure, the amount of steam injected is controlled (see figure A.1). The piston is connected to a membrane via a piston rod, this membrane is controlled by high pressure air and pulls the piston rod up and down. By moving the piston the amount of diffusion holes exposed is controlled. The diffusion cylinder is submerged in the water while containing high pressured steam on the inside. When the piston moves to expose the diffusion holes, the steam sprays out of the diffusion cylinder into the surrounding water at high velocity. The high velocity of the injected steam causes large amounts of turbulence in the water flow which effectively mixes the steam and water, making the temperature even in the entire water flow.

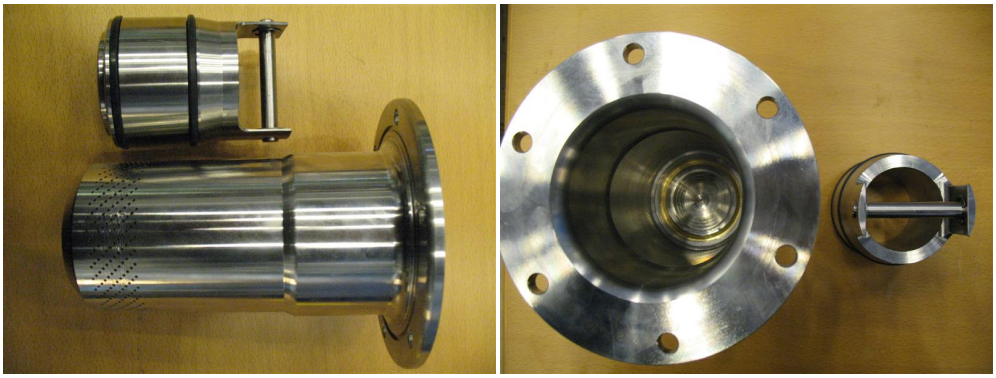

Figure A.1 The diffusion cylinder and piston

Temperature Sensor

The sensors used to measure the water temperature are called thermocouples and measures the temperature using the Seebeck effect. When a temperature difference occurs across a metal an electric potential difference is induced between the warm and the cold end. The Seebeck coefficient gives the relationship between the potential difference and the temperature difference, expressed in volts per kelvin. By using two metals with different Seebeck coefficients in a loop, a potential difference occurs between them inducing a small current even though they are exposed to the same temperature difference. By measuring this potential difference the temperature can be estimated. Since the sensor measures its own temperature it needs to be in direct contact with the water to give a estimate of the water temperature, see figure A.2 showing the sensor probe that is inserted into the water flow.

Flow sensor

Measurement of the water flow through the ECU is made using a single flow sensor situated close to the tap water inlet to the ECU. The flow meter uses a paddle wheel that rotates as water moves through the sensor. The housing used to direct the water flow past the paddle wheel comes with a predefined coefficient used to convert the rotations per second of the paddle wheel into liters per second. The counter combined with the housing makes up the flow sensor unit, see figure A.3.

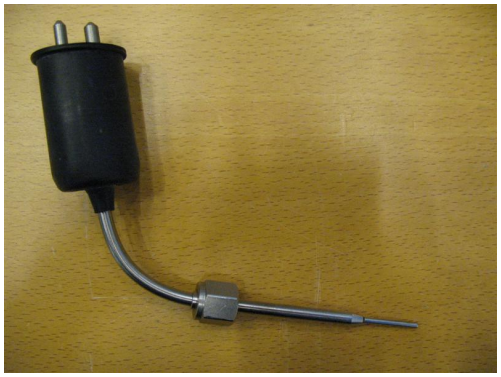

Figure A.2 Overview of the temperature sensor

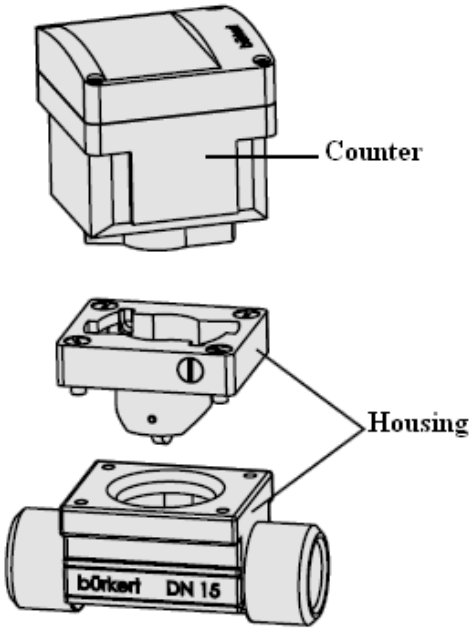

Figure A.3 The flow counter and paddle housing.

B. Adaptive Model

B.1 Chapter Overview

The final product used a feedforward control strategy which is covered in the thesis. As an alternative a control loop based on state estimation was also developed. It did not perform as well as the feedforward solution but did spawn some interesting solutions and ideas. This chapter covers the development of an adaptive model intended to be used as a state estimator and can be seen as a continuation of chapter 3, covering the development of the Simulink model.

Even though the ECU units are made to the same specifications the different sensors and actuators vary individually resulting in that a non adaptive model will not match the different processes. If aging is taken into consideration, an adaptive model becomes even more important because of the wear and tear on the moving parts and the diminishing efficiency this results in.

B.2 RLS Estimated Model

Overview

RLS is a efficient way of estimating model parameters online and is a natural part of all advanced control courses focusing on adaptive models and predictive control. RLS is also commonly used in signal processing. Recursive least squares uses a quadratic error criteria that it tries to minimize recursively online with the goal of reaching the same solution as the offline least squares method.

Theory

The goal of both RLS and LS is to minimize the squared error, the minimization criterion is formulated in equation B.1 for LS and equation B.2 for RLS.

$$J = e^2 \tag{B.1}$$

$$J = \sum_{n=0}^{k} e(n)^2 \tag{B.2}$$

Without going into too much detail the update equations for the RLS is stated below.

$$\hat{\theta}_k = \hat{\theta}_{k-1} + P_k \phi_k \varepsilon_k \tag{B.3}$$

$$\varepsilon_k = y_k - \phi_k^T \hat{\theta}_{k-1} \tag{B.4}$$

$$P_k = P_{k-1} - \frac{P_{k-1}\phi_k \phi_k^T P_{k-1}}{1 + \phi_k^T P_{k-1} \phi k} \tag{B.5}$$

The core of the method is equation B.4 were the error ε_k is calculated by deducting the model output from the measurement of the output from the real process. The model output consists of the parameter estimate $\hat{\theta}$ and the regression vector $\hat{\phi}$, which contains the input signals. These are multiplied together which results in an estimate

of the output from the process. The hardest part to penetrate of the RLS method is the update equation of the, commonly called, covariance matrix P_k. The interested reader can find more information in System Modeling & Identification by Dr. Johansson.

Implementation

When this was implemented the water model consisted of the following polynomial.

$$\Delta T = (\frac{a}{q} + b)\frac{S_i}{23} \tag{B.6}$$

This can be rewritten to equation B.7

$$\Delta T = c\frac{S_i}{q} + dS_i \tag{B.7}$$

It was decided to estimate the c and d constants of the B.7 equation. The matching regression vector used was $[\frac{Si}{q} Si]$. The test of the method was performed in Simulink, using a premade RLS block from the Control Department at LTH.

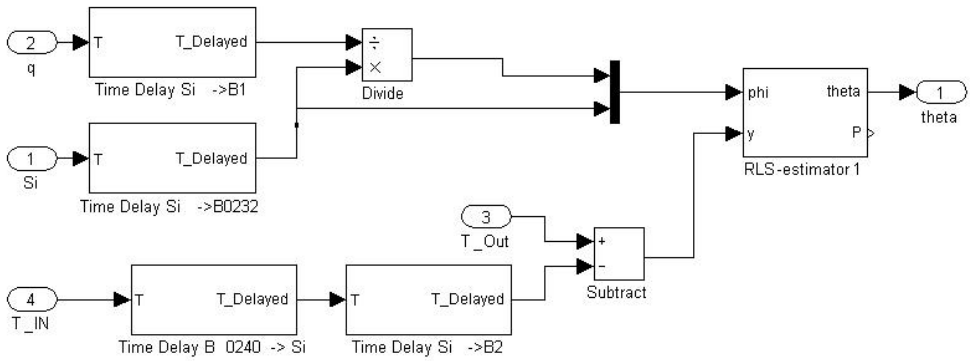

Figure B.1 Overview of the Simulink RLS estimation block

Because of the time delays between the temperature sensors and the steam injector it was important to delay the signals accordingly so that all the measurements and control signals were performed on the same volume of water.

Results

During the dry run in figure B.2 a forgetting factor value of $\lambda = 0.999$ was used.

The RLS works well during normal operation of the process but when something unexpected or a malfunction occurs then the RLS might collapse. It would therefore be important to include comprehensive safety functions around the online estimation if it were to be used live on the process. Performance wise a solution like this is costly when run on the process because of all the calculations involved and it is hard to implement because of the lack of support for basic calculations using arrays in the PLC software. Another problem is that the RLS does not care if the solution is physically correct with reality and therefore can switch the sign of the constants when not properly excited by the in signal, this would result in a catastrophe if something sudden were to happen in the process since the control system would compensate in the wrong direction. Since there frequently is sudden changes in the process this solution as it stands is not recommended.

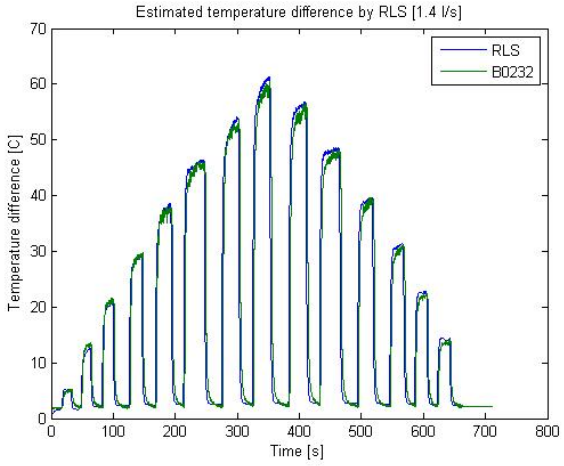

Figure B.2 Dry run using data from the test rig

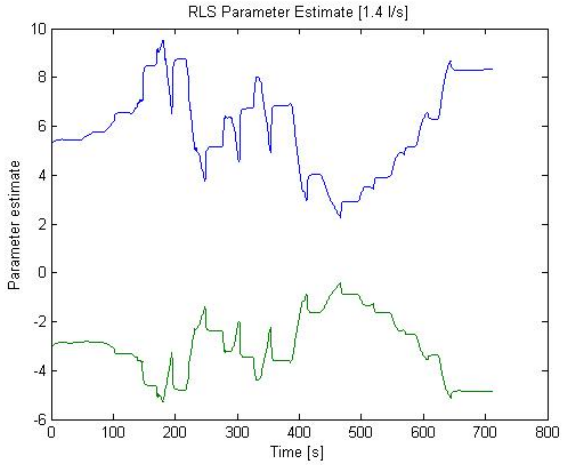

Figure B.3 Parameter estimation during dry run on data from test rig

B.3 Compensated Model

Overview

Using the experience from the RLS estimated model it was decided not to estimate the model parameters online in the water model, instead it was decided to scale them to match the measured output.

$$\Delta T = (\frac{a}{q^2} + \frac{b}{q} + c)Si\frac{k_c}{23} \tag{B.8}$$

When developing this method the main goals were system stability and robustness. Additionally, a big priority lay on minimizing the computational stress on the PLC system.

Theory

The water model polynomial, equation B.8, uses the water flow and the effective steam injection to calculate the temperature difference between the in and out water of

the steam injector. Somewhat simplified the model can be approximated as equation B.9, where the flow part of the polynomial is a measurement of the heated mass while the effective steam injection is a measurement of the injected energy.

$$\Delta T = \frac{Ek_c}{m} \tag{B.9}$$

Scaling the polynomial is equivalent of changing the heated mass or changing the amount of energy injected. This would mean that either the polynomial is wrong or that the calibration of the flow meter or the steam injector is off. There is also the possibility of scaling the model temperature directly.

$$T_{out} = T_{model}k_c \tag{B.10}$$

Adding the scaling directly on the out temperature means that the temperature of the in water to the steam injector is also scaled. When the heat exchanger is used, this temperature is roughly two thirds of the out temperature during steady state in warm rinse. Scaling the out temperature is unproportional since the error occurs in the water model which stands for a smaller portion of the temperature difference in comparison to the heat exchanger. It should also be added that the temperature after the heat exchanger is directly measurable and it would make no sense at all to scale a direct measurement as long as the sensor is performing correctly.

Calculating the compensation factor

The water model calculates the temperature difference between the in and out water of the steam injector, therefore it was natural to look at the quota between the real temperature difference and the models, see equation B.11.

$$Q_{\Delta T} = \frac{\Delta T}{\Delta T_{model}} \tag{B.11}$$

Another biased method of comparison would be to take the quota between the process and model out temperature, see equation B.12 .

$$Q_T = \frac{T}{T_m} \tag{B.12}$$

Using the Q_T quota would result in a biased calculation of the correction factor since the effect of the steam injector is not separated from the effect of the heat exchanger. When calculating the $Q_{\Delta T}$ quota it is important to take the different time delays into consideration, since the in temperature may change rapidly when using the heat exchanger.

$$K_{c(t)} = \frac{T_{out}(t) - T_{in}(t - \Delta 1 - \Delta 2)}{T_m(t - \Delta 2) - T_{in}(t - \Delta 1 - \Delta 2)} \tag{B.13}$$

All the measured signals are delayed to match the water passing through the water temperature sensor B0232 after the steam injector. In equation B.13 the delay $\Delta 1$ is the travel time from sensor B0234 to the steam injector and $\Delta 2$ is the travel time from the steam injector to the B0232 temperature sensor.

The purpose of the scaling factor is to slowly compensate for the model error without disturbing the model dynamics. Therefore the scaling factor was passed through a low pass filter to ensure that the old measurement dynamics would not disturb the new model estimations of the process dynamics. Additionally the low

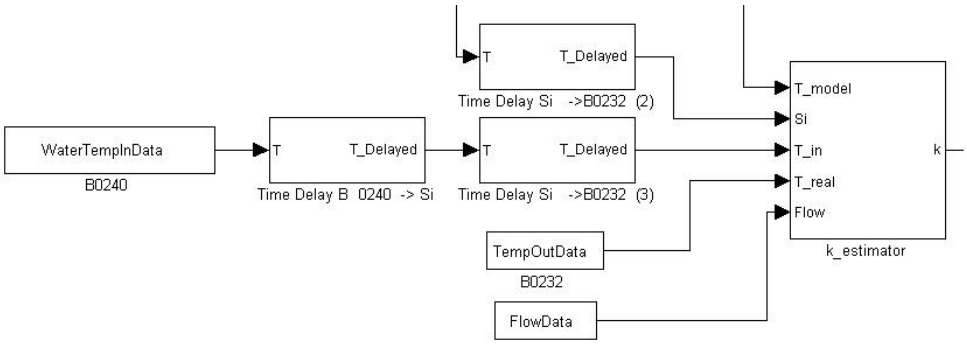

Figure B.4 Overview of the different timedelays

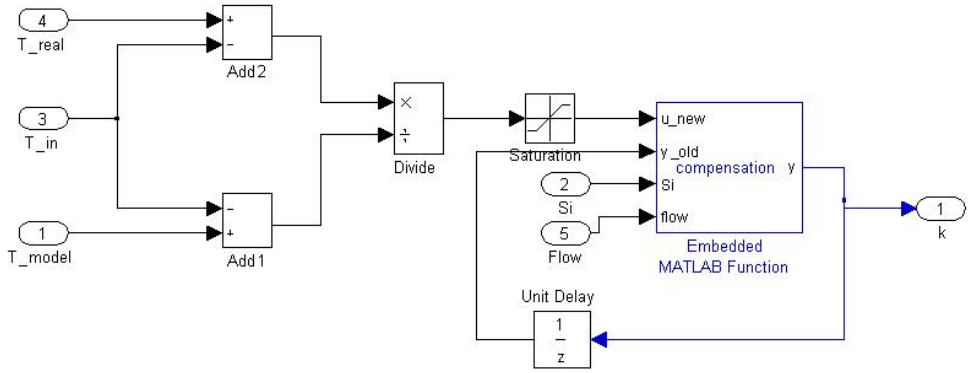

Figure B.5 Inside the k_c estimation block in Simulink

pass filtering makes the estimate much more precise since the filter introduces averaging over more than 100 samples and therefore efficiently removes process noise. The low pass filter was implemented using equation B.14.

$$y_{k+1} = (1-\lambda)u_{k+1} + \lambda y_k \tag{B.14}$$

By tuning the low pass filter it is possible to change the adaptation speed of the correction factor. This is useful if the process works over a larger flow interval and therefore needs faster adaptation to compensate for the larger jumps in the working flow. That is the case when the machine works in warm rinse mode, but when switching over to foaming the flow is more stable and the process itself contains less dynamics and therefore a slower correction of the scaling factor is preferred. When testing the solution it became apparent that the most problems occurred during large steps in the steam injection. Since the time delays and the time constant of the steam injector is not perfect, there is a large difference between the delayed model temperature and the measured temperature during the temperature step. To ensure smoothness of the estimation there was a compensation implemented for this by changing the low pass filter constant during the step responses and by limiting the in signal to the low pass filter. A saturation block from the standard Simulink toolbox was used to limit the quota between 0.75 and 1.25. Additionally the low pass filter equations were modified, see below.

IF $flow > 0.15$ AND $Si > 7$ AND $u_{new}/y_{old} < 1.1$ AND $u_{new}/y_{old} > 1/1.1$
$y = 0.005u_{new} + 0.995y_{old}$

ELSE
$y = 0.001u_{new} + 0.999y_{old}$
END

The modification introduces a speed window for how fast the temperature difference may change without trigging the step detection. When that window is exceeded, the update equations trusts the new data less than during slow changes in the temperature difference quota. This removes unwanted spikes in the estimation during step responses.

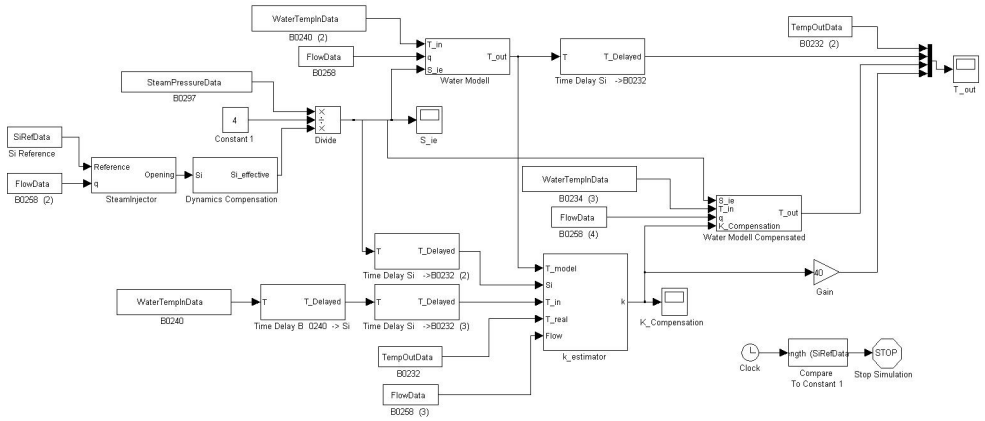

Figure B.6 Overview of the entire Simulink program

The compensation is implemented in Simulink with the possibility of directly comparing it to the regular model, see figure B.6. The same functionality was possible with the function block diagram implementation on the PLC system. Since FBD and Simulink are very similar in usage the program structure is similar although the FBD program being enormous because of the lack of subsystems and the scaling of the block icons.

Results

Figure B.7 shows how the compensation gets increasingly more accurate the longer the process runs. Being able to calculate the steady state temperature directly at the reference change. The model that was compensated is the one for the small test rig, the dry run data came from the complete P1 impact machine. From the data it was concluded that the steam injector on the P1 machine is roughly 20% more efficient compared to the steam injector on the test rig. The time constant relating to the step response time is also different compared to test rig, resulting in an offset between the model and the measured temperature during the step responses. This offset was well coped with by the modified low pass filter.

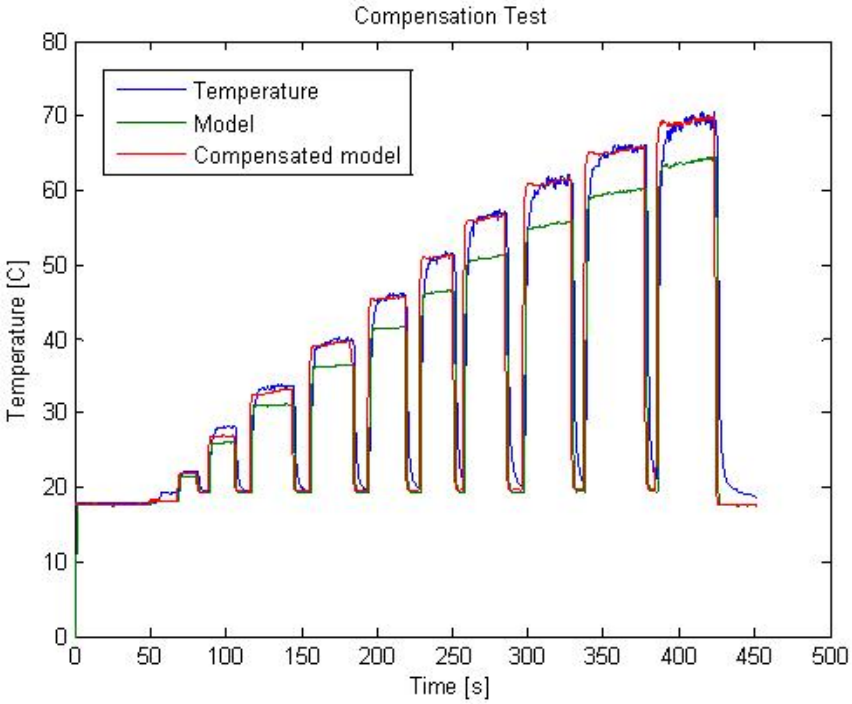

Figure B.7 Simulink test run on data from real Impact machine