CHALMERS

Automatic Control of a Roller Mill using Simulations and Experiments on a Real Machine

Master of Science Thesis

JOHAN KARLSSON

Department of Signals and Systems
Division of Automatic Control, Automation and Mechatronics
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden, 2008
Report No. EX055/2008

Automatic Control of a Roller Mill using Simulations and Experiments on a Real Machine

Master of Science Thesis

JOHAN KARLSSON

Supervisor: Remo Heusi

Examiner: Jonas Sjöberg

Department of Signals and Systems
Division of Automatic Control, Automation and Mechatronics
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden, 2008
Report No. EX055/2008

Abstract

In today's modern wheat mills, the roller mill is the central machine used to create flour. A mill consists of many roller mills and they are operated manually by skilled head millers. By adjusting the distance between the rollers, the wanted outgoing particle size of the wheat from the roller mill can be achieved. In the milling industry, there is a strong will to automate the process of tuning the roller mills. This would in theory give wheat mills with increased throughput of flour and also decrease the cost of producing flour.

In this master thesis which has been done in cooperation with Bühler Group AG, the possibility to apply automatic control to a roller mill has been investigated. Controller algorithms where found in literature, and where implemented both in simulations and in experiments on a real roller mill.

To be able to make computer simulations of the controller algorithms, a model of the behavior of the roller mill was derived. The model aimed at being relatively simple, but to still capture the main characteristics of the roller mill. Using MATLAB, a simulation environment was created. The simulation environment made it possible to test and verify the controller algorithms, before implementing them on the real roller mill machine.

Among the controller algorithms that where tested, the incremental I-controller performed well. The I-controller contains few tuning parameters and its simpleness makes it an attractive choice. However, since the condition of the wheat might change during operation (e.g. wheat type or moisture content), the I-controller might require re-tuning of the controller parameters.

The adaptive feed-forward plus feedback controller can adapt itself to these changing conditions. The parameter adaptation is done by sending the inputs and outputs from the process, to a least mean squares estimation algorithm. This controller requires no re-tuning during operation and is therefore better suited for an automated milling process.

In the future it is desired to, using various new sensors, measure other qualities of the wheat and to use the outputs for automatic control. Multiple-Input Multiple-output controllers have therefore been tested on a simulation model. Here the MIMO optimal controller was found to be the best choice. This controller estimates a model of the system, and performs an optimization on this model to find the best control signals in each simulation step.

KEYWORDS: automatic control, PID control, adaptive control, run-to-run control, optimal control, least mean squares estimation, roller mill, wheat milling

Acknowledgments

I did my master thesis at Bühler Group AG in Uzwil, Switzerland, during the first half of 2008. Thank you everyone at Bühler, for making my stay in Uzwil very enjoyable and memorable. Especially, I would like to thank my supervisor at Bühler, Remo Heusi, for all the help and for always having time to answer my questions. Thank you also to both Mukul Agarwal at Bühler and Jonas Sjöberg, my supervisor at Chalmers, for making it possible for me to write my thesis in Switzerland.

Finally I would like to thank all my friends and family, who have supported me during the course of time. You know who you are, take the credits you deserve!

Varberg, July 22, 2008

Johan Karlsson

Contents

Notation

Abbreviations	Description
PSD	Particle Size Distribution
SISO	Single-Input, Single-Output
MIMO	Multiple-Input, Multiple-Output
PID	Proportional Integral Derivative
LMS	Least Mean Squares
FF	Feed-Forward
FB	Feedback

Chapter 1

Introduction

This chapter contains a background to introduce the reader to the field of milling, and also gives an introduction and motivation for this project. After that the purpose and delimitations of the report are presented. Finally an outline of the different chapters of the report are shown.

1.1 Background

The process of grinding wheat to make flour is something that we have been doing for many hundreds of years. This process has over the years become more and more advanced and nowadays flour is produced in very big mills, capable of producing thousands of tons of flour per day.

The process of creating flour starts by cleaning and filtering the wheat to sort out unwanted particles such as stones, corn and other objects. The next step is to add water to the wheat to give it the correct moisture level, which is crucial to get the best possible milling properties. After that the wheat is ready to be grinded. The grinding usually takes place in *roller mills*, such as in Figure 1.1. These are machines with two or more spinning rollers through which the wheat is let through. The roller mill is the focus of this project, and a more detailed description of the roller mill can be found in Section 2.1.

Figure 1.1: *A typical roller mill that is used in today's mills.*

After the grinding process, the product is sieved in big sieving machines which make it possible to separate the wheat into different fractions. The different fractions are then brought to different places in the mill. Some of it is already flour but some is brought to another roller mill, which further grinds the product. The goal is to get as much flour as possible from the wheat kernels that go into the system.

Although the process today is very refined and advanced with high degree of automation, the tuning of the roller mill machines is still done manually. This is normally done by the head miller, who is a very skilled and experienced person in the field of milling. The head miller can by feeling the product in the hands, make a decision on how to adjust the gap between the rollers, from here on *roller gap*, to get the desired average particle size of the product. Another parameter that can be tuned to change the particle size of the wheat, is the *differential speed*. The differential speed is defined as the ratio between the faster and the slower roller. This parameter is however not used by the head millers at the moment.

In order to more exactly determine the particle size distribution of the wheat, samples of the grinded product can be taken and analyzed using a laboratory sieve unit. This is explained more in detail in Section 2.2.

The manual control of the roller mill has several disadvantages. The head miller needs to be very skilled and experienced, which means that the wheat mills productivity is limited to how good the head miller is. Also, the tuning of the roller mills is very time consuming since there are many machines that needs to be tuned. Finally it requires 24 h/day supervision of the mill.

Automatic control of the roller mill aims at overcoming the problems stated above and means that the *particle size distribution* of the wheat is continuously measured by a sensor. The parameters of the roller mill are then adjusted by an automatic controller, in order to reach a *set-point* specified by the head miller. A simplified diagram of the process is shown in Figure 1.2.

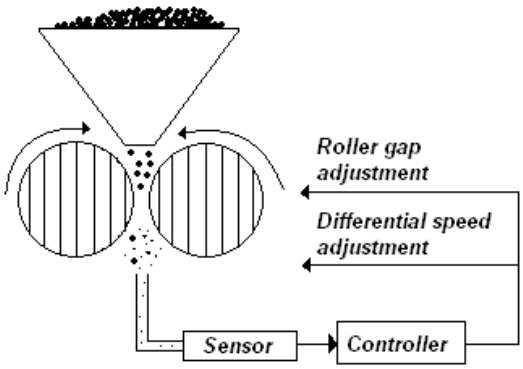

Figure 1.2: *An illustration showing how the roller mill and the particle size sensor are connected. The particle size distribution of the grinded wheat is measured by the sensor. The controller calculates how the roller gap and differential speed should be adjusted, in order to achieve wanted particle size distribution.These steps are then repeated to form a closed loop controller.*

Automatic control of all the roller mills in a wheat mill, would ideally increase the productivity of the entire mill. At the same time, it would reduce the amount of time that the head miller has to spend on tuning the roller mills.

More about milling can be found in (NAM 2008), from where the facts used in this introduction where taken.

1.2 Project Objectives

This project concerns a first attempt towards achieving automatic control of a roller mill. The focus is to find suitable controller algorithms for implementation on a real roller mill experiment machine.

The objectives are summarized into the points shown below:

- Make a model of the roller mill and a computer simulation interface.
- Investigate different control algorithms found in control literature and test them on the computer simulation interface.
- Review the simulated controller algorithms and implement suitable candidates on a real roller mill.
- Give suggestion on which one of the investigated algorithms that is suitable for use in future roller mills.

1.3 Delimitations

For this project, the following delimitations are made:

- The influence the controller algorithms has on the quality of the flour, e.g. ash content, moisture content, starch content etc., is not taken into account. This is because no sensor is available to measure these parameters.
- The experiments on the real roller mill machine are limited to a so called *break passage*. A break passage means that whole wheat kernels enter the roller mill and that the rollers are corrugated.
- The project is a feasibility study, meaning that it is only going to result in an overview of what controller algorithms that are suitable to use.
- The *MIMO controller* (Multiple-Input Multiple-Output) is limited to not be bigger than two inputs and two outputs and is only investigated in simulations.

1.4 Report Outline

The outline of the report is as follows:

Chapter 2 gives a more detailed problem analysis of the roller mill and the particle size sensor. The inputs and outputs of the process are defined and the controller goals are stated.

Chapter 3 introduces the subject of control theory and the different controller algorithms used. Section 3.1 can be skipped if the reader is familiar enough with the subject of control theory.

Chapter 4 gives a description of the roller mill model used for the computer simulations. Important contribution are the experiments done on the roller mill. These experiments shows that the differential speed does not influence the measured particle size distribution, significantly enough to be used as control parameter. They also show that the relation between the roller gap and the outgoing particle size distribution can be approximated as linear.

Chapter 5 presents the controller algorithms used in the SISO simulations. Plots which demonstrates the different algorithms are shown and a comparison of the different controller algorithms is made.

Chapter 6 presents the controller algorithms used in the MIMO simulations.

Chapter 7 shows some of the controllers from chapter 5, implemented on a real roller mill experiment machine. Here it is shown, that it is indeed possible to apply automatic control to the roller mill.

Chapter 8 contains a discussion of the results achieved and the conclusions drawn from this project. Also a proposal for future work is made.

Chapter 2

Problem Analysis

In this chapter, a problem analysis is done in order to scale down the problem and to be able to find suitable controller algorithms. The roller mill and the particle size sensor are presented in more detail. Finally the controller goals are presented.

2.1 The Roller Mill Experiment Machine

The roller mill used for the experiments in this project is shown in Figure 2.1. The machine consists of two parallel rollers, rotating with opposite rotating direction. Usually one roller is rotating with higher speed in order to create shearing forces which cause the wheat kernels to break. The rollers are driven by electric motors and their speeds can by adjusted by a variable frequency converter. The roller gap is adjusted with a separate *PID controller*, which maintains the gap specified by the controller script.

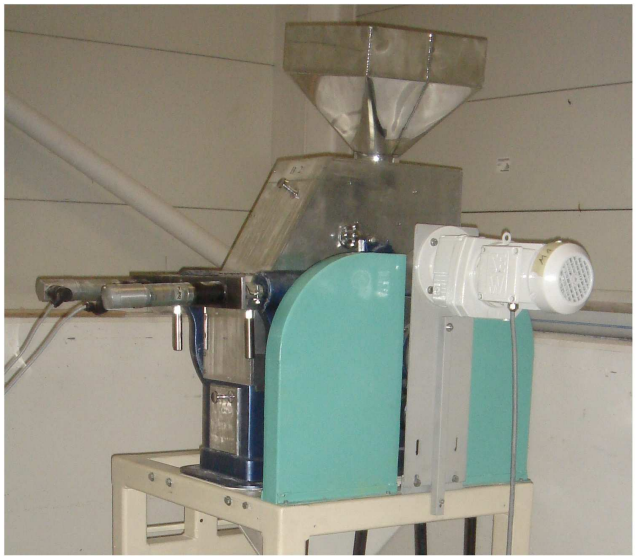

Figure 2.1: *The roller mill experimental machine.*

Located above the roller pair, is a feeding mechanism and a bin with wheat. By sending a control signal, the feeding mechanism will start to feed product with an

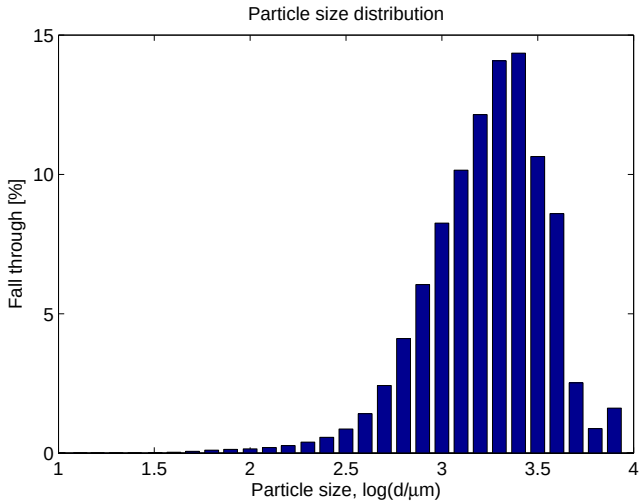

Figure 2.2: *Typical particle size distribution of wheat grinded with corrugated rollers. Note that the scale on the x-axis is logarithmic, e.g. 1 = 10 µm, 2 = 100 µm etc.*

even flow to the roller pair. The grinded product is then, with the help of an air stream, led through a pipe where a sensor is located. This sensor measures the particle size distribution of the wheat and gives an output signal to the controller mechanism.

The roller mill experiment machine is a prototype machine, with smaller dimensions than the existing roller mills that are used in today's mills. The roller-length is 25 cm, instead of the normal of between 0.8 and 1.5 m . This means that less wheat is needed to maintain the mass flow through the rollers. The difference is very clear if the roller mill from Figure 1.1 and Figure 2.1 are compared.

2.2 Measuring the Particle Size of the Wheat

In order to be able to apply automatic control to the roller grinder process, the particle size distribution of the outgoing wheat particles has to be measured. How this is done is explained in this section.

2.2.1 Definition of Particle Size Distribution

After the wheat particles have been grinded by the roller mill, the particles vary in size spreading from very small to very big particles. This is described with a particle size distribution. Nowadays in the milling industry, the most used technique for manually measuring the particle size distribution of the wheat, is to sieve the product. The goal of the process is to divide the wheat into different bins, depending on how big the wheat particles are. To be able to know how many percentages of the flour that are in respective bin, the grinded product is put in a sieving machine. This machine contains several compartments with sieves with varying mesh sizes. The machine is shaking the product for approx. 10 min. This process divides the wheat into the different compartments depending on the size of the wheat. The last step is to weigh the content of each compartment and calculate how many % of the

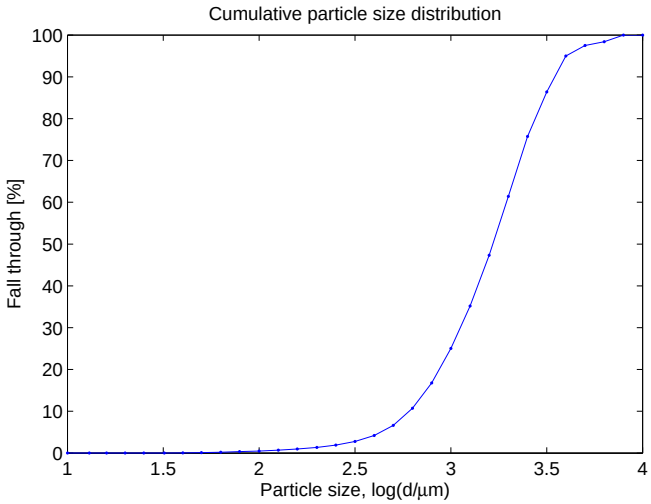

Figure 2.3: *A cumulative particle size distribution curve of wheat grinded with corrugated rollers. Note that the scale on the x-axis is logarithmic, e.g. 1 = 10 µm, 2 = 100 µm etc.*

product which are in the different compartments. The result from such a particle size distribution analysis is shown in Figure 2.2.

The cumulative distribution curve is shown in Figure 2.3. This curve is the key to how the process outputs are defined, which is shown in Section 2.3.1.

2.2.2 Sensor Description

An online sensor has been used to measure insitu particle size distributions of wheat products after the roller mill, as shown in Figure 1.2, in Chapter 1. Basic sensor principles are explained in this section, and the sensor outputs are summarized in Section 2.3.1.

The sensor uses Spatial Filter Velocimetry and a fiber-optic measurement principle to measure the diameter of each particle that passes through the sensor. The particle size distribution is then obtained by assuming spherical particles of constant density.

The sensor continuously measures the length of the particles, and repeats the procedure above until a certain amount of particles, N, which is set by the user, is reached. N is usually set in the region of $40000 < N < 130000$ particles. The sensor then calculates the mean value of the N latest measured particles and sends an output signal which can be used for automatic control of the roller mill.

It should be noted that increasing N, reduces the measurement noise of the sensor, but also decreases the time constant (speed) of the sensor, i.e. it will take longer time for the sensor to show the correct output. There is clearly a trade of between accuracy and measurement time.

What complicates things is that the particle-rate can vary depending on the type of wheat. For example, when the wheat is dry, the sensor can measure approx. 1500 particles per second. But with a wheat with higher moisture content, this number may decrease to approx. 1200 particles per second.

Also worth noticing is that the sensor will not give exactly the same result as a manual sieving (explained in Section 2.2.1) of the product would give. This is a result of the fact that they are two completely different measurement techniques. In this project however, sieving is ignored as this is not an online measurement technique.

2.2.3 Sampling Time

The sampling time, T_s has been specified to not be lower than 10 s. The high sampling time is in reality not a problem, because of the slow time constant of the system.

2.3 Definition of the Inputs and Outputs

In this section, the inputs and outputs of the process are defined.

2.3.1 The Sensor Outputs

The process outputs are values on the cumulative distribution curve at a certain particle size. The outputs can also be interpreted as how many percentages of the total wheat, that falls through a certain sieve size. The outputs will therefore from here on be referred to as *fall through*. An example of process outputs is illustrated in Figure 2.4. In this figure two different process outputs are shown, $y_1 = 32\%$ ($< 1120\ \mu m$) and $y_2 = 14\%$ ($< 630\ \mu m$). [1] A summary of the process outputs can be seen in Table 2.1

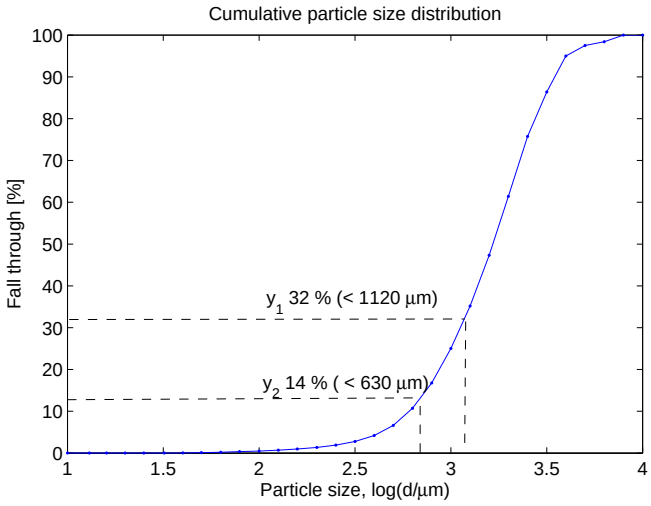

Figure 2.4: *A cumulative particle size distribution curve of wheat grinded with corrugated rollers. In the figure, two possible outputs are shown. Note that the scale on the x-axis is logarithmic, e.g. 1 = 10 µm, 2 = 100 µm etc.*

[1] In theory, the sieve size can be chosen arbitrarily, e.g. 431 μm or 598 μm, but because of practical reasons the number of different sieves have been limited to $\{125, 250, 355, 450, 530, 630, 710, 840, 900, 1120\}$ μm.

Process output	Symbol	Unit	Range	Description
Fall through	$y_{1,2}$	%	0 - 100	Amount of particles in % that fall through a certain sieve size.

Table 2.1: *The defined process outputs with short descriptions, symbols, units and ranges.*

2.3.2 The Control Signals - the Process Inputs

In section the control signals or process inputs are defined.

Roller Gap The parameter which has the biggest influence of the grinding process, is without a doubt the distance between the roller. This is intuitively easy to realize, as a smaller roller gap would give more finely grinded particles than what a bigger roller gap would give. The roller gap is measured in μm where the smallest gap is $350\,\mu m$ and the biggest gap is $850\,\mu m$.

The control signal however is not measured in μm, but in an artificial unit ranging from 533 [-] to 600 [-]. In this unit, the smallest roller gap (350 μm) corresponds to 533 [-] and the biggest roller gap (850 μm) corresponds to 600 [-].[2]

Differential Speed The second parameter to control the roller mill with is the differential speed between the rollers. The range is between 2 and 4, where 2 means that the faster roller spins two times as fast as the slower roller. Increasing the differential speed between the rollers, means that the breaking forces become bigger and that the particles on average become smaller.

Summary of control signals The inputs to the process are summarized in the Table 2.2.

Process input	Symbol	Unit	Range	Description
Roller gap	u_1	-	533 - 600	Distance between the rollers
Differential speed	u_2	-	2 - 4	Ratio between the speed of the faster roller and the slower roller

Table 2.2: *The defined process inputs, also called the control signals. Shown in the table are the symbols, units, ranges and a short description.*

2.4 Key Properties of the Roller Mill

When changing the input parameters to the roller mill e.g. the gap between the rollers or the differential speed between the rollers, the particle size distribution of the outgoing product is changing in a dynamic behavior. However the dynamics are

[2]The reason for this artificial unit has to do with the traditional way of adjusting the roller gap, used by the head millers. For more details and a unit conversion table, see Appendix A.

so fast and in the order of milliseconds. This makes it impossible and of no use to model this behavior, at least not for the purpose of automatic control. The behavior of the system to a change in input parameters is therefore considered to be static, i.e. when a parameter is changed, the system immediately reaches a new steady state.

Even though the process itself can be considered to be static, the sensor measuring the particle size of the outgoing product introduces dynamics into the system. The sensor will show a transient behavior of which length depends on the number of particles measured.

With the problem analysis done in this chapter, the following key properties of the process are identified:

- The dynamics of the grinding process are very fast. After change in gap size, the particle size of the exiting product is affected within milliseconds.
- The system is stable, i.e. bounded input gives bounded output. However, a bad controller can cause the output signal to oscillate.
- The sensor measuring the particle size distribution has a slow time constant and shows a transient behavior for approx. 2 min before showing steady state behavior. This time is however dependant on N, how many particles the sensor measures.
- Although many studies and experiments have been carried out, there exists no quantitative model of the roller mill [3]. Since so many parameters affect the grinding properties, i.e. wheat type, moisture levels, temperatures etc, the studies can only be interpreted in a qualitative manner.

2.5 Controller Goals

With the above description of the roller mill and the particle size sensor the following controller goals are defined. With these goals, different controller algorithms are developed, which are explained in the remaining part of the report.

2.5.1 Controller Goals - SISO Control

The SISO controller (Single-Input Single-Output) should in both simulation and implementation, do the following:

1. The controller should make the output signal, the fall through of wheat particles, follow a set-point. This should be done by changing the gap between the roller pair.
2. The controller should be able to bring the output signal from one set-point to another within the time it takes for the sensor to show two steady state outputs. This means that e.g. if it takes 2 minutes for the sensor to show a steady state value, the controller should be able to reach the new set-point in 4 minutes.

[3] These studies have been carried out by Bühler Group AG and are therefore confidential.

3. The controller should be able to handle the changing conditions of the wheat, e.g. different types of wheat and different moisture levels of the wheat. No re-tuning of the controller should be required if these events occur.

4. The changing conditions explained above should not result in bad controller behavior, i.e. that the output signal becomes unstable.

5. The output should not oscillate around the set-point.

6. The controller should be able to handle measurement noise from the sensor and also the fact that the measurement noise can very depending on the amount of particles, N, that are measured, as explained in Section 2.4.

7. The controller should be able to handle the transient behavior of the sensor which occurs when the grinding parameters are changed. The length of the transient behavior can vary depending on N, the amount of particles that are measured.

2.5.2 Controller Goals - MIMO Control

The MIMO controller (Multiple-Input Multiple-Output) should in both simulation and implementation, do the following in simulations:

1. The controller should be able to control two process outputs by adjusting the roller gap and the differential speed.

2. The controller should be able to handle simulation models where the variables have large cross couplings between each other. With cross couplings it is meant that one input has large effect on both outputs.

3. The controller should be able to handle changing conditions of the input/output relations of the process models, i.e. adapt itself to changing process models.

Chapter 3

Control and Modeling Theory

In this chapter, the concept of control theory is introduced followed by a description of the different controller algorithms used in this project. Lastly an introduction to modeling is presented.

3.1 Introduction to Automatic Control

Automatic control can be divided into two segments; feedback control and feed-forward control, which are briefly explained below.

3.1.1 Feedback Control

The concept of feedback control is the foundation of which automatic control is based on. The basic structure/block scheme of how a feedback controller looks can be found in all books concerning control theory, e.g. (Lennartson 2000). The principle behind feedback control is shown in Figure 3.1.

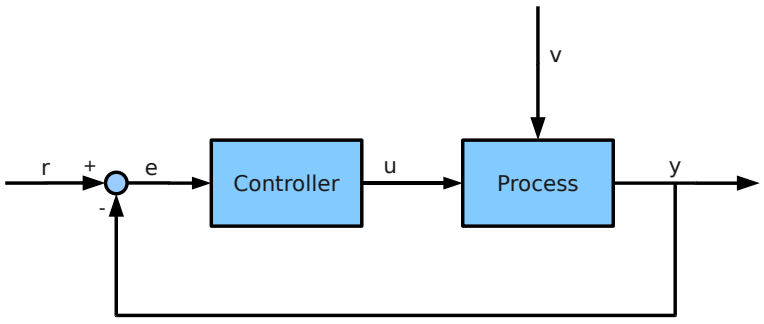

Figure 3.1: *A block scheme of a feedback controller. The loop is closed by feeding back the output signal from the process to create an error signal. This error signal is used to calculate an output from the controller.*

The two basic blocks are the process and the controller, where the process is what we desire to control e.g. a model of a system. The controller block is what decides

what control signal, u, to send to the process. The reference signal, r, is the desired output signal, from here on called the set-point. This is with other words what we wish the output, y, from the process to be.

The feedback loop is closed by subtracting the set-point signal from the current output of the process. The signal received is called the error, e and this is the signal which the controller receives. What the controller sends out is called the control output and is in this report called u.

Present can also be a disturbance, v, which affects the process in an often unknown way. The disturbance could for example be a steep hill the case of a cruise control mechanism in a car. It can be shown that feedback control will compensate for disturbances, non-linearities in the process and modeling errors, (Lennartson 2000).

3.1.2 Feed-forward Control

Sometimes it can be desirable to add a feed-forward controller to the control loop. Feed-forward is another word for open-loop controller, which means that there is no feedback between the output from the process and the set-point. A block scheme with the added feed-forward is shown in Figure 3.2.

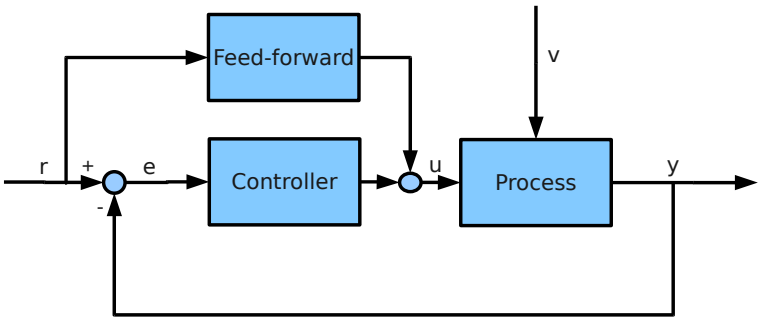

Figure 3.2: *The same block scheme as in Figure 3.1, but with the added feed-forward controller. The feed-forward controller is outside of the control loop and reacts immediately when the set-point, r, changes.*

The goal of the feed-forward block is to react immediately, i.e. when the set-point is changed, the feed-forward block calculates a new control output which should ideally make the output and the set-point equal. Feed-forward cannot handle disturbances or model errors, since there is no feedback present. On the other hand, a feed-forward controller can react immediately upon a external change, whereas a feedback controller can only react after an error has been discovered.

3.2 PID Control

The PID controller is the most used control strategy in the world, some say that up 95 % of all controllers implemented are PID controllers. The reason to why it is so popular is probably because of its simplicity and relatively understandable tuning

parameters. It also does not require an exact model of the system to control, i.e. it can be seen as a model free controller algorithm, which makes it an interesting candidate for implementation in a roller mill. The PID controller algorithm in continuous time in parallel form is described as

$$u(t) = K_p\, e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}\,, \tag{3.1}$$

where K_p is the proportional gain, which amplifies the current error and thus increasing K_p, will give a faster response of the system.

K_i is the integral gain, which sums up all the errors previously and can therefore remove a steady state error.

K_d is the derivative gain which looks at the rate of change of the error and multiplies with a gain. It can be seen as a prediction of how big the error will be in the next time step.

3.2.1 Digital PID Normal Form

Since this project concerns computer implemented control algorithms, the digital form of the PID controller is better used. The advantage is that it is easily implemented in a computer programming language, in this case MATLAB.
The digital form of the PID controller can be found in e.g. (D.A.Pierre 1999), and is written as

$$u(k) = K_p\, e(k) + K_i \sum_{m=0}^{k} e(k) + K_d[e(k) - e(k-1)]\}\,, \tag{3.2}$$

where k is the current sample. This expression is basically the same as Equation 3.1, but with the integral operator replaced by a summation, and the differential operator replaced by the first backward difference.

3.2.2 Digital PID Incremental Form

The recursive PID controller updates the control signal recursively, i.e. each iteration the old control signal is updated with a control change. This has several advantages, e.g. bumpless transfer between manual and automatic mode, (Wittenmark and Åström 1997) and its' easiness to implement in a computer controller. The recursive PID controller can be derived by first considering the digital PID controller one sample back in time which is written as

$$u(k-1) = K_p\, e(k-1) + K_i \sum_{m=0}^{k-1} e(k-1) + K_d[e(k-1) - e(k-2)]\,. \tag{3.3}$$

Subtracting 3.3 from 3.2 gives

$$\begin{aligned} u(k) - u(k-1) = {} & K_p\,[e(k) - e(k-1)] \\ & + K_i[\sum_{m=0}^{k} e(k) - \sum_{m=0}^{k-1} e(k-1)] \\ & + K_d[e(k) - 2e(k-1) + e(k-2)]\,. \end{aligned} \tag{3.4}$$

The summation sign can now be removed and the expression rearranged

$$\begin{aligned} u(k) &= u(k-1) \\ &\quad + K_p\,[e(k) - e(k-1)] \\ &\quad + K_i\,e(k) \\ &\quad + K_d[e(k) - 2e(k-1) + e(k-2)]\,. \end{aligned} \tag{3.5}$$

Finally the equation is sorted after the time displacement

$$\begin{aligned} u(k) &= u(k-1) \\ &\quad + (K_p + K_i + K_d)\,e(k) \\ &\quad - (K_p + 2K_d)\,e(k-1) \\ &\quad + K_d\,e(k-2)\,. \end{aligned} \tag{3.6}$$

From this it can be seen that when controlling a static system, the only parameter that is important is the K_i parameter. If the system is in steady state it would be wrong to take into account the error in the previous time step. When not using the K_p and K_d parameters, the control law is simplified into what from here on will be known as incremental I-control, and is described as

$$u(k) = u(k-1) + K_i\,e(k)\,. \tag{3.7}$$

3.3 Parameter Estimation using LMS

To be able to use some of the adaptive control algorithms that are presented in this report, some kind of parameter estimation technique has to be used. In this section, the least means squares estimation, (from now on LMS estimation), is presented. It is assumed that the system to estimate can be written as

$$y(i) = \Phi(i)\Theta\,, \tag{3.8}$$

where Θ, and Φ are

$$\begin{aligned} \Phi(i) &= [\phi_1(i) \quad \phi_2(i) \quad \cdots \quad \phi_n(i)] \\ \Theta &= [\theta_1 \quad \theta_2 \quad \cdots \quad \theta_n]^T\,. \end{aligned} \tag{3.9}$$

ϕ is a vector containing the different inputs to the system, y are the outputs to the system and i is the number of data points used for the estimation. The goal is to decide the parameters, Θ, in such a way that the system that is going to be identified is approximated as good as possible.

By expressing the goal function as the error between the measured output and the estimated output, it can be shown that the parameter selection of θ that minimizes this error is given by the least squares estimator (Wittenmark and Åström 1995)

$$\hat{\Theta} = (\Phi^T\Phi)^{-1}\Phi^T Y\,. \tag{3.10}$$

As seen, this equation contains a matrix inversion, which can cause numerical difficulties if the matrix is close to singular. Therefore a small distorted identity matrix can be added to prevent this, (T.Larsson *et al.* 2000) The new expression becomes

$$\hat{\Theta} = (\Phi^T\Phi + \rho I)^{-1}\Phi^T Y\,, \tag{3.11}$$

where I is the identity matrix and ρ is a small number which can be seen as a tuning parameter. Worth noticing is that Equation 3.11 solves a slightly different problem than Equation 3.11. ρ should therefore not be chosen bigger than what is necessary to reduce the risk of numerical difficulties during the matrix inversion.

3.4 Adaptive I-Control

The adaptive control algorithm aims to control systems that are slowly time-varying. The algorithm presented below comes from (T.Larsson *et al.* 2000) and (Roover *et al.* 1998) where it is referred to as run-to-run control. The algorithm uses the jacobian J, which is an estimated model of the system, in order to make a control signal change that will bring the output signal towards the set-point. The control law is described as

$$u(k) = u(k-1) + \alpha J^{-1} e(k)\,, \tag{3.12}$$

where

$$J = \begin{bmatrix} \frac{\Delta y_1}{\Delta u_1} & \cdots & \frac{\Delta y_1}{\Delta u_n} \\ \vdots & \ddots & \vdots \\ \frac{\Delta y_n}{\Delta u_1} & \cdots & \frac{\Delta y_n}{\Delta u_n} \end{bmatrix}, \quad u = \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix}, \quad e = \begin{bmatrix} e_1 \\ \vdots \\ e_n \end{bmatrix}, \tag{3.13}$$

and $y_1 \dots y_n$ are the output from the system, and $u_1 \dots u_n$ are the inputs to the system.

J is the jacobian containing the gradients, which states how much the different inputs affect the outputs. The control signal is calculated by inverting the jacobian matrix, which with this control law requires J to be a square-matrix. α is a parameter between 0 and 1. This parameter tells how aggressive the controller should be, where a small α means that the control signal change, du, will be smaller than if α is chosen closer to 1.

To estimate the jacobian, J, the LMS estimation algorithm from equation 3.10 is used.

Worth noticing is that in the SISO case, the adaptive control law is the same as the incremental digital PID controller from Section 3.2.2, but where K_i is replaced by αJ^{-1}, hence the name adaptive I-control.

Also worth noticing, is that the system must be relatively linear for this controller to work. This is because the jacobian is a linear approximation of the system. However if the system is linear, α is set to 1, and the jacobian is estimated correctly, the control law from Equation 3.12 will reduce the error with just one control signal change. The explanation for this, is that the jacobian is an approximation of the static gain of the system, i.e. $J = \frac{\Delta y}{\Delta u}$. Figure 3.3 shows an illustration.

If the current output is assumed to be y_1, and the desired output is y_2, a change of Δy is required. As seen in the figure, this means that the input must make a change Δu to go from u_1, to u_2, which means that

$$\begin{aligned} &\Delta y = J\Delta u \\ &\Longrightarrow \\ &\Delta u = \frac{1}{J}\Delta y\,. \end{aligned} \tag{3.14}$$

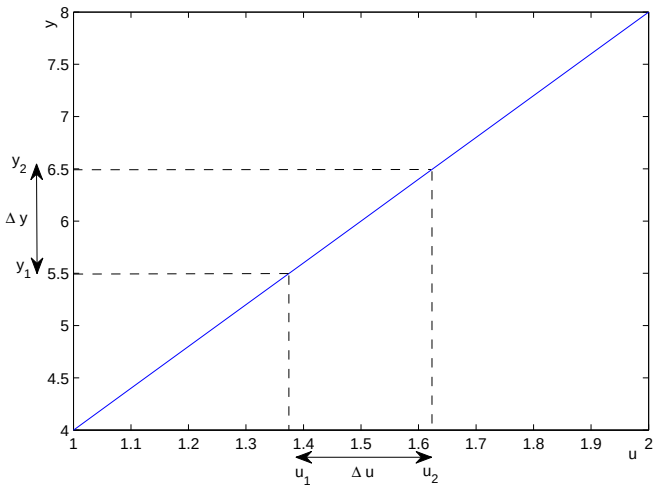

Figure 3.3: *A linear function with the output signal, y, on the y-axis and the control signal u on the x-axis. The figure illustrates why the adaptive I-controller can reduce the error between the output signal and set-point in one step.*

Since Δy is known to the controller, the output signal will therefore be reached in one step, if J is estimated correctly.

By multiplying J with α, a number smaller than 1, it is also possible to make the control less aggressive. This means that the output signal will not be reached with one control signal change, but also that there is less risk of overshoot of the output signal, if J is incorrectly estimated.

The last thing to say is that although the control law can be used for MIMO systems, in this project only the SISO case has been evaluated.

3.5 MIMO Optimal Control

This control algorithm can be seen as a more general form of the adaptive I-control algorithm, where an optimization problem is solved. The algorithm can be found in e.g. (T.Larsson *et al.* 2000) and is written as

$$u(k) = \min_{u(k) \in U} (||r(k) - \hat{y}(k)||^2 \,, \tag{3.15}$$

with $\hat{y}$ described as

$$\hat{y}(k) = \hat{J}(u(k) - u(k-1)) + y(k-1) \,, \tag{3.16}$$

where u and y are vectors containing the inputs and the outputs from the controller.

This algorithm uses a model of the system, namely the jacobian, J, which is estimated with the LMS-algorithm from Equation 3.10. As seen in Equation 3.15, it then tests all possible control signal combinations. The control signal combination that gives the least error between the set-point, $r(k)$ and the estimated output $\hat{y}(k)$ is then used as control signal. This approach is very good at handling constrains, since it is possible to decide a subset of the control signals, U, which should be optimized.

To put more emphasis on minimizing a certain error, the following criteria can be used

$$V = \frac{1}{2} E^T W E \,, \tag{3.17}$$

where $E = [r(k) - \hat{y}(k)]^T$.

In the 2x2 case, the weight matrix, W, is expressed as

$$\begin{bmatrix} \gamma & 0 \\ 0 & 1-\gamma \end{bmatrix}, \tag{3.18}$$

where $0 < \gamma < 1$.

The new control law can now be written as

$$u(k) = \min_{u(k) \in U} V \,. \tag{3.19}$$

This is the control law which is used by the MIMO controller which is presented in Section 6.3.2.

3.6 Signal Filtering and Buffer Algorithm

When controlling systems where the process output signal is very noisy, it is often wanted to filter the signal before sending the signal to the controller algorithm. With less noise in the output signal, it is possible to apply more precise control. When the K_d-parameter of a PID controller is used, it is usually necessary to apply filtering to the output signal. Otherwise the noise will be amplified when the differentiation is carried out, which can cause an unstable controller (Wittenmark and Åström 1997).

3.6.1 Signal Filtering

One filter algorithm is the well known first order low-pass filter, which is written as

$$y_k(k) = \alpha \cdot y_s(k) + (1-\alpha) \cdot y_k(k-1) \,, \tag{3.20}$$

where y_k is the filtered signal,y_s is the unfiltered signal and α is the filter constant. The first order low-pass filter can also be called a moving average filter, since it calculates the next signal by averaging the two latest signal samples.

A first order low-pass filter is many times enough to reduce the noise level of the signal, but in some cases it can be useful to use a filter that averages more samples, written as

$$y_{kb}(k) = \frac{y_s(k) + y_s(k-1) + \cdots + y_s(k-M)}{M} \,, \tag{3.21}$$

where M is the number of samples that are averaged. The higher M is, the lower the noise lever of $y_k b$ will be. However for each sample more that is averaged, the delay becomes longer. If the signal is filtered with moving average filter of length 5, the delay would be 5 samples etc. When filtering in real-time, which is usually the case for a controller algorithm, M therefore cannot be chosen too high.

3.6.2 Buffer Algorithm

To combine the advantages of the low delay with the moving average signal, and the fast response given by the low-pass filter, the following algorithm is proposed,

$$\begin{aligned} &\text{if} \quad |r(k) - y(k)| < 0.5 \\ &\text{use } y_{kb} \text{ for control} \\ &\text{if} \quad r(k) \neq r(k-1) \quad \text{or} \quad |y(k) - y(k-1)| > 4 \\ &\text{use } y_k \text{ for control}\,, \end{aligned} \tag{3.22}$$

where y_{kb} is the moving average signal, and y_k is the low-pass filtered signal. The idea is to use y_k when the a fast signal is needed, i.e. when the set-point has changed and the output signal should move to the new set-point. When the output signal is close to the set-point, y_{kb} can be used to get a signal with less noise, so that the controller can fine-tune the output signal.

3.7 Steady-State Detection

Sometimes it is desired to only apply automatic control when the output signal is in steady-state, i.e. when the transient behavior of the signal is gone. To be sure that the sensor is in steady-state, one can use a very high sampling time for worst-case scenario. This will make the control algorithm slow, since it always waits a fixes time, even if it is not necessary, and the signal already is in steady-state.

A better option would be to detect when the sensor is in steady-state, and the algorithm proposed can be written as

$$\begin{aligned} y_{b1}(k) &= \frac{y_s(k) + y_s(k-1) + \cdots + y_s(k-M)}{M} \\ y_{b2}(k) &= \frac{y_s(k-1) + y_s(k-2) + \cdots + y_s(k-M-1)}{M} \\ &\text{When } y_{b1}(k) - y_{b2}(k) < e_{threshold} \\ &\qquad \text{Steady state } = 1\,, \end{aligned} \tag{3.23}$$

where $e_{threshold}$ is a tuning parameter which sets how tough the algorithm should be, and M tells how many samples that should be averaged. The algorithm consists of two moving average signals, y_{b1} and y_{b2}, who each takes M of the latest output signals and creates a mean value. The only difference is that one of signals is time delayed one time sample. At each iteration, the difference between y_{b1} and y_{b2} is calculated, and when the difference is within $e_{threshold}$, the system is said to be in steady state.

3.8 Introduction to Modeling

The purpose of making a model of a system is to be able to answer questions concerning the system without having to make experiments. A model of a system can be expressed in different ways, for example mental models, verbal models, physical models and mathematical models, (Ljung and Glad 2004).

The mathematical model is the most common form of modeling in technical systems and is what has been investigated in this project and in the following sections the methods are described.

3.8.1 Modeling based on Physical Principles

Models of this type are based on known physical relations, i.e Kirchhoffs laws, laws of nature etc. To be able to use this kind of model, the system has to be of such kind that it is possible to set up balance equations of how the system behaves. Usually, most systems can to a certain extent be described with this modeling method.

3.8.2 Modeling based on Identification

Identification, also known as system identification, is the method of by making observation or experiment on the system, deciding a mathematical model. The system can be seen as a black box, where only the inputs and the outputs of the system are known. With the help of the inputs and outputs from the black box, a model can be estimated using various techniques found in modeling literature. This approach means that no consideration is taken to the underlying physical relations and can be good for very complex systems that would take to much time and effort to model using the first principle of modeling.

3.8.3 Static Models vs Dynamic Models

Normally a system consists of dynamic and static parts. A static part can be for example the well known Ohms-law, $U = R \cdot I$, where R and I have a direct effect on how U is changed if the temperature is kept constant. Dynamic systems have parameters that not only depends on current time, but also on previous time. Normally differential equations are used to describe dynamics in a system, for example how the current in a capacitor varies over time.

Chapter 4

Modeling the Roller Mill

In this chapter, the models of the roller mill, used for both the SISO and MIMO case are presented. The experiments carried out on the real roller mill machine are shown, and the model parameter selection is explained.

4.1 Purpose of the Roller Mill Model

It is generally seen as very difficult or almost impossible to make an accurate model of how all parameters, such as moisture, wheat type, temperature, roller gap etc, affect the grinding properties of the roller mill. To make a model this detailed would take too much time and effort considering the limited time of the project.

The purpose of the model in this project was therefore limited to only capturing the main characteristics of the roller mill. The model inputs where limited to the roller gap and differential speed between the rollers. The output of the model was limited to be the fall through, i.e. measurements on the the cumulative particle size distribution curve, as explained in Section 2.3.1. The main goal with the model was to test the different controller algorithms in order to verify the functionality before implementing them on the real roller mill experiment machine.

4.2 Experiments for the SISO Model

To get more information about the process and to be able to make a model of it, two different experiments where conducted. With the result of these experiments, a model for use in simulations has been proposed in Section 4.3.

4.2.1 Step Response Experiment

A step response is a relatively easy experiment to carry out on a real process. Despite its simpleness, it gives a lot of valuable information of the system, e.g. the time constant, delay, amplification, etc. The purpose of this step response experiment was to see how a change in gap between the rollers affected the particle size of the outgoing particles.

The number of particles the sensor measures was set to 130 000 and the control signal was set to change from 550 to 567. The result of the step response is shown i Figure 4.1.

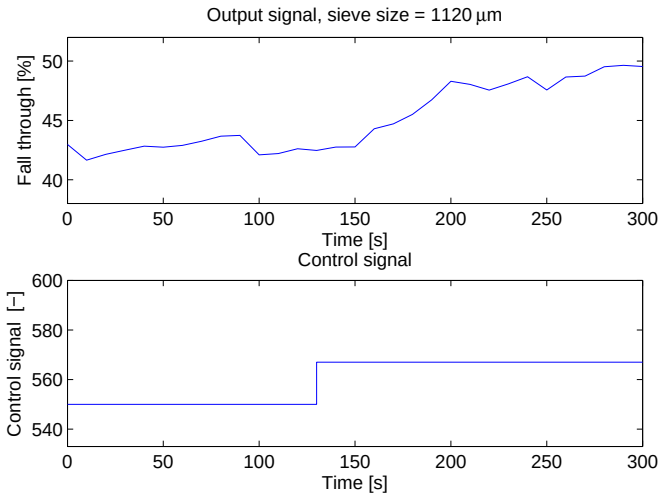

Figure 4.1: *A step change is carried out on the experiment machine and the resulting output signal is registered.*

As seen, the output changes from approx. 42 %, to approx. 48 % when the control signal is changed. A small delay can also be seen in the step response, because it takes approximately 15-20 seconds for the output to start to react to the control change. What also can be seen is the large amounts of noise present in the output signal.

4.2.2 Relation between Roller Gap and Particle Size

Previously there has been many attempts to make experiments on the roller mill to see how the different variables affect the different outputs from the roller mill. These reports however, only shows linear relations between the different parameters. This is because not enough measurements where taken to be able to make more than linear interpolations.

To get more information on how the relation between the gap size between the rollers and the particle size output, a bigger experiment was carried out. The gap between the rollers where changed in 7 steps, from a big gap and then in increments down to a very small gap. The product grinded from the respective experiments where then collected in separate bins, and then the particle size distribution was measured offline with the help of the sensor. The result is shown in Figure 4.2.

As seen, there seems to be an almost completely linear relation between the gap size between the rollers and the particle size output from the sensor. As a conclusion to this, it was decided that the process could be approximated with a linear model. Worth noticing is that only gaps within the operating range of the roller mill have been tested. For gaps outside of the operating range, it might be that the relation is not quite so linear.

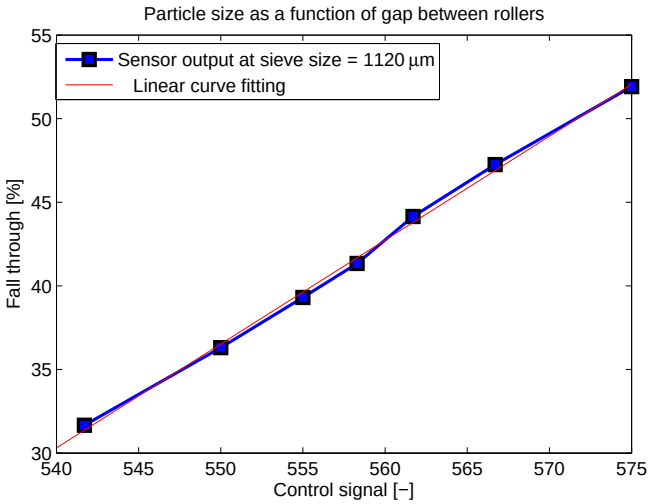

Figure 4.2: *The roller gap was adjusted in seven steps and the resulting fall through is plotted.*

4.3 Derivation of the Roller Mill Model

With the experiments that where conducted in Section 4.2 a model of the roller mill was built. This is described next.

4.3.1 Proposed Model Structure

By looking at the step response in Figure 4.1, it can be seen that the sensor seems to behave similar to the step response of a first order transfer function. Also considering that the system seems to be quite linear, as seen in Figure 4.2, it was decided to use this structure to model the roller mill. The standard form of the first order transfer function is written as

$$G(s) = \frac{Y(s)}{U(s)} = \frac{K}{Ts+1}\,, \tag{4.1}$$

where K is the static gain of the system, which is defined as the value of the transfer function when $s = 0$ (Lennartson 2000). T is the time constant of the system, i.e. the time it takes for the output signal to reach 63 % of its final value when a step response is applied.

To estimate K and T, the two experiments from Section 4.2 where used. By fitting a linear curve to the points in Figure 4.2, the static gain is estimated to $K = \frac{\Delta y}{\Delta u} \approx 0.55$.

The time constant of the system is found in Figure 4.1. In this figure the final value is approximated to 48.5 % and the start value to 43 %. This means that the output signal reaches 63 % of its final value at $y \approx 46.5$ %, which means that the time constant can be estimated to $T = 30$ s.

4.3.2 Model on state-space form

To be able to implement the model in controller simulations in Chapter 5, the transfer function model from 4.1 is transformed into a state-space model. The model on state-space form is described as

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) + c \\ y(k+1) &= Cx(k) + +dw(k)\,, \end{aligned} \tag{4.2}$$

where A, B and C are the standard parameters of a state-space model. These parameters are given when the transfer function from 4.1 are transferred into state-space, using the MATLAB command *ss*.

The c-parameter gives the offset of the model, which tells what the output should when the input (the control signal) is 560 [-]. This parameter depends on which sieve size that is chosen.

The d-parameter is a noise parameter which determines how much noise should be present in the simulations. This parameter can be varied to emulate the amount of particles the particle-size sensor measures. The type of noise, $w(s)$, was chosen to be normally distribute zero mean with standard deviation 1. The amplitude of the noise is then decided by the d-parameter.

4.3.3 Model Parameter Selection

What is important to remember is the characteristics are very much dependent on the type of wheat, the moisture level, temperature, etc. To account for that in the simulations, the model parameters where chosen in intervals, with starting point in the parameters found in 4.3.1. The value of the parameters and the intervals within they can vary, are shown in Table 4.1. The intervals have not been chosen based on

Parameter	Interval
K	0.45 - 0.55
T	20 - 30
c	(-245) - (-275)
d	0.5 - 1.5

Table 4.1: *The intervals in which the model parameters can change. The parameters where estimated in 4.3.1 and then extended in intervals in order to create a more flexible simulation model.*

experiments, since this would take too long time. Instead the intervals are chosen to give a big enough difference between the minimum and maximum parameter values. The response for the maximum and minimum value for each parameter can be seen in Section 4.4.

Since these parameters can be hard to interpret, they have been translated to parameter which all have a physical meaning on the real roller mill experiment machine. The new parameters can be set by the users in the simulation interface developed and are explained in Table 4.2.

User parameter	Description	Effect on model parameters	Interval
N	The amount of particles that should be measured, higher number means less noise and slower model response. Corresponds to value set in the particle-size sensor.	$\uparrow N$ $\Rightarrow$ $\downarrow T$ $\downarrow d$	40000 - 130000
Sieve size	The sieve size which should be used in the simulation, sets the offset of the model. Can be set in discrete values.	$\uparrow$ sieve size $\Rightarrow$ $\uparrow c$	450, 570, 630, 710, 1000, 1120 $[\mu m]$
Wheat type	The type of wheat changes the static gain of the model.	Dry wheat $\Rightarrow \uparrow$ K Moist wheat $\Rightarrow \downarrow$ K	normal, very dry, very wet

Table 4.2: *Description of the parameters that can be chosen in the simulation program.*

The third column is the most important one, and tells the relation between the parameters in Table 4.1 and Table 4.2. For example, if N is set to its maximum of 130000, both T, the time constant of the model, and d, the noise level of the model, will be at their minimum level. This is also intuitively easy to understand, as increasing N, means that the amount of measurement noise decreases and the time constant increases. N was derived in order to emulate the way the particle size sensor in the real experiment setup is tuned.

4.4 Model testing and verification

In this section the model is tested for different parameters to verify the behavior.

4.4.1 The speed of the model

How fast or slow the model reacts to a step response change is an important factor. In Figure 4.3, the step response of the model is shown for both the highest and lowest value of the T-parameter. In order to make the difference between the two curves more clear, no noise has been used in this simulation. The slower curve reaches its final value of 51 % at $t = 260$ s, whereas the faster one reaches its final value at $t = 200$ s. The difference between the two curves corresponds well to what the difference is the real system, when comparing measuring many particles with few particles. The most important thing with the model is that it roughly estimates the behavior of the real system.

4.4.2 The Gain of the Model

The moisture level of the wheat is modeled by varying the static gain, K of the model. In Figure 4.4, a comparison between the biggest and smallest static gain of the model is shown. As seen with $K = 0.55$, the output signal reaches y = 56 %,

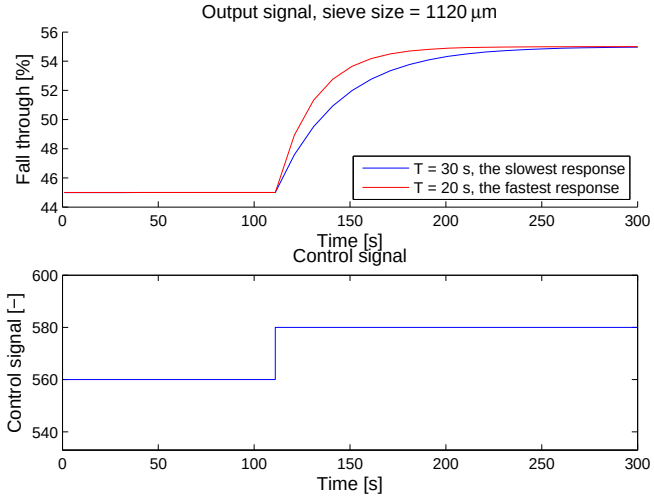

Figure 4.3: *A comparison between the fastest and slowest model response. A step change on the model and the resulting output is shown.*

and with $K = 0.45$, it reaches 54 %, which makes a difference of two percentage points between the two signals. This can be seen as a good approximation of how it looks like in the real system when changing the moisture level of the wheat.

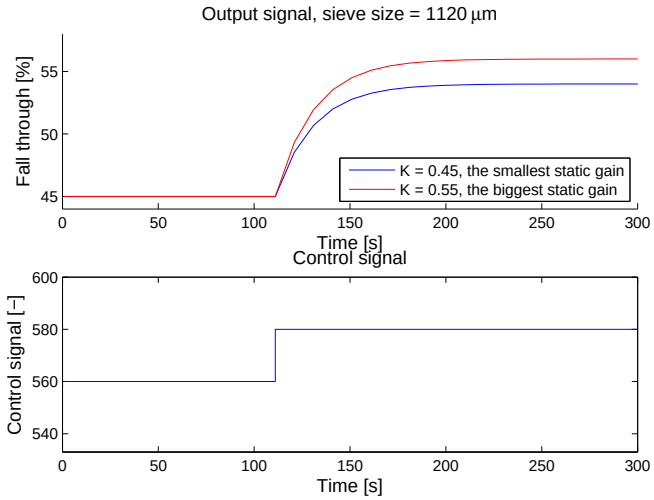

Figure 4.4: *Aa comparison between the highest and lowest static gain parameter of the model. A step response is made and the resulting output is shown.*

4.4.3 The Sensor Noise

The level of sensor noise in the model is determined by the d-parameter. Since it is hard to determine the noise level of the sensor signal, as explained in Section 2.2.2, it is important that the model has an adjustable noise-parameter. This makes the controller simulations more realistic, as the control algorithm can be tested with different sensor noise levels. In Figure 4.5 the step responses of the model with the highest sensor noise, $d = 1.5$, and the lowest sensor noise $d = 0.5$, are shown.

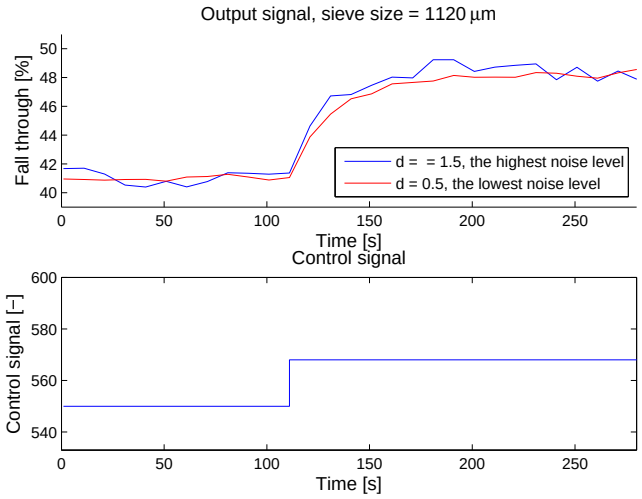

Figure 4.5: *Step response of the model, showing the maximum noise level and the minimum noise level.*

As seen, the noise level is representative for how the real system looks like, shown in Figure 4.1. With the noise added to the model, it was possible to make more realistic controller algorithm simulations, as shown in Chapter 5.

4.5 Experiment for the MIMO Model

For the MIMO system, a bigger experiment series was carried out. This was done in order see how the two inputs affect the two outputs of the system, and to see if it was possible to realize a controller for the MIMO system.

4.5.1 Experiment Result

The MIMO model has as explained in Section 2.3, two inputs and two outputs. The inputs are the roller gap and the differential speed. The outputs are two points on the particle size distribution curve, e.g. 50 % fall through at 1120 μm and 70 % fall through at 630 μm.

In this experiment the roller gap and speed differential was changed in 5 steps each. First, the roller gap was kept constant and the differential speed was changed in 5 steps. After that the roller gap was changed to a new value, and then kept constant once again, while the differential speed was changed in the same 5 steps. This was repeated until all combinations of differential speeds and roller gaps where tested, making it 25 experiments in total. For each input combination, the fall through was measured. The result from one of the experiments is shown in Table 4.3.

In Figure 4.6 and Figure 4.7, the result of the experiment is shown for the sieves 1120 μm and 630 μm. The figures graphically show the result of experiments such as in Table 4.3. As seen the figures looks like tilting planes, with quite linear behavior. Also seen, is that the speed differential does not seem to affect the fall through very much.

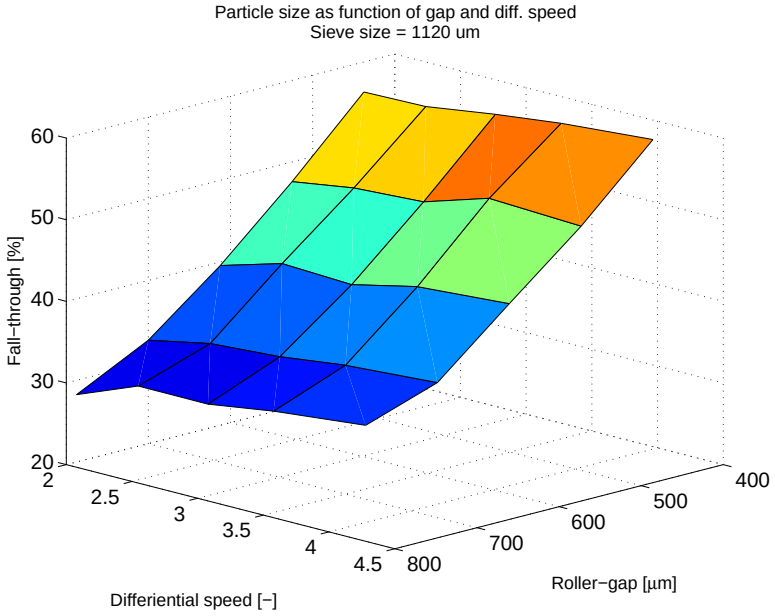

Figure 4.6: *Experiment result of how the roller gap and the differential speed affect the fall through at the sieve size 1120 µm.*

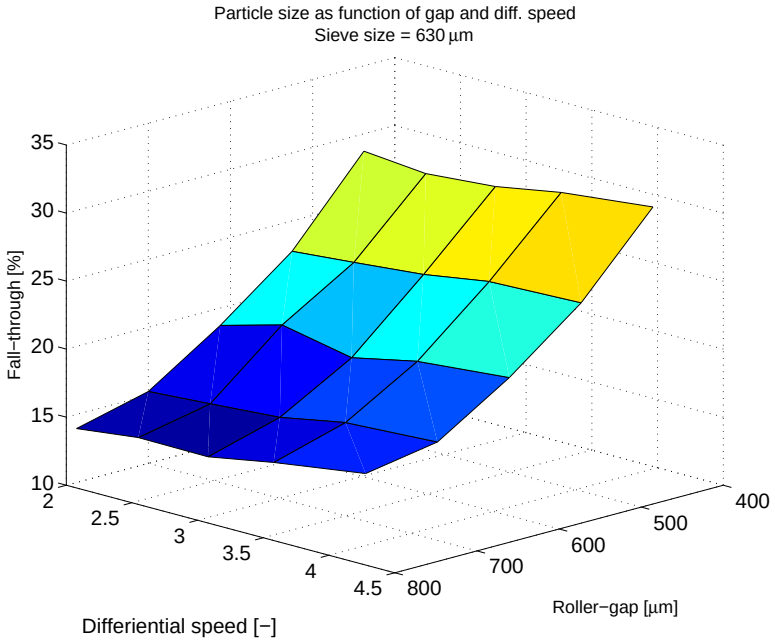

Figure 4.7: *Experiment result of how the roller gap and the differential speed affect the fall through at the sieve size 630 µm.*

	Differential speed [-]				
Roller gap [μm]	2	2.47	3	3.5	4.2
437.5	56.3	56.5	57.7	58.6	59.5
525	47.6	48.7	49.2	51.7	51.2
612.5	39.6	41.8	41.3	43.2	43.9
700	32.7	34.2	34.8	35.8	36.5
787.5	28.2	31.2	31.2	32.4	33.6

Table 4.3: *This table shows how the parameters where varied during the MIMO experiment. The column furthest to the left, shows the roller gap settings. The row highest up in the table shows the settings for the differential speed. The 25 remaining slots shows the measured fall through in %, at the specific roller gap and differential speed combination, for one of the experiment runs*

4.5.2 Interpretation of MIMO experiment results

Figure 4.6 and Figure 4.7, show that the differential speed has very little influence on the grinding result, i.e. the fall through of particles. This result shows that the idea of having the differential speed as second control parameter had to be abandoned. It might be pointed out that a higher speed differential of around 10-15 [-], could have given a bigger influence on the particle size distribution. This would however cause the electric motors driving the rollers, to consume more energy which was not desired. [1]

Because of the above result, it was decided to focus the MIMO system controller algorithms, to simulations only. The controller algorithms in Chapter 5 where tested on similar surfaces as in Figure 4.6 and Figure 4.7. The reason for making simulations of MIMO controllers, is that they might be of use in the future, if other sensors, measuring other qualities of the product are developed. For example, it might be that the differential speed has a big influence on the ash content of the wheat. It would then be possible to test the controller algorithms in Chapter 5 on the real roller mill.

[1] When the differential speed between the rollers is too high, the rollers will work against each other. I.e. the faster roller will try increase the speed of the slower roller, and the the slower roller will try to decrease the speed of the faster roller. This phenomena will cause an unnecessary high energy consumption.

Chapter 5

Simulation of SISO Controllers

In this chapter, the SISO controller algorithms are tested and evaluated on a simulation model of the roller mill. First the computer control interface is explained and then the controller algorithms and the results are explained.

5.1 Preparative Steps

In order to implement the controller algorithms digitally, a couple of preparative steps had to be taken, explained in this section.

5.1.1 The Computer Control Interface

A simulation program has been developed in MATLAB. The simulation program consists of a MATLAB script which contains all the necessary initializations and also the model of the roller mill. The MATLAB script then calls a MATLAB function containing the controller algorithm. The controller algorithm receives the current output from the model and the current set-point. The controller algorithm then calculates the appropriate new control signal and sends it back to the MATLAB script. The reason for keeping the model and initialization phase separate from the controller algorithms, was to better emulate the way the controller algorithms are implemented on the real roller mill experiment machine. Because of secrecy reasons, no MATLAB code is shown in this report, but a more detailed explanation of how the simulation program works can be found in Appendix B.

5.1.2 Signal Processing

In this chapter and the rest of the report has, unless anything else is stated, a first order low-pass filter been used, explained in 3.6.1. This is done in order to reduce the amount of noise of the output signal from the model, i.e. the fall through. The low-pass filter is used with $\alpha = 0.5$.

To reduce the control signal activity when the output signal is close to the set-point,

the following algorithm has been used

$$\begin{aligned} &\text{If } |r(k) - y(k)| < e_{min} \\ &\Delta u(k) = 0\,, \end{aligned} \tag{5.1}$$

where $0.5 \leq e_{min} \leq 1$. This algorithm sets the controller output to 0 when the error is small enough and will reduce the likelihood of output signal oscillations upon set-point convergence.

Because of physical limitations of the real roller mill machine, i.e. the roller gap cannot be changed in too small steps. The minimum control signal change has therefore been constrained. If the control signal from the controller algorithm is smaller than a certain minimum, Δu_{min}, the control signal is set to Δu_{min}, which is formulated as

$$\begin{aligned} &\text{If } 0 < |\Delta u(k)| < u_{min} \\ &\Delta u(k) = u_{min}(k)\,, \end{aligned} \tag{5.2}$$

where $u_{min} = 1.1$

5.2 Simulation Results of the Implemented Controllers

In this section, the different controller algorithms that have been used in simulations are presented. It is important to point out, as explained in Section 2.4, that no real reliable and accurate enough model of the roller mill exists. Because of this, more advanced controller algorithms, such as i.e. output state feedback control or internal model control to mention a few, where ruled out. This is because these algorithms require an good model of the process to be controlled. The controller algorithms presented below, requires no or little knowledge about the process and contain few tuning parameters which make them suitable for the roller mill process.

To be able to compare the different algorithms, the sieve size of 1120 μm has been kept constant throughout the SISO simulations. Also in all the simulations besides from the adaptive controller, the parameter for choosing the type of wheat has been set to normal.

5.2.1 Incremental I-control

As a first attempt of control, the incremental I-controller, explained in Section 3.2.2 was chosen. Because of the high noise level in the simulation model, it was concluded not to use the K_d parameter in the controller. The parameters for the simulation where: $K_i = 0.5$, $T_s = 30$ s, and $N = 65000$.

Choosing N that small, means that much measurement noise will be present in the signal. But a small N also means a faster output signal response, which is needed with this type of controller. Otherwise the controller goal of reaching a new set-point within 4 minutes, from Section 2.5.1 would not be possible to reach. This is because the control signal changes made in each iteration are too small with the I-controller.

The result of a simulation with the incremental I-controller is shown in Figure 5.1. As seen, the controller manages to track a set-point change that goes from 50 to

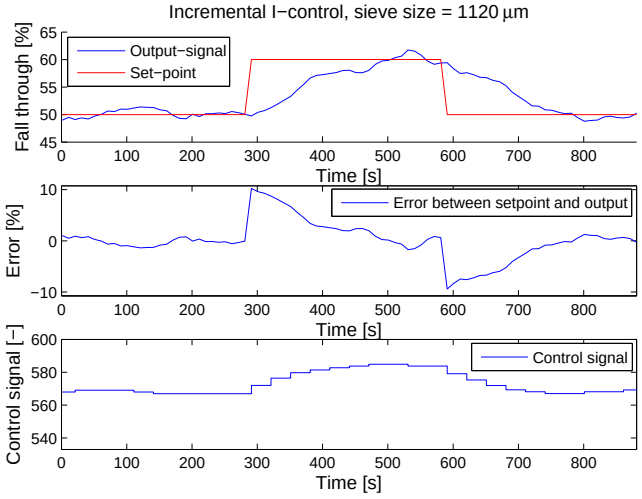

Figure 5.1: *Simulation result using an incremental I-controller.*

60 and then back to 50 again. Also seen is that the output signal shows oscillating behavior around the set-point, which can be seen between $t = 0$ s and $t = 300$ s. The controller tries to compensate for this by giving control signals, but this causes the control signal alternate up and down. The oscillations where tried to be eliminated with the controller showed in Section 5.2.2.

5.2.2 Incremental I-Control with Buffer

This controller deals with the problem of the oscillating control signal when the set-point is reached, which is present with the pure I-controller from Section 5.2.1. The oscillations in the control signal happens when the output signal, the fall through, is close to the set-point. Because the output signal contains noise, it means that it will never completely reach the set-point, i.e. it will move around in the region close to the set-point. The controller will try to compensate for this by sending a control signal, but since the noise level of the output signal is so high, it causes the control signal to oscillate around the set-point.

One solution to this problem would be to increase e_{min}, explained in Section 5.1.2. This would increase the region in which the controller is inactive, but would also cause less accurate control.

Instead, the buffer algorithm from Section 3.6.2 is used. The same controller parameters as 5.2.1 is used, but with the buffer algorithm added to choose between the low-pass filtered signal, and the moving average signal.

The result of the new controller algorithm, is shown in Figure 5.2. The two different filtered signals are shown in the first subplot. When the output signal is close to the set-point, the moving average signal with less noise is used to calculate the control signal. When the set-point changes at $t = 500$ s, the more noisy low-pass filtered signal is used to calculate the control signal. When the new set-point is reached, the moving average signal once again is started and used to calculate the control output. As seen, the control signal shows no, or much less oscillating behavior, meaning that the buffer algorithm seems to work quite well. Worth noticing is that the buffer

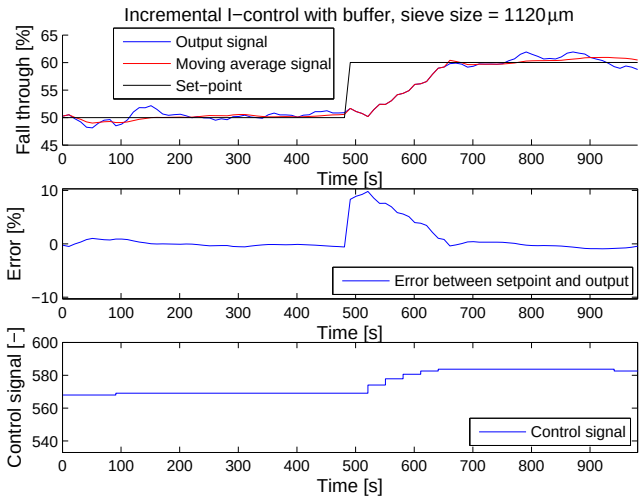

Figure 5.2: *Simulation result using an I-controller with added moving average algorithm to reduce control oscillations.*

algorithm probably needs to be analyzed more thoroughly that what has been done in this report.

5.2.3 Static Feed-Forward plus Feedback Control

The following controllers have a different approach than the controllers above. Instead of controlling the dynamic behavior of the output signal, only the static behavior is controlled. This means that the controllers sampling time is longer than the time it takes for the sensor to reach a new steady state when a control signal change has been made. The reason for trying this approach is that the sensor may give some strange behavior during the transient period. Also the risk of making the system unstable is reduced significantly if a static controller is being used.

Maybe the most important motivation for using a static controller, is that the relation between the roller gap and the particle size distribution is static, i.e. the particle size distribution immediately changes when the roller gap changes. This means that with the correct control signal, it should be possible to reach the a new set-point with only one control signal.

Since the goal with this controller is to reduce the error between the set-point and output in one step, first a feed-forward control signal is used which is activated when the set-point changes. This feed-forward controller is an incremental I-controller with a bigger gain than in the dynamic case. This feed-forward controller should ideally be able make the output signal reach the set-point in one try. But since always modeling errors and other unwanted effects exists, a feedback controller with smaller gain is used in order to remove the steady state error.

The parameters chosen for the simulation are: $K_{FF} = 1.5$, $K_{FB} = 0.5$, $N = 100000$ and $T_s = 100$ s. The result from the simulation is shown in Figure 5.3.

As seen in Figure 5.3, the controller first makes a big control signal change to bring the output close to the set-point. After that, smaller control signals changes are made in order to reduce the remaining error as much as possible. In this simulation,

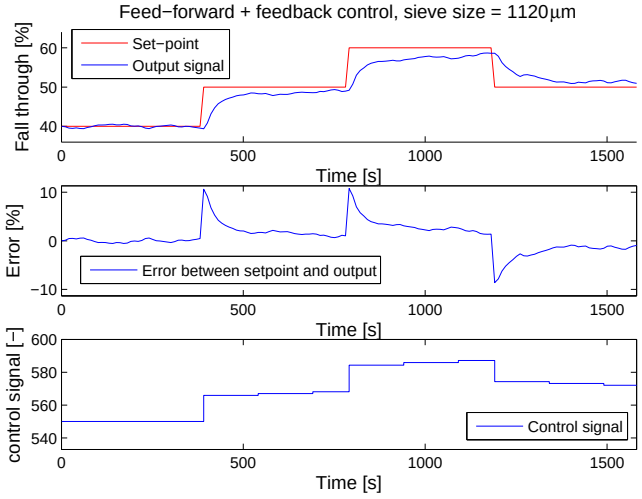

Figure 5.3: *Simulation result using a feed-forward controller with static gain plus a feedback controller.*

the controller is not capable of removing the steady state error before the set-point is changed. The solution to this problem would be to use a higher value of K_{FF}, which should be able to bring the output signal closer to the set-point on the first try. A simulation with the same parameters as before, but with the difference that $K_{FF} = 1.9$ is shown in Figure 5.4.

This time the controller manages to reach the new-set point before it is changed to a new value. As seen the controller only needs to give one feedback control signal, and at one time two in order to reach the set-point. It can also be seen that the time it takes for the controller to reach the new set-point is on average smaller than 4 minutes which was the demand given in the controller goals in Section 2.5.1. The problem remaining to be solved, is what value of K_{FF} that should be chosen? A

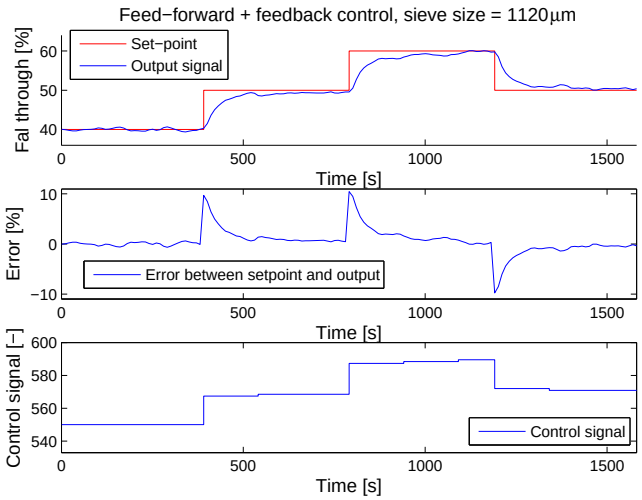

Figure 5.4: *Simulation result using a feed-forward controller with static gain plus a feedback controller. Here a higher feed-forward gain than in Figure 5.3 has been used.*

solution to this problem is presented in Section 5.2.4.

5.2.4 Adaptive Feed-Forward plus Feedback Control

Changing condition of the wheat, e.g. temperature, wheat type, and moisture level, has a big effect on how the output reacts to a control signal. It can therefore be difficult to have a static feed-forward gain, K_{FF}, since that would mean that for some types of conditions, the gain would be too big, i.e. there would be an overshoot. In some cases the output change would be too small, i.e. an undershoot. Clearly some kind of adaptation of K_{FF} is needed. The adaptive I-control algorithm, which was explained in Section 3.4, was chosen to solve this problem.

To get a good estimate of J in presence of measurement noise, several measurements of of $\frac{\Delta y}{\Delta u}$ are needed, and the optimal choice of J is calculated using the LMS algorithm as explained in Equation 3.10 in Section 3.3. In this simulation, the LMS estimator was set to collect three pairs of $\frac{\Delta y}{\Delta u}$ before calculating the first estimate of J. The simulation parameters are chosen to: $K_{FF,initial} = 1.4$, $K_{FB} = 0.5$, $N = 100000$, and $T_s = 100$ s. The simulation result is shown in Figure 5.5.

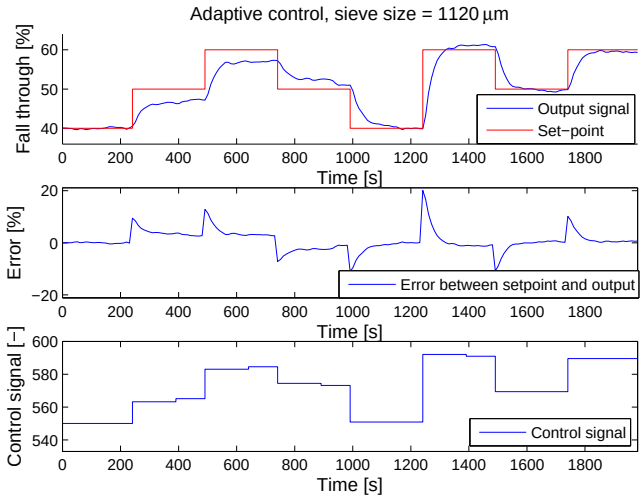

Figure 5.5: *Simulation result using a feed-forward controller with an adaptive gain plus a feedback controller.*

As seen, during the first three control signal changes, the output signal does not reach the set-point. When three input/output samples have been saved, the LMS estimator calculates an estimate of the gain, J of the system, which is then used as control parameter for the remaining part of the simulation. With the estimated gain, the controller manages to track the set-point changes with only one control signal and at one point two control signals. Table 5.1 shows how the estimate of K_{FF} changed during the simulation. The estimation seems to be quite stable in the region of $K_{FF} = 2$

$\hat{J}$	K_{FF} $(= \frac{1}{J})$
0.4935	2.0262
0.4884	2.0473
0.4993	2.0027
0.5042	1.9835

Table 5.1: *The estimated gradients and the corresponding control parameter, K_{FF}*

5.2.5 Steady State Detection Algorithm

The speed of the sensor, i.e. how fast it can measure the particle size of the wheat, can vary depending on various things, explained in Section 2.2.2. For example, moist wheat is harder for the sensor to measure than dry wheat. Also when the roller gap is smaller, the sensor can measure with a higher particle rate, since the time it takes to measure a small particle is shorter than the time it takes to measure a big particle.

The above explained phenomena makes it difficult to choose the correct sampling time. For the adaptive controller explained in Section 5.2.4, it is very important that the sensor is in steady state at the correct time. Otherwise it can happen that the LMS algorithm saves incorrect sensor outputs, which means that the gradient estimation will be faulty.

One way of solving this would be to use a sampling time for the worst case scenario. This means that for very moist wheat and big roller gap, the sampling time is still sufficiently long for the sensor to reach steady state before a new control signal is given. In this section however, an approach with a varying sampling time has been tested. To determine wether or not the signal is in steady-state, the steady-state detection algorithm, explained in Section 3.7, is used. The parameters are chosen to: $K_{FF} = 1.5$, $K_{FB} = 0.5$, $N = 60000$, $e_{threshold} = 0.2$, $M = 5$, and $T_{s,FB} = 100$ s.

In Figure 5.6, the result of a simulation is shown. As seen in the fourth sub-figure, peaks are shown when the signal is in steady state. When the set-point changes,

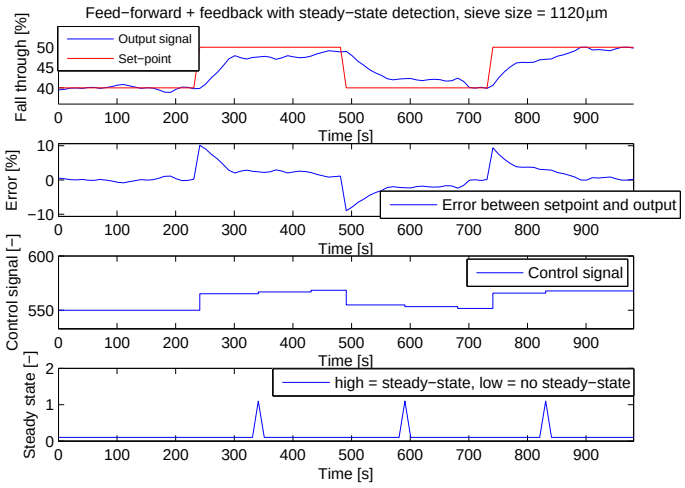

Figure 5.6: *Simulation result using a feed-forward controller with variable sampling time.*

a feed-forward signal is given and not until the output signal level is leveled out, a new control signal is given. After that the feedback controller takes over, which uses a constant sampling time.

5.3 Comparison of Simulated Controllers

The implemented controllers are here judged by looking at the controller goals from Section 2.5.1.

I-control The incremental I-controller from Section 5.2.1 and 5.2.2 is a simple controller, where K_i is more or less the only tunable parameter. Controller goal nr. 2 is fulfilled, since the controller reaches a new set-point within 4 min (the time to show a steady-state value was approximated to 2 min).

Controller goal nr. 3, requiring no re-tuning of controller parameters can not said to be fulfilled. This is because for some circumstances, e.g. a different wheat type or a higher moisture level, K_i might need to be changed. Otherwise unstable behavior can happen, as was later verified in experiments in Section 7.3.1, where the algorithm was tested on the real roller mill. The risk of unstable behavior also concludes that controller goal nr. 4 is not fulfilled.

Controller goals nr. 6 can be said to be fulfilled, since the I-controller can handle quite a lot of measurement noise. Controller goal nr. 7 is also fulfilled, since the I-controller also is gives control signals during the transient behavior of the sensor, i.e. is not required for the signal to be in steady-state for the controller to work properly.

Static feed-forward plus feedback control This controller fulfills the same goals as the I-controller from above does. The only exception is controller goal nr. 7, since the feed-forward control signal only should be given when the signal is in steady-state. In some circumstances, it could happen that the sensor has not had time to reach a steady-state behavior, but a control signal is still given. This could cause the controller to become unstable, also violating controller goal nr. 4. Also controller goal nr. 3 is not fulfilled, since the controller might require re-tuning.

Adaptive feed-forward plus feedback control This controller fulfills all but the last of the controller goals stated. Although the controller algorithm itself is more advanced than the other algorithms, it does not have any tuning parameters. If for example the the wheat type is changed, the controller can adapt its controller parameter in order to adapt to the new situation.

There might be a risk of faulty gradient estimations if too much measurement noise is present, violating controller goal nr. 6.

The biggest issue with the adaptive controller algorithm, is that the sensor signal has to be in steady-state when the gradient estimation is made. Otherwise the estimation will be faulty and the behavior of the controller algorithm might deteriorate. The steady state algorithm shown in Section 5.2.5 seems promising and it manages to

prevent the controller from giving a control signal until the output signal is in steady state.

5.3.1 Controller Goals Achievement Summary

In Table 5.2, a summary of the controller goals and their fulfillment for each of the experimental controller are shown. Here it is more clear that the adaptive controller algorithm is the best choice, because it fulfills most of the controller goals. The only one that is not sure is the 7th and last controller goal. Probably more research has to be done before a steady-state algorithm can be reliable enough to use for this purpose.

Controller goal	I-control	Feed-forward plus feedback	Adaptive FF + FB control
Nr. 1	x	x	x
Nr. 2	x	x	x
Nr. 3	-	-	x
Nr. 4	-	x*	x*
Nr. 5	x	x	x
Nr. 6	x	x	x
Nr. 7	x	x*	x*

Table 5.2: *Table showing the the different controllers tried in the experiments and wether or not the controller goals from Section 2.5.1, are fulfilled.*

The $*$ means that they are fulfilled if the steady-state detection algorithm, explained in Section 5.2.5 is used.

Chapter 6

Simulation of MIMO Controllers

In this chapter, MIMO controller algorithms are tested and evaluated on a simulation model of the roller mil. The simulations are carried out with two different model parameter selections, in order to see what influence it has on the controllers.

6.1 Preparative Steps

Before MIMO simulations of the controller algorithms are possible the following steps need to be explained.

6.1.1 Normalization

The inputs to the MIMO controller have been normalized between 0 and 1. This is done in order to keep the controller algorithm as general as possible, and to make sure that parts of the controller algorithm do not have to be rewritten if the intervals of the control inputs change. The normalized parameters are shown in Table 6.1.

Parameter	Raw value	Normalized value	Description
Roller gap	533 - 600	0 - 1	The distance between the rollers
Differential speed	2 - 4	0 - 1	The differential speed between the rollers

Table 6.1: *Decription of how the control variables have been normalized*

6.1.2 Derivation of the MIMO Model

In Section 4.5, an experiment was carried out on the roller mill. This experiment indicated the relationship between the input signals; roller gap, differential speed, and the output signal; fall through. Although this experiment showed that the

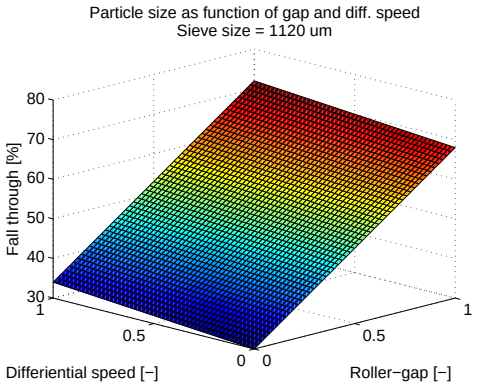

Figure 6.1: *How the roller gap and the differential speed affect the fall through at the sieve size 1120 μm*

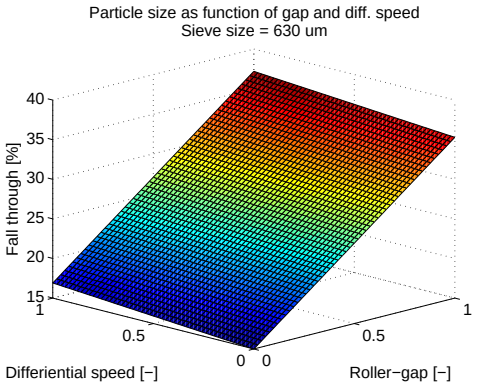

Figure 6.2: *How the roller gap and the differential speed affect the fall through at the sieve size 630 μm*

differential speed did not have much influence on the output signal, this experiment result was used as starting point for the models used in the remaining of this chapter.

The inputs and outputs from the experiment was put into an LMS estimation algorithm, as explained in Section 3.3, where the system to estimate is expressed as

$$\begin{aligned} y_1(k) &= k_1 u_1(k) + k_2 u_2(k) + m_1 \\ y_2(k) &= k_3 u_1(k) + k_3 u_2(k) + m_2 \,. \end{aligned} \tag{6.1}$$

As seen, the equations show how both the outputs relate to the inputs and also that the equations are linear. After letting the LMS algorithm estimate the parameters k_1, k_2, k_3 and k_4, the model is written as

$$\begin{aligned} y_1(k) &= 38 u_1(k) + 4 u_2(k) + 21.9 \\ y_2(k) &= 20.3 u_1(k) + 1.9 u_2(k) + 9.5 \,. \end{aligned} \tag{6.2}$$

Looking at these equations, it is clear that the first input, the roller gap has much more influence on both output 1 and output 2. This can also be confirmed by plotting 3D-plots of the functions from Equation 6.1, as seen in Figure 6.1 and 6.2

6.2 The Test Models

The model derived in equation 6.2 is almost impossible to apply automatic control to. This is because the second control parameter, the differential speed has so little influence on the output signal. This means that one control parameter has to be able to control two output signals, which in most case is impossible, unless one is very lucky when choosing the set-points. Therefore the model was modified in two different ways, and the the controller algorithms where simulated on the new models. Below, the two different test models are presented.

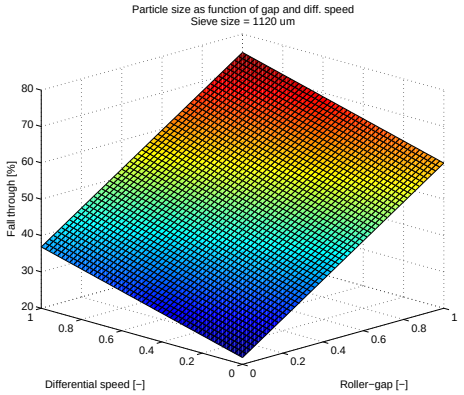

Figure 6.3: *The fall through at sieve size 1120 µm is shown as a function of the differential speed and roller gap. The differential speed has more influence on the grinding result, but the variables are still very coupled.*

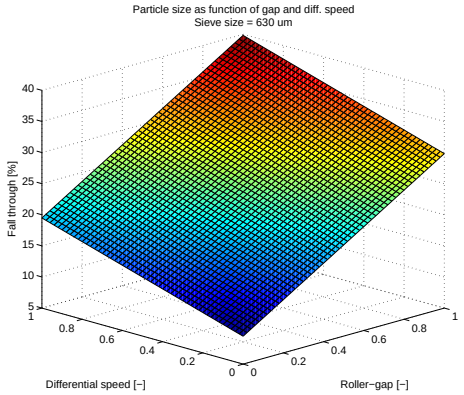

Figure 6.4: *The fall through at sieve size 630 µm is shown as a function of the differential speed and roller gap. The differential speed has more influence on the grinding result, but the variables are still very coupled.*

6.2.1 Model 1 - More Influence from Differential Speed

In this model, the differential speeds influence on the output signals have been increased by assigning higher values to k_2 and k_4. The new model is written as

$$\begin{aligned} y_1(k) &= 38u_1(k) + 15u_2(k) + 21.9 \\ y_2(k) &= 20.3u_1(k) + 10u_2(k) + 9.5\,, \end{aligned} \tag{6.3}$$

where as seen, an output signal change, now is influenced both by the roller gap and the differential speed. The result of the change can be seen in Figure 6.3 and Figure 6.4. Even though both functions seem to be affected by both the roller gap and differential speed, it is still not trivial to apply automatic control. This is because the parameters are very coupled, i.e. it is not possible to take the one parameter to control one output, and the other parameter to control the other output.

6.2.2 Model 2 - Control Variables less Coupled

In this model, the parameters have been chosen in order to be more decoupled. In order to achieve this, the functions have been changed as in Equation 6.4.

$$\begin{aligned} y_1(k) &= 38u_1(k) + 6u_2(k) + 21.9 \\ y_2(k) &= 4u_1(k) + 15u_2(k) + 9.5 \end{aligned} \tag{6.4}$$

As seen, k_2 and k_3 are much lower and k_1 and k_4 higher. This means that the roller gap has much influence on output 1 but not much influence on output 2. The opposite goes for the differential speed, meaning that it has much influence on output 2 but not much influence on output 1. This is illustrated more clearly in Figure 6.5 and Figure 6.6. This model is less coupled meaning that is should be easier for an automatic controller to control the system.

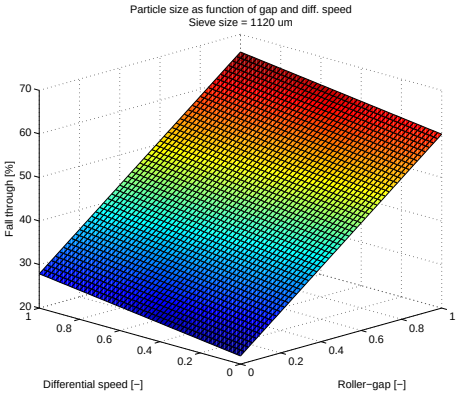

Figure 6.5: *The fall through at sieve size 1120 µm is shown as a function of the differential speed and roller gap. The roller gap has a big influence on the output signal and the differential speed has little influence on the output signal.*

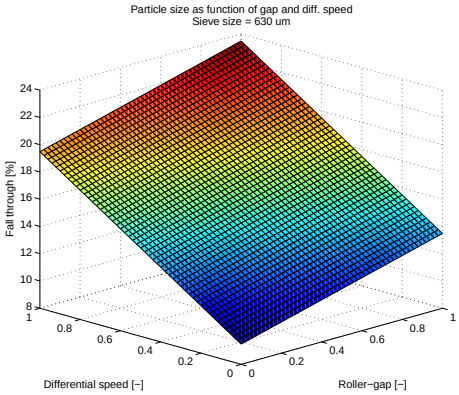

Figure 6.6: *The fall through at sieve size 630 µm is shown as a function of the differential speed and roller gap. The roller gap has little influence on the output signal and the differential speed has a big influence on the output signal.*

6.3 Simulation Results of the MIMO Controllers

Two different controllers where tested in this section, first a simple PID controller and then the more advanced controller which finds the optimal control signal changes in each sample.

6.3.1 MIMO PID Controller

As a first attempt to control the process, the incremental I-controller was chosen, as explained in Section 3.2.2. With this simple approach to the problem, the system is assumed to be decoupled enough to let one controller control the roller gap and the other controller control the differential speed. The control laws are

$$\begin{aligned} u_1(k+1) &= u_1(k) + K_{i,1}e(k) \\ u_2(k+1) &= u_2(k) + K_{i,2}e(k)\,. \end{aligned} \tag{6.5}$$

The first simulation is done with model 2 from Section 6.2.2. The fact that the variables are quite decoupled from each other, should mean that it should be possible to control with the simple PID controller structure shown in Equation 6.5. The controller parameters, $K_{i,1}$ and $K_{i,2}$, where both chosen to 0.01. The result of the simulation is shown in Figure 6.7 and Figure 6.8. As seen the controller manages to track the set-point changes in both of the output signals. For the simple case, when the control variables are enough decoupled, it seems that the MIMO PID controller is a good choice.

Worth noticing is that with this controller structure, the two controllers work independently of each other. This means that there can be unwanted behavior if the variables are not decoupled enough. This "unwanted" behavior is shown in the next simulation.

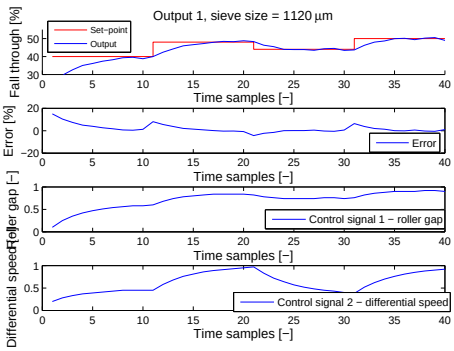

Figure 6.7: *Simulation with MIMO PID controllerand model 2, showing output 1, the error signal and the two control signals.*

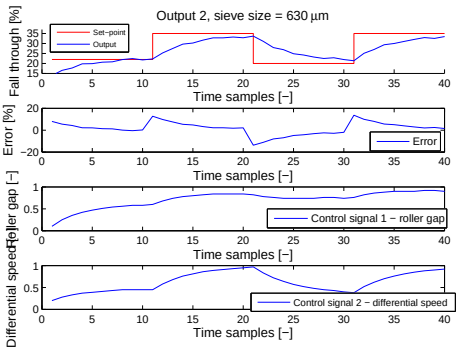

Figure 6.8: *Simulation with MIMO PID controller and model 2, showing output 2, the error signal and the two control signals.*

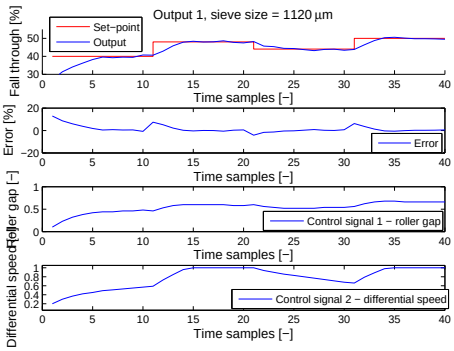

Figure 6.9: *Simulation with MIMO PID controller and model 1, showing output 1, the error signal and the two control signals.*

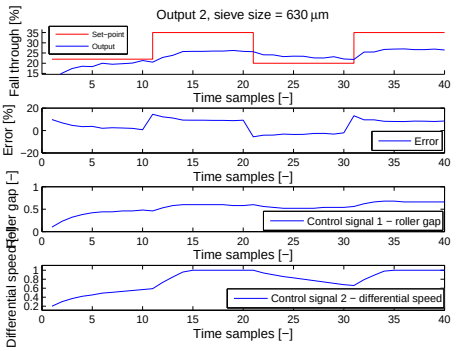

Figure 6.10: *Simulation with MIMO PID controller and model 1, showing output 2, the error signal and the two control signals.*

The second test for the PID controller was to do a simulation using model 1, from Section 6.2.1. Here the control inputs are more coupled, and can therefore seen as a more difficult system to control. The controller parameters, $K_{i,1}$ and $K_{i,2}$, where both chosen to 0.01 as previously. The simulation result is shown in Figure 6.9 and Figure 6.10. As seen, this time the controller only manages to track the set-point changes in output 1, but not output 2. The reason is that u_1 has more influence on y_1, than what u_2 has on y_2. Since both the controller gains, K_i are set to equal, the first output therefore "wins". The solution to this problem would be to increase $K_{i,2}$. However, it is not trivial to know how much to increase the gain, hence the need for the more advanced controller explained in Section 6.3.2.

6.3.2 MIMO Optimal Controller

In this section, the problems that occured with the MIMO PID controller from Section 6.3.1. The controller that was chosen was the optimal controller, explained in Section 3.5. Only model 1 from section 6.2.1 is being simulated here, since for

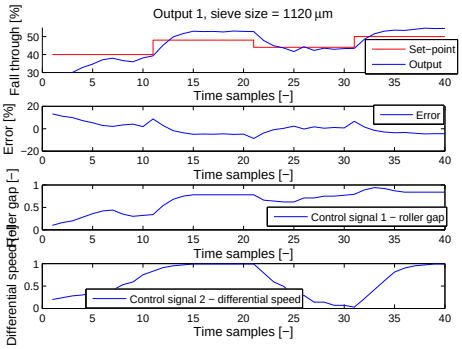

Figure 6.11: *Simulation with optimal MIMO controller and model 1, showing output 1, the error signal and the two control signals.*

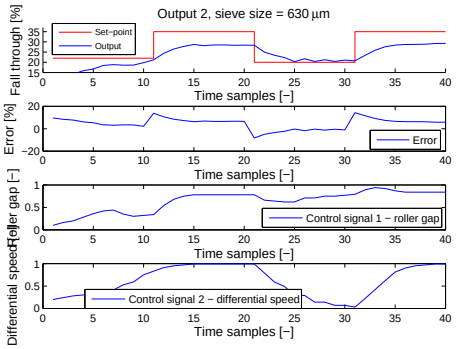

Figure 6.12: *Simulation with optimal MIMO controller and model 1, showing output 2, the error signal and the two control signals.*

model 2, the MIMO PID controller would be enough.

In Figure 6.11 and Figure 6.12 the result of a simulation with the MIMO optimal controller is shown. As seen, the optimal controller performs much better in the sense that a compromise between the two output errors was reached. This is because the controller takes into account the couplings between the control variables and calculates the control signals that minimizes the combined least error of the two outputs.

An interesting aspect with the optimal controller, is that it is possible to weight the importance of the different outputs. For example, if output 1 is more important than output 2, then it is possible to tell this to the controller using the γ parameter. In the previous simulation, γ was set to 0.5, meaning that both outputs are equally important. In the simulation below, γ was set to 0.1 to tell the controller to care more about reducing error created by output signal 2 than the error created by output signal 1. The result of this simulation is shown in Figure 6.13 and Figure 6.14. As seen, this time the controller reduces error two, but on with the downside that error one becomes bigger.

6.4 Comparison of MIMO Controllers

From the simulation results presented above, it is clear that the MIMO optimal controller performs better when the system is complex. The big advantage is that it is possible to tell the controller which output signal it should emphasize to minimize.

If the system contains no or small cross couplings between the control variables, the MIMO PID controller is probably the better choice. The PID controller is a much simpler controller both when it comes to the number of tuning parameters and the complexity of the computer implementation.

Before deciding which controller to implement on the real roller mill, it has to be investigated how the cross couplings between the variables look. If it turns out that the differential speed does have great influence of the ash content of the wheat (a potential second process output), and at the same time the roller gap has small

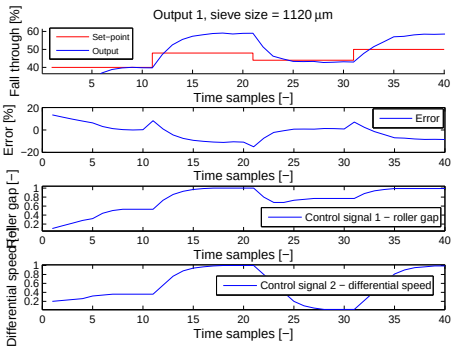

Figure 6.13: *Simulation with optimal MIMO controller and model 1, showing output 1, the error signal and the two control signals.*

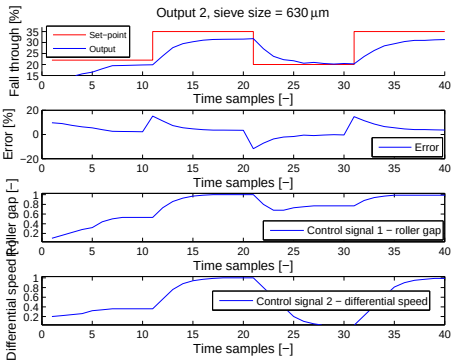

Figure 6.14: *Simulation with optimal MIMO controller and model 1, showing output 2, the error signal and the two control signals.*

influence of the ash content, the MIMO PID controller is probably the best choice.

Chapter 7

Implementation of SISO Controllers on a real Roller Mill

In this chapter, some of the SISO controller algorithms from Chapter 5, have been tested on the real roller mill experiment machine. First the experimental setup is explained and then the experiment results are showed. Finally the advantages and disadvantages of the different controllers are compared.

7.1 The Experimental Setup

The different controller algorithms have been tested on a real experimental setup. This setup consists of a roller mill machine, a particle size sensor and the software/hardware interface needed to send control signals to the system and to read the output from the particle size sensor. The controller algorithms are implemented as MATLAB functions. The MATLAB function is called from a LabVIEW interface where a graphical interface and all the necessary low-level connections are made.

7.2 Differences between Simulations and Experiments

Implementing the controllers in the real roller mill had a few complications. Probably the most limiting factor was the experiment time. Since it takes approx. 2-3 minutes for the sensor to show a new steady state value when the roller gap is changed, an experiment where the reference signal changes a couple of times can easily take 10-15 minutes, This means that the experiments has to be planned very carefully before being carried out.

Also a considerable amount of wheat is required which is both expensive and requires that the wheat bin is continuously refilled during the experiments. The third problem was the sensor, which from time to time can give some very noisy output signals which causes difficulties for the controller algorithms.

All these things together limited the amount of controller algorithms that could be tried on the real system. Instead of testing a very advanced controller, it was divided into smaller controllers, where different parts were tested in each experiment.

7.3 Experiment Results of the Implemented Controllers

In this section, the controller algorithms that have been implemented on the roller mill machine are presented.

7.3.1 Incremental I-Control

The very simple structure and few tuning parameters, made the incremental I-controller the best candidate for the first experiments on the real roller mill. The goal was to see if the results from the simulations in Section 5.2.1, and the results from the experiments would match. Therefore, the same parameters as the controller simulation in Section 5.2.1 where chosen. The parameters where chosen to: $K_i = 0.5$, $N = 65000$, and $T_s = 30$ s.

N was chosen quite low, because otherwise the sensor output would react too slowly, i.e. the delay before the sensor starts to show a descending or ascending trend is too big. The result from the experiment with the I-controller is shown in Figure 7.1.

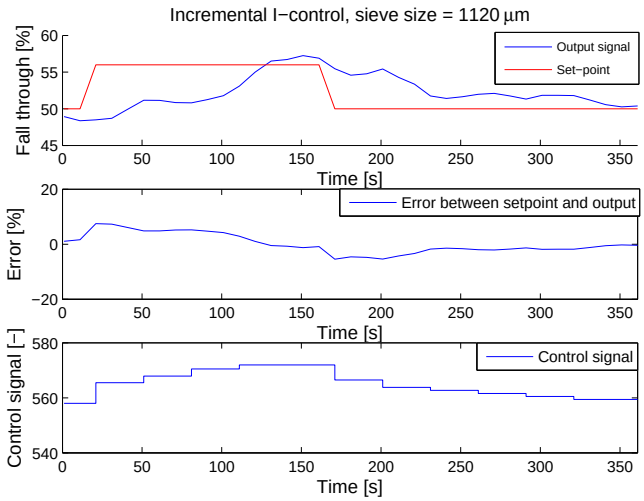

Figure 7.1: *Experiment result using an I-controller.*

As seen, the controller manages to track a set-point change from 57 % to 50 %. The outputs signal is quite noisy but the low-pass filter helps in removing the higher frequencies of the noise. However, low frequency noise is still present. What makes this controller a little bit dangerous, is that if unlucky, the sensor can show unwanted behavior. This can be seen Figure 7.2, which shows a second experiment with the same parameter choice.

As seen when the reference changes from 60 % to 50 %, the output signal goes down as would be expected but then it goes up again. This causes the controller to make another big control change, with the result that the output signal makes an overshoot. The controller then compensates for this overshoot by increasing the control signal again. This experiment shows a potential danger with this controller type. However choosing a small enough controller gain, K_i, should prevent this from happening. This will of course also make the controller slower, which is not desired.

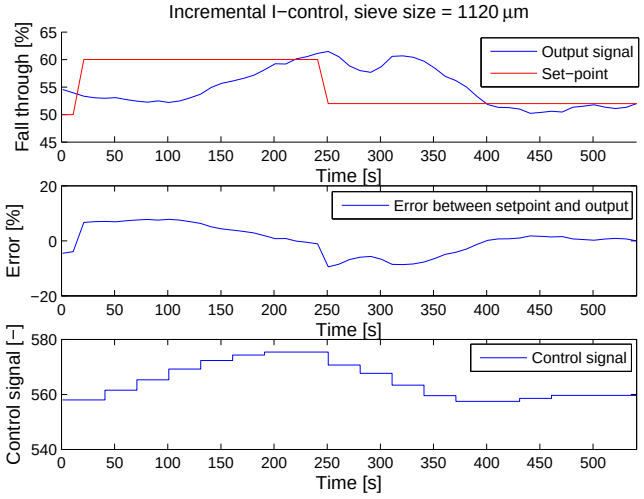

Figure 7.2: *Second experiment result using an I-controller.*

7.3.2 Static Feed-Forward plus Feedback Control

The second controller that was implemented, was the static feed-forward combined with a feedback controller. This controller was also tested in simulations in Section 5.2.3. The parameters for the experiment where chosen to: $K_{FF} = 1.7$, $K_{FB} = 0.5$, $N = 130000$, and $T_s = 130$ s.

Figure 7.3 shows the result of the experiment. As seen, the controller manages to reach a new set-point in approx. 200 seconds usually. However, when the set-point changes from 60 to 50, it takes the controller approx. 300 seconds to reach the new set-point. When the reference changes from 54 % 46 %, at $t = 620$ s, the controller manages to reach the set-point in approx. 100 seconds, which is very fast compared to the other set-point changes. The explanation could be that the set-point was changed before the sensor showed a steady-state value, and therefore did not show the "true" value.

7.3.3 Adaptive feed-forward control

Changing wheat temperature and moisture have effect on how big the feed-forward gain should be. The adaptive I-controller, explained in Section 3.4 was chosen to solve this problem. The goal was to estimate the level of the feed-forward gain by using the LMS parameter estimation technique.

To make the experiment easier and shorter, no feedback control was used. When the controller had made one control signal, the reference signal was changed, no matter if there still was an error between the reference and the output. The parameters where: $K_{FF} = 1.5$, $N = 130000$, and $T_s = 140$ s.

The result of the experiment is shown in Figure 7.4. During the initial phase, the output does not reach the set-point. During this time, the controller uses the initial value of $K_{FF} = 1.5$. After three control signal changes, the LMS algorithm has gathered enough information to estimate $\hat{J}$, the gain of the system. $\hat{J}$ is then used in the remaining part of the experiment to update K_{FF}, the controller parameter

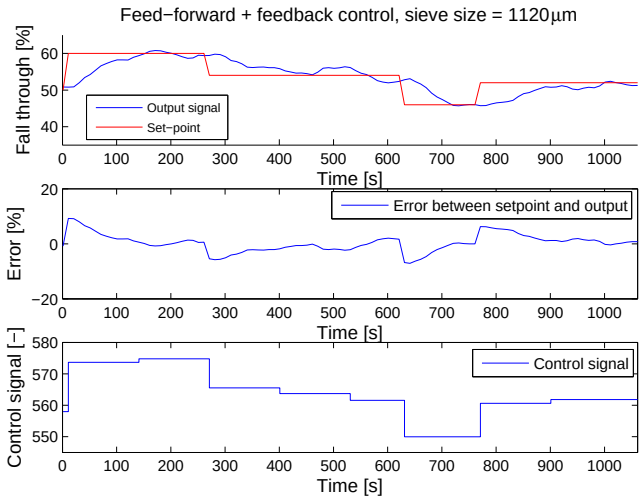

Figure 7.3: *Experiment result using a static feed-forward combined with a feedback-controller*

($K_{FF} = \frac{1}{J}$). As seen in the figure, the output tracks the set-point change much better in the end, when the updated K_{FF} is used.

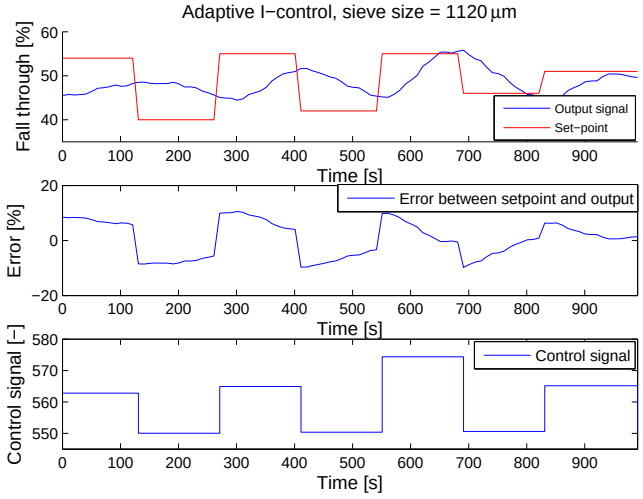

Figure 7.4: *Experiment result using an adaptive controller. The controller estimates $\hat{J}$, the gain of the system, and then uses this estimation to achieve better control.*

Table 7.1 shows the gradient estimated during the experiment run and the corresponding control parameter, K_{FF} that was chosen.

7.3.4 Steady State Detection Algorithm

The sampling time of the controller can be hard to determine, as explained in Section 2.2.2. The speed of the sensor is determined by how fast it can measure the particles that flow through the sensor.

The following algorithm aims at solving the problem, by detecting when the system

$\hat{J}$	K_{FF} $(= \frac{1}{J})$
0.4092	2.4438
0.4133	2.4195
0.4382	2.2821

Table 7.1: *The estimated gradients and K_{FF}, the corresponding control parameter.*

is in steady state before giving a control signal. How the steady state algorithm works is explained more thorough in Section 3.7. The parameters chosen for the experiment where: $K_{FF} = 1.5$, $K_{FB} = 0.5$, and $N = 130000$. The result from an experiment with the steady state detection algorithm is shown in Figure 7.5.

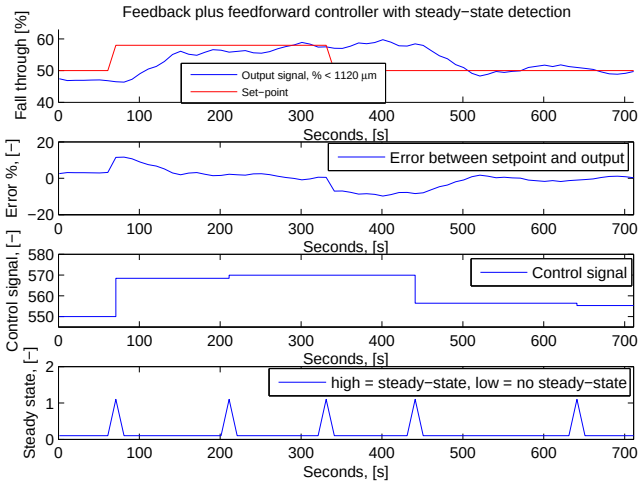

Figure 7.5: *Experiment result using a feed-forward combined with feedback-controller with steady state detection algorithm.*

As seen, now the controller only gives a control signal, when the fourth subplot shows a peak. No control signal is given when the output signal is changing. It is also seen that the peaks are not appearing with even intervals, meaning that sometimes the sensor reacts faster than other times.

The algorithms is however very sensitive in parameter selection, especially the $e_{threshold}$-parameter (see Section 3.7). The parameter is difficult to choose, and choosing it wrongly can result in a different behavior that desired. The fact that the measurement noise of the sensor can vary, also means that the parameters of the steady-state detection algorithm has to be re-tuned.

7.4 Discussion of the Implementation Results

In the experiments conducted on the real experiment machine, it was even more obvious what could happen when using the incremental I-controller. Results from Figure 7.2, shows that there indeed is a risk of oscillations in the output signal, and that by choosing K_i too big, there is a risk of the system becoming unstable. Worth noticing is that longer sampling time is not possible to use for the pure feedback

controller, since it then would take too much time to reach a new set-point. The I-controller might also require re-tuning which is not desired.

The feed-forward plus feedback controller from Figure 7.3, seems to be a better algorithm compared to the pure feedback controller. This is mainly because of its longer sampling time, which gives that the controller only send a control signal when the sensor is in steady state, i.e. the output signal shows no more ascending or descending behavior. Also this controller might require re-tuning of the controller parameters.

As seen in 7.4, the adaptive algorithm also worked on the real roller mill, despite the large amounts of measurement noise from the sensor. The adaptive feed-forward could also in the future be combined with a feedback controller, similar to the simulations in Section 5.2.4, in order to remove the steady state error that the adaptive feed-forward could not remove. The adaptive controller is also the only controller that does not require any manual re-tuning of controller parameters, which was a required controller goal from Section 2.5.1.

There is a risk of bad gradient estimation with high sensor noise values. If the sensor can be improved to give less noisy signals, this should not be a problem though. One option would be to use some kind of monitoring algorithm which discards the gradient estimate if too much noise or is present.

The steady state algorithm shown in Figure 7.5 seems promising and it manages to prevent the controller from giving a control signal until the output signal is in steady state. However, considering that the algorithm is rather difficult to tune and the fact that it is rather slow, concludes that more research on the subject has to be done. Possibly also other steady state algorithms should in the future be investigated.

Chapter 8

Conclusions and Future Work

In this chapter, the results are discussed and suggestions for future work are made. The results are discussed from the view-point of the controller goals and thesis objectives.

8.1 Conclusions

In this report different controller algorithms used for automatic control of a roller mill have been tested. Controllers have been implemented successfully, both in simulations and experiments on a real roller mill.

Initial experiments carried out on the roller mill, showed that the differential speed between the rollers, did not have enough influence of the grinding process. This made it impossible to use the differential speed as second control parameter and that the implementation of controller algorithms on roller mill, where limited to the SISO case. The experiments also showed that the relation between the roller gap and the outgoing particle size distribution could be approximated as linear.

The I-controller is a simple controller, but because of the nature of the controller it is not 100 % that the controller will not cause unstable behavior of the machine. The I-control algorithm also might require re-tuning when e.g. a new wheat type is put into the machine, or the moisture level of the wheat is changed.

The most successful algorithm was the adaptive feed-forward plus feedback controller, which showed to be promising in simulations and also to work well on the real experiment machine. This controller fulfils the controller goals stated in Chapter 2. The main advantage of the adaptive controller, is that it does not require any manual tuning of the controller parameters.

The steady-state detection algorithm shows promising results, but because of the fact that it is relatively difficult to tune the parameters, more research in the area needs to be done. It might be that other steady-state algorithms are better suited, than the algorithm used in this project.

No MIMO controller was implemented on the real system, since no second variable was found that had enough influence on the particle size distribution. In simulations although, some some algorithms have been tested. The proposed algorithm is the optimal control algorithm, which calculates the optimal control change in each

sample with the help of an estimated model of the system. The method is very flexible and can, both weigh the importance of the different outputs, and also handle constraints on the control signals. If the system turns out to be enough decoupled, the MIMO PID controller probably is the better choice, considering its simplicity and easiness to implement.

8.2 Proposal for Future Work

In this project, a first attempt to apply automatic control to a roller mill has been taken. The next step would be to take the controller algorithm proposed and do more refinements and more thorough testing on the roller mill.

The next step would also be to try the algorithm proposed for the MIMO system on the experiment machine. This would of course require that new outputs can be measured, such as e.g. the ash content of wheat.

To be able to make longer experiments on the roller mill, an automatic refilling of the bin that contains the wheat, should replace the manual refilling process used currently.

One could also imagine that the different feed-forward gains used by the adaptive controller could be saved in a database. When changing the type of wheat or moisture level of the wheat, the database could be used to give an initial feed-forward gain to the controller. This initial gain could then be improved further by the adaptive controller during the run.

Improving the particle size sensor would also improve the automatic control. A faster sensor would make it possible for the controller algorithm to faster reach a new set-point. A more accurate sensor giving a less noisy output signal, would also improve the controller

Another task would also be to implement the controller algorithm the target hardware, e.g. microcontroller, microprocessor, DSP etc. and aim to implement automatic control on future commercial roller mills.

Bibliography

D.A.Pierre (1999). Pid-control. pp 446–456. John Wiley & Sons.

Lennartson, B. (Ed.) (2000). *Reglerteknikens grunder*. Studentlitteratur. 4th edn. Studentlitteratur. Lund.

Ljung, L. and Glad, T. (Eds.) (2004). *Modellbygge och simulering*. 2nd edn. Studentlitteratur. Sweden.

NAM (2008). North american millers' association, internet homepage. [Access date: 2008-07-22] `http://www.namamillers.org/ci_products_wheat_mill.html`.

Roover, D.de, Kosut, R.L, Emami-Naeini, A. and Ebert, J.L. (1998). Run-to-run control of static systems. *IEEE conference on decision and control, Florida*, **37th edition**, 695–700.

T.Larsson, L.Ma and D.Filev (2000). Adaptive control of a static multiple input multiple output system. *Proceedings of the american control conference*, **62**(4b), 2573–2577.

Wittenmark, B. and Åström, K.J. (Eds.) (1995). *Adaptive control*. Addison-Wesley series in electrical engineering. control engineering. 2nd edn.

Wittenmark, B. and Åström, K.J. (Eds.) (1997). *Computer Controlled Systems*. 3rd edn. Prentice-Hall. USA.

Appendix A

Translation between Control Units

Here a table for translating between the different control units are shown.

The traditional way of tuning the gap of a roller mill is by setting a gap in "hours". This is a unit that has 60 as a base instead of 100 as normal and gives the head-miller a intuitively understanding for how big or small the roller gap is. The range of this unit from 05:20 o'clock to 06:00 o'clock. In a computer controller system the base 60 does not work so well, and therefore a control unit similar to the hours, but with 100 as base is available. This unit is completely artificial and ranges from 533 [-] to 600 [-]. With this unit, an increase in the control signal will mean an increase in the sensor output, which is the way a control system usually works. With the roller gap in μm as control signal, an increase in control signal would give a decrease in the sensor output, which intuitively feels wrong.

Below is a table which contains a translation between the different units.

Hours [time]	Control signal [-]	Roller gap [μm]
520	533.3	850.0
521	535.0	837.5
522	536.7	825.0
523	538.3	812.5
524	540.0	800.0
525	541.7	787.5
526	543.3	775.0
527	545.0	762.5
528	546.7	750.0
529	548.3	737.5
530	550.0	725.0
531	551.7	712.5
532	553.3	700.0
533	555.0	687.5
534	556.7	675.0
535	558.3	662.5
536	560.0	650.0

537	561.7	637.5
538	563.3	625.0
539	565.0	612.5
540	566.7	600.0
541	568.3	587.5
542	570.0	575.0
543	571.7	562.5
544	573.3	550.0
545	575.0	537.5
546	576.7	525.0
547	578.3	512.5
548	580.0	500.0
549	581.7	487.5
550	583.3	475.0
551	585.0	462.5
552	586.7	450.0
553	588.3	437.5
554	590.0	425.0
555	591.7	412.5
556	593.3	400.0
557	595.0	387.5
558	596.7	375.0
559	598.3	362.5
600	600.0	350.0

Appendix B

The Simulation Model

The simulation framework is built in entirely in the MATLAB programming language and has been built such that it is easy to create new controller algorithms and to test them on the roller mill model. The simulation framework therefore consists of two parts. The model-script where the model and the necessary initializations are made, and several different controller functions that are called by the model-script. This approach also emulates the way the controller algorithms are implemented on the real roller mill experiment machine.

The simulation can be started by in the MATLAB command window writing:
▷▷ *start_simulation*

The user is now presented with a series of choices, which all can affect the result of the simulation.

1. ▷▷ *Choose controller type*
   Here the user can choose between the available controller algorithms. The correct input is a number between 1 and forward.

2. ▷▷ *Enter a sieve size*
   The user is can here choose a number of different sieves, which emulates the way the roller mill experiment setup works. The correct input is a number between 1 and forward.

3. ▷▷ *Enter a reference signal vector*
   Here the user defines a reference signal that the controller will try to follow. The correct input is a vector within brackets, e.g. [50 40 60 55].

4. ▷▷ *Enter the simulation length*
   A simulation time in seconds is to be given here. The correct input is a scalar.

5. ▷▷ *Enter how many particles the sensor should measure*
   Here the user can choose the amount of measurement noise in the simulation, which is represented by the number of particles the sensor should measure. The lower the value is set, the more measurement noise will be present. It also affects the time constant of the model, e.g. a low value means that the model reacts faster to a control signal change than a big value. This is to emulate the way the sensor is tuned in the real roller mill experiment machine. The correct input is a scalar between 40 000 and 130 000.

6. ▷▷ *Enter the type of wheat*
   The last input is the type of wheat, which can be chosen to be normal, very dry or very wet. This parameter affects the gain of the model, i.e. how much a change in input affects the output. This parameter is mainly for testing the adaptive controller. The correct input is a value between 1 and forward.