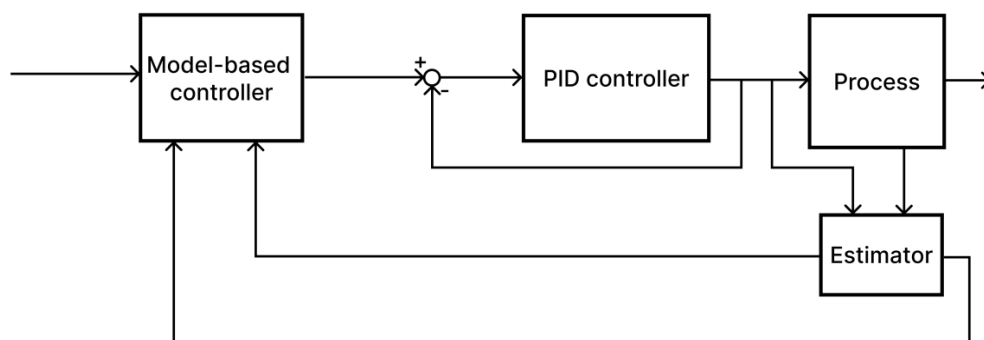


FMH606 Master's Thesis 2024
Industrial IT and Automation

Model-based control for optimal polymer addition to the drum screens in a wastewater treatment plant



Torbjørn Mørk Madsen

15. May 2024

Faculty of Technology, Natural sciences and Maritime Sciences
Campus Porsgrunn

Course: FMH606 Master's Thesis, 2024

Title: Model-based control for optimal polymer addition to the drum screens in a wastewater treatment plant

Number of pages: 85

Keywords: WWTP, MPC, Modeling, Dry Solids, Drum Screen, Python, Profibus

Student: Torbjørn Mørk Madsen

Supervisor: Finn Aakre Haugen

External partner: Veas

Summary:

This study examines the process of optimizing the amount of polymer added to the sludge that goes into the drum screens at Veas, which is the biggest wastewater treatment plant in Norway. The goal is to reduce the amount of polymer used while still achieving efficient separation of sludge and water. In order to address the absence of direct dry solids measurements at the drum screens, a comprehensive two-week sampling period was carried out to gather crucial data for subsequent analysis.

Three modeling methodologies were investigated in order to estimate the dry solids content and enable the implementation of Model Predictive Control (MPC) in polymer dosing. The approaches consisted of a linear model combining time delay and time constant, a non-linear model obtained through system identification, and a mechanistic model founded on mass balance principles. After a thorough evaluation, the non-linear model was chosen for both estimating dry solids and integrating with MPC.

The study highlights the difficulties and restrictions linked to the implementation of MPC, which requires continuous testing and improvement. Furthermore, a connection is established between the created model and MPC, which is implemented using Python programming language, in order to facilitate communication with the Programmable Logic Controller (PLC) through the Profibus protocol.

Future work will involve integrating a Kalman filter to enhance the accuracy of estimating dry solids and further enhance the efficiency of the process. This study makes a valuable contribution to the improvement of wastewater treatment methods, which has the potential to increase efficiency in operations and promote environmental sustainability.

ACKNOWLEDGEMENTS

I am truly grateful to Veas for giving me the chance to work together on this master's thesis. This marks the conclusion of my three years as a master's student specializing in industrial automation and IT. During that time, I worked fifty percent a year at Veas and 66% of the time at USN to complete my master's degree.


A special thanks goes to Morten Rostad Haugen, a Senior Process Engineer at Veas, for his invaluable mentorship and assistance in enhancing my understanding of the process and proofreading this thesis. I would like to express my gratitude to Kjell Rune Jonassen, a Development Engineer at Veas, and Jonas Pettersen, a Senior Development Engineer at RVM systems, for their valuable support in helping me define and identify a research topic that closely matched the challenges encountered by Veas. Their contribution served as the basis for my master's thesis. I would like to express my appreciation to Alex Tvedt, an apprentice automation technician at Veas, and Joachim Holdhus, an automation technician at Veas, for their assistance in collecting samples for laboratory testing.

Finally, I would like to express my gratitude to Finn Aakre Haugen from USN for his invaluable mentorship and expertise throughout the duration of this master's thesis.

It is important to mention that although I received aid from OpenAI and QuillBot for formulating and proofreading, the majority of the content and concepts in this master's thesis are my original work. I made strategic use of OpenAI's support, ensuring that my ideas and contribution remained the main focus throughout the writing process.

GitHub has been utilized for easy access to all the codes used in this project. The appendix contains a hyperlink to the github repository.

Through this project, I have gained invaluable knowledge and skills that I will apply in my future career.



Torbjørn Mørk Madsen

CONTENTS

| | | |
|-------|---|----|
| 1 | Introduction | 1 |
| 2 | System description | 2 |
| 2.1 | Water treatment | 2 |
| 2.2 | Sludge treatment | 3 |
| 2.2.1 | Drum Screens | 3 |
| 2.3 | Automation at Veas | 5 |
| 3 | Theory | 6 |
| 3.1 | SCADA | 6 |
| 3.1.1 | PLC | 6 |
| 3.1.2 | Revolution Pi | 7 |
| 3.1.3 | Profibus | 7 |
| 3.1.4 | OPC DA | 7 |
| 3.1.5 | Linea | 8 |
| 3.2 | Unique measurement units | 8 |
| 3.2.1 | Dry solids | 8 |
| 3.2.2 | Loss on ignition | 9 |
| 3.2.3 | Turbidity | 9 |
| 3.3 | Singular value decomposition | 9 |
| 3.4 | Principal component analysis | 9 |
| 3.4.1 | How to calculate principal components | 9 |
| 3.4.2 | Loadings | 10 |
| 3.5 | Proportional, integral, and derivative controller | 10 |
| 3.6 | Optimal control system (model predictive controller) | 10 |
| 3.6.1 | Linear MPC | 11 |
| 3.6.2 | Non-linear MPC | 13 |
| 3.7 | System identification | 14 |
| 3.7.1 | Library for system identification | 15 |
| 3.7.2 | Akaike's information criterion | 15 |
| 3.7.3 | Forward regression orthogonal least squares | 15 |
| 3.8 | Mass balance | 16 |
| 4 | Methods | 17 |
| 4.1 | Getting data | 18 |
| 4.1.1 | Data gathered from laboratory analysis | 18 |
| 4.1.2 | Parameters gathered from historical data | 18 |
| 4.2 | Naming the dry solids variable for use in SCADA system | 18 |
| 4.3 | Principal component analysis | 19 |
| 4.3.1 | Dimensionality reduction using PCA | 19 |
| 4.4 | Interpolations of the laboratory results | 19 |
| 4.5 | Model creation | 23 |
| 4.5.1 | NARX model | 23 |
| 4.5.2 | Construction of a linear model with time delay and time constant. | 25 |
| 4.5.3 | Usage of mass balance | 26 |
| 4.6 | Finding the time delay for the process | 27 |
| 4.6.1 | Time delay of flocculation tank into the drum screen | 27 |
| 4.6.2 | Time delay of dry solids into the drum screen | 28 |
| 4.6.3 | Time delay of Turbidity into the drum screen | 28 |
| 4.7 | MPC creation | 29 |
| 4.7.1 | MPC program | 29 |
| 4.8 | Platform for running estimation and MPC | 31 |
| 4.8.1 | Setup | 31 |
| 4.8.2 | Access of Profibus in Python | 32 |
| 4.8.3 | Estimator | 33 |
| 4.8.4 | MPC | 33 |

| | | |
|-------|---|----|
| 4.8.5 | Alarms | 34 |
| 4.8.6 | Sequence diagram | 34 |
| 5 | Results and Discussion | 35 |
| 5.1 | Laboratory sample results | 35 |
| 5.2 | Principal component analysis | 36 |
| 5.2.1 | Dimensionality reduction using PCA | 36 |
| 5.3 | Interpolations results | 37 |
| 5.4 | Model Creation | 39 |
| 5.4.1 | NARX model | 39 |
| 5.4.2 | ARX model | 42 |
| 5.4.3 | Mass balance model | 42 |
| 5.4.4 | Time delay | 43 |
| 5.4.5 | Mixer in tank | 43 |
| 5.4.6 | Testing models on the real system | 44 |
| 5.5 | MPC results | 45 |
| 5.5.1 | Setup of the control problem | 45 |
| 5.5.2 | Choosing model | 46 |
| 5.5.3 | Simulation | 46 |
| 5.5.4 | Testing MPC on the real system | 48 |
| 5.5.5 | Closing the loop and let the MPC dosage polymer | 50 |
| 6 | Further discussions | 53 |
| 6.1 | Interpolation | 53 |
| 6.2 | Comparing results from dimensionality reduction using PCA and number of inputs used in NARX model | 53 |
| 6.3 | Model | 53 |
| 6.3.1 | Time delay | 54 |
| 6.3.2 | NARX Model Limitations | 54 |
| 6.3.3 | Washing of the drum screen | 54 |
| 6.3.4 | TAG names for models | 55 |
| 6.4 | MPC | 55 |
| 6.4.1 | MPC during periods of poor sludge settling | 56 |
| 7 | Future work | 57 |
| 7.1 | Kalman filter | 57 |
| 7.2 | Model | 57 |
| 7.3 | MPC | 58 |
| 8 | Conclusion | 59 |
| | Appendix | 63 |

LIST OF FIGURES

| | | |
|-------------|--|----|
| Figure 2.1 | Veas total process [1] | 2 |
| Figure 2.2 | The thickening process at Veas | 3 |
| Figure 2.3 | The inside of a drum screen. | 4 |
| Figure 2.4 | a block diagram of polymer dosing | 5 |
| Figure 3.1 | Example of a SCADA system | 6 |
| Figure 3.2 | RevPi with extra modules [2] | 7 |
| Figure 3.3 | OSI modell compeard against Profibus protocol [3] | 7 |
| Figure 3.4 | The Web Trend system Linea with the functionality of exporting data. | 8 |
| Figure 3.5 | Example of PCA for Turbidity in SED6 at Veas[4] | 10 |
| Figure 3.6 | Example of local versus global minimum | 14 |
| Figure 4.1 | Visualization of a solution to how the problem could be solved | 17 |
| Figure 4.2 | Taking the sample for laboratory analysis | 18 |
| Figure 4.3 | A step response test was conducted on the system by altering the polymer dosage. | 20 |
| Figure 4.4 | The deviation between Q_{in} and Q_w can be observed | 21 |
| Figure 4.5 | The interpolation of the step response | 21 |
| Figure 4.6 | A step response test performed on the system | 22 |
| Figure 4.7 | Divided the dataset into two parts: one Train sett and one test sett | 24 |
| Figure 4.8 | Block diagram of linear model | 25 |
| Figure 4.9 | PiCtory program for setups of RevPi modules | 31 |
| Figure 4.10 | Simplified sequence diagram | 34 |
| Figure 5.1 | PCA is used to find variables that describe the variation in the output DS_{out} | 36 |
| Figure 5.2 | PC2 against PC4 | 37 |
| Figure 5.3 | Interpolation of dry solids with input variables | 38 |
| Figure 5.4 | Interpolation of dry solids with input variables over 2 days | 38 |
| Figure 5.5 | NARX model using a block diagram | 40 |
| Figure 5.6 | A simulation from the validation data and the residues from over all time delay used | 40 |
| Figure 5.7 | Simulation for all data from interpolation | 41 |
| Figure 5.8 | Plot of Narx model over a two-week period against laboratory analysis points | 41 |
| Figure 5.9 | Plot of ARX model over a two-week period against laboratory analysis points | 42 |
| Figure 5.10 | Plot of Mass balance model over a two-week period against laboratory analysis points | 43 |
| Figure 5.11 | Test of comparing the models | 43 |
| Figure 5.12 | Step response to find time delay | 44 |
| Figure 5.13 | Lab results compered to the models | 44 |
| Figure 5.14 | Visualization of MPC | 46 |
| Figure 5.15 | Plot of simumalted test of MPC | 47 |
| Figure 5.16 | Time usage of the MPC in seconds | 47 |
| Figure 5.17 | First results with bad "tuning" of MPC | 48 |
| Figure 5.18 | Normal conditions for MPC compared to operator dosage | 49 |
| Figure 5.19 | Simulated MPC with DS_{in} outside limitations | 50 |
| Figure 5.20 | Closed loop run of polymer dosage by MPC in a 6 hour time span | 51 |
| Figure 5.21 | MPC initiates dosing at 09:00, replacing the previous dosing strategy in use | 51 |
| Figure 5.22 | The MPC system manages the stoppage of a drum screen at roughly 11:15. | 52 |

LIST OF CODES

| | | |
|------|--|----|
| 3.1 | Code for giving SysIdentPy the information needed [5] | 15 |
| 4.1 | Code for scaling | 19 |
| 4.2 | Code for change the dry solids quantity | 23 |
| 4.3 | Code for dividing the dataset into two | 24 |
| 4.4 | Code for giving SysIdentPy the information needed | 25 |
| 4.5 | Code for fitting a ARX model | 26 |
| 4.6 | Code for minimizing the objective function | 30 |
| 4.7 | The objective function methode | 30 |
| 4.8 | Command for starting graphical interface | 31 |
| 4.9 | Retrieve data from the Profibus communication for byte number 1. | 32 |
| 4.10 | Setting 4 bytes into one float | 32 |
| 4.11 | Simple estimation script | 33 |
| 4.12 | How to use MPC module | 33 |

LIST OF TABLES

| | | |
|---------|--|----|
| Table 1 | The points to interpolate between | 20 |
| Table 2 | Laboratory results for all samples taken. If a "-" is indicated in the table, it signifies that the corresponding test was not conducted by the laboratory for that particular sampling. | 35 |
| Table 3 | NARX model training and evaluation results | 39 |

INTRODUCTION

A wastewater treatment plant (WWTP), named Veas is essential for effectively managing the waste water from Oslo, Bærum, and Asker municipalities, serving a substantial population. Ensuring the effectiveness of its operations is crucial for maintaining environmental standards and protecting public health. Nevertheless, Veas encounters difficulties in optimizing specific processes, specifically those related to the drum screens, similar to other wastewater treatment facilities. Chapter 2 will provide an in-depth description of Veas's processes.

Objective

This study seeks to tackle these challenges by examining different types of models and evaluating model-based control techniques that are specifically designed for the drum screens at Veas. The main goal is to improve process efficiency and reduce the need for human assistance in managing and maintaining the drum screens.

Specific focus

- **Model Development:** An essential goal is to create a strong model that can precisely estimate the dry solids content that comes out of the drum screens. Achieving this will yield valuable knowledge about the efficiency of the thickening process and improve decision-making in terms of process optimization.
- **Control Strategy Enhancement:** Another crucial element of this study is to incorporate advanced control strategies for polymer dosing. The process of thickening is greatly influenced by the amount of polymer added, and by optimizing the dosage, the efficiency of the process can be greatly improved while also reducing operational costs.
- **Integration with SCADA system:** Furthermore, the purpose of this study is to create a communication interface between the Veas SCADA system and the developed model. By integrating the model with the SCADA system, operators will be able to monitor and control in real-time, enabling them to make more precisely adjustments and proactively respond to process variations.

Addressed challenges

The existing control systems used at the drum screens face multiple issues, including dosage inaccuracies and the requirement for frequent human interaction. The objective of this study is to boost the efficiency of the thickening process by employing sophisticated modeling techniques and control algorithms to enhance the current manual control system.

SYSTEM DESCRIPTION

Veas is Norway's biggest WWTP responsible for treating sewage from Oslo, Bærum, and Asker municipalities, serving approximately 867,000 people [6]. The plant is located in Slemmestad, Asker. To transport the sewage from Oslo to Slemmestad, there is a 42 km long tunnel through which the water flows by gravity. It takes between 5 to 7 hours for the water to travel from Oslo to Veas [6].

When waste water reaches Veas, it undergoes treatment through three different methods: the main facility, Actiflo (a stormwater treatment plant), or Bypass (which removes only grid waste). Actiflo comes into operation only when the total waste water volume entering Veas exceeds 6800 l/s, while Bypass is utilized if it surpasses 8500 l/s. Together, these three facilities have the capacity to treat up to 11,000 l/s [1]. Figure 2.1 illustrates all the treatment processes at Veas with detailed steps. Since this project focuses on a small part of the main facility, the stormwater treatment plant and bypass are neglected. Additionally, it's noteworthy that approximately 80% of the time, the main facility is the only operational system at Veas.

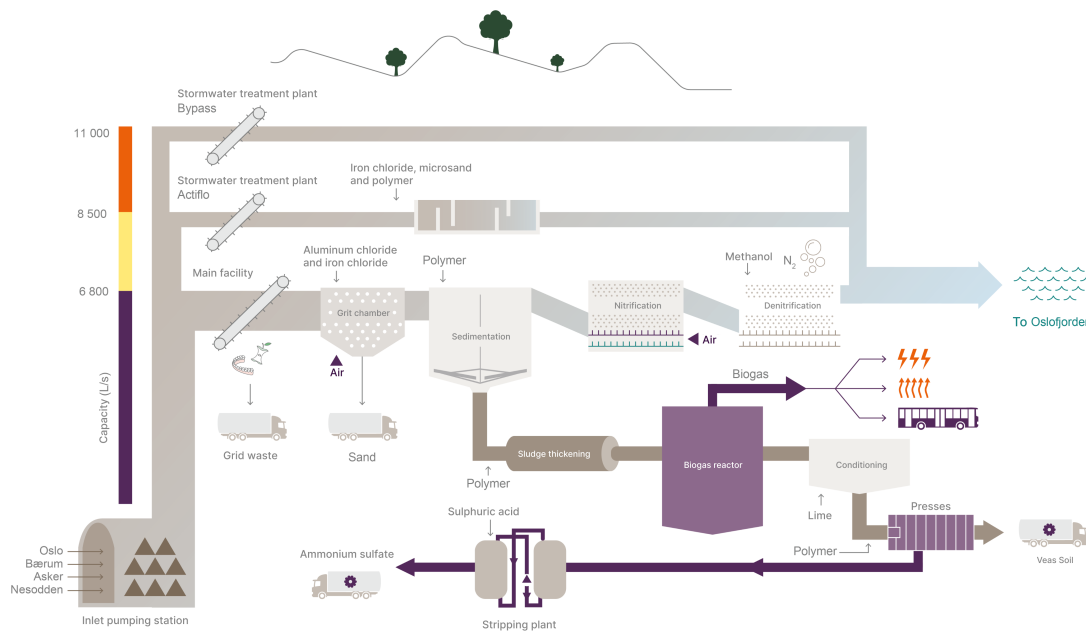


Figure 2.1: Veas total process [1]

Veas main process consists of two primary tasks: water treatment and sludge treatment.

2.1 Water treatment

The water treatment process at Veas consists of three main treatment stages:

- Mechanical cleaning.
- Chemical cleaning.
- Biological cleaning.

The mechanical purification process consists of a series of screens that act as coarse filters, removing solid waste such as large debris, plastics, and other non-biodegradable materials from the wastewater (grid waste) [7].

The Chemical cleaning process starts by adding aluminum- and iron chloride to the wastewater. Subsequently, the water is directed into a basin where sand particles settle at the bottom, while fats rise to the surface (aerated grit chamber). Polymer is then added before the wastewater proceeds to a sedimentation basin (SED). Within the SED, sludge is separated from the water. The sludge settles at the basin's base for sludge treatment, while the water phase continues to the next step of water treatment from the top of SED [7].

The final stage of water treatment before discharge into the Oslo Fjord involves biological treatment to remove dissolved nitrogen. This process begins in a basin where air is injected into the water, aiding in the microbial conversion of ammonia into nitrate (nitrification) [8]. Subsequently, methanol is introduced into the water before it enters a final basin, where nitrate is reduced to N_2 -gas through denitrification [8].

2.2 Sludge treatment

The sludge treatment process starts at the bottom of the sedimentation tank and primarily consists of three components:

- Thickening
- Anaerobic digestion
- Dewatering

Once the sludge has settled at the bottom of the sedimentation tank, it is pumped out and undergoes the thickening process. This process begins by adding polymer to the sludge. The sludge then passes through a drum screen, which filters out the water, resulting in a thicker sludge [7].

After thickening, the sludge is sent to anaerobic digestion where biogas is produced. Subsequently, the digested sludge (digestate) is mixed with polymer and lime before being dewatered and hygienized (heat treated) in chamber filter presses. This process is carried out to produce a soil enhancement product used for fertilization in agriculture throughout the broader of Oslo region [7].

2.2.1 Drum Screens

Since this project aimed to improve the thickening process at Veas, focusing on the drum screens, it is essential to dive into how this is carried out.

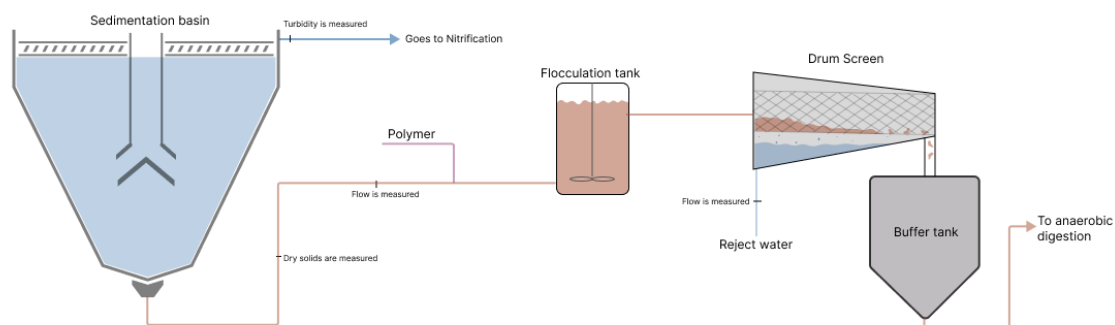


Figure 2.2: The thickening process at Veas

Veas has 8 water treatment lines, each equipped with a SED. From each of these 8 SED tanks, sludge is pumped out from the bottom. Currently, the process lines operate in pairs, meaning SED₁ and SED₂ cannot pump simultaneously, nor can SED₃ and SED₄, and so on. This arrangement is due to the fact that SED pairs share the same sludge pump.

Before being pumped to the drum screens, the sludge undergoes measurement of its dry solids content. Next, it is divided into five branch pipes, each designated for a drum screen. Prior to entering the drum screen, the sludge velocity is measured, and polymer is mechanically mixed in for optimal performance.

Following this initial stage, the sludge is transported to a flocculation tank. Here, sufficient time is allocated for the polymer to interact with the sludge, promoting the formation of stable flocs essential for an effective drum screen operation [9]. Ensuring a loose floc structure requires rotating the drum screen at a low rotation to minimize shear forces. This rotational motion facilitates the separation of the water phase from the sludge [10].

Determining the appropriate strength and type of polymer for floc formation is a detailed process conducted by polymer experts at a laboratory scale [10]. This specific aspect is beyond the scope of this master's thesis.

The drum screen consists of a rotating perforated filter cloth that allows liquid to pass through while retaining the sludge. As the filter cloth rotates, the sludge is pushed to the end of the drum screen, where it falls into a buffer tank named FOR2. From there, the sludge is sent to anaerobic digestion. Figure 2.2 illustrates a system diagram of the thickening process.

As described, the drum screens feature a rotating filter cloth that, due to their perforated nature, are vulnerable to clogging. To prevent this, they are rinsed with cold water approximately once every minute, with each rinsing cycle lasting about 10 seconds. Additionally, they are rinsed twice a day with hot water, which takes about 40 minutes. In Figure 2.3, the interior of the drum screen is represented, showing the filter cloth positioned on a cylindrical drum, with the rinsing nozzles located to the left of the drum.

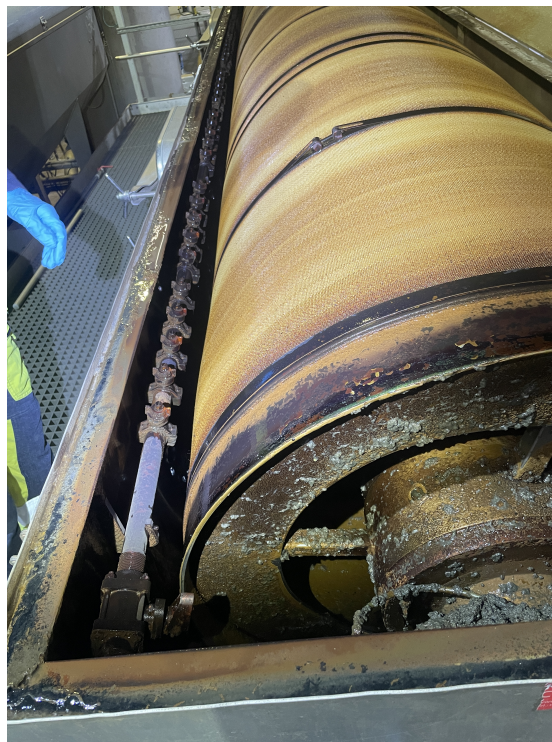


Figure 2.3: The inside of a drum screen.

The polymer addition is regulated by a feedforward control system together with a PID controller. This is achieved by the PID controller receiving a calculated setpoint based on the formula in Equation 2.1 [11].

$$SP = \min\left(\frac{DS_{in}Q_{in}\theta}{1000DS_p}, 1\right) \quad (2.1)$$

Where DS_{in} is the percentage of dry solids entering the drum screen, Q_{in} , represents the flow entering the drum screen, DS_p represents the dry solids content in the polymer addition, and θ is an operator-set value indicating the amount of polymer to be added per ton of dry solids [11].

A visualization of the current polymer dosing process is illustrated in a block diagram in Figure 2.4.

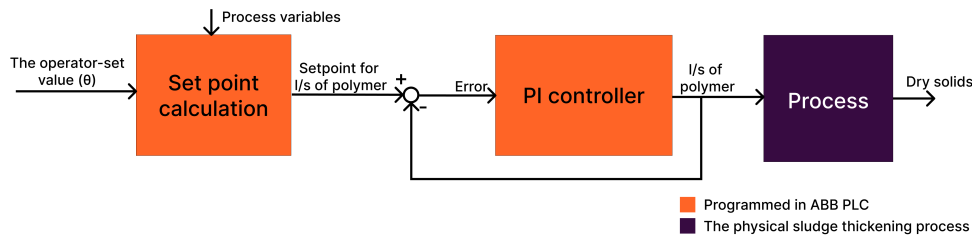


Figure 2.4: a block diagram of polymer dosing

Operatives use a dry solids meter placed on the pipeline for extracting sludge from FOR2, as physically measuring the dry solids content from the drum screen is impractical. Relying on this meter makes the process unstable and challenging for operatives to ascertain the accurate values for θ . The duration from separation in the drum screen to pumping out from FOR2 is substantial, ranging between one to two days depending on the volume of sludge being pumped to anaerobic digestion at Veas [12].

Moreover, this measurement includes all drum screens and isn't specific to any one screen.

This situation presents an opportunity for improvement because overdosing polymer in the process is easy, which, although not harmful, is costly for Veas. For example, knowing the dry solids content that flows into FOR2 would help regulate the polymer dosage into the drum screens more effectively.

2.3 Automation at Veas

Veas operates with a high degree of automation, enabling continuous operation without the need for 24-hour staffing. The facility runs its processes, facilitated by PLCs primarily from ABB and Siemens. ABB PLCs are the most dominant, with Veas employing a total of 61 ABB PLCs. One of these ABB PLCs, named *FOR – DKS – PLS01*, controls the majority of the thickening process at Veas [11].

3.1 SCADA

Industrial operations often require connecting various devices using different communication protocols. The distances between these devices can range from a few meters to several hundred kilometers. Telemetry is used to transmit commands, programs, and monitor data from remote locations [13].

Supervisory Control And Data Acquisition (SCADA) is a combination of various telemetry and data acquisition methods [13]. For Veas, such a SCADA system consists of several components, with the main areas including Programmable Logic Controller (PLC), Distributed Control System (DCS), and a central station where necessary tasks are performed. Figure 3.1 is an example of a SCADA system

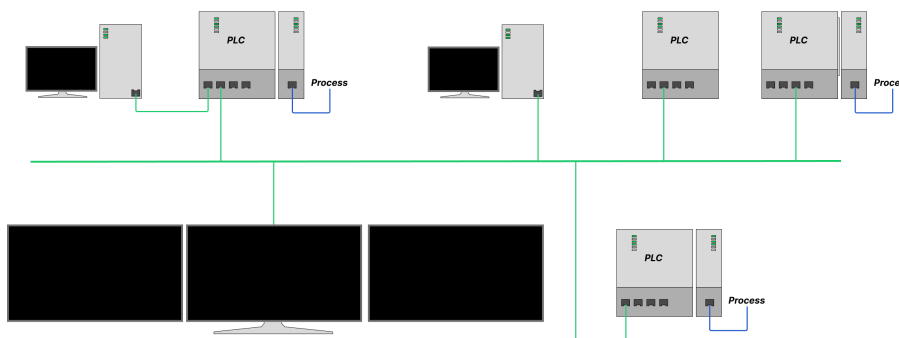


Figure 3.1: Example of a SCADA system

3.1.1 PLC

A Programmable Logic Controller (PLC) is a specialized computer designed for industrial control and automation of process facilities [14].

A PLC comprises several key components, including:

- The Central Processing Unit (CPU), responsible for processing signals [14].
- Input/Output (I/O) modules, which collect and send signals to external devices like sensors and motors. These signals can be digital or analog [14].
- Communication modules, facilitating interaction between different systems. Separate modules are often used for different communication protocols, such as Profibus [15] and OPC [16].

Signals

Analog signals, mirroring physical quantities, display continuous variations over time. Measuring devices capture these signals by modulating either current or voltage. In contrast, digital signals are discrete, presenting only two distinct values: high or low [14].

PLCs can also detect analog signals by sending different currents through signal cables. The most common method is using 4-20mA, although other combinations such as voltage changes ($\pm 10V$) are also used. [14].

3.1.2 Revolution Pi

The Revolution Pi (RevPi) is an open-source and modular PC that can be easily compared to an industrial PLC. The system is based on a Raspberry Pi but instead of relying on complicated connectivity methods, RevPi utilizes modules such as I/O cards or Profibus modules, making it more suitable for industrial use. It also complies with the requirements set forth in EN 61131-2 and IEC 61131-2 standards [2]. Figure 3.2 illustrates an example setup with RevPi and additional modules.

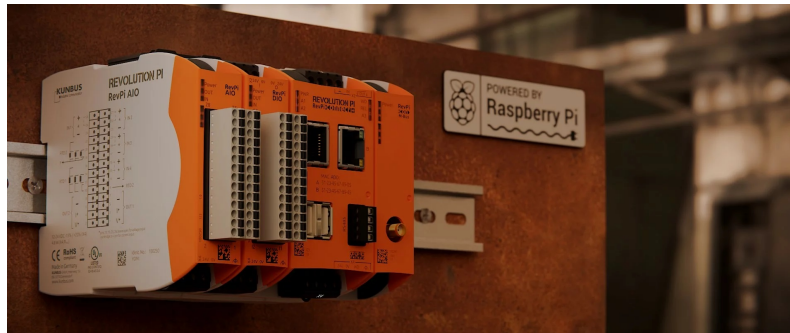


Figure 3.2: RevPi with extra modules [2]

3.1.3 Profibus

Profibus is the most commonly used communication protocol in the industry and is used across various physical systems. Built upon Fieldbus technology, Profibus operates at the application layer in the OSI model [15]. Figure 3.3 illustrates the comparison between Profibus architecture and the OSI model. To facilitate easy access to Profibus, the General Station Description (GSD) is widely utilized [15].

| | User Program | Application profiles |
|---|--------------|-----------------------------|
| 7 | Application | DP-V0; DP-V1; DP-V2 |
| 6 | Presentation | NOT USED |
| 5 | Session | |
| 4 | Transport | |
| 3 | Network | |
| 2 | Data link | Master-Slave; Token-Passing |
| 1 | Physical | RS485; Fiber-Optic; MBP |
| | OSI Model | PROFIBUS OSI Model |

Figure 3.3: OSI modell compared against Profibus protocol [3]

The GSD file is specifically created for each device communicating over Profibus. This file contains information about the module and enables access to and reading of inputs and outputs (I/O) [15].

In a Profibus communication setup, there is usually one master device and multiple slave devices. The master device is responsible for initiating communication and exchanging data with the slaves. The GSD file, containing specific information about each device and needs to be loaded into the master device [15]. A common configuration example involves a PLC functioning as the master device, controlling several frequency converters acting as slave devices.

3.1.4 OPC DA

A SCADA system typically consist of various functions such as advanced control, I/O and HMI operations. A communication protocol enabling real-time data exchange between these functions is OLE for Process Control Data Access (OPC DA). OPC DA utilizes Distributed Component Object Model (DCOM) to facilitate communication among the different parts of the SCADA system. Consequently, to utilize OPC DA, it is often necessary to operate on a Microsoft machine, as DCOM is part of Microsoft's communication protocol [16].

3.1.5 Linea

To extract historical data for comparison and model creation, Veas employs an ABB tool called "ABB AbilityTM EdgeInsightTM." This database stores all values from instruments. To obtain data from the SCADA system, Veas uses the OPC DA protocol to send data to "ABB AbilityTM EdgeInsightTM". [17].

Subsequently, this data can be easily retrieved online using Linea, which is also an online web trend system developed by ABB. From this Trend System, data can easily be exported by selecting the specific time frame and variables of interest. Figure 3.4 displays Linea as a web trend system, showing the capability to export data [17].



Figure 3.4: The Web Trend system Linea with the functionality of exporting data.

3.2 Unique measurement units

There are several specialized measurement units used in this project, although they are SI units, they may considered to be uncommon. Some of the units used in this project include Dry Solids, Loss on Ignition, and Turbidity.

3.2.1 Dry solids

Dry solids in sludge refers to the percentage of solid material in the sludge, indicating the amount of water present. By measuring the dry solids, it becomes possible to evaluate the efficiency of the purification process. Veas calculates the unit of dry solids as a percentage (%DS). This can be calculated using the eq. 3.1 [18]:

$$DS = \frac{m_d}{m_{tot}} * 100\% \quad (3.1)$$

Here:

- DS represents dry solids (%DS).
- m_d represents the mass of dry solids in the sludge.
- m_{tot} represents the total mass of the sludge before drying at 105°C [19].

3.2.2 Loss on ignition

Loss on ignition (LOI) is recognized as a method for quantifying organic material, typically expressed in Percentage Volatile Solids (%VS). To conduct this measurement, one begins with the dried mass obtained from dry solids testing, denoted as m_d . Subsequently, the sample undergoes high temperatures. Although there isn't an official standard for the temperature used, Veas allows the sludge to undergo combustion at a temperature of 550°C for one hour [20] [21]. After the combustion process, the remaining mass m_b is determined through weighing. LOI can then be calculated using the following equation 3.2. [21]

$$LOI = \left(1 - \frac{m_b}{m_d}\right) * 100 \quad (3.2)$$

3.2.3 Turbidity

Turbidity measure the clarity of water and is often expressed in Formazin Turbidity Units (FTU). This is achieved by measuring the amount of light that is reflected in the water. Essentially, if there are numerous particles in the water (such as proteins, minerals, bacteria, and more), the water will reflect more of the light source, resulting in higher turbidity levels [22]. Veas utilizes this measurement principle to assess the effectiveness of sludge settling in the SED.

3.3 Singular value decomposition

Consider a matrix Y defined in equation 3.3 [23]:

$$Y = \begin{bmatrix} y_1 & y_2 & \dots & y_n \\ y_2 & y_3 & \dots & y_{n+1} \\ \vdots & \vdots & \ddots & \vdots \\ y_m & y_{m+1} & \dots & y_{m+n} \end{bmatrix} \quad (3.3)$$

The Singular Value Decomposition (SVD) of Y is then represented as [24]:

$$Y \approx USV^T \quad (3.4)$$

In equation 3.4, U and V are orthogonal matrices, and S is a diagonal matrix. The values in S are the singular values, where S_1 is the most significant singular value, followed by S_2 , and so forth. The diagonal matrix S is structured as shown in equation 3.5 [24][23]:

$$S = \begin{bmatrix} S_1 & 0 & \dots & 0 \\ 0 & S_2 & \dots & 0 \\ 0 & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & S_n \end{bmatrix} \quad (3.5)$$

3.4 Principal component analysis

Principal Component Analysis (PCA) is a technique used to analyze multivariate systems in a simplified manner. It is possibly one of the most widely used and oldest techniques for describing and analyzing multivariate datasets [25]. The objective of such analysis is to extract the essential information from a dataset. This is achieved by transforming the data into new orthogonal axes, referred to as principal components (PCs). In these PCs, the data is represented as points on an x-y axis, which are orthogonal to each other [25].

3.4.1 How to calculate principal components

PCA is conducted through SVD. SVD is applied to a dataset, denoted as Z . By performing SVD, three matrices are derived: U , S and V [24]. The process can be represented as in eq. 3.6 [25]

$$Z = USV^T \quad (3.6)$$

$$F = US \quad (3.7)$$

In eq. 3.7, the components are described by F . Additionally, matrix V provides information about the coefficients of a linear combination to generate the results as a score. The score for each individual row from the dataset can be visualized on a plot with x-y axes. Here in V , the number of rows remains the same as the number of rows in the dataset, while the number of columns corresponds to the number of PCs, and the value represents the coefficient of the linear combination of the dataset. [25] [24]

Figure 5.1 exemplifies a PCA represented by plotting the results in two coordinate systems: one for scores and the other for loadings.

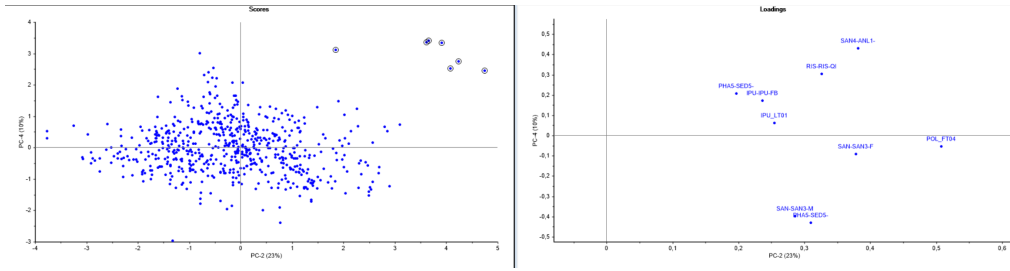


Figure 3.5: Example of PCA for Turbidity in SED6 at Veas[4]

3.4.2 Loadings

A loading represents the correlation between a PC and the variables represented in the dataset. It is crucial to remember that the sum of the squared loadings is always equal to 1 [25]. To easier analyze and display the data from the loadings, they are often plotted in a coordinate system where the loading is the coordinate and the axes represent the different PCs [25].

3.5 Proportional, integral, and derivative controller

PID controllers have a long history in automatic regulation. The first development of a PID controller occurred in 1911 for the US Navy [26].

The main principle of PID is to close the loop of the control problem by minimizing the deviation between the setpoint and the measured value. A PID controller is known for its three components: Proportional, Integral, and Derivative. In equation 3.8, the formula for calculating the control action in a PID controller is provided, where e represents the deviation between the setpoint and the measured value [26].

$$u(t) = K_p \left(e(t) + \frac{1}{T_i} \int e(t) dt + T_d \frac{de(t)}{dt} \right) \quad (3.8)$$

In this equation, there are three parameters that need to be tuned for the controller to function: K_p , T_i , and T_d . There are various methods available for tuning these parameters. [26]

3.6 Optimal control system (model predictive controller)

There are various methods to control a system. While PID control is commonly used, sometimes the system may not be adequately satisfied with PID control alone. In such cases, model-based control methods like Model Predictive Control or Linear Quadratic Control can improve control.

MPC is an advanced control method that utilizes a dynamic model of the controlled process. The basic concept of MPC is to use the control inputs of a model to forecast the future behavior of the system for a specific number of steps, n , in advance. Through understanding the present

condition of the system and predicting its future development, MPC can forecast the effects of various control inputs on the system's behavior. [27].

The main goal of MPC is to optimize the selection of control inputs in order to minimize the error between intended setpoints and the forecasted behavior of the system. This optimization procedure evaluates a variety of potential input paths and chooses the one that produces the best possible result according to predetermined performance criteria. MPC achieves precise and responsive control in dynamic situations by constantly recalculating optimal control actions within a limited prediction horizon, allowing it to react to changes in system dynamics and disturbances [27].

Generally, there are two main types of MPC: linear MPC and non-linear MPC. Both types of MPC typically start with the same foundation, employing a quadratic objective function to be minimized. In equation 3.9, the objective function is denoted by J [27].

$$J = \frac{1}{2} \sum_{k=1}^N e_k^T Q_k e_k + u_{k-1}^T P_{k-1} u_{k-1} \quad (3.9)$$

Here, e is a vector representing the error between the setpoint and the calculated values from the model at time interval k , while u denotes the controlled input signal to the system during time interval k . Minimizing J will enable finding the smallest u where the process reaches the setpoint [27].

Q and P weigh which parts of the objective function are more important. If Q is high and P is low, the minimization will prioritize the process reaching the setpoint rather than u being low [27].

In some cases, the optimal solution may not involve minimizing u directly but rather minimizing the variation in u . Hence, an alternative approach is offered through the objective function specified in Eq. 3.10. In this formulation, Δu is utilized not for the direct minimization of u , but rather to minimize fluctuations in u . [27].

$$J = \sum_{k=1}^N e_k^T Q_k e_k + \Delta u_{k-1}^T P_{k-1} \Delta u_{k-1} \quad (3.10)$$

An MPC can control a Single Input Single Output (SISO) system, but it can also control a Multiple Input Single Output (MISO) or Multiple Input Multiple Output (MIMO) system. [27].

3.6.1 Linear MPC

One way to solve a linear MPC problem is by using Linear Quadratic Optimal Control (LQ). LQ utilizes a quadratic objective function, as shown in equation 3.11 [27].

$$\min_z = \frac{1}{2} z^T H z + c^T z \quad (3.11)$$

where there are equality constrains, inequality constrains and bounds as shown in eq. 3.12-3.14 [27].

$$A_e z = b_e \quad (3.12)$$

$$A_i z \leq b_i \quad (3.13)$$

$$z_L \leq z \leq z_U \quad (3.14)$$

In equation 3.11, the variable z represents the manipulable entity (typically a vector) that the LQ problem solver utilizes. Specifically within the MPC framework, this denotes a control input responsible for regulating the relevant system [27].

To utilize LQ control in an MPC framework, the objective function J needs to be incorporated into the equation as shown in Eq. 3.11. This is achieved by first defining z as illustrated in Eq.

3.15. Here, u denotes the control input, x represents the state vector, e signifies the error term, and y indicates the output variable [27].

$$z = \begin{bmatrix} u \\ x \\ e \\ y \end{bmatrix} \quad (3.15)$$

Since MPC regulates values into the future, z becomes a longer vector than 4 in the manner shown in eqs. 3.16-3.25 [27].

$$u^T = [u_0^T \quad u_1^T \quad \dots \quad u_N^T] \quad (3.16)$$

$$x^T = [x_0^T \quad x_1^T \quad \dots \quad x_N^T] \quad (3.17)$$

$$e^T = [e_0^T \quad e_1^T \quad \dots \quad e_N^T] \quad (3.18)$$

$$y^T = [y_0^T \quad y_1^T \quad \dots \quad y_N^T] \quad (3.19)$$

After defining z , the next step is to define H and c . And they can be populated as shown in eq. 3.20-3.22 [27].

$$H = \begin{bmatrix} P & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & Q & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad (3.20)$$

$$z^T H z = \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_N \\ x_0 \\ \vdots \\ x_N \\ e_0 \\ \vdots \\ e_N \\ y_0 \\ \vdots \\ y_N \end{bmatrix}^T \begin{bmatrix} \begin{bmatrix} P_0 & 0 & \dots & 0 \\ 0 & P_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & P_{N-1} \end{bmatrix} & \begin{bmatrix} 0 & \dots & 0 \\ 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix} & \begin{bmatrix} 0 & \dots & 0 \\ 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix} & \begin{bmatrix} 0 & \dots & 0 \\ 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix} \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} 0 & \dots & 0 \\ 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \\ 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \\ 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \\ 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \\ 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_N \\ x_0 \\ \vdots \\ x_N \\ e_0 \\ \vdots \\ e_N \\ y_0 \\ \vdots \\ y_N \end{bmatrix} \quad (3.21)$$

$$c = 0 \quad (3.22)$$

Since LQ only works with linear models, the model can be put into state-space form. Thus, the equality constraints are as shown in Eqs. 3.23-3.25 [27], where r_k is the setpoint/reference point.

$$x_{k+1} = Ax_k + Bu \quad (3.23)$$

$$y_k = Cx_k \quad (3.24)$$

$$e = r_k - y_k \quad (3.25)$$

This implies that it's possible to convert these three equality constraints into the form $A_e z = b_e$. This conversion is demonstrated in eqs. 3.26-3.29 where I is the identity matrix [27].

$$A_e = \begin{bmatrix} A_{u1} & A_{x1} & A_{e1} & A_{y1} \\ A_{u2} & A_{x2} & A_{e2} & A_{y2} \\ A_{u3} & A_{x3} & A_{e3} & A_{y3} \end{bmatrix} \quad (3.26)$$

$$A_e = \begin{bmatrix} \begin{bmatrix} -B & 0 & \dots & 0 \\ 0 & -B & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & -B \end{bmatrix} & \begin{bmatrix} I & 0 & \dots & 0 \\ -A & I & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & I \end{bmatrix} & \begin{bmatrix} 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix} \\ 0 & 0 & \dots & 0 & \begin{bmatrix} -C & 0 & \dots & 0 \\ 0 & -C & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & -C \end{bmatrix} & \begin{bmatrix} 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix} & \begin{bmatrix} I & 0 & \dots & 0 \\ 0 & I & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & I \end{bmatrix} \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & \begin{bmatrix} I & 0 & \dots & 0 \\ 0 & I & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & I \end{bmatrix} \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & \begin{bmatrix} I & 0 & \dots & 0 \\ 0 & I & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & I \end{bmatrix} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & \begin{bmatrix} I & 0 & \dots & 0 \\ 0 & I & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & I \end{bmatrix} \end{bmatrix} \quad (3.27)$$

$$b_e = \begin{bmatrix} b_{e1} \\ b_{e2} \\ b_{e3} \end{bmatrix} \quad (3.28)$$

$$b_e = [[Ax_0 \ 0 \ \dots \ 0] \ [0 \ \dots \ 0] \ [r_1 \ \dots \ r_N]] \quad (3.29)$$

Having defined and transformed these matrices, a problem solver can be used to effectively solve the LQ problem.

3.6.2 Non-linear MPC

The non-linear model y are expressed as shown in eq. 3.30 where f is a non-linear function [27].

$$y = f(x, t, u) \quad (3.30)$$

By employing this system model, it becomes feasible to formulate the objective function represented in eq. 3.10. This process involves adhering to the procedures shown in eqs. 3.31 and 3.32 [27].

$$e_k = r_k - f(x, t, u) \quad (3.31)$$

$$\Delta u = u_k - u_{k-1} \quad (3.32)$$

Where the objective function is constrained as illustrated in Eq. 3.33, where c_j represents constants denoting the limitations of the system.[27].

$$g(x, t, u) \leq c_j \quad (3.33)$$

Having defined these fundamentals – a proper model for the system, a reference point, and knowledge of the actuator – it is then possible to combine these parameters to create the objective function that needs to be minimized to find a setpoint where the error between the reference point and the model is minimized. The objective function is depicted in Eq. 3.34 [27].

$$\min J = \sum_{k=1}^N e_k^T Q_k e_k + \Delta u_{k-1}^T P_{k-1} \Delta u_{k-1} \quad (3.34)$$

The minimization of the function can be approached in various ways, with software such as Python or MATLAB being common choices due to their extensive libraries for optimization. An example is the "minimize" function from the SciPy package in Python. This minimization algorithm typically finds a local minimum rather than the global minimum. This behavior is illustrated in Figure 3.6 [27].

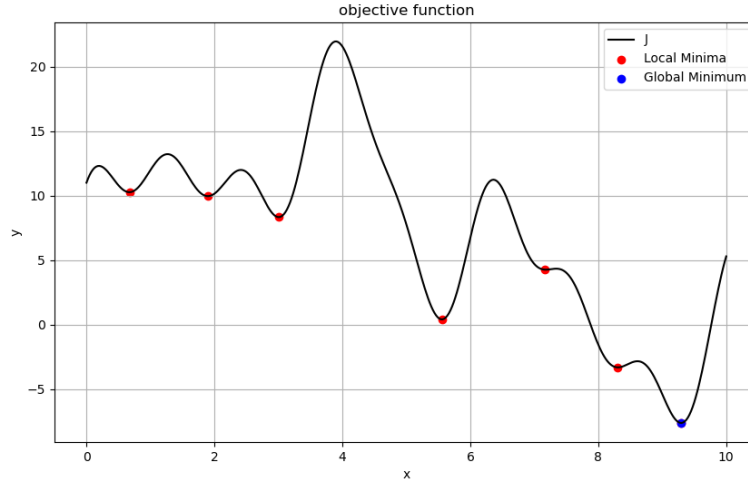


Figure 3.6: Example of local versus global minimum

3.7 System identification

System identification includes various methods, but typically shares the same objective: to construct a mathematical model describing a system’s dynamic behavior based on previously given values. There are many effective ways system identification can describe a linear system, but non-linear systems pose more challenges. Therefore, one solution to this is the Nonlinear AutoRegressive Moving Average with eXogenous input (NARMAX). This is perhaps one of the most commonly used models for non-linear systems in system identification [28].

A NARMAX model can be formulated as shown in Eq. 3.35 [28]:

$$y(t) = F[y_{k-1}y_{k-2}, \dots, y_{k-n_y}, X_{k-d-1}, X_{k-d-2}, \dots, X_{k-d-n_x}, e_{k-1}, e_{k-2}, \dots, e_{k-n_e}] + e_k \quad (3.35)$$

where:

- X_k is the systems input at time k
- y_k is the systems output at time k
- e_k is the systems noise at time k
- $n_y, n_x,$ and n_e represent the maximum lag orders for the system output, input, and model error.
- F is the nonlinear function
- d is the time delay

There are various methods to determine F , but perhaps the most common is the polynomial NARMAX model, as shown in eq. 3.36 [28].

$$y_k = \sum_{i=1}^p \theta_i \times \prod_{j=1}^{n_x} x_{k-j}^{b_{i,j}} \prod_{l=1}^{n_e} e_{k-l}^{b_{i,l}} \prod_{m=1}^{n_y} y_{k-m}^{b_{i,m}} \quad (3.36)$$

Here, θ represents the model coefficients, and $X, e,$ and y are the input, noise, and output, respectively [28].

Almeida & Martins provide an example of a Nonlinear AutoRegressive eXogenous (NARX) polynomial. Similar to NARMAX, NARX excludes moving average terms. This example is illustrated in equation 3.37 [29]. In this example, θ in equation 3.36 denotes all the coefficients, while u represents X .

$$y(k) = 1.7911y_{k_1} - 0.8073y_{k-2} + 0.0127u_{k_1} + 0.0071 \quad (3.37)$$

Determining the parameters is a straightforward task if the degree of the NARMAX polynomial is known. However, if the system being estimated has an unknown degree of the NARMAX polynomial, the task of finding this degree can be quite challenging [28].

3.7.1 Library for system identification

There are several methods to determine the NARMAX polynomial degree, one of which involves using Python and the SysIdentPy (System Identification in Python) library. This library can identify SISO and MISO models. It can find NARMAX models as well as AutoRegressive (AR), AutoRegressive with eXogenous input (ARX), AutoRegressive Moving Average with eXogenous input (ARMAX), Nonlinear AutoRegressive (NAR) and NARX models. To find such a polynomial, the library requires a dataset containing information about both already known X and Y [28].

Before SysIdentPy can generate a model, the user needs to specify the number of terms the model should include, the number of past values of y and X the model should consider, and the polynomial degree [28]. Code 3.1 presents an example of the information required from the user before SysIdentPy can find a model [5].

Code 3.1: Code for giving SysIdentPy the information needed [5]

```
basis_function = Polynomial(degree=2)

model = FROLS(
    order_selection=True,
    n_terms=4,
    extended_least_squares=False,
    ylag=1,
    xlag= [[1,2], [1,2]],
    info_criteria="aic",
    estimator="least_squares",
    basis_function=basis_function
)
```

3.7.2 Akaike's information criterion

Controlling the complexity of the model is crucial in order to prevent overfitting, which can happen when the model is very complex. Akaike's information criterion (AIC), is a method that can be used to impose restrictions or constraints. The calculation of AIC is derived from eq. 3.38 [30]:

$$AIC = -2\ln(L) + 2k \quad (3.38)$$

AIC is calculated using the number of parameters k in the model and the maximum likelihood L achieved by the model. Lower AIC values indicate either better fit or lower complexity. Thus, models with lower AIC are preferred [30].

3.7.3 Forward regression orthogonal least squares

Forward Regression Orthogonal Least Squares (FROLS) is an algorithm used in system identification. This algorithm can find an appropriate model, such as a NARMAX model. FROLS aims to achieve this by testing various combinations of inputs and outputs, multiplying a limited number of inputs with each other. It then selects the combination of inputs that yields the lowest error compared to the actual output. This process is repeated 'n' times or until a maximum limit is

reached. In this way, FROLS seeks to construct a model that best fits the observed data while limiting the complexity of the model [31].

3.8 Mass balance

The principle of mass balance is one of the most fundamental laws in physics. For mass, the unit used is kilograms, often denoted by the letter m [32].

The law states that if the conversion of mass into energy is disregarded, mass will always remain constant. This means that when considering the derivative of mass (mass flow), eq. 3.39 can be applied [32].

$$\frac{dm}{dt} = \dot{m}_{in} - \dot{m}_{out} \quad (3.39)$$

METHODS

To improve the thickening process at Veas, which was currently controlled using feedforward and PID regulation, it was decided to develop a Model-based controller. However, before this is possible, a model is needed to provide insight into the actual dry solids content exiting the drum screen. This is achieved by developing a model to estimate the dry solids content and to forecast the next setpoints for polymer dosing. Figure 4.1 illustrates the conceptual framework for how the system was intended to be structured and regulated.

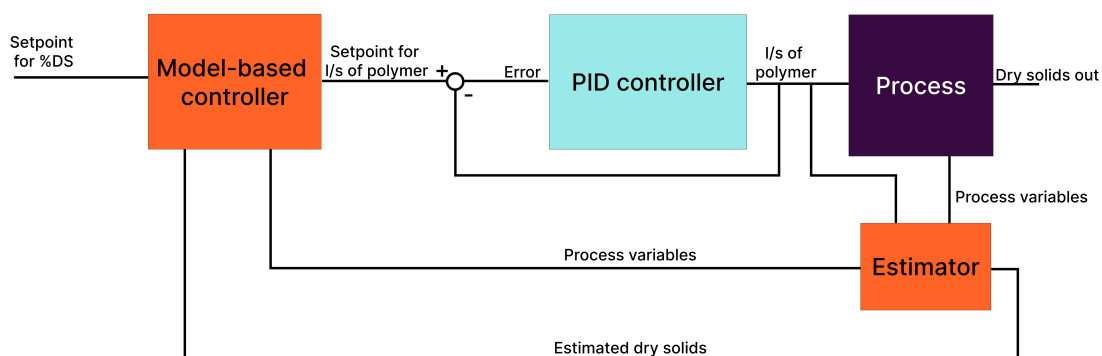


Figure 4.1: Visualization of a solution to how the problem could be solved

Before it's even possible to create a model, there needs to be data from which a model can be built. This was the starting point of this project: collecting data from online measurements and conducting physical analysis samples analyzed in a laboratory.

4.1 Getting data

Two different methods were utilized to gather data for the project. The first involved laboratory analysis, while the second relied on historical data stored in EdgeInsightTM.

4.1.1 Data gathered from laboratory analysis

Veas has designed the drum screens in a way that makes it impractical to measure DS directly out of the drum screens using an online meter. Therefore, physical samples of the sludge had to be taken. This was done by using a "spade" inserted at the end of the drum screen. This captures a portion of the sludge that falls into the FOR2 silo. The sludge to be analyzed is collected in a beaker and transported to Veas' local laboratory where it was analyzed. They tested for DS and LOI (%VS). A sampling process can be seen in Figure 4.2.

To obtain the most varied data possible, samples were therefore taken twice a day for 10 days, with the sampling times spread throughout the day. The sample gathering plan can be seen in appendix: Lab Analysis Scheduled Plan

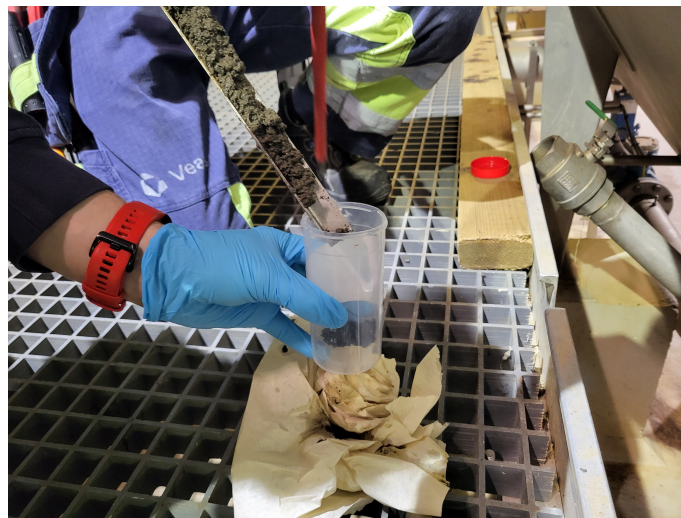


Figure 4.2: Taking the sample for laboratory analysis

4.1.2 Parameters gathered from historical data

To extract data from the process, Linea was utilized. It was crucial to ensure that the lab sample, taken twice a day, was labeled with the timestamp and date. This labeling allowed for the retrieval of corresponding data from the process at the same time. The variables that are extracted from Linea can be seen in Appendix: Symbol list and TAG names at page 64. In the appendix, variables marked with a "ζ" are sourced from Linea.

4.2 Naming the dry solids variable for use in SCADA system

Veas has its own naming system for various types of instruments and pipes. This means that all objects, whether they are virtual or physically located in the facility, must have a TAG name. Therefore, the estimation of the dry solids content must also be assigned a TAG name. According to Veas' TAG System document, TAGs should consist of three parts: "Area-System-Type" [33]. Since this belongs to the thickening area, the first part will be FOR. The next part is the system, and the system is the drum screen, hence TRS. The last part is divided into two, where it will start with "Q" since this is a quality control, followed by "B" for calculated. Then, a serial number with the first available number. Therefore, the TAG name for the estimator will be [33][34]:

FOR – TRS5 – QB01

4.3 Principal component analysis

Through the utilization of PCA, the aim was to facilitate the identification of variables that effectively show the system dynamics, specifically related to the dry solids output from a drum screen. Additionally, the analysis aimed to explore the potential for dimensionality reduction, thus optimizing the descriptive capacity of the model.

Principal components (PCs) are derived from the variables from appendix: Symbol list and TAG names, representing them on a linear x-y axis. The first PC summarize the highest variance in the dataset, followed by PC2, which captures the subsequent variance, and so on. This approach allows for a more streamlined representation of the underlying data structure, giving deeper insights into the system's behavior [25].

The first step before running a PCA is to scale the data under consideration. Equation 4.1 was utilized to scale the data.

$$X_{scaled} = \frac{X - X^{mean}}{X^{max} - X^{min}} \quad (4.1)$$

With 14 variables requiring scaling, Python was used to perform the calculation. This task was carried out using the pandas library in code 4.1:

Code 4.1: Code for scaling

```
mean = X.mean()
mini = X.min()
maxi = X.max()

for index1, row in X.iterrows():
    for name, value in row.items():
        X.at[index1, name] = (value-mean[name])/ \
            (maxi[name]-mini[name])
```

Following the data scaling process, a PCA was conducted using Python with the scikit-learn library [35]. This library includes a built-in PCA function that use SVD. The complete code for this analysis is provided in Appendix: GitHub Repository.

4.3.1 Dimensionality reduction using PCA

Since PCA was utilized to analyze which variables best describe the quantity of dry solids discharged (DS_{out}), one can examine which variables occupy the same region on the loading plot. To utilize the loading plots effectively, it is necessary to first determine which PCs describe DS_{out} . Once this was done, variables close to zero on the axis of the corresponding PC could be removed. This process continues until only variables that can describe DS in the same PC remain.

4.4 Interpolations of the laboratory results

Since only 2 samples were taken per day for 2 weeks (working days), resulting in only 20 samples, this amount of data is relatively small for constructing model. Therefore, the majority of the data was utilized to understand the behavior of the process. This involved examining various variables and their values during different samplings of lab tests.

On the final day of sampling, a step response test was conducted. This involved reducing the amount of polymer added to the system by half. Subsequently, the system was allowed to reach its estimated delay time, with an additional couple of minutes of waiting before taking a sample. Then, another sample was taken after the system had returned to normal operation,

allowing it to be back to normal. Figure 4.3 illustrates the step response test performed on the system.

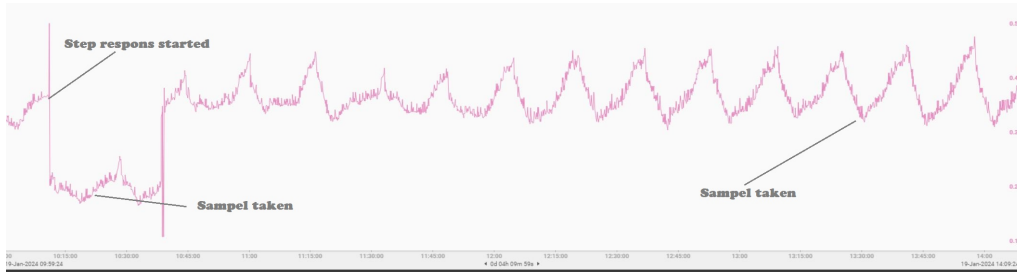


Figure 4.3: A step response test was conducted on the system by altering the polymer dosage.

During the sampling taken on the same day, it was considered representative of the typical dry solids percentage discharged from the drum screen for that particular day. This occurred following the completion of the step response test. It became feasible to interpolate during the step response test by utilizing the laboratory results, estimating the time delay, and identifying the point at which the reject water flow resumed to its normal state, indicative of the system returning to its usual operation. Table 1 delineates the identified interpolation points.

Table 1: The points to interpolate between

| Time of sample | Percentage Dry solids | Lab results? |
|----------------|-----------------------|--------------|
| 1/19/24 10:10 | 9 | No |
| 1/19/24 10:26 | 5.74 | Yes |
| 1/19/24 10:38 | 4.5 | No |
| 1/19/24 10:39 | 4.6 | No |
| 1/19/24 10:42 | 9 | No |
| 1/19/24 14:03 | 9.02 | Yes |

The points in Table 1 were generated by examining a trend showing the difference between the inflow to the drum screen (Q_{in}) and the reject water out of the drum screen (Q_w). This difference remained relatively stable until the step response was performed, after which the deviation between these two variables increased. This deviation will lead to a decrease in dry solids content according to mass balance principles. In Figure 4.4, the deviation between Q_{in} and Q_w can be observed, with an increase during the step response.

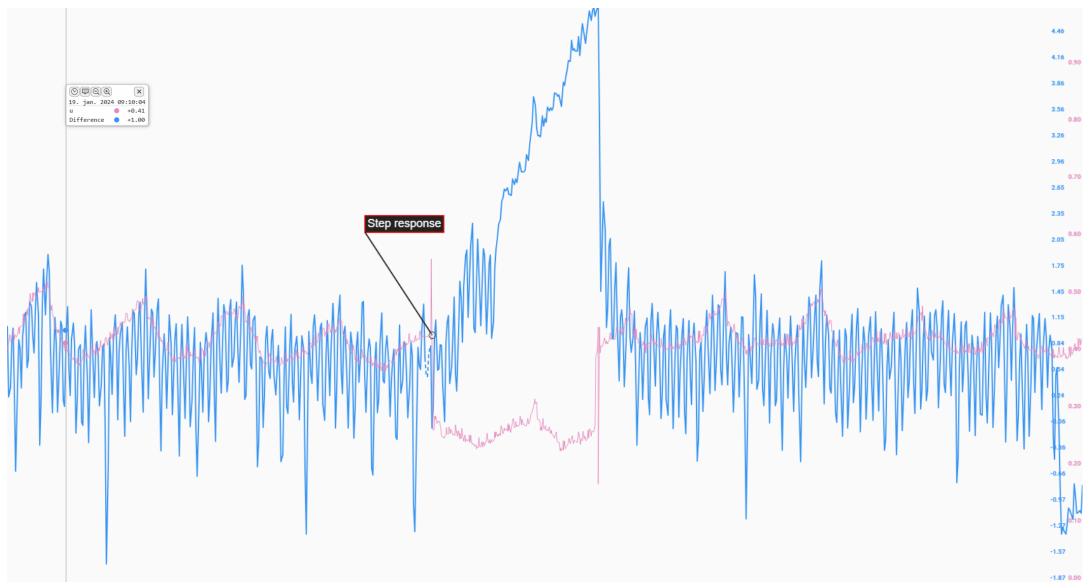


Figure 4.4: The deviation between Q_{in} and Q_w can be observed

Since the step response was completed within an hour, interpolation was carried out using minutes for the data points outlined in Table 1. For example, when the time was 10:10, $t = 10$, and when it was 10:42, $t = 42$. Furthermore, as there seemed to be two distinct behaviors observed for a decreasing step response and another for an increasing step response, two separate interpolations were carried out: one for the decreasing step response and another for the increasing step response. These interpolations are depicted in Figure 4.5.

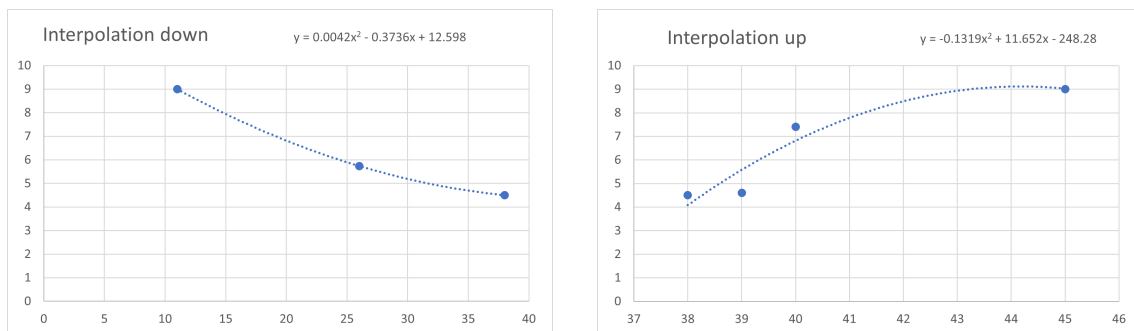


Figure 4.5: The interpolation of the step response

Combining these two interpolations, as depicted in Figure 4.5, and setting the dry solids to a constant value as the last day's sample, yields a value as shown in Figure 4.6.

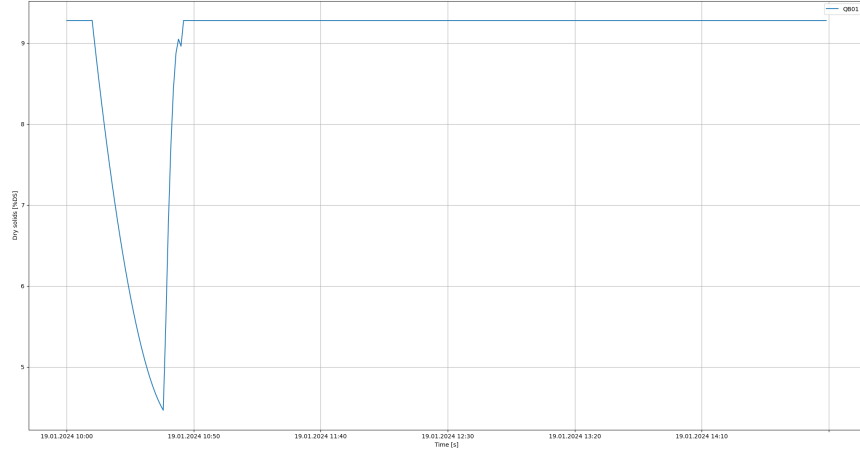


Figure 4.6: A step response test performed on the system

Considering the unrealistic assumption of a constant dry solids quantity before and after the step response test, it was considered necessary to incorporate the influence of significant changes in model inputs on the dry solids output. To address this, parameters were introduced to adjust the dry solids output.

To adjust the dry solids quantity, the formula shown in equations 4.2 to 4.5 was utilized. These formulas were selected based on the understanding of the system dynamics. If the dry solids input to the drum screen varies, the dry solids output from the drum screen will also vary. However, the most significant variation can be observed in the flow rates entering (Q_{in}) and the reject water (Q_w) of the drum screen. If the difference in flow rates between the inlet and reject water is high, it will affect the dry solids content in the drum screen due to mass balance considerations.

$$c_1 = \frac{DS_{in,k}}{DS_{in,k-1}} \quad (4.2)$$

$$c_2 = \frac{Q_{in,k}}{Q_{in,k-3}} - \frac{Q_{w,k}}{Q_{w,k-3}} \quad (4.3)$$

$$c_3 = \frac{u_k}{u_{k-3}} \quad (4.4)$$

$$y = 9.02 + \left(\left[\frac{c_1}{5} + \frac{c_2}{9} + \frac{c_3}{8} \right] + 1 \right) \quad (4.5)$$

To perform this calculation effectively, the Python code shown in Code 4.2 was utilized.

Code 4.2: Code for change the dry solids quantity

```

# Define start and stop points
# Loop through each row in the data
for index, row in data.iterrows():
    if index > start: # Check if the index is greater than the start point
        if all(sample is not None for sample in previous_samples):
# Check if all elements in previous_samples are not None
        # Calculate changes in various parameters
        change1 = (row['Q_in']) / (previous_samples[0]['Q_in'])
        change4 = (row['Q_w']) / (previous_samples[0]['Q_w'])
        change1 = change1 - change4
        change2 = row['DS_in'] / previous_samples[0]['DS_in']
        change2 = change2 - 1
        change3 = row['f_T'] / previous_samples[0]['f_t']
        change3 = (1 - change3) #inverting change3

        # Calculate the change factor
        change = (change1 / 5 + 1 * change3 / 8 + change2 / 9) + 1

        # Update TS3 based on the change factor
        TS3[index - start] = TS3[index - start] * change

        print(change) # Print the change factor

    else:
        # If not enough previous samples, keep the TS3 value unchanged
        TS3[index - start] = TS3[index - start]

# Update the previous samples list:
# Remove the oldest previous sample
previous_samples.pop(0)
# Add the current row as the newest previous sample
previous_samples.append(row)

```

4.5 Model creation

There are various methods to develop a model. In this project, there was on three different approaches. One involved using system identification and NARX models, while another relied on physical laws, particularly the law of conservation of mass. The last one relies on a linear model with a time constant and a time delay.

4.5.1 NARX model

To construct a model using a NARMAX/NARX polynomial, the SysIdentPy library in Python was used. Prior to model construction, a dataset was necessary, formed by interpolating the step response, yielding 1323 data points for training. The interpolation data was merged with the input data in a CSV file, subsequently imported into Python using the pandas library. The variables chosen as input are all variables evaluated to be used from PCA and dimensionality reduction.

Dividing dataset

Even though a dataset was generated for model construction, it needed to be divided into two parts: a training set and a validation set. Since a NARX model depends on previous steps of itself to predict the next value at time k , it was chosen not to randomly split the dataset but rather divide the first 90% of the dataset for training and the last 10% for validation. The dataset splitting was performed as illustrated in Code 4.3.

Code 4.3: Code for dividing the dataset into two

```
# data is the original DataFrame
total_rows = len(data)
split_index = int(0.9 * total_rows)

# Select the first 90% for df_train
df_train = data.iloc[:split_index]
y_train = ((df_train['DS_in']).to_numpy()).reshape(-1, 1)
x_train = (df_train.drop(['DS_in'], axis=1)).to_numpy()

# Select the remaining 10% 'for df_test
df_test = data.iloc[split_index:]
y_test = ((df_test['DS_in']).to_numpy()).reshape(-1, 1)
x_test = (df_test.drop(['DS_in'], axis=1)).to_numpy()
```

After splitting the dataset as shown in code 4.3, there were 1190 data points for training and 133 data points for testing.

To validate the data to be used, it can be manually inspected by plotting the data separately to verify that the data needed to generate a good model is in the training set. Additionally, it's important to ensure that the validation set contains good enough values to test the model afterward. In Figure 4.7, the data used for validating and training the model can be seen.

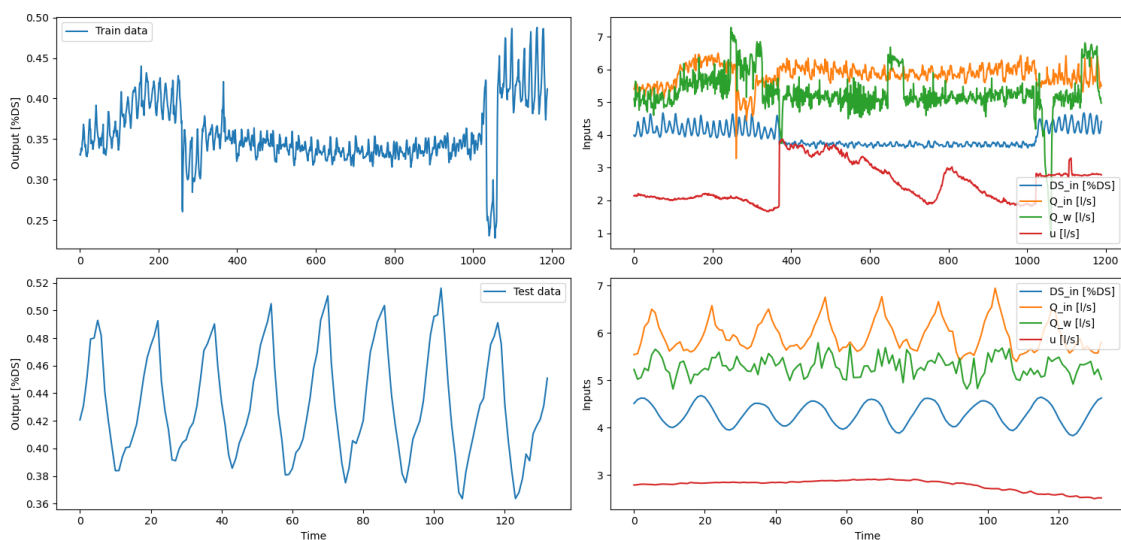


Figure 4.7: Divided the dataset into two parts: one Train sett and one test sett

Select candidates to construct NARX polynomial.

Since there is no information available about the degree or the number of terms the NARX polynomial should contain, it is important to provide good initial conditions for how many time samples back in time each input should use and from its own previous values. Additionally, it is also important to choose the correct number of terms and polynomial degree. Overall, the points below must be completed before a model can be created:

- n_{terms} is numbers of terms in the polynom.
- y_{lags} is number of time samples back in time for y .
- x_{lags} is a matrix that gives number of time samples back in time for each individual x .
- P_{degree} is the degree of a polynomial to use.

To specify this to the library, the code shown in Code 4.4 was used. Here, it can be seen that there are more options than those mentioned in the points above. These are *info_criteria* and *estimator*. "AIC" was used for *info_criteria* and the "least squares method" for *estimator*. Additionally, the FROLS algorithm was applied to determine the total number of terms.

Code 4.4: Code for giving SysIdentPy the information needed

```
#Build the model
basis_function = Polynomial(degree=3)

model = FROLS(
    order_selection=False,
    n_terms=4,
    extended_least_squares=False,
    ylag=1,
    xlag=[[1, 2, 3, 4], [1,], [1], [1, 2, 3, 4], [1, 2, 3, 4]],
    info_criteria="aic",
    estimator="least_squares",
    basis_function=basis_function,
)
```

4.5.2 Construction of a linear model with time delay and time constant.

An attempt was made to create a linear model with only u as input and y as output, with the rest of the variables acting as disturbances. Figure 4.8 illustrates this.

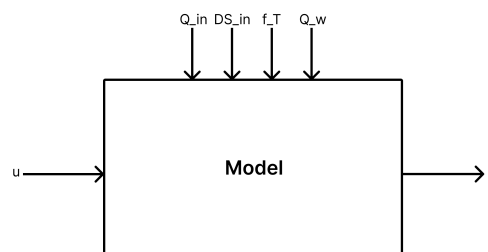


Figure 4.8: Block diagram of linear model

To create such a model, a model structure was first defined. If one ignore the noise for a moment, a simple linear model can be used also known as ARX, as shown in equation 4.6.

$$\hat{y}_k = -\frac{y_{k-1}}{\tau} + ku(t - \Delta t_f) \quad (4.6)$$

Here, τ represents the time constant, and Δt_f represents the time delay of the input signal u . Hence, one can proceed to introduce noise as input parameters, as depicted in equation 4.7.

$$\hat{y}_k = -\frac{y_{k-1}}{\tau} + k_1 u(t - \Delta t_f) + k_2 Q_{in}(t) + k_3 DS_{in}(t - \Delta t_p) + k_4 f_T(t - \Delta t_F) + k_5 Q_w(t) \quad (4.7)$$

In order to utilize such a model, it was necessary to determine the correct values for the parameters: τ , k_1 through k_5 , and find the appropriate time delays.

To find these parameters, the same interpolated dataset as used for the NARX model was employed. The code in Code 4.5 was utilized for this purpose. The time delays are calculated in the section titled [Finding the time delay for the process](#).

Code 4.5: Code for fitting a ARX model

```
# Defie model
def model(t, tau, k1, k2, k3, k4, k5):
    return -1/tau * y + k1 * u1 + k2 * u2 \
        + k3 * u3 + k4 * u4 + k5 * u5

# Fitting model
popt, pcov = curve_fit(model, t_data, y_data)
```

4.5.3 Usage of mass balance

The aim was also to develop a model that incorporates physical laws as a foundation. In this context, mass balance plays a crucial role. Equation 4.8 represents the fundamental mass balance equation, which serves as the starting point for constructing a model describing the percentage of dry solids.

$$\frac{dm}{dt} = \dot{m}_{in} - \dot{m}_{out} \quad (4.8)$$

If there is no change in mass within the drum screen, one can then apply the law of mass balance, which enables the creation of a more detailed mass balance equation. Given the presence of two outlets and one inlet to the drum screen, eq. 4.9 can be applied.

$$\dot{m}_{in} - (\dot{m}_s + \dot{m}_w) = 0 \quad (4.9)$$

Where \dot{m}_{in} is the inlet to the drum screen, \dot{m}_s is the outlet of concentrated sludge, and \dot{m}_w is the outlet of the reject water. Equation 4.9 can be rearranged, allowing for the determination of the flow rates exiting the drum screen as concentrated sludge, as shown in equation 4.10.

$$\dot{m}_s = \dot{m}_{in} - \dot{m}_w \quad (4.10)$$

$$\dot{m}_s = \text{unknown} \quad (4.11)$$

Since Veas measures flow in l/s and the law of mass conservation relates to mass flow, this can be adjusted as shown in equations 4.12 and 4.13.

$$\dot{m}_{in,tot} = Q_{mix}\rho_{mix} \quad (4.12)$$

$$\dot{m}_{w,tot} = Q_w\rho_w \quad (4.13)$$

Additionally, knowing the amount of dry solids in the sludge passing through the inlet pipe allows for determining the proportion of the total volume in the pipe that consists of dry solids. This calculation is performed as shown in equations 4.14 and 4.15.

$$\dot{m}_{in,s} = \dot{m}_{in,tot} \frac{DS_{in}}{100} \quad (4.14)$$

$$\dot{m}_{w,s} = \dot{m}_{w,tot} \frac{DS_w}{100} \quad (4.15)$$

Since dry solids content is not directly measured and is likely very low in the reject water from the drum screen, suspended solids (SS) can be used instead. Although SS is also not directly measured, it is assumed to range between 20 and 50 mg/l. Therefore, it can be converted to dry solids content using the method shown in equation 4.16.

$$DS_w = \frac{SS}{\rho} 100\% \quad (4.16)$$

Thus, the mass flow rate of pure dry solids can be calculated, as well as the total mass flow rate of the combined water and sludge. This is illustrated in equations 4.17 and 4.18.

$$\dot{m}_{s,s} = \dot{m}_{in,s} - \dot{m}_{w,s} \quad (4.17)$$

$$\dot{m}_{s,tot} = \dot{m}_{in,tot} - \dot{m}_{w,tot} \quad (4.18)$$

When both the total mass flow rate and the proportion of this flow that is dry solids are known, the dry solids content can be calculated as shown in equation 4.19.

$$DS_{out} = \frac{\dot{m}_{s,s}}{\dot{m}_{s,tot}} \times 100\% \quad (4.19)$$

4.6 Finding the time delay for the process

An effort was made to determine the time delay of the system. However, due to the fact that various inputs enter the system at different junctures, each input is subject to its own unique time delay. Specifically, polymer injection occurs immediately prior to the flocculation tank, thus only the time delay associated with this tank influences it as shown in 2.2. Moreover, the measurement of dry matter entering the drum screen is positioned 42.5 meters distant from the flocculation tank, leading to a distinct time delay for drum screen 5. Additionally, determining the time delay for turbidity presents a considerable challenge.

4.6.1 Time delay of flocculation tank into the drum screen

To determine the time delay from polymer addition to its arrival at the drum screen, fluid dynamics principles can be applied [36]. Since polymer is added just before the flocculation tank, the time delay for this specific process can be calculated. First, the total flow rate must be determined, as two fluids are being mixed together:

$$Q_{tot} = Q_w + Q_{poly} \quad (4.20)$$

Then, it can be determined how long it will take to replace all the fluid in the tank by using equation 4.21 and 4.22.

$$Q_{tot} = \frac{V}{\Delta t_f} \quad (4.21)$$

$$V = \pi r^2 h \quad (4.22)$$

Then, these formulas can be combined and calculated. The time delay is calculated in equations 4.23 - 4.25.

$$\Delta t_f = \frac{\pi r^2 h}{Q_w + Q_{poly}} \quad (4.23)$$

$$r = 0.6m, \quad h = 2m, \quad Q_w \approx 6l/s, \quad Q_{poly} \approx 0.4l/s \quad (4.24)$$

$$\Delta t_f = \frac{\pi 0.6m^2 \times 2m}{(6 + 0.4)/1000m^3/s} \approx 353.42s \approx 6min \quad (4.25)$$

This implies that there is an approximate 6-minute time delay from the introduction of polymer until it reaches the drum screens.

4.6.2 Time delay of dry solids into the drum screen

To determine the time delay from when dry solids is measured until it reaches the drum screens, the time it takes for the sludge to flow through a pipe needs to be calculated using fluid dynamics [36]. This involves determining the flow velocity v of the sludge through the pipe, which can be computed using equation 4.26 and 4.27.

$$Q = Av \quad (4.26)$$

$$A = \pi \frac{d^2}{2} \quad (4.27)$$

Once the velocity of the sludge has been calculated, the time it takes for the sludge to travel through the pipe can be determined using the eq 4.28

$$v = \frac{l}{\Delta t_p} \quad (4.28)$$

Hence, these calculations can be combined and solved as shown in equations 4.29 through 4.32.

$$Q = \frac{l}{\Delta t_p} \frac{d^2}{2} \quad (4.29)$$

$$\Delta t_p = \frac{l\pi d^2}{4Q} \quad (4.30)$$

$$d = 0.25m, \quad l = 42.5m, \quad Q \approx 25l/s \quad (4.31)$$

$$\Delta t_p = \frac{42.5m \times \pi \times 0.25m^2}{4 \times (25/1000)m^3/s} \approx 83.44s \approx 1min \quad (4.32)$$

Finally, the delay from the tank can then be added to the pipe delay:

$$\Delta t_D = \Delta t_p + \Delta t_f = 83.44s + 353.42s = 436.87s \approx 7min \quad (4.33)$$

Thus, the delay from the measurement of dry solids to the drum screen is approximately 7 minutes.

4.6.3 Time delay of Turbidity into the drum screen

In the results, a significant observation was made regarding the turbidity in the SED basins, which has a pronounced impact on the dry solids content out of the drum screens. Therefore, determining the time delay for turbidity becomes crucial. This task poses a challenge, primarily because turbidity, which indicates the amount of suspended solids that fail to settle at the bottom of the SED tanks, is not directly correlated with the drum screen operation. In this study, turbidity serves as an average value across all 8 SED tanks at Veas.

The challenge arises due to the varying distances between the basins and the drum screen, with one basin located just tens of meters away while another may be several hundred meters distant. Consequently, determining the time delay for turbidity proves complex. To address this, the time delay for turbidity is assumed to be equivalent to that of dry solids plus an additional 3 minutes. This assumption is supported by the observation that turbidity exhibits minimal variation over short periods. Therefore, whether the time delay is 10 minutes or 20 minutes becomes inconsequential when considering that turbidity typically changes by no more than 10 FTU over a 3-day period.

4.7 MPC creation

To create an MPC, it's crucial to have a model that accurately describes the system. In this case, the model was derived from system identification. To view the model used, see eq. 5.5 in results. For simplicity, the model can be described as eq. 4.34

$$\hat{y}_k = f(x_k, x_{k-1}, \dots, x_{k-N}, y_{k-1}, u_k, u_{k-1}, \dots, u_{k-N}) \quad (4.34)$$

In the nonlinear model, \hat{y}_k represents the estimated dry solids from the drum screens at time k . Here, x is a matrix containing various uncontrollable input values to the drum screens, while u denotes a controllable input.

Having established the model to be used, the next step was to decide between using non-linear MPC or linear MPC and thus potentially utilize the LQ optimization. The model generated by SysIdentPy is in non-linear form, which would require linearizing the model to use LQ optimization. However, the linearized model can yield significant deviations when operating outside its linearized range. Therefore, non-linear MPC was chosen to proceed, ensuring robust performance across a wider operating range.

The next step is to formulate the objective function. Since the aim of using an MPC as a controller is to reduce the operator's workload in adjusting θ from Eq. 2.1, while also reducing polymer consumption, the objective function can be formulated as shown in Eq. 4.35.

$$\min J = \sum_{k=1}^N e_k^T Q_k e_k + u_{k-1}^T P_{k-1} u_{k-1} \quad (4.35)$$

This formulation of the objective function could potentially introduce a steady-state deviation between the setpoint and the measured value. Additionally, it was desired for the controller to respond slowly. By specifying minimal use of the control input and a slow response, the objective function shown in eq. 4.36 was employed instead.

$$\min J = \sum_{k=1}^N e_k^T Q_k e_k + u_{k-1}^T R_{k-1} u_{k-1} + \Delta u_{k-1}^T P_{k-1} \Delta u_{k-1} \quad (4.36)$$

By using this objective function, it's possible to minimize both the control effort u and the rate of change of u , which can slow down the control response. However, as long as u is minimized, the chance of having a steady-state deviation is still there.

Once the objective function had been determined, attention turned to constraints and bounds. The polymer feed rate in the previous dosage formula used in feed forward regulation had a maximum bound of 1 l/s. Since this constraint was in place, the limitation for the MPC would be the same. Therefore, the bounds are defined as shown in equation 4.37.

$$0\text{l/s} \leq u_k \leq 1\text{l/s} \quad (4.37)$$

There are no other constraints besides the bounds, unless considering physical constraints such as the inertia of the system. For instance, the regulation valve for polymer cannot open faster than 5% per second, but since the MPC calculates a new setpoint every minute, valve inertia and other types of system restrictions are neglected.

4.7.1 MPC program

The program running the MPC was written in Python, utilizing the minimize function from SciPy to minimize the objective function. To create a well-structured program, a class for MPC was developed. This class utilizes a separate model class to estimate the model based on previous values of y and x . The MPC class was designed in a way that only requires the use of one method: "calculate". This method executes the minimize algorithm. In Code 4.6, this method is shown, where the bounds are defined and applied.

Code 4.6: Code for minimizing the objective function

```

def calculate(self):
    """Perform optimization and update control input."""
    b = np.zeros((self.n, 2))

    for i in range(self.n):
        b[i][0] = 0
        b[i][1] = 1

    x0 = self.u
    """Calculate the n next setpoint"""
    res = minimize(self.ObjectiveFunction, x0, method='SLSQP', bounds=b)
    #If not able to find minima print it
    if not(res.success):
        print("Problem")
    self.U = res.x[0]

    return res

```

To develop and derive the appropriate objective function, a dedicated method was implemented within the MPC class, as illustrated in Code 4.6. This method accepts the parameter n , representing the number of future time samples to predict. It proceeds to construct a vector containing setpoints and predictions by using the model class to estimate the subsequent n samples based on the input signal u . Subsequently, it calculates the change in u over the time span n , culminating in the calculation of the objective function, which is then returned. Refer to Code 4.7 for the implementation details.

Code 4.7: The objective function method

```

def ObjectiveFunction(self, u):
    """Objective function for MPC optimization."""

    #Making a vector for the setpoint
    ref = np.ones(self.n) * self.SP

    #Estimating
    y = self.RunModel(u)

    #Finding the error
    e = ref - y
    #Setting the previous used u into u
    u = np.insert(u, 0, self.U)
    #Finding the change in u
    deltaU = u[:self.n] - u[1:]

    #Calculating objective function
    J = (e.T * self.Q).dot(e) + \
        (deltaU.T * self.P).dot(deltaU) + \
        (u.T * self.R).dot(u)

    return self.J

```

The entire program for the development and testing of the MPC is located in Appendix: GitHub Repository. This program can be found within the directory "MPC testing/MPC.py".

4.8 Platform for running estimation and MPC

Multiple approaches exist for running an MPC and estimator. However, the chosen platform must seamlessly communicate with the existing SCADA system. For Veas, this SCADA system is provided by ABB and comprises AC800 PLCs and the overall SCADA system: xA800.

One considered solution for communication involved using the OPC DA protocol between a Python program running the estimation and MPC. However, Python do not support OPC DA well. Therefore, it was decided to run the system not from the SCADA level but at the PLC level using the Profibus protocol. This generated possibilities for using, for example, simple compute modules or more advanced systems.

The RevPi Connect S with Gateway Profibus module was chosen. This allowed a Python program to simultaneously convert Profibus signals from the PLC and vice versa while performing the main tasks: MPC and dry solids estimation.

Hence, the platform's architecture is divided into five main components: the setup of RevPi, accessing Profibus signals in Python, estimation, MPC and finally, alarms.

4.8.1 Setup

First and foremost, several primary tasks had to be completed before configuring RevPi with its modules. This process can be reviewed by clicking [here](#) [37].

After completing all configurations, the next step was to initiate the command shown in Code 4.8. This command transitioned the system from the command line interface to a graphical interface.

Code 4.8: Command for starting graphical interface

```
sudo startx
```

Subsequently, it was necessary to select the modules that would run alongside RevPi Connect S. In this scenario, only a Profibus Gateway was utilized. To configure this, the integrated program PiCtory was launched. PiCtory provides an overview of all available modules for RevPi, enabling users to select the physically connected modules. Upon selecting the Profibus Gateway, it was essential to adjust the configuration from 20 input bytes and 20 output bytes to 512 input and output bytes. Figure 4.9 illustrates a screenshot of the PiCtory program with the correct setup.

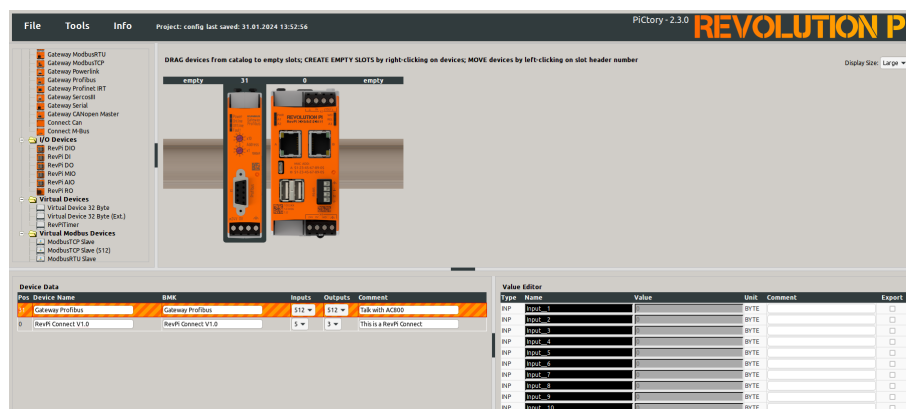


Figure 4.9: PiCtory program for setups of RevPi modules

Once the RevPi was correctly configured, the next step involved setting up the PLC to act as the master for Profibus communication. To ensure proper data reading by the PLC, a GSD file needed to be loaded and configured accordingly. The GSD file can be accessed from the appendix labeled GitHub Repository, and the specific setup used for this master's thesis is detailed in the appendix: GSD file setup for Profibus.

4.8.2 Access of Profibus in Python

To extract bytes from Profibus communication within a Python script, the RevPi-specific library "revpimodio2" was used. This versatile library provides various functions for interacting with RevPi modules. To read bytes, the method "RevPiModIO" was utilized, allowing for the retrieval of bytes by specifying the desired byte number. Table 4.9 illustrates the usage of this module for byte reading.

Code 4.9: Retrieve data from the Profibus communication for byte number 1.

```
# Import the RevPiModIO module for communication with RevPi
import revpimodio2

# Create a RevPiModIO object with automatic refresh of I/O values
rpi = revpimodio2.RevPiModIO(autorefresh=True)

# Read the value of Input__1 and store it in the byte1 variable
byte1 = self.rpi.io.Input__1.value
```

In this configuration, it's important to recognize that the GSD file settings specify that the majority of signals should be interpreted as Real values from the perspective of the PLC. These Real values correspond to floating-point numbers in Python, which occupy 32 bits of memory. Therefore, translating these Real values into bytes entails reading four separate bytes from the Profibus card in Python and then assembling them into a float. All code related to this byte-to-float conversion process can be accessed in the appendix under [GitHub Repository](#), specifically within the directory `MasterThesis/RevPiCode/Libs/ProfiBusCom.py`. An illustrative example of this conversion process is provided in Code 4.10.

Code 4.10: Setting 4 bytes into one float

```
# Import necessary modules for communication with RevPi and for byte manipulation
import revpimodio2
import struct

# Create a RevPiModIO object with automatic refresh of I/O values
rpi = revpimodio2.RevPiModIO(autorefresh=True)

# Read the values of Input_4 to Input_7 and store them in separate variables
byte1 = rpi.io.Input__4.value
byte2 = rpi.io.Input__5.value
byte3 = rpi.io.Input__6.value
byte4 = rpi.io.Input__7.value

# Concatenate the bytes in little-endian order to form a byte string
byte_string = bytes([byte4, byte3, byte2, byte1])

# Unpack the byte string as a float using little-endian byte order
float_value = struct.unpack('<f', byte_string)[0]
```

4.8.3 Estimator

To estimate %DS, a dedicated module named ModelEstimator was developed. This module includes everything from time delays to the utilization of the model. The entire codebase can be found in the appendix: [GitHub Repository](#), located at MasterThesis/RevPiCode/Libs/ModelEstimation.py. The module consists of two classes, where one class serves as the primary interface, enabling easy usage with just a few simple lines, as demonstrated in Code 4.11. The program requires instantaneous values of the input variables for the model and returns the instantaneous value of %DS. The second class is more advanced and is used to estimate and store current values for later use when the correct time delay has been achieved.

Code 4.11: Simple estimation script

```
# Import the ModelEstimation module
import ModelEstimation

# Create an instance of the EstimateModel class from ModelEstimation module
ME = ModelEstimation.EstimateModel()

StartEstimation = True

# Infinite loop to continuously perform estimation
while True:
    # Check if StartEstimation flag is True
    if StartEstimation:
        # If True, initialize the model with the first 5 samples
        # for initial conditions
        ME.StartUp5x(x)
        StartEstimation = False
    else:
        # Perform estimation using the current input x and update the model
        y = ME.Estimate(x)
```

4.8.4 MPC

To facilitate the execution of MPC in a manner similar to model estimation, a dedicated module was also developed. However, this module requires model estimation to find the correct setpoint for the polymer. The complete program is available in Appendix [GitHub Repository](#) at MasterThesis/RevPiCode/Libs/MPC.py, but Code 4.12 illustrates how the MPC module can be employed.

The RunMPC method requires three inputs: the system state over the last 10 minutes, the state of what y has been over the last 10 minutes, and the desired setpoint for %DS.

Code 4.12: How to use MPC module

```
# Import the MPC module
import MPC

# Create an instance of the StartMPC class from MPC module
TRS5mpc = MPC.StartMPC()

# Infinite loop to continuously perform estimation
While True:
    u, SPFound = TRS5mpc.RunMPC(x_matrix, y_array, SP)
```


4.8.5 Alarms

During the development of this program intended to run on the RevPi, potential issues were identified. For instance, if the RevPi stops functioning or crashes, it would no longer be possible to receive new setpoints for polymer dosing. To reduce this risk, a watchdog was implemented, which is simply a boolean variable in the Python program that toggles on and off every 3 seconds. This ensures that if either the Python program hangs or the entire RevPi stops working, the watchdog will trigger. This watchdog signal is sent to the PLC via Profibus, where the PLC generates an alarm if the state has not changed after 10 seconds.

Another alarm sent to the PLC is triggered if the MPC fails to find a setpoint. In Table 4.12, the variable "SPFound" is used. This variable is set to True if the MPC found a setpoint for polymer dosing. It is then sent back to the PLC, where if it goes low, an alarm is activated, allowing operators to validate and potentially revert to the previous regulation of polymer dosing to the drum screen.

One final issue identified was if Veas experiences a power outage, one would need to manually start the Python program in the facility. This was resolved by setting the Python program as one of the startup applications for the RevPi. This means that if there is a power outage at Veas, the MPC will automatically restart as soon as power is restored.

4.8.6 Sequence diagram

For illustrating how the modules communicate with each other, a simplified sequence diagram was designed to provide a primary focus on the sequence of events. This diagram, depicted in Figure 4.10, presents the interaction between the modules and the order in which actions occur.

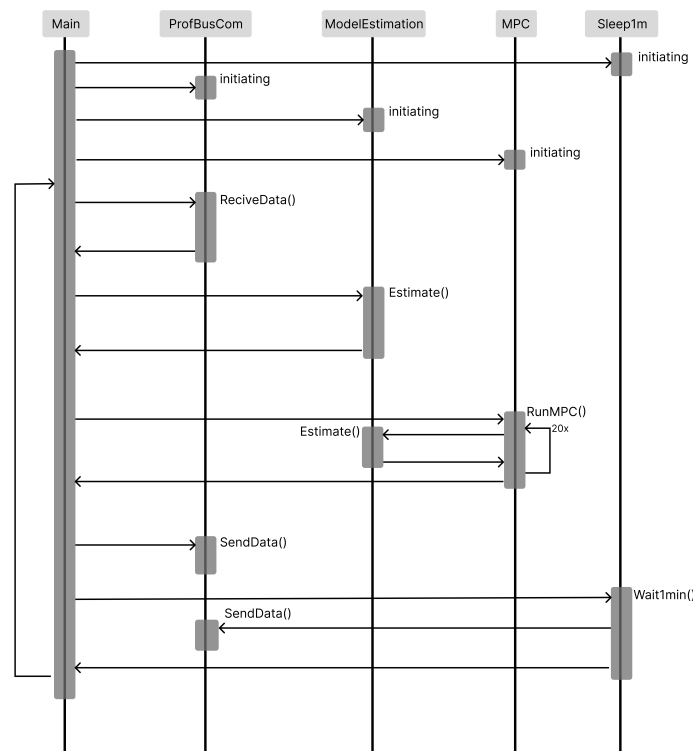


Figure 4.10: Simplified sequence diagram

RESULTS AND DISCUSSION

This chapter presents the results obtained through this project. Given the diversity of topics covered, the results will be divided into five main sections: Laboratory Samples, PCA, Interpolation, Model Creation, and MPC.

5.1 Laboratory sample results

After performing physical tests on the dry solids extracted from the drum screen a total of 20 times over a period of 10 days, the obtained results revealed variability, as illustrated in detail in Table 2.

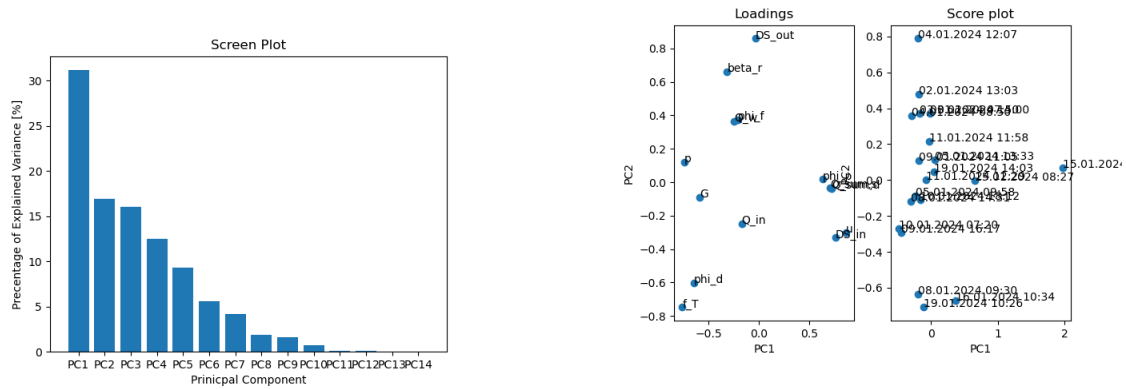
Table 2: Laboratory results for all samples taken. If a "-" is indicated in the table, it signifies that the corresponding test was not conducted by the laboratory for that particular sampling.

| Time | Dry solids [%DS] | Loss of ignition [%VS] |
|------------------|------------------|------------------------|
| 02.01.2024 13:03 | 9.34 | 84.58 |
| 03.01.2024 07:50 | 9.17 | 83.57 |
| 03.01.2024 14:00 | 9.37 | 82.77 |
| 04.01.2024 08:50 | 9.44 | 84.09 |
| 04.01.2024 12:07 | 8.93 | 83.58 |
| 05.01.2024 09:58 | 7.33 | 82.49 |
| 05.01.2024 13:33 | 8.81 | 82.16 |
| 08.01.2024 09:30 | 6.88 | 83.08 |
| 08.01.2024 14:31 | 7.47 | 82.49 |
| 09.01.2024 11:05 | 7.11 | 83.65 |
| 09.01.2024 16:17 | 8.31 | - |
| 10.01.2024 07:20 | 7.06 | - |
| 10.01.2024 18:12 | 7.25 | - |
| 11.01.2024 11:58 | 8.99 | - |
| 11.01.2024 12:29 | 9.05 | - |
| 15.01.2024 08:27 | 10.08 | - |
| 15.01.2024 12:27 | 8.79 | - |
| 16.01.2024 10:34 | 6.53 | - |
| 19.01.2024 10:26 | 5.74 | 83.39 |
| 19.01.2024 14:03 | 9.02 | 83.08 |

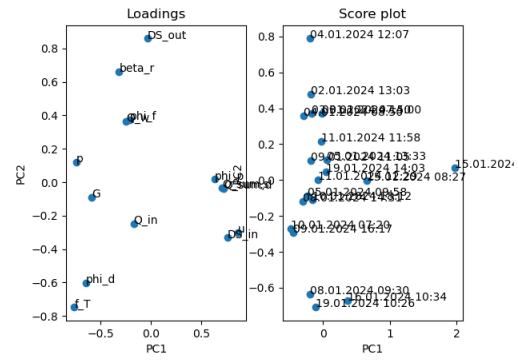
In table 2, the initial dry solids output from the drum screen was approximately 9%DS during normal operating periods, up until January 5th. After a weekend without sampling, measurements resumed January 8th, revealing a lower dry solids content of 6.88%DS, which continued until January 10th. This decrease was attributed to a period of poor sludge settling experienced by Veas, where the sludge did not settle properly in the SED basins but continued to flow through the water treatment process. Consequently, the sludge reaching the drum screen was much less concentrated, potentially requiring a higher dosage of polymer for effective separation of water from the sludge.

5.2 Principal component analysis

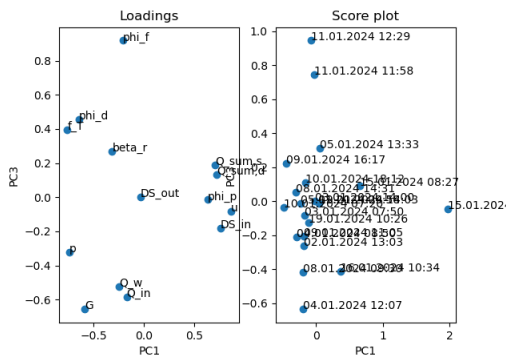
The results of the PCA, including the transformed data, are depicted in Figure 5.1. Figure 5.1a displays the screen plot, indicating the number of PC's needed to fully describe all variables. It shows that 14 PC's are necessary for a complete representation of all variables. However, to capture approximately 75% of the dataset, only 4 PC's are required. Therefore, plots 5.1b, 5.1c, and 5.1d showcase these four PCs, with PC₁ consistently appearing on the x-axis.



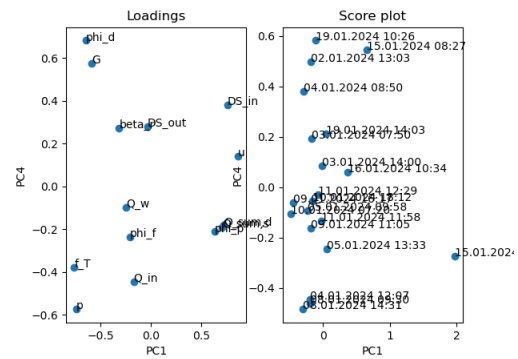
a: Screen plot, where the 4 first PC capture 75%



b: PC₁ plotted against PC₂, it is evident that PC₂ provides a significant description of the variation in DS_{out}



c: PC₁ plotted against PC₃, it is evident that PC₃ do not provides any significant description of the variation in DS_{out}



d: PC₁ plotted against PC₄, it is evident that PC₄ provides some description of the variation in DS_{out}

Figure 5.1: PCA is used to find variables that describe the variation in the output DS_{out}

5.2.1 Dimensionality reduction using PCA

Since PCA was utilized to analyze which variables best describe the quantity of dry solids discharged (DS_{out}), one can examine which variables occupy the same region on the loading plot. To utilize the loading plots effectively, it is necessary to first determine which PCs describe DS_{out} . According to Figure 5.1, these are PC₂ and PC₄. Therefore, Figure 5.2 displays a coordinate plot of these two PCs against each other.

Examining Figure 5.2, it's evident that three variables cluster near the origin: $Q_{sum,s}$, $Q_{sum,d}$, and ϕ_p . Their proximity to the origin implies they have minimal influence on explaining PC₂ or PC₄ and, consequently, cannot effectively describe DS_{out} .

Furthermore, DS_{out} is better represented in PC₂ than PC₄, making it more practical to focus on significant variables in PC₂. These include f_T , ϕ_d , and β_r . ϕ_d represents the drum

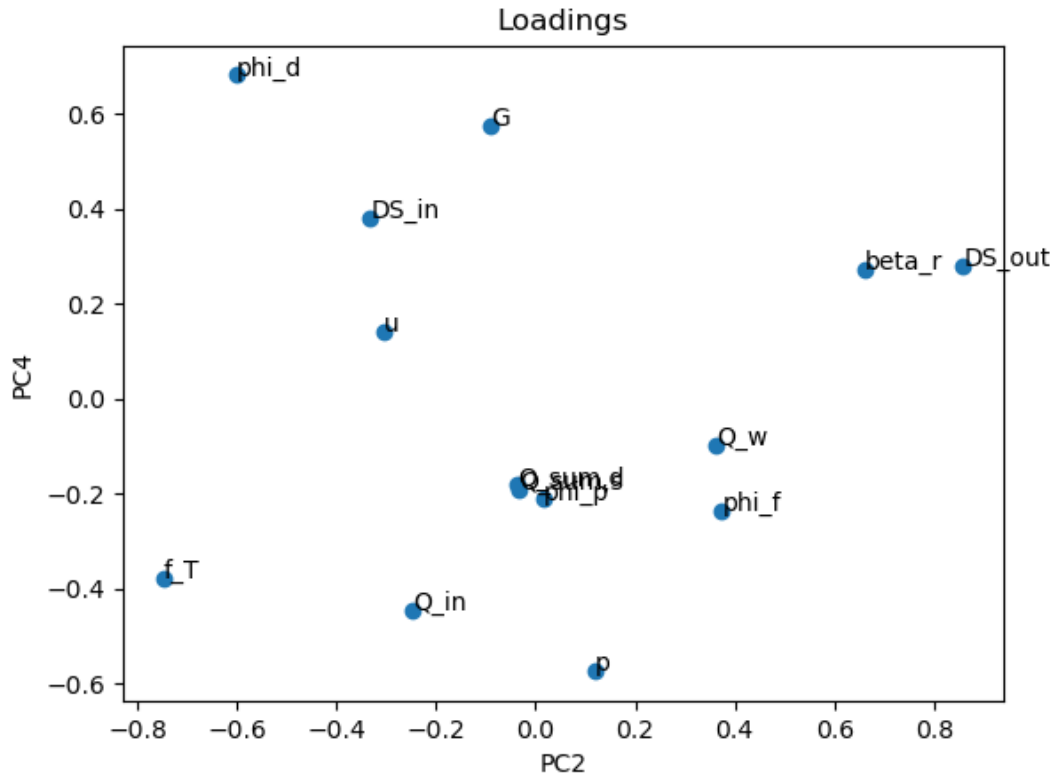


Figure 5.2: PC2 against PC4

screen's rotational speed, which remains constant and is negligible. Similarly, β_r is the regulating valve for sludge input, consistently fully open during this analysis, also rendering it negligible.

Regarding PC4, G (LoI) is highly described. However, G has remained relatively constant throughout the experiment and necessitates laboratory testing, making it impractical for maintaining a continuous model for dry solids.

Thus, six variables effectively describe DS_{out} according to the PCA. These variables are:

- DS_{in} = Dry solids into drum screen
- f_T = The average turbidity in all SED basins
- Q_{in} = Flow of sludge into drum screen
- Q_w = Flow of water out of drum screen (Reject water)
- u = Polymer dosage to drum screen
- p = Pressure in the polymer string into drum screen
- ϕ_f = Motorized stirring in the flocculation tank before the drum screen

By doing this, the dimensions have been effectively reduced from 14 to 6. For more information on the meanings of the various variables, go to the appendix labeled [Symbol list and TAG names](#).

5.3 Interpolations results

After interpolating the step response based on the laboratory results obtained on January 19, 2024, two interpolation formulas were derived: one for the decrease in %DS and one for the increase in %DS. Equation 5.1 represents the interpolation during the decreasing polymer stage, while Equation 5.2 represents the interpolation when the polymer setpoint is restored to normal.

$$DS_{decrease} = 0.0042t^2 - 0.3736t + 12.598 \quad 10 \leq t \leq 38 \quad (5.1)$$

$$DS_{increase} = -0.1319t^2 + 0.652t - 248.28 \quad 38 \leq t \leq 45 \quad (5.2)$$

After merging these two interpolations, as depicted in Figure 4.6, the outcomes appeared non-viable. This mismatch arises because laboratory results indicate variations throughout the day rather than complete static conditions. Therefore, as explained in the methods, small variations were added after changes in the input to generate a more realistic and dependent model of the input variables. The result of this can be seen in Figure 5.3. This figure displays both the input and output, where the bottom graph represents the output (DS_{out}).

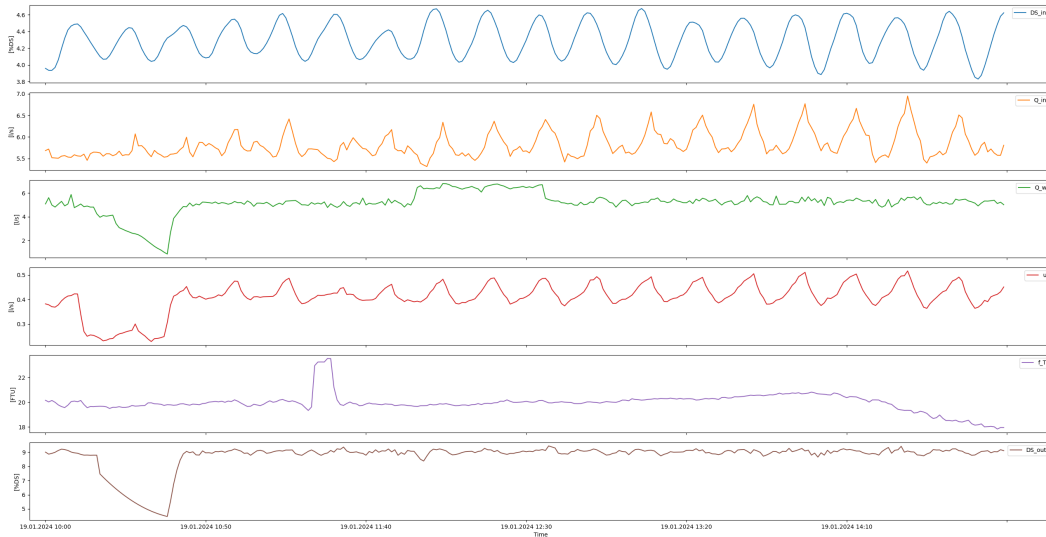


Figure 5.3: Interpolation of dry solids with input variables

Attempts were made to create models based on the results of this interpolation in Figure 5.3. However, the issue with this approach was that the models generated from it failed to estimate when the quantity of dry solids discharged from the drum screen was low, due to poor sludge settling in SED composition as presented earlier. Therefore, an additional day was included in the interpolation where the dry solid value from the drum screen was low. The interpolation was conducted in the same manner, and the results of all interpolations combined can be seen in Figure 5.4.



Figure 5.4: Interpolation of dry solids with input variables over 2 days

5.4 Model Creation

Three distinct models were developed and tested: an ARX model, a NARX model, and a mechanistic model. Both the ARX and NARX models were configured with a sampling time of 1 minute. This section presents the results obtained from testing these models.

5.4.1 NARX model

After generating a series of NARX, and NARMAX models, it was evaluated that the best model SysIdentPy managed to generate was a NARX model using the following inputs:

- $DS_{in} = x_1$.
- $Q_{in} = x_2$.
- $DS_p = x_3$
- $u = x_4$.
- $f_T = x_5$.

The model employs a 3rd-degree NARX polynomial, incorporating five inputs alongside its own value one time sample back in time. Upon running SysIdentPy, a table containing the model results is generated, facilitating practical application. This table, illustrated in Table 3, provides valuable insights into the model's performance and parameters.

Table 3: NARX model training and evaluation results

| | Regressors | Parameters | ERR |
|---|--------------------------|-------------|------------|
| 0 | $y(k-1)$ | 1.0239E+00 | 9.9983E-01 |
| 1 | $x_2(k-1)x_1(k-4)y(k-1)$ | -1.4126E-03 | 2.0525E-06 |
| 2 | $x_4(k-3)x_1(k-1)^2$ | 1.6950E-01 | 2.9189E-06 |
| 3 | $x_5(k-1)y(k-1)^2$ | -3.3035E-04 | 2.6460E-06 |
| 4 | $x_5(k-2)y(k-1)^2$ | 2.7617E-04 | 1.8902E-06 |

Incorporating these results into a model yields the representation presented in eq. 5.3.

$$\hat{y}_k = 1.0239y_{k-1} + (-1.4126 \times 10^{-3})x_{2,k-1}x_{1,k-4}y_{k-1} + (1.6950 \times 10^{-1})x_{4,k-3}(x_{1,k-1})^2 - (3.3035 \times 10^{-4})x_{5,k-1}y_{k-1}^2 + (2.7617 \times 10^{-4})x_{5,k-2}y_{k-1}^2 \quad (5.3)$$

It is noteworthy that x_3 is not actively utilized in the model, but its inclusion is crucial for accurate estimation since it represents the dry solids portion of the polymer dosage, which remains relatively constant. Therefore, x_3 was multiplied by x_4 before model creation. Incorporating this adjustment results in a revised model as shown in Equation 5.4.

$$\hat{y}_k = 1.0239y_{k-1} + (-1.4126 \times 10^{-3})x_{2,k-1}x_{1,k-4}y_{k-1} + (1.6950 \times 10^{-1})x_{4,k-3}x_{3,k-3}(x_{1,k-1})^2 - (3.3035 \times 10^{-4})x_{5,k-1}y_{k-1}^2 + (2.7617 \times 10^{-4})x_{5,k-2}y_{k-1}^2 \quad (5.4)$$

Additionally, referring to the appropriate symbols from the appendix: Symbol list and TAG names. The comprehensive model is represented with the correct symbols in Eq. 5.5.

$$\hat{DS}_{out,k} = 1.0239DS_{out,k-1} + (-1.4126 \times 10^{-3})Q_{in,k-1}DS_{in,k-4}DS_{out,k-1} + (1.6950 \times 10^{-1})u_{k-3}DS_{p,k-3}(DS_{in,k-1})^2 - (3.3035 \times 10^{-4})f_{T,k-1}DS_{out,k-1}^2 + (2.7617 \times 10^{-4})f_{T,k-2}DS_{out,k-1}^2 \quad (5.5)$$

A visualization of the NARX model using a block diagram can be seen in Figure 5.5.

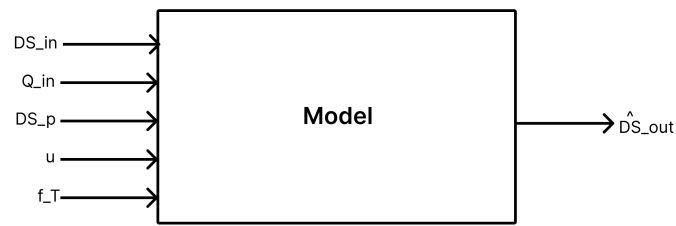


Figure 5.5: NARX model using a block diagram

Testing model against validation set

The initial evaluation involves testing the model's performance against the validation dataset. This assessment was carried out, and the outcomes are illustrated in Figure 5.6.

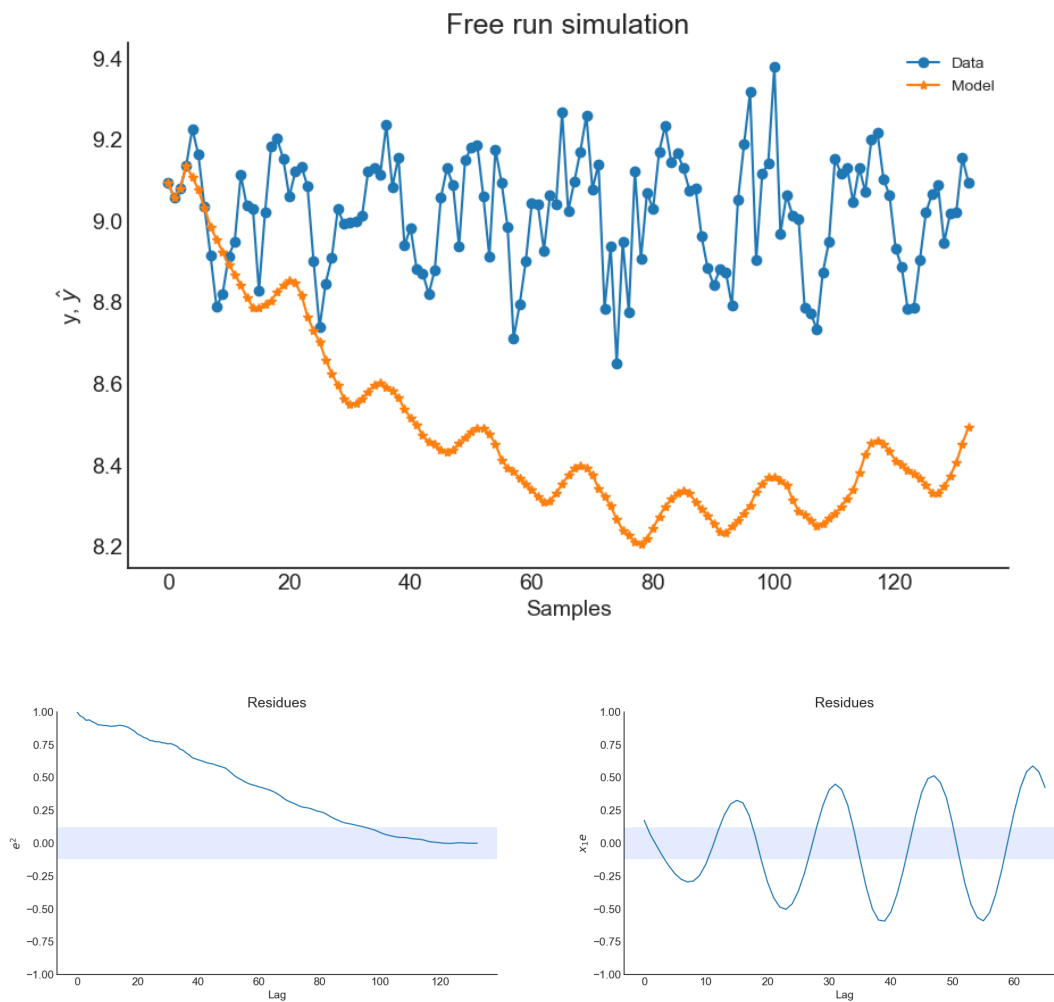


Figure 5.6: A simulation from the validation data and the residues from over all time delay used

Upon reviewing the results from the validation dataset, the outlook appears somewhat unpromising. The estimated value initially drops significantly but starts to rise again towards the end. Although it may not seem like a robust model at first glance, it's possible that this estimation reflects a more realistic value compared to the initial interpolation. Therefore, a test was also conducted on the entire interpolation dataset to observe how the model would estimate throughout. This can be seen in Figure 5.7.

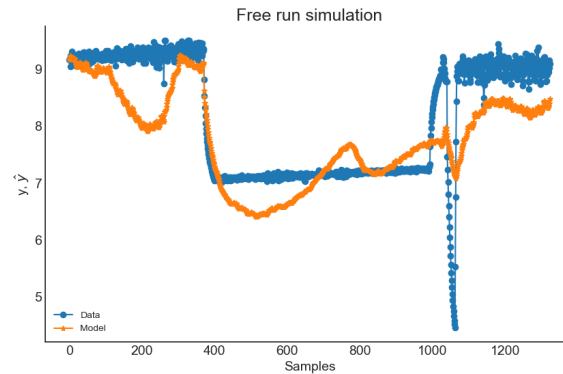


Figure 5.7: Simulation for all data from interpolation

These results appear more promising, but they cannot be included in the evaluation since 90% of the estimated data belongs to the training set used to construct the model.

Estimating values over a 2-week period.

The next test conducted to assess the model's usability was to estimate values over a two-week period. This two-week period corresponds to the same timeframe as the laboratory analysis, enabling a comparison between the model's estimates and the laboratory results. The comparison is depicted in Figure 5.8.

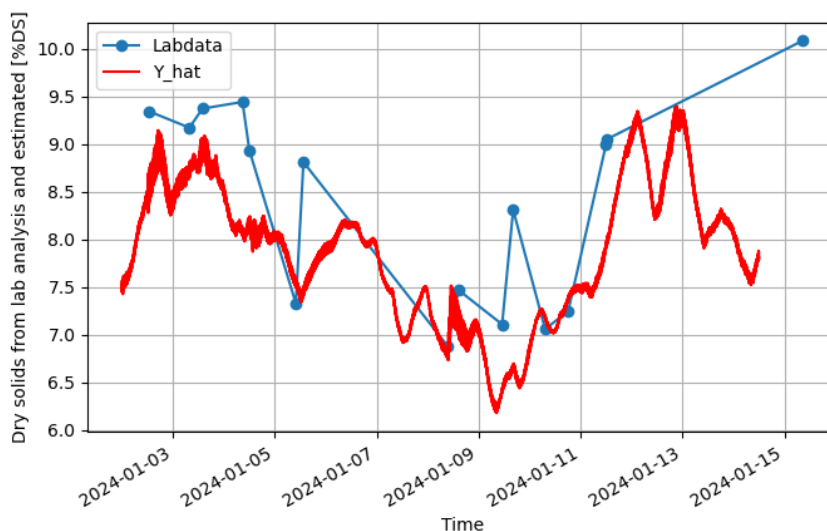


Figure 5.8: Plot of Narx model over a two-week period against laboratory analysis points

The outcome reveals that while it doesn't precisely align with most of the sample points, it does track the general trend of the data. Notably, during the middle phase where the dry solids quantity decreases, mirroring the observed trend in the samples, it struggles to accurately estimate minor fluctuations over brief intervals.

5.4.2 ARX model

After fitting the model to the interpolated dataset, the resulting model is as shown in equation 5.6. This includes 1 input parameter, 4 noise parameters, and an output.

$$\hat{y} = -\frac{y_{k-1}}{\tau} + k_1 u_{k-4} + k_2 DS_{in,k-5} + k_3 Q_{in,k} + k_4 Q_{w,k} - k_5 f_{T,k-10} \quad (5.6)$$

In equation 5.6, there are a total of 6 parameters that needed to be determined, and these are presented in eq. 5.7.

$$\tau = 68201174.81 \quad k_1 = 27.59 \quad k_2 = 1.41 \quad k_3 = 0.0197 \quad k_4 = 0.5 \quad k_5 = 0.082 \quad (5.7)$$

To test the functionality of the model, the same dataset used for the NARX model was used. This test dataset spans a two-week period during which laboratory tests were conducted to evaluate how well the model performs compared to the lab results. The results of this test can be seen in Figure 5.9.

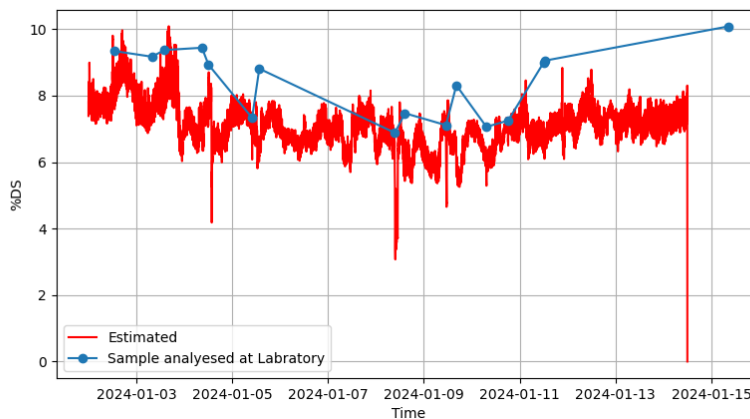


Figure 5.9: Plot of ARX model over a two-week period against laboratory analysis points

Based on the results, it appears that the model generally underestimates most of the data points where laboratory tests were conducted. Additionally, there seems to be more noise in the results compared to the NARX model.

5.4.3 Mass balance model

The last model tested on the two-week dataset was a mechanistic model based on the principles of mass balance. The results from this mass balance can be observed in Figure 5.10.

In the figure, the mass balance model exhibits numerous spikes ranging from 20 to 0. These spikes are a result of the constraints imposed on the model within the code. Without these constraints, these spikes would tend towards infinity and negative infinity. This is primarily due to the multiplication and division by very small numbers, which can occur when the suspended solids leaving the drum screen in the reject water are low, with an average value of approximately 100 mg/L. This value can be converted to approximately 0.01%DS using equation 4.16. It's worth noting that this average was derived from 5 laboratory analyses of the reject water originating from the drum screen and does not fall within the expected range of 20-50 mg/L as previously assumed.

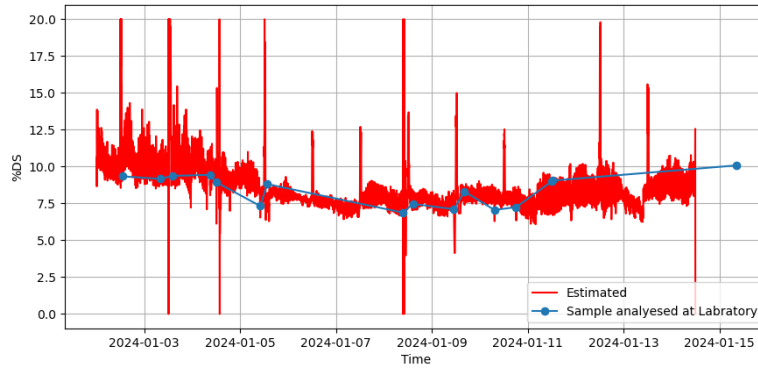


Figure 5.10: Plot of Mass balance model over a two-week period against laboratory analysis points

5.4.4 Time delay

To assess the computed time delay, a test was conducted utilizing the NARX model, which had previously identified its own time delays.

Incorporating calculated time delay into NARX

Since the data used to train the NARX model is subject to uncertainty, there is little guarantee that the time delay estimated by the model is accurate. Therefore, the NARX model was modified to incorporate the calculated fluid dynamics-based time delays instead of relying solely on its internally estimated delays.

Thus, by incorporating the calculated time delays, the model is represented as shown in equation 5.8.

$$\hat{y} = 1.0239y_{k-1} + (-1.4126 \times 10^{-3})x_{2,k-1}x_{1,k-7}y_{k-1} + (1.6950 \times 10^{-1})u_{k-6}x_{3,k-6}(x_{1,k-7})^2 - (3.3035 \times 10^{-4})x_{5,k-9}y_{k-1}^2 + (2.7617 \times 10^{-4})x_{5,k-10}y_{k-1}^2 \quad (5.8)$$

A test was conducted to compare the model with NARX time delay and one with delays calculated from fluid dynamics. This test is illustrated in figure 5.11.

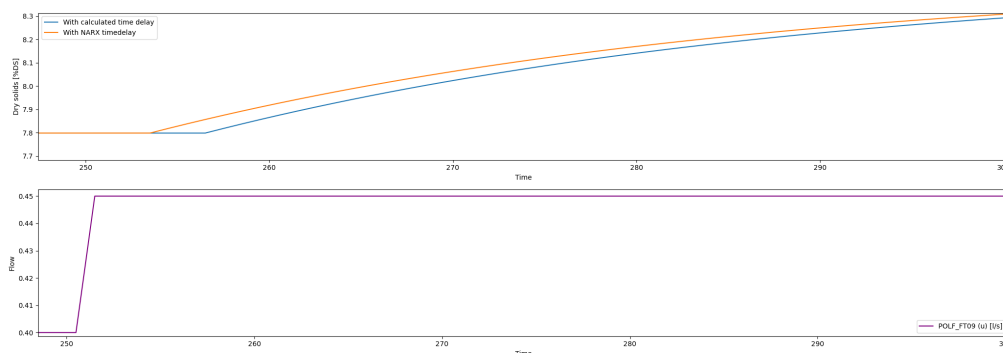


Figure 5.11: Test of comparing the models

5.4.5 Mixer in tank

Up to this point, little attention has been given to the presence of a mixer in the flocculation tank. This factor shortens the time delay within the tank, making it impractical to calculate it in the same manner. Therefore, a step response was conducted on the polymer dosage instead. By halving the dosage, it was observed when the reject water from the drum screen began to decrease. Figure 5.12 illustrates this step response.

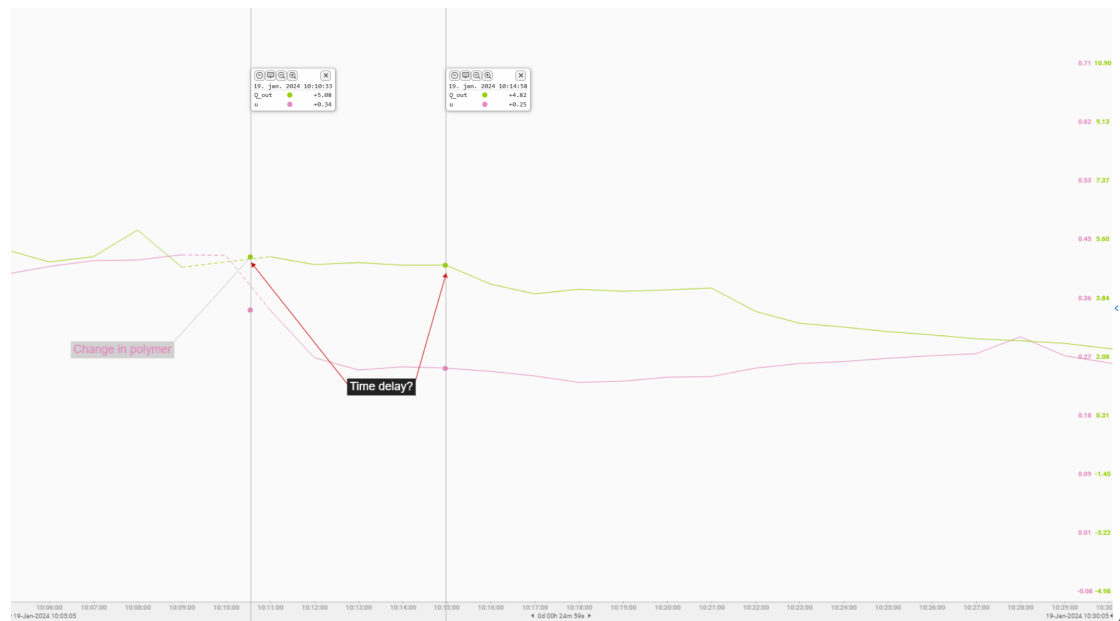


Figure 5.12: Step response to find time delay

Upon analyzing the results following the step response, it is evident that the time delay is not 6 minutes, but rather closer to 4 minutes for the flocculation tank. Therefore, the time delay is revised as shown in equation 5.9.

$$\Delta t_f \approx 4min \quad \Delta T_D \approx 5min \quad \Delta t_F \approx 10min \quad (5.9)$$

5.4.6 Testing models on the real system

Following the evaluation of three distinct model types on simulated processes, the subsequent phase involved testing these models on the live process utilizing real-time data. The models subjected to testing were the mass balance model and the NARX model. To ascertain their precision, five samples were collected over a span of 40 minutes. The outcomes of this examination are depicted in Figure 5.13.

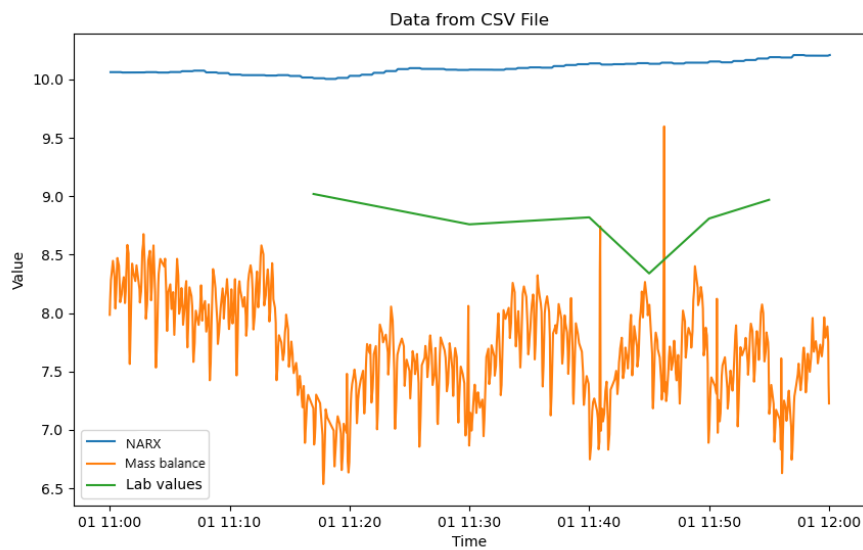


Figure 5.13: Lab results compared to the models

Here, it can be observed that the mass balance model (orange) exhibits significant noise and consistently lies between 0.5 and 1 %DS below the actual laboratory measurements. Meanwhile, the NARX model (blue) consistently overestimates the values.

The mass balance model consistently underestimates laboratory values and yields noisy results due to its inability to account for polymer input. Consequently, it is dismissed after a review of these findings.

Furthermore, the drum screen undergoes flushing once a minute. However, Veas does not measure this water volume, complicating the accurate estimation of dry solids content during these flushes.

During the testing period when the lab samples were taken, there was a snow melting period in Oslo. This often causes significant issues for the sludge process at Veas since the incoming sludge has low DS_{in} , cold and consists of more salt than usual. Therefore, a two-week-long test was conducted where the model was tasked solely with calculating the dry solids content at any given time. Despite the NARX model overestimating the actual dry solids content, various tests continued. During this test, it was found that the model has substantial limitations regarding the possible inputs. These limitations are depicted in equation 5.10.

$$DS_{in} \geq 2.2\%DS \quad 5.2l/s \leq Q_{in} \leq 8l/s \quad (5.10)$$

Another limitation of the model was also discovered: when both the dry solids content inflow is below 2.5%DS and the flow rate into the drum screen is below 5.4 l/s simultaneously, the model struggles to provide accurate estimates. This is because the model multiplies the dry solids content inflow by the inflow rate. Consequently, if both values become unusually low, the model will fail to estimate accurately.

5.5 MPC results

To enhance MPC performance, defining the prediction horizon is a crucial step. Since the data collection interval for the model and MPC is in minutes, the prediction horizon was initially set at 10 minutes. However, it was later extended to 20 minutes to account for the time delay of specific parameters in the model. This adjustment enabled the MPC to precisely compute the optimal u for the subsequent 20 minutes, with only the first setpoint implemented before recalculating 20 new setpoints, every minute.

Once the prediction horizon was established, tuning the MPC required defining the values of Q , P , and R . Veas aimed for minimal fluctuations in the polymer supplied to the process while enabling a slower response of the process value to setpoint changes. Furthermore, to minimize polymer consumption, R was fixed at 1, as it was deemed to have the least significance. The precise values selected for Q , P , and R are depicted in equations 5.11. Here, P was assigned the highest weighting, reflecting Veas's priority of ensuring minimal variation in control input.

$$Q = 16, \quad P = 33, \quad R = 1 \quad (5.11)$$

5.5.1 Setup of the control problem

Figure 4.1 outlines the proposed construction of the control system. The concept involved a model-based controller determining polymer dosing setpoints, which would then be transferred to a PI controller for polymer dosing implementation. Moreover, an estimator would be responsible for estimating dry solids exiting the drum screen. The model-based controller was realized as an MPC, while the estimator was a model that remained uncorrected by actual values, for instance through the use of a Kalman filter. Figure 5.14 showcases the actual execution of the control problem.

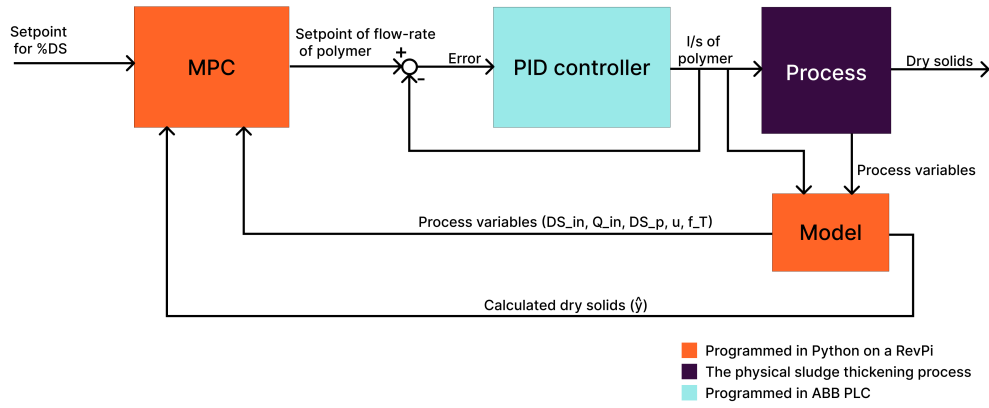


Figure 5.14: Visualization of MPC

5.5.2 Choosing model

To implement the MPC, a decision must be made regarding the model to utilize. In this scenario, only two models are viable options: either the ARX model or the NARX model. This is because the mechanistic model based on mass balance lacks adequacy due to its failure to incorporate the manipulated variable to be controlled as an input.

By comparing the performance of the test dataset, as shown in Figure 5.8 for the NARX model and in Figure 5.9 for the ARX model, it appears that the NARX model is closer to several data points compared to the ARX model. Additionally, there seems to be less noise in the NARX model's results. Therefore, it was decided to proceed with testing the MPC using the NARX model.

5.5.3 Simulation

To test the MPC, a simulated process was utilized to generate input values for the system. The simulation covered a 2-day period. Upon examination, it was found that sinusoidal curves could effectively simulate all input variables. This observation originates from the oscillatory nature observed in the various process variables, which could be replicated using sine functions. Equations 5.12 to 5.16 illustrate the sinusoidal shape of the uncontrollable input signals.

$$DS_{in,k} = 4.1 + 0.15 \sin\left(2\pi \frac{1}{1320s} k\right) \quad (5.12)$$

$$Q_{in,k} = 6.2 + 0.28 \sin\left(2\pi \frac{1}{720s} k\right) \quad (5.13)$$

$$DS_{p,k} = 0.14\%DS \quad (5.14)$$

$$f_{T,k} = 20 + 4.5 \sin\left(2\pi \frac{1}{1440} k\right) \quad (5.15)$$

$$0 \leq k \leq (2 \times 24 \times 60)\text{min} \quad (5.16)$$

Using these values during a simulation, the MPC was tested for its ability to maintain setpoints and respond to step changes. Additionally, the time taken for each run was evaluated. Figure 5.15 illustrates the simulation results. It can be observed that the process variable oscillates slightly around the setpoint. This behavior is attributed to the higher weight assigned to the rate of change in the control input (P) compared to the setpoint deviation (Q). Consequently, the MPC prioritizes minimizing changes in the control input over precisely tracking the setpoint, as reflected in the methods.

Given the significant oscillations in the variables, assuming their constancy for the next 20 minutes in MPC prediction would be inaccurate. Hence, linear interpolation of x_1 , x_2 , and x_5 is conducted within the MPC. This entails analyzing the values over the past 10 minutes and interpolating a straight line between them to determine the next 20 values.

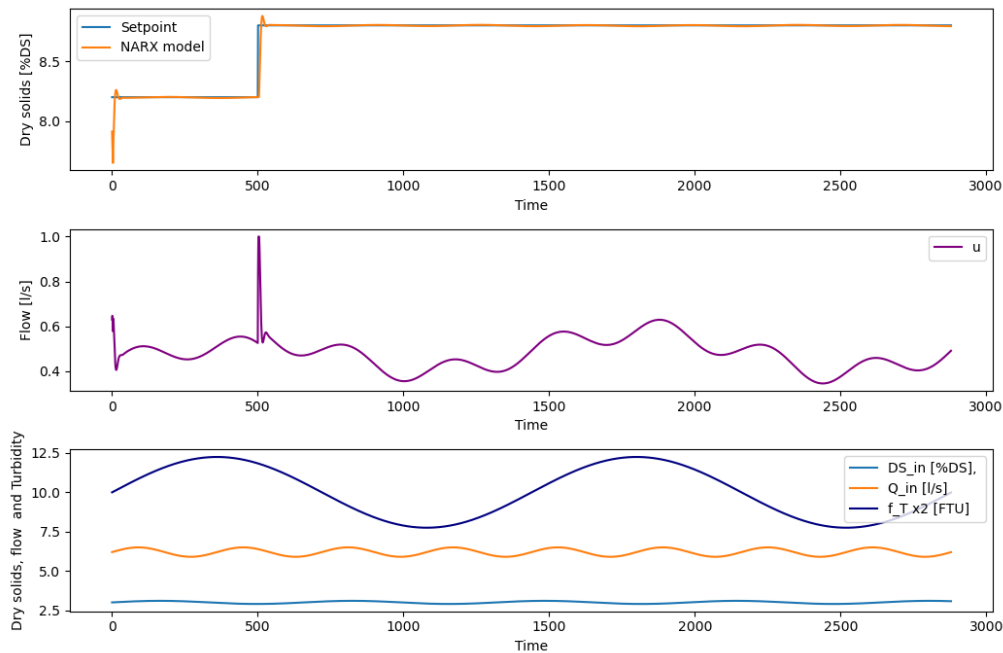


Figure 5.15: Plot of simulated test of MPC

By examining the polymer dosing, a spike is observed during setpoint changes. This behavior might not be physically feasible for the process. However, since the MPC calculates minute by minute and the PID controller would have one minute to adjust the correct dosage, this spike could potentially be achievable. To confirm this hypothesis, further tests would need to be conducted on the actual process.

Another consideration is whether the MPC computes values faster than the physical time set. For instance, if the process requires minute-by-minute updates, it's impractical for the MPC to take 1.02 minutes to compute the next control signal sample. Therefore, the time taken per MPC run was also logged, and the results are illustrated in Figure 5.16.

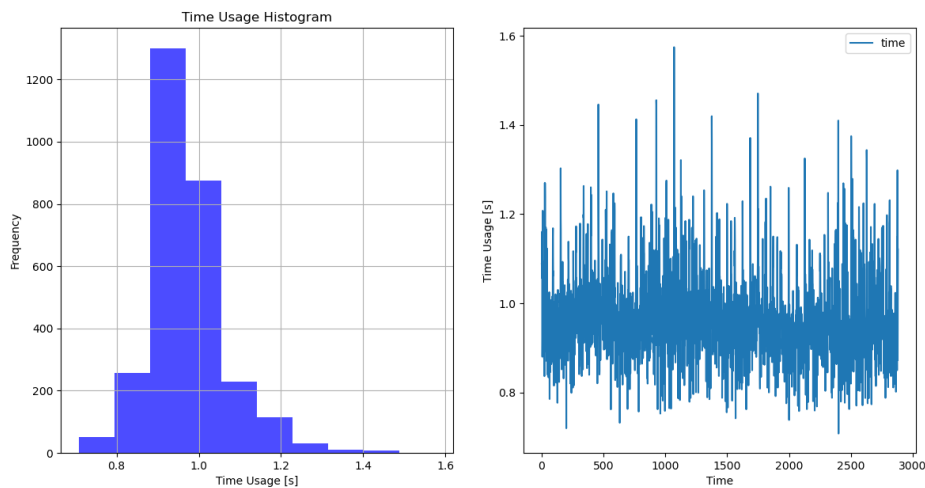


Figure 5.16: Time usage of the MPC in seconds

By studying the time usage, it is possible to observe that most of the time spent ranges between 0.9 and 1.1 second, but it has also taken up to 1.6 seconds to compute the next setpoints. This test was conducted on an HP laptop with an Intel i7 processor, so when transferring this to a RevPi, the calculation time can be expected to increase. However, since the MPC only needs to compute a new setpoint every minute, the RevPi will likely have more than enough time to calculate u . Specifically, the calculation would need to take 97% longer on the RevPi for it to be infeasible.

5.5.4 Testing MPC on the real system

After testing the models, it was time to assess whether the MPC accurately calculates the correct polymer dosage to the drum screen. This was done by activating the MPC and having it calculate setpoints that can be compared with the actual dosage. This comparison aims to determine the agreement between the operators controlling the polymer dosage and the MPC. The results of the initial test can be seen in Figure 5.17.

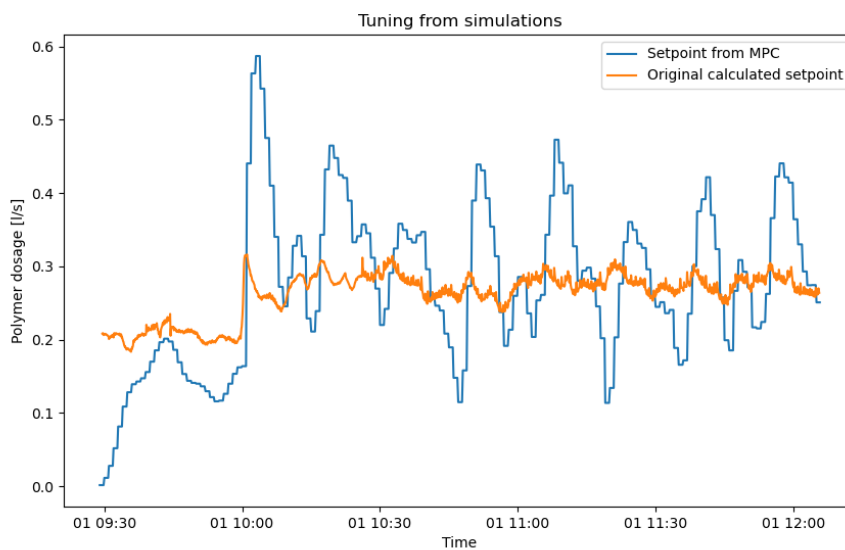


Figure 5.17: First results with bad "tuning" of MPC

Based on the results shown in Figure 5.17, it is evident that the setpoint from the PID controller, derived from the MPC, exhibits significant oscillation. This is because the simulation did not account for a high-frequency oscillation in the flow rate into the drum screen. Therefore, a new parameter for P was chosen, as shown in equation 5.17.

$$Q = 16, \quad P = 998, \quad R = 1 \quad (5.17)$$

Once these new parameters are chosen, the MPC demonstrates a more stable dosing behavior. The outcomes are illustrated in Figure 5.18, where the middle graph shows both the calculated polymer usage by the MPC and the actual polymer dosed by the operator.

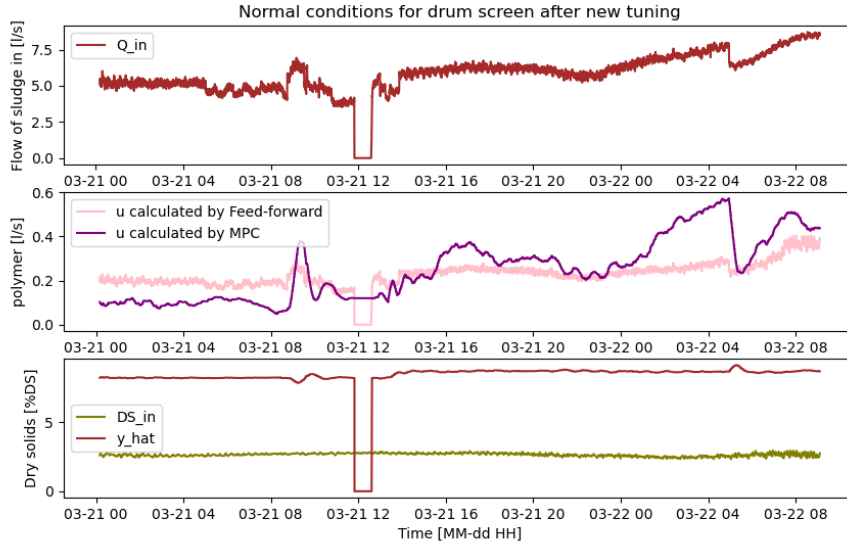


Figure 5.18: Normal conditions for MPC compared to operator dosage

Issues with the MPC arise when the inputs are outside the limitations of the model

When the inputs fall below the lower limits of the model, the MPC tends to set the polymer dosage to zero. To address this issue, the bounds within which the MPC is allowed to operate were adjusted. These are depicted in equation 5.18.

$$0.051/s \leq u \leq 11/s \quad (5.18)$$

By incorporating this adjustment, the MPC will ensure that it never outputs less than 0.05 l/s. Furthermore, a penalty has been introduced to the objective function of the MPC in case u is set to less than 0.15 l/s. This penalty aims to encourage the MPC to avoid setting values for u below 0.15 l/s unless absolutely required. To incorporate this penalty into the objective function, an additional term was introduced. The modified objective function is depicted in equations 5.19 and 5.20.

$$\min J = \sum_{k=1}^N e_k^T Q_k e_k + u_{k-1}^T P_{k-1} u_{k-1} + \Delta u_{k-1}^T R_{k-1} \Delta u_{k-1} + Z^T \lambda Z \quad (5.19)$$

$$Z = \max(0, 0.15 - u) \quad (5.20)$$

Since there are multiple variables constituting constraints for the model, λ is set to be a dynamic value with the function shown in equation 5.21 and values defined in equation 5.22.

$$\lambda = \max(\lambda_0 + K(k_1 DS_{in} + k_2 Q_{in} + k_3 f_T), 0) \quad (5.21)$$

$$\lambda_0 = 6247.25 \quad K = 6.71 \quad k_1 = -121.25 \quad k_2 = -87.54 \quad k_3 = -3.51 \quad (5.22)$$

These parameters were determined by testing the MPC under various model constraints. Subsequently, for the same constraints, a value for λ was identified. This process was repeated for all the different constraints. This resulted in three points, which were then interpolated to find the values for λ_0 , K , k_1 , k_2 , and k_3 .

In a simulated process where the dry solids content inflow to the drum screen falls below the range of the model's limitations, the MPC will calculate u as depicted in Figure 5.19.

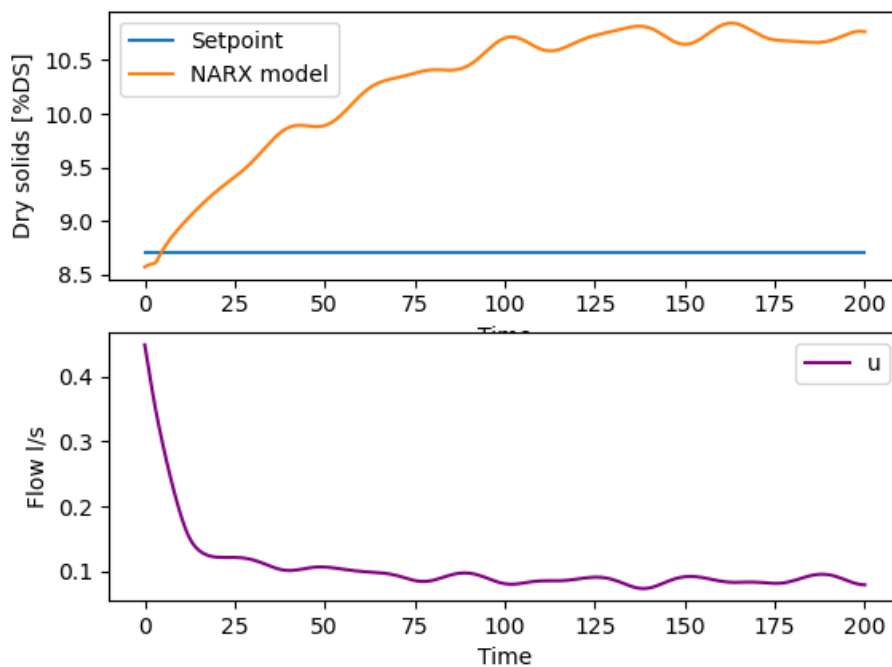


Figure 5.19: Simulated MPC with DS_{in} outside limitations

Based on the results shown in Figure 5.19, even when the flow rate is below the constraints of the model, the MPC calculates the polymer flow to approximately 0.1 l/s instead of setting it to its minimum limit of 0.05 l/s. Instead, it allows the model to estimate a higher dry solids value.

5.5.5 Closing the loop and let the MPC dosage polymer

After comparing the polymer dosages administered by operators with the MPC-calculated dosages, Veas decided to initiate a three-month testing period to assess the efficacy of polymer dosing using the model and MPC regulation. The test commenced on April 15th. It was determined that the MPC would operate only during daytime hours, as Veas is not staffed at night.

To assess the effectiveness of MPC over prolonged durations without the need for constant laboratory analysis, an examination of the difference between Q_{in} and Q_w can be insightful. Ideally, these two values should exhibit a minimal inequality, with Q_{in} typically slightly exceeding Q_w . Figure 5.20 depicts data from a day of MPC control operation, showing a negligible difference between Q_{in} and Q_w , suggesting favorable dry solids content. From 12:00 to 12:45, a hot water flush was conducted, resulting in inaccurate readings for the reject water due to additional water being introduced besides the sludge.

During the periods shown in Figure 5.20, 0.9% more polymer was dosed compared to what would have been done using the old calculation method outlined in Eq. 2.1. This is not ideal, as the objective of this project was to minimize polymer dosing. However, a 0.9% increase in polymer dosing may not be significant, and it appears that the MPC is providing better polymer quantities where needed and reducing polymer where it is not needed as much.

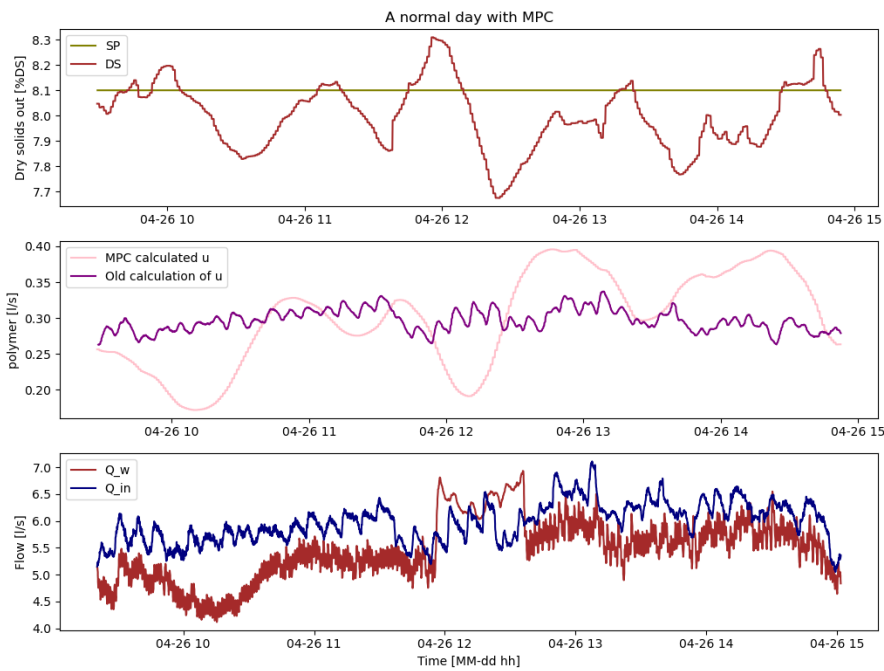


Figure 5.20: Closed loop run of polymer dosage by MPC in a 6 hour time span

Discovery of effective regulation during periods of low sludge settling

After closing the loop of the MPC controller for approximately two weeks, it was discovered that during periods of low sludge settling in the SED tanks, regulating polymer dosage correctly was generally challenging. However, with the use of the MPC controller, this dosing improved significantly. The results from such a period can be observed in Figure 5.21. After closing the loop of the MPC controller for approximately two weeks, it was discovered that during periods of low sludge settling in the SED tanks, regulating polymer dosage correctly was generally challenging. However, with the use of the MPC controller, this dosing improved significantly. The results from such a period can be observed in Figure 5.21.

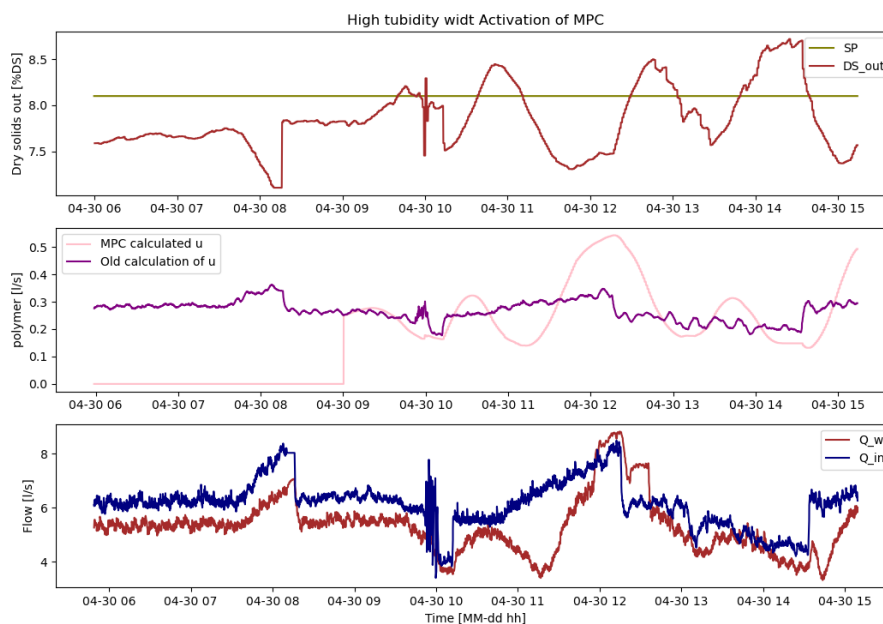


Figure 5.21: MPC initiates dosing at 09:00, replacing the previous dosing strategy in use

It is worth noting that there is still room for improvement with the MPC. As seen in Figure 5.21, the regulation becomes unstable when the MPC is active. This instability may indicate that the weighting of Q , P , and R is not yet optimal, and there are other tunings that should be made. However, overall, it appears that the dry solids content from the drum screen increased from a steady 7.5% to an oscillating value with an average of approximately 8%. Therefore, new weights were applied to P to make the MPC less unstable. The latest updated parameters for the MPC are shown in Equation 5.23.

$$Q = 33 \quad P = 9998 \quad R = 1 \quad (5.23)$$

In addition, Veas has a total of 5 drum screens. Not all of them are consistently functioning; their operational status is depending upon the amount of sludge, which can either cause them to stop or start. Therefore, if one of the drum screens is stopped or started, there will be a considerable change in the flow into the other drum screens. When the value of P is increased to a high level, as indicated in eq. 5.23, the system will slowly adapt in order to attain the desired setpoint. Thus, when the additional drum screen is stopped or started, the variable P will be altered to 556 for a period of 10 minutes. The method by which the MPC handles a stopped drum screen can be observed in figure 5.22, where a drum screen stops at 11:15.

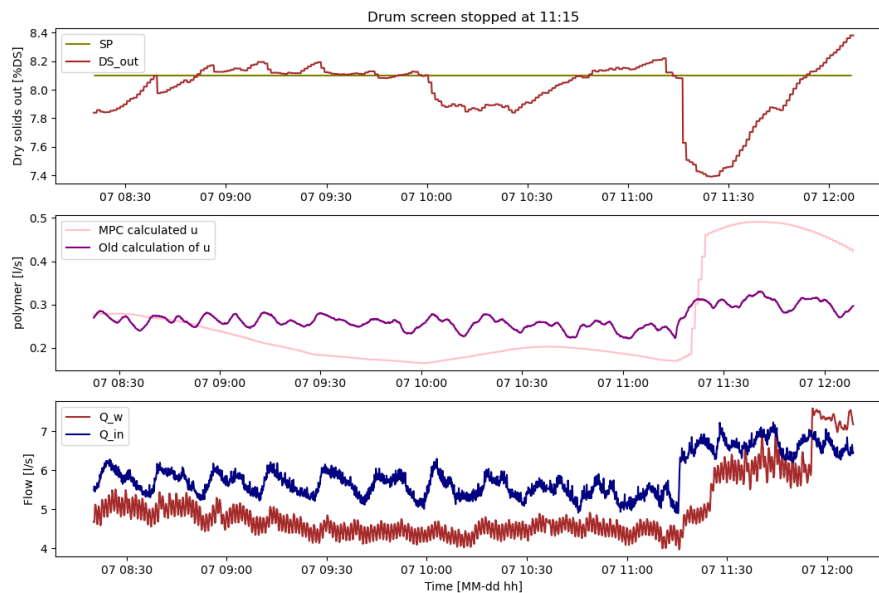


Figure 5.22: The MPC system manages the stoppage of a drum screen at roughly 11:15.

6.1 Interpolation

Upon reviewing the interpolation results depicted in Figure 5.4, it becomes evident that the timeline spanning an entire day is not accurate. This mismatch arises because the interpolation process involved collecting data from three distinct days (05.01.24, 08.01.24, and 19.01.24). The first day was normal, the second day was during a period of poor sludge settling in SED, and the last day involved a step response test. Consequently, this composite dataset does not accurately represent a typical day but rather a mix of information from these three separate extreme days. Presenting this dataset as representative of a normal day would be highly misleading.

While extreme situations may not accurately reflect normal operating conditions, they provide essential insights for the creation of models. These scenarios, generated by combining two highly contrasting days, offer a chance to thoroughly assess the behavior of all input variables under different settings. By including data from these extreme scenarios into the process of constructing the model, the resulting model effectively represents the dynamics of the system under a broad spectrum of conditions. This strategy improves the strength and thoroughness of the analysis, allowing all input variables to be properly evaluated and given suitable importance in the model.

6.2 Comparing results from dimensionality reduction using PCA and number of inputs used in NARX model

Once seven variables that accurately represent the amount of dry solids have been identified by PCA, it is important to compare this quantity with the maximum number of input variables that the SysIdentPy library can use to create a "good" NARX model. Prior to conducting this comparison, it is crucial to take into account that the quantity of variables used in constructing the NARX model is limited by the outcomes of dimensionality reduction by PCA. Consequently, the NARX model is limited to a maximum of seven inputs and is restricted to only these seven variables that have been found by PCA.

Given that x_1 represents DS_{in} , x_2 represents Q_{in} , x_3 represents the percentage of dry solids in the polymer, x_4 represents u , and x_5 represents f_T , a comparison reveals that the NARX model utilizes these 5 inputs. These inputs correspond to 4 out of the 7 variables identified through PCA. The inclusion of x_3 was done subsequently to better estimate the amount of polymer needed to achieve the desired dry solids content. Hence, this comparison offers reassurance regarding the efficacy of PCA in dimensionality reduction, as not all variables were used. If all variables were included, there might have been variables left out that should have been included.

Conversely, it's reassuring to observe that almost all variables used in the NARX model originate from the PCA analysis, ensuring that the model is estimated based on the correct parameters.

6.3 Model

The comparison of the model predictions and the measured dry solids output from the drum screen shows certain inconsistencies, as depicted in Figure 5.13. The model constantly exhibits a tendency to overestimate the dry solids content to varying degrees. This observation is consistent with similar patterns observed during the prolonged two-week testing period, as depicted in Figure 5.8. These findings indicate that although the model accurately represents the general behavior of the system, it faces difficulties in accurately estimating minor changes over time.

An effective method to enhance the accuracy of the model is by integrating a Kalman filter. By incorporating a Kalman filter into the modeling framework, it is feasible to improve the model's capacity to adjust to dynamic changes and reduce the influence of measurement noise or uncertainties in the system. The algorithm of the Kalman filter allows for real-time estimation of system states using noisy measurements, thus providing a way to enhance the accuracy of model calculations.

Based on these observations, future research will focus on exploring the integration of a Kalman filter into the modeling framework. This attempt will aim to refine the model's estimation capabilities, mainly in capturing minor variations in the dry solids output over time. By addressing these limitations, the model can serve as a more reliable tool for process control and optimization in practical applications.

6.3.1 Time delay

After assessing various modeling approaches, the NARX model with computed time delays was determined to be the most appropriate option for implementation, as shown in eq. 5.9. The decision was influenced by concerns about the accuracy and reliability of the data used for system identification. The time delays obtained from system identification were derived from an interpolated dataset, which may not consistently reflect the actual dynamics of the system. Due to the possibility of errors occurring during interpolation, the chosen method involved calculating the time delays rather than only relying on the identified values.

The objective was to improve the reliability of the model by reducing the impact of any errors in the interpolated dataset through the calculation of time delays. Despite the added complexity in determining the suitable time delays, this approach provided a higher level of certainty in the model's capacity to accurately represent the actual dynamics of the system. Furthermore, the integration of calculated time delays allowed for the flexibility to adjust the model parameters in order to more accurately match the real-world behavior of the process.

6.3.2 NARX Model Limitations

By comparing the constraints of the model with those of the actual drum screen operation, the limitations do not seem too severe anymore. Veas aims to pump out a dry solids content of 3.3%DS from all SEDs, so the model's constraint of 2.2%DS could be deemed acceptable. Additionally, the operating range for the drum screen is set with an inflow between 5 l/s and 7 l/s, so the model's limitations are not too far off either in this aspect.

The second limitation poses a more significant challenge as both the dry solids content and the flow rate are low. This occurs in proximity to the operational range of the system. The solution implemented was to create an alarm system to alert operators when the model exceeds its limitations and can no longer be relied upon.

6.3.3 Washing of the drum screen

As previously mentioned, the drum screens undergo periodic cleaning cycles, receiving a brief rinse with cold water every minute for approximately 10 seconds and a more thorough wash with hot water twice daily for 40 minutes each time. When developing the models for the drum screens, these cleaning cycles were not explicitly considered.

The oversight regarding these cleaning cycles may contribute to inconsistencies between model predictions and actual observations. During cleaning cycles, changes in flow dynamics and material composition passing through the screens may occur, which the models fail to account for. Consequently, deviation between model predictions and observed data may arise.

Although the absence of consideration for these cleaning cycles in the models is noteworthy, it does not undermine the overall validity of the models. Future iterations of the models should include adjustments to incorporate the effects of these cleaning cycles, thereby enhancing the accuracy of the predictions, particularly during periods of cleaning activity.

6.3.4 TAG names for models

Since three different models were developed and two of them were tested on the actual process, two TAG names were created instead of just one as mentioned in the methods. This is because both the NARX model and the mass balance model were to be displayed simultaneously. Therefore, the original name *FOR – TRS5 – QB01* was assigned to the mass balance model, and the NARX model was assigned the name *FOR – TRS5 – QB02*.

6.4 MPC

During the simulation and testing stages, the implementation of MPC for polymer dosing at the drum screen faced several challenges. While the simulation phase provided valuable insights into the behavior of the system, it did not uncover all potential issues before system testing. In perspective, it would have been beneficial to perform additional simulations, such as step-response tests, to identify and address any limitations or constraints prior to implementing the MPC in the operational environment.

Throughout the testing phase, a notable observation arose regarding the necessity of modifying the weight, P in the MPC controller to accommodate variations in the control signal (u). Originally set at 33 according to simulations, it became clear during testing on the real system that this value was not suitable to achieve optimal control performance. As a result, the value of P was raised to 9998. This significant modification highlights the complex nature of the system dynamics and highlights the need to carefully adjust control parameters to ensure efficient operation in real-world situations.

Moreover, the process of transitioning from MPC recommendations to direct control by the system, which is referred to as closing the loop, is an essential phase in implementing the MPC for practical use in real-world scenarios. Veas aims to optimize the dosing process and improve overall efficiency by allowing the system to automatically implement calculated dosage adjustments without the need for operator involvement. This transition represents a notable achievement in the implementation of complex control strategies at Veas and demonstrates a dedication to utilizing technology to enhance the efficiency of wastewater treatment operations.

To tackle the situation where the model operates beyond its limitations, especially during MPC control, a penalty term for dosages below 0.15 l/s was introduced as a solution. This adjustment was necessary to guarantee that the MPC does not generate control signals that breach the operational limitations of the system. The MPC has been driven to prevent dosages below the specified threshold by imposing penalties. This encourages the MPC to avoid extremely low values unless they are absolutely necessary, thus improving the stability and reliability of the control process. This approach aids in preserving the system within acceptable parameters, reducing the likelihood of unfavorable consequences and ensuring more seamless operation in general.

To sum up, the implementation of MPC for polymer dosing at the drum screen marks a noteworthy progress in improving process control and efficiency at Veas. Even if there have been obstacles and modifications needed along the road, this project represents a shift to more complex control methods in a setting where PID and feedforward regulation dominate. The implementation of MPC shows promise for simplifying wastewater treatment procedures and enhancing overall system performance at Veas, despite the ongoing refining process.

6.4.1 MPC during periods of poor sludge settling

After running MPC for a few weeks with the loop closed, it was observed that, on average, MPC tends to use slightly more polymer than what the operators currently dose. However, as shown in the results, MPC shines in dosing accurately during periods of poor sludge settling, with the dry solids content increasing by approximately 0.5%DS, as depicted in Figure 5.21. It has been proven that during those periods, it can be challenging to accurately provide the exact dosage. Therefore, perhaps the solution for this MPC implementation would be to rely on the dry solids estimation during regular operation but automatically activate the control to take over from the operators when poor sludge settling occurs in the SED tanks. The testing is currently ongoing during daytime, and the decision will not be made before August.

7.1 Kalman filter

As discussed earlier, incorporating a Kalman filter could prove to be a valuable method for updating and refining the model to better match real-world values. Given that the model is nonlinear in nature, an extended Kalman filter could be employed for this purpose. By updating the model according to the principles outlined in eq. 7.1 until 7.5, it becomes possible to adjust the model's parameters based on observed data, thereby improving its accuracy over time [38]:

If the model is represented by eq. 7.1 and 7.2 [38].

$$x_{k+1} = f(x_k) \quad (7.1)$$

$$y_k = g(x_k) \quad (7.2)$$

The measurement update can be calculated as follows [38]:

$$\hat{x}_{k,k} = \hat{x}_{k-1} + K_k(y_k - g(\hat{x}_{k-1})) \quad (7.3)$$

In addition to providing measurement updates, time updates must also be calculated. This can be computed as shown in equation 7.4. Here, F_k represents the Jacobian of $f(x_k)$ [38].

$$P_k = F_k \cdot P_{k-1} F_k^T + Q_k \quad (7.4)$$

The only remaining step is to update the Kalman gain. This can be done as represented in eq. 7.5 [38].

$$K_k = P_{k-1} G_k^T (G_k P_{k-1} G_k^T + R_k) \quad (7.5)$$

Where G_k represents the Jacobian of $g(x_k)$ [38].

The challenge in implementing a Kalman filter lies in determining the measurement update. Ideally, daily samples of dry solids output from the drum screens would be taken. However, Veas was reluctant to adopt this approach due to the additional workload it would entail compared to the perceived benefits for the model. As an alternative, measurements could be updated from a sensor located after the buffer tank, FOR2. However, this introduces a time delay of approximately 1.5 days, which varies depending on the amount of sludge received by Veas. During this delay, the sludge undergoes degradation of dry solids due to hydrolysis creating water molecules, resulting in inconsistencies between the measured values and the actual output from the drum screens. This presents a challenge that should be addressed in future research endeavors.

7.2 Model

Incorporating the cleaning process of the drum screen into the model would provide a more comprehensive representation of the system dynamics. The cleaning process, which involves periodic rinsing with cold and hot water, has a significant impact on the operation of the drum screen.

By including the cleaning process in the model, it becomes possible to simulate how these periodic rinsing cycles affect the performance of the drum screen and subsequently adjust the control strategies accordingly. This can lead to more accurate predictions and better control

of the thickening process, ultimately improving the overall efficiency and reliability of the wastewater treatment system at Veas.

7.3 MPC

During the ongoing three-month testing period, spanning from April to August, continued adjustments and developments in the MPC for polymer dosing will take place. These adaptations aim to optimize the controller's operation and ensure its effectiveness in regulating polymer dosing at the drum screen. It's crucial during this testing phase to closely monitor and address any occasions where the MPC delivers incorrect polymer quantities, allowing for prompt resolution and refinement of the control strategy. This iterative process will contribute to the continuous improvement of the MPC system.

CONCLUSION

The initiation of this master's thesis involved a preliminary period of two weeks, during which samples of dry solids were gathered and examined in the laboratory. After the initial phase, three different modeling approaches were studied to help estimate dry solids and implement MPC in polymer dosing at the drum screens. The approaches included a linear model incorporating time delay and time constant, a non-linear model obtained through system identification, and a mechanistic model rooted upon mass balance principles. The non-linear model (NARX) was chosen for dry solids estimation and MPC deployment after a thorough evaluation.

It is crucial to acknowledge that the selected model has specific constraints, which require careful consideration for both the implementation of MPC and the interpretation of the model. The model will undergo continuous testing and evaluation until August, with the goal of identifying and resolving any challenges and errors that arise during its operation. The outcomes acquired during this testing phase will have a crucial impact on Veas' determination regarding the implementation of MPC in its operational procedures.

In addition, potential future developments in the model could include integrating a Kalman filter between the model and sensors that measure the dry solids output from the FOR2 buffer tank. This enhancement is intended to enhance the accuracy and dependability of estimates for dry solids, thereby further improving the effectiveness of the MPC system.

To summarize, although MPC holds potential for enhancing polymer dosing at Veas' drum screens, its continuous operation may not be essential in specific circumstances, particularly considering its tendency to consume a higher average amount of polymer compared to the current practices of operators. Hence, employing a hybrid strategy that employs the MPC system exclusively when sludge settling is poor, resulting in a notable 0.5% increase in dry solids content, may prove to be a more efficient and successful solution. In summary, this study is a notable advancement in improving the effectiveness and dependability of the wastewater treatment process at Veas.

BIBLIOGRAPHY

- [1] Veas, "Prosesoversikt i png," Document ID: 23884-1, 2022, [Internal document].
- [2] n.d. (2024) Meet the revolution pi family. KUNBUS. [Online]. Available: <https://revolutionpi.com/en/revolution-pi-series?noredirect=en-US>
- [3] V. Watson, X. Lou, and Y. Gao, "[figure] a review of profibus protocol vulnerabilities - considerations for implementing authentication and authorization controls," in *International Conference on Security and Cryptography*, 2017. [Online]. Available: <https://api.semanticscholar.org/CorpusID:13167442>
- [4] T. Madsen and I. Skeie, "Assignment da3 multivariate analysis of turbidity at veas," 2023.
- [5] W. Rocha, "Build dynamic models with sysidentpy," 2022. [Online]. Available: <https://sysidentpy.org/>
- [6] Veas, "Standardpresentasjon powerpoint," Document ID: 23639-4, 2022, [Internal document].
- [7] Veas-employees, Veas meetings, 2022, training of the Veas process for employees.
- [8] Sudarno, *Nitrification in Fixed Bed Reactors Treating Saline Wastewater*. Karlsruhe: KIT Scientific Publishing, Jul 2011.
- [9] J. Kopp and N. Dichtl, "The influence of free water content on sewage sludge dewatering," in *Chemical Water and Wastewater Treatment VI: Proceedings of the 9th Gothenburg Symposium 2000, October 02-04, 2000, Istanbul, Turkey*. Springer, 2000, pp. 347–356.
- [10] M. R. Haugen, 2024, conversation with Senior Process Engineer Morten Rostad Haugen.
- [11] 2024, information retrieved from Control system (DKS) at Veas.
- [12] M. R. Haugen, "Veas høy viskositet, rapport - mulighetsstudie," Project ID: 10232228, 2022, [Internal document].
- [13] B. D. andd Wright E, *Practical SCADA for Industry*. Elsevier Science Technology., 2003.
- [14] S. Tjerner, M. Listaul, and T. M. Madsen, "Unmanned inspection of remote installations," 2021.
- [15] F. T. Mamo, A. Sikora, and C. Rathfelder, "Legacy to industry 4.0: A profibus sniffer," *Journal of Physics: Conference Series*, vol. 870, no. 1, p. 012002, jul 2017. [Online]. Available: <https://dx.doi.org/10.1088/1742-6596/870/1/012002>
- [16] Y. Veryha, "Going beyond performance limitations of opc da implementation," in *2005 IEEE Conference on Emerging Technologies and Factory Automation*, vol. 1, 2005, pp. 4 pp.–50.
- [17] Abb ability edgeinsight. ABB. [Online]. Available: https://library.e.abb.com/public/eb80b7efd91e477ab0064756834c7203/4JZZ000274_ABB%20Ability%20EdgeInsight%20Brochure.pdf
- [18] n.d. (2023) Tørrstoff. AquaOptic. [Online]. Available: <https://www.aquaoptic.no/t%C3%B8rrstoff-m%C3%A5ling-avl%C3%B8psrensprosess-landbaser-fiskeoppdrett>
- [19] —, "Sludge, treated biowaste, soil and waste - calculation of dry matter fraction after determination of dry residue or water content," p. 14, 2012, [EN 15934].
- [20] M. B. Holm, "Ss og gløderest i avløpsvann," Document ID: 22885-6, 2022, [Internal document].

- [21] M. J. J. Hoogsteen, E. J. B. E. A. Lantinga, J. C. J. Groot, and P. A. Tittnell, "Estimating soil organic carbon through loss on ignition: effects of ignition conditions and structural water loss," 2015. [Online]. Available: <https://bsssjournals-onlinelibrary-wiley-com.ezproxy1.usn.no/doi/10.1111/ejss.12224>
- [22] n.d. (2022) Turbiditet. PS Prosess-styring AS. [Online]. Available: <https://www.prosess-styring.no/turbiditet>
- [23] W. Sun and C. Zhang, "Analysis and forecasting of the carbon price using multi-resolution singular value decomposition and extreme learning machine optimized by adaptive whale optimization algorithm," *Applied Energy*, vol. 231, pp. 1354–1371, 2018. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0306261918314296>
- [24] D. D. Ruscio, *SUBSPACE SYSTEM IDENTIFICATION Theory and applications*, 2022.
- [25] H. Abdi and L. J. Williams, "Principal component analysis," *WIREs Computational Statistics*, vol. 2, no. 4, pp. 433–459, 2010. [Online]. Available: <https://wires-onlinelibrary-wiley-com.ezproxy1.usn.no/doi/abs/10.1002/wics.101>
- [26] R. P. Borase, D. K. Maghade, S. Y. Sondkar, and S. N. Pawar, "A review of pid control, tuning methods and applications," *International Journal of Dynamics and Control*, pp. 818–827, jul 2020. [Online]. Available: <https://link-springer-com.ezproxy2.usn.no/article/10.1007/s40435-020-00665-4>
- [27] R. Sharma, *Lecture notes for the course IIA 4117: Model Predictive Control*, 2011. [Online]. Available: https://web01.usn.no/~roshans/mpc/downloads/lecture_notes_MPC.pdf
- [28] W. R. Lacerda, L. P. C. da Andrade, S. C. P. Oliveira, and S. A. M. Martins, "Sysidentpy: A python package for system identification using narmax models," *Journal of Open Source Software*, vol. 5, no. 54, p. 2384, 2020. [Online]. Available: <https://doi.org/10.21105/joss.02384>
- [29] W. R. Lacerda, V. M. Almeida, and S. A. M. Martins, "Identifica ao de um motor/gerador cc por meio de modelos ~ polinomiais autorregressivos e redes neurais artificiais," *Journal of Open Source Software*, p. 152, 2017. [Online]. Available: https://www.sba.org.br/Proceedings/SBAI/SBAI2017/SBAI17/papers/paper_58.pdf
- [30] M. R. E. Symonds and A. Moussalli, "A brief guide to model selection, multimodel inference and model averaging in behavioural ecology using akaike's information criterion," p. 10, 2010. [Online]. Available: <https://www.proquest.com/scholarly-journals/brief-guide-model-selection-multimodel-inference/docview/847311904/se-2?accountid=43239>
- [31] W. R. L. Junior, S. A. M. Martins, and E. G. Nepomuceno, "Meta-model structure selection: Building polynomial narx model for regression and classification," 2021.
- [32] B. Lie, *Modeling of Dynamic Systems*, 2019.
- [33] M. Stamnes, "Tag system," Document ID: 16719-24, 2024, [Internal document].
- [34] "Process measurement control functions and instrumentation," ISO 3511-2:1984, 1984.
- [35] scikit-learn developers. (2024) Getting started. scikit-learn. [Online]. Available: https://scikit-learn.org/stable/getting_started.html
- [36] A. A. G. . P. J. . B. Sletbak, *Fluidmekanikk*. [Online]. Available: <https://www.uio.no/studier/emner/matnat/fys/nedlagte-emner/FYS1000/v11/Fluidmekanikk.pdf>
- [37] n.d. (2024) Quick start guide. KUNBUS. [Online]. Available: <https://revolutionpi.com/en/tutorials/quick-start-guide>
- [38] K. Rapp, "Nonlinear estimation and control in the iron ore pelletizing process: An application and analysis of the extended kalman filter," 2004.

APPENDIX

Table of appendices

| | |
|---|----|
| Appendix A: Symbol list and TAG names | 64 |
| Appendix B: Acronyms | 66 |
| Appendix C: GitHub Repository. | 68 |
| Appendix F: GSD file setup for Profibus | 70 |
| Appendix E: Lab Analysis Scheduled Plan. | 72 |
| Appendix F: Progress Plan | 74 |
| Appendix G: Task Description. | 76 |

Appendix A:

Symbol list and TAG names

Symbol list and TAG names

| Symbol | Description | TAG name |
|------------------------|---|--------------------------|
| SP | Setpoint for polymer dosage | POLF_SP09 |
| θ | Operator-set value, amount of polymer to be added per ton of dry solids [kg/tonn %DS] | POLF_TRS5_DOS_SP |
| DS_p | Dry solids in polymer [%DS] | POLF_TS_POLYMER |
| DS_{in} | Dry solids into drum screen [%DS] | ζ FOR_QT03 |
| DS_w | Dry solids in the reject water line [%DS] | - |
| DS_{out} | Dry solids out of drum screen calculated from mass balance [%DS] | FOR-TRS5-QB01 |
| u | Polymer dosage to drum screen [l/s] | ζ POLF_SP09 |
| f_T | The average turbidity in all SED basins [FTU] | ζ PHA18-SED18-QB01 |
| Q_{in} | Flow of sludge into drum screen [l/s] | ζ TRS5_FT01 |
| Q_w | Flow of water out of drum screen (Reject water) [l/s] | ζ TRS5_FT01 |
| $Q_{sum,s}$ | Sum of the sludge coming from the SED basin [l/s] | ζ SED_FTB_SLAM |
| $Q_{sum,d}$ | Sum of sludge going into all 5 drum screens [l/s] | ζ TRSx_FT01_SUM |
| ϕ_p | A motorized mixer that mixes polymer into the sludge for drum screen 5 [RPM] | ζ POLF_MIX05 |
| ϕ_f | Motorized stirring in the flocculation tank before the drum screen [RPM] | ζ TRS5_RX01 |
| ϕ_d | Motorized rotation of the drum screen [rpm] | ζ TRS5_Sl01 |
| β_r | The position of the control valve to control the flow into drum screen 5 [%] | ζ TRS5_RV01 |
| β_{cw} | Valve for cold water to rinse the drum screen [open/closed] | ζ DVN_PV80 |
| β_{ww} | Valve for warm water to rinse the drum screen [open/closed] | ζ DVN_PV81 |
| p | Pressure in the polymer string into drum screen [bar] | ζ POLF_PT09 |
| $\dot{m}_{in,tot}$ | Mass flow of mixture into the drum screen | - |
| $\dot{m}_{w,tot}$ | Mass flow of mixture in the reject Water line | - |
| $\dot{m}_{s,tot}$ | Mass flow of mixture in the concentrated sludge line | - |
| $\dot{m}_{in,s}$ | Mass flow of sludge into the drum screen | - |
| $\dot{m}_{w,s}$ | Mass flow of sludge in the reject Water line | - |
| $\dot{m}_{s,s}$ | Mass flow of sludge in the concentrated sludge line | - |
| SS | Suspended solids in reject water line | - |
| G | Los of Ignition | - |
| Δt_f | Time delay for u | - |
| Δt_p | Time delay for DS_{in} | - |
| Δt_F | Time delay for f_T | - |
| For NARX modell | | |
| x_1 | DS_{in} | |
| x_2 | Q_{in} | |
| x_3 | DS_p | |
| x_4 | u | |
| x_5 | f_T | |
| \hat{y} | Dry solids out of drum screen caclulated from NARX model [%DS] | FOR-TRS5-QB02 |

Appendix B:

Acronyms

Acronyms

| Acronyms | Meaning |
|-----------------|--|
| MPC | Model Predictive Control |
| NARX | Nonlinear AutoRegressive with eXogenous inputs |
| SED | Sedimentation |
| DS | Dry solids |
| FOR2 | Buffer tank after drum screens |
| PLC | Programmable Logic Controller |
| RevPi | Revolution Pi |
| SCADA | Supervisory Control And Data Acquisition |
| GSD | General Station Description |
| LoI | Loss on Ignition |
| SVD | Singular Value Decomposition |
| PCA | Principal Component Analysis |
| AIC | Akaike's Information Criterion |
| FROLS | Forward Regression Orthogonal Least Squares |

Appendix C:

GitHub Repository

GitHub Repository

All code used in this master's thesis is available on GitHub and can be accessed by clicking [here](#), or by using the following link: <https://github.com/DISToby99/MasterThesis>

The structure of the repository includes various folders. One folder contains all the code that runs continuously on the RevPi.

Another folder is dedicated to System Identification, and a third folder is for PCA.

It is advisable to read the README file before using the programs.

Appendix D:

GSD file setup for Profibus

| I/O | Port | Numbers of bytes | Byte size | Variable type | Connected to | Comments |
|--------|------|------------------|-----------|---------------|------------------|--|
| Empty | 1 | 0 | 0 | - | | |
| Out | 2 | 1 | 1 | BOOL | FOR-TRS5-YS04 | Activating Estimation Enabling MPC |
| | | | | BOOL | FOR-TRS5-YS02 | |
| | | | | BOOL | | |
| | | | | BOOL | | |
| | | | | BOOL | | |
| Out | 3 | 2 | 1 1 | Dint Dint | | |
| Out | 4 | 4 | 4 | Real | PHA18-SED18-QB01 | Mean turbidity |
| Out | 5 | 8 | 4 | Real | FOR-TRS5-SP01 | Setpoint for %DS |
| | | | 4 | Real | FOR_QT01 | %DS out of FOR2 |
| Out | 6 | 16 | 4 | Real | FOR_QT03 | %DS innto TRS |
| | | | 4 | Real | TRS5_FT01 | Flow to TRS5 |
| | | | 4 | Real | POLF_TS_POLYMER | %DS in polymer |
| | | | 4 | Real | POLF_FT09 | Flow Polymer to TRS5 |
| In | 7 | 1 | 1 | BOOL | FOR-TRS-XS01 | WatchDog for program The MPC failed |
| | | | | BOOL | FOR-TRS5-XA01 | |
| | | | | BOOL | | |
| | | | | BOOL | | |
| | | | | BOOL | | |
| | | | | BOOL | | |
| In | 8 | 2 | 1 | Dint | | |
| | | | 1 | Dint | | |
| In | 9 | 4 | 4 | Real | | |
| In | 10 | 8 | 4 | Real | FOR-TRS5-QB01 | Estimation %TS |
| | | | 4 | Real | FOR-POLF-SP09 | Setpoint for polymer |
| In | 11 | 16 | 4 | Real | | |
| | | | 4 | Real | | |
| | | | 4 | Real | | |
| | | | 4 | Real | | |
| In/Out | 12 | 1 | 1 | BOOL | | |
| | | | | BOOL | | |
| | | | | BOOL | | |
| | | | | BOOL | | |
| | | | | BOOL | | |
| | | | | BOOL | | |
| In/Out | 13 | 2 | 1 | Dint | | |
| | | | 1 | Dint | | |
| In/Out | 14 | 4 | 4 | Real | | |
| In/Out | 15 | 8 | 4 | Real | | |
| | | | 4 | Real | | |
| In/Out | 16 | 16 | 4 | Real | | |
| | | | 4 | Real | | |
| | | | 4 | Real | | |
| | | | 4 | Real | | |

Appendix E:

Lab Analysis Scheduled Plan

Lab Analysis Scheduled Plan

| Day | Date | 1 st sample | 2 nd sample |
|-----------|------------|------------------------|------------------------|
| Tuesday | 02.01.2024 | - | 13:00 |
| Wednesday | 03.01.2024 | 08:00 | 14:00 |
| Thursday | 04.01.2024 | 09:00 | 12:00 |
| Friday | 05.01.2024 | 10:00 | 13:30 |
| - | | | |
| - | | | |
| Monday | 08.01.2024 | 09:30 | 14:30 |
| Tuesday | 09.01.2024 | 11:00 | 16:00 |
| Wednesday | 10.01.2024 | 07:00 | 18:00 |
| Thursday | 11.01.2024 | 12:00 | 12:30 |
| Friday | 12.01.2024 | 10:30 | 14:00 |
| - | | | |
| - | | | |
| Monday | 15.01.2024 | 10:30 | 12:30 |
| Tuesday | 16.01.2024 | 10:30 | - |

Appendix F:

Progress Plan

Progress Plan

| | 01.jan | 08.jan | 15.jan | 22.jan | 29.jan | 05.feb | 12.feb | 19.feb | 26.feb | 04.mar |
|---------------------------------------|----------|----------|----------|----------|----------|----------|----------|----------|----------|-----------|
| Project/Week | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Sampling for analysis | █ | | | | | | | | | █ |
| PCA analysis | | █ | | | | | | | | |
| Creating a Modell | | | █ | | | | | █ | | |
| Testing Modell | | | | █ | | █ | | | | |
| Creting MPC | | █ | | | | | | | | |
| Find a system that connects 800xA | █ | | | | | | | | | |
| Create a System that talks with 800xA | | | | █ | | | | | █ | |
| TestingConnection to 800xA | | | | | | █ | | | | |
| Generate Profibus and MPC app | | | | | | █ | | | | |
| Testing | | | | | | █ | | | | |
| Writing Tehisis | █ | | | | | | | | | |

| | 11.mar | 18.mar | 25.mar | 01.apr | 08.apr | 15.apr | 22.apr | 29.apr | 06.mai | 13.mai |
|---------------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| Project/Week | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| Sampling for analysis | | | | | | █ | | | | |
| PCA analysis | | | | | | | | | | |
| Creating a Modell | | | | | | | | | | |
| Testing Modell | █ | | | | | | | | | |
| Creting MPC | █ | | | | | | | | | |
| Find a system that connects 800xA | | | | | | | | | | |
| Create a System that talks with 800xA | █ | | | | | | | | | |
| TestingConnection to 800xA | | | | | | | | | | |
| Generate Profibus and MPC app | | | █ | | | █ | | | | |
| Testing | | | | | | █ | | | | |
| Writing Tehisis | █ | | | | | | | | | |

Appendix G:

Task Description

FMH606 Master's Thesis

Title: Model-based control for optimal polymer addition to the drum screens in a wastewater treatment plant.

USN supervisor: Finn Aakre Haugen

External partner: Veas (Morten Rostad Haugen and Jørn Tore Nummestad)

Task background:

Veas is Norway's largest wastewater treatment plant. The Veas process can be described by dividing it into two main process lines:

- Water Treatment: Mechanical cleaning, chemical cleaning, and biological cleaning.
- Sludge Treatment: Thickening, anaerobic digestion, and dewatering.

One major challenge at Veas is related to the production of biogas in the digesters. To maintain a stable process and ensure an adequate amount of dry matter that goes into the anaerobic digesters, polymer is added to the wastewater before it passes through drum screens. The addition of polymer helps coagulate the particles in the sludge, allowing water to separate more efficiently. However, there are challenges associated with this process:

- Manual adjustment of polymer addition: The adjustment of polymer addition is done manually, which introduces a degree of uncertainty. This can lead to variations in the dry matter content going into the digesters, which can range from 4% to 8% DM. This lack of consistency can negatively affect biogas production.
- Clogging of drum screens: The drum screens, which are motorized cylindrical sieves, tend to become clogged over time. This requires regular flushing to maintain separation efficiency.

Task description:

1. Brief survey of methods for using polymer to coagulate the particles in the sludge.
2. Performing PCA analysis on all variables related to the drum screen to gain a deeper understanding of the parameters influencing the modelling of dry matter. PCA = Principal Component Analysis.
3. Design a control system for polymer addition to drum screen in Veas water resource recovery facility.

Student category: Reserved for IIA industry master student Torbjørn Mørk Madsen

Is the task suitable for online students (not present at the campus)? Yes

Practical arrangements:

Students will have access to all the data that Veas possesses.


Supervision:

As a general rule, the student is entitled to 15-20 hours of supervision. This includes necessary time for the supervisor to prepare for supervision meetings (reading material to be discussed, etc).

Signatures:

Supervisor (date and signature):

08 Feb 2024



Student (write clearly in all capitalized letters):

TORBJØRN MØRK MADSEN

Student (date and signature):

37.07.24

