

Summary of Master Thesis

This thesis investigates how intelligent control strategies combining Model Predictive Control (MPC) and Reinforcement Learning (RL) can be used to optimize the operation of Wastewater Treatment Plants (WWTPs). The motivation stems from the high energy consumption associated with these facilities, especially during biological treatment stages that require intensive aeration. Traditional control mechanisms, such as fixed-setpoint PID-controllers or manual tuning by operators, are often suboptimal in such dynamic and complex environments. With growing emphasis on sustainability and operational efficiency, particularly under the framework of the United Nations Sustainable Development Goals, the need for more adaptive and intelligent control solutions has become increasingly important.

The project builds upon a previous study focused on the Modified Ludzack-Ettinger (MLE) process but transitions to the Benchmark Simulation Model no. 1 (BSM1), which offers a standardized framework for assessing wastewater treatment control strategies. BSM1 incorporates a five-tank activated sludge reactor followed by a settling tank and simulates the biological, chemical, and physical processes central to modern WWTPs. It is designed to support the evaluation of new control techniques under various weather and inflow conditions.

The proposed intelligent control framework integrates several components. A high-fidelity digital twin of the BSM1 plant is implemented in the WEST simulation environment developed by DHI. This model captures the detailed internal dynamics of the plant. A simplified version of the plant is modeled in UPPAAL, which is used together with UPPAAL Stratego to apply reinforcement learning methods for synthesizing near-optimal control strategies. These strategies are coordinated through a tool called STOMPC, which implements the MPC logic, bridging the digital twin in WEST and the optimizer in UPPAAL Stratego. While the design also envisions an inflow prediction model and dynamic parameter estimation to better emulate real-world variability, these parts were not implemented due to the scope of the project.

Simulations were performed using realistic influent data under three scenarios: dry weather, rainy periods, and storm conditions. The MPC-RL pipeline demonstrated potential in reducing energy usage from aeration, while maintaining or improving effluent quality. This was achieved by dynamically adjusting key operational setpoints of the PI-controllers for the internal recirculation rate and oxygen transfer coefficients in the aerobic tanks. The evaluation of control strategies was based on effluent quality (EQ), a metric aggregating the weighted concentration

of pollutants in the outflow, as defined in the BSM1 framework and the energy consumption of the system.

Despite promising results, several challenges are also highlighted that must be overcome for real-world implementation. These include the need for fast simulation environments to support reinforcement learning, reliable state estimation in the presence of partial observability and noisy measurements, and predictive models for influent characteristics. Accurate, continuously updated parameter estimation is also crucial for maintaining the fidelity of the digital twin over time.

In conclusion, the thesis provides a proof-of-concept for how MPC combined with reinforcement learning can enable more intelligent and adaptive control of WWTPs. By replacing static control strategies with learned, dynamic strategies, it is possible to improve both energy efficiency and environmental outcomes. While the results are constrained by simplified assumptions and an offline simulation environment, the framework developed lays the groundwork for future extensions and real-world testing in collaboration with industrial partners like DHI.

Intelligent Control of Wastewater Treatment using Model Predictive Control & Reinforcement Learning

- A Master Thesis Project -

Project Report
DAT10

Aalborg University
Computer Science



Computer Science
Aalborg University
<http://www.aau.dk>

AALBORG UNIVERSITY

STUDENT REPORT

Title:

Intelligent Control of Wastewater Treatment using Model Predictive Control & Reinforcement Learning

Theme:

Computer Science, Experiments, Wastewater Treatment

Project Period:

Spring Semester 2025

Project Group:

cs-25-sv-10-04

Participant(s):

Christoffer Brejnholm Koch
Lise Bech Gehlert
Malthe Peter Højen Jørgensen

Supervisor(s):

Kim Guldstrand Larsen

Page Numbers: 92**Date of Completion:**

June 13, 2025

Abstract:

This thesis explores the use of Model Predictive Control (MPC) combined with Reinforcement Learning (RL) to optimize wastewater treatment plant (WWTP) operations, aiming to reduce energy consumption/cost and improve effluent quality. The report proposes a realistic implementation pipeline using a high-fidelity digital twin in WEST (by DHI) and a low-fidelity optimizer in UPPAAL Stratego, connected via the STOMPC MPC interface. Experiments under dry, rain, and storm weather scenarios show that MPC can improve both energy usage and effluent quality simultaneously. For dry weather conditions, MPC reduced energy consumption by up to 24% depending on the scenario.

The content of this report is freely available, but publication (with reference) may only be pursued due to agreement with the author.

Contents

| | |
|--|-----------|
| Preface | vi |
| 1 Introduction | 1 |
| 2 Case Study | 4 |
| 3 Theoretical Background | 6 |
| 3.1 WWTP overview | 6 |
| 3.2 Benchmark Simulation Model no. 1 | 9 |
| 3.2.1 Motivation | 9 |
| 3.2.2 Plant Layout | 10 |
| 3.2.3 BSM1 Dynamics | 12 |
| 3.2.4 Plant Controllers | 18 |
| 3.2.5 Base control | 19 |
| 3.2.6 Control evaluation | 20 |
| 3.2.7 Weather scenarios | 22 |

| | |
|--|-----------|
| Contents | iii |
| 3.3 BSM1 & MLE differences | 25 |
| 3.4 Model Predictive Control | 27 |
| 4 Related Work | 30 |
| 4.1 BSM1 and MPC | 30 |
| 4.2 Combining MPC & Reinforcement Learning | 31 |
| 5 Problem Statement | 32 |
| 6 Tools | 34 |
| 6.1 WEST & Tornado API | 34 |
| 6.1.1 BSM1 modeled in WEST | 34 |
| 6.1.2 Simulation in WEST | 36 |
| 6.1.3 Results in WEST | 36 |
| 6.1.4 Tornado API | 37 |
| 6.2 UPPAAL | 38 |
| 6.2.1 UPPAAL Core | 38 |
| 6.2.2 UPPAAL SMC | 39 |
| 6.2.3 UPPAAL Stratego | 44 |
| 6.3 STOMPC | 47 |
| 7 Implementation | 49 |
| 7.1 Pipeline | 49 |

| | | |
|----------|---------------------------------------|-----------|
| 7.2 | Modeling | 51 |
| 7.2.1 | Update_Discrete | 52 |
| 7.2.2 | Activated Sludge Tanks | 54 |
| 7.2.3 | Municipality Model | 56 |
| 7.2.4 | PI-Controller | 59 |
| 7.2.5 | Controller | 60 |
| 7.2.6 | Cost function | 61 |
| 7.2.7 | Model Correctness and Speed | 62 |
| 7.3 | MPC Implementation | 64 |
| 7.3.1 | TornadoExperiment class | 64 |
| 7.3.2 | BSM1_MPCsetup Class | 65 |
| 7.3.3 | MPC_Experiment Class | 66 |
| 8 | Experiments | 69 |
| 8.1 | Setup | 69 |
| 8.2 | Results | 72 |
| 8.2.1 | Dry Scenario | 72 |
| 8.2.2 | Rain Scenario | 75 |
| 8.2.3 | Storm Scenario | 77 |
| 8.2.4 | Optimizing energy cost | 79 |
| 9 | Discussion | 81 |

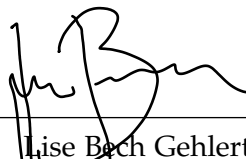
| | |
|--------------------------|-----------|
| Contents | v |
| 10 Future Work | 84 |
| 11 Conclusion | 86 |
| Bibliography | 88 |
| A Control Handels | 91 |

Preface

This report is written by the computer science group “cs-25-sv-10-04”. It is a master thesis project conducted from the 1st of February 2025 to the 13th of June 2025 at Aalborg University. The project is a collaboration with Dansk Hydraulisk Institut (DHI), focusing on optimizing wastewater treatment plants with regard to energy consumption/cost and effluent quality using MPC.

The group would like to thank the project supervisor, Kim G. Larsen, as well as Christopher Eugen Gaszynski, Enrico Ulisse Remigi and Trine Dalkvist from DHI for their continued cooperation and assistance throughout the project.

Aalborg University, June 13, 2025



Lise Bech Gehlert
<lgehle20@student.aau.dk>



Christoffer Brejnholm Koch
<ckoch20@student.aau.dk>



Malthe Peter Højen Jørgensen
<mphj20@student.aau.dk>

Chapter 1

Introduction

Wastewater treatment plants (WWTPs) implement physical, biological and chemical processes in order to remove pollutants from wastewater. They are crucial in preserving aquatic environments, preventing waterborne diseases, and indirectly preserving the quality of drinking water. While WWTPs are a necessity, some of these processes are very energy intensive. In fact, it is estimated that around 4% of the world's total energy consumption lies in the water sector, where a quarter of that energy consumption is from WWTPs [1].

The importance of WWTPs in environmental sustainability and the huge energy use makes WWTPs an enticing focus for optimizations. In fact, this is also a part of the United Nations Sustainable Development Goals. Specifically, this regards the sixth goal called "Clean Water and Sanitation", which aims to improve water quality and optimize use of water resources [2].

In the WWTP industry, these complex processes are often controlled by humans or PID controllers with fixed setpoints. As WWTPs are very dynamic and complex environments, the optimal control strategy of a plant is continuously changing, putting into question the effectiveness of manual human or fixed control.

It is important to note that this report is a continuation of the work done in a previous report [3]. As such, it is disclosed in each section, if the section has been fully or partially been carried over from the previous report, as much of the theoretical material has not changed. The previous report was also a collaboration with DHI, considering the exact same case study as the one in this report. In the previous report, a WWTP sub-plant, called the Modified Ludzack-Ettinger (MLE)

process, was modeled in the modeling and analysis tool UPPAAL. Hereafter, a variety of fixed setpoints were tested to explore the effect of changing setpoints on the outgoing water quality and energy consumption. This essentially laid the foundation for exploring the effect of implementing a model predictive control (MPC) setup for intelligent control of WWTPs, which is the goal of this report.

A major change in the approach from the last report to this report is the replacement of the MLE plant with the Benchmark Simulation Model no. 1 (BSM1) plant. BSM1 was chosen because it is a standardized model, designed specifically for testing various control strategies. BSM1 provides standardized influent data and established methods for evaluating control strategies [4]. Further justification for selecting the BSM1 plant can be found in section 3.2.1 and a description of the differences between the two plants can be found in 3.3. In this report, we propose a pipeline for intelligent control of the BSM1 plant. It is an MPC and digital twin setup, with several components:

- *Digital twin (WEST)* - This is the high fidelity virtual version of the actual plant, designed and modeled in WEST (tools provided by DHI, modeled by group). The state of the digital twin is updated with very low frequency to reflect the real plant.
- *Optimizer (UPPAAL)* - This consists of a low-fidelity model of the plant in UPPAAL. UPPAAL also provides built-in reinforcement learning techniques using UPPAAL Stratego. As such, the optimizer is used to compute near-optimal control strategies using reinforcement learning.
- *MPC (STOMPC)* - The STOMPC tool is used to implement the MPC scheme. It acts as an interface between the digital twin in WEST and the UPPAAL optimizer. It relays the plant state from the digital twin to the optimizer, and relays back the computed intelligent control strategy to be used.
- *Inflow prediction model* - This is a model that provides the optimizer with predictions on the characteristics of the incoming wastewater. This enables the optimizer to compute control strategies that not only are immediately effective, but also effective regarding the future state of the system.
- *Hourly electricity prices* - The optimizer is fed hourly energy prices to enable optimization of energy consumption timing.
- *Periodic parameter estimation* - The accurate simulation of a real plant largely relies on the accuracy of the model parameters. These model parameters can vary over time, depending on the weather, season and many other factors. As

such, periodic parameter estimation is needed to ensure accurate simulation, in turn increasing the likelihood of computing effective control strategies.

All the listed components are essential for intelligent control of a WWTP using MPC. However, due to the project’s scope, only a subset was implemented.

First, we modeled the BSM1 plant as a high-fidelity digital twin in WEST. Additionally, we created a low-fidelity BSM1 model in UPPAAL to leverage UPPAAL Stratego and reinforcement learning within the MPC setup. The MPC scheme was implemented using STOMPC, integrating UPPAAL Stratego with WEST.

We did not build an inflow prediction model but assumed its existence in our setup. Similarly, we did not develop a parameter estimation model and instead treated ASM1 model parameters as constant and accurate. Consequently, the real plant was excluded, as we assumed the digital twin to be fully precise at all times.

Using our BSM1 plant model and MPC implementations, we demonstrate potential reductions in energy consumption and enhancements in effluent quality under our assumptions.

Chapter 2

Case Study

This section has been carried over from the previous report [3], as this report is a continuation of the work done in the previous report. Thus, the case study remains the same.

This project is carried out in collaboration with the company Dansk Hydraulisk Institut, also known as DHI¹. DHI is an international consulting and research organization that specializes in the aquatic environment. DHI aims to develop and construct models for a sustainable future for water treatment in various aspects. Their main focus in research is how water can be managed and shared, adapted to climate changes, how solutions can be implemented, and how water quality can be improved. From the research, DHI tries to create new tools and techniques used to preserve, handle, and protect water-related ecosystems. Their main technology is their MIKE software portfolio [5], which focuses on water modeling software that delivers exceptional precision in simulations.

A particular water-related ecosystem is that of wastewater treatment and wastewater treatment plants (WWTPs). As a part of MIKE, DHI offers wastewater treatment process modeling, through their software tool called WEST. With this tool, one can model and simulate various WWTP processes through mathematical models, experimenting with different setups to improve effluent quality, energy consumption, and cost efficiency. Section 3 will elaborate on the purpose and functionality of WWTPs.

WEST is a mechanistic/deterministic modeling tool, and it is very efficient in

¹<https://www.dhigroup.com>

designing and simulating WWTPs and associated processes. WWTPs contain a variety of sensors, blowers, valves, gates, and other mechanical equipment, that can be controlled to optimize the water treatment process. A common control strategy is to use PID-controllers with specified target values or setpoints. These setpoints can then be changed manually by a WWTP operator. However, WWTPs are operating in extremely dynamic environments, which means that the optimal control strategy for the plant is continuously changing.

To this end, DHI proposes a case study, for the application of Model Predictive Control (MPC) to automatically find optimal control strategies that optimize WWTP energy use, effluent quality and other key points of interest. These control strategies can be tested on the mechanistic model (WEST), to see how a plant responds to the changes. The exact phrasing of their proposed case study is:

Mechanistic/Deterministic models, such as WEST, are strong tools for simulating wastewater treatment plants. This includes the biological, chemical, and physical processes throughout the plant. Wastewater treatment plants are extremely dynamic environments, and as such, the optimal control strategy for the plant is continuously changing. The plants contain a variety of sensors, blowers, valves, gates, and other mechanical equipment that is used to optimise the treatment process. A common strategy is to use controller-based logic throughout the plant, such as PIDs, set with target values or setpoints to control and optimise the processes.

This topic will investigate if data-driven models can be used to perform model predictive control (MPC) using the mechanistic model as the virtual treatment plant, testing how the plant will respond to controller settings changes, which can then be relayed to the physical treatment plant for optimal control.

Chapter 3

Theoretical Background

The following chapter lays the theoretical foundation needed to understand the different components of this report and the work done in the report.

3.1 WWTP overview

The following section 3.1 is from the previous report [3], with a few modifications to better explain and illustrate the general concept and purpose of Wastewater Treatment Plants.

Wastewater Treatment Plants (WWTPs) serve the purpose of cleaning polluted water to a state where it is less harmful to the environment when it is reintroduced into the natural water cycle of the surrounding area. The extent to which water is polluted depends on the concentration of different pollutants found in the water, as well as the intended use of the water. That is, water is polluted if the concentration of pollutants makes the water unfit for specific use cases, such as drinking, swimming, or fishing. There exist different forms of pollutants in wastewater.

One such pollutant is organic material, such as, protein, fat, carbohydrates and other organic chemicals. In wastewater, organic material is measured by biochemical oxygen demand (BOD). BOD is the amount of oxygen required by microorganism to decompose the organic material. The higher the BOD, the more organic material exist in the wastewater. Organic material/BOD is one of the most important pollutants to remove during the wastewater treatment processes, as it can

deplete the amount of dissolved oxygen in water, harming aquatic ecosystems.

Another important pollutant are suspended solids. This refers to small particles that do not dissolve in water, typically decayed plant material, algae, and other particulates. In relation to wastewater treatment, these are referred to as Total Suspended Solids (TSS). It is important to minimize the amount of TSS in the WWTP effluent, as it can cause water to look visibly impure and can cause taste and odor problems [6].

Lastly, plant nutrients are also considered pollutant material. Particularly, domestic wastewater contains compounds of nitrogen and phosphorus, which are basic compounds for plant growth. Here, the problem arises as excessive levels of nitrogen and phosphorus can cause rapid algae growth in rivers and lakes, causing damage to aquatic life and accelerating the natural aging of lakes [7].

An overview of the different pollutants can be seen in Figure 3.1 below.

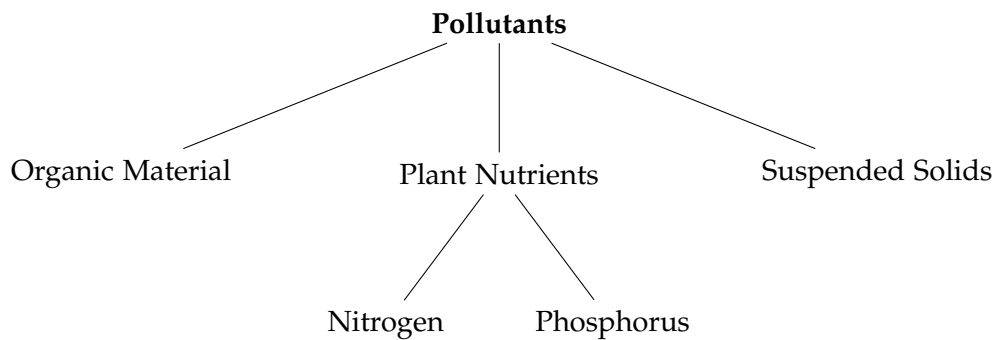


Figure 3.1: Overview of wastewater pollutants categorized into organic material, plant nutrients, and suspended solids [3].

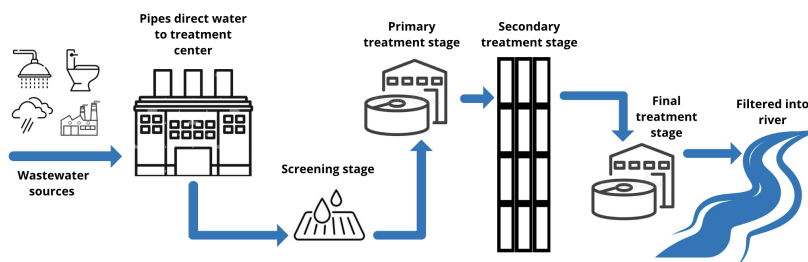


Figure 3.2: A depiction of a general WWTP setup. The icons on the left are examples of different influent sources. Figure inspired from [8].

Wastewater

In Figure 3.2, the water entering the “Treatment center” is known as wastewater, or influent. This type of water can originate from various sources, including domestic use in households and public facilities, industrial processes, and from stormwater or melted snow that enters the system through infiltration. All of these sources contribute to the wastewater that is carried through sewer and runoff collection networks before arriving at the WWTP. For domestic wastewater specifically, it typically comes from activities as taking a shower, washing hands or dishes in the sink, flushing toilets, and any similar water usage that leads to the sewer system. Wastewater is generally described in terms of three main characteristics: physical, biological, and chemical [9], each of which is addressed in a specific stage of the WWTP process.

Biochemical Terms

Organic material: Protein, fat, carbohydrates and other organic chemicals

Suspended solids: Decayed plant material, algae and other particulates

BOD: Biochemical oxygen demand

TSS: Total Suspended Solid

Plant nutrients: Nitrogen and phosphorus

Pretreatment

This stage is also called the screening stage. It is a mechanical removal process, which larger objects are removed from the wastewater such as excrement, sticks, grease, sand, and wipes. Anything large enough that it could damage pipes/pumps and other mechanical equipment.

Primary Treatment

The primary treatment focuses on the physical constituents and uses physical/mechanical processes to remove lingering solids too small for the pretreatment to catch. This can be done through screening, sedimentation, filtration and more.

Secondary Treatment

Secondary treatment focuses on the biological constituents and uses bacteria to change colloidal or dissolved biodegradable organic substances to gasses that can escape out through the air, or to biological cell tissues that will settle to the bottom or can be removed using different methods.

Tertiary treatment

This stage also known as the final stage. The Tertiary treatment is a more rigorous last step some WWTPs take to remove harmful microbiological bacteria masses still present in the wastewater. This could for example be exposing the wastewater to UV-light (the sun) to kill the bacteria.

The focus of this report is on the secondary treatment stage, where the water goes through a biological pollutant removal process. The specific setup is motivated and described in detail in the following section.

3.2 Benchmark Simulation Model no. 1

This section will first motivate the decision to adopt the BSM1 framework. Next, the layout of the BSM1 plant is explained in detail, followed by a description of the biological dynamics of the plant. Afterwards, the mathematical description of a PI-controller is presented, alongside the base control strategy for the BSM1 plant. Lastly, the section will look at how one evaluates the performance of a control strategy in BSM1 and then the different influent scenarios are presented.

3.2.1 Motivation

There are many factors contributing to the complexity of modeling and optimizing wastewater treatment plants, including the variability of the influent, the complexity of biological processes, the large number of variables involved, and the evaluation criteria, which are influenced by regulatory requirements and restrictions. Benchmark Simulation Model no. 1 (BSM1) standardizes the plant layout, simulation model, influent data, different weather scenarios, test procedures, and evaluation criteria. BSM1 provides standardized methods for assessing control strategies, ensuring consistency in evaluation [4].

The activated sludge reactor featured in BSM1 is among the most common approaches for achieving biological nitrogen removal in contemporary wastewater treatment plants (WWTPs). By introducing oxygen into the system, the reactor not only facilitates the reduction of nitrogen in the treated water but also significantly lowers the concentration of organic matter, measured as biochemical oxygen demand (BOD). These combined effects are critical for minimizing the environmental footprint of effluent discharge.

Despite its effectiveness, the activated sludge process comes with a major limitation: high energy demands. Maintaining the required levels of dissolved oxygen during the aerobic treatment phase relies heavily on aeration—a process that is notably energy intensive. In fact, aeration alone can account for over 50% of a WWTP's total energy usage [10]. This makes the process a key focus for energy

optimization efforts, prompting ongoing exploration into how aeration can be minimized without compromising the removal efficiency of nitrogen and organic pollutants.

In conclusion, BSM1 provides a mutual framework for comparing the performance of different control strategies. This paper will thus adopt the framework in order to more appropriately assess the performance of an MPC setup, as proposed in the case study.

3.2.2 Plant Layout

As seen in Figure 3.3, the BSM1 plant consists of a five-tank activated sludge reactor, followed by the settling tank. The reactor is designed to facilitate nitrogen removal through a combination of nitrification and pre-denitrification processes. The first two tanks operate under anoxic (oxygen free) conditions, while the remaining three tanks are aerobic (oxygen rich). The settling tank ensures the separation of biomass from the treated effluent. Lastly there is an internal recycler, transferring nitrate from the last aerobic tank to the first anoxic tank. Controlling parts of the plant are two PI-controllers, one changing the amount of flow to recycle back from tank 5 into tank 1, based on the amount of nitrate in tank 2. The other PI-controller reads the amount of dissolved oxygen in tank 5 and uses the reading to adjust how much oxygen the aerator needs to pump into tank 5, depending on some predefined oxygen concentration setpoint.

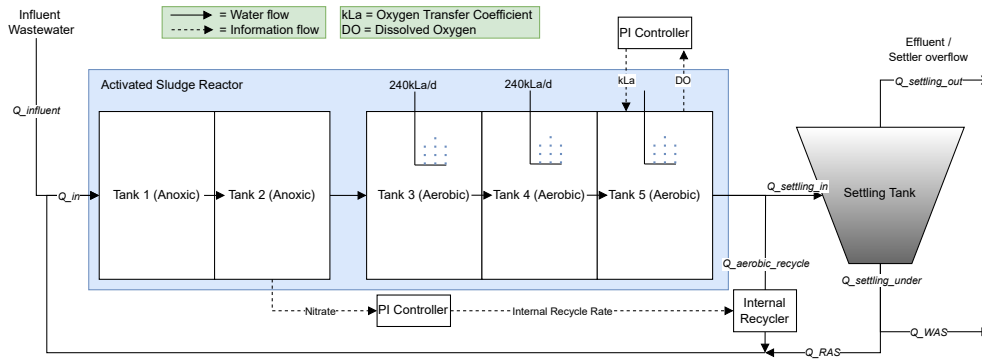


Figure 3.3: Layout for the default BSM1 plant

The two anoxic tanks each have a volume capacity of $1,000 \text{ m}^3$. They are non-aerated, meaning that no oxygen is actively supplied to these tanks, supporting

the denitrification process. At the start of the process, wastewater enters the first anoxic tank. In these tanks, heterotrophic bacteria convert nitrate NO_3^- into nitrogen gas N_2 using organic material as an energy source. This process prevents nitrate pollution in the discharged effluent. The nitrate required for denitrification is supplied through the internal recycling ($Q_{\text{aerobic_recycle}}$), which transports nitrate-rich water from the last aerobic tank (tank 5) back to the first anoxic tank (tank 1).

Following the anoxic tanks, the wastewater moves through three aerobic tanks, where nitrification occurs. Here, ammonium NH_4^+ is oxidized into nitrate NO_3^- by autotrophic nitrifying bacteria, such as:

- Nitrosomonas, which converts ammonium NH_4^+ into nitrite NO_2^- .
- Nitrobacter, which converts nitrite NO_2^- into nitrate NO_3^- .

After the biological treatment, the treated wastewater from the last aerobic tank, which has not been recycled, flows into the settler, which separates solids (biomass) from the treated water. The purpose of this is to maintain biomass in the tanks, because this is critical for optimal nitrification and denitrification. In the model, the settler is represented as a non-reactive, 10-layer vertical tank with a total volume of $6,000 \text{ m}^3$, a surface area of $1,500 \text{ m}^2$, and a total height of 4 m . The inflow to the settler ($Q_{\text{settling_in}}$) enters at the sixth layer from the bottom, designated as the feed layer.

The settler simulates key physical processes including gravity-driven settling, sludge thickening, and counterflow separation of clarified water and sludge. As the solids settle downward through the tank, clarified water rises and exits from the top as treated effluent ($Q_{\text{settling_out}}$). The settling dynamics are modeled using a one-dimensional flux model that incorporates hindered settling, compression settling, and dispersion [4]. An in depth description of the mathematical model of the settler is not included in this report, as it is not necessary to understand the primary objectives of the report. However, if interested in a full explanation, we refer to the previous report [3].

To maintain appropriate biomass levels in the biological reactor, two flows are extracted from the settler:

- **Waste Activated Sludge (WAS)**, denoted Q_{WAS} : Excess sludge is removed from the bottom of the settler to control the solids' retention time (SRT) and prevent overgrowth of biomass.

- **Return Activated Sludge (RAS)**, denoted Q_{RAS} : A portion of the settled biomass is returned to the beginning of the biological process to maintain a high concentration of active microorganisms, thus sustaining effective denitrification and nitrification.

3.2.3 BSM1 Dynamics

The following section 3.2.3 is taken from the previous report [3], but has been modified to fit within this report.

BSM1 uses the Activated Sludge Model No. 1 (ASM1) to describe how the concentrations of the compounds in each of the activated sludge tanks evolve and change over time. Both anoxic tanks and the aerobic tanks are activated sludge tanks in the BSM1 plant. ASM1 describes the different biological processes and how they influence the different dynamics of the concentration of compounds in the anoxic and aerobic tanks. ASM1 assumes that the temperature in the tanks is constant. As the name ASM1 suggests there are successors to ASM1 that model more biological processes that are not accounted for in ASM1. For the scope of this project the ASM1 is sufficient as it effectively captures the basic dynamics of a WWTP.

The ASM family of models represents a progression in activated sludge modeling, each building upon the previous version to enhance the representation of biological wastewater treatment processes. ASM1 laid the foundation, focusing on carbon and nitrogen removal. ASM2 introduced phosphorus dynamics, expanding the model's scope to include biological phosphorus removal. ASM2D further refined this by incorporating additional denitrification processes. ASM3 took a different approach, modifying microbial decay mechanisms and adjusting how growth and storage compounds are handled, leading to a more realistic simulation of sludge behavior [11].

Model Compounds

The carbon material that is in ASM1 is divided into biodegradable chemical oxygen demand (COD), non-biodegradable COD and biomass. The substances which are soluble are denoted (S) and particulate compounds denoted (X). The biodegradable COD is further divided into readily biodegradable substrate (S_S), which represent the simple soluble molecules that are readily absorbed by organisms. Slowly

biodegradable substrate (X_S), these compounds are assumed to be made up from particulate organic molecules, and require breakdown of enzymatic, prior to the absorption process. Likewise, the non-biodegradable COD are divided into soluble (S_I) and particulate material (X_I) but are seen as unaffected by the biological system. Lastly, the active biomass, which is divided into following types of organisms: heterotrophic biomass (X_{BH}) and autotrophic biomass (X_{BA}). (X_P) is included for inert particulate products, stemming from the biomass decay. Figure 3.4 shows the relations of the compounds just mentioned [12].

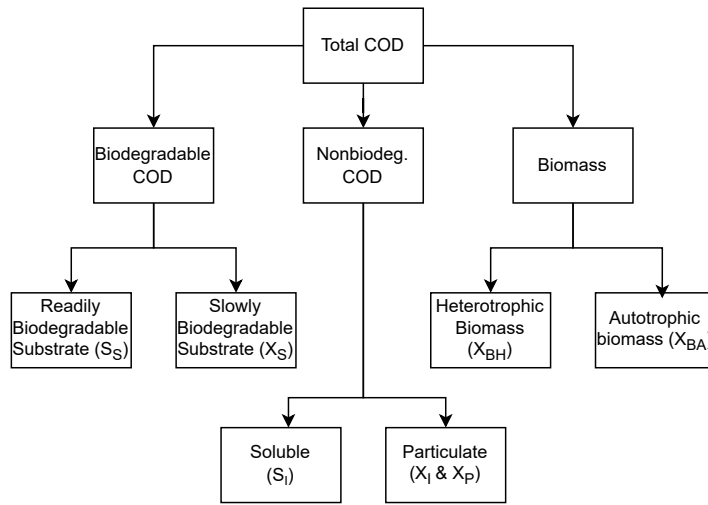


Figure 3.4: A tree of the carbon material and their relation to each other in ASM1

In ASM1 there is also nitrogenous material in the wastewater, also known as the total Kjeldahl nitrogen (TKN), which is the sum of all the nitrogen bound in the water. The nitrogen bound compounds are ammonia nitrogen (S_{NH}), organically bound nitrogen, and active mass nitrogen, which is biomass that is assumed to be nitrogen. Organically bound nitrogen is divided into soluble and particulate fractions, where both are further divided into non-biodegradable and biodegradable organic nitrogen. Only soluble

Biochemical Terms

Biodegradable: Able to be broken down by bacteria or other organisms

Substrate: A material or substance used in for enzymatic reactions

Heterotrophic: Organisms that have to eat/consume food/material for nutrition

Autotrophic: Organisms that can produce their own food/material for nutrition from energy sources from the environment

and particulate biodegradable organic nitrogen is given a variable name here, those being (S_{ND}) and (X_{ND}) respectively. There is also nitrate nitrogen (S_{NO}) as part of the nitrogenous material. Figure 3.5 shows these relations of the nitrogenous materials. Lastly, in the model, we have the alkalinity (S_{ALK}) and dissolved oxygen concentration (S_O) being expressed as negative COD [12]. An overview of the compounds mentioned are listed in table 3.1.

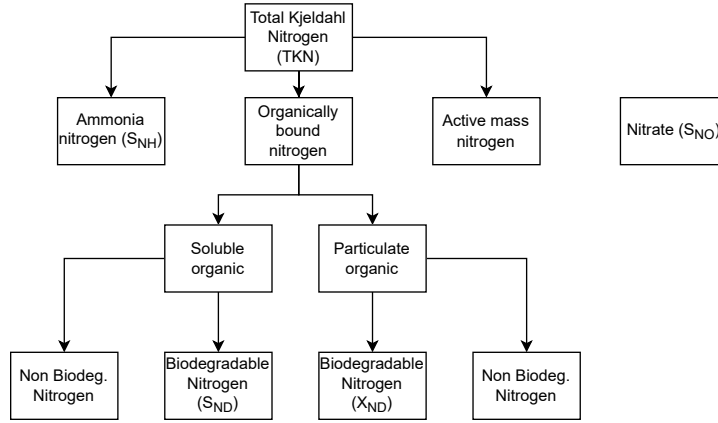


Figure 3.5: A tree of the nitrogenous material and their relation to each other in ASM1

| Definition | Notation |
|--|-----------|
| Readily biodegradable substrate | S_S |
| Slowly biodegradable substrate | X_S |
| Soluble inert organic matter | S_I |
| Particulate inert organic matter | X_I |
| Heterotrophic biomass | $X_{B,H}$ |
| Autotrophic biomass | $X_{B,A}$ |
| Particulate products arising from biomass decay | X_P |
| NH_4^+ and NH_3 ammonia nitrogen | S_{NH} |
| Soluble biodegradable organic nitrogen | S_{ND} |
| Particulate biodegradable organic nitrogen | X_{ND} |
| Nitrate and nitrite nitrogen | S_{NO} |
| Alkalinity | S_{ALK} |
| Oxygen | S_O |

Table 3.1: List of ASM1 concentration (g/m^3) variables.

In ASM1, each compound is measured as a concentration in a given tank. Be-

low is a list of all the differential equations for the compounds that explains their dynamics over time. Each differential equation reflects a weighted sum of different biological processes, which are weighted by how much those processes impact the compound. All the processes are explained in section 3.2.3. Looking at the first equation which is for readily biodegradable substrate (S_S), we can see that three different processes impact its dynamics, $AerGrowthHetero$, $AnGrowthHetero$ and $HydrolOfEntrOrg$, as well as the weights for each process being $\frac{-1}{Y_H}$, $\frac{-1}{Y_H}$ and 1 respectively. The weights and the processes contain constants which can be seen in table 3.2, containing what the parameter is, the symbol used for it and the value of it at 20°C.

$$\begin{aligned}
\frac{dS_S}{dt} &= \frac{-1}{Y_H} AerGrowthHetero + \frac{-1}{Y_H} AnGrowthHetero + HydrolOfEntrOrg \\
\frac{dS_O}{dt} &= \frac{-(1 - Y_H)}{Y_H} AerGrowthHetero + \frac{-(4.57 - Y_A)}{Y_A} AerGrowthAuto + \\
&\quad kLa(S_{O_saturation} - S_O) \\
\frac{dS_{NO}}{dt} &= \frac{-(1 - Y_H)}{2.86Y_H} AnGrowthHetero + \frac{1}{Y_A} AerGrowthAuto \\
\frac{dS_{ND}}{dt} &= -AmmonOfSolOrgN + HydrolOfEntrOrgN \\
\frac{dS_{NH}}{dt} &= -i_{XB} AerGrowthHetero + (-i_{XB} AnGrowthHetero) + \\
&\quad \left(-i_{XB} - \frac{1}{Y_A}\right) AerGrowthAuto + AmmonOfSolOrgN \\
\frac{dS_{ALK}}{dt} &= \frac{-i_{XB}}{14} AerGrowthHetero + \left(\frac{1 - Y_H}{14 \cdot 2.86Y_H} - \frac{i_{XB}}{14}\right) AnGrowthHetero + \\
&\quad \left(\left(-\frac{i_{XB}}{14} - \frac{1}{7 \cdot Y_A}\right) AerGrowthAuto\right) + \frac{1}{14} AmmonOfSolOrgN \\
\frac{dX_{BH}}{dt} &= AerGrowthHetero + AnGrowthHetero - DecayOfHetero \\
\frac{dX_{BA}}{dt} &= AerGrowthAuto - DecayOfAuto \\
\frac{dX_P}{dt} &= f_P \cdot DecayOfHetero + f_P \cdot DecayOfAuto \\
\frac{dX_S}{dt} &= (1 - f_P) DecayOfHetero + (1 - f_P) DecayOfAuto - HydrolOfEntrOrg \\
\frac{dS_S}{dt} &= (i_{XB} - f_P \cdot i_{XP}) DecayOfHetero + (i_{XB} - f_P \cdot i_{XP}) DecayOfAuto - \\
&\quad HydrolOfEntrOrgN
\end{aligned}$$

| Parameter | Symbol | Value |
|--|---------------------|----------|
| Heterotrophic yield | Y_H | 0.67 |
| Autotrophic yield | Y_A | 0.24 |
| Fraction of biomass yielding particulate products | f_P | 0.24 |
| Mass N/mass COD in biomass | i_{XB} | 0.08 |
| Mass N/mass COD in products from biomass | i_{XP} | 0.06 |
| Heterotrophic maximum specific growth rate | μ_H | 0.002778 |
| Heterotrophic decay rate | b_H | 0.000208 |
| Half-saturation coefficient (hsc) for heterotrophs | K_S | 10.0 |
| Oxygen hsc for heterotrophs | K_{OH} | 0.2 |
| Nitrate hsc for denitrifying heterotrophs | K_{NO} | 0.50 |
| Autotrophic maximum specific growth rate | μ_A | 0.000347 |
| Autotrophic decay rate | b_A | 0.000035 |
| Oxygen hsc for autotrophs | K_{OA} | 0.40 |
| Nitrate hsc for denitrifying autotrophs | K_{NH} | 1.0 |
| Correction factor for anoxic growth of heterotrophs | n_g | 0.80 |
| Ammonification rate | k_a | 0.000035 |
| Maximum specific hydrolysis rate | k_h | 0.002083 |
| Hsc for hydrolysis of slowly biodegradable substrate | K_X | 0.10 |
| Correction factor for anoxic hydrolysis | n_h | 0.80 |
| Maximum oxygen saturation | $S_{O_Saturation}$ | 8.0 |
| Oxygen transfer coefficient | kLa | varies |

Table 3.2: Parameters for the ASM1 model and their value at 20°C

ASM1 Biological Processes

In the ASM1 model there are 8 biological processes described and the equations to calculate them use the parameters in table 3.2 [12]. The processes are listed below with a short explanation of what the processes are. Each process also has an definition for it describing how much of that process occurs given the concentration of the compounds from section 3.2.3 in a tank:

1. Aerobic growth of heterotrophic biomass

With oxygen present, a fraction of available readily biodegradable substrate (S_S) is consumed for growth of heterotrophic biomass and dissolved oxygen (S_O) is also consumed.

$$AerGrowthHetero \stackrel{def}{=} \mu_H \cdot \frac{S_S}{K_S + S_S} \cdot \frac{S_O}{K_{OH} + S_O} \cdot X_{BH}$$

2. Anoxic growth of heterotrophic biomass

With no oxygen present, heterotrophic organisms will use nitrate (S_{NO}) and readily biodegradable substrate (S_S) to produce heterotrophic biomass and nitrogen gas.

$$AnGrowthHetero \stackrel{def}{=} \mu_H \cdot \frac{S_S}{K_S + S_S} \cdot \frac{K_{OH}}{K_{OH} + S_O} \cdot \frac{S_{NO}}{K_{NO} + S_{NO}} \cdot n_g \cdot X_{BH}$$

3. Aerobic growth of autotrophic biomass

Ammonia (S_{NH}) is oxidized to nitrate via nitrification, which results in the production of autotrophic biomass and leads to an increased demand for dissolved oxygen (S_O).

$$AerGrowthAuto \stackrel{def}{=} \mu_A \cdot \frac{S_{NH}}{K_{NH} + S_{NH}} \cdot \frac{S_O}{K_{OA} + S_O} \cdot X_{BA}$$

4. Decay of heterotrophic biomass

Heterotrophic biomass (X_{BH}) dies at a certain rate and part of it becomes non-biodegradable substrate (X_P) and the other part becomes slowly biodegradable substrate (X_S).

$$DecayOfHetero \stackrel{def}{=} b_H \cdot X_{BH}$$

5. Decay of autotrophic biomass

Same as decay of heterotrophic biomass but for autotrophic biomass (X_{BA}).

$$DecayOfAuto \stackrel{def}{=} b_A \cdot X_{BA}$$

6. Ammonification of soluble organic nitrogen

Biodegradable soluble organic nitrogen (S_{ND}) is converted to ammonia.

$$AmmonOfSolOrgN \stackrel{def}{=} k_a \cdot S_{ND} \cdot X_{BH}$$

7. Hydrolysis of entrapped organics

Slowly biodegradable substrate (X_S) is broken down and becomes readily biodegradable substrate (S_S).

$$HydrolOfEntrOrg \stackrel{def}{=} k_h \cdot \frac{X_S/X_{BH}}{K_X + X_S/X_{BH}} \cdot \left(\frac{S_O}{K_{OH} + S_O} + n_h \cdot \frac{K_{OH}}{K_{OH} + S_O} \cdot \frac{S_{NO}}{K_{NO} + S_{NO}} \right) \cdot X_{BH}$$

8. Hydrolysis of entrapped organic nitrogen

Biodegradable particulate organic nitrogen (X_{ND}) is broken down to soluble organic nitrogen

$$HydrolOfEntrOrgN \stackrel{def}{=} \left(k_h \cdot \frac{X_S/X_{BH}}{K_X + X_S/X_{BH}} \cdot \left(\frac{S_O}{K_{OH} + S_O} + n_h \cdot \frac{K_{OH}}{K_{OH} + S_O} \cdot \frac{S_{NO}}{K_{NO} + S_{NO}} \right) \cdot X_{BH} \right) \cdot \frac{X_{ND}}{X_S}$$

3.2.4 Plant Controllers

As seen in Figure 3.3, there are two PI (Proportional-Integral) controllers. PI-controllers are a variant of PID controllers, which leaves out the derivative term. PID controllers respond to the error of a system $e(t) = PV_t - SP$, where PV_t is a process variable at time t and SP is the set point, which is the value you want the PI-controller to guide the process variable towards. Each letter serves a particular purpose. The proportional term $K_p e(t)$ is used to amplify the *present error*. The control signal is greater the larger the error is. The integral term $K_i \int_0^t e(\tau) d\tau$ is an accumulation of *past error*. If an error goes uncorrected over time then the integral term would increase, resulting in a larger control signal. The derivative term $K_d \frac{de(t)}{dt}$ can be seen as an indication *future error*. The errors rate of change indicates how large the control signal should be [13]. The coefficients for the integral and derivative term are defined in proportion to the coefficient of the proportional term:

$$K_i = \frac{K_p}{T_i} \text{ and } K_d = K_p T_d$$

The full equation for a PI-controller can be seen in equation (3.1) and is just a sum of the two different terms.

$$u(t) = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau \right) \quad (3.1)$$

One of the controllers for the BSM1 plant looks at the concentration of nitrate in tank 2, and based on the setpoint changes how much of the flow coming out of tank 5 is recycled. The other controller senses the concentration of dissolved oxygen in tank 5 then controls how much the aeration devices should add, in simulations it is setting the oxygen transfer coefficient kLa .

In [4] there are also other controllable actions suggested, also listed in appendix A. In this paper we have tried four controls from this list, those being controlling the recycling as mentioned above and controlling the kLa in tank 3, 4 and 5.

Energy usage of aeration

As mentioned in 3.2.1, the aeration process energy usage is great. A differential equation is used to model the energy usage of the aeration devices. Equation (3.2) has the differential equation where OTR_E is the oxygen transfer rate per energy inputted, with oxygen transfer rate being the amount of oxygen gas that passes

through the water over a given period, $S_{O_saturation}$ is the maximum oxygen saturation possible in water at a certain temperature and pressure, $kLa_i(t)$ is the oxygen transfer coefficient from the aeration device in tank i at time t and V_i is the volume of tank i .

$$\frac{dE_{aeration}}{dt} = \sum_{i=1}^5 (1/OTR_E) S_{O_saturation} \cdot kLa_i(t) \cdot V_i \quad (3.2)$$

3.2.5 Base control

In BSM1, a basic control strategy is implemented to assess plant performance. In the base control strategy, the first two aerobic tanks maintain stable oxygen concentrations to support microbial activity, with a fixed oxygen transfer coefficient of $kLa = 10h^{-1} = 240d^{-1}$, meaning that oxygen is continuously supplied at a constant rate. Unlike the two first aerobic tanks, the last aerobic tank actively controls the dissolved oxygen (DO) concentration, maintaining it at $2\frac{g}{m^3}$ using a PI-controller.

The internal recycle flow rate is actively controlled by a PI-controller with a nitrate concentration setpoint of $1\frac{g}{m^3}$, to optimize denitrification efficiency. The flow rate of internal recirculation $Q_{aerobic_recycle}$ is constrained within the range of 0 to 5 times the influent flow rate $Q_{municipality_out}$ to ensure process stability [4].

A full list of all controllable components in the BSM1 plant can be seen below [4].

- Internal flow recirculation rate ($Q_{aerobic_recycle}$)
- Return sludge flow rate (Q_{RAS})
- Wastage flow rate (Q_{WAS})
- Anoxic/aerobic volume – all five biological reactors are equipped with both aerators and mechanical mixing devices; i.e., in a discrete fashion, the volumes for anoxic and aerobic behavior can be modified;
- Aeration intensity individually for each reactor
- External carbon source flow rate for each activated sludge tank, where the carbon source is considered to consist of readily biodegradable substrate, i.e., COD_{EC} ;

- Influent distribution by use of step feed (fractions of the influent flow to each of the five biological reactors)
- Distribution of internal flow recirculation (fractions of the internal recirculation flow to each of the five biological reactors)
- Distribution of return sludge flow (fractions of the return sludge flow to each of the five biological reactors)

3.2.6 Control evaluation

The following section will describe how the quality of a particular control strategy is evaluated in the BSM1 framework. A control strategy can for example be the base control strategy presented previously in section 3.2.5. New control strategies can be created by utilizing the available control handles from the list in section 3.2.5.

There are multiple factors that contribute to the overall quality of a control strategy. However, the most prominent and simple measurement is effluent quality (EQ). This report will mostly rely on EQ for the evaluation of different control strategies. The following description of the EQ function has been carried over from the previous report [3].

The function can be seen in (3.3) and it defines a differential equation for calculating the weighted amount of pollutant in the plant effluent. The weights and their value can be seen in table 3.3. These are included as different compounds have different effects on the quality of the receiving water.

$$\frac{dEQ}{dt} = (W_{SS} \cdot SS_e + W_{COD} \cdot COD_e + W_{NKj} \cdot S_{NKj,e} + W_{NO} \cdot S_{NO,e} + W_{BOD5} \cdot BOD_{5,e}) \cdot Q_{settling,out} \quad (3.3)$$

| Weight | W_{SS} | W_{COD} | W_{NKj} | W_{NO} | W_{BOD5} |
|---|----------|-----------|-----------|----------|------------|
| Value (g pollution unit · g ⁻¹) | 2 | 1 | 30 | 10 | 2 |

Table 3.3: W_i values

In equation (3.3) the summation of the five weighted factors, is multiplied by the water flow $Q_{settling,out}$ ($\frac{m^3}{min}$), to get the total weighted pollutant mass flow, EQ.

For clarity, in each equation: (3.4), (3.5), (3.6), (3.7), the $,e$ subscript means the compound concentration of the plant effluent. The factor with the largest weight is Kjeldahl Nitrogen ($S_{NKj,e}$), which can be seen in equation (3.4). This accounts for all nitrogenous material in the water, except for S_{NO} , which is separately weighted. The nitrogen compounds in Kjeldahl Nitrogen, belong to the Plant Nutrients pollutant group, as seen in Figure 3.1 in section 3.1.

$$S_{NKj,e} = S_{NH,e} + S_{ND,e} + X_{ND,e} + i_{XB} (X_{B,H,e} + X_{X,A,e}) + i_{XP} (X_{P,e} + X_{I,e}) \quad (3.4)$$

Equation (3.5) considers the compounds that are part of the Suspended Solids pollutant group and lastly, equations (3.6) and (3.7) consider biological oxygen demand and chemical oxygen demand, which are part of the Organic Material pollutant group. In equation (3.6) f_P is the fraction of biomass yielding particulate products, previously presented in table 3.2.

$$SS_e = 0.75 \cdot (X_{S,e} + X_{I,e} + X_{B,H,e} + X_{B,A,e} + X_{P,e}) \quad (3.5)$$

$$BOD_{5,e} = 0.25 \cdot (S_{S,e} + X_{S,e} + (1 - f_P) \cdot (X_{B,H,e} + X_{B,A,e})) \quad (3.6)$$

$$COD_e = S_{S,e} + S_{I,e} + X_{S,e} + X_{I,e} + X_{B,H,e} + X_{B,A,e} + X_{P,e} \quad (3.7)$$

In table 3.4 below, the concentration limits for the different pollutants in the effluent can be seen. In the BSM1 framework, these limits are not treated as safety properties, and the base control strategy presented in section 3.2.5 exceeds these limits frequently. How exactly such effluent limits are governed differs from country to country and region to region. According to DHI, in Denmark, the government implements these effluent limits as strict safety properties where the specific pollutant limits differ from WWTP to WWTP. However, in Denmark, despite complying with these limits, the WWTPs are taxed based on the amount of pollutants in the effluent [14]. Here, N_{tot} is equal to $S_{NO,e} + S_{NKj,e}$.

| Variable | Value |
|-----------|---|
| N_{tot} | $< 18 \text{ g N} \cdot \text{m}^{-3}$ |
| COD | $< 100 \text{ g COD} \cdot \text{m}^{-3}$ |
| S_{NH} | $< 4 \text{ g N} \cdot \text{m}^{-3}$ |
| SS | $< 30 \text{ g SS} \cdot \text{m}^{-3}$ |
| BOD_5 | $< 10 \text{ g BOD} \cdot \text{m}^{-3}$ |

Table 3.4: Effluent pollutant limits [4]

3.2.7 Weather scenarios

The influent data is provided by [4]. The time is given in days, the flow rate is expressed in cubic meters per day ($m^3.d^{-1}$), and the concentration is expressed in grams per cubic meter ($g.m^{-3}$). Within the BSM1 framework, three different weather scenarios are presented: Dry weather, storm weather, and rain weather. Each scenario contains two weeks of dynamic weather data, and are downloaded from (<http://www.benchmarkWWTP.org/>).

Dry Weather

The influent data for the dry weather scenario consists of two weeks of dynamic dry weather.

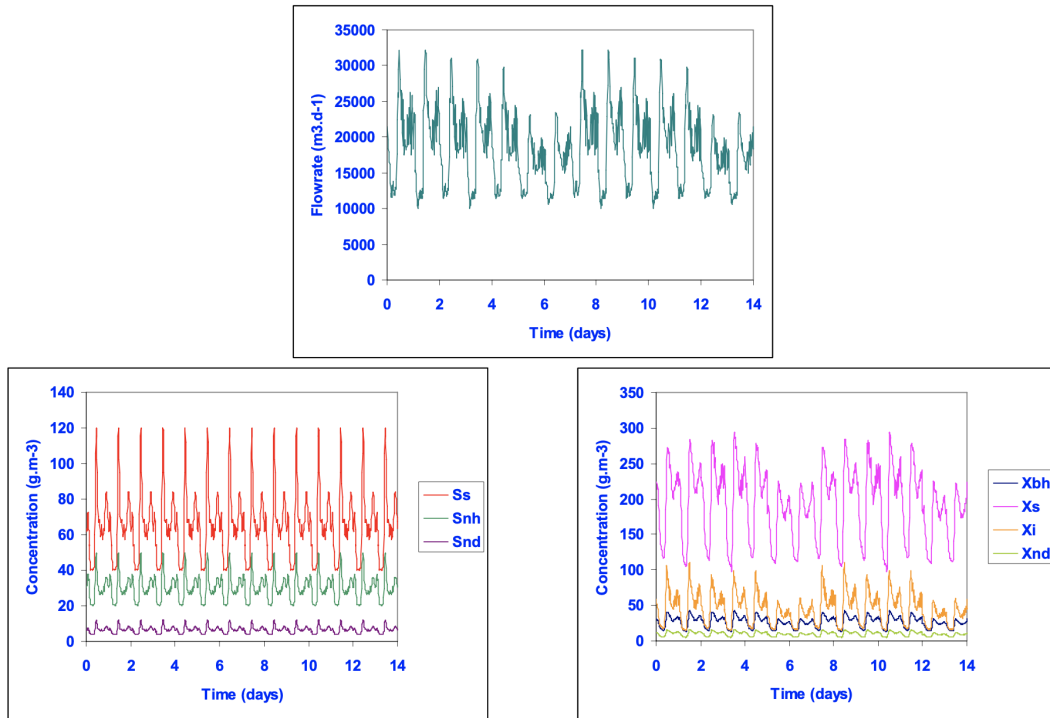


Figure 3.6: Dry Weather influent

Figure 3.6 illustrates the influent characteristics under a dry weather scenario. This influent emulates wastewater in steady, dry conditions. The x-axis represents time, spanning a period of 14 days.

In the top diagram, the y-axis shows the flow rate in units of $(g \cdot m^3 \cdot d^{-1})$. The average flow rate is approximately $18,446 m^3/day$, with clear diurnal fluctuations that reflect typical residential and industrial wastewater generation patterns.

In the bottom diagrams, the y-axes represent the concentration of various compounds in units of $g \cdot m^3$.

The left diagram shows the compound's readily biodegradable substrate (S_S), ammonium (S_{NH}) and soluble biodegradable organic nitrogen (S_{ND}), with a clear fluctuating pattern each day.

The right diagram shows the compounds heterotrophic biomass (X_{BH}), slowly biodegradable substrate (X_S), particulate inert organic matter (X_I), and particulate biodegradable organic nitrogen (X_{ND}), which also remains relatively stable over time. Here we do see some variations on the weekends.

Storm Weather

The influent data for the storm weather scenario consists of one week of dynamic dry weather followed by a second week that includes two storm events imposing on the dry conditions.

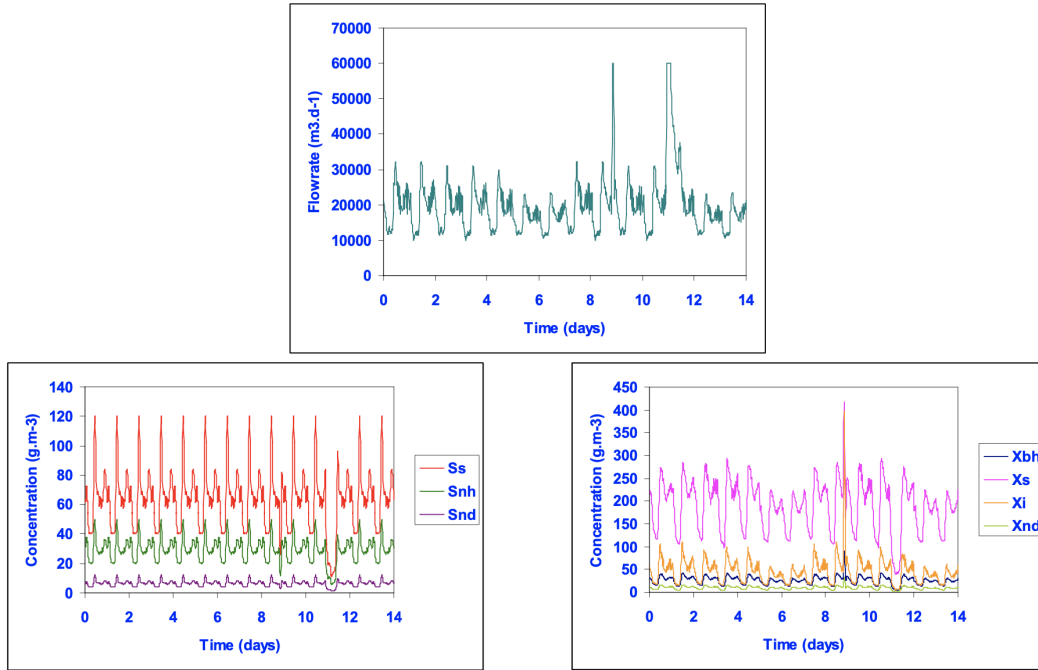


Figure 3.7: Storm Weather influent

Figure 3.7 illustrates the influent characteristics under a storm weather scenario. Here we have the same three diagrams as the dry weather, with notable spikes in the flow rate and X_L , and X_{BH} , and dips in S_S , S_{NH} and S_{ND} during the storm event.

Rain Weather

The influent data for the rain weather scenario consists of dynamic dry weather during the first week, while the second week features a prolonged rain event.

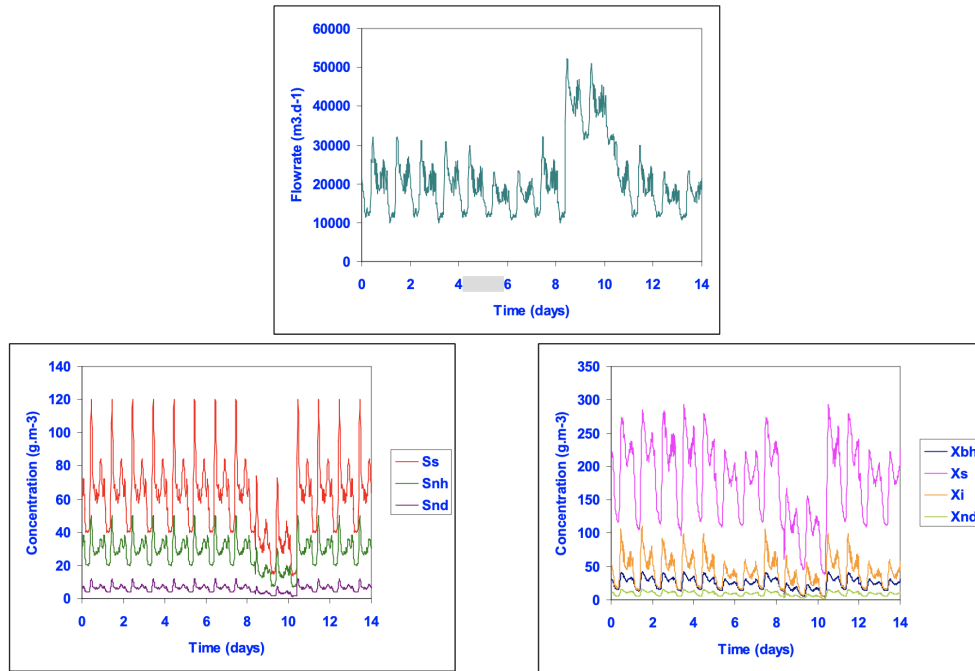


Figure 3.8: Rain Weather influent

Figure 3.8 illustrates the influent characteristics under a rainy weather scenario. Here we have the same three diagrams as the dry weather, with notable increase of flow rate and dips of all the compound concentrations during the rain event in the second week, due to increased flow of water.

3.3 BSM1 & MLE differences

As mentioned in the introduction, this report is a continuation of the previous report, that had a focus on the Modified Ludzack-Ettinger (MLE) process. MLE and BSM1 share a common foundation in biological nitrogen removal but differ in complexity, implementation, and purpose. These similarities and differences can be seen when comparing the layout of the plants of BSM1 plant in Figure 3.3 and MLE plant in Figure 3.9. MLE serves as a widely applied process design in operational wastewater treatment plants (WWTPs), whereas BSM1 is a simulation model and a framework designed to research and evaluate different controls strategies for optimization.

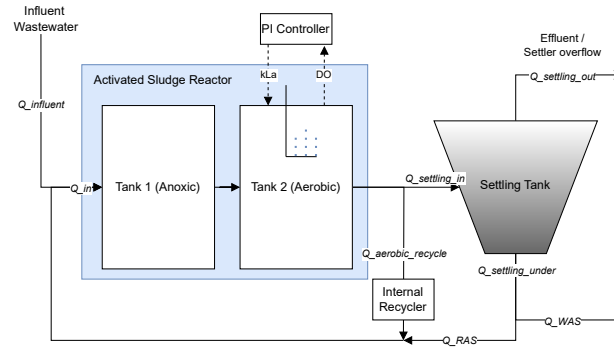


Figure 3.9: Layout of an MLE plant

Both the MLE and BSM plants are based on the principle of pre-denitrification, where nitrate produced during aerobic treatment is recycled back to an anoxic zone. The process, as illustrated in both Figures, starts in an anoxic tank where denitrification occurs, followed by an aerobic tank (or sequence of tanks) facilitating nitrification. The nitrate formed in the aerobic section is recycled to the anoxic section to be converted into nitrogen gas by heterotrophic bacteria. In both systems, a settling tank follows the biological treatment to separate solids from treated effluent. Additionally, the Figures depict sludge recycle and sludge removal lines. Returned activated sludge (RAS) helps sustain microbial populations, while waste activated sludge (WAS) removal regulates solids retention time.

Despite these shared design principles, Figure 3.3 illustrates BSM1's increased complexity relative to the simplified layout of MLE in Figure 3.9. BSM1 implements a more detailed structure by dividing the biological reactor into five sequential tanks: the first two (Tank 1 and Tank 2) are anoxic, while the last three (Tanks 3 to 5) are aerobic. Each aerobic tank features an oxygen transfer capacity of 240 kLa/d, with a PI-controller regulating dissolved oxygen levels dynamically in tank 5. In contrast, the MLE plant (Figure 3.9) consolidates the anoxic and aerobic processes into single tanks, with aeration managed via a PI-controller in the aerobic tank.

The internal nitrate recycle loop—essential to pre-denitrification—is present in both diagrams. In an MLE plant, it is a simple loop transferring flow from the aerobic to the anoxic tank. This is the same for the BSM1 plant from Tank 5 to Tank 1 with the addition that the amount recycled is controlled by a PI-controller.

3.4 Model Predictive Control

Model Predictive Control (MPC) is a way to create advanced control strategies for a dynamic system in a changing environment. In MPC, there is the real system and a model of the system. The model is used to predict where the real system is headed, and this allows for testing how actions taken in the real system could affect the future. By having an optimization criterion the best actions can be estimated, also known as a strategy.

Definition 3.4.1 (Deterministic Strategy) *Let $S \in \mathbb{R}^k$ denote the state space of a system, and A the set of possible actions. A (deterministic) strategy σ is a function*

$$\sigma : S \rightarrow A$$

that maps each state $s \in S$ to an action $\sigma(s) \in A$.

The model of the system need not be an highly precise estimate of the real system to be useful. The model might deviate from the real system due to a number of factors such as relying on forecasts, systems containing stochastic behavior or approximations of real behavior. The model will be less like the real system over time due to the accumulation of such factors, and therefore only the first action of a learned strategy is applied for a given control period P . A new strategy is then learned from the state of the real system after applying the action, and the procedure is repeated [15]. This behavior can be seen in Figure 3.10a, where the predicted dashed line extends from the real systems solid blue line state. The solid green line is the actions that have been taken, and the dashed green line is the future actions given by the strategy calculated when in the predicted states. But as time progresses on Figure 3.10b the solid line deviates from the previously predicted trajectory and a new dotted blue line prediction is made, and the strategy is recalculated and thus changes to fit the new prediction.

This cycle is shown in Figure 3.11 with three parts to it. At time $t = k$, the initial state $x(k)$ is passed from the MPC controller to the optimizer (OPT), which calculates an optimal strategy $\sigma_{x(k)}^*$ that is returned to the MPC controller. The MPC controller then passes the action $\sigma_{x(k)}^*(x(k))$ to the environment (ENV). The environment uses the action and after P time the new state $x(k + P)$ passed back to the MPC controller and the cycle begins again.

The framework does not rely on a specific method for optimizing, and any method that can create a strategy can be used. This makes the framework very flexible. When calculating a strategy the optimizer predicts to a horizon H which

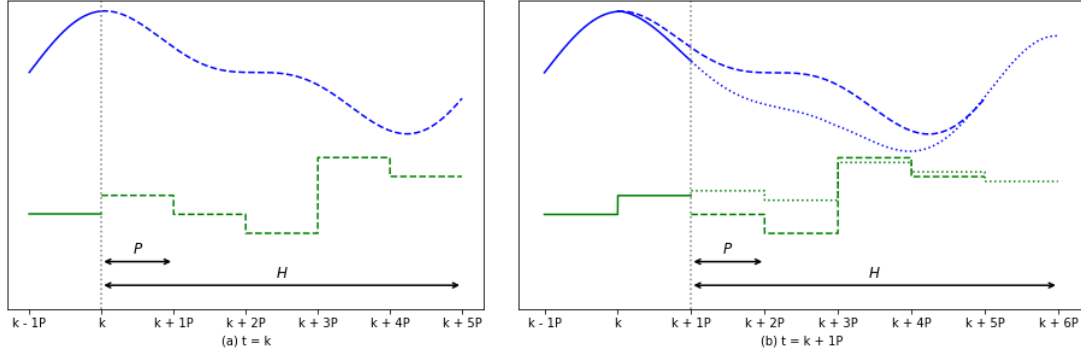


Figure 3.10: Model Predictive Control: Figure (a) shows the systems trajectory at time $t = k$ with a control Period P and a horizon H . The blue line represents the trajectory of the system's state, with the solid portion indicating the past trajectory and the dashed portion indicating the predicted future trajectory. Below, the green line depicts the sequence of control actions, where the solid portion corresponds to past control actions and the dashed portion to the future control actions. Figure (b) illustrates the system's trajectory at time $t = k + P$. A dotted line is added for both the system's state and the control actions, representing the updated predicted future trajectory. The Figures and caption is from our previous report [3].

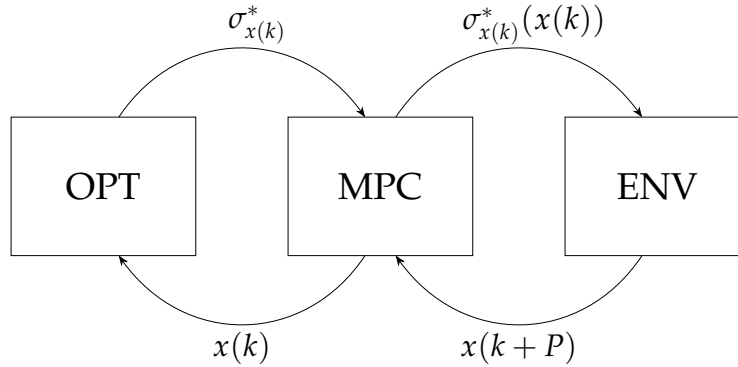


Figure 3.11: MPC framework

is a multiple of P . Both P and H are targets for optimization, where finding good values for them can improve the optimizer in both speed and performance. Looking too far ahead in a prediction might lead to learning strategies on wildly inaccurate scenarios, but it could also be good if there is a future weather forecast that needs actions early to prepare for it.

One thing Figure 3.11 does not account for is the time it takes to calculate a good strategy. The Figure assumes that it is instant, but in reality it can take quite a while. Since we need an action once every P time, the P is also the maximum time we have to calculate a strategy. So in order to have a strategy ready one could

start calculating strategy for time k , one period early at time $k - P$ and therefore use the previous state $x(k - P)$. This means learning on not the most up-to-date state.

A type of optimizer that works well with MPC is reinforcement learning. Reinforcement learning is a type of machine learning where a agent tries different actions in an environment and evaluating the outcome based on a cost or reward function. Doing so it can learn which actions provide the most benefit. Not just in the short run but also over a longer period. There is a balance in rewarding immediate beneficial actions and long term beneficial actions.

Chapter 4

Related Work

This section will explore related work on control of WWTP processes using MPC, as well as, literature surrounding the combination of MPC and Reinforcement learning.

4.1 BSM1 and MPC

In the field of WWTP optimization, a study [16] utilizing the BSM1 benchmark proposed the application of Model Predictive Control (MPC). The study focuses on developing a systematic tuning procedure for MPC to optimize internal recycling flow and air flow rate by adjusting the oxygen transfer coefficient. The goal is to control effluent quality while minimizing operational costs. The paper evaluates MPC performance under different tuning strategies, analyzing their impact through various simulations. The results highlight the advantages of MPC over traditional control methods, particularly in reducing overshoot and response time. Both our report and the study apply advanced MPC to the BSM1 benchmark, using it as the control mechanism for WWTP operations. However, while the [16] study relies on fine-tuning MPC parameters through systematic adjustments to achieve optimal setpoints. We propose integrating reinforcement learning (RL) to enhance control adaptability. This approach enables the system to learn and improve performance dynamically, rather than depending solely on predefined tuning methods. Additionally, our study incorporates real-time energy prices to further optimize operational costs.

In the discovery of related works, no study has been found that combines MPC and reinforcement learning to integrate intelligent control in the BSM1 plant.

4.2 Combining MPC & Reinforcement Learning

The following section on combining MPC and reinforcement learning has been retained from the previous report [3], as these studies remain relevant.

As explored in the previous section, no work in the literature was found that uses reinforcement learning with MPC to optimize BSM1 processes. In order to showcase the efficiency and potential of combining MPC and reinforcement learning, this section will present a few examples in current literature that uses MPC with reinforcement learning, applied to a problem setup that is similar to that of this project. Firstly, the general problem setup consists of a dynamical system. In relation to MLE, this is the set of biological and physical dynamics that explain how essential compounds evolve over time, in both the anoxic tank, the aerobic tank and the clarifier. The second component of the general problem setup is an optimization problem that involves a quality parameter, a cost parameter, or both. For MLE, there is both a quality parameter (Effluent quality) and a cost parameter (Energy consumption).

Goorden et al. [17] utilizes MPC and reinforcement learning through the UPPAAL tool suite [18, 19, 20] to optimize water discharge quality from storm water detention ponds and avoid stream erosion. UPPAAL was used to model the dynamical system (flow of water) and the extension UPPAAL Stratego was used to find near-optimal control strategies for the outflow regulation. The optimization parameter was here to maximize the water level in the pond, and thus maximizing particle sedimentation, while satisfying a safety constraint of no overflow. The result of applying MPC with reinforcement learning was a 95% overflow duration reduction and a 29% pollutant sedimentation improvement compared to traditional static control.

Hasrat et al. [21] also utilizes MPC and reinforcement learning through the UPPAAL tool suite. They use automatic model identification to model the thermodynamics of a house and its rooms. They then use UPPAAL Stratego to synthesize near-optimal control strategies for the control of a domestic floor-heating setup, optimizing for both comfort and energy cost, by looking at hourly energy prices and weather forecasts. Utilizing MPC and reinforcement learning, they save up to 33% energy cost, while maintaining the same comfort level as a traditional controller.

Chapter 5

Problem Statement

Wastewater treatment plants are necessary to remove harmful pollutants from incoming wastewater, before it is released into surrounding water bodies. However, in the biological removal stage, a significant amount of energy is used to blow oxygen into the biological reactors. The amount of oxygen input is often controlled by fixed setpoints which are altered manually by humans. Such manual control may be inefficient, creating the possibility of improving energy consumption and quality of the outgoing water (effluent) through intelligent control using MPC and reinforcement learning.

In related works, we explored previous studies that were successful in combining MPC and reinforcement for intelligent control of various real-life systems. As these studies utilized UPPAAL and UPPAAL Stratego, this project will also employ UPPAAL for intelligent control of the BSM1 plant. While this project's setup resembles those in *Goorden et al.* [17] and *Hasrat et al.* [21], the BSM1 plant involves a significantly more complex dynamical system than those addressed in the referenced studies.

With this motivation, we construct the following problem statement, and below it, a set of associated challenges.

Can MPC and reinforcement learning be used to reduce energy consumption/cost and improve effluent quality of the BSM1 plant?

Challenges

- *Simulation efficiency* - As reinforcement learning is a simulation-based method, it is important that simulation of a given system is quick and efficient. This proposes a challenge as WWTP models are very complex, where even the relatively simple ASM1 model consists of 13 differential equations for each activated sludge tank in the model. It is therefore important that work be put into optimizing simulation speed, to increase the likelihood of computing effective control strategies in acceptable time.
- *Partial observability & Measurement noise* - In an MPC setup, you would typically receive periodic information from sensors in the real world system, to update the true state of the system model. In BSM1, this could be information about the concentration of compounds in the different activated sludge tanks. However, in most real WWTPs such sensors do not exist, are too expensive to invest in or are sensitive to noise. Instead, manual tests are performed, but with a very low frequency, if even at all. For an MPC setup to work efficiently, the true state of the system needs to be updated every period, but here this is not accessible.
- *Parameter estimation* - The simulation accuracy of a WWTP model is largely determined by the correctness of the parameters used to calculate the biological, chemical, and physical behaviors of the system. That is, the closer the simulation parameters are to the actual real world parameters, the more accurately the system can be simulated in an MPC setup. However, these parameters change dynamically over time, which means they have to be constantly updated for accurate simulation of the system.
- *Inflow characteristics prediction model* - The characteristics of the wastewater going into the WWTP are unknown beforehand. As such, a model that can predict the future characteristics of the incoming wastewater is desired to assist the reinforcement learning algorithm in finding control strategies that are effective, not only for the current state of the system, but also the future state of the system.

Resolving these challenges is necessary to realistically implement such an MPC setup on a real WWTP. As such, in this paper, we propose a pipeline that addresses each of these challenges in the MPC setup. A description of the pipeline can be found in section 7.1.

Chapter 6

Tools

6.1 WEST & Tornado API

The following section 6.1 is from the previous report [3], and has been adapted to fit the BSM1 plant layout.

WEST by DHI is a software tool designed for modeling and simulating wastewater treatment plants (WWTPs). WEST serves as a digital platform for exploring, designing, and optimizing the biological and chemical processes within WWTPs.

WEST enables users to refine plant design, operations, and automation. The tool supports the creation of performance indicators under dynamic simulation conditions and allows for experimental queries.

One of WEST's competence is its ability to create virtual replicas of real systems, providing a testing ground to evaluate, compare, and validate real-world processes by simulating various operational strategies and configurations.

6.1.1 BSM1 modeled in WEST

Figure 6.1 below provides an overview of the BSM1 process as modeled in WEST. At the leftmost part of the process, we see the "municipality," which represents the source of the wastewater. In this simulation, the incoming inflow from a municipi-

pality, containing organic matter, nutrients, and solids, is defined using a file with detailed information.

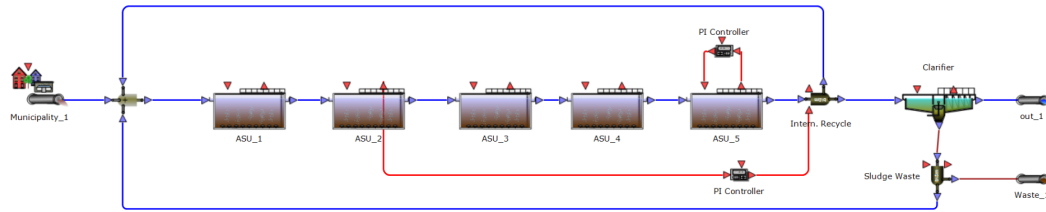


Figure 6.1: A screenshot of the graphic BSM1 model in WEST.

The entire incoming flow is directed to the first anoxic tank 1 (ASU_1 in the Figure), a reactor where denitrification takes place. This process involves reduction of nitrates (NO_3^-) to nitrogen gas (N_2). Removing nitrogen compounds from the wastewater. All the blue lines represent flow.

After the removal of nitrogen in the Anoxic Tanks, the flow is sent to the Aerated Tanks, which is responsible for nitrification and organic matter removal. The aeration PI-controller supplies oxygen to the tank.

From the last aerated tank (ASU_5), a portion of the flow is recirculated back to ASU_1 through internal recirculation. This recirculated water brings nitrate from the Aerated Tank back to the Anoxic Tank, to further remove nitrate from the system.

After internal recirculation, the flow continues to the settling tank, *also called the Clarifier*, where the separation of solids (sludge) from treated water occurs. The treated water (effluent) exits through the outlet ("out"), while solids settle at the bottom of the tank as sludge.

Part of the settled sludge is recycled back to the beginning of the process. Any excess sludge is removed as waste sludge.

The BSM1 dynamics described in 3.2.3 are similarly defined in WEST. These are found in a tool called the "Model Editor". Here one can also find all important parameters and constants for the modeling of BMS1, as well as, code for the functionality of different components, such as splitters and aeration control.

6.1.2 Simulation in WEST

The Figure 6.2 provides an overview of the WEST simulation tool. It displays the control center, where simulations can be run to analyze the model. The dynamic setting is used for simulating the model using a dynamic inflow dataset.

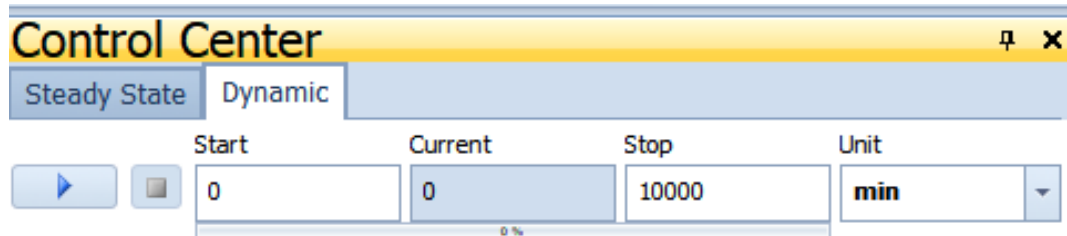


Figure 6.2: The Control Center in WEST to configure simulations of the model.

One can change the unit of the simulation and the duration of the simulation, as well as, the start and stopping point of the simulation.

6.1.3 Results in WEST

In the below Figure 6.3, the result of the specific variables in the simulation can be seen. We can see that it is possible to plot all variables in the model through the graph tools in WEST. This result showcases how the inflow to the anoxic tank (light blue line) changes over the given time frame, and how the internal recycling inflow (the orange line) changes in the given time frame. Lastly, variables can be checked and unchecked to be visible and hidden from in the graph.

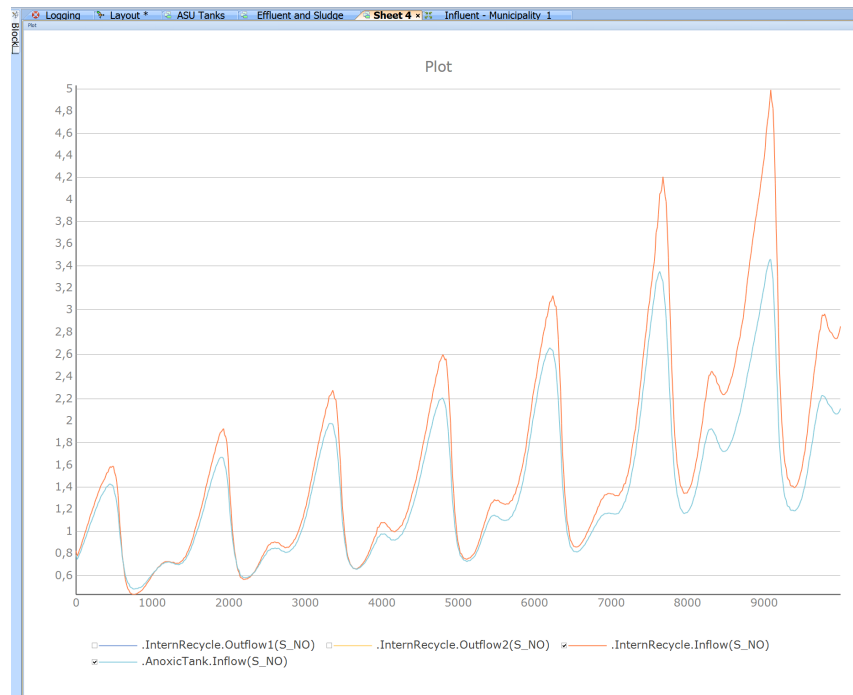


Figure 6.3: An example of plotting different variables in the BMS1 process during simulation.

6.1.4 Tornado API

WEST also have an API called Tornado, which can be used to programmatically control much of the functionality of WEST. Such as initializing variables, running simulations, changing variables mid-simulation and retrieve the results of simulations. Not much documentation was provided for Tornado, making it hard to work with going only off of function signatures. Tornado is available as a .NET DLL, allowing for it to be imported into python using a library such as pythonnet. Tornado is built around handling buffer files. The buffer files are huge .XML files which hold the entire state of an “experiment”, which is what a project in WEST is called. Some useful methods in the Tornado are:

- `Tornado.ExpLoad(path: string)`: loads the initial state of an experiment from a given path as a string, and returns an experiment `dynExp` object.
- `dynExp.ExpSetInitialValue(variable: string, value: float)`: Sets the initial value for one of the variables in the experiment, useful for applying actions from a strategy.

- `dynExp.Run()`: runs a simulation based on the values set for the start and stop times. These can be set in WEST control center as shown in Figure 6.2.

Due to limited documentation, we did run into some problems. For example, we needed to set the start and stop times of the experiments. None of the functions we tried in the API seemed to do what we wanted, but as mentioned huge .XML buffer files make up the experiment, and as such we could just change those values directly in the .XML file.

6.2 UPPAAL

UPPAAL is a tool used for modeling, simulation, and verification of real-time systems. UPPAAL utilizes constraints and interactions between transitions and sub-processes to define timed automata. It provides a framework for model checking, verification, and simulation, enabling testing of the implemented model and learning of strategies using reinforcement learning.

6.2.1 UPPAAL Core

The classic version of UPPAAL is the most fundamental. It is based on timed automata. Figure 6.4 shows a simple example of a UPPAAL model. This model imagines a simple train line with 4 locations, the `depot` and three different `stops`, forming a loop. We consider the case with six trains operating on the same train line. A parameter is given to each train shown in the top of the Figure, so that each train has a unique `id`, which in this example is an integer in the range 1 to $N = 6$ inclusive. The type `id_t` is defined by `typedef int[1,N] id_t;`. Each of the locations has a name associated with it, and an invariant in purple (`x <= 2id`) which must hold true for a train to be in that location. x is a variable which is of type `clock`. Clocks are a type which automatically increases in value over time. Between locations there can be transitions. The transition between `depot` and `stop1` has a guard in green (`x >= id`) which must hold true for a train to travel from `depot` to `stop1`. When the transition is taken, the model can execute update code, which in this case is the two statements in blue. Firstly, `x=0` simply resets the clock x back to zero, and secondly the variable `depot_count` is decremented by one. In fact, `depot_count` gives at any point in time the number of trains which are at the depot location.

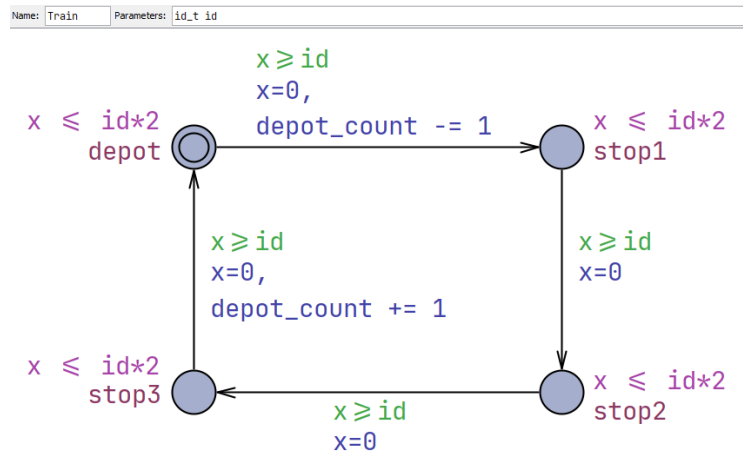


Figure 6.4: Simple UPPAAL train model example

UPPAAL supports automatic model checking of queries that specify relevant properties of a given system. Model checking of queries can be used to verify and validate the correctness of the model using a symbolic model checker. These queries express properties such as reachability, absence of deadlocks, and system properties like safety, liveness, and "leads to" conditions. For the train example we could ask if it is possible for the depot to have no trains with the query:

```
E<> depot_count == 0
```

which evaluates to true. Here $E<> P$ is a query that evaluates to true if eventually there is a reachable state which satisfies the predicate P [22]. The query:

```
E<> forall (id:id_t) Train(id).stop2
```

uses a forall statement to see if it is possible to reach a state where all trains are at **stop2** at the same time. Checking the query does evaluate to true, but depending on the type of state space search order used, it can take over a minute to check with a breadth first search or 0.2 seconds with a random depth first search.

6.2.2 UPPAAL SMC

An extension of the core UPPAAL tool is UPPAAL SMC (Statistical Model Checking), which enhances the verification capabilities of the original version. While the core UPPAAL relies on exhaustive state space exploration, UPPAAL SMC introduces probabilistic analysis, enabling systems to be modeled and analyzed statistically, including stochastic behaviors.

UPPAAL SMC supports the estimation of probabilities and the verification of model properties under uncertainty via an extended query language similar to that of the core version. It allows users to formulate probabilistic queries to estimate the likelihood of certain predicates being true and to simulate variable trajectories within the model. Figure 6.5 shows three examples of queries. The first query is a simulate query which, which simulates the model for an amount of time units specified and has a list of expressions which it keeps track of. Afterwards you can show a plot of the values of the expression. The result of the example simulate query can be seen in Figure 6.6. The second and third query in Figure 6.5 are probability queries, where you specify a bound on the time to simulate a run, an optional amount of runs and a predicate. The query:

$$\text{Pr}[\leq 20](\langle \rangle \text{depot_count} \geq 4 \ \&\& \ t \geq 5)$$

estimates that within 20 time units there is a $60\% \pm 5\%$ chance that there are at least 4 trains in the depot at some point after 5 time units. The other query:

$$\text{Pr}[\leq 500; 500](\langle \rangle \text{forall } (id:id_t) \text{ Train}(id).\text{stop2})$$

shows that over 500 runs that are simulated for 500 time units, there is a $16\% \pm 3\%$ chance that at some point all the trains are in the `stop2` location.

```
simulate[≤ 20] {depot_count}
Pr[≤ 20](⟨ ⟩ depot_count ≥ 4 && t ≥ 5)      0.606249 ± 0.0491512 (95% CI)
Pr[≤ 500; 500](⟨ ⟩ forall (id : id_t) Train(id).stop2) 0.166017 ± 0.0334058 (95% CI)
```

Figure 6.5: UPPAAL SMC queries for train example

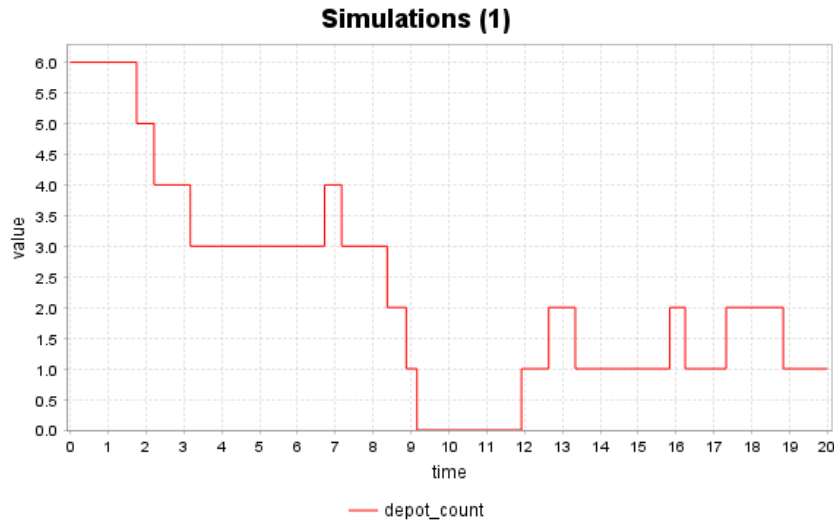


Figure 6.6: Result from running query `simulate[≤ 20] {depot_count}` on the train example

For WWTP, we need more than just time. We need concentration levels of various compounds—and descriptions in terms of differential equations as to how these evolve. So we need general continuous variables and not just clocks as in timed automata.

Fortunately, UPPAAL supports statistical model checking of Hybrid Automata. A key feature is the inclusion of hybrid clocks, in contrast to the core version that only supports static clocks. Hybrid clocks have variable rates, meaning they can update at a user-defined rate per time unit. This functionality is enabled by incorporating differential equations, where the clock rate is specified in the location invariants using Lagrangian derivative notation.

Example: Two Tank model

An example of a system that can be modeled as a Hybrid automaton is the two tank system as introduced in [23]—and it can be seen in Figure 6.7a. The system consists of two tanks, each with their own water level x_1 and x_2 , two controllable gates, Q_1 and Q_2 . The Q_1 adds inflow into the first tank if opened, and Q_2 drain the second tank if opened. There is also a constant inflow into the first tank Q_0 , and a drainage Q_A from tank 1 to 2 which depends on x_1 . Finally, there is a drainage Q_B from tank 2. The behavior of x_1 is given by:

$$\frac{dx_1}{dt} = \begin{cases} -x_1 - 2 & \text{if gate 1 is closed} \\ -x_1 + 3 & \text{if gate 1 is opened} \end{cases} \quad (6.1)$$

Likewise, the behavior of x_2 is given by:

$$\frac{dx_2}{dt} = \begin{cases} x_1 & \text{if gate 2 is closed} \\ x_1 - x_2 - 4 & \text{if gate 2 is opened} \end{cases} \quad (6.2)$$

The UPPAAL model for two tank model consists of two parts, the controller and tanks. The controller as seen in Figure 6.7b decides the action of which gates to open and close. Every P time units it picks a new action to do. The goal is to try and keep the water level in the tanks between 1 and 10. The guards are what limit which actions that are enabled. The tanks shown in Figure 6.7c shows three differential equations, one for each of the water levels in the tanks, and one for the cost function we want UPPAAL to optimize. The variables for the model are shown in Listing 6.1 and the functions for the differential equations shown in 6.2. $X1_change$ and $X2_change$ are simple implementations of equation (6.1) and (6.2).

The `calc_cost` function simply incurs a cost of 10, if the level of the tanks are not between 1 and 10.

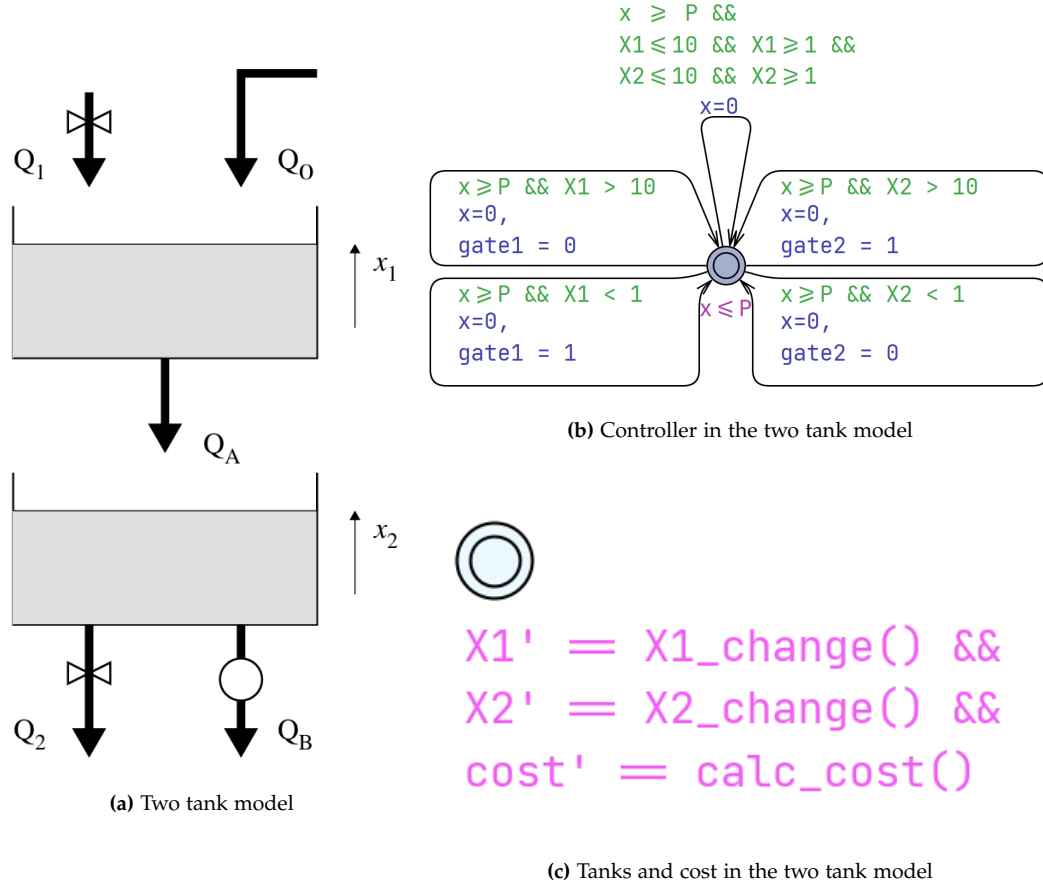


Figure 6.7: Overview of the two tank model

Now that we understand the model, let's try evaluating some queries on it. Figure 6.8 shows two queries. The first simulates for H time units and keeps track of the value of the variables in the brackets, results of which can be seen in Figure 6.9. The second query is an estimation query, that estimates the maximum value of the variable `cost` after H time units over 100 runs. It estimates that the maximum cost is ≈ 45 . The strategy does a good job at keeping the cost low. Maybe it can be improved, but it might be hard to see how, and this is where UPPAAL Stratego comes into play.

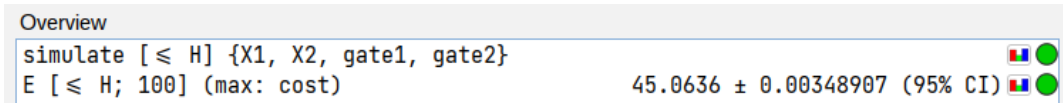


Figure 6.8: Simulation and Estimations queries for the two tank model

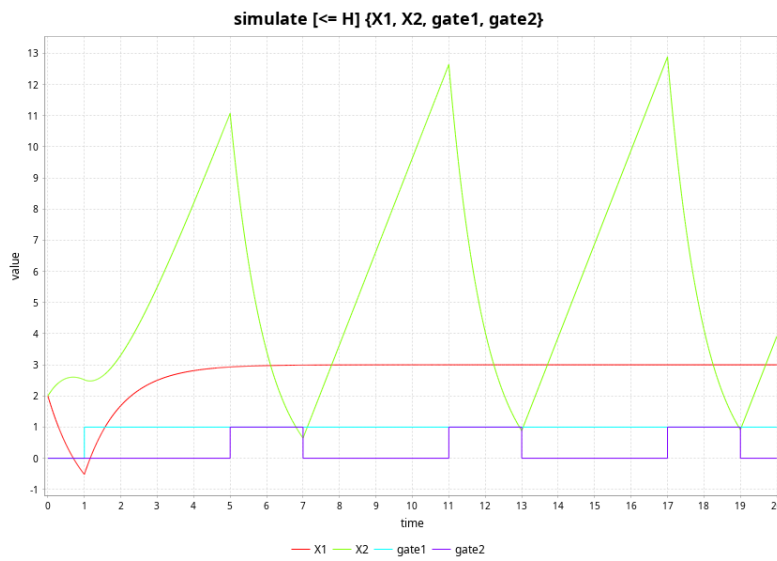


Figure 6.9: Plot of the result of query `simulate [≤ H] simulate [≤ H] {X1, X2, gate1, gate2}`

```

1 clock t;
2 clock x;
3 int gate1 = 0;
4 int gate2 = 0;
5
6 hybrid clock X1 = 2.0;
7 hybrid clock X2 = 2.0;
8 hybrid clock cost = 0.0;
9
10 const int P = 1;
11 const int H = P * 20;
12

```

Listing 6.1: two tank model variables

```

1 double X1_change(){
2     if(gate1){
3         return -X1 + 3.0;
4     }
5     else {
6         return -X1 - 2.0;
7     }
8 }
9
10 double X2_change(){
11     if(gate2){
12         return X1 - X2 - 4.0;
13     }
14     else {
15         return X1;
16     }
17 }
18
19 double calc_cost(){
20     if (X1 <= 1 || X1 >= 10 ||
21         X2 <= 1 || X2 >= 10){
22         return 10.0;
23     }
24     return 0.0;
25 }
26

```

Listing 6.2: two tank model functions

6.2.3 UPPAAL Stratego

UPPAAL Stratego includes features that enable the learning of strategies for models of systems, which can be defined as Hybrid Markov Decision Processes. There are two types of strategies:

1. **Safety strategies**, sometimes called a shield, which ensure that actions taken never lead to a property being violated whenever possible. This means the model avoids violating safety properties and prevents reaching undesirable states.
2. **Optimization strategies**, which focus on maximizing or minimizing a given optimization criterion.

UPPAAL Stratego employs reinforcement learning to derive a strategy that op-

timizes a specified objective. To learn an optimal strategy using reinforcement learning, the following query syntax is used:

```
strategy Name = minE(Cost) [Bound] { ExprList1 } -> { ExprList2 }:<> Goal
```

where:

- Name is an identifier used in other queries.
- minE(Cost) specifies that the expected value of the expression Cost should be minimized. maxE(Reward) is also available to maximize an expected reward instead.
- Goal defines a goal state, once a run ends up in a goal state, the run terminates.
- ExprList1 and ExprList2 are again lists of comma-separated expressions that limit what part of the system is used to learn a strategy. ExprList1 specifies the discrete state expressions and ExprList2 the continuous state expressions, thus having partial observability. These lists can be omitted to specify that the entire system should be observed.

Example: Two Tank model continued

Now, let's utilize UPPAAL Stratego to find a better strategy for the two tank model than the one used in section 6.2.2. To use UPPAAL Stratego we will need a new "controller", where a learned strategy can decide the actions to take. Controller is in quotes here because without a strategy to choose the actions it does not do much. This controller is shown in Figure 6.10. In UPPAAL Stratego a dashed line means that it is uncontrollable by a strategy and instead a part of the stochastic environment, whereas a solid line means it is controllable. The notation $i:\text{int}[0, 1]$ means that UPPAAL will create a transition for each value in the given range, so all permutations of open and closed of gate one and two will be made.

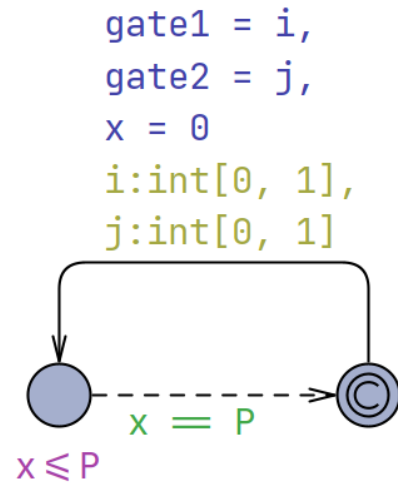


Figure 6.10: New two tank "controller"

With new control options made, we can try evaluating some queries on the model. Figure 6.11 shows the results of all the queries. Firstly in Figure 6.11a, we have two queries where no strategy is used, meaning that whenever there is a choice to make, it is picked randomly from a uniform distribution over the available options. The first simulates for H time units and keeps track of the value of the variables in the brackets, results of which can be seen in Figure 6.11b. The second query is an estimation query, that estimates the maximum value of the variable cost after H time units over 100 runs, yielding a maximum cost of ≈ 181 . The next query learns the strategy named *opt* by minimizing the estimated value of cost. It learns from runs that are H time units long. Once the strategy is trained, the same two queries are repeated—but this time, the choices are derived from the strategy *opt* instead of being random. As shown in Figure 6.11c, the behavior of the strategy is intentional, and the estimated maximum value from the second query is ≈ 18 using the strategy, which is lower than both the random strategy, and the manually made strategy from Figure 6.7b. The strategy appears to have learned an effective approach.

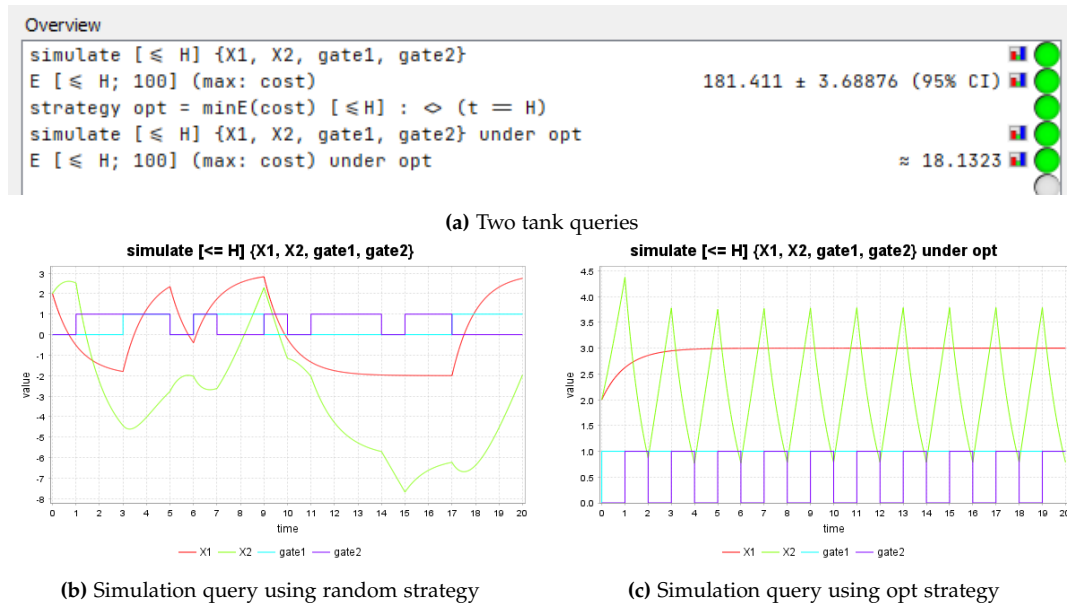


Figure 6.11: UPPAAL Stratego two tank results

6.3 STOMPC

Strategoutil is a python library that facilitates functionality to interface with UPPAAL Stratego via python. STOMPC is built into Strategoutil and allows to easily create MPC functionality with either UPPAAL as both the model and real system, or UPPAAL as the model and an external simulator as the real system [24].

STOMPC implements the class `MPCsetup` which has most of the functionality to run UPPAAL as a model and simulator. The only things you need to provide is to:

- The initial state
- The learning parameters for `verifyta` (UPPAAL command line utility)
- Implement the `create_query_file` method to create a fitting UPPAAL *.q file
- A UPPAAL model

To run MPC using an external simulator, a class which inherits `MPCsetup` is needed, and the method `run_external_simulator` needs to be implemented on

the class. This function takes the action found by the optimizer and needs to interface with the simulator—which in our case is the TORNADO API—and relays the action to it. The latest true state of the system is then returned from the simulator and then the method, so that for the next iteration the MPC loop, UPPAAL can use the new values. This is shown in Figure 6.12 and also shows where STOMPC fits in between UPPAAL Stratego and WEST. Figure 6.12 also shows that the optimizer can use extra external information for the optimization, in our case, inflow prediction data and the future energy prices.

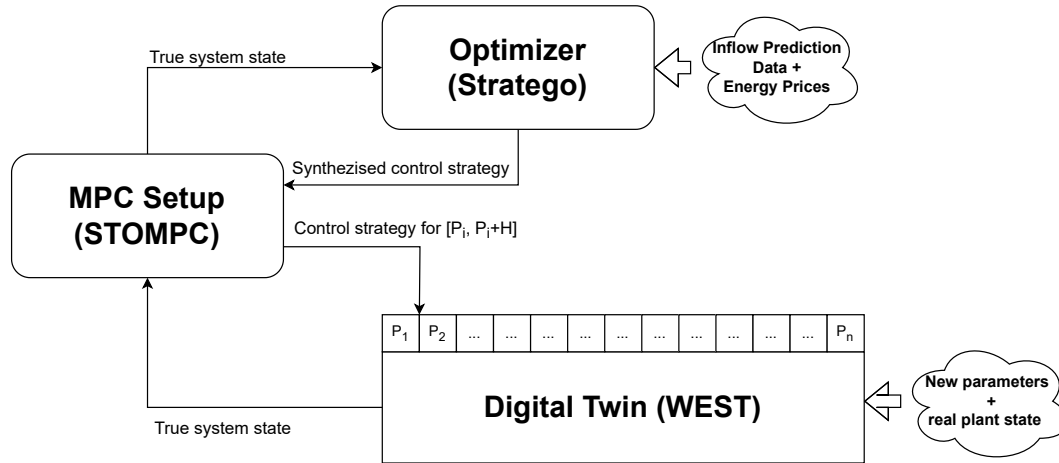


Figure 6.12: How STOMPC interfaces between UPPAAL Stratego and WEST

Some changes were needed to STOMPC for it to work like we needed which were:

- Windows support
- Allow array indexes to be watched

To allow STOMPC to run on Windows operating system, a change was needed to a regular expression used to parse values from a list outputted by a UPPAAL simulate query. The regular expression only matches on newline escape characters `\n`, but due to backwards compatibility, Windows also has a carriage return character `\r` which needs to be accounted for in the regular expression and is matched with an optional operator `\r?`.

For allowing STOMPC to support inserting and watching values in an array. This was not possible due to square brackets—used for indexing arrays—having a syntactical meaning and as such needs to be escaped first before being used in the regular expression.

Chapter 7

Implementation

This chapter first describes the pipeline mentioned in chapter 5. Next, the UPPAAL model of the BSM1 plant described in section 3.2 will be explained in detail. Lastly, the implementation of STOMPC and the code written for interfacing between WEST and UPPAAL is presented.

7.1 Pipeline

In this paper we propose a pipeline for intelligent control of the BSM1 plant using MPC and reinforcement learning, and a digital twin.

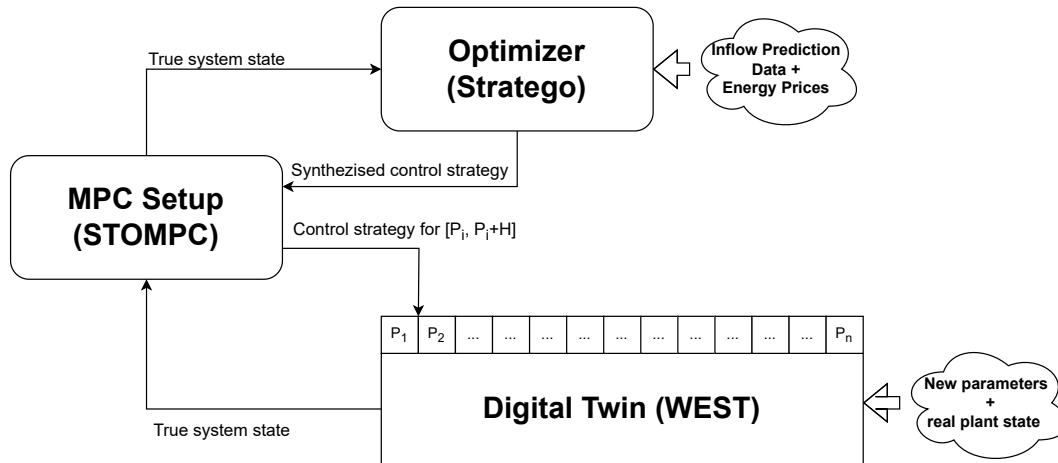


Figure 7.1: The MPC and digital twin pipeline.

- *Digital twin (WEST)* - This is the high fidelity virtual version of the actual plant, modeled in WEST. The state of the digital twin is updated to reflect the real plant. Because measurements of different compounds in a WWTP tank are manual, slow, and expensive, these measurements are very rarely conducted. This means that the state of the digital twin will be updated to the real state of the plant quite infrequently. However, with good parameter estimation, we predict simulation accuracy to be adequate for MPC.
- *Optimizer (UPPAAL)* - This consists of a low fidelity model of the plant in UPPAAL. UPPAAL also provides built in reinforcement learning techniques using UPPAAL Stratego. As such, the optimizer is used for computing near-optimal control strategies using reinforcement learning, which are then relayed back to STOMPC.
- *MPC (STOMPC)* - The tool STOMPC is used to implement the MPC scheme. It acts as an interface between the digital twin in WEST and the UPPAAL optimizer. It relays the plant state from the digital twin to the optimizer, and relays back the computed intelligent control strategy to be used.
- *Inflow prediction model* - This is a model that provides the optimizer with predictions on the characteristics of the incoming wastewater. This enables the optimizer to compute control strategies that not only are immediately effective, but also effective regarding the future state of the system.
- *Hourly electricity prices* - The optimizer is fed hourly energy prices to enable optimization of energy consumption timing. Using this, the algorithm could learn to ramp up aeration when energy is cheap, to preemptively build up

the biomass in the tanks, which is then useful later when energy prices are higher.

- *Periodic parameter estimation* - The accurate simulation of a real plant largely relies on the accuracy of the ASM1 model parameters. The model parameters can vary over time, depending on the weather, season and many other factors. As such, periodic parameter estimation is needed to ensure accurate simulation, in turn increasing the likelihood of computing effective control strategies.

Although we deem all the components in the list above to be necessary for intelligent control of a WWTP using MPC, only a subset were actually implemented due to the scope of this project. First, we have modeled the BSM1 plant as a high fidelity digital twin in WEST. Similarly, we have modeled a low fidelity BSM1 plant in UPPAAL to utilize UPPAAL Stratego and reinforcement learning in the MPC setup. We have implemented the MPC scheme using STOMPC, interfacing UPPAAL Stratego and WEST.

We have not built an inflow prediction model, but instead assumed that such a model exists in our setup. We have not implemented a parameter estimation model, but instead assume the ASM1 model parameters to be constant and correct. As such, the real plant has also been excluded as we essentially assume the digital twin to be absolutely accurate at all times.

7.2 Modeling

As a reference, below we insert the Figure from section 3.2 of the BSM1 plant.

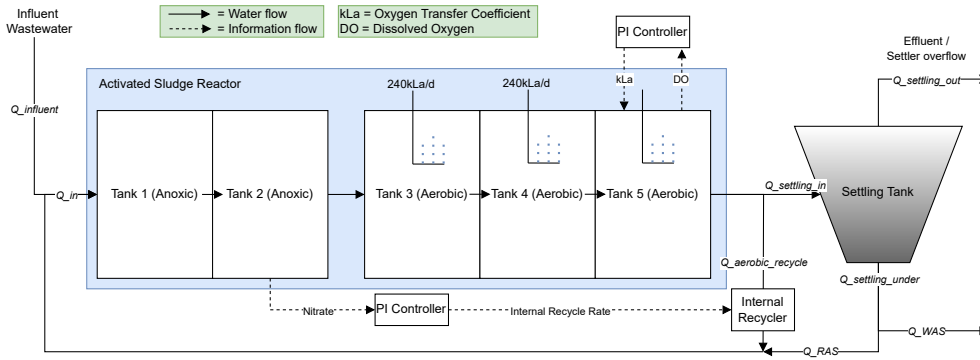


Figure 7.2: Layout for the default BSM1 plant

7.2.1 Update_Discrete

Below in Figure 7.3 is presented the Update_Discrete template in UPPAAL, which implements the settling tank model and the water flow logic.

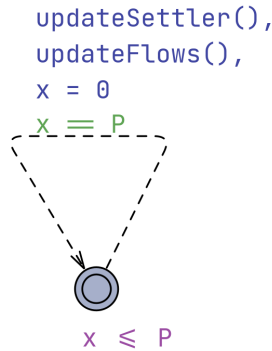


Figure 7.3: Modeling of the Waterflow

As shown in Figure 7.3, the settling tank and the water flow are modeled using a single location in the UPPAAL template. This location contains an invariant $x \leq P$, which ensures that the system remains in the state until clock x reaches P . The dashed transition with guard $x == P$ represents a cyclic update that when the clock x reaches P , the system performs a transition where the clock is reset ($x = 0$), and the update functions `updateSettler()` and `updateFlows()` are called. These functions calculate the water and sludge flows as well as update the component concentrations in the settler after one period P .

Waterflow

The following section has largely been carried over from the last report [3], as not much has changed in the modeling of the flows.

The flow is, as mentioned, updated using the `updateFlows` shown in listing 7.1. Q_{in} can be calculated as the sum of the incoming flows $Q_{influent} + Q_{RAS} + Q_{aerobic,recycle}$. From the last aerobic tank the amount recycled is limited by the recycler PI action variable u_r , and anything above u_r flows to the settling tank, giving us:

$$Q_{aerobic,recycle} := \min(Q_{in}, u_r)$$

and

$$Q_{settling,in} := Q_{in} - Q_{aerobic,recycle}$$

$Q_{settling,under}$ is the flow which exits under the settling tank. This is limited using $under_{max}$ where the rest of the flow goes to $Q_{settling,out}$, giving us:

$$Q_{settling,under} := \min(Q_{settling,in}, under_{max})$$

and

$$Q_{settling,out} := Q_{settling,in} - Q_{settling,under}$$

Lastly a bit of the flow is separated to be further processed as waste activated sludge (WAS). Again we have a limiting desired maximum WAS_{max} and the rest of the flow is the return activated sludge (RAS), which completes the cycle, giving us:

$$Q_{WAS} := \min(Q_{settling,under}, WAS_{max})$$

and

$$Q_{RAS} := Q_{settling,under} - Q_{WAS}$$

`updateFlows` is also where we calculate the RAS proportion:

$$RAS_{proportion} := \begin{cases} \frac{Q_{RAS}}{Q_{settling,under}} & \text{if } Q_{settling,under} \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

```

1  const double recycle_max = 38.42916; // m3/min
2  const double under_max = 13.077083; // m3/min
3  const double WAS_max = 0.26736111; // m3/min
4
5  void updateFlows() {
6      Q_influent = read_double(TID, row_index, H2O);
7      Q_in = Q_influent + Q_aerobic_recycle + Q_RAS;
8      Q_aerobic_recycle = fmin(Q_in, u_r);
9      Q_settling_in = Q_in - Q_aerobic_recycle;
10     Q_settling_under = fmin(Q_settling_in, under_max);
11     Q_settling_out = Q_settling_in - Q_settling_under;
12     Q_WAS = fmin(Q_settling_under, WAS_max);
13     Q_RAS = Q_settling_under - Q_WAS;
14     RAS_proportion = Q_settling_under != 0.0 ? Q_RAS/Q_settling_under
15     : 0.0;
16 }

```

Listing 7.1: Code snippet from the updateFlows function

Settling tank

The *updateSettler()* function from Figure 7.3 implements the separation of the solids and soluble compounds.

Afterward, dissolved and particulate components are distributed to either the effluent (the treated water) or the underflow (sludge) using two functions. For dissolved substances, the flow is applied directly, while for particulate substances, a fraction is used based on sludge concentration and flow to ensure realistic distribution.

For a more in depth description of the settling tank model, we refer to the previous report [3, p. 56-58], as the model used in the previous report has not undergone any changes.

7.2.2 Activated Sludge Tanks

The model represents a time-controlled process for updating five Activated Sludge Tanks (ASTs) in a wastewater treatment plant. The anoxic and aerobic tanks share the same construction and are therefore modeled together. The dynamics are implemented in C code and utilized in UPPAAL using the built-in external code functionality. The dynamics follow the ASM1 model as described in section 3.2.3.

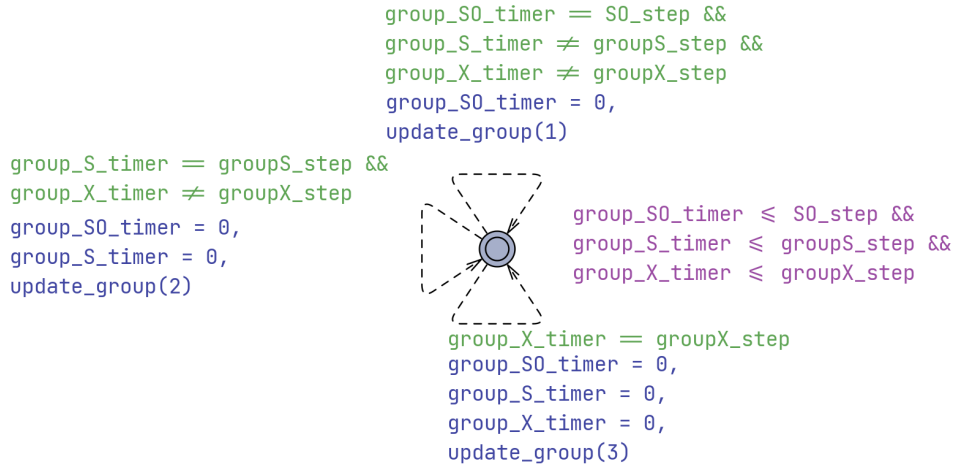


Figure 7.4: Modeling of discretely updating the compound concentrations in the AST's.

The model 7.4 introduces three distinct transitions, each corresponding to a specific category of compounds. The 13 compounds are grouped based on the numerical step-size each compound require for stable and accurate simulation [25]. Each group is updated independently according to its assigned step size.

These transitions are governed by the three clocks: *group_SO_timer*, *group_S_timer*, and *group_X_timer*. Each timer is associated with a compound group, and transitions occur only when the corresponding timing conditions are met. These conditions enforce controlled updates, preventing any group from being updated too frequently and thereby exceeding its allowed step size.

The state machine in the model uses these timers to determine which group to update, and the groups are defined with `update_group(x)` where *x* is a placeholder for group 1, 2 and 3. Group 1 consists of compounds {*S_O*, *S_{NO}*}, with a step size of 0.5 (minute). Group 2 consists of compounds {*S_I*, *S_S*, *S_{NH}*, *S_{ND}*, *S_{ALK}*}, with a step size of 1.0 (minute). Group 3 consists of compounds {*X_S*, *X_{BH}*, *X_I*, *X_{BA}*, *X_P*, *X_{ND}*}, with a step size of 10.0 (minutes). A group is updated only when its timer reaches the defined threshold. When this happens, the corresponding update function is called. If none of the thresholds are reached, the model remains in a waiting state.

The `update_group(int group)` function performs the internal computations for updating the five ASTs. It calculates the inflow into each unit based on the incoming influent, recycled activated sludge (RAS), and aeration contributions. Each AST receives inflow from the preceding tank, and oxygen (*S_O*) is transferred into

tank 3, tank 4 and tank 5.

It is important to note that while many of the other UPPAAL templates are similar to the last report [3], a lot of work has been dedicated to refine and optimize the code of the activated sludge tanks template.

In the last report, we experienced several issues with the UPPAAL integral solver Runge-Kutta. It was therefore decided to rewrite the code, to perform simple manual integration of the compound differential equations. First off, this solved the issue of reaching negative concentrations for rapidly evolving compounds like oxygen (S_O). Secondly, it allowed for costume fixed step-sizes for different compounds. For example, oxygen (S_O) and nitrate (S_{NO}) are fast evolving compounds, so to achieve sufficient accuracy and avoid negative concentrations for these two compounds, the step-size needs to be appropriately small. On the other hand, any X compound is slow evolving, so step-size of these compounds can be significantly larger than oxygen and nitrate. The grouping of different compounds to different step-sizes significantly reduced the amount of calculations when simulating, and resulted in a $\approx 9x$ simulation speed improvement, compared to the models presented in the last report.

In addition to this, when calculating the compound differential equations, many compounds dependent on the same processes in the ASM1 model. For example, to calculate how the oxygen concentration (S_O) evolves over time, one needs to compute the rate of process 1 (Aerobic Heterotrophic Growth). However, the compounds' substrate (S_S) and heterotrophic bacteria (X_{BA}), also need to compute the rate of process 1. In the old model, process 1 was thus computed 3 times, instead of just once. By optimizing the reuse of computed process results and eliminating redundant calculations, resulted in a factor two speed-up in simulation time.

7.2.3 Municipality Model

This template is designed to retrieve all inflow data from a CSV file, implementing the inflow to the BSM1 plant. Each row in the CSV file represents the inflow for one minute, with compound inflow values given in $\frac{g}{min}$ and water inflow in $\frac{m^3}{min}$. The inflow data is based on the BSM1 framework, which simulates weather conditions as described in section 3.2.7. The dataset spans a two-week period. To simulate steady-state inflow conditions, the model continuously reads the first row of the CSV file.

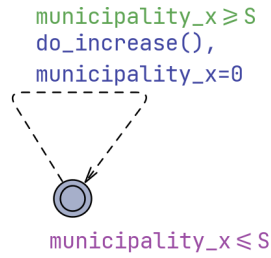


Figure 7.5: Modeling of Municipality

The model shown in Figure 7.5 consists of a single location. It features a local clock, *municipality_x*, with an invariant that triggers the transition function *do_increase()* every *S* time units. The *do_increase()* function advances the pointer to the next row in the CSV file, retrieving the corresponding inflow data for the next time step.

To emulate the existence of a prediction model as described in section 7.1, we apply stochastic uncertainty to the influent data, that approximates a realistic or worse error of such a prediction model. By doing so, UPPAAL Stratego will learn a control strategy that is effective in all different influent scenarios. This uncertainty is calculated in line 14 of listing 7.2, and is an accumulative error controlled by the two constants *alpha* and *std_error*. *alpha* controls how smoothly the error behaves, a higher *alpha* will result in slower error fluctuations, where a lower value will result in more random behavior. *std_error* controls the magnitude of uncertainty at each step.

```
1  const double alpha = 0.99;
2  const double std_error = 0.05;
3
4  double smooth_error() {
5      double ran_err = random_percent();
6      double prev_error = error;
7      return alpha * prev_error + std_error * ran_err;
8  }
9
10 void do_increase(){
11     if(ROWS != row_index + 1){
12         double rand = random_percent();
13         row_index += 1;
14         error = smooth_error();
15         for (i : int[0,num_compounds-1]) {
16             double value = read_double(TID, row_index, i);
17             influent_comp[i] = random_influent ? value*(1+error) :
value;
18         }
19     }
20 }
```

Listing 7.2: Code snippet from the Municipality template

To showcase how this looks when simulating, an example of the inflow of a random compound can be seen in Figure 7.6 below, shown in black, alongside the same compound inflow after uncertainty has been added in red.

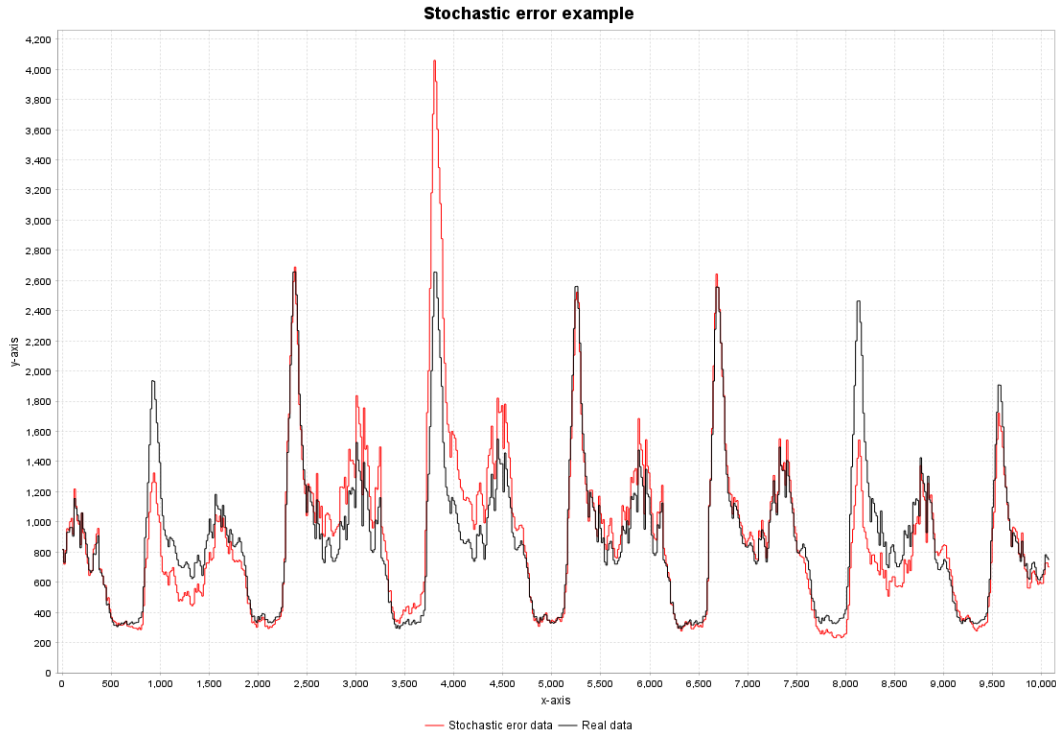


Figure 7.6: Compound inflow of some compound before and after uncertainty has been added.

When running several simulations with this accumulative error, we observe uncertainty of anywhere from 0-60%. In [26] they evaluate multiple different models for predicting the characteristics of influent. These model errors range, but are typically between 5-20%.

7.2.4 PI-Controller

The AerationPI template implements the PI-controller logic for the aeration of aerobic tanks in the BSM1 model, as well as reading current energy prices from a CSV file to calculate the cost of the aeration energy. The automaton regulates the oxygen levels in tank 5 using a Proportional-Integral (PI) controller. Furthermore, the model controls updating the energy cost data at regular time intervals. The implementation uses the PID controller theory in section 3.2.5 and the template modeled in UPPAAL can be seen in Figure 7.7.

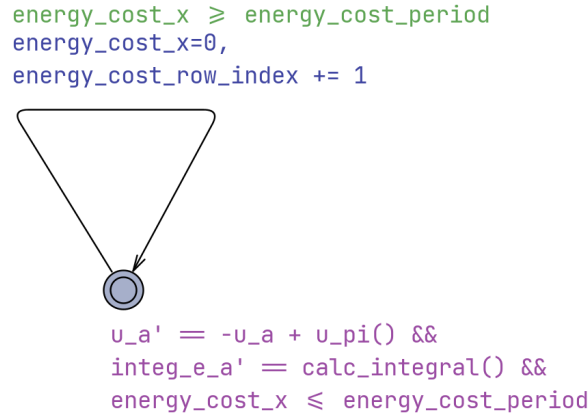


Figure 7.7: The AerationPI template

The model 7.7 consist of a single location with continuous behavior. Within this location, there are two differential equations defined in the invariants. The control signal u_a evolves according to the expression $u_a' = -u_a + u_{pi}()$, which gradually drives u_a toward the value computed by the PI-controller. $integ_e_a' = calc_integral()$ accumulates the error over time between the desired and actual oxygen concentrations. Lastly, $energy_cost_x \leq energy_cost_period$ ensures that the system remains in the current location until a predefined amount of time has passed. The clock $energy_cost_x$ keeps track of time and triggers an update when it reaches the value $energy_cost_period$.

There is another template that implements the PI-controller of the internal recirculating rate. Since its approach is the same as that of the AerationPI template, we do not show it here.

7.2.5 Controller

The last UPPAAL template in the model is the Controller template, which can be seen in Figure 7.8 below. The idea here is that this automaton primarily is in the **Wait** location, but periodically is forced to make a control decision, whenever *PC* time has passed. The template defines the available control options for the model, which consists of an array of oxygen concentration setpoints for the PI-controller controlling aeration in tank 5.

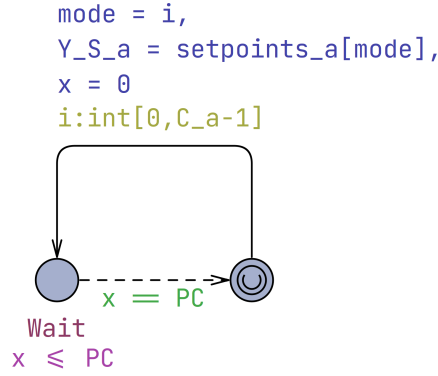


Figure 7.8: The Controller template

Note that the initial location is also an urgent location, meaning that no time can pass in this location. This ensures that a control decision is made immediately.

7.2.6 Cost function

In order for a optimization strategy to be learned a cost function is needed to evaluate the states explored in order to find the ones that minimize the cost. There are two factors that the cost function should take into account. The effluent quality (EQ) mentioned in section 3.2.6 and energy consumption from aeration $E_{aeration}$ mentioned in section 3.2.4. A weighting is added to the cost function such that one factor can be prioritized over the other. With two weights W_{EQ} and W_w such that $W_{EQ} + W_w = 1$, one can try many different weightings to find one that fits ones needs. A problem arises though because the domain of EQ and $E_{aeration}$ are different, meaning that a weight of $W_{EQ} = W_w = 0.5$ is not a equal priority for both factors. Therefore a normalization factor v_f is applied to $E_{aeration}$. This yields the differential equation shown in (7.1)

$$\frac{dcost}{dt} = W_{EQ}EQ + v_f W_w E_{aeration} \quad (7.1)$$

When energy is cheap it could be wise to use more of it for the same overall energy cost. Therefore the current energy price C_E can be multiplied to the energy usage, yielding Equation (7.2).

$$\frac{dcost}{dt} = W_{EQ}EQ + v_f W_w E_{aeration} C_E \quad (7.2)$$

Energy Prices

The energy cost data are obtained from the Energy Data Service [27]. The site provides Elspot prices for day-ahead spot prices in Denmark (DK) and neighboring countries. The spot price (in DKK) is calculated based on the spot price (in EUR) and the euro exchange rate from Danmarks Nationalbank. The data collected includes the hour and date for each entry in DK, along with the corresponding spot price (in DKK). While the spot prices on the website are given in DKK/MWh, they have been converted to DKK/kWh in the dataset. The dataset covers a period of two weeks starting from the 20th of January, 2025. The variation in Elspot prices over the dataset period is illustrated in Figure 7.9.

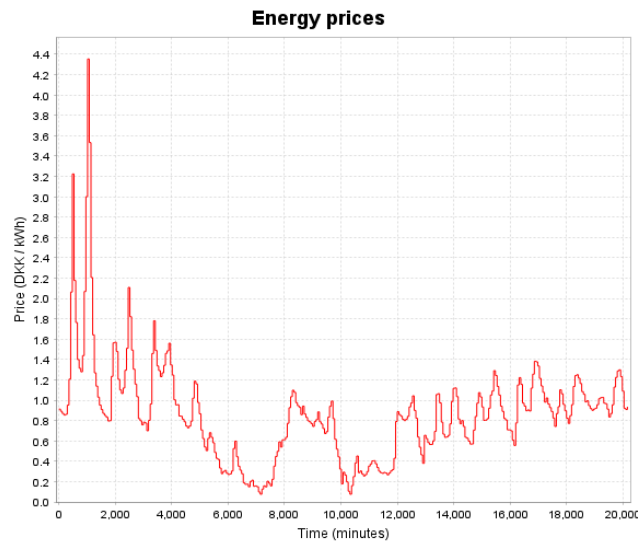


Figure 7.9: energy prices over a two week period starting from the 20th of January, 2025

7.2.7 Model Correctness and Speed

To validate that the UPPAAL model works correctly, some simulations were run on both it and the model in WEST with the same initial state. Variations for the step sizes for group-SO, group-S and group-X as described in section 7.2.2, were tested to find a good compromise between speed and accuracy. The result of the WEST simulation serve as the reference point for validation. The effluent of the UPPAAL simulations are then compared to that of the WEST simulation using a metric called Mean Absolute Percentage Error (MAPE). MAPE is sensitive to relative errors and the equation for calculating MAPE can be seen in equation (7.3) [28].

$$\text{MAPE}(y, \hat{y}) = \frac{1}{n} \cdot \sum_{i=1}^n \frac{|y_i - \hat{y}_i|}{\max(\epsilon, |y_i|)} \cdot 100 \quad (7.3)$$

The results in Figure 7.10 display five points, each labeled with a triple representing the step sizes for group-SO, group-S, and group-X, respectively. The step sizes (0.5, 1.0, 10.0) complete the simulation in less than half the time of the step sizes (0.1, 0.2, 2.0), with only a minor increase in the MAPE score. However, further increasing the step sizes significantly raises the MAPE score, as seen with (1.0, 2.0, 20.0), while providing only a marginal reduction in simulation time. Consequently, the red dot (0.5, 1.0, 10.0) was selected as the optimal step size configuration for further use.

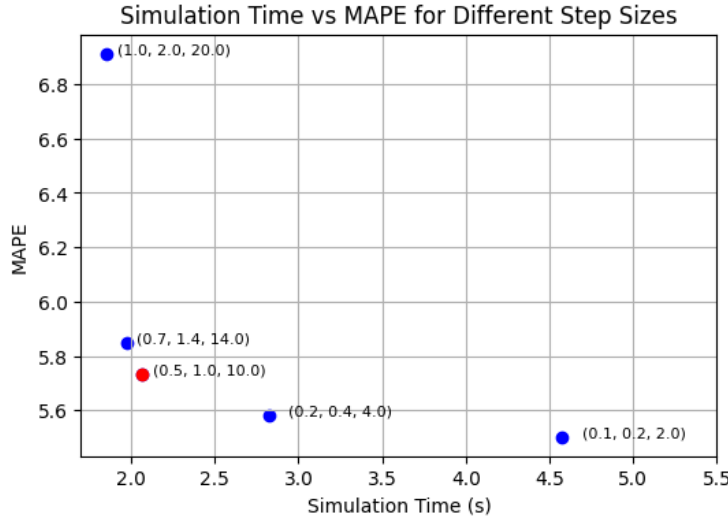


Figure 7.10: Trade off between simulation time and MAPE score of the UPPAAL BSM1 plant model compared to the BSM1 plant in WEST. Each point represents a variation in step sizes for group-SO, group-S, and group-X. The red dot indicates the selected step sizes for further use.

One important observation is that the MAPE scores for the compounds S_S and S_{NH} are more than twice those of the other model compounds, which contributes to an overall increase in the MAPE score. However, this is not a major concern, as when using MPC, the true state is periodically updated, preventing continued deviation.

7.3 MPC Implementation

For this project there are three main things we need to be able to do in order to run MPC with many different hyperparameters.

- Pass actions to and simulate the WEST BSM1 plant using the TORNADO API, returning a time series of data from the simulation.
- Run STOMPC to bridge between learning strategies in UPPAAL and applying them in TORNADO API.
- Conduct highly parallel MPC experiments with many different permutations of hyperparameters.

To achieve this, three Python classes were created, `TornadoExperiment()`, `BSM1_MPCsetup(MPCsetup)` (which inherits the base `MPCsetup()` class from `Strategoutil`) and `MPC_Experiment`.

7.3.1 TornadoExperiment class

This class manages all interactions with the TORNADO API. Upon initialization, it sets the necessary environment variables for the API to function correctly. This includes executing a file called "vcvars64.bat" which is a part of Microsoft Build Tools and defining paths to various folders required by the TORNADO API. The class is made such that you only need to call the run method after initialization. The signature of the method is:

```
def run(self, start, sim_time, controller_config)
```

Where the parameters are:

- `self`, Provides access to the `TornadoExperiment` class instance (Python's standard practice).
- `start`, Controls the time at which the simulation should start.
- `sim_time`, How long to simulate for.
- `controller_config`, A dictionary with set points for PI-controllers.

Before an experiment file from WEST can be executed using the TORNADO API, it must first be simulated in WEST. This simulation generates a file called "Dynamic.ObjEval.Exp.xml", which contains the initial state of the experiment.

To preserve this initial state, the file is copied so that simulations run through the `TornadoExperiment` class do not overwrite it, allowing it to be reused multiple times. This copying occurs only if the `start` parameter is set to zero. If the start time is nonzero, an experiment is already in progress, and the copied file is used instead.

After running the simulation via the TORNADO API, a time series containing a subset of the experiment's variables is returned.

7.3.2 BSM1_MPCsetup Class

As mentioned in section 6.3 the responsibility of `MPCsetup` class and those that inherit from it, is to facilitate interaction between UPPAAL and an external simulator, which is the TORNADO API for us. When the class is initialized an instance of the `TornadoExperiment` class is also initialized and stored in the property `self.experiment`. Also mentioned in section 6.3 is the need to implement the method `create_query_file`, which results a UPPAAL query file containing two queries, the first being derived from the f-string:

```
f"strategy opt = minE({cost}) [<={horizon}] {P0_var_string}: <>( t=={final})"
```

Which creates a strategy that learns to minimize the value of the cost function over runs that are H long. It is also incorporates partial observability as describe in section 6.2.3. The second query is given by the python f-string:

```
f"simulate [<= {period}+1] {{ {self.var_names_as_string()} }} under opt"
```

This query simulates the system for $P + 1$ time units. Since an action is applied every P time units, at $P + 1$, UPPAAL has just executed an action, allowing it to be captured in the simulation results and parsed. The call to `self.get_var_names_as_string()` ensures that all relevant variables are included in the simulation query.

The main loop for MPC is contained within the `MPCsetup.run` method, which consists of three major steps:

1. **Learning a strategy** using either the initial state or the state from a previous iteration, then extracting the best action from the result.

2. **Simulating the WEST model** using the extracted action via the `TornadoExperiment` class.
3. **Updating the state** based on the results of the TORNADO simulation so it can be used in the next iteration.

To enable STOMPC to run the external simulator, the following method is implemented:

```
def run_external_simulator(self, chosen_action, control_period, step)
```

where the parameters are:

- `self`: Provides access to the `BSM_MPCsetup` class instance (Python's standard practice).
- `chosen_action`: The action extracted from the strategy.
- `control_period`: The length of a period P .
- `step`: The number of MPC iterations completed.

The `TornadoExperiment.run` method, described in section 7.3.1, is used to perform the simulation via the TORNADO API. The simulation output is appended to a results file for future analysis, and the last row of the simulation data is returned from `run_external_simulator`.

7.3.3 MPC_Experiment Class

This class provides a method to generate multiple permutations of experiment hyperparameters and run MPC for each in parallel. Since each MPC experiment is independent, parallel execution is straightforward. The hyperparameters available and used in this project are shown in Table 7.1.

These hyperparameters are given as lists, allowing multiple options for a single hyperparameter. The Cartesian product is used to generate all possible experiment variable lists. A hash can be generated for each list of experiment variables. Any part of the code that generates or copies files uses the hash to create unique filenames.

| Hyperparameter | Default value | Description |
|--------------------------------|---------------------------|---|
| Period | 90 (90 minutes) | Length of the control period P |
| Horizon | 720 (12 hours) | Length of the horizon H |
| Actions (Aeration) | {0.0, 0.5, 1.5, 3.0, 5.0} | The action space available in RL. These are the PI-controller oxygen concentration setpoints. |
| Actions (Recycling) | {0.5, 1.0, 2.0, 3.0} | The action space available in RL, for the setup that includes recycling control. |
| Cost Function Weight | 0.5 | Rather to prioritize EQ or Energy cost in the cost function |
| Partially Observable Variables | Empty String | String containing the variables to observe |
| UPPAAL Model | "BSM1_STOMPC.xml" | List containing different names of UPPAAL models |
| good-runs | 200 | UPPAAL Stratego learning parameter |
| total-runs | 500 | UPPAAL Stratego learning parameter |
| runs-pr-state | 200 | UPPAAL Stratego learning parameter |
| eval-runs | 200 | UPPAAL Stratego learning parameter |

Table 7.1: Available experiment hyperparameters and their default value

The `MPC_Experiment` class implements `def run(self, exp_ID)`, where `self` follows Python's standard practice of referencing the instance, and `exp_ID` is an experiment hash. This method starts an experiment by instantiating a `BSM1_MPCsetup` class and calling its `run` method with all experiment hyperparameters associated with the experiment hash.

If an interruption occurs during an MPC experiment run, a built-in functionality checks whether a corresponding results file exists. If the time column of the last row in the time series does not match the total expected experiment duration, the experiment can resume from the most recent iteration. When the `MPC_Experiment.run` method is called, it first performs this results file check to determine whether the experiment should restart from the beginning or resume from its last completed iteration.

Interruptions may arise from various sources, such as a code exception, an UP-PAAL error, a TORNADO API failure, or other unforeseen issues, such as Windows updating.

Chapter 8

Experiments

Various experiments were conducted to evaluate the effectiveness of using MPC to control the BSM1 plant. Section 8.1 outlines the different BSM1 plant variations and the hyperparameters applied in the experiments. In sections 8.2.1, 8.2.2, and 8.2.3, the focus is on energy consumption, where the cost function balances energy use and effluent quality. In section 8.2.4, the focus shifts to energy cost, with the cost function balancing energy cost and effluent quality. Energy cost is computed using the hourly energy data presented in section 7.2.6.

8.1 Setup

The experiments include five variations of the BSM1 plant setups: one using a fixed control strategy, while the remaining four employ MPC and reinforcement learning to synthesize near-optimal control strategies. These strategies adjust one or more PI-controller setpoints for aeration, internal recycling, or both. The five BSM1 plant setups are:

- **Fixed control** - In this setup, a PI-controllers are added to control aeration in tanks 3, 4, and 5. Each PI-controller follows the same constant oxygen concentration setpoint throughout the experiment. This setup serves as a baseline for comparing traditional control methods to intelligent control using MPC.
- **1 tank AC** - This setup follows the BSM1 base control configuration described in section 3.2.5. However, instead of maintaining a fixed aeration setpoint of

2.0 in tank 5, the PI-controller's setpoint is dynamically adjusted using MPC and UPPAAL Stratego.

- **3 tank AC** - This setup builds upon 1 *tank AC* by additionally assigning PI-controllers for aeration control to tanks 3 and 4. However, all three aeration PI-controllers share a single oxygen concentration setpoint, which continues to be adjusted dynamically by MPC and UPPAAL Stratego.
- **3 tank AC and RC** - This setup builds upon 3 *tank AC* by additionally manipulating the setpoint of the PI-controller that regulates internal recycling flow. Like before, MPC and UPPAAL Stratego are used to dynamically adjust the setpoints.
- **Individual and combined AC** - This setup explores the concept of closely interconnecting denitrification and nitrification by allowing MPC to assign a shared setpoint to tanks 3 and 5, while tank 4 receives a separate setpoint. In theory, tanks 3 and 5 would maintain higher oxygen concentration levels, fostering aerobic environments, while tank 4 would operate at a lower setpoint, creating an anoxic environment. The rapid switching between aerobic and anoxic conditions is expected to enhance the efficiency of the denitrification and nitrification processes.

The following describes the experiment setup for the energy consumption experiments in sections 8.2.1, 8.2.2, and 8.2.3. Each setup presented above is run for one week under each weather scenario (dry, rain, and storm), starting from the second week in the file. The second-week start is necessary because the rain and storm weather does not start until the start of the second week. To compare the performance of the different setups, each setup will also be run with four different weighting pairs of energy consumption and effluent quality (EQ). The set of weights for energy consumption are {0.2, 0.4, 0.6, 0.8}, and vice versa for weights on effluent quality. These variations generate four Pareto curves, visually illustrating the trade-off between energy consumption and effluent quality, aiding in performance comparison across setups. Additionally, a Pareto curve for traditional fixed control of the BSM1 plant is included to benchmark MPC performance.

Hyperparameter Finding

Using MPC and UPPAAL Stratego, there are several hyperparameters that can be manipulated to fine-tune learning and optimize performance. The selected hyperparameters and their values are presented in Table 8.1 below. Note that this is

almost the same table as in section 7.3.2, but is presented here to clearly define the experimental hyperparameters.

| Hyperparameter | Default value | Description |
|--------------------------------|---|--|
| P | 90 (90 minutes) | Length of the control period P |
| H Actions (Aeration) | 720 (12 hours) {0.0, 0.5, 1.5, 3.0, 5.0} | Length of the horizon H The action space available in RL. These are the PI-controller oxygen concentration setpoints. |
| Actions (Recycling) | {0.5, 1.0, 2.0, 3.0} | The action space available in RL, for the setup that includes recycling control. |
| Partially Observable Variables | All (Full observability) | Variables in the model that Stratego can observe during learning |
| good-runs | 200 | Number of good runs for each learning iteration |
| total-runs | 500 | Number of total runs to attempt for learning |
| runs-pr-state | 200 | Number of good runs stored per discrete state |
| eval-runs | 200 | Number of runs used for evaluation of strategies |
| alpha | 0.99 | Controls the smoothness of influent uncertainty error—higher values slow fluctuations, lower values increase randomness. |
| std_error | 0.05 | Determines the magnitude of the influent uncertainty at each step. |

Table 8.1: Available MPC and RL hyperparameters and their default value chosen for the experiments.

The four hyperparameters—good-runs, total-runs, runs-pr-state, and eval-runs—constitute what is referred to as the reinforcement learning budget. A larger budget allocates more computational resources to discovering an effective control strategy, consequently increasing the time required to synthesize a strategy for the given period.

As noted in section 7.2.3, the specific combination of `alpha` and `std_error` from the table introduces significant uncertainty in the influent, reaching up to 60% error.

To determine the above hyperparameter values, we conducted coarse initial tuning, focusing on one hyperparameter at a time and testing different values to observe their impact on learning over a short duration. From running MPC with differing amount of observable variables we found that full observability performed the best. Which is surprising considering that less variables should in principle allow the partitioning algorithm—that UPPAAL Stratego uses—to find more optimal strategies. Having less variables means having a smaller state space to explore and therefore be more likely to find good partitions.

8.2 Results

This section presents the experiment results, following the approach outlined in the previous section, for each weather scenario.

8.2.1 Dry Scenario

The results of the dry weather experiment are shown in Figure 8.1. In general, all four setups that utilize MPC for intelligent control outperform the fixed control approach, represented by the black curve. Additionally, there appears to be a consistent trend where higher setpoints in the fixed control setup correlate with greater potential improvements in effluent quality and energy consumption.

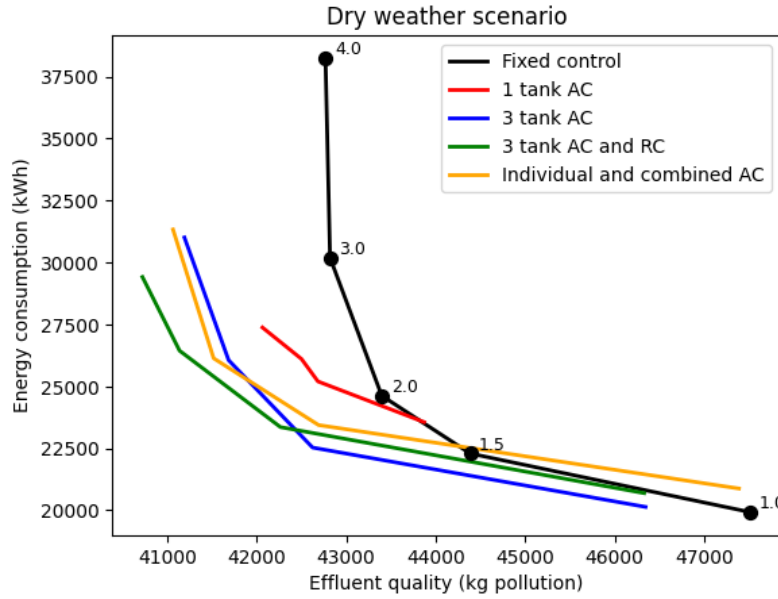


Figure 8.1: The accumulated energy consumption and effluent quality over a week of the dry weather file for the four different setups. They are plotted against the fixed control results in black. Each fixed control data point is labeled with the given fixed oxygen concentration PI-controller setpoint. The results are presented as Pareto curves, balancing between a focus on energy consumption or a focus on effluent quality (EQ).

The *1 tank AC* setup is limited to manipulate the aeration of the last tank in the BSM1 setup, which might explain the relatively small Pareto curve. There is not a lot of room for influencing the energy use and thus the effluent quality of the BSM1 plant in this setup. It still shows an improvement to fixed control in higher aeration setpoints, while the performance drops off and, if following the trend, actually drops below the performance of fixed control in the lower aeration setpoints.

The *3 tank AC and RC* looks to be the best performing setup, for higher aeration setpoints, utilizing the additional control of the internal recycling rate in the BSM1 plant. Comparing it to fixed control with a setpoint of 3.0, it achieves an energy consumption reduction of 24% while keeping the same effluent quality. Additionally, it achieves a 5% improvement in effluent quality, at the same energy consumption as fixed control with setpoint 3.0. At a fixed control setpoint of 2.0, which is a very common oxygen concentration setpoint in the industry, the results are an 8% relative energy consumption reduction and a 4% relative effluent quality improvement.

| Fixed setpoint | Violation time (min) |
|----------------|----------------------|
| 1.0 | 6200 (12.3%) |
| 1.5 | 4253 (8.4%) |
| 2.0 | 3776 (7.5%) |
| 3.0 | 4070 (8.1%) |
| 5.0 | 5338 (10.6%) |

(a) Effluent limit violation time at different fixed control setpoints.

| EQ weight | Violation time (min) |
|-----------|----------------------|
| 0.2 | 5955 (11.8%) |
| 0.4 | 3428 (6.8%) |
| 0.6 | 2608 (5.2%) |
| 0.8 | 2853 (5.7%) |

(b) Effluent limit violation time at different EQ weights for the 3 tank AC and RC setup.

Table 8.2: Comparison of effluent limit violation times between fixed control and the 3 tank AC and RC setup during the dry weather scenario. The violation time is the total violation time summed over all different pollutant groups for which there exists an effluent limit. Note that because this is a summation over all pollutant groups, some violation times may exceed the number of minutes in a week (10080). The percentage value, refers to the percentage of time the limits were violated out of the collective maximum violation time for all five pollutant groups. The collective maximum violation time is $10080 \cdot 5$, since each pollutant group could potentially violate the limits for the entire duration of the week. The effluent limits of the different pollutant groups can be seen in section 3.2.6.

In Table 8.2 above, it can be seen that there is a general relationship between the prioritization weight on effluent quality and the total effluent limit violation time. Although it is difficult to make a direct comparison between a fixed control setpoint and an EQ weighting used in the reinforcement learning cost function, the two tables give an intuition on the benefits of intelligent control using MPC on violation times. Note that the effluent violation limits are not included as a component in the cost function used in reinforcement learning. Instead, the numbers are a result of an improved effluent quality when applying intelligent control using MPC and reinforcement learning.

For both MPC control and fixed control, a point is reached where more oxygen input negatively impacts the violation time. This makes sense, as too much oxygen will overproduce nitrate and Kjeldahl nitrogen, which are then in danger of exceeding their effluent limits.

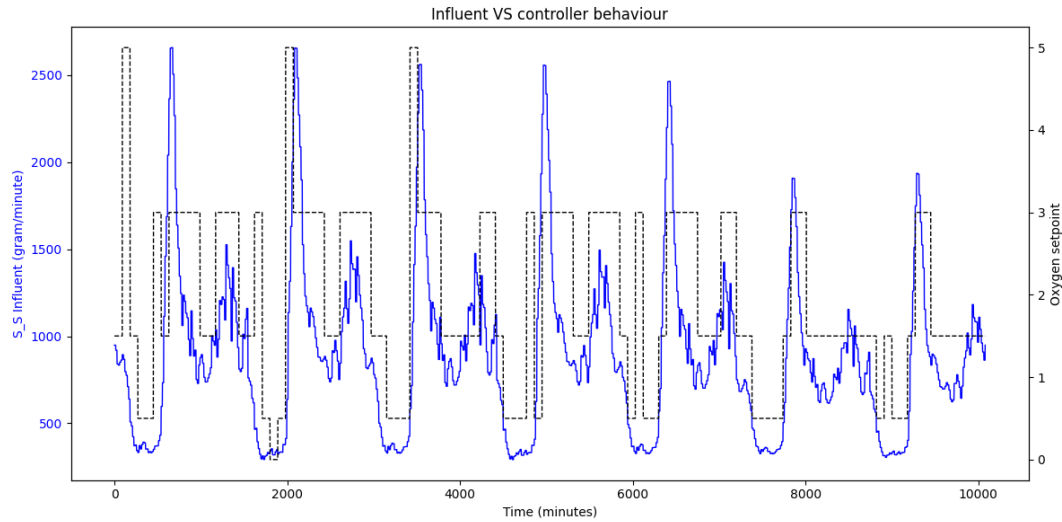


Figure 8.2: A graph showcasing how the 3 *tank* AC setup with an energy consumption weight of 0.4, reacts to changes in influent. Here, the S_S compound is used, but the influent trend is the same for all compounds, so S_S sufficiently represents the influent trends.

In Figure 8.2 above, it can be seen how the 3 *tank* AC setup reacts to changes in influent. In general there is a clear pattern in the influent, where influent spikes during the morning when people wake up, varies throughout the day, and then falls to a minimum during the night. Interestingly, it can be seen how the oxygen concentration setpoint for the PI-controllers of the 3 tanks roughly follows the peaks and valleys of the influent. This makes sense, as the more influent you have coming in, the more oxygen you need to treat it. This also means that aeration can be significantly reduced during the night, where influent is reduced. The 3 *tank* AC setup showcased above is with a weight of 0.4 on energy consumption. This means that it is slightly leaning towards prioritizing effluent quality. This could explain why on some days, the oxygen setpoint does not spike to 5.0 when the influent spikes.

8.2.2 Rain Scenario

The results of the rainy weather experiment are shown in Figure 8.3. Much like the results of the dry weather scenario, MPC is generally more effective when allowed to use more energy. However, in the rain scenario, performance for all setups are worse than fixed control, for aeration setpoints below 3.0.

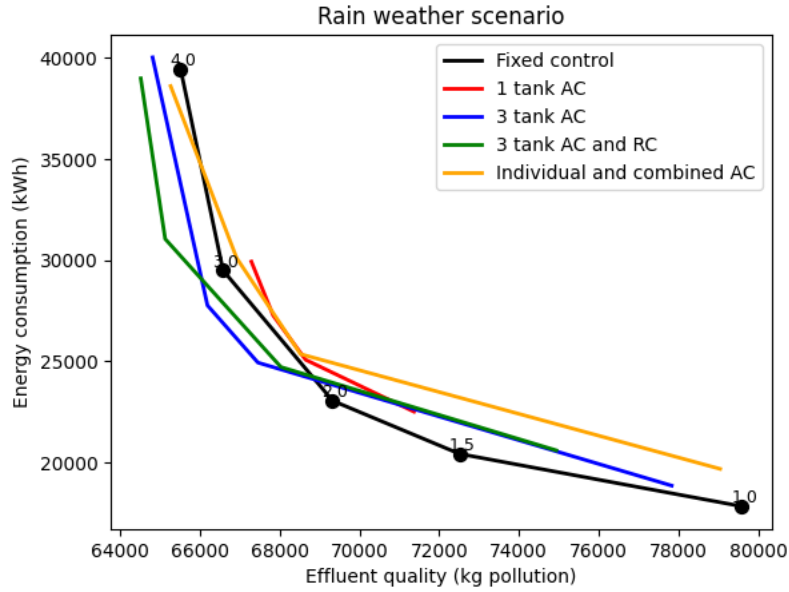


Figure 8.3: The accumulated energy consumption and effluent quality over a week of the rain weather file for the four different setups. They are plotted against the fixed control results in black. Each fixed control data point is labeled with the given fixed oxygen concentration PI-controller setpoint. The results are presented as Pareto curves, balancing between a focus on energy consumption or a focus on effluent quality (EQ).

Despite the worsened performance at lower setpoints in the rain scenario, the *3 tank AC and RC* setup still manages to improve energy consumption by 6% at the same effluent quality as fixed control with setpoint 3.0. It also achieves a 2% improvement in effluent quality at the same energy consumption. Especially during rain periods, when the retention time of the compounds fall because of the increase in flow rate, it is important to recycle more, so the water can be properly cleaned. This could be a reason that the added intelligent control of the internal recycler improves overall performance during the rain scenario.

The *Individual and combined AC* setup performs worse across almost all setpoints during the rain scenario. It is possible that the additional switching between denitrification and nitrification is doing more harm than good to the effluent quality of the plant.

| Fixed setpoint | Violation time (min) |
|----------------|----------------------|
| 1.0 | 11941 (23.7%) |
| 1.5 | 10058 (20.0%) |
| 2.0 | 9341 (18.5%) |
| 3.0 | 8759 (17.4%) |
| 5.0 | 8862 (17.6%) |

(a) Effluent limit violation time at different fixed control setpoints.

| EQ weight | Violation time (min) |
|-----------|----------------------|
| 0.2 | 11335 (22.5%) |
| 0.4 | 9253 (18.4%) |
| 0.6 | 7956 (15.8%) |
| 0.8 | 8198 (16.3%) |

(b) Effluent limit violation time at different EQ weights for the 3 tank AC and RC setup.

Table 8.3: Comparison of effluent limit violation times between fixed control and the 3 tank AC and RC setup during the rain weather scenario. The violation time is the total violation time summed over all different pollutant groups for which there exists an effluent limit. Note that because this is a summation, some violation times may exceed the number of minutes in a week (10080). The percentage value, refers to the percentage of time the limits were violated out of the collective maximum violation time for all five pollutant groups. The collective maximum violation time is $10080 \cdot 5$, since each pollutant group could potentially violate the limits for the entire duration of the week. The effluent limits of the different pollutant groups can be seen in section 3.2.6.

In the two tables in Table 8.3, the total effluent violation time can be seen for fixed and MPC control of the rain scenario. Evidently, there is significantly more pollutant in the incoming wastewater, compared to the dry scenario. Despite this, the same conclusion about the oxygen input and violation time can be made as for the dry weather scenario in the previous section. Additionally, intelligent control using MPC again generally improved on the total effluent violation time.

8.2.3 Storm Scenario

The following graph in Figure 8.4 presents the results of the storm scenario experiment. Generally, it closely resembles the results of the rain scenario from the previous section, although slightly improved performance across the four setups.

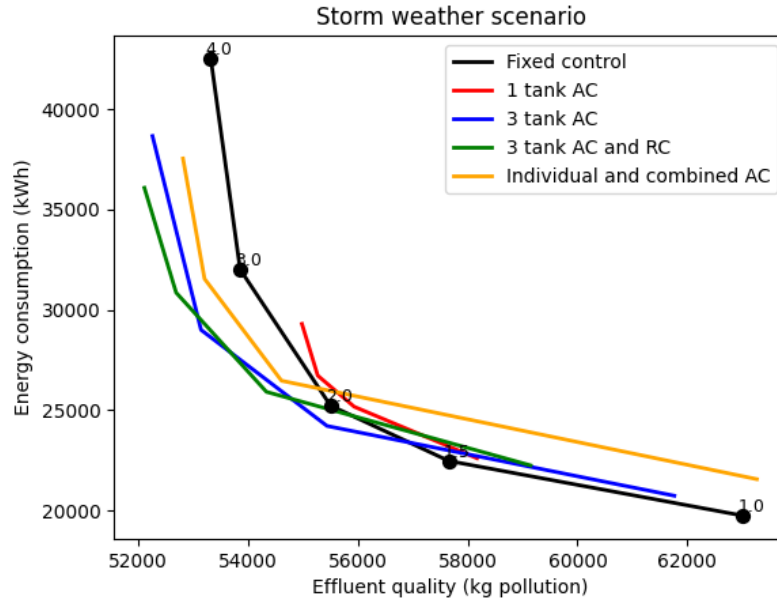


Figure 8.4: The accumulated energy consumption and effluent quality over a week of the storm weather file for the four different setups. They are plotted against the fixed control results in black. Each fixed control data point is labeled with the given fixed oxygen concentration PI-controller setpoint. The results are presented as Pareto curves, balancing between a focus on energy consumption or a focus on effluent quality (EQ).

Looking at the *3 tank AC and RC* setup, it reduces energy consumption by 15% compared to fixed control using a setpoint of 3.0, at the same effluent quality. At the same time, it improves effluent quality by 3% at the same energy consumption.

| Fixed setpoint | Violation time (min) |
|----------------|----------------------|
| 1.0 | 8915 (17.7%) |
| 1.5 | 7143 (14.2%) |
| 2.0 | 6139 (12.2%) |
| 3.0 | 6203 (12.3%) |
| 5.0 | 6439 (12.5%) |

(a) Effluent limit violation time at different fixed control setpoints.

| EQ weight | Violation time (min) |
|-----------|----------------------|
| 0.2 | 7919 (15.7%) |
| 0.4 | 5823 (11.5%) |
| 0.6 | 5155 (10.2%) |
| 0.8 | 5062 (10.0%) |

(b) Effluent limit violation time at different EQ weights for the 3 tank AC and RC setup.

Table 8.4: Comparison of effluent limit violation times between fixed control and the 3 tank AC and RC setup during the storm weather scenario. The violation time is the total violation time summed over all different pollutant groups for which there exists an effluent limit. Note that because this is a summation over all pollutant groups, some violation times may exceed the number of minutes in a week (10080). The percentage value, refers to the percentage of time the limits were violated out of the collective maximum violation time for all five pollutant groups. The collective maximum violation time is $10080 \cdot 5$, since each pollutant group could potentially violate the limits for the entire duration of the week. The effluent limits of the different pollutant groups can be seen in section 3.2.6.

As shown in Table 8.4, we observe similar overall improvements in effluent limit violation time. Here, the improvements are also better than the rain scenario.

8.2.4 Optimizing energy cost

In the following experiment, the cost function has been adapted to minimize the specific timely cost of energy (DKK) instead of the amount of energy (kWh). Additionally, the influent uncertainty has been regulated, such that *std_error* is now 0.02 instead of 0.05, as it was in the previous experiments. This creates error of up to 20%, instead of 60%, which is significantly more realistic. Lastly, the 1 tank AC setup has been excluded, and instead the other setups have included a bigger variety of energy cost/effluent quality weights. Other than these factors, the visualization of the results are still exactly the same as for the energy consumption experiments, and they can be seen in Figure 8.5 below.

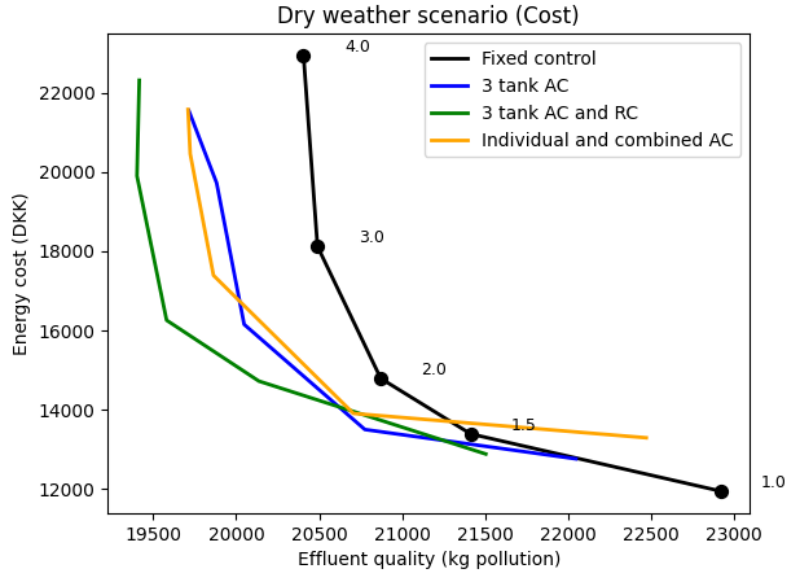


Figure 8.5: The accumulated energy cost and effluent quality over three days of the dry weather file for three different setups. They are plotted against the fixed control results in black. Each fixed control data point is labeled with the given fixed oxygen concentration PI-controller setpoint. The results are presented as Pareto curves, balancing between a focus on energy cost or a focus on effluent quality (EQ).

In the experiment we see an improvement in the performance of the *3 tank AC and RC* setup at lower setpoints, when comparing to the first energy consumption experiment. This might be explained by the lower error used in this experiment. However, the trend is still the same overall, where there are bigger performance improvements at higher energy usage. Specifically, the *3 tank AC and RC* setup reduces energy cost by 21% at the same effluent quality as fixed control with a setpoint of 3.0. In the same manner, it improves effluent quality by 5%. At a fixed control setpoint of 2.0, the energy cost reduction is 8% and effluent quality improvement is 4%.

Chapter 9

Discussion

Inflow uncertainty

As mentioned in section 7.1, we assume to be in possession of a model that can predict the characteristics of the wastewater coming into a WWTP. We replicate the error of such a model, by applying an accumulative error to the influent, as presented in section 7.2.3. In the experiments, *alpha* and *std_error* are 0.99 and 0.05 respectively, resulting in up to 60% error or uncertainty on the influent during reinforcement learning. This error is much too large, considering the normal range of 5-20% error for such influent characteristics prediction models in the literature. This could explain some of the poor results for lower setpoints in the experiments. Due to time constraints, the power consumption focused experiments could not be re-run with a lower, more appropriate influent error.

With that said, in the last experiment with cost, when the error was reduced significantly, only small improvements at lower oxygen setpoints were seen. Another possible explanation could be that the aeration action space does not enough possible setpoints. Setpoints 2.0 and 1.0 are not part of the action space, so possibly a more fine-grained action space could help performance at lower setpoints.

Lastly, there could simply be a natural inclination to there being less room for improvement when prioritizing energy. When looking at figure 8.2 in the first experiment, we see that MPC is learning to increase aeration during the day, and decrease it during the night to save energy. So during the day, it is using more aeration to build up the biomass (bacteria) in the tanks, so that during the night,

it can lower aeration, thereby slightly decreasing the biomass, which over time will simply maintain or slightly increase the biomass. Biomass is very important for treating the wastewater. If the amount of biomass is not sufficient, the treatment becomes much less effective. So, what could be happening for low aeration setpoints, is that because there is more focus on energy, there is only room for maintaining the biomass during the day, where at night, the amount of biomass falls or is simply maintained, because there is not enough incoming waste. This means that the MPC setup can no longer utilize lowering aeration at night, because this would do too much harm to the biomass and in turn the effluent quality. All in all, this significantly shrinks the potential for improvement using intelligent control when prioritizing energy more, which might explain what we are seeing in the experiments.

Effluent quality improvements

In the experiment results, we observe effluent quality improvements when comparing against fixed control of the BSM1 plant. These improvements are around 2%-5% improvement in the total amount of pollutants accumulated in the effluent, depending on the setup and weather scenario. It is not intuitive whether such improvements are meaningful. It may depend on the scenario of a given WWTP. If one considers a WWTP that struggle to meet effluent pollution limits, these smaller improvements may help clear the limits, or at least reduce the time in which they are violated. As mentioned in section 3.2.6, in Denmark you are taxed on all pollutants in the WWTP effluent. So depending on the size of the taxations, improving effluent quality by a few percent, may outweigh the price you have to pay for the extra aeration energy.

Effluent pollutant limits

In the experiment results we included tables for comparison of effluent pollutant limit violations between fixed control and intelligent control using MPC. In section 3.2.6 it was mentioned that these pollutant limits are not treated as strict safety properties in the BSM1 framework, but this can change from country to country. Safety property or not, the effluent limits are an important control performance evaluation criteria, and as such should ideally have been included in the reinforcement learning cost function alongside energy and effluent quality. Because the WWTP dynamical system model is quite complex and the pipeline presented in 7.1 has several components, it was omitted to keep the experiments relatively sim-

ple. The plan was to add it at a later point, but in the end it was not implemented due to time constraints.

In addition to pollutant limits, the BSM1 framework provides several other metrics for measuring the performance of a control strategy. These are things like sludge production to be exposed and the total sludge production. However, these were not prioritized for the same reasons as explained above.

Hyperparameter fine-tuning

As mentioned in section 8.1 we have only performed coarse tuning of the hyperparameters presented in the section. Instead of thoroughly experimenting with and tuning each hyperparameter, the experiments focused on different ways of controlling the BSM1 plant, using the four different setups presented in the section. This decision was discussed with DHI, who found experimenting with different setups more interesting. As such, the MPC and reinforcement learning performance in the experiments may be suboptimal, leaving a desire for further experimentation and tuning.

Chapter 10

Future Work

The following will present suggestions to future work in relation to this project.

Automatic Conversion

To improve the efficiency of optimization and experimentation with WWTPs, future work could focus on automating the conversion of WEST models into modular UPPAAL models. This would be beneficial not only by reducing the time spent manually designing equivalent models, but also by supporting the practical implementation of such systems. This would mean that many different variations of a plant layout could easily be made and be experimented on with the methods utilized in this report.

Shielding

As shown in the experiments, the effluent limits provided by the BSM1 framework are violated, even for the best performing setup.

A potential solution to this limitation is the use of shields. Shields can be applied both during the learning phase and during deployment. A shield is a non-deterministic strategy that, given a current state, returns all actions that ensure adherence to safety properties. Within these safety-preserving constraints, the controller can optimize efficiency and performance. This approach enables the

learning of a strategy that is as close to optimal as possible, while still guaranteeing safety.

For example, in the context of a WWTP, a shielded controller would aim to minimize energy usage without ever exceeding the permissible effluent concentration limits. It is uncertain whether it is always possible to remain below the limits.

Hyperparameter Fine-tuning

The hyperparameters chosen for the experiments may not represent the most optimal configuration. Conducting more rigorous testing and fine-tuning these parameters could significantly enhance both the efficiency and overall performance of the BSM1 plant. By systematically evaluating a broader range of hyperparameter settings, it may be possible to identify configurations that improve stability, reduce operational costs, and optimize resource utilization. A deeper exploration of these variables could ultimately lead to a more effective and reliable system.

Use Effluent Pollutant Tax

In Denmark, there is a tax on different effluent groups based on the amount of kilograms released. For example, in 2025, total nitrogen N_{tot} is taxed at 33.39DKK/kg [14]. Based on this, the cost function could be adjusted from considering effluent quality and energy cost to focusing on pollutant tax and energy cost. This approach eliminates the need to balance two distinct objectives, as both factors are expressed in monetary terms and can be directly integrated into a unified cost function. As a result, reductions in effluent quality would only occur when financially advantageous, which could lead to an unintended consequence—if economic incentives do not strongly favor pollutant reduction, overall effluent quality may deteriorate, potentially harming environmental and public health.

Chapter 11

Conclusion

This report presents the modeling of the BSM1 plant and the synthesis of control strategies to enhance effluent quality and optimize energy consumption. This was achieved using an MPC setup, where the BSM1 plant modeled in UPPAAL was utilized to learn control strategies, while the BSM1 plant modeled in DHI's WEST software was used as the real plant to apply these strategies and obtain the true state of the system.

The experiments involved five different setups of the BSM1 plant. One setup employed fixed control without MPC—used as a baseline for comparison—while the remaining four all integrated MPC, each controlling different components within the BSM1 plant. The experiments were conducted under three distinct weather scenarios provided by the BSM1 framework [4].

The experiments demonstrated that the *3 tank AC and RC* setup outperformed the baseline *fixed control* at higher aeration setpoints. In the dry weather scenario, it achieved a 24% reduction in energy consumption while maintaining the same effluent quality compared to a fixed setpoint of 3.0. When the cost function weighting prioritized effluent quality, the *3 tank AC and RC* setup was able to deliver a 5% improvement in effluent quality while using the same amount of energy as the *fixed control*. Performance across all setups declined in other weather scenarios, but the *3 tank AC and RC* configuration still outperformed the *fixed control* when comparing high aeration setpoints. At lower fixed setpoints the performance of MPC was often comparative to fixed control or worse, across the different weather scenarios.

Another way to assess the improvement in effluent quality of the *3 tank AC*

and RC setup is by examining the duration of effluent limit violations. The *3 tank AC and RC* setup demonstrates lower violation time compared to the *fixed control*, especially when the weighting prioritizes effluent quality.

When switching the cost function to focus on energy cost instead of energy consumption, for the *3 tank AC and RC* setup, an energy cost improvement of 21% was observed at the same effluent quality as fixed control with a setpoint of 3.0.

Can MPC and reinforcement learning be used to reduce energy consumption/cost and improve effluent quality of the BSM1 plant?

When looking at the problem statement above, we have shown through the experiments that MPC and reinforcement learning *can* be used to improve both energy consumption/cost and effluent quality of the BSM1 plant. Additionally, we have proposed a digital twin MPC+RL pipeline in section 7.1 that address the practical implementation challenges presented in chapter 5.

Bibliography

- [1] International Energy Agency (IEA). *World Energy Outlook 2016*. IEA, Paris, 2016. Licence: CC BY 4.0. 1
- [2] United Nations. The 17 sustainable development goals. <https://sdgs.un.org/goals>, n.d. [Online; accessed: 2025-01-10]. 1
- [3] Jørgensen MPH, Gehlert LB, Koch CB, and Gebauer H. Enhancing effluent quality and energy efficiency of the modified ludzack-ettinger process: A model predictive control approach. https://kjdk-aub.primo.exlibrisgroup.com/permalink/45KBDK_AUB/a7me0f/alma9921967369105762, 2025. 1, 4, 6, 7, 11, 12, 20, 28, 31, 34, 53, 54, 56
- [4] Jens Alex, Lorenzo Benedetti, Jb Copp, Krist Gernaey, Ulf Jeppsson, Ingmar Nopens, Marie-Noelle Pons, Leiv Rieger, Christian Rosen, and J-P Steyer. Benchmark simulation model no. 1 (bsm1). *Report by the IWA Taskgroup on Benchmarking of Control Strategies for WWTPs*, 01 2008. 2, 9, 11, 18, 19, 21, 22, 86, 91, 92
- [5] DHI. Mike powered by dhi | water modelling software. <https://www.dhigroup.com/technologies/mikepoweredbydhi>, 3 2024. [Online; accessed 2024-12-29]. 4
- [6] wpadmin. How to remove suspended solids from water? – etch2o. <https://www.etch2o.com/how-to-remove-suspended-solids-from-water/>, 2 2023. [Online; accessed 2024-12-22]. 7
- [7] Archis Ambulkar and Jerry A. Nathanson. Wastewater treatment - pollutants, contamination, purification | britannica. <https://www.britannica.com/technology/wastewater-treatment/Sources-of-water-pollution>, 1 2010. [Online; accessed 2024-12-22]. 7

- [8] Avijit Mallik and Md. Arman Arefin. Clean water: Design of an efficient and feasible water treatment plant for rural south-bengal. *Journal of Mechanical Engineering Research and Developments*, 41:156–167, 04 2018. 7
- [9] Metcalf & Eddy Inc., George Tchobanoglous, H. Stensel, Ryujiro Tsuchihashi, and Franklin Burton. *Wastewater Engineering: Treatment and Resource Recovery*. McGraw-Hill Education, New York, 5th edition, 2013. 8
- [10] George Crawford and Julian Sandino. *Energy Efficiency in Wastewater Treatment in North America: A Compendium of Best Practices and Case Studies of Novel Approaches*. Water Environment Research Foundation (WERF) and IWA Publishing, London, United Kingdom, Alexandria, USA, 2010. 9
- [11] Mogens Henze, Willi Gujer, Takashi Mino, and Mark Loosdrecht. *Activated Sludge Models ASM1, ASM2, ASM2D, ASM3*, volume 5. 01 2000. 12
- [12] Ulf Jeppsson. Modelling aspects of wastewater treatment processes. 11 2012. 13, 14, 16
- [13] Robert Paz. The design of the pid controller. 01 2001. 18
- [14] Skatteministeriet. Spildevandsafgiftsloven. <https://skm.dk/tal-og-metode/satser/satser-og-beloebsgraenser-i-lovgivningen/spildevandsafgiftsloven>. [Online; accessed 2025-06-04]. 21, 85
- [15] James B. Rawlings, David Q. Mayne, and Moritz M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, Madison, WI, 2nd edition, 2017. 27
- [16] M. Francisco, Pastora I. Vega, and Silvana Revollar. Model predictive control of bsm1 benchmark of wastewater treatment process: A tuning procedure, 2011. 30
- [17] Martijn A. Goorden, Kim G. Larsen, Jesper E. Nielsen, Thomas D. Nielsen, Weizhu Qian, Michael R. Rasmussen, Jiří Srba, and Guohan Zhao. Optimal control strategies for stormwater detention ponds. *Nonlinear Analysis: Hybrid Systems*, 53:101504, 2024. 31, 32
- [18] Gerd Behrmann, Agnès Cougnard, Alexandre David, Emmanuel Fleury, Kim G. Larsen, and Didier Lime. Uppaal-tiga: Time for playing games! In Werner Damm and Holger Hermanns, editors, *Computer Aided Verification*, pages 121–125, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. 31
- [19] Peter Bulychev, Alexandre David, Kim Guldstrand Larsen, Marius Mikućionis, Danny Poulsen, Axel Legay, and Zheng Wang. Uppaal-smc: Statistical model checking for priced timed automata. *EPTCS*, 85, 07 2012. 31

- [20] Alexandre David, Peter Gjør Jensen, Kim Guldstrand Larsen, Marius Mikućionis, and Jakob Haahr Taankvist. Uppaal stratego. In Christel Baier and Cesare Tinelli, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 206–211, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. 31
- [21] Imran Riaz Hasrat, Peter Gjør Jensen, Kim Guldstrand Larsen, and Jiří Srba. End-to-end heat-pump control using continuous time stochastic modelling and uppaal stratego. In Yamine Aït-Ameur and Florin Crăciun, editors, *Theoretical Aspects of Software Engineering*, pages 363–380, Cham, 2022. Springer International Publishing. 31, 32
- [22] Gerd Behrmann, Alexandre David, and Kim G. Larsen. A tutorial on uppaal. <https://homes.cs.aau.dk/~adavid/RTSS05/UPPAAL-tutorial.pdf>, 25/10-05. 39
- [23] Ian Hiskens. Stability of limit cycles in hybrid systems. page 6 pp., 02 2001. 41
- [24] Martijn A. Goorden, Peter G. Jensen, Kim G. Larsen, Mihhail Samusev, Jiří Srba, and Guohan Zhao. Stompc: Stochastic model-predictive control with uppaal stratego. In Ahmed Bouajjani, Lukáš Holík, and Zhilin Wu, editors, *Automated Technology for Verification and Analysis*, pages 327–333, Cham, 2022. Springer International Publishing. 47
- [25] Mogens Henze, Leslie Grady Jr, W Gujer, G. Marais, and T Matsuo. Activated sludge model no 1. *Wat Sci Technol*, 29, 01 1987. 55
- [26] Markéta Andreides, Petr Dolejš, and Jan Bartáček. The prediction of wwtp influent characteristics: Good practices and challenges. *Journal of Water Process Engineering*, 49:103009, 2022. 59
- [27] EnergyDataService. Elspot prices. <https://energidaservice.dk/tso-electricity/Elspotprices>. [Online; accessed 2025-06-05]. 62
- [28] scikit learn.org. 3.4. metrics and scoring: quantifying the quality of predictions — scikit-learn 1.6.0 documentation. https://scikit-learn.org/stable/modules/model_evaluation.html#mean-absolute-percentage-error. [Online; accessed 2024-12-29]. 62

Appendix A

Control Handels

This appendix includes the table from [4] mentioned in section 3.2.4. The names of the controls have been changed to fit Figure 3.3.

| Control handle | Minimum value | Maximum value | Comments |
|--|---------------|---------------|--|
| $Q_{aerobic_recycle}$ ($m^3.d^{-1}$) | 0 | 92230 | |
| Q_{RAS} ($m^3.d^{-1}$) | 0 | 36892 | |
| Q_{WAS} ($m^3.d^{-1}$) | 0 | 1844.6 | |
| $kLa1$ (d^{-1}) | 0 | 360 | Tank 1 |
| $kLa2$ (d^{-1}) | 0 | 360 | Tank 2 |
| $kLa3$ (d^{-1}) | 0 | 360 | Tank 3 |
| $kLa4$ (d^{-1}) | 0 | 360 | Tank 4 |
| $kLa5$ (d^{-1}) | 0 | 360 | Tank 5 |
| q_{EC1} ($m^3.d^{-1}$) | 0 | 5 | Tank 1 Carbon source conc. 400,000 g COD. m^{-3} in case of low biomass in the tank |
| q_{EC2} ($m^3.d^{-1}$) | 0 | 5 | Tank 2 Otherwise same as above |
| q_{EC3} ($m^3.d^{-1}$) | 0 | 5 | Tank 3 Otherwise same as above |
| q_{EC4} ($m^3.d^{-1}$) | 0 | 5 | Tank 4 Otherwise same as above |
| q_{EC5} ($m^3.d^{-1}$) | 0 | 5 | Tank 5 Otherwise same as above |
| $f_{Q_influent1},$ $f_{Q_influent2},$ $f_{Q_influent3},$ $f_{Q_influent4},$ $f_{Q_influent5}$ | 0 | 1 | Part of the influent flow rate distributed to each biological reactor Note: the sum of all five must always equal one |
| $f_{Q_aerobic_recycle1},$ $f_{Q_aerobic_recycle2},$ $f_{Q_aerobic_recycle3},$ $f_{Q_aerobic_recycle4},$ $f_{Q_aerobic_recycle5}$ | 0 | 1 | Part of the internal recirculation flow rate distributed to each biological reactor Note: the sum of all five must always equal one |
| $f_{Q_RAS1}, f_{Q_RAS2},$ $f_{Q_RAS3}, f_{Q_RAS4},$ f_{Q_RAS5} | 0 | 1 | Part of the sludge return flow rate distributed to each biological reactor Note: the sum of all five must always equal one |

Table A.1: Available control handles and their limitations [4].