Predicting Phosphorus Concentrations in WWTPs Using Data-Driven Methods

Master Thesis

June 2021

Laura Debel Hansen

Department of Energy Technology
Aalborg University Esbjerg

Department of Energy Technology
Aalborg University
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Predicting Phosphorus Concentrations in WWTPs Using Data-Driven Methods

Theme:
Master Thesis

Project Period:
Spring semester 2021

Project Group:
OES10-3-F21

Participant:
Laura Debel Hansen

Supervisors:
Petar Durdevic Løhndorf

Copies: 1

Page Numbers: 69

Date of Completion:
May 28, 2021

Preface

The project was prepared in the spring of 2021 from February 2021 to May 2021 and processes the overall subject: data-driven system identification.

The project is made as a continuation of a project conducted on the student's 3'rd semester, where she was a voluntary trainee at the wastewater department at Krüger A/S. The project is based on the results of the 3'rd semester project and knowledge previously achieved when studying Master of Science in Sustainable Energy Engineering.

Aalborg University, May 28, 2021

Instructions for reading

The report is written in LaTeX, and each chapter is marked with a certain number, and is divided into sections. All the references used throughout the report are indicated by the method referred to as the Institute of Electrical and Electronics Engineers (IEEE). The bibliography is made in Mendeley and BibTeX, and the citations used throughout the sections are noted in the text either at the beginning of a section, or as each individual statement is made. Citations of figures and tables are mentioned in the caption.

Abstract

Modelling wastewater treatment processes is the key to improve and optimize treatment performance, and the task has been the topic of various research for decades. However, the problem remains a major challenge in both academia and industry as the wastewater processes are highly nonlinear, coupled and time-varying dynamic systems containing both physical and biochemical reactions and large time delay features. As a result, the use of data-driven system identification has increased, introducing the artificial neural networks as predictive models for the processes.

This study proposes several data-driven identification methods to predict the phosphorus concentration at a case plant. The wastewater treatment plant (WWTP) of interest is located in Agtrup, Denmark, and the plant uses a combination of chemical precipitation and biological phosphorus removal.

In this study, both linear and nonlinear data-driven methods are investigated to obtain the best model for phosphorus concentration in wastewater. Dynamic mode decomposition with control is applied to obtain a linear model, however, the model shows poor generalizability, and is assessed inadequate to predict the inherently nonlinear process.

To accurately model the nonlinearities in the system, two neural network structures are proposed; a NARX neural network and a long short-term memory network. Bayesian optimization is applied to optimize the model structure, and results shows that a LSTM structure with Bayesian optimized hyperparameters has the best prediction performance. The obtained models are compared based on several statistical measures, including temporal evaluations, ensuring that the model dynamics reflects the dynamics of the actual system.

The best model is concluded to the a LSTM with 25 inputs, 2 hidden LSTM layers with 93 units in each and a output layer with a single unit. When validated on new data, the best model shows strong performance estimating the phosphorus concentration with a low MSE of 0.0848 and $R^2 = 0.42$.

Resumé

System identifikation af renseprocesser for spildevand er nøglen til at øge effektiviteten og forbedre behandlingsprocesserne, og har derfor også været fokus i meget forskning igennem de sidste årtier. Alligevel er det stadig en stor udfordring i både universitetsverdenen og industrien, da renseprocesserne udviser meget ulineær, tidsafhængig og koblet dynamisk adfærd. Derfor er brugen af data-drevne metoder for system identifikation blevet mere og mere populære, og har bl.a. introduceret det kunstige neurale netværk som en prædiktiv model for flere af spildevandsanlæggets processer.

I dette studie præsenteres flere data-drevne metoder til prædektion af fosforkoncentrationen på et anlæg beliggende i Agtrup, nær Kolding. Dette anlæg fjerner fosfor ved brug af både kemisk fosforfjernelse or biologisk fosforfjernelse. Både lineære og ulineære data-drevne metoder er præsenteret og sammenlignet i dette studie for at identificere metoden der resulterer i den bedste prædiktion af fosfor. Dynamic mode decomposition with control er anvendt til at etablere en lineær dynamisk model, men er dog erklæret uegnet til opgaven, da modellen mangler intern hukommelse og inkorporering af en tidsfaktor. Derfor femstilles to ulineære strukture i form a kunstige neurale netværk. Et neuralt netværk i form af en NARX og den velkendte LSTM trænes på datasættet og deres performance sammenlignes. Ydermere udvikles en algoritme til automatisk hyperparameter tuning, og denne algoritme anvendes på LSTM strukturen, hvilket resulterer i at give den bedste performance for prædektion af fosfor. De præsenterede modeller sammenlignes baseret på flere statistiske data, som bl.a. også inkluderer lighed over tid, og dermed reflekterer dynamikken af modellen i forhold til systemet. Når den bedst fundne model evalueres på et hidtil uset datasæt resulterer det i en lav MSE på 0,0848 og en $R^2 = 0,42$.

Contents

1 Introduction

1.1 Motivation

Dynamic models are usually developed to predict or estimate dynamical systems or to obtain better process understanding. The models can be used for e.g. fault detection, model based control design and predictive control for the system of interest. The model performance, i.e. how accurately it estimates or predicts the actual system is of fundamental importance, as the quality of the model typically sets an upper bound for the quality of the final solution, i.e. control development [1].

As the challenge to meet more stringent effluent standards arises, mathematical models have been frequently employed to wastewater treatment processes to understand, control, monitor and optimize the processes [2]. The fundamental objectives for any WWTP are [2]:

1. To maintain the effluent quality requirements.
2. To maintain the controlled variables at the desired outputs despite the changing influent loads and the outward disturbances.
3. To minimize the energy consumption, operational cost and environmental footprint during the processes.

WWTP's main purpose is to limit the eutrophication effects of surface waters and remove phosphate (PO_4^{3-}), ammonium (NO_4^{+}) and nitrate (NH_3^{-}) from the wastewater before it is discharged. As these nutrients are the main contributors to eutrophication effects, they are of key importance in the control process of the WWTP [2]. In recent years, the research area of resource recovery from wastewater processes has rapidly emerged due to the increased focus on nutrients (especially phosphorus) as a limited resource [3], [4]. As a result, engineering practice is facing a transition from the conventional WWTPs to wastewater resource recovery facilities (WRRFs). In places where phosphorus is in short supply in the soil, farmers add phosphate-based fertilizers to increase agricultural yields. As there exists no substitute for phosphorus in the agriculture, recycling the nutrient is essential to make the sector sustainable [3].

Commonly, phosphorus is removed from the wastewater using chemical precipitation, where metal salts are added to the wastewater causing the phosphate to be incorporated into suspended solids (SS) [5]. Chemical dosing is generally reliable and widely accepted, but also expensive and it entails increased sludge production [2], [5].

Due to the high operational costs and the produced chemical sludge using chemical phosphorus removal, biological phosphorus removal (BPR) is gaining more attention. Notoriously, BPR is one of the more difficult processes to control in WWTPs, which has been experienced at several WWTPs where the process suddenly can release huge amounts of phosphorus in the effluent [6].

Modeling of WWTPs has been ongoing for decades, but the problem remains a major challenge in both academia and industry as the wastewater processes are highly nonlinear, coupled and time-varying dynamic systems containing both physical and biochemical reactions and large time delay features.

Traditionally, WWTPs have been modelled using mechanistic models also known as deterministic models or white-box models [7]. These models has been used to increase plant performance, contribute to process understanding and evaluate design alternatives. However, a major drawback of mechanistic models is the complexity of the models, demand for prior knowledge and model calibration before utilization [7]. In order to describe coupled biological and chemical processes in the system, the deterministic models often involve a very large number of state variables.

As a result of the inconveniences connected to mechanistic models, data-driven models based on input-output data are gaining acceptability and are being increasingly applied in WWT systems [8]–[10]. As model parameters are determined from experimental modeling, these models are especially suitable for system identification when a lot of sensorial information is available, where white-box models are not valid or when the data needed to calibrate white-box models is unavailable. Nevertheless, building dynamic nonlinear models from data is a challenging task as it requires merging of two expertise communities: control engineers and researchers and machine learning people and statisticians [1].

A benefit of the data-driven models is that large amounts of sensorial data is already collected to monitor the WWTPs, making it cheap to implement and develop as no changes are required for the system of interest. The data-driven models can simply be developed to extract the important information from the currently logged data, and predict the desired outputs from the chosen input variables.

1.2 Phosphorus Removal in Wastewater

Phosphorus (P) is present in wastewater in the form of phosphate (PO_4^{3-}), and it can be removed from the wastewater by incorporating it into chemical precipitates or biological solids and then subsequently removed from those solids [11]. The fundamentals of both chemical and biological phosphorus removal is described in the following sections.

1.2.1 Chemical Precipitation

In chemical phosphorus precipitation, lime or metal salts such as calcium, [Ca(II)], aluminum [Al(III)] and iron [Fe(III)] is added to the process to incorporate phosphorus into

the chemical precipitates. The precipitation process for ferric iron and alum is almost identical, and they can be explained collectively as Me^{+++} in the following simplified precipitation model [11], [12]:

$$\text{Primary reaction: } Me^{3+} + H_2PO_4^- \rightarrow MePO_4 + 2H^+ \tag{1.1}$$

$$\text{Side reaction: } Me^{3+} + 3HCO_3^- \rightarrow Me(OH)_3 + 3CO_2 \tag{1.2}$$

Note that the reactions presented in equations (1.1) and (1.2) are simple, and cannot be used to estimate the required chemical dosage directly. They must be seen in the light of the many competing reactions taking place, and also the effects of external conditions such as alkalinity, pH, trace elements and ligands found in wastewater [11].

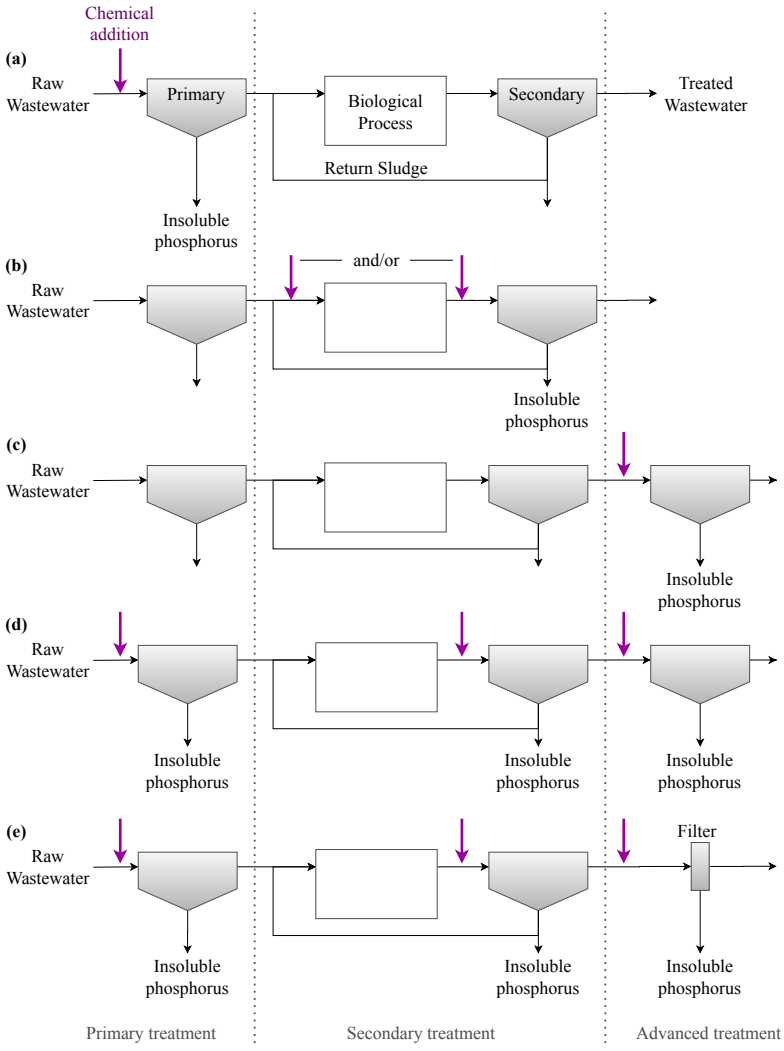

Figure 1.1: Strategies of chemical addition for phosphorus removal: (a) before primary sedimentation, (b) before and/or following biological treatment, (c) following secondary treatment, and (d-e) at several locations (known as "split treatment")(figure modified from [11]).

Various different strategies for chemical phosphorus precipitation exists, and they are usually categorized depending on where in the process the chemical is added and what

treatment is used to collect the phosphorus-incorporated solids. Figure 1.1 shows several removal strategies. The locations for chemical addition can be classified into three methods; (1) pre-precipitation, (2) coprecipitation, and (3) postprecipitation.

Sludge from the secondary settler is returned to the biological process tanks, meaning that a portion of the added metal salts are returned as well. If the strategies shown in figure 1.1 (a), (b), (d) and (e) are used, the removal of phosphorus is affected by a reservoir of chemical precipitant which is established over time, due to the flow of return sludge.

1.2.2 Biological Phosphorus Removal

Chemical precipitation using iron or aluminum is the most commonly applied technique in WWT processes [11]. However, since the 1980s, biological removal has been successfully implemented in full scale plants, leading to extended use of the technology, especially in Denmark [11]. Commonly, biological phosphorus removal (BPR) is implemented in the activated sludge process by leading the wastewater through anaerobic tanks or 'zones' (oxygen and nitrate absent) prior to the aeration process with aerobic/anoxic conditions. The fundamental principle of BPR is the same as for chemical removal; the phosphorus is incorporated into biomass and removed from the process as sludge. The BPR processes occurring in the different zones are:

- Anaerobic conditions: A group of bacteria called phosphorus accumulating organisms (PAOs) are enriched and produce storage capacity in the cell.
- Aerobic conditions: The PAOs can accumulate large quantities of polyphosphate in the bacterial cell, and phosphorus is removed with disposed sludge.

Figure 1.2 illustrates the dynamic behaviour of the phosphorus concentration during anaerobic and aerobic zones.

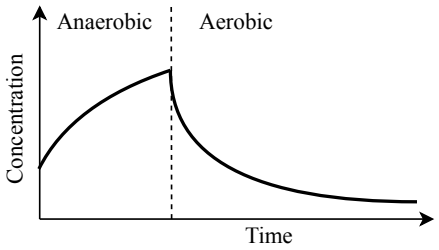

Figure 1.2: Phosphorus concentration in BPR zones (figure modified from [11]).

1.3 Previous Work by the Author

On the previous semester, the author enrolled in a traineeship at the company Krüger A/S, where the student compiled a report regarding system identification of phosphorus precipitation using machine learning [13]. Data was extracted from the online control platform Hubgrade™ (previously known as AQUAVISTA™) and analysed using linear correlation analysis and principal component analysis (PCA) to identify essential input variables for the mathematical model. Out of 41 available signals, 28 were identified

as relevant for the phosphorus dynamics, and henceforth used as inputs to an artificial neural network used for prediction of the phosphorus concentration. The network was designed as a so-called *vanilla recurrent neural network* (vRNN) with online sensorial feedback, and online signal inputs. Cross-correlation and auto-correlation analyses were used to determine the number of previous inputs and outputs to the model. The best network architecture was determined through multiple training tests, and the result was a recurrent neural network with 28 input signals, 4 hidden units, 2 outputs (phosphorus concentration the process tanks and at the outlet). Input measurements of the previous 50 minutes and phosphorus measurements of the previous 20 minutes were used in the model. When validated on new data, the network showed strong performance on forecasting one time step ahead of the phosphorus concentration with $R^2 = 0.99$ and 0.98 for the biological process and outlet, respectively. The methodology will be discussed in detail, as the network is further developed in this work to estimate the system dynamics and perform predictions on the phosphorus concentration.

1.4 State of the Art

Various research has been conducted regarding modelling and control of WWT processes in the recent three decades[7]–[10], [14]–[25]. It is, however, still a major challenge in both industry and academia as the processes are very nonlinear, time-varying dynamic systems containing both physical and biochemical reactions [23].

One of the first attempts at modelling the processes was proposed by Henze, Harremoës, Jansen, *et al.* in the late 1980's as they published the Activated Sludge Model No. 1 (ASM1) [12]. Together with the subsequent augmented ASM models, the ASM model has provided a basis for for many WWTP optimization studies. However, the main drawback of these first principle white-box models is their complexity, lack of reliability, need for expert knowledge and acquaintance of the specific system of interest [10]. Furthermore, mechanistic models like the ASM lack statistical identifiability, and as a result, grey-box (GB) models are often preferred over white-box models [19], [26]. Stentoft, Munk-Nielsen, Vezzaro, *et al.* proposes a GB model set up of stochastic differential equations (SDE) in [19], which is applicable to online model predictive control (MPC). Grey-box models are based upon first principle and incorporate some knowledge about the system of interest, usually in terms of data.

A system may be modelled without any prior understanding of the system using a black-box (BB) model. Using purely data-driven modelling (DDM) methods, the BB models are cost-effective to set up, and the method captures the dominant processes to relate input to output. Black-box models can consequently be seen as alternatives to white-box or grey-box models, when the deterministic models are inapplicable (which can be the case when expert knowledge of the system is unavailable) or simply not valid [7], [26].

As artificial intelligence is advancing with the incursion of the fourth industrial revolution [22], all industries, including the industry of wastewater treatment, are moving towards smarter monitoring and control systems using data-driven models.

Multivariate statistical methods is one branch of data-driven modelling, and a commonly applied method is the principal component analysis (PCA). This method has been applied for WWT processes, and especially the adaptive-dynamic PCA (AD-PCA) has shown great performance in dimensionality reduction and process control [27], [28].

In addition, the classical perspectives on dynamical systems, such as statistical methods are being complemented by an operator-theoretic perspective based on the evolution of measurements in time [29]. This theory provides a linear framework representing nonlinear dynamics. Dynamic mode decomposition (DMD) applies the operator theory and is an algorithm to identify the best-fit linear operator that advances measurements forward in time. It is especially advantageous for high-dimensional measurements, and has been used to discover dynamic patterns from infectious disease data [30]. Dynamic mode decomposition with control (DMDc) is an extended DMD method, originally motivated by decease modelling, specifically epidemiological systems, where it is not possible to stop actuation efforts (like vaccinations) to characterize the unforced dynamics [29]–[31]. Hence this method includes the control in the system identification.

Today, DDM and machine learning algorithms (MLA) like artificial neural networks (NN) are used in many real world industrial problems like function prediction or system identification where processes are not well understood or highly complex [9], [10], [32], [33].

Deep learning has shown its strengths when modelling WWT processes, where it has out-performed conventional machine learning algorithms [34]. Especially the multilayer perceptron (MLP) [1] network and recurrent neural networks (RNN) [32], [35] are established methods for data-driven (black-box) identification. The MLP is the most widely known and used NN architecture, and referrers to a network of nodes where information is passed forward in the structure (known as feed-forward neural networks (FFNN)). However, this method has in general been outperformed by the recurrent neural network when used for dynamic system identification. Hence, a network structure of considerable interest when identifying the process dynamics of the WWTP is the dynamic recurrent neural network (RNN). For decades, RNNs have been used for system identification [35], [36], and are known to have high performance when dealing with time series and highly time-correlated signals [37], [38].

The nonlinear autoregressive exogenous model (NARX) is a dynamical neural network and one type of RNNs, where a MLP with feedback loop of delayed outputs and delayed inputs are used in the architecture. This network has been efficiently applied to prediction of time series [39] and is in theory a computationally powerful method for modeling of nonlinear dynamical systems. NARX has been used for WWT process simulation and prediction [39]–[42], and has proven useful for modelling the nonlinear processes.

Another recurrent network of great interest is the long short-term memory (LSTM) network, which is a very promising neural network architecture with internal dynamics [1]. LSTMs have been successfully applied to problems with complex dynamics, and it is currently a task for future research to determine their properties in an engineering context for system identification. Despite its only recent entry to the field of dynamic

system identification, the architecture has already shown great performance in modelling and prediction of WWT processes. In [22] a LSTM is used to predict ammonium and total nitrogen concentrations in a WWTP. Similarly, a LSTM is designed in the work [16] and [43] to estimate the N_2O emissions from waste water treatment plants.

Most machine learning algorithms and data-driven methods transforms a problem into an optimization task and uses different methods to solve the problem [44]. For neural networks, the task of the optimization (also called *training*) is to determine weights and bias values. However, the optimization algorithm itself has parameters which heavily affect how the model is fit to the data. Those are called *hyperparameters*, and there exists several methods to tune and determine the optimum hyperparameters. Manual tuning is the most basic approach and usually only performs well with good process understanding and expert knowledge within the field [37]. Another approach is automatic hyperparameter tuning, where several methods exists; Grid search [45], random search [46] and model based optimization such as Bayesian optimization [47].

1.5 Report Structure

This report is divided in 10 chapters, where chapter 2 gives a detailed description of the case plant considered in this work. The physical system is described alongside the currently implemented online control scheme.

In chapter 3, the scope of the project is presented. A research question is formulated, providing the basis of the entire project. Subgoals and limitations to the project are formulated around the research question to constrain the problem.

In chapter 4, the data acquisition and pre-processing is described. The data is processed, scaled and presented in various plots and subsequently investigated for linear correlation between variables.

A linear model is presented in chapter 5 while nonlinear models are presented in chapter 6. The applied methods are explained in the introductory sections of each chapter.

The obtained dynamic models are then evaluated and compared in chapter 7. First, the model training performance is evaluated, and later each network is evaluated on a new, independet set of data.

Results of the project are discussed in chapter 8 and concluded in chapter 10, while further investigations in the area are discussed in chapter 9.

1.6 Terminology

Building dynamic nonlinear models from data is a challenging task as it requires merging of two expertise communities: control engineers and researchers and machine learning people and statisticians [1]. Furthermore, as the system of interest in this project lies within the field of biochemical engineering, terminology from three different expertise communities are used and merged in this work. The terminology of this work will mainly follow standard system identification and optimization literature, however, for the sake of convenience and brevity, the following expressions are often used:

- Neural Network (NN): Short for artificial neural network.
- Process: Used as synonym for the system under study.
- Model: The digital reconstruction of the process. Other synonyms used can be simulation or (neural) network.
- Reactor: Biological aeration tanks. In biochemical engineering these are often referred to as process tanks, but that usage is avoided in this work.
- Parameters: Constant terms that influence the model in an either linear or nonlinear way.
- Variables: General term for changing features of the process. Other synonyms used can be signals, inputs, outputs.
- Training: Optimization of the model structure and/or parameters in order to minimize a given cost function for the training data. This literal definition is adopted from [1].
- Generalization: Evaluation of the model output for input data samples that are not contained in the training data set. This literal definition is adopted from [1].
- Estimation: The procedure of building the model i.e optimizing the parameters that best describes the system dynamics.
- Prediction: The procedure of estimating the outcomes for unseen data using an already developed model. Prediction can be, but is not necessarily time based.
- Forecasting: Is a sub-discipline of prediction, where temporal information is included in the procedure. Forecasting is hence time-based prediction where future unseen events are predicted.

2 Case Plant: Kolding Central WWTP

The plant of interest is Kolding central WWTP located in Agtrup Denmark. The system is introduced in this chapter, where first the physical specifications of the plant is presented in section 2.1 followed by a description of the implemented control scheme in section 2.2 with a presentation of the sensors and data signals. Within this section is a presentation of the data gathered with the current control scheme.

2.1 System Description

Agtrup WWTP has a capacity of 125,000 population equivalent (PE), and is currently operating with a load of approximately 65.5% [48]. The wastewater is led from the primary treatment to the biological treatment and distributed via two pipelines to two pairs of biological tanks/reactors. An illustration of the system is shown in figure 2.1. To remove phosphorus from the wastewater, chemical precipitant is added to the process at two locations; (1) to the mixed liquor of wastewater and sludge at the inlet to the biological reactors, and (2) to the effluent from the biological treatment process. This strategy has previously been illustrated in figure 1.1 (b).

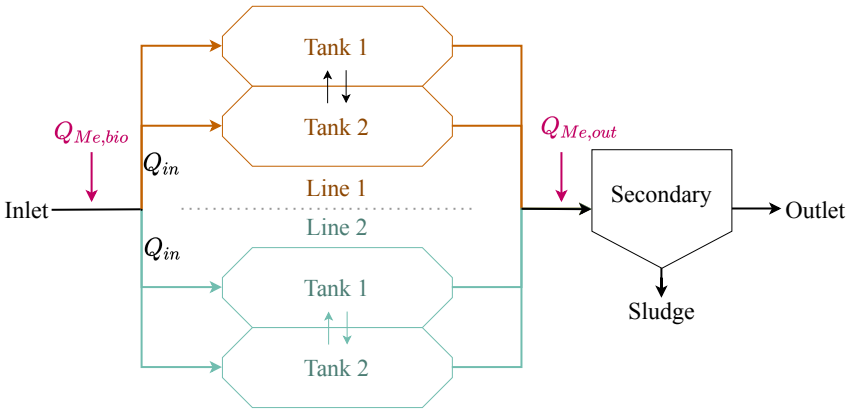

Figure 2.1: Schematic of the flow lines in the system. All relevant measurements are explained in table 2.1.

In addition to chemical precipitation, the plant also utilizes biological phosphorus removal. BPR takes place when the plant experiences low-load conditions, which is usually at night. During the low-load periods, the BPR control, called BioP focus is activated and prolonged denitrification periods can be imposed, where anaerobic conditions can be obtained to promote BPR.

2.2 Chemical Precipitation Control

2.2.1 Hubgrade™

The chemical precipitation strategy is executed using a platform called Hubgrade™ (previously known as AQUAVISTA) provided by Krüger A/S. The platform is a supervisory control and data acquisition (SCADA) system, which controls the dynamic set-point of precipitation dosage based on the online measurement of phosphate (PO_4^{3-}) in the biological process and at the outlet.

The precipitation control uses the following online measurements:

- Inflow of raw wastewater and return sludge (Q_{in})
- Phosphate concentration in biological process tanks (P_{bio})
- Phosphate concentration in outlet of the plant (P_{out})

The scheme of the precipitation module is shown as a block diagram in figure 2.2.

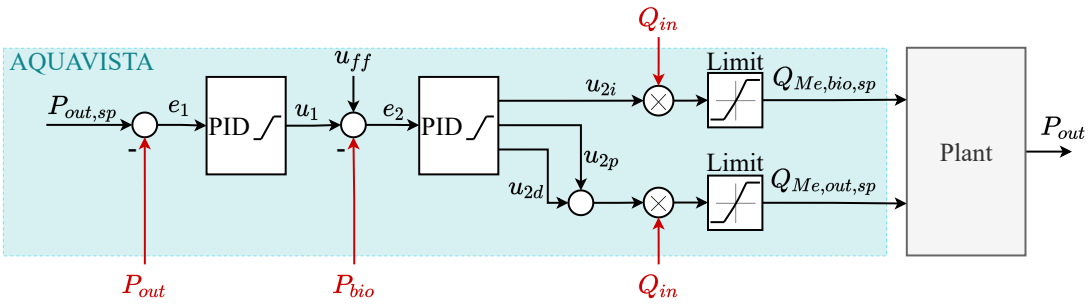

Figure 2.2: Hubgrade phosphorus precipitation module. Measurement feedback signals are marked with red. The notation is explained in table 2.1

Table 2.1: Hubgrade symbol explanation.

Symbol	Description	Unit
$P_{out,sp}$	Set-point for phosphate in outlet	mg/L
u_{ff}	Default feedforward term	mg/L
$Q_{Me,sp}$	Set-point flow of chemical precipitant	mg/L
P_{out}	Phosphate concentration in outlet	mg/L
P_{bio}	Phosphate concentration in process tank	mg/L
Q_{in}	Inflow of raw WW and return sludge	m^3/h
$Q_{Me,bio,sp}$	Flow set-point for chemical precipitant to biology	m^3/h
$Q_{Me,out,sp}$	Flow set-point for chemical precipitant to settler	m^3/h
e_1	Outlet residual	mg/L
e_2	Biology residual	mg/L
u_1	Control output of 1st PID	
u_2	Control output of 2nd PID	

Hubgrade generates two flow set-points for chemical precipitant; one for the addition of chemical to the biological tanks and one for the settler. As shown in figure 2.2, the P and D-terms of the last PID controller are added together and used as the set-point for the pump delivering precipitant to the settler. The I-term is fed to the pump controlling the chemical precipitant to the process tanks. Limitations to the controller outputs and the flow set-point of chemical precipitant are applied, and flow set-points are saturated. However, these flow limits have been changed several times over the period of which the data is collected. The same accounts for the PID gains [k_p,k_i,k_d]. They have been changed in an attempt to increase controller performance based on engineering judgement and experience with the process.

Hubgrade provides the flexibility of choosing different control levels, according to the hardware (sensors) implemented in the system. Six control levels can be chosen, where the most profitable level will be used if possible. The control levels are ranked as follows:

1. Utilize the method shown in figure 2.2, where all measurements are available (P_{bio}, P_{out} and Q_{in}).
2. If phosphate measurements in the outlet (P_{out}) are not available, only use the default phosphate set-point in the process tank, and disregard the outer control loop in figure 2.2.
3. If phosphate measurements in the process tank (P_{bio}) are not available temporarily, use a mean value of the measurements before it became unavailable.
4. If both phosphate measurements are unavailable of this line, use the set-point generated from another line (a so-called *master line*).
5. If all phosphate measurements are unavailable, generate a set-point for chemical precipitant, which is proportional to the inflow of raw wastewater and sludge (Q_{in}). Referring to figure 2.2, the two outer control loops are disregarded in this control level.
6. If the measurement of inflow wastewater and sludge is not available, disregard all feed-back loops of figure 2.2, and use a default chemical precipitant dosage as set-point.

Since two phosphate sensors are available at the case plant; one in reactor 1 and one at the outlet, the case plant can utilize control where chemical addition is done at two points.

The BPR is controlled using integer code to describe the conditions governing each biological tank. A four digit code is used in Hubgrade to describe the process for each of the biological lines shown in figure 2.1. These codes are referred to as biological phase codes, and they are presumed to have a major impact on the phosphorus concentration in the reactors, especially when Bio-P focus is enabled. The meaning of each digit in the phase codes are described in table 2.2.

Table 2.2: Biological phase codes definition.

Digit	Description
1	Denotes which tank the wastewater flows to. Can take the value of 1 or 2
2	Denotes which tank the effluent flows from. Can take the value of 1 or 2
3	Denotes the conditions in tank 1. Can take the value of 0, 1 or 2.
4	Denotes the conditions in tank 2. Can take the value of 0, 1 or 2.

As noted in table 2.2, digit 3 and 4 can take values between 0-2. They define the conditions in tank 1 and 2, respectively. The three possible options are:

`0`: Anaerobic conditions without aeration and mixing.
`1`: Anoxic conditions (denitrification) without aeration and with mixing.
`2`: Aerobic conditions (nitrification) with aeration and mixing.

An example could be: The wastewater in biology line 1 flows to tank 1 and the effluent out of tank 2. In tank 1 there is nitrification and in tank 2 there is denitrification. The phase code, ϕ, for biology line 1 will accordingly be:

$$\phi = 1221$$

2.2.2 Data Visualization

Figures 2.3 and 2.4 show how the current Hubgrade control strategy performs. The phosphate concentration in both the biology and outlet experiences oscillations, which most likely is caused by the linear controllers regulating this inherently nonlinear process.

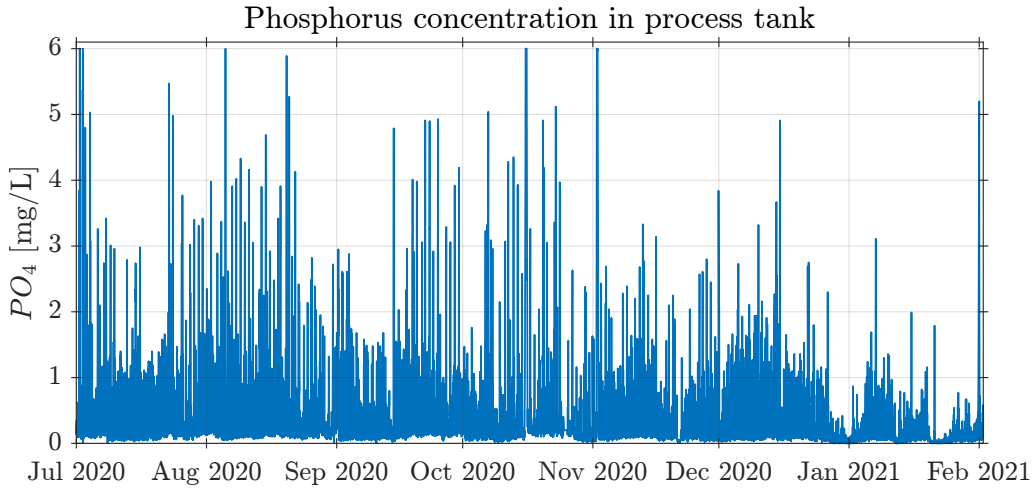

Figure 2.3: PO_4 concentration in the biological tank with the current Hubgrade control.

In some periods, the outlet concentration is very low, which is an indication of unnecessary high precipitation. An example of this can be seen in figure 2.5, where the PO_4^{3-} concentration in the biological tank is above the discharge limit around time 06:00, so

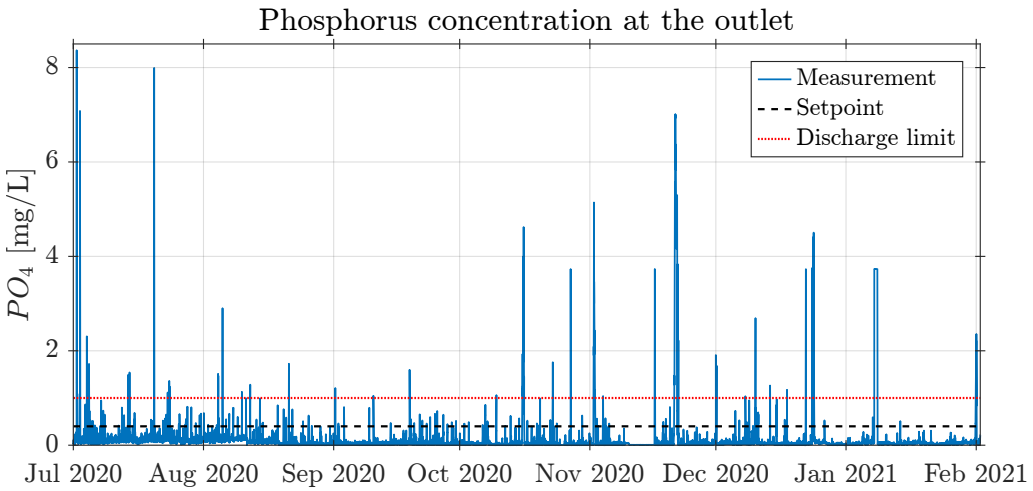

Figure 2.4: Outlet concentration of PO_4 with the current Hubgrade control.

removal must take place. However, when observing the outlet concentration in the same period and the following hours, it is near zero.

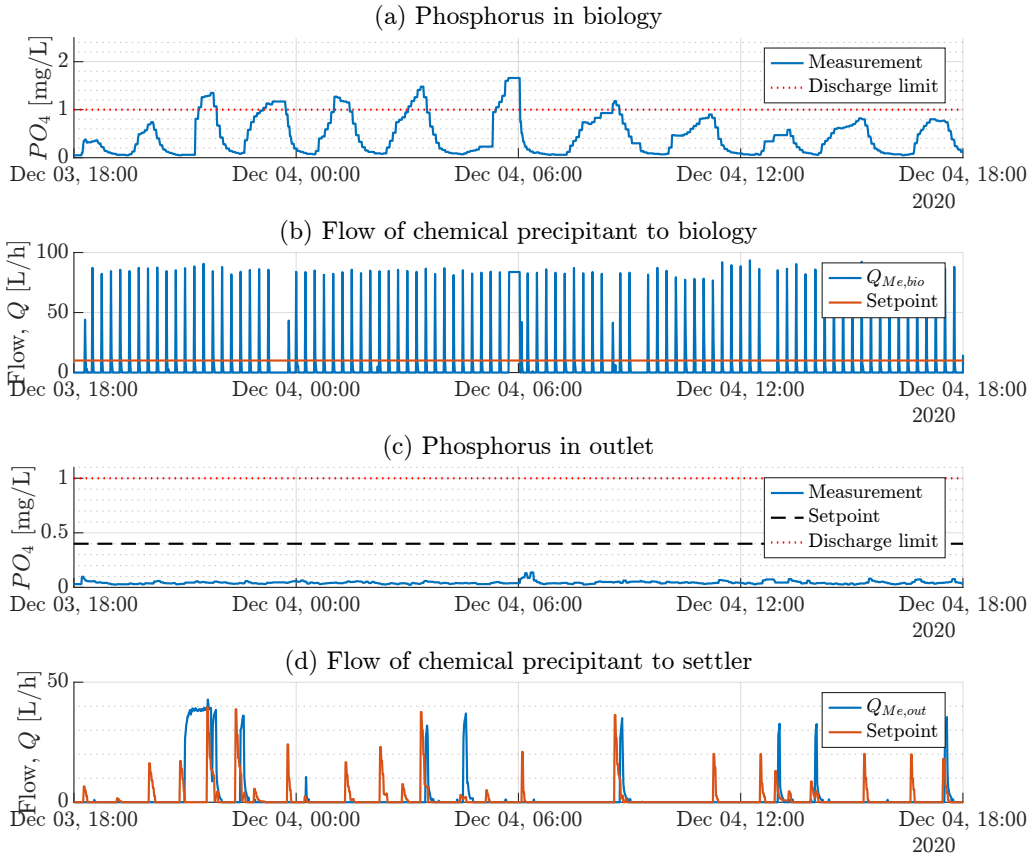

Figure 2.5: An example of unnecessary high chemical removal with the current control scheme.

3 Scope of the Project

Research Question

The overall motivation for this project is to design a model which captures the dynamic behaviour of the phosphorus in wastewater and accurately estimates the concentration in the wastewater. Hence, this work will provide the foundation for model-based or model predictive control of phosphorus removal in WWTPs. It will be investigated how data-driven system identification and machine learning can be used to predict the phosphorus concentration in wastewater. Both linear and nonlinear models will be investigated to gradually increase the complexity of the models developed throughout this work. This yields the research question:

> *Which methods within data-driven system identification and machine learning yields the best performance when used to predict phosphorus concentration in wastewater?*

Project Goals

With the knowledge obtained from the literature review conducted in section 1.4, some specific goals are set to answer the research question. The goals are listed below.

- Propose a linear dynamic model to estimate the phosphorus concentration in the wastewater at two different locations in the WWTP.
- In continuation of the previous project, extend the RNN model to estimate the phosphorus concentration. Investigate if the model prediction performance is improved if the LSTM neural network structures is used.
- Propose an optimization algorithm to determine the best model structure.
- Cross-validate the proposed dynamic models and discuss each model's advantages and disadvantages.

Design Specifications and Limitations

To constrain the problem, the solution should meet the following design specifications and limitations.

- No experimental tests should be conducted at the case plant, thus, validation and

evaluation of the proposed model will be based solely on data from the daily operation.
- Data treatment and handling of missing data will be kept to a minimum and only included to the extend of necessity.
- Only the phosphorus in the biological reactor is modelled.

4 Data Processing and Analysis

High dimensional data usually contains noise, missing values and redundant information. Hence, in order to understand the process behaviour properly, it is necessary to extract the important information from the large number of measured variables. Data pre-processing and multivariate statistical methods are powerful tools to organize, clean up and investigate the original data and reduce its dimensionality so that it only contains essential information. In this section, the data pre-processing is described and the dataset is subsequently investigated in an exploratory data analysis for dependency between variables. The variables used as model inputs are described and presented.

4.1 Plant Process & Data Gathering

Sensor data and operation data from the case plant is collected and stored in the online control platform Hubgrade. The data is extracted from the cloud using the software R, merged and extracted in matrices to compile with MATLAB. A total of 71 data variables are extracted from Hubgrade, including sensor measurements, watchdogs, process modes, control outputs, and sometimes smoothed measurements or engineered variables. All types of data are described below:

Measurements are sensor measurements of the process. The available measurements are presented in table 4.1.

Watchdogs are alarm signals activated when certain signals exceed predefined limits. There are watchdogs monitoring the phosphorus concentration, inlet flow, set-point generation and quality of the sensor measurements. The activation of a watchdog typically entail a different process mode to start.

Process Modes are Boolean signals or integer code depicting the process mode in action. There are three process modes of interest to this work; chemical phosphorus removal, biological phosphorus removal and forced aeration when the phosphorus concentration is too high. Furthermore, the phase codes described in chapter 2 are characterized as process modes in this work, as they take values between [0,2] and describe events similar to the process modes.

Engineered variables are virtual variables constructed and calculated in Hubgrade. These can appear in the form of filtered/smoothed measurements, and as variables constructed by addition, subtraction, multiplication or division of other signals.

Table 4.1: Available measurements in the system.

Measurement	Location of the sensor	Notation	Unit
PO_4	Tank 1 line 1 and outlet	P_{bio}/ P_{out}	mg/L
NH_{4+}	Tank 1 line 1, tank 1 line 2	NH	mg/L
NO_{3-}	Tank 1 line 1	NO	mg/L
pH	Inlet	pH	-
Temperature	Inlet, outlet, all tanks	T	°C
Return sludge flow	All tanks and the accumulated flow	Q_{RS}	m^3/h
Waste sludge flow	Secondary settler	Q_{WS}	m^3/h
Wastewater flow	Inlet, line 1, line 2	Q	m^3/h
Suspended solids	Tank 1 line 1, tank 2 line 2 and waste sludge	SS	kg/m^3

The collected data is ranging from July 1, 2020 to February 1, 2021, resulting in 7 months of data with a total of 71 signals.

4.2 Rebuilding & Detecting Feasible Data

Data quality assessment is incorporated in Hubgrade, meaning that each data sample logged in Hubgrade is assigned a quality value. Data samples appointed a bad quality are removed from the dataset and replaced with the most resent data sample of good quality, hence applying zero-order hold for the data sampling.

Occasionally, no value is logged, either due to the sensors simply skipped a sample, because the sensor is re-calibrating or because the computations performed in the cloud by Hubgrade were not completed within the 2 minutes. When data is missing, the same procedure applied for bad quality data is applied, and the sample is appointed a value of the most resent data sample of good quality.

The sample frequency varies from signal to signal. Some variables are sampled with 1 minute intervals, and other variables with 2 minute intervals. Hence, all data logged with 2 minute sample intervals are upsampled in order to not erase any dynamics in the signals sampled with 1 minute intervals. This method is used primarily because the flow of chemical precipitant, which is one of the most significant contributions to the phosphorus removal, is timer-controlled and the pumps operates every 15 minutes. Hence, downsampling of these signals would result in removing significant dynamics from the dataset.

4.3 Data Scaling

Data scaling is performed to ensure better performance of the machine learning models, as it is common for the relative values of features to vary more than an order of magnitude [49]. If that is the case, the training process can face the problem of ill-conditioning, where

the cost function has an inherent tendency to become more sensitive to some features. There are typically two types of data scaling used in machine learning algorithms [49];

1. Standardization: In order to remove bias effects, the data can be mean-centered. Mean centering is often paired with *standardization*, where each feature is divided with its standard deviation, resulting in a dataset with features presumed to have been drawn from a normal distribution with zero mean and unit variance.
2. Min-max normalization: The min-max normalization is useful when the data needs to be scaled in the range [0,1]. This is performed using equation (4.1), where X_{min} and X_{max} is the minimum and maximum value of the feature, respectively.

$$X_{scaled} = a + \frac{X - X_{min}}{X_{max} - X}(b - a) \tag{4.1}$$

For min-max normalization, $a = 0$ and $b = 1$, but the normalization can be applied to scale the features between any chosen interval [a,b].

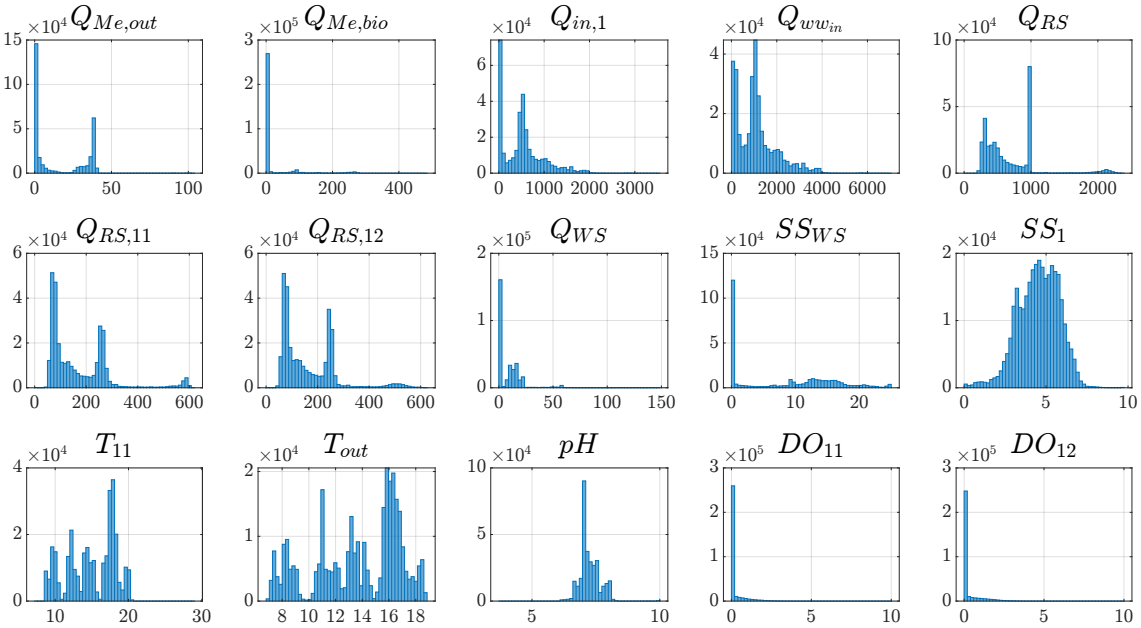

Figure 4.1: Histograms of all input signals. The used symbols are explained in table 4.2.

In order to effectively apply standardization, the data must be normally distributed. However, that is not the case for most of the measurements as shown in figure 4.1. Hence, the dataset is scaled using min-max normalization according to equation (4.1), and a boxplot of the scaled data is shown in figure 4.2. Compared to the original data shown in figure 4.3, the signals are more evenly distributed, and the order of magnitude variation for variables like $Q_{in,1}$ and $Q_{ww,in}$ is eliminated.

4.4 Correlation Analysis

A linear correlation analysis was performed on raw data in previous work by the author[13], and the applied methods and conclusions are presented in this section to sup-

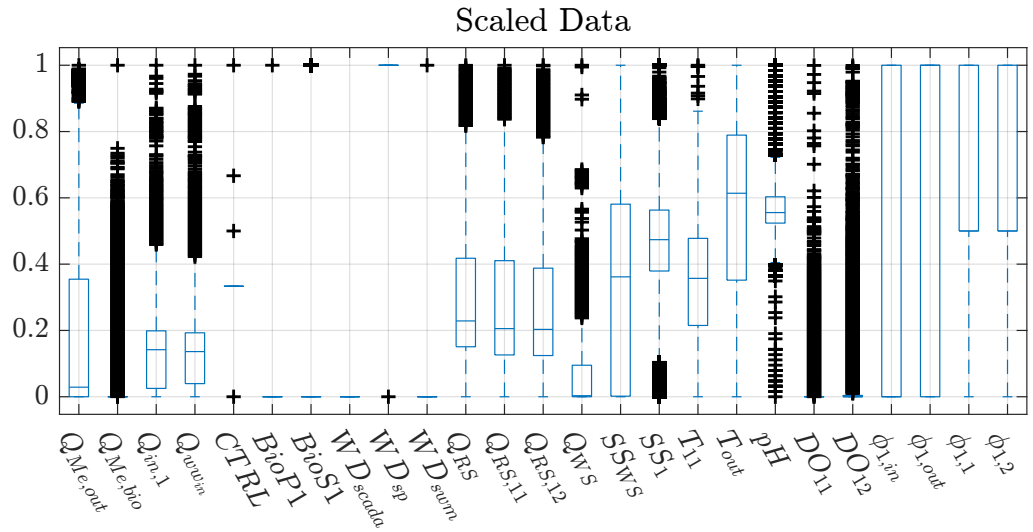

Figure 4.2: Boxplot of the scaled data using min-max normalization.

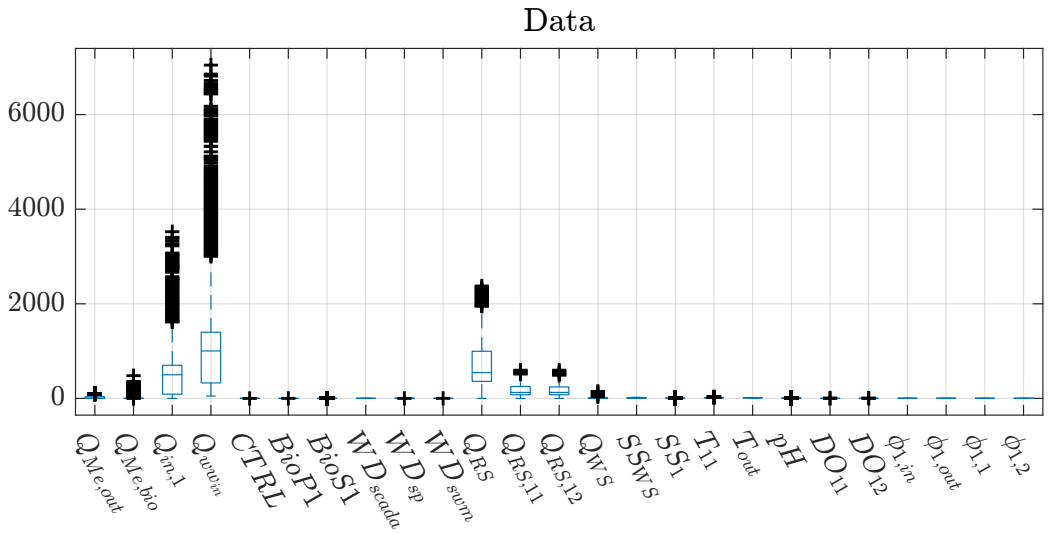

Figure 4.3: Boxplot of the original data.

port the statement that the process is nonlinear and to underline the difficulties associated with modelling this system. Furthermore, the analysis presented in this section is expected to give the reader a more thorough understanding of the biochemical processes.

In the previous work, a subset of relevant input variables were identified, and the chosen input variables of previous work are likewise used as the inputs in this work. Three of the 28 input variables are excluded, as these were related only to the outlet phosphorus concentration, leaving 25 features as input variables to the model. Those 25 inputs are explained in table 4.2.

Initially, linear correlation analysis is applied to investigate if there are linear relationships between variables. It is however important to state that *correlation does not apply causation*, meaning that strong linear correlation between variables only implies linear system dynamics. To confirm such a statement, a more thorough analysis must be con-

ducted. The linear correlation between input variables extracted from Hubgrade is examined, and the result is shown in figure 4.4.

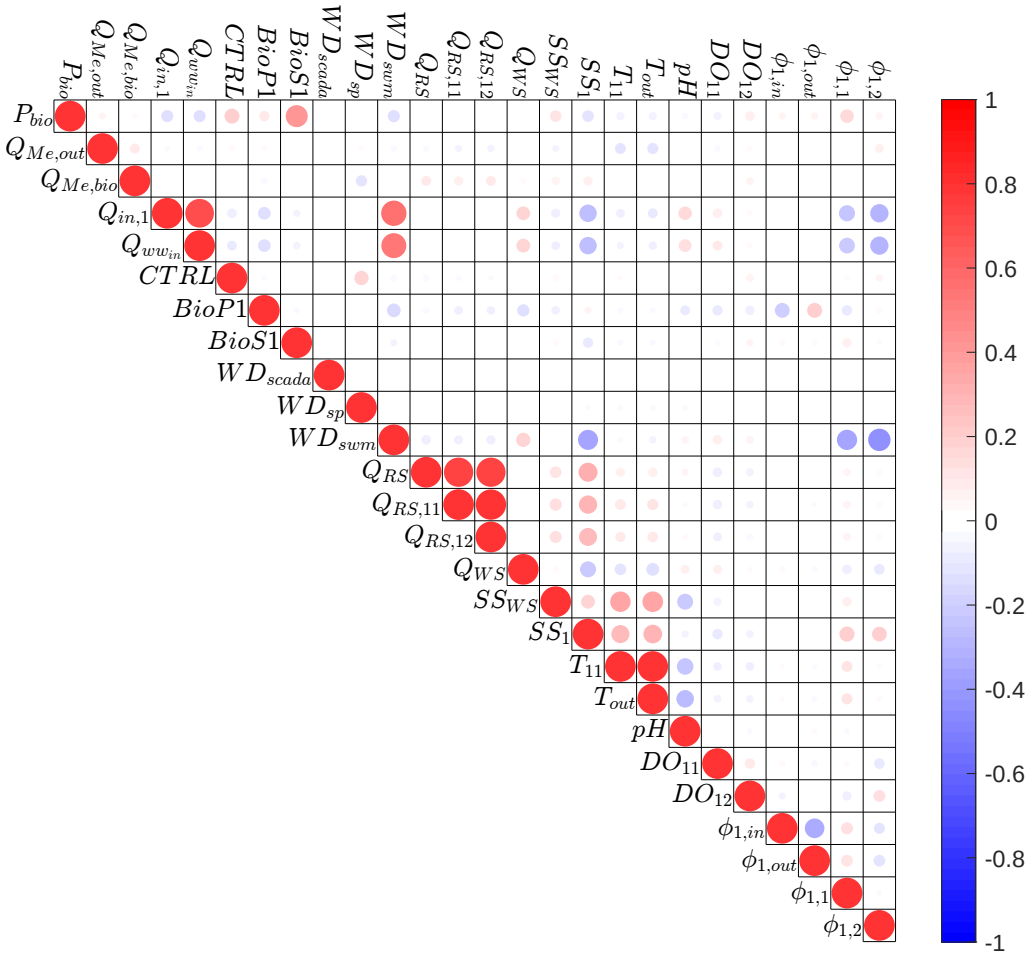

Figure 4.4: Linear correlation matrix of logged variables using the Pearson correlation coefficient.

The figure shows dependency between variables based on Pearson's correlation coefficient arranged in an upper triangular correlation matrix. Table 4.2 explains the symbols and variable names used in the correlation matrix shown in figure 4.4, where the type denotes if a variable is a measurement, watchdog or process mode.

Table 4.2: Variables logged in Hubgrade.

Notation	Type	Description
P_{bio}	Measurement	PO_4 measurement in line 1 tank 1
$Q_{Me,out}$	Measurement	Flow of metal salt to secondary settler
$Q_{Me,bio}$	Measurement	Flow of metal salt to biological tanks
$Q_{in,1}$	Measurement	Inlet flow of WW and return sludge to biology line 1
$Q_{ww,in}$	Measurement	Inlet flow of WW to the entire biological process
CTRL	Process mode	Control level of chemical precipitation module

Table 4.2 continued from previous page

BioP1	Process mode	Biological phosphorus removal (BioP focus) line 1
BioS1	Process mode	Enforces aeration if PO_4 concentration is too high during BioP focus in line 1
WD_{scada}	Watchdog	Set-point watchdog monitoring the quality and alarms if concerning
WD_{sp}	Watchdog	Set-point watchdog monitoring the set-point generated by Hubgrade and alarms if concerning
WD_{swm}	Watchdog	Storm water mode (swm) watchdog. Alarms if inflow of wastewater is too high. The biological tanks are then used for sludge settlement.
Q_{RS}	Measurement	Flow of return sludge from secondary settler
$Q_{RS,ij}$	Measurement	Flow of return sludge from secondary settler to line i tank j
Q_{WS}	Measurement	Flow of discharged (waste) sludge from secondary settler
SS_{RS}	Measurement	Suspended solids (SS) measurement of return sludge
SS_{WS}	Measurement	Suspended solids (SS) measurement of waste sludge
SS_i	Measurement	Suspended solids (SS) measurement line i
T_{ij}	Measurement	Temperature of wastewater in line i tank j
T_{out}	Measurement	Temperature of wastewater at outlet
pH	Measurement	pH value of inlet wastewater
DO_{ij}	Measurement	Dissolved oxygen concentration in line i tank j
$\phi_{i,in}$	Process mode	Denotes which tank the wastewater flows to in line i
$\phi_{i,out}$	Process mode	Denotes which tank the effluent flows from in line i
$\phi_{i,j}$	Process mode	Denotes the process phase in line i tank j

When the process mode BioP safe is activated, it enforces aeration in the reactors and aerobic conditions are established to decrease the phosphorus concentration in the reactor. As a result of this, the phosphorus concentration is naturally positively correlated to this process mode.

There is a negative correlation between the PO_4 measurement and SS in the reactors, which can be explained by the reservoir of metal salts in the return sludge as described in section 1.2.

Some of the other strong correlations, which does not affect phosphorus concentration explicitly are mentioned below:

- SS in supply lines (SS_1) is negatively correlated with wastewater inflow ($Q_{ww,in}$ and $Q_{in,1}$) and the storm water mode (SWM) watchdog. When the inflow of wastewater is high, it is often due to rainfall and then SWM activates (hence the positive correlation between Q_{in} and WD_{swm}). The proportions of sludge in the raw wastewater

naturally decreases, and SS is a measure of this proportion.

- SS in line 1 is positively correlated to the flow of return sludge. SS is a measure of sludge per volume mixed water and sludge. Hence, a higher flow of return sludge results in a higher SS in biological reactors. The same principles applies for SS_{WS} and Q_{WS}.
- The temperatures of the wastewater in the biological reactor and at the outlet are strongly correlated, which is natural as they are measured simultaneously with the same environmental conditions.

The correlation matrix gives a good indication of how the variables affects the phosphorus concentration and which inputs should be included in the model design. It is nevertheless difficult to determine which variables contain the most information and have the highest contribution when designing the model for phosphorus concentration predictions and estimations. Additionally, the correlation coefficients determined in figure 4.4 are linear correlation coefficients, meaning that nonlinear relationships can occur without the analysis depicting it.

5 Linear Data-driven System Identification

Whenever possible, it is desired to work with linear systems or approximate systems with linear models of the form presented in equation (5.1). Linear systems are simple to solve, and there exists many methods for analysis, prediction, simulation and control of such systems [29].

$$\frac{d}{dt}\mathrm{x} = \mathbf{A}\mathrm{x} \tag{5.1}$$

The dynamics of a linear system are entirely described by the eigenvalues and eigenvectors of the the matrix $\mathbf{A}$, and can easily be transformed into eigenvector coordinates where the dynamics are decoupled [29].

As data has become easily available and increasingly abundant, data-driven regression and machine learning has become vital tools to discover dynamical systems, and one particularly interesting technique is the dynamic mode decomposition (DMD). DMD was developed in the fluid dynamics community to identify dynamics from high dimensional data, but has subsequently been used in various fields of study, including disease modelling [30] and stock price prediction [50]. In this chapter, the system is modelled using DMD and DMD with control (DMDc).

5.1 Dynamic Mode Decomposition

The DMD method utilizes the computational efficient singular value decomposition (SVD) to determine a matrix pseudo-inverse. DMD can be used to provide effective dimensionality reduction of very high dimensional system, and is commonly applied in the fluid dynamics community where the systems are difficult to analyze and construct controllers for due to the enormous number of spatial states needed for modelling. The method is inherently data-driven, and the first step of the algorithm is to arrange the collected data into two data matrices as shown in equations (5.2) and (5.3).

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathrm{x}(\mathrm{t}_1) & \mathrm{x}(\mathrm{t}_2) & \dots & \mathrm{x}(\mathrm{t}_\mathrm{m}) \\ | & | & & | \end{bmatrix} \tag{5.2}$$

$$\mathbf{X}' = \begin{bmatrix} | & | & & | \\ \mathrm{x}(\mathrm{t}'_1) & \mathrm{x}(\mathrm{t}'_2) & \dots & \mathrm{x}(\mathrm{t}'_\mathrm{m}) \\ | & | & & | \end{bmatrix} \tag{5.3}$$

The data matrices consists of the column vectors $\{\mathrm{x}(\mathrm{t_k}), \mathrm{x}(\mathrm{t'_k})\}_{\mathrm{k=1}}^{\mathrm{m}}$ denoting snapshots of the system states as they evolve in time where $t'_k = t_k + \Delta t$, and Δt is sufficiently small to resolve the highest frequencies in the dynamics. Assuming uniform sampling time, we use the notation $\mathrm{x_k} = \mathrm{x}(\mathrm{k}\Delta\mathrm{t})$. DMD then seeks to find the best fit linear operator, $\mathbf{A}$, that best advances snapshot measurements forward in time as given by equation (5.4).

$$\mathbf{X}' = \mathbf{A}\mathbf{X} \tag{5.4}$$

Which establishes the linear dynamic system given in equation (5.5).

$$\mathrm{x_{k+1}} = \mathbf{A}\mathrm{x_k} \tag{5.5}$$

Where $\mathbf{A}$ is defined in

$$\mathbf{A} = \mathrm{argmin}\left\|\mathbf{X}' - \mathbf{A}\mathbf{X}\right\|_{\mathrm{F}} = \mathbf{X}'\mathbf{X}^{\dagger} \tag{5.6}$$

Where $\|\cdot\|_F$ is the Frobenius norm and † denotes the pseudo-inverse. Equation (5.6) presents the *exact* DMD method. However, for a high dimensional state vector ($\mathrm{x} \in \mathbb{R}^{\mathrm{n}}$) the matrix $\mathbf{A}$ has n^2 elements. Hence, the DMD algorithm can be defined using the SVD of $\mathbf{X}$, making it unnecessary to compute $\mathbf{A}$ and the pseudo-inverse $\mathbf{X}^{\dagger}$ directly [29].

5.2 Dynamic Mode Decomposition with Control

If the system of interest is affected by external inputs, applying the DMD algorithm will often result in incorrect estimations of the dynamics, as the effects of actuation are mistaken with the effects of internal dynamics [29]–[31]. Hence, an extension of DMD exists where the method incorporates the effect of control, and is now dynamic mode decomposition with control (DMDc) [31]. DMDc capitalizes on all the advantages of DMD and is furthermore able to disambiguate between underlying dynamics and the effects of actuation. Similar to DMD, the first step is to define the snapshot matrices $\mathbf{X}$ and $\mathbf{X}'$ as in equations (5.2) and (5.3). Furthermore, the input snapshots are arranged in a matrix as shown in

$$\boldsymbol{\Upsilon} = \begin{bmatrix} | & | & & | \\ \mathrm{u}(\mathrm{t}_1) & \mathrm{u}(\mathrm{t}_2) & \dots & \mathrm{u}(\mathrm{t}_\mathrm{m}) \\ | & | & & | \end{bmatrix} \tag{5.7}$$

Assuming uniform sampling, we can use the notation $\mathrm{u_k} = \mathrm{u}(\mathrm{k}\Delta\mathrm{t})$. Equation (5.5) is modified, so the future state $\mathrm{x_{k+1}}$ now relies on the current state $\mathrm{x_k}$ and the current control input $\mathrm{u_k}$ and is given by equation (5.8)

$$\mathrm{x_{k+1}} = \mathbf{A}\mathrm{x_k} + \mathbf{B}\mathrm{u_k} \tag{5.8}$$

Which can be written as equation (5.9).

$$\mathbf{X}' = \mathbf{A}\mathbf{X} + \mathbf{B}\boldsymbol{\Upsilon} \tag{5.9}$$

If the map $\mathbf{B}$ is unknown, approximations of both $\mathbf{A}$ and $\mathbf{B}$ must be found from state and input snapshots. First the system in equation (5.9) is rewritten as equation (5.10) where $\mathbf{\Omega} = \begin{bmatrix}\mathbf{X}^\top & \mathbf{\Upsilon}^\top\end{bmatrix}^\top$ and $\mathbf{G} = [\mathbf{A}\ \mathbf{B}]$.

$$\mathbf{X}' = \begin{bmatrix}\mathbf{A} & \mathbf{B}\end{bmatrix} \begin{bmatrix}\mathbf{X} \\ \mathbf{\Upsilon}\end{bmatrix} = \mathbf{G\Omega} \tag{5.10}$$

Now $\mathbf{G}$ can be found through least-squares regression and is given by equation (5.11).

$$\mathbf{G} = \mathbf{X}'\mathbf{\Omega}^\dagger \tag{5.11}$$

5.3 Implementing Linear Models

The benefit of the DMDc algorithm is the possibility to utilize it as an online adapting algorithm. Hence, the method is tested on data from the case plant. Using a 12-hour period of case plant data, the principles of DMD and DMDc can be visualized, and the pitfall of naively applying DMD to data from a system with actuation can be illustrated. Applying DMD/DMDc to the data from the case plant, results in a least squares (LS) linear fit, which is shown in figure 5.1. The top plot (figure 5.1 (a)) shows 1 out of the 12 hour dataset the two models were fitted to. The narrow snapshot of the data is chosen to show how the two estimates of the next step seems to be a time-delayed signal of the original data, hence indicating the $\mathbf{A}$ matrices are identity matrices. Recall, the one step ahead prediction in figure 5.1 (a) is performed utilising equations (5.12) and (5.13).

$$\text{DMD: } \hat{\mathrm{x}}_{k+1} = \mathbf{A}\mathrm{x}_\mathrm{k} \tag{5.12}$$

$$\text{DMDc: } \hat{\mathrm{x}}_{k+1} = \mathbf{A}\mathrm{x}_\mathrm{k} + \mathbf{B}\mathrm{u}_\mathrm{k} \tag{5.13}$$

Where $\hat{}$ marks that the value is an estimate of the model. Using the just found linear models, the system can be predicted utilising equations (5.14) and (5.15), where the next state estimate is based on the current state estimate.

$$\text{DMD: } \hat{\mathrm{x}}_{k+1} = \mathbf{A}\hat{\mathrm{x}}_\mathbf{k} \tag{5.14}$$

$$\text{DMDc: } \hat{\mathrm{x}}_{k+1} = \mathbf{A}\hat{\mathrm{x}} + \mathbf{B}\mathrm{u}_\mathrm{k} \tag{5.15}$$

As expected when naively applying the DMD algorithm, it identifies incorrect system dynamics, which is clear from figure 5.1 (b). However, the DMDc algorithm shows promising performance on the 12-hour prediction illustrated in figure 5.1(b). Hence, the model generalizability is further evaluated by using it to predict the process with new data. The DMDc model is obtained by least squares fitting on a 12-hour dataset ranging November, 22 19:00 - November 23, 07:00. It is then subsequently used to predict the next 4 hours, and the result is shown in figure 5.2.

Figure 5.2 shows the data which the model was fit to (left) and the following 4 hours of data compared to the prediction using the obtained model. The model shows poor generalizability, even for a short horizon as 4 hours.

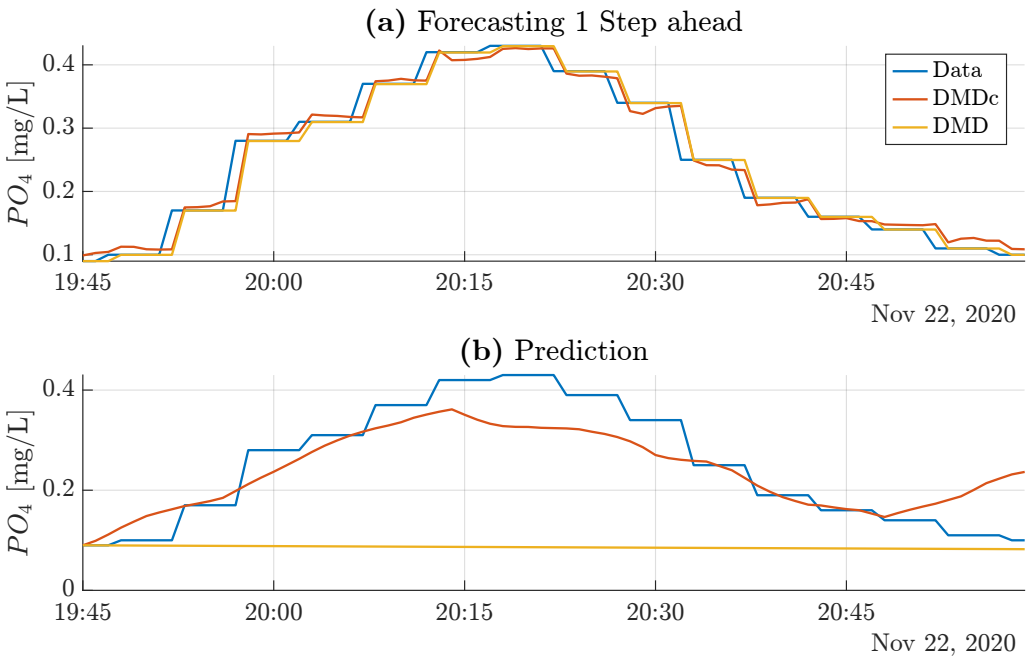

Figure 5.1: (a) Forecasting 1 step ahead for the process dynamics, and (b) Prediction of the process dynamics given initial conditions.

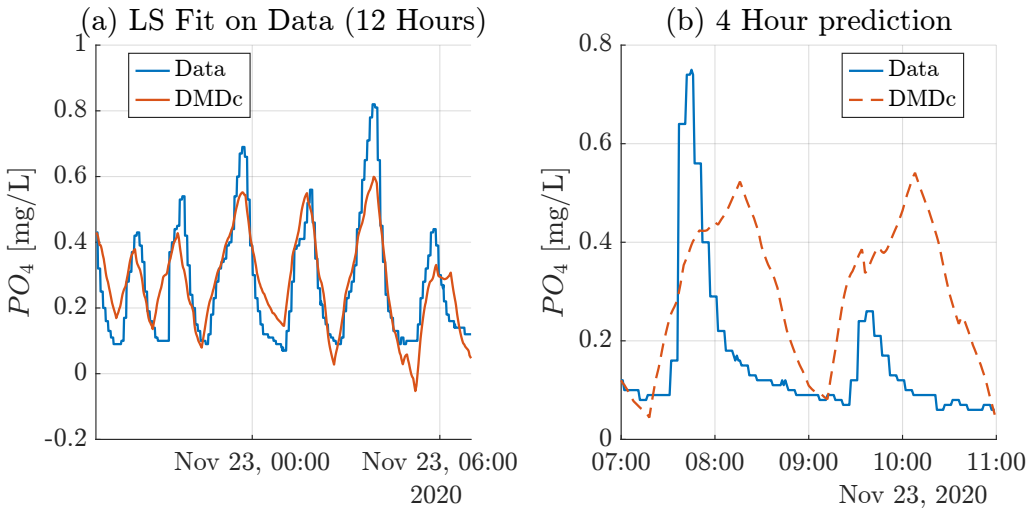

Figure 5.2: Left: least squares fit to the data set. Right: Predicting 4 hours using the estimated model.

The DMDc algorithm is examined at different window sizes ranging from 1-48 hours, to investigate if the model performance increases when a larger or smaller dataset is used to fit the linear model. The result is presented in figure 5.3, and clearly shows that the optimum data range to use for a least squares fit of this system is between 10-26 hours. Nevertheless, the mean squared error (MSE) does not improve significantly regardless of window size.

The main drawback of the DMDc is the lack of incorporated memory in the model. As the biochemical process of interest is not only nonlinear but also time-delayed, the DMD/DMDc approach is concluded inadequate for the purpose of estimating the phos-

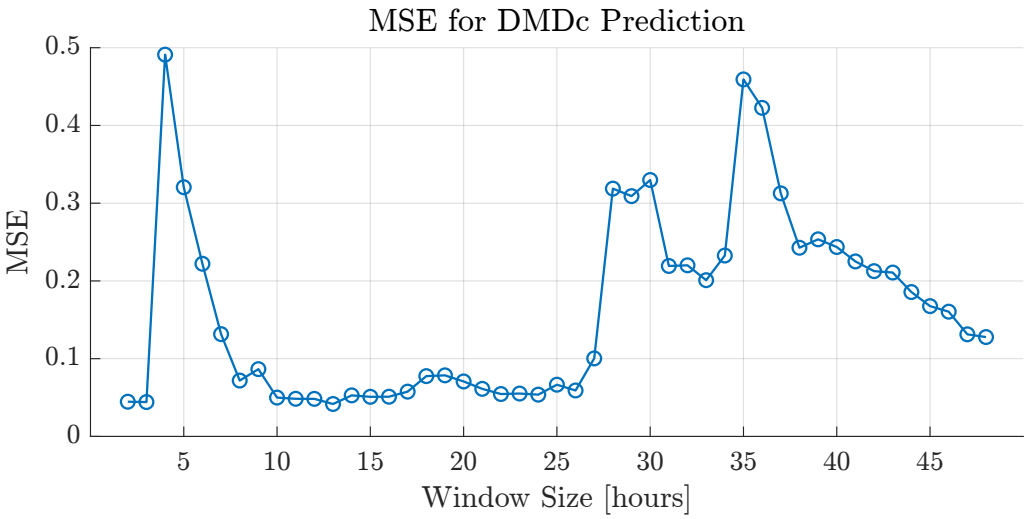

Figure 5.3: Prediction performance for the DMDc algorithm when fittet to different lengths of data (window size) and used to predict 4 hours.

phorus dynamics efficiently and consistently. Hence, in the following chapter, a neural network approach is applied to investigate if the incorporation of memory and nonlinearities such as biases and time delays can improve the prediction performance.

6 Nonlinear Data Driven System Identification

Nonlinear model structures are presented in the form of recurrent neural networks, as the linear model obtained in chapter 5 showed to be inadequate to describe the system dynamics. Furthermore, this chapter explains the basic principles of neural networks and hyperparameters related to the training algorithm and structure of the networks.

6.1 The Artificial Neural Network

Artificial neural networks are parallel distributed processors made of a large number of strongly connected simple units. The most important characteristics of the NNs are that the networks are robust against the failure of single units and they learn from data. The artificial neural network was originally motivated by the biological structures in the human brain, which are very powerful and efficient for tasks such as information processing, learning, and adaptation [1]. Henceforth, artificial neural networks are referred to as *neural networks* or *NNs*. Each unit in the NN performs some calculation and outputs a value which then propagates through the network as the outgoing connections are inputs to other units. Connections between units have weights corresponding to how strong two units are linked. A network structure is often visually illustrated as shown in figure 6.1, and can be presented in the reduced form used in equation (6.1) [18], [24].

$$p - q_1 - \cdots - q_l - r \tag{6.1}$$

Where p is the number of inputs, q_1 is the number of neurons in the first hidden layer, q_l is the number of neurons in the l'th hidden layer and r is the number of outputs.

Due to the biological origin of the neural network, many publications use their own terminology. A translation into system identification and statistics terminology is given in table 6.1 for the most important terms. Furthermore, the input, u, and output, y, refer to states in the network (independent and predicted variables), while the target, T, is the measurement which the network is to predict (observed values).

6.1.1 Deep Feed-forward Neural Networks

The most widely known and used NN structure is the multilayer perceptron (MLP) which is a network where information is passed forward in the model, also known as

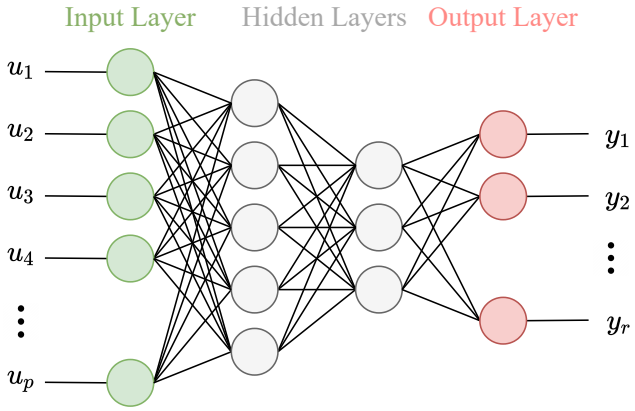

Figure 6.1: Example of a FFNN.

Table 6.1: Translations from machine learning into system identification language. The explanations are partly taken from [1].

Machine learning terminology	System identification terminology
Neural Network	Model
Neuron	Basis function
Weight	Parameter
Bias	Offset
Hidden Layer	Set of basis functions
Input layer and output layer	Set of inputs and set of outputs
Input	Independent variable
Output	Predicted value
Target	Observed value (that is to be predicted)
Learning or training	Estimation or optimization

forward propagation. Hence the term feed-forward neural network (FFNN) is often used to describe a MLP. The network illustrated in figure 6.1 is a FFNN.

The FFNN can be trained using the gradient-based *back-propagation algorithm*, where information is allowed to flow backwards through the network in order to compute the gradient of some predefined cost function with respect to all network weights and biases. For regression problems, this cost function is commonly a performance measure such as mean squared error (MSE) or similar.

6.1.2 The Neuron

A model of a neuron is shown in figure 6.2. This model includes input signals, u_j, weights of neuron i, w_{ij}, an external bias, w_{i0}, the activation function, $\varphi(\cdot)$, and the output, Φ_i. We may describe the output, Φ_i, of a neuron with index i by equations (6.2) and (6.3).

$$v_i = \sum_{j=0}^{p} w_{ij} u_j \tag{6.2}$$

$$\Phi_i = \varphi(v_i) \tag{6.3}$$

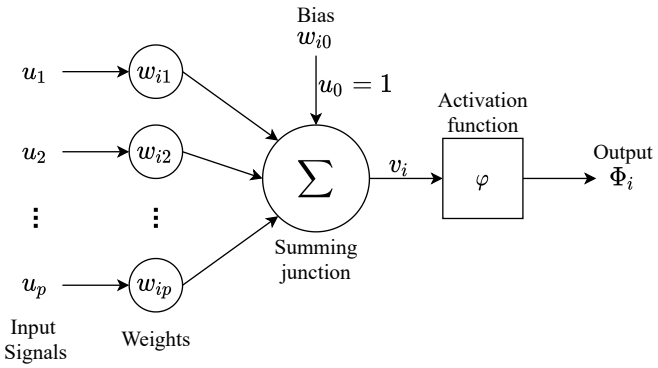

Figure 6.2: Model of an artificial neuron (modified from [51]).

The function $\varphi(\cdot)$ is called the activation function. A variety of activation functions can be applied and is chosen specific to the task. Some of the most commonly used activation functions are shown in figure 6.3 and equations (6.4) to (6.9). If gradient decent is used to train the network, the activation function must be differentiable as the derivative of $\varphi(\cdot)$ is needed.

By using the bias w_{i0}, an affine transformation is applied to the output of the neuron Φ_i, and the activation function shifts on the horizontal axis if the bias is different from zero.

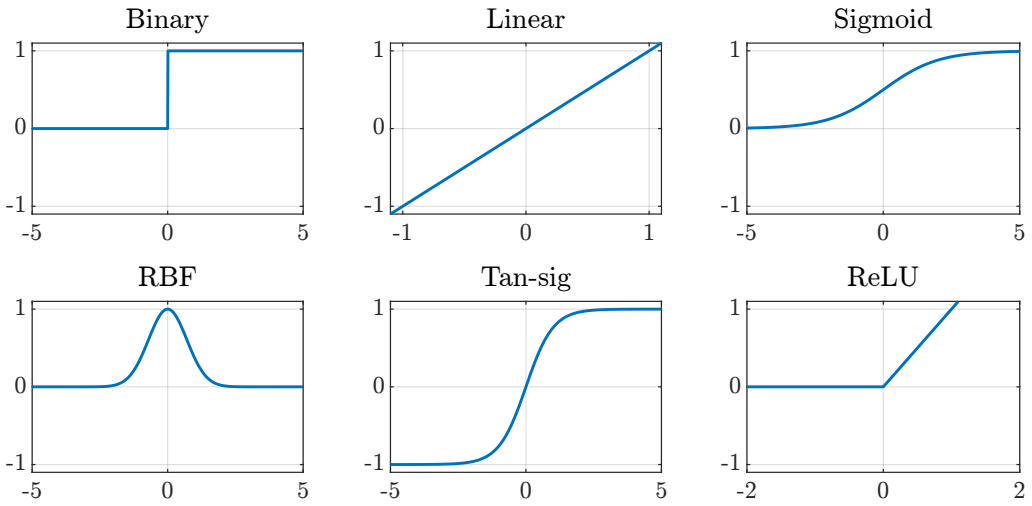

Figure 6.3: Commonly used activation functions in neural network literature.

$$\text{binary}: \Phi_i = \begin{cases} 0 & \text{for } v_i \leq 0 \\ 1 & \text{for } v_i > 0 \end{cases} \tag{6.4}$$

$$\text{Linear}: \Phi_i = v_i \tag{6.5}$$

$$\text{Logistic Sigmoid}: \Phi_i = \sigma(v_i) = \frac{1}{1+\exp(-v_i)} \tag{6.6}$$

$$\text{RBF}: \Phi_i = \exp\left(v_i^2\right) \tag{6.7}$$

$$\text{Tangent-Sigmoid}: \Phi_i = \tanh v_i \tag{6.8}$$

$$\text{ReLU}: \Phi_i = \max(0, v_i) \tag{6.9}$$

Neurons, like the one shown in figure 6.2 are combined in a network, e.g. as shown in figure 6.1, where the network is described by an input layer, hidden layers and an output layer.

6.1.3 Backpropagation

The back-propagation algorithm is a method for gradient calculation of a neural network to determine the weights and biases in the network resulting in the minimal cost function value [52]. The method has its name as it allows the information to flow backwards through the network to compute the gradient of the cost function. For regression tasks, the cost function is usually mean squared error (MSE). In fact, the backpropagation is identical to the application of the well-known chain rule for derivative calculation [1].

The method is described briefly based on [1], [52].

The derivatives of the network output $\hat{y}$ with respect to the i'th output layer weight w_i are:

$$\frac{\partial \hat{y}}{\partial w_i} = \Phi_i \qquad \text{with } \Phi_0 = 1 \tag{6.10}$$

where $i = 0 \cdots q$. For the weight $w_{i,j}$ at the connection between the j'th input and the i'th hidden neuron, the derivatives of the network output with respect to the hidden layer weights w_{ij} are

$$\frac{\partial \hat{y}}{\partial w_{i,j}} = w_i \frac{\mathrm{d}\varphi(v_i)}{\mathrm{d}v_i} u_j \qquad \text{with } u_0 = 1 \tag{6.11}$$

where $i = 1, \cdots, q$ and $j = 0, \cdots, p$. The function $\varphi(\cdot)$ is the activation function. If the activation function is given by equation (6.8), then equation (6.11) becomes equation (6.12).

$$\frac{\partial \hat{y}}{\partial w_{i,j}} = w_i(1 - \Phi_i^2) u_j \qquad \text{with } u_0 = 1 \tag{6.12}$$

A major disadvantage of the back-propagation training of networks is that the gradient becomes very small when propagated back through the layers. This is often explained as the *vanishing gradient*, and has resulted in very slow learning and training for deep networks.

6.2 Dynamic Neural Network

The recurrent neural network (RNN) is a special variant of network capable of processing sequential data such as time series. The RNNs have feedback loops, which allows previous inputs and outputs to affect the current output. Elman [53] and Jordan [54] proposed the first versions of the RNN, which applies feedback from the hidden and output layer, respectively. Contrary to the FFNN, which gives a static approximation of a system, the RNN has memory, and is hence also known as dynamic or memory neural networks. The differences between the static feed-forward network and dynamic networks are stated below and also visualized in figure 6.4.

Static FFNN The response of a static feed-forward network (FFNN) lasts as long as the input. The network has no memory, so the output of the network at a time point depends only on the value of the input at that same time point.

Dynamic FFNN The response of a dynamic FFNN lasts longer than the input. This network has memory, and the output at a given point in time depends not only on the input at that time point, but also previous inputs. The history of inputs is created with tapped delay lines on the input signals (shown as q^-1), and the amount of memory in the network is limited by the number of tapped delays. Linear dynamic FFNNs are commonly called finite impulse response (FIR).

Dynamic RNN The output of a dynamic recurrent network lasts longer than that of the Static FFNN and Dynamic FFNN . The output of this network depends on the history of inputs and outputs, meaning an infinite amount of history will affect the response. Linear dynamic RNNs are commonly called infinite impulse response (IIR).

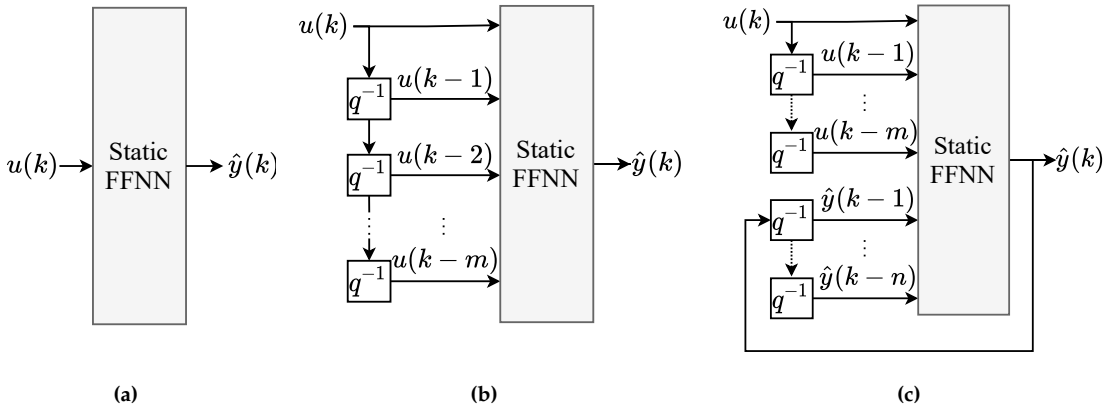

Figure 6.4: Types of neural networks used for system identification. (a) static, (b) feedforward-dynamic, and (c) Jordan recurrent-dynamic network.

Nowadays, the classical Elman and Jordan networks are often referred to as vanilla recurrent neural networks (vRNN), as they are of simple structure and inefficiently trained when the sequence is long [55]. Hence, many dynamic NNs has emerged from the

vRNNs, improving performance by adopting logic gates, attention mechanisms, internal memory or external memory [55].

6.2.1 NARX Neural Network

In the previous work by the author [13], a vanilla RNN was obtained in the form of a nonlinear autoregressive model with exogenous input (NARX) to forecast phosphorus concentrations 1 step ahead. The network was trained as an open-loop NARX as shown in figure 6.5b, where the previous outputs passed to the input layer is actual target values (measurements in this case). However, when the designed network is closed-looped it is a shallow neural network structure identical to the Jordan network shown in figure 6.5a.

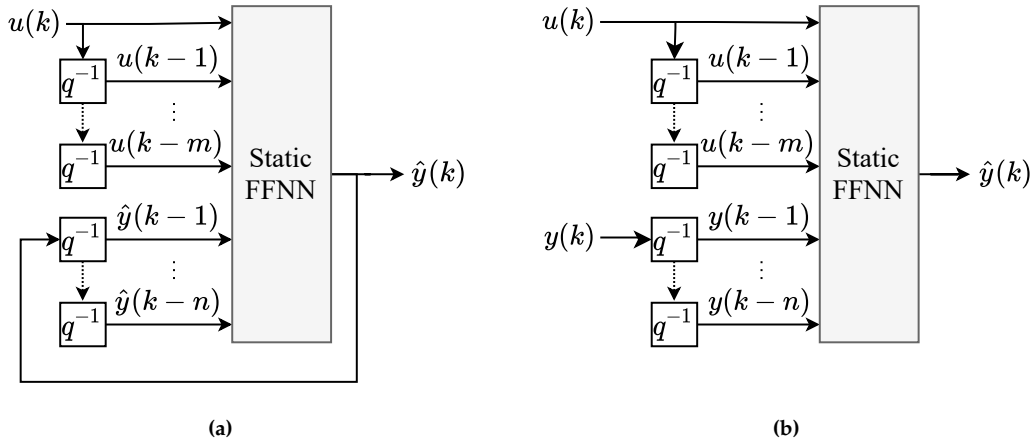

Figure 6.5: The nonlinear autoregressive network with exogenous input in (a) closed-loop form and (b) in open-loop form.

The dynamical behaviour of a system modelled by a multiple input-multiple output NARX network with p inputs is described in equation (6.13).

$$\begin{cases} \mathbf{y}(t) & = f\big[\mathbf{x}(t)\big] \\ \mathbf{x}(t) & = \big[y_1(t-1),\ \cdots,\ y_1(t-m)\cdots y_r(t-1),\ \cdots,\ y_r(t-m) \\ & \quad u_1(t),\ u_1(t-1),\ \cdots,\ u_1(t-n)\cdots u_p(t),\ u_p(t-1),\ \cdots,\ u_p(t-n)\big] \end{cases} \tag{6.13}$$

Where $f[\cdot]$ is a nonlinear mapping performed by a FFNN. $\mathbf{y} = [y_1 \cdots y_r]$ are dependent variables (outputs), $\mathbf{u} = [u_1 \cdots u_p]$ are the independent variables (inputs). The input and output order are determined by n and m, respectively. m is the number of feedback delays of the output and n is the number of input delays. This NARX model has the following tuning parameters:

- Number of previous inputs in the input layer, $\mathbf{u}(t) \cdots \mathbf{u}(t-n)$
- Number of previous outputs in the input layer, $\mathbf{y}(t-1) \cdots \mathbf{y}(t-m)$
- Number of units in the hidden layer, p

Those parameters were determined in [13] and are summarized here.

Selecting the memory-order, m and n, is analogous to that of choosing an optimal subset of regressor variables in statistical modelling [39]. The input and output order were determined through autocorrelation analysis of the output and cross correlation analysis of the inputs and outputs. The input order were determined to be $n = 50$ and output order to $m = 20$. The number of units in the hidden layer were determined using a systematic approach where the performance were evaluated over 5 experimental tests and the optimum number of units were determined based on the average R^2 and mean squared error (MSE). Generally, selecting the number of hidden units is a difficult task as there exists no guidelines and the optimum number is highly task depend.

The network is trained on a subset of the dataset and results are presented in chapter 7.

6.3 Long Short-term Memory Networks

Recurrent neural networks are trained using the backpropagation through time, where the network is unfolded for many time steps. However, a problem that arises from this method is that the gradient of the weights starts to become too small or too large, which is referred to as the vanishing and exploding gradient [49], [56]. Neural networks composed of only multiplicative updates (like the the RNN and FFNN) are good at learning over short sequences, and is said to have good short-term memory, but poor long-term memory [49]. A method to work around the problems associated with vanishing and exploding gradients and to apply some long term memory to the network, is by implementing computing units composed by gates in the hidden layer. The computing units "traps" the error in the block, forming a so-called "error carousel". The long short-term memory neural network (LSTM) avoids the vanishing gradient problem by using three gated units; the forget gate, input gate and output gate, through which the memory of the network can be efficiently controlled [57]. As a result, almost all exciting results regarding RNNs for sequence modeling have been achieved by LSTMs, leading the LSTMs to become a focus of deep learning [58].

LSTMs are designed to learn long-term dependencies and are widely used for tasks such as speech recognition, natural language processing and other pattern recognition applications. Over the last 5-10 years, the LSTM has also been applied for system identification [16], [57], [59], where it has shown to outperform more established methods like the vRNN and FFNN. The mathematical formulation of the LSTM cell at time t with input $\mathrm{u_t}$ and hidden states $\mathrm{h_t}$ is given in equations (6.14) to (6.19).

$$f_t = \sigma_g(W_{fh}\mathrm{h_{t-1}} + \mathrm{W_{fu}u_t} + \mathrm{b_f}) \tag{6.14}$$

$$i_t = \sigma_g(W_{ih}\mathrm{h_{t-1}} + \mathrm{W_{iu}u_t} + \mathrm{b_i}) \tag{6.15}$$

$$g_t = \tanh(W_{gh}\mathrm{h_{t-1}} + \mathrm{W_{gu}u_t} + \mathrm{b_g}) \tag{6.16}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \tag{6.17}$$

$$o_t = \sigma_g(W_{oh}\mathrm{h_{t-1}} + \mathrm{W_{ou}u_t} + \mathrm{b_o}) \tag{6.18}$$

$$\mathrm{h_t} = o_t \circ \tanh c_t \tag{6.19}$$

The matrix W_{qp} contains the weights of the input and recurrent connections where the subscript $_q$ can either be the forget gate f, input gate i, output gate o or cell update g, and the subscript $_p$ denotes the hidden state (recurrent information) h or the input u. The bias is described by b_q, the cell state by c and the operator $\circ$ denotes the element-wise multiplication of two vectors. σ_g is a sigmoid function. Figure 6.6 presents the inner connections of an LSTM cell.

When the cell state is updated the three gates decides the following: The forget gate decides what information will be disposed from the cell state. If $f_t = 1$ the information is kept, and if $f_t = 0$ the information is thrown away. The input gate decides what information to store in the cell state, and the output gate decides what information can be output based on the cell state.

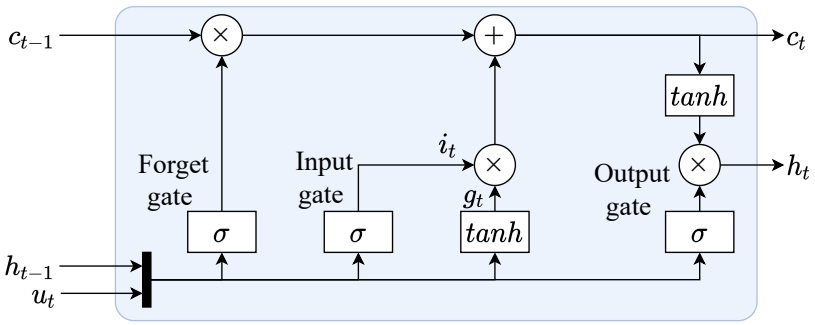

Figure 6.6: The LSTM cell architecture.

Computer science notation uses c_t to describe the internal state of the system contrary to controls notation where x_t usually is used. Nevertheless, the LSTM structure can be expressed in controls notation as this: The key idea of the LSTM is to have a memory cell in each neuron with a state, c_t, which allows to model first order dynamics as described by equation (6.20) [1].

$$c_t = f_t c_{t-1} \tag{6.20}$$

The decisive part is the forget gate which controls the pole of the first order system. This is illustrated in figure 6.7. As f_t is the output of a sigmoid function, it will take a value in the interval [0,1], resulting in only stable first-order dynamics or marginally stable dynamics for $f_t \approx 1$, i.e., integral behavior [1].

Similar to all other neural networks, the LSTM network can be constructed as deep neural networks, meaning that several LSTM layers can be stacked and connected internally. This yields two structure-related tuning parameters for the LSTM:

- Number of hidden layers
- Number of units in each layer

Additionally, in order to train the model, several hyperparameters must be selected as well. Hyperparameters are parameters whose value is used to control the training process. A few hyperparameters are described and discussed in the following section 6.4. In

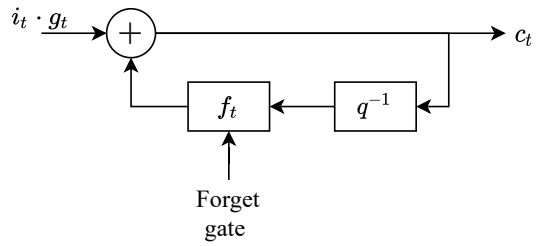

Figure 6.7: Internal dynamics of the LSTM cell which corresponds to a first order filter (figure modified from [1]).

chapter 7, a LSTM architecture and the training process will be presented and evaluated where the network will be trained on a subset of the dataset.

6.4 Hyperparameter Tuning

Most deep learning algorithms have several hyperparameters that affects many aspects of the algorithm's behavior. Some hyperparameters affect the need for computational power, time or memory cost of running the algorithm. Other hyperparameters affect the performance of the recovered model and its ability to predict accurate results when deployed on new data. Neural networks have between 10-50 hyperparameters, depending on how the model is parameterized and how many parameters the developer has chosen to fix at a default value [45]. In any case, the task of tuning the hyperparameters can be time-consuming and difficult, if done manually, as it requires understanding of how each hyperparameter affects the algorithms and how NN models achieve good generalization [37]. There is therefor great appeal for automatic tuning of hyperparameters, which in many cases is preferred if the computational power is available.

Automatic hyperparameter selection reduces the need for expert knowledge and rules of thumb, but they often require much more computational power [37]. Henceforth the term *hyperparameter optimization* is used to describe the automatic tuning of hyperparameters . Several methods exists for automatic tuning of hyperparameters; grid search [45], random search [46] and model based optimization such as Bayesian optimization [47] are some of the commonly used methods. Common for all the automatic hyperparameter tuning strategies is that they wrap an optimization algorithm around the problem, hence hides the hyperparameters for the developer.

In this work, Bayesian optimization is applied to identify the best hyperparameters of an LSTM network, as it has shown great performance in research studies [47], [60]. Bayesian optimization is especially advantageous for problems where function evaluation is expensive (high duration), not easily differentiable or expressive.

There are many hyperparameters which can be included to optimize the training and performance of the recovered model. A thorough evaluation of different hyperparameters of the LSTM is presented in [61], where several LSTM networks for sequence labeling tasks are presented. Contrary to many other studies where the focus lies on identifying the specific configuration that performs best on the given task, this study ([61]) focus on

finding design choices that perform robustly. This means that the results are not task specific, but yields good performance when the architecture is applied to new tasks or new domains. This work will focus on four of the most important parameters [37], [61];

- Learning rate
- Number of layers
- Number of hidden units
- Weight decay coefficient

Other important hyperparameters includes mini-batch size, dropout regularization, choice of solver, but these will be set to default values in this work. Different Hyperparameters are described in the following sections.

6.4.1 Solver

The solver (or optimizer) is responsible of minimizing the cost function during training of the neural network. A common choice of optimizer is the stochastic gradient descent (SGD), which is a stochastic version of the backpropagation. However, as SGD can be sensitive to the learning rate [61], several other optimization algorithms has been proposed, including Adam (adaptive moment estimation) [62] and RMSProp (root mean square propagation). Both Adam and RMSProp usually outperforms SGD and in [61] it was concluded that the two optimizers produced more stable and better results than SGD. In this work, Adam is selected as the solver and hence not included in the hyperparameter optimization.

6.4.2 Dropout Regularization

Dropout is a powerful regularization strategy where new inputs are constructed by multiplication by noise [63]. The strategy is computationally inexpensive, yet very powerful and prevents overfitting. The key idea of dropout regularization is to train the ensemble of sub-networks that can be formed by removing non-output units from the original *base* network. In this work, dropout is applied with a default dropout probability value, and hence not included in the hyperparameter optimization.

6.4.3 Mini-batch Size

When training the neural network, the training data can be divided into mini-batches. The gradient of the loss function is evaluated on a single mini-batch at a time, meaning that an iteration corresponds to a mini-batch. The full pass of the training algorithm over the entire training set using mini-batches is one epoch. Weights and biases are updated each iteration. Hence, the mini-batch size heavily affects the duration and performance of the training process. The mini-batch size is given default value in this work, and hence not included as a tuning parameter in the hyperparameter optimization. However, some manual tuning has been performed, and it is described when relevant in chapter 7.

6.4.4 Number of Hidden Layers and Hidden Units

The number of layers and number of hidden units in a layer are two hyperparameters that affects both the training process and the model capacity. The term capacity describes a model's ability to fit a wide variety of functions. Hence, a model with low capacity struggles to fit to the training data while a model with high capacity may overfit and memorize properties of the training data. The number of layers and units are two parameters where overfitting occurs when the value of the hyperparameter is large. This is because increasing the number of hidden units or layers increases the capacity of the model. High model capacity does not necessarily result in high model performance, if the training algorithm cannot discover the required functions. The number of hidden layers and units in each layers can be viewed as the depth and width of the network, respectively. In this work, both the number of hidden layers and number of hidden units in each layer are included in the Bayesian optimization algorithm to select hyperparameters of the LSTM network. The depth is limited to maximum 4 stacked LSTM layers while the width is optimized within the interval of 50-200 units in each hidden layer. Results are presented in section 7.3.

6.4.5 Learning Rate

Goodfellow, Bengio, and Courville states in [37]; "If you have time to tune only one hyperparameter, tune the learning rate". The learning rate is perhaps the most important hyperparameter as it may apply constraints to the training performance, if not chosen correctly. A learning rate chosen too low results in slow training and may cause the algorithm to get stuck with high training error. On the other hand, an excessive learning rate may cause the training error to increase rather than decrease. A learning rate which is far from the optimum will, whether too high or too low, result in low model capacity due to optimization failure [37]. The Learning rate is included in the Bayesian optimization algorithm to select hyperparameters of the LSTM network, and results are presented in section 7.3.

An advantage of tuning the learning rate, is that it requires monitoring of only the training error, compared to other hyperparameters where validation error must be monitored as well to diagnose overfitting or underfitting.

6.4.6 Weight Decay

Weight decay adds a regularization term for the weights to the cost function, meaning that a weight decay coefficient, λ, larger than 0 penalizes high weight values. Using L_2 regularization, the cost function $E()$ is updated to $E_R()$ in equation (6.21).

$$E_R(\theta) = E(\theta) + \lambda\Omega(W) \tag{6.21}$$

Where W is the weight matrix, λ is the regularization factor (weight decay coefficient), and the regularization function $\Omega(W)$ is given by equation (6.22).

$$\omega(W) = \frac{1}{2} W W^T \tag{6.22}$$

Regularization reduces overfitting and forces the training algorithm to put weight on fewer of the features.

Weight decay in the form of L_2 regularization is included as a tuning parameter in section 7.3.

7 Results

This chapter presents the trained networks and optimization of hyperparameters described in chapter 6. The software used for the application is MATLAB, where the Deep Learning Toolbox [64] is used to train and validate the models while the Statistics and Machine Learning Toolbox [65] is used for the Bayesian Optimization of the hyperparameters.

The entire dataset is divided into three subsets of data; *training data*, *validation data* and *test data*. During training of the network, training data is used to optimize and update weights and biases each iteration while the validation data is used to select the best parameters in order to avoid overfitting the model to the training. Hence, the validation dataset is not used for parameter optimization, but solely to achieve generalizability of the model. If the recovered models shows great performance on new data which is previously unseen, the model is generalizable. Hence, the test dataset is excluded from the training process and used to evaluate the model performance on new data. For legibility, table 7.1 defines some of the technical terms used throughout this chapter to describe the training procedure of the networks.

Table 7.1: Neural network training terminology.

Field	Description
Epoch	An epoch corresponds to a full pass of the training data.
Iteration	Iteration number. An iteration corresponds to a mini-batch.
Training duration	Time elapsed in hours/minutes/seconds to execute the training algorithm.

Common for all the designed networks is that there are 25 inputs (the variables shown in figure 4.4 and explained in table 4.2) and a single output (P_{bio}).

All networks are compared based on a set of statistical measures, which are listed in equations (7.1) to (7.8). The used statistics are coefficient of determination (R^2), mean squared error (MSE), root mean squared error (RMSE), normalized root mean squared error (NRMSE), mean error (ME), standard error (SE), linear correlation coefficient (ρ_P) and rank correlation coefficient (ρ_S).

$$R^2 = 1 - \frac{\sum_{i=1}^{k}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{k}(y_i - \bar{y}_i)^2} \tag{7.1}$$

$$MSE = \frac{1}{k}\sum_{i=1}^{k}(y_i - \hat{y}_i)^2 \tag{7.2}$$

$$RMSE = \sqrt{MSE} \tag{7.3}$$

$$NRMSE = \frac{RMSE}{\bar{y}} \tag{7.4}$$

$$ME = \frac{1}{k}\sum_{i=1}^{k}(y_i - \hat{y}_i) \tag{7.5}$$

$$SE = \sqrt{\frac{(y_i - \bar{y}_i)^2}{k}} \tag{7.6}$$

$$\rho_P = \frac{\sum_{i=1}^{k}(y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})}{\sqrt{\sum_{i=1}^{k}(y_i - \bar{y})^2 \sum_{i=1}^{k}(\hat{y}_i - \bar{\hat{y}})^2}} \tag{7.7}$$

$$\rho_S = 1 - \frac{6\sum_{i=1}^{k} d_i^2}{k(k^2 - 1)} \tag{7.8}$$

Where the linear and rank correlation coefficients, ρ_P and ρ_S, are computed as Pearson's linear correlation coefficient and Spearman's rho, respectively. In equations (7.1) to (7.8), y_i is the observed value and $\hat{y}_i$ is the predicted value at time i of a time series with k time steps. A bar accent, $\bar{\cdot}$, denotes the mean value of a time series. The statistical measures are throughout this chapter presented in figures when relevant and subsequently summarized in tables.

Furthermore, the cross-correlograms of the target values and predicted outputs, $T \star Y$, are used to compare the transient dynamics of the actual system and model. The cross correlation is a measure of similarity of the two time series, and a high correlation at lag 0 indicates that the model neither leads or lags the time series of the observed values.

7.1 NARX Neural Network

The NARX network is constructed of three layers; an input layer where the memory order of the inputs is $n = 50$ and memory order of the output is $m = 20$, a hidden layer with 4 units and an output layer with a single output. The network is trained and evaluated on a subset of the data, corresponding to 44 days. This is due to the computational power required to train a massive vRNN, and using more data resulted in very high training duration (several days using a GPU). Predicted and observed values are compared in figures 7.1 to 7.3, where smaller sections of the training, validation and test data are shown. A figure showing the entire dataset is included in appendix A.1. The partitioning of data is performed as follows: 70% training data ($\approx$ 31 days), 15 % validation data ($\approx$ 6.5 days), 15% test data.

Figure 7.1 shows a comparison of the predicted output and target for a section of the training data. As expected when evaluated on the training dataset, the model is able

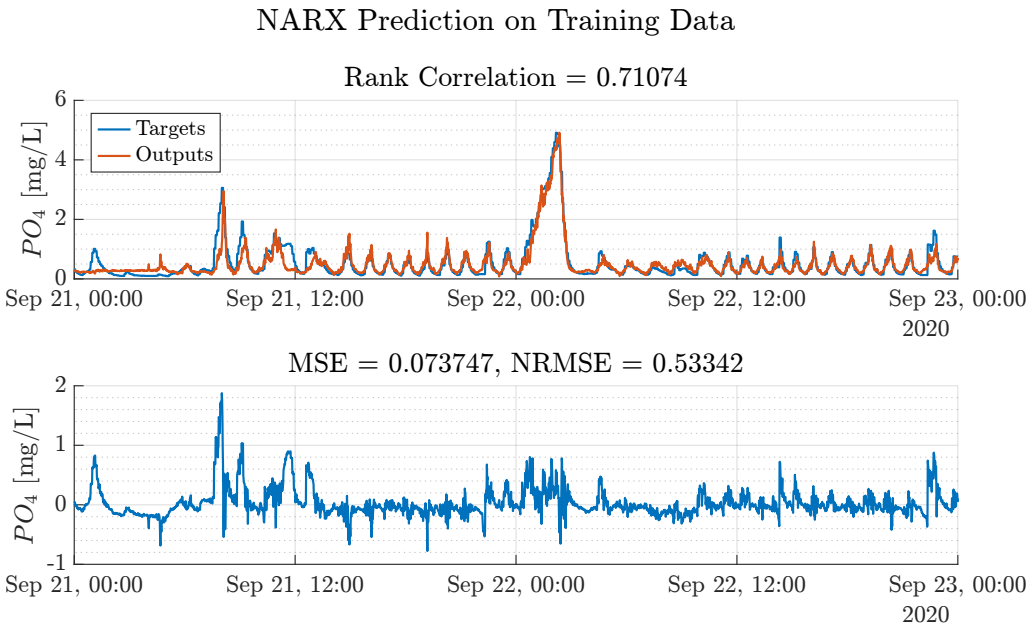

Figure 7.1: Prediction of phosphorus concentration using the NARX network on training data.

to predict the target values good with a rank correlation of $\rho_S = 0.71$ over the entire training dataset. Similarly, the model predicts the validation data with low MSE and NRMSE (see figure 7.2), however a little higher than that for the training dataset. The rank correlation has also experienced a drop, presumably due to incorrect predictions for high phosphorus concentrations.

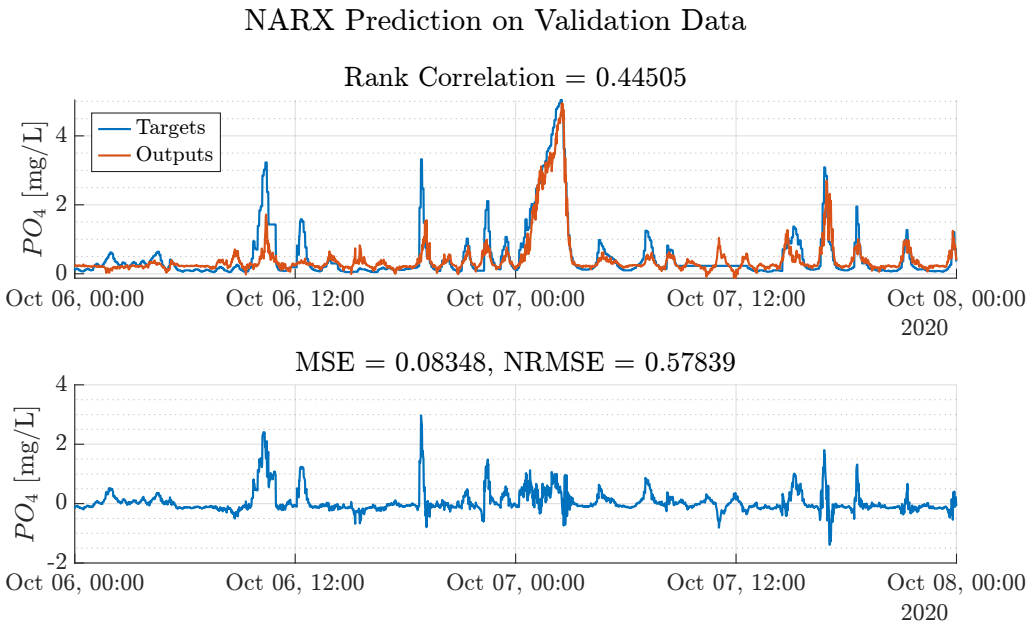

Figure 7.2: Prediction of phosphorus concentration using the NARX network on validation data.

The NARX network is used to predict new data and the result is shown in figure 7.3. The period shown in the figure illustrates how the model at some times fails to predict an increase in phosphorus concentration. However, the rank correlation is close to that of

the validation data, indicating that the model seem to capture the dynamics, even though the magnitude is incorrectly predicted.

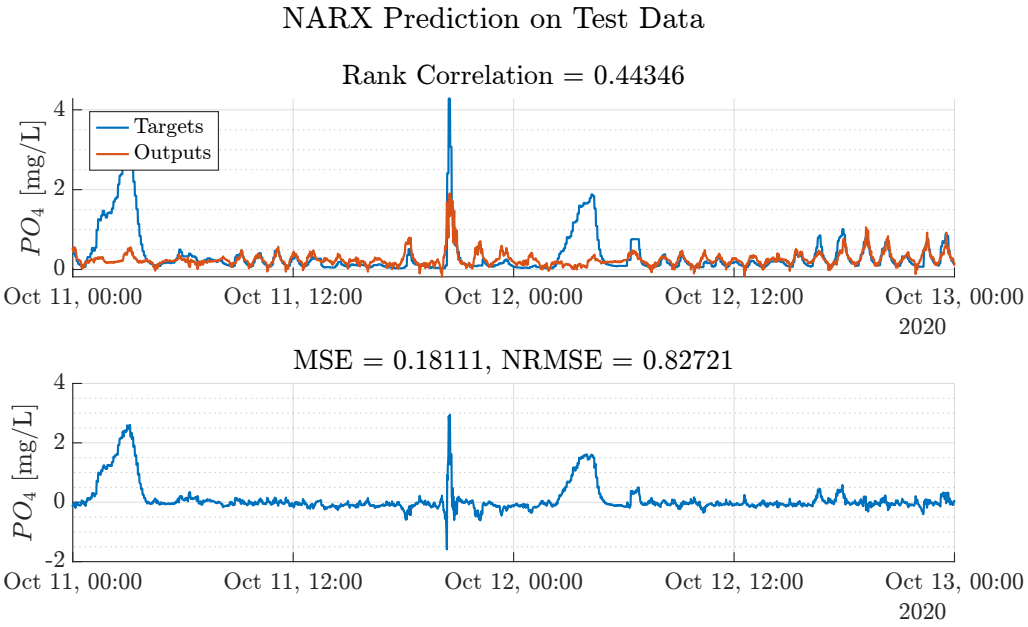

Figure 7.3: Prediction of phosphorus concentration using the NARX network on test data.

The scatter plots shown in figure 7.4 indeed shows that the model struggles to estimate phosphorus concentrations above 1.5-2 [mg/L], as the data points mostly lie under the line $Y = T$.

Table 7.2: Performance measures for the NARX on training, validation and test data.

	Training	Validation	Test
R^2	0.72	0.67	0.32
MSE	0.074	0.083	0.18
RMSE	0.27	0.29	0.43
NRMSE	0.53	0.58	0.83
ME	0.00025	-0.0082	-0.013
SE	0.27	0.29	0.43
lin corr	0.85	0.82	0.56
rank corr	0.71	0.45	0.44

A reasonable explanation for this is that the training and validation dataset does not contain many periods with high concentrations compared to periods with low P-concentration. The phosphorus concentration is usually high when BPR is utilized, and the model struggles to consistently predict the concentration in those periods (see the periods 00:00 - 06:00 in figure 7.3).

Furthermore, as shown in figure 7.5, the cross-correlation is highest at lag -1, indicating that the model lags the actual system with a single time step (1 min).

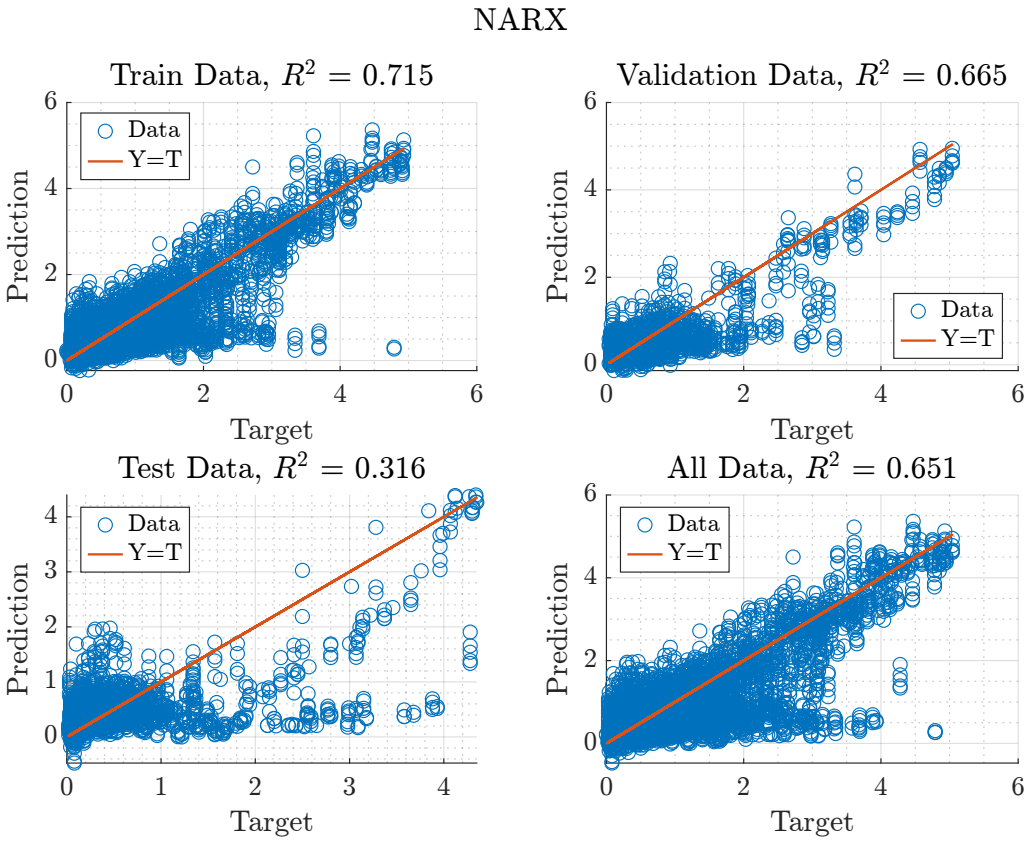

Figure 7.4: Scatter plots of the output value, Y, relative to the target value, T, at the same point in time.

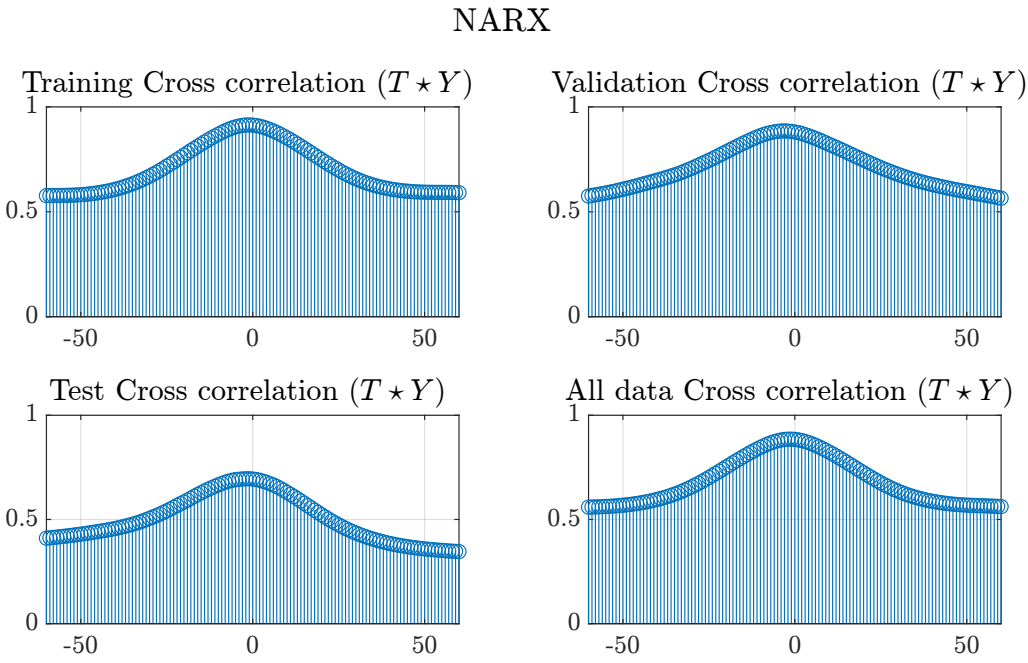

Figure 7.5: Cross correlogram for the different datasets.

All performance statistics (equations (7.1) to (7.8)) are listed in table 7.9 in the end of this chapter to compare all obtained models.

7.2 LSTM

In section 6.3, the structure of an LSTM neural network was described. Using MATLAB default options and a some manual tuning of hyperparameters, an LSTM network is constructed and evaluated in this section.

The LSTM network is constructed of three layers; an input layer, a hidden layer with 100 units and an output layer with a single output. Through the tuning process, the number of hidden units and learning rate of the training algorithm was the only tuned hyperparameters. The values of the significant hyperparameters (described in section 6.4) are displayed in table 7.3

Table 7.3: Hyperparameter values used to train the LSTM network. The * indicates that MATLAB default values are used.

Hyperparameter	Value/setup
Solver	Adam
L_2 regularization coef.	0.0001 *
Mini-batch size	1000
Dropout Probability	0.5 *
Learning Rate	0.005
Hidden Layers	1
Hidden units in each hidden layer	100

The network is trained and evaluated on a the entire dataset. Predicted and observed values are compared in figures 7.6 to 7.8, where smaller sections of the training, validation and test data are shown. A figure showing the entire dataset is included in appendix A.2. The partitioning of data is performed as follows: 80% training data, 10 % validation data, 10% test data. The data partitioning is different from that used in section 7.1, since 10 % of the entire dataset corresponds to 20 days, leaving plenty of test data while contributing to better generalizability by including more training data.

Figure 7.6 shows a comparison of the predicted output and target for a section of the training data. It is expected that the model predicts well on the training dataset, however, that is not the case here as the model in general fails to predict both low and high concentration levels. The performance measures are similar to that of the NARX testing performance with rank correlation $\rho_S = 0.47$, MSE=0.25 and NRMSE=0.80. The statistical performance measures of the LSTM are presented in table 7.4.

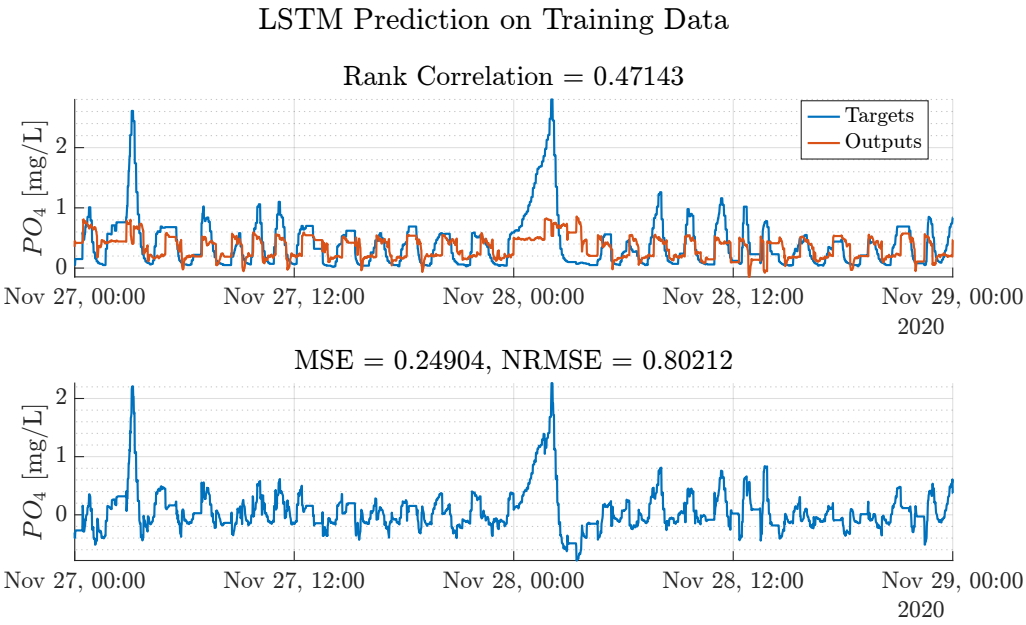

Figure 7.6: Prediction of phosphorus concentration using the LSTM network on training data.

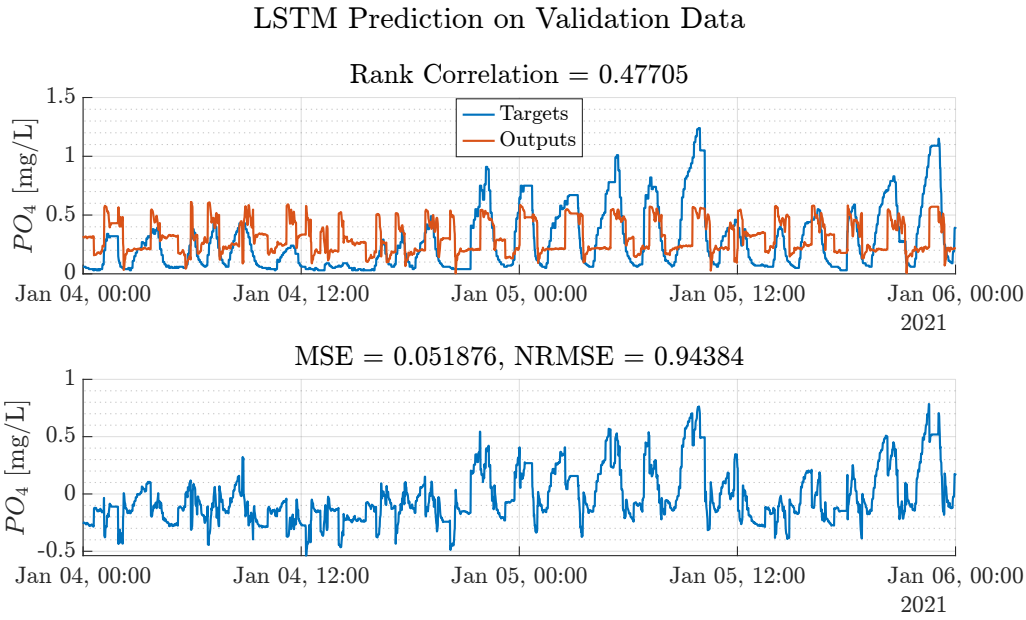

Figure 7.7: Prediction of phosphorus concentration using the LSTM network on validation data.

In figure figure 7.7, the model prediction is compared to target values. The model predicts the validation data, with lower MSE and higher NRMSE, compared to the predictions of the training data. A decreased MSE could indicate better model performance, however, as the average concentration over the validation dataset is lower than over the training data, the MSE statistic alone is misleading. The NRMSE facilitates comparison between datasets with different scales, and this is especially useful in the just mentioned instance. A NRMSE = 1 corresponds to a prediction equally powerful as if the estimator $\hat{y}_i = \bar{y}$ was used.

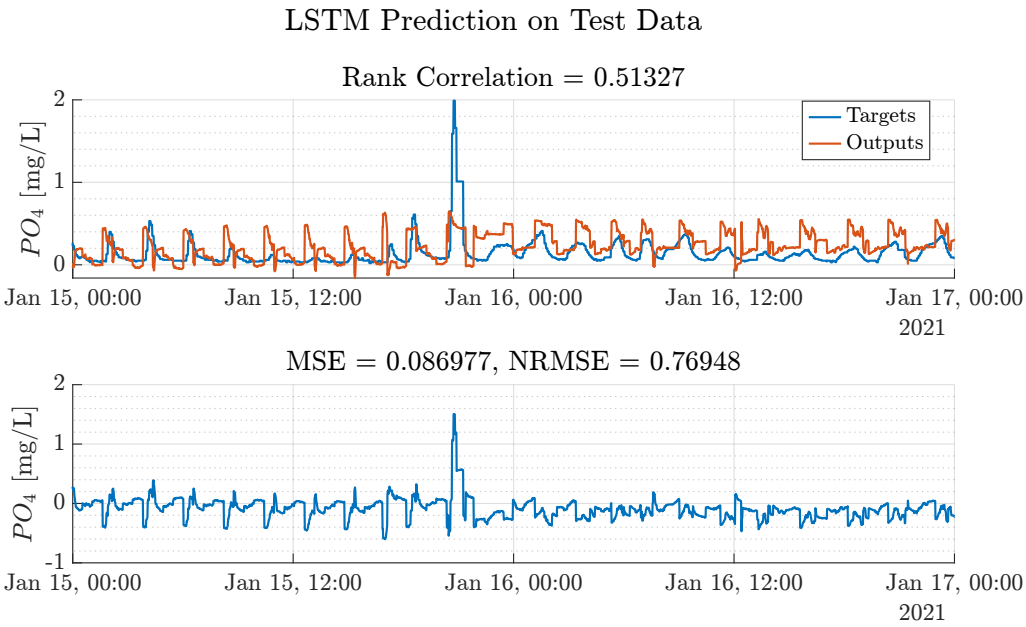

Figure 7.8: Prediction of phosphorus concentration using the LSTM network on test data.

The LSTM network is used to predict new data and the result is shown in figure 7.8. The period shown in the figure illustrates how the model generally fails to predict both low and high concentrations. However, as was the case with validation data, the rank correlation is close to that of the training data, indicating that the model is not overfittet during the training.

Table 7.4: Performance measures for the LSTM on training, validation and test data.

	Training	Validation	Test
R^2	0.35	0.11	0.41
MSE	0.25	0.052	0.087
RMSE	0.5	0.23	0.29
NRMSE	0.8	0.94	0.77
ME	0.054	-0.051	-0.11
SE	0.5	0.22	0.27
ρ_P	0.6	0.49	0.71
ρ_S	0.47	0.48	0.51

The scatter plots presented in figure 7.9 shows that the model, regardless of the dataset used to evaluate it, only has two regions in which it can predict the phosphorus concentration. The two regions are around 0-1.5 [mg/L] and 4 [mg/L]. The empty "gab" in between the two regions may indicate that the capacity is too low to match the complexity of the task.

The cross-correlation between the target and predicted output is shown in figure 7.10. The figure shows that the cross-correlation is highest at lag 0, -1 and +6 for the training,

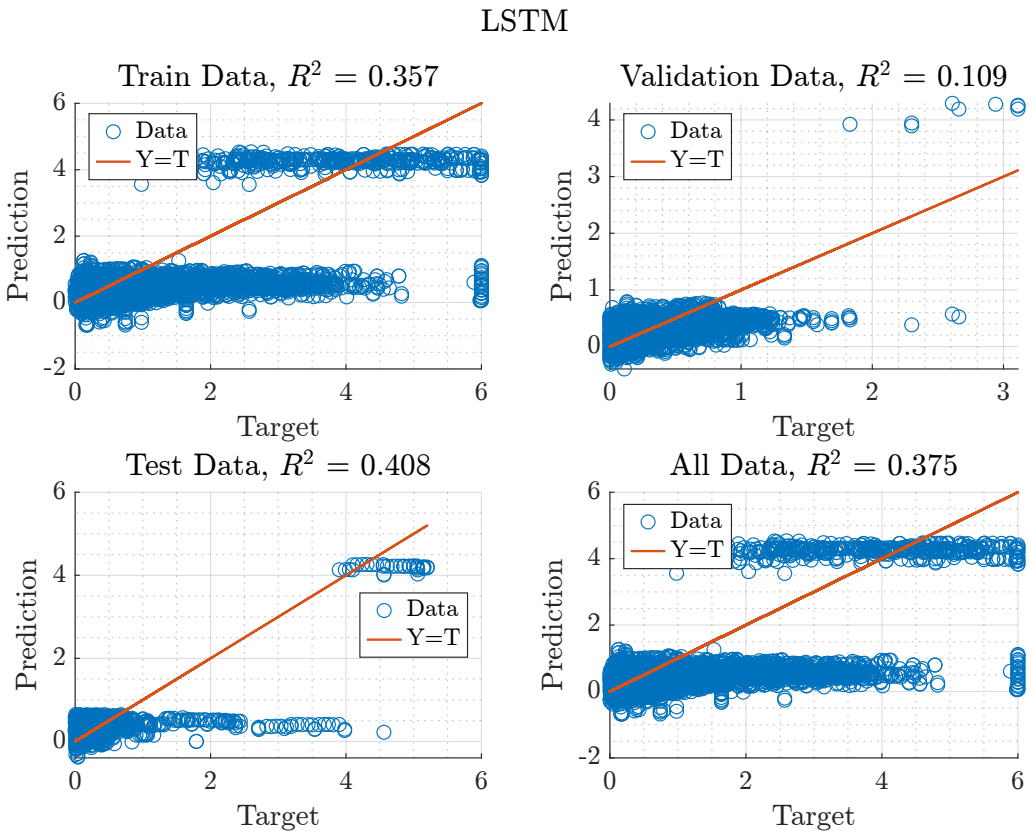

Figure 7.9: Scatter plots of the output value, Y, relative to the target value, T, at the same point in time.

validation and test data, respectively. As the model performance in general is relatively poor, the lack of consistency in cross-correlation across the different datasets is expected.

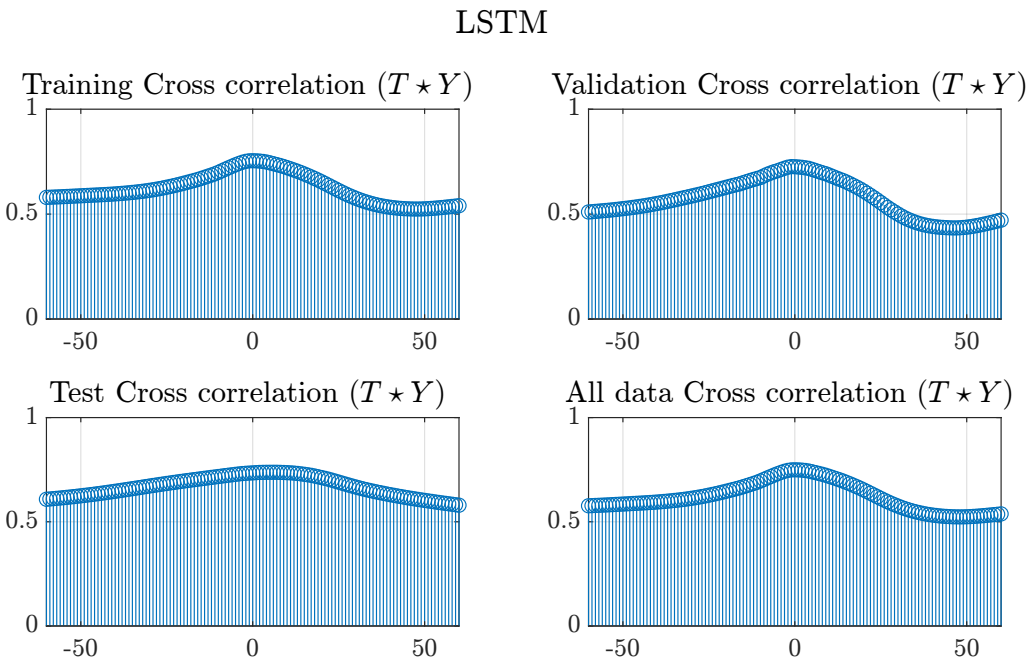

Figure 7.10: Cross correlograms for the different datasets.

As mentioned earlier, the evaluation of the LSTM model indicates that the model requires more capacity in order to truly capture the complexity of the system. The following adjustments to hyperparameters can be used to increase the model capacity:

- Increase the number of hidden layers
- Increase the number of hidden units in each hidden layer
- Use the correct learning rate
- Relaxing the regularization (dropout and weight decay

Unfortunately, tuning the hyperparameters is often a "black art", and changing one parameter may affect the optimum value of another. Hence, automatic hyperparameter tuning using Bayesian optimization is utilized to increse the LSTM model performance.

7.3 Bayesian Hyperparameter Optimization

As the LSTM model with manually tuned hyperparameters showed poor prediction performance compared to the NARX model, Bayesian hyperparameter tuning is applied to obtain an optimum LSTM structure with a capacity that matches the complexity of the task. Initially, the hyperparameters which are to be tuned must be picked, and the appertaining ranges within the optimum values are to be found. As described in section 6.4, only the learning rate, weight decay and depth and width of the network is included in the optimization algorithm. Consequently, Adam is chosen as the solver, the dropout probability is set to 0.5 and mini-batch size is set to 1000 samples. The variables which are optimized are described in table 7.5 with specified ranges to search in.

The framework of the machine learning and statistics toolbox [65] is used to perform the Bayesian optimization on the LSTM structure using the function *bayesopt*.

Table 7.5: Hyperparameters that are included in the optimization algorithm and the corresponding search bounds.

Hyperparameter	Optimization range
L_2 regularization coef.	$[1 \cdot 10^{-10} \quad 0.01]$
Learning rate	$[0.01 \quad 1]$
Hidden Layers	$[1 \quad 4]$
Hidden units in each hidden layer	$[50 \quad 200]$

Bayesian optimization attempts to minimize an objective function, $f(x)$ in a bounded domain for x. In the case of hyperparameter optimization of the LSTM, x contains the parameters described in table 7.5. There are three key elements to Bayesian optimization:

1. A Gaussian process (GP) model of the objective function.
2. A Bayesian update procedure at each new evaluation of the objective function.
3. An acquisition function, $a(x)$, that determines the next point x_{next} to be evaluated as $x_{next} = \text{argmax}_{x \in \mathcal{X}} \, a(x)$.

It is assumed that the objective function values is drawn from a Gaussian process, hence that the observations $\{x_n, y_n\}_{n=1}^{N}$ are of the form:

$$y_n \sim \mathcal{N}(f(x_n), \nu) \tag{7.9}$$

In which ν is the variance of the noise introduced into the observations.
The algorithm is then executed as follows:

1. Evaluate $y_i = f(x_i)$ for a random point x_i taken within the hyperparameter bounds.
2. Update the Gaussian process model of $f(x)$ to obtain a posterior over functions.
3. Find the new point that maximizes the acquisition function, $a(x)$

Step 2 and 3 are repeated until a specified stopping criterion, such as fixed number of iterations or a fixed time is reached.

There are several popular choices of acquisition function, one being the *expected improvement* (EI), given in equation (7.10)

$$a_{EI} = \mathrm{E}(\max(f(x) - f(x_{best}), 0)) \tag{7.10}$$

in which $x_{best} = \mathrm{argmin}_{x_n} f(x_n)$ denotes the best current point.

The MATLAB function *bayesopt* uses the *expected-improvement-per-second-plus* acquisition function, which originates from equation (7.10) but additionally includes time-weighting (adding the per-second term) and overexploiting modifications (adding the plus term). Hence this acquisition function determines x_{next}, not only based on the estimated improvement, but also the mean of the GP model of the objective function evaluation time and an algorithm that detects overexploiting in an area.

The objective function is defined to take the following steps for each iteration, k;

1. Take the values of the hyperparameters as inputs. The *bayesopt* function calls the objective function with the current values, x_k of the hyperparameters.
2. Define the network with the x_k hyperparameters.
3. Train and validate the network.
4. Save the validation error and the used hyperparameters x_k.
5. Return the validation error.

To best utilize the power of Bayesian optimization, at least 30 objective function evaluations should be performed [65]. Hence, the optimization is performed with 30 evaluations. The minimum objective of each iteration is tracked, and figure 7.11 shows the evolution of the optimization algorithm.

A subset of approximately 2 days are used to perform the Bayesian optimization, corresponding to 1 % of the entire dataset. The minimum objective was achieved with the hyperparameter values listed in table 7.6.

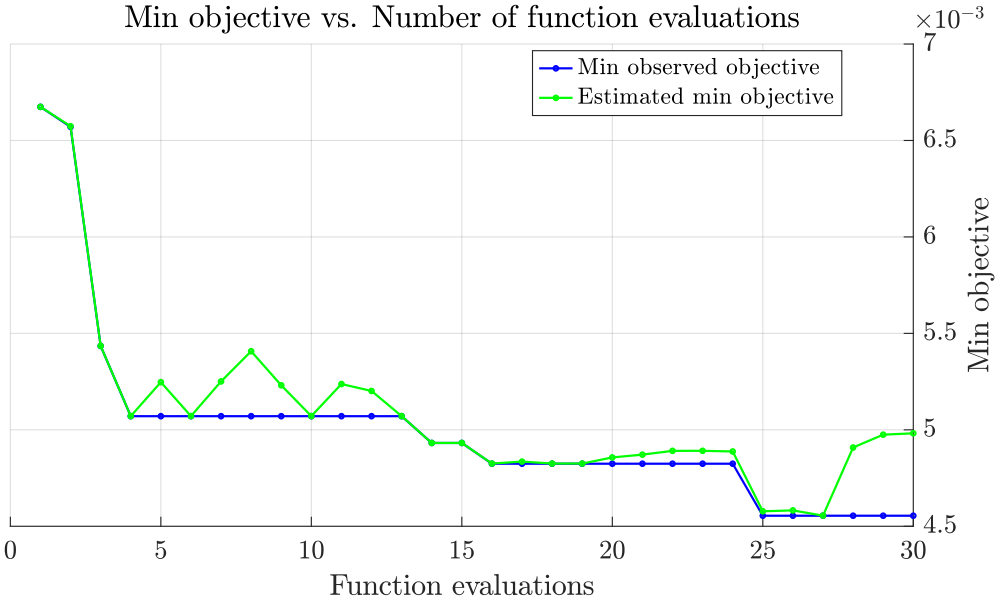

Figure 7.11: Bayesian optimization of the LSTM structure.

Table 7.6: Optimum hyperparameter values determined using Bayesian optimization.

Hyperparameter	Optimized values
L_2 regularization coef.	$2.55 \cdot 10^{-9}$
Learning rate	0.0108
Hidden Layers	2
Hidden units in each hidden layer	93

7.4 LSTM with Bayesian Hyperparameter Optimization

The optimum LSTM network structure determined using Bayesian optimization is evaluated in this section. The model with manually tuned parameters (presented in section 7.2) will be referred to as "LSTM" while the network with Bayesian optimized hyperparameters is referred to as "LSTM-BHP".

The LSTM-BHP network is constructed of four layers; an input layer, two hidden layers with 93 units each and an output layer with a single output. Through Bayesian optimization, the values of the significant hyperparameters (described in section 6.4) were determined and they are displayed in table 7.7.

Compared to the hyperparameters describing the manually tuned LSTM (table 7.3), the L_2 regularization coefficient is reduced by an order 10^5 while the learning rate is approximately doubled. The LSTM-BHP is deeper, but approximately the same width as the manually tuned LSTM.

The network is trained and evaluated on the entire dataset and predicted and observed values are compared in figures 7.12 to 7.14, where smaller sections of the training,

Table 7.7: Optimized hyperparameter values used to train the LSTM-BHP network. The * indicates that MATLAB default values are used while † are shown for values manually chosen (not included in the optimization algorithm).

Hyperparameter	Value/setup
Solver	Adam†
L_2 regularization coef.	$2.55 \cdot 10^{-9}$
Mini-batch size	1000†
Dropout Probability	0.5 *
Learning rate	0.0108
Hidden Layers	2
Hidden units in each hidden layer	93

validation and test data are shown. A figure showing the entire dataset is included in appendix A.3. The partitioning of data is identical to the one described in section 7.2, and divided as follows: 80% training data, 10 % validation data, 10% test data.

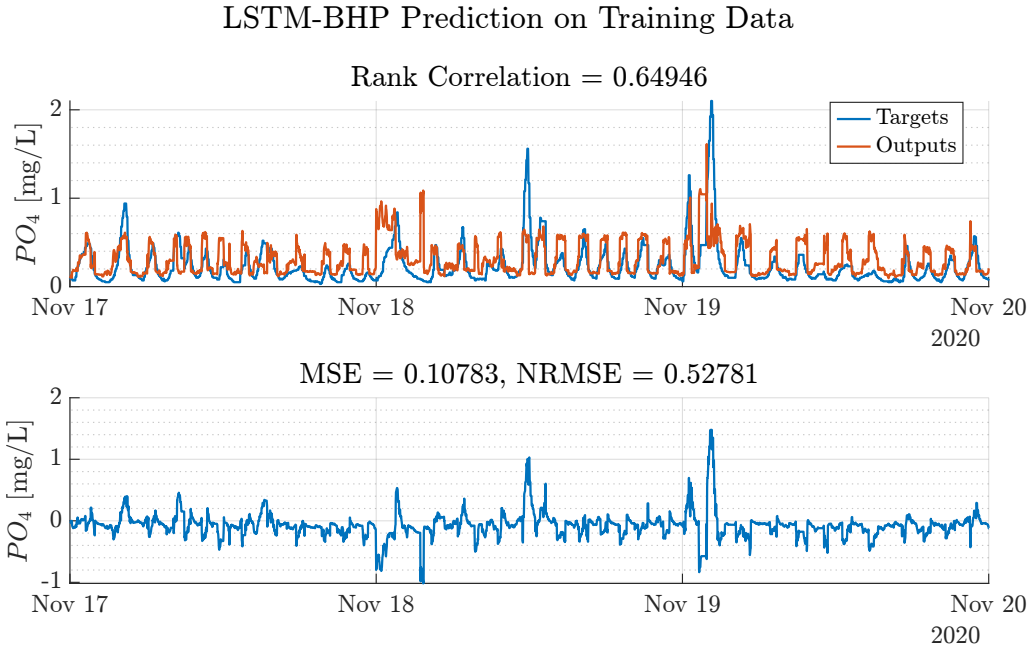

Figure 7.12: Prediction of phosphorus concentration using the LSTM-BHP network on training data.

Figure 7.12 shows a comparison of the predicted output and target for a section of the training data. As expected, the model is able to predict the target values good with a rank correlation of $\rho_S = 0.65$ over the entire training dataset. Similar to the LSTM model presented in section 7.2, the prediction MSE decreases when the model is evaluated on validation data, while the NRMSE increases for the same dataset. Nevertheless, the LSTM-BHP shows better performance than the LSTM in section 7.2, indicating that the model capacity is closer to the required capacity needed for the task. The statistical performance measures of the LSTM-BHP on training, validation and test data are pre-

sented in table 7.8.

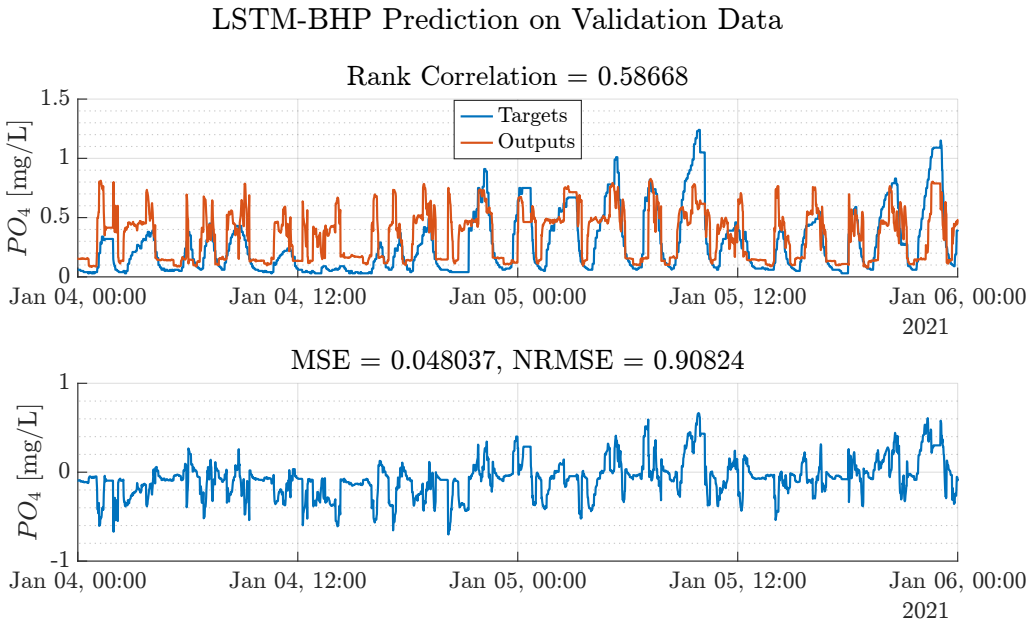

Figure 7.13: Prediction of phosphorus concentration using the LSTM-BHP network on validation data.

Since the validation error is used to choose the best model parameters (weights and biases) during training, it is possible that the best network overfits on the validation data. Hence, the model performance is examined based on the independent test dataset to estimate the generalization error.

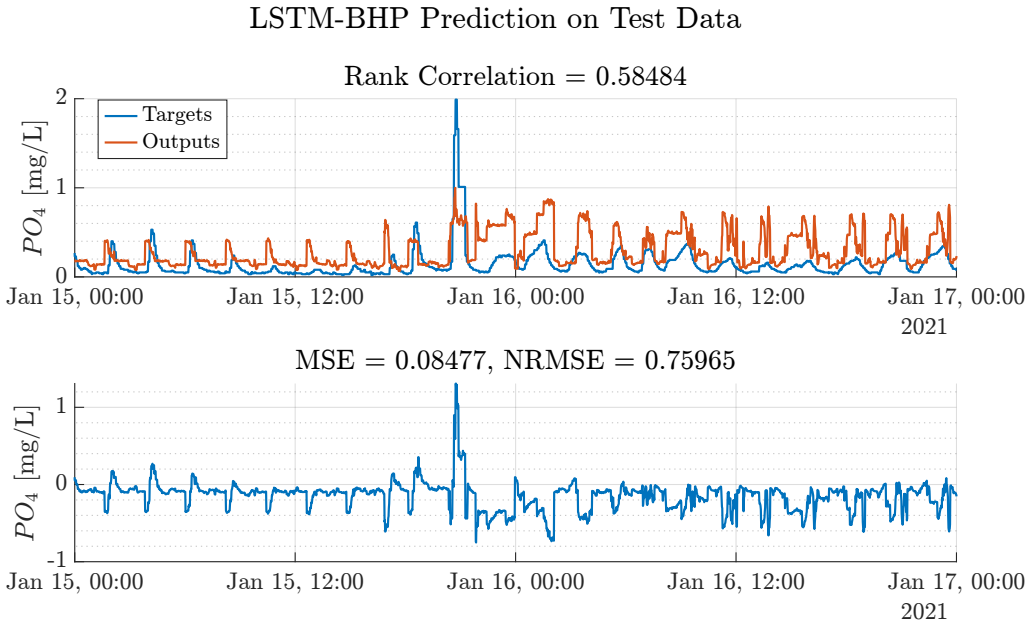

Figure 7.14: Prediction of phosphorus concentration using the LSTM-BHP network on test data.

A period of the test data is shown in figure 7.14, where we see how the model generally struggles to predict both low concentrations (around 0 [mg/L]) and also high concentrations (>1) of phosphorus. However, as was the case with the LSTM, the rank correlation

of the test data is close to that of the validation data, indicating that the model does not overfit to the validation data.

The results presented in figure 7.15, supports the conclusions drawn from figures 7.13 and 7.14; that the model struggles to estimate the true phosphorus concentration, especially for values above 1 [mg/L].

Table 7.8: Performance measures for the LSTM-BHP on training, validation and test data.

	Training	Validation	Test
R^2	0.72	0.18	0.42
MSE	0.11	0.048	0.085
RMSE	0.33	0.22	0.29
NRMSE	0.53	0.91	0.76
ME	0.0067	-0.088	-0.096
SE	0.33	0.2	0.27
lin corr	0.85	0.59	0.71
rank corr	0.65	0.59	0.58

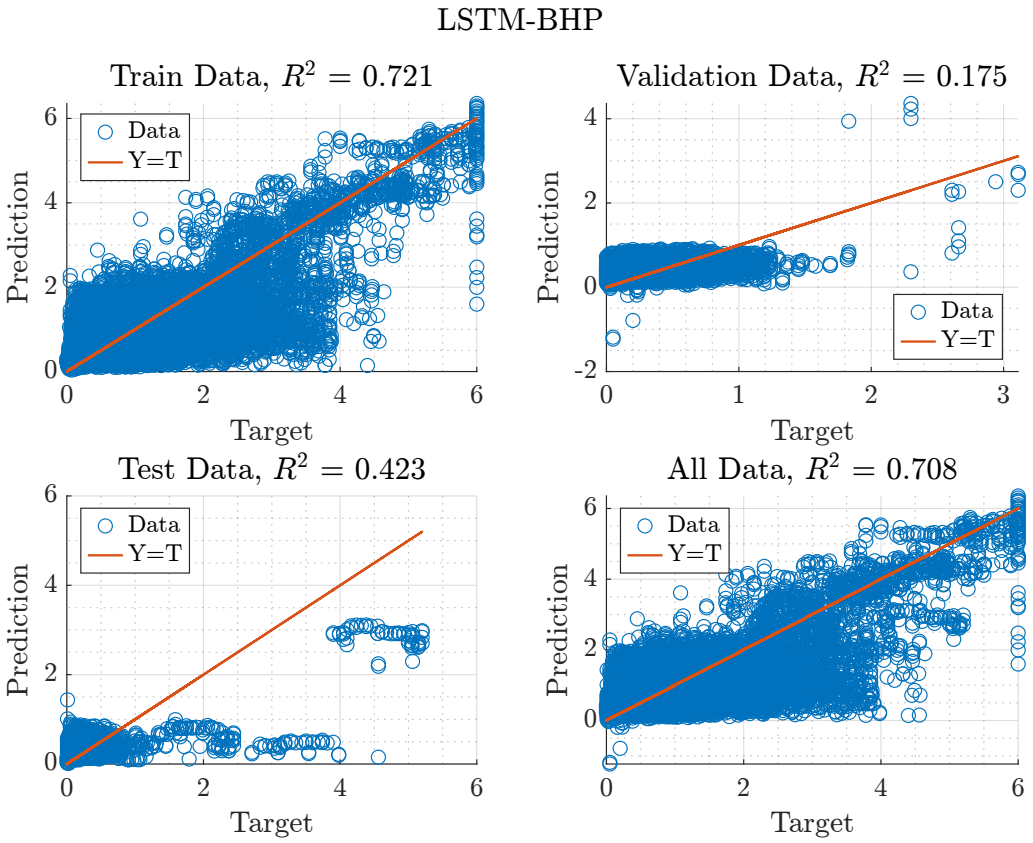

Figure 7.15: Scatter plots of the output value, Y, relative to the target value, T, at the same point in time.

Cross-correlograms are shown in figure 7.16 to examine the dynamic behaviour of the proposed model. Figure 7.16 are similar to the diagrams evaluating the other models, and

shows that the cross-correlation is highest at lag 0, -1 and +6 for the training, validation and test data, respectively. Considering how the model generally struggles to estimate the true magnitude of the system, it is assessed that the slightly higher correlation not located at lag 0 is the result of modelling errors.

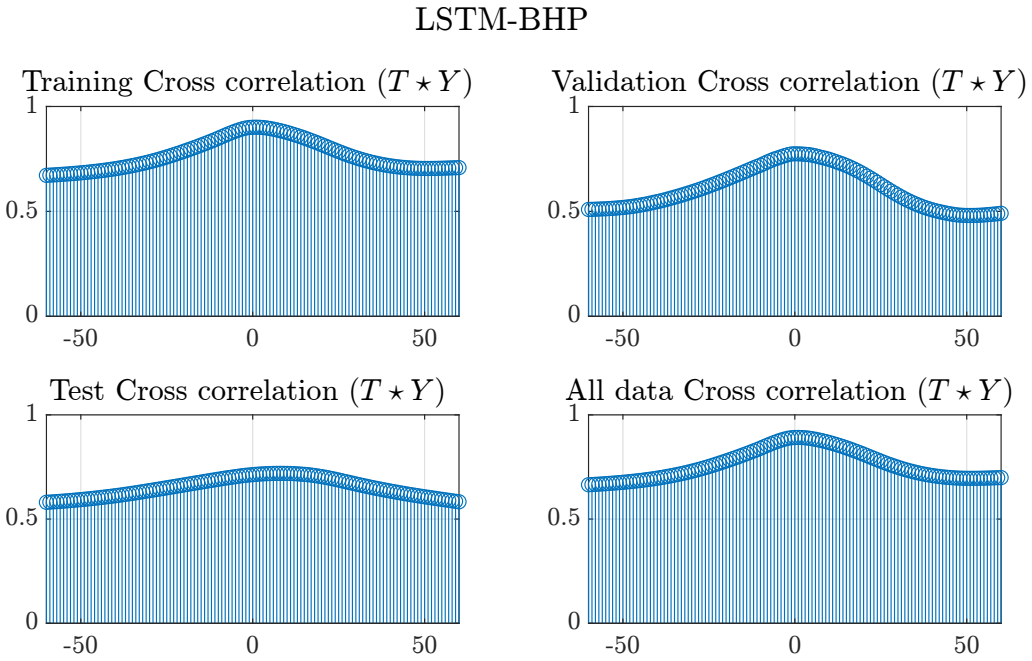

Figure 7.16: Cross correlogram for the different datasets.

7.5 Summary

All performance statistics (equations (7.1) to (7.8)) of the the three proposed models are listed in table 7.9 to compare all results of this chapter.

Table 7.9: Comparison of performance measures for the three obtained networks. If not stated otherwise, the measure is calculated for the independent test dataset.

	NARX	LSTM	LSTM-BHP
R^2 (tr & val)	0.708	0.362	**0.716**
R^2 (test)	0.316	0.408	**0.423**
MSE	0.181	0.087	**0.0848**
RMSE	0.426	0.295	**0.291**
NRMSE	0.827	0.769	**0.76**
ME	**-0.0125**	-0.114	-0.0962
SE	0.425	**0.272**	0.275
ρ_P	0.563	0.707	**0.715**
ρ_S	0.443	0.513	**0.585**

The best performance for each of the statistics is marked with a light green color and bold font, while the model with lowest performance is marked with a light red.

The comparison clearly shows that the LSTM-BHP results in the highest prediction performance when evaluated using multiple statistical measures. Furthermore, it shows a significant improvement of the LSTM-BHP compared to the LSTM, indicating that the Bayesian optimization, although not evaluated on the entire dataset, determined better hyperparameter values than those found by manual tuning and MATLAB default options.

8 Discussion

The motivation behind this project is to create a tool in the form of a mathematical model which provides the foundation for model based control of the phosphorus precipitation process. The current control strategy is presented in chapter 2, where the control scheme shows poor set-point tracking utilized by simple feed-back controllers. The overall goal is to design a model which captures the dynamic behaviour of the phosphorus in wastewater and accurately predicts the concentration. Based on a thorough literature study conducted in section 1.4, several data-driven methods were identified as potential approaches to model the complex system.

By reference to the TSTF principle (try simple things first), which suggest to test simple, cheap, ready-made models before trying more complex ones, a linear model is developed. However, when tested on the data, the linear model shows poor model performance. As the linear model has no incorporated memory, the model is inadequate for the purpose of predicting the not only nonlinear but also time-delayed process.

To accurately model the nonlinearities in the system, two neural network structures are proposed. The NARX network was developed in previous work by the author, and two LSTM models are designed in this work; one with manually tuned hyperparameters, and one with automatically tuned hyperparameters (LSTM-BHP). As there is no benchmark model to compare the models against, they are compared against each other. Since the models are dynamical, evaluating them using a single performance measure, such as MSE is not adequate. Statistical tools may evaluate steady state or static models well, but when it comes to dynamic models, evaluating their performance is much more complex. As the neural networks consists of many thousand parameters, an algebraic approach would be extremely comprehensive, hence, several statistical measures are used for the comparison along with the cross-correlograms to include temporal information in the evaluation.

The LSTM network with Bayesian hyperparameter optimization yields the best performance across several evaluation statistics, and is concluded the best method for phosphorus concentration prediction in wastewater. The LSTM-BHP captures the dynamics of the system both for training and independent test data, although it struggles to predict the magnitude of the concentration correctly. However, as the phosphorus dynamics are presumed cyclostationary, this could be solved by including a full year of data (or maybe more) in the training process.

The finding of this work paves the way to improve the phosphorus removal strategies through model-based or model-predictive control, as the obtained LSTM-BHP provides

the stepping stone for mathematical modelling of phosphorus in WWTPs.

In the following sections, the applied methods and results are discussed in detail.

Data

The data used for modelling is subject to a simple pre-processing approach, where missing data and data points with bad quality (assessed by the online control platform Hubgrade) is replaced with new data points of the previous available measurement of good quality. Hence, for longer periods with bad quality (e.g. when sensors are calibrated), the data used to fit the models is untrue. This yields an unknown upper bound for the performance of the proposed models, as the models either (1) are fit to false data, meaning the performance when predicting true data inevitable decreases, or (2) the models are fit to and accurately predicts the true data but the evaluated performance decreases due to incorrect predictions of the false data.

In some studies, smoothing the data has resulted in increased model prediction performance [16], [43]. However, smoothing the data can result in removing significant dynamics, and the optimal smoothing method is known to be task dependent. Hence the task of smoothing data is seen as an ad-hoc task and solution and omitted in this work. Furthermore, the focus of this project lies within the system identification methodology, hence, signal processing has been reduced to a minimum.

Linear model

Dynamic mode decomposition with control (DMDc) is a method to extract a linear model of a system from data. The method is well established for disease modelling [31], where the actuation (e.g. vaccination) cannot be stopped to determine the underlying dynamics without the effects of actuators. In that way, modelling the precipitation system is similar to modelling the spread of a disease, as the actuation (addition of metal salts in this case) cannot be stopped or changed without environmental consequences.

It is evident that the system of interest is showing nonlinear behavior, as the transient behaviour constantly changes. Hence fitting a linear model to the entire dataset was eliminate as an option. Instead, the aim was to investigate if an online adapting linear model would be adequate to model the nonlinear behavior. However, the linear model has no long term memory and is unable to account for the time delays in the actual system.

Nonlinear models

Two nonlinear model structures are developed in this work; NARX and LSTM. The NARX is a recurrent neural network with external memory in the form of previous inputs in the massive input layer, whereas the LSTM has internal memory through internal recurrent connections and so-called gates. The NARX network hyperparameters were determined in previous work by the author, and is hence regarded a completed solution in this work.

The manually tuned LSTM shows worse performance than the NARX, and the used performance statistics indicates that the model capacity is too small for the complexity of the task. Hence, automatic hyperparameter tuning is used in the form of Bayesian hyperparameter optimization, and the new LSTM model architecture and training procedure shows superior performance compared to the manually tuned LSTM.

As the NARX training duration is significantly longer than the training duration of the LSTM, the NARX was only trained and evaluated on 44 days of data. In contrast, the two LSTM models were trained and evaluated on approximately 200 days of data. Consequently, the NARX performance evaluation may seem to be better than what actually is the case, as it is easier to fit a model to fewer data points.

THE NARX network could be improved through automatic hyperparameter tuning, but the long training duration ranks the NARX below LSTM. Additionally, to obtain high model fidelity of the NARX, the amount of external dynamics must be determined by the developer where an extensive analysis is needed to justify the choice of input and output memory order.

Hyperparameter optimization

Some of the most important hyperparameters were included in a Bayesian optimization algorithm to find the optimum hyperparameter values for the LSTM. As a result, the LSTM architecture experienced a major increase in performance, presumably because the capacity of the neural network was increased and hence approached the optimum capacity for the task.

9 Future Work

Phosphorus removal from wastewater is a complicated process, which is highly dependent on the changing characteristics of the wastewater. This work showed that a LSTM can be used to predict this complex process.

In this work, 25 input variables are used to model the phosphorus concentration. The used inputs were selected in previous work by the author from the principle that no system dynamics should be removed from the mathematical model due to poor engineering judgement. As the process of interest is not well understood, this resulted in the chosen 25 input variables being fed to the neural network. However, several studies has successfully modelled other wastewater processes using 5-10 input variables for the neural networks [8], [16], [18], [20], [24], [42], [43], and a procedure of reducing the number of inputs to the model should hence be investigated in future work. One way to reduce the input layer is by trial and error. Alternatively, a sensitivity analysis can be conducted on the LSTM network recovered in this study. The analysis will make it possible to exclude the input variables that does not affect the predicted value.

For the Bayesian hyperparameter tuning, four variables were included in the optimization algorithm, while two were determined manually. The remaining hyperparameters were given a MATLAB default value. Given that it is a relatively new concept to utilize LSTM networks for system identification [57], the default MATLAB hyperparameters may not be a good choice for tasks regarding system identification. Hence, in future work, more hyperparameters should be included in the optimization with the focus on achieving the optimum model capacity.

10 Conclusion

In this project, data from Kolding central WWTP is used to obtain several data-driven models to predict the phosphorus concentration in the wastewater. The project is motivated by the desire to improve chemical phosphorus precipitation at the plant. Hence, the dynamic model is proposed to act as foundation for model-based controller design. A linear model obtained by applying dynamic mode decomposition with control was presented, but showed poor prediction performance since the model has no incorporated memory. Since the system of interest is inherently nonlinear, neural network models were proposed in order to incorporate memory and temporal information into the models. The NN models showed better performance, and the best prediction performance was achieved using a LSTM model with Bayesian optimized hyperparameters. The Bayesian hyperparameter optimization included four of the most important hyperparameters; Leaning rate and weight decay factor of the training algorithm and number of hidden units and hidden layers in the LSTM architecture. The best model was concluded to the a LSTM with 25 inputs, 2 hidden LSTM layers with 93 units in each and a output layer with a single unit. When tested on new data, the best model shows strong performance estimating the phosphorus concentration with a low MSE of 0.0848 and $R^2 = 0.42$.

Bibliography

[1] O. Nelles, *Nonlinear System Identification: From Classical Approaches to Neural Networks, Fuzzy Models, and Gaussian Processes*, English, 2nd ed. Springer International Publishing, 2020.

[2] P. S. N. Garikiparthy, S. C. Lee, H. Liu, S. S. Kolluri, I. J. Esfahani, and C. K. Yoo, "Evaluation of multiloop chemical dosage control strategies for total phosphorus removal of enhanced biological nutrient removal process", *Korean Journal of Chemical Engineering*, vol. 33, no. 1, pp. 14–24, 2016.

[3] N. Gilbert, "The disappearing nutrient", *Nature*, vol. 461, no. 7265, pp. 716–718, 2009.

[4] H. Feldman, X. Flores-Alsina, P. Ramin, K. Kjellberg, U. Jeppsson, D. J. Batstone, and K. V. Gernaey, "Modelling an industrial anaerobic granular reactor using a multi-scale approach", *Water Research*, vol. 126, pp. 488–500, Dec. 2017.

[5] J. T. Bunce, E. Ndam, I. D. Ofiteru, A. Moore, and D. W. Graham, *A review of phosphorus removal technologies and their applicability to small-scale domestic wastewater treatment systems*, 2018.

[6] P. Ingildsen, C. Rosen, K. V. Gernaey, M. K. Nielsen, T. Gulldal, and B. N. Jacobsen, "Modelling and control strategy testing of biological and chemical phosphorus removal at Avedøre WWTP", *Water Science and Technology*, vol. 53, no. 4-5, pp. 105–113, 2006.

[7] A. M. A. Dias and E. C. Ferreira, "Computational Intelligence Techniques for Supervision and Diagnosis of Biological Wastewater Treatment Systems", in *Computational Intelligence Techniques for Bioprocess Modelling, Supervision and Control*, M. do Carmo Nicoletti and L. C. Jain, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 127–162. [Online]. Available: `https://doi.org/10.1007/978-3-642-01888-6_5`.

[8] M. H. Husin, M. F. Rahmat, N. A. Wahab, and M. F. M. Sabri, "Neural Network Ammonia-Based Aeration Control for Activated Sludge Process Wastewater Treatment Plant", in *Proceedings of the 11th National Technical Seminar on Unmanned System Technology 2019*, Z. Md Zain, H. Ahmad, D. Pebrianti, M. Mustafa, N. R. H. Abdullah, R. Samad, and M. Mat Noh, Eds., Singapore: Springer Singapore, 2021, pp. 471–487.

[9] L. Zhao, T. Dai, Z. Qiao, P. Sun, J. Hao, and Y. Yang, "Application of artificial intelligence to wastewater treatment: A bibliometric analysis and systematic review of technology, economy, management, and wastewater reuse", *Process Safety and Environmental Protection*, vol. 133, no. 92, pp. 169–182, 2020. [Online]. Available: https://doi.org/10.1016/j.psep.2019.11.014.

[10] K. B. Newhart, R. W. Holloway, A. S. Hering, and T. Y. Cath, "Data-driven performance analyses of wastewater treatment plants: A review", *Water Research*, vol. 157, pp. 498–513, 2019. [Online]. Available: https://doi.org/10.1016/j.watres.2019.03.030.

[11] Metcalf & Eddy Inc, G. Tchobanoglous, F. L. Burton, and H. D. Stensel, *Wastewater Engineering: Treatment and Reuse*, 4th. McGraw-Hill Education, 2004.

[12] M. Henze, P. Harremoës, J. Jansen, and E. Arvin, *Wastewater treatment: Biological and chemical processes*, 3rd. Springer, 2002.

[13] L. D. Hansen, "System Identification of Phosphorus Precipitation in Activated Sludge Wastewater Treatment Plant using Neural Networks", Aalborg University, Esbjerg, Tech. Rep., 2021.

[14] L. D. Hansen, M. Veng, and P. Durdevic, "Compressor Scheduling and Pressure Control for an Alternating Aeration Activated Sludge Process—A Simulation Study Validated on Plant Data", *Water*, vol. 13, no. 8, p. 1037, Apr. 2021. [Online]. Available: https://doi.org/10.3390/w13081037.

[15] M. Bongards, "Controlling Biological Wastewater Treatment Plants Using Fuzzy Control and Neural Networks", in *Computational Intelligence*, B. Reusch, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 142–150.

[16] S. Hwangbo, R. Al, X. Chen, and G. Sin, "Integrated Model for Understanding N2O Emissions from Wastewater Treatment Plants: A Deep Learning Approach", *Environmental Science and Technology*, vol. 55, no. 3, pp. 2143–2151, 2021.

[17] M. S. Gaya, N. A. Wahab, Y. M. Sam, and S. I. Samsuddin, "Feed-Forward Neural Network Approximation Applied to Activated Sludge System", in *AsiaSim 2013*, G. Tan, G. K. Yeo, S. J. Turner, and Y. M. Teo, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 587–598.

[18] D. Aguado, J. Ribes, T. Montoya, J. Ferrer, and A. Seco, "A methodology for sequencing batch reactor identification with artificial neural networks: A case study", *Computers and Chemical Engineering*, vol. 33, no. 2, pp. 465–472, 2009.

[19] P. A. Stentoft, T. Munk-Nielsen, L. Vezzaro, H. Madsen, J. K. Møller, and P. S. Mikkelsen, "Towards model predictive control: Online predictions of ammonium and nitrate removal by using a stochastic ASM", *Water Science and Technology*, vol. 79, no. 1, pp. 51–62, Jan. 2019. [Online]. Available: http://iwaponline.com/wst/article-pdf/79/1/51/735663/wst079010051.pdf.

[20] H. Han, S. Zhu, J. Qiao, and M. Guo, "Data-driven intelligent monitoring system for key variables in wastewater treatment process", *Chinese Journal of Chemical Engineering*, vol. 26, no. 10, pp. 2093–2101, 2018. [Online]. Available: https://doi.org/10.1016/j.cjche.2018.03.027.

[21] J. Keskitalo and K. Leiviskä, "Artificial Neural Network Ensembles in Hybrid Modelling of Activated Sludge Plant", in *Intelligent Systems'2014*, P Angelov, K Atanassov, L Doukovska, M Hadjiski, V Jotsov, J Kacprzyk, N Kasabov, S Sotirov, E Szmidt, and S Zadrożny, Eds., Cham: Springer International Publishing, 2015, pp. 683–694.

[22] I. Pisa, R. Vilanova, I. Santín, J. L. Vicario, and A. Morell, "Artificial Neural Networks Application to Support Plant Operation in the Wastewater Industry", in *IFIP Advances in Information and Communication Technology*, vol. 553, Springer New York LLC, 2019, pp. 257–265. [Online]. Available: https://doi.org/10.1007/978-3-030-17771-3_22.

[23] W. Cao and Q. Yang, "Prediction Based on Online Extreme Learning Machine in WWTP Application", in *Neural Information Processing*, L. Cheng, A. C. S. Leung, and S. Ozawa, Eds., Cham: Springer International Publishing, 2018, pp. 184–195.

[24] M. Besharati Fard, S. A. Mirbagheri, A. Pendashteh, and J. Alavi, "Estimation of effluent parameters of slaughterhouse wastewater treatment with artificial neural network and B-spline quasi interpolation", *International Journal of Environmental Research*, vol. 14, no. 5, pp. 527–539, 2020. [Online]. Available: https://doi.org/10.1007/s41742-020-00274-1.

[25] X. Meng, Y. Zhang, and J. Qiao, "An adaptive task-oriented RBF network for key water quality parameters prediction in wastewater treatment process", *Neural Computing and Applications*, vol. 0123456789, no. 3, 2021. [Online]. Available: https://doi.org/10.1007/s00521-020-05659-z.

[26] D. Dürrenmatt and W. Gujer, "Data-driven modeling approaches to support wastewater treatment plant operation", *Environmental Modelling and Software*, vol. 30, pp. 47–56, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.envsoft.2011.11.007.

[27] C. Rosen and J. A. Lennox, "Multivariate and multiscale monitoring of wastewater treatment operation", *Water Research*, vol. 35, no. 14, pp. 3402–3410, Oct. 2001.

[28] G. J. Odom, K. B. Newhart, T. Y. Cath, and A. S. Hering, "Multistate multivariate statistical process control", *Applied Stochastic Models in Business and Industry*, vol. 34, no. 6, pp. 880–892, 2018.

[29] S. L. Brunton and J. N. Kutz, *Data Driven Science & Engineering - Machine Learning, Dynamical Systems, and Control*, 2017. [Online]. Available: databook.uw.edu.

[30] J. L. Proctor and P. A. Eckhoff, "Discovering dynamic patterns from infectious disease data using dynamic mode decomposition", *International Health*, vol. 7, no. 2, pp. 139–145, 2015.

[31] J. L. Proctor, S. L. Brunton, and J. N. Kutz, "Dynamic mode decomposition with control", *SIAM Journal on Applied Dynamical Systems*, vol. 15, no. 1, pp. 142–161, 2016.

[32] K. S. Narendra and K Parthasarathy, "Identification and control of dynamical systems using neural networks", *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 4–27, 1990.

[33] M. R. Meireles, P. E. Almeida, and M. G. Simões, "A comprehensive review for industrial applicability of artificial neural networks", *IEEE Transactions on Industrial Electronics*, vol. 50, no. 3, pp. 585–601, 2003.

[34] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning", *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[35] G. Calderon, J. P. Draye, D. Pavisic, R. Teran, and G. Libert, "Nonlinear dynamic system identification with dynamic recurrent neural networks", *Proceedings of International Workshop on Neural Networks for Identification, Control, Robotics, and Signal/Image Processing, NICROSP*, pp. 49–54, 1996.

[36] K. S. Narendra and K Parthasarathy, "Adaptive identification and control of dynamical systems using neural networks", in *Proceedings of the 28th IEEE Conference on Decision and Control*, 1989, pp. 1737–1738.

[37] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[38] N. K. Sinha, M. M. Gupta, and D. H. Rao, "Dynamic neural networks: An overview", *Proceedings of the IEEE International Conference on Industrial Technology*, vol. 1, pp. 491–496, 2000.

[39] H. Xie, H. A. O. Tang, and Y.-h. Liao, "Time series prediction based on NARX neural nertworks: An advanced approach", in *Proceedings - International Conference on Machine Learning and Cybernetics*, Boading, 2009, p. 690.

[40] A. Wunsch, T. Liesch, and S. Broda, "Forecasting groundwater levels using nonlinear autoregressive networks with exogenous input (NARX)", *Journal of Hydrology*, vol. 567, pp. 743–758, 2018. [Online]. Available: `https://doi.org/10.1016/j.jhydrol.2018.01.045`.

[41] S. Çoruh, F. Geyikçi, E. Kiliç, and U. Çoruh, "The use of NARX neural network for modeling of adsorption of zinc ions using activated almond shell as a potential biosorbent", *Bioresource Technology*, vol. 151, pp. 406–410, 2014. [Online]. Available: `http://dx.doi.org/10.1016/j.biortech.2013.10.019`.

[42] M. Zounemat-Kermani, D. Stephan, and R. Hinkelmann, "Multivariate NARX neural network in prediction gaseous emissions within the influent chamber of wastewater treatment plants", *Atmospheric Pollution Research*, vol. 10, no. 6, pp. 1812–1822, 2019. [Online]. Available: `https://doi.org/10.1016/j.apr.2019.07.013`.

[43] S. Hwangbo, R. Al, and G. Sin, "An integrated framework for plant data-driven process modeling using deep-learning with Monte-Carlo simulations", *Computers and Chemical Engineering*, vol. 143, 2020.

[44] J. Wu, X. Y. Chen, H. Zhang, L. D. Xiong, H. Lei, and S. H. Deng, "Hyperparameter optimization for machine learning models based on Bayesian optimization", *Journal of Electronic Science and Technology*, vol. 17, no. 1, pp. 26–40, 2019. [Online]. Available: https://doi.org/10.11989/JEST.1674-862X.80904120.

[45] J. Bergstra, R Bardenet, Y. Bengio, B. Kégl, and R. Bardenet, "Algorithms for Hyper-Parameter Opti-mization", Tech. Rep., 2011. [Online]. Available: https://hal.inria.fr/hal-00642998.

[46] J. Bergstra, J. B. Ca, and Y. B. Ca, "Random Search for Hyper-Parameter Optimization Yoshua Bengio", Tech. Rep., 2012, pp. 281–305. [Online]. Available: http://scikit-learn.sourceforge.net..

[47] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian Optimization of Machine Learning Algorithms", *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[48] Kolding Spildevand AS, "Miljø- og arbejdsmiljø redegørelse", 2013. [Online]. Available: https://greennetwork.dk.

[49] C. C. Aggarwal, *Neural Networks and Deep Learning*. Springer, Cham, 2018, pp. 389–411.

[50] D. P. Kuttichira, E. A. Gopalakrishnan, V. K. Menon, and K. P. Soman, "Stock price prediction using dynamic mode decomposition", in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2017, pp. 55–60.

[51] S. Haykin, *Neural networks: a comprehensive foundation*, 2nd ed. Pearson Education Inc., 1999.

[52] T. M. Mitchell, *Machine learning*. McGraw-hill New York, 1997.

[53] J. L. Elman, "Finding Structure in Time", *Cognitive Science*, vol. 14, no. 2, pp. 179–211, Mar. 1990.

[54] M. I. Jordan, "Serial Order: A Parallel Distributed Processing Approach-Ies Report 8604 Institute for Cognitive Science University of California , San Diego", no. 667, 1986.

[55] Y. Ma and J. C. Principe, "A Taxonomy for Neural Memory Networks", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 6, pp. 1780–1793, 2020.

[56] J. C. B. Gamboa, "Deep Learning for Time-Series Analysis", *arXiv*, Jan. 2017. [Online]. Available: http://arxiv.org/abs/1701.01887.

[57] Y. Wang, "A new concept using LSTM Neural Networks for dynamic system identification", in *Proceedings of the American Control Conference*, Institute of Electrical and Electronics Engineers Inc., Jun. 2017, pp. 5324–5329.

[58] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: LSTM cells and network architectures", *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[59] N. Lanzetti, Y. Z. Lian, A. Cortinovis, L. Dominguez, M. Mercangoz, and C. Jones, "Recurrent neural network based MPC for process industries", *2019 18th European Control Conference, ECC 2019*, pp. 1005–1010, 2019.

[60] A. H. Victoria and G. Maragatham, "Automatic tuning of hyperparameters using Bayesian optimization", *Evolving Systems*, vol. 12, no. 1, pp. 217–223, Mar. 2021. [Online]. Available: https://doi.org/10.1007/s12530-020-09345-2.

[61] N. Reimers and I. Gurevych, *Optimal hyperparameters for deep LSTM-networks for sequence labeling tasks*, 2017. [Online]. Available: https://arxiv.org/abs/1707.09861.

[62] D. P. Kingma and J. Lei Ba, "Adam: A method for Stochastic Optimization", 2017.

[63] N. Srivastava, G. Hinton, A. Krizhevsky, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", Tech. Rep., 2014, pp. 1929–1958.

[64] MathWorks, *Deep Learning Toolbox - MATLAB*. [Online]. Available: mathworks.com/products/deep-learning (visited on 05/28/2021).

[65] ——, *Statistics and Machine Learning Toolbox - MATLAB*. [Online]. Available: mathworks.com/products/statistics (visited on 05/28/2021).

A Appendix: Results

A.1 Prediction using NARX

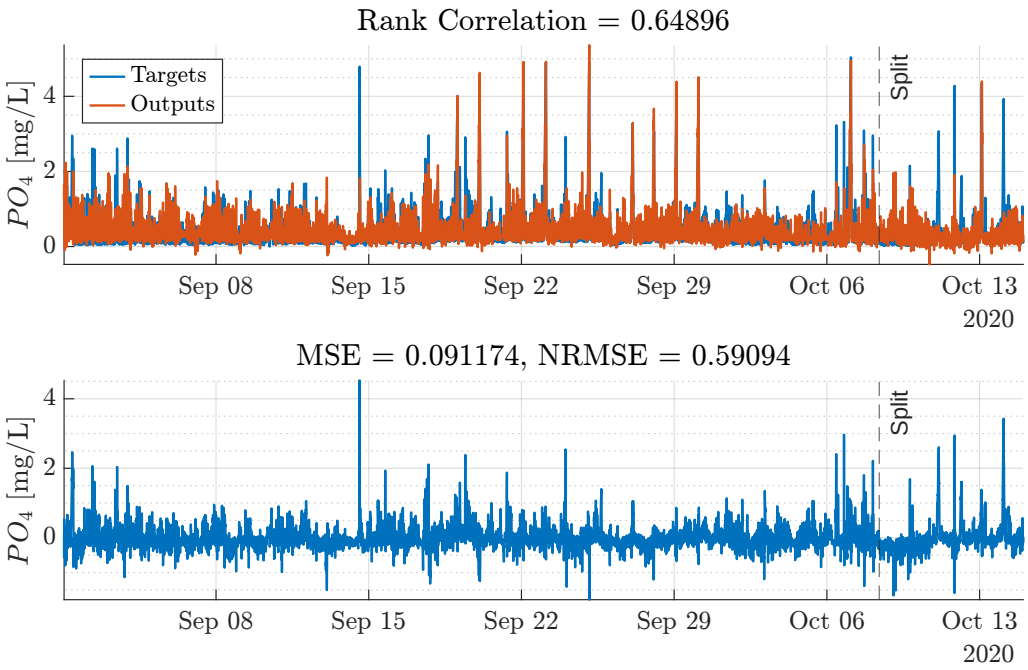

Figure A.1: All data used to train and evaluate the NARX network. The vertical dashed line shows the division of data used for training (training + validation data) and test data.

A.2 Prediction using LSTM

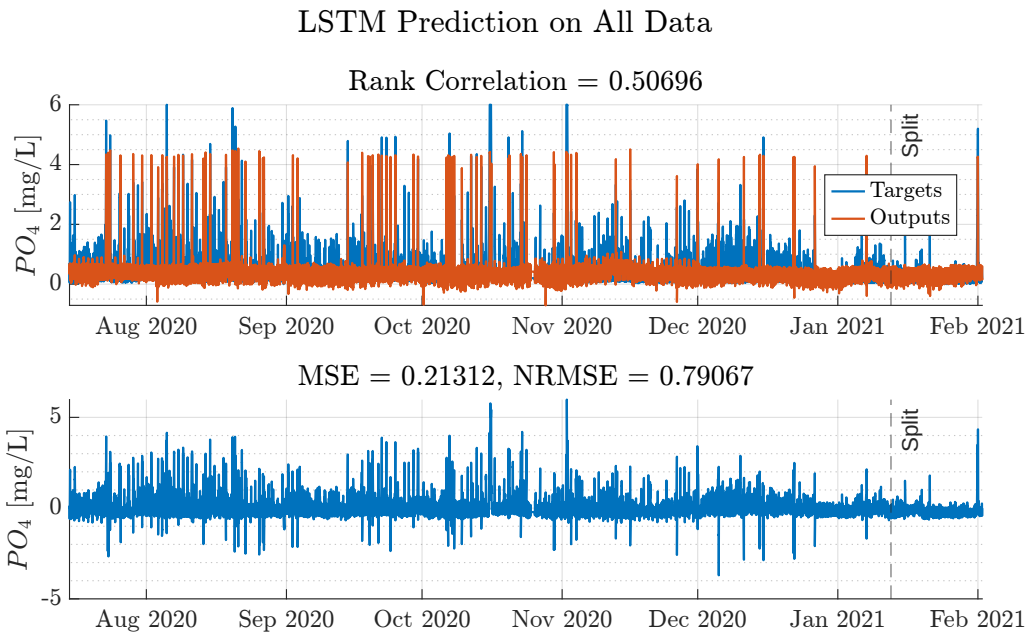

Figure A.2: All data used to train and evaluate the LSTM network. The vertical dashed line shows the division of data used for training (training + validation data) and test data.

A.3 Prediction using LSTM with Hyperparameter Tuning

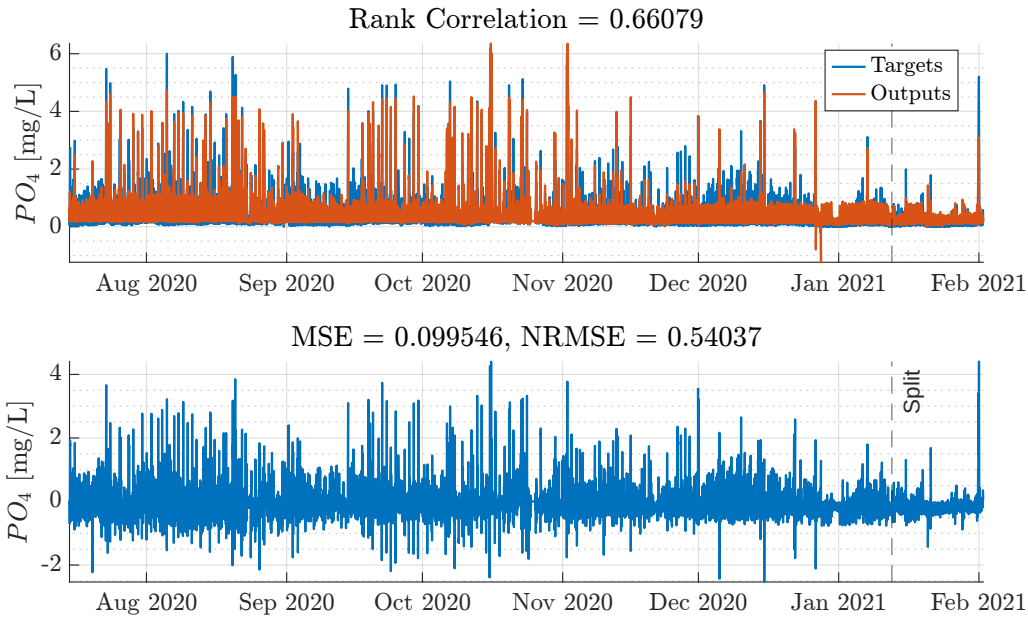

Figure A.3: All data used to train and evaluate the LSTM network with optimal hyperparameters. The vertical dashed line shows the division of data used for training (training + validation data) and test data.