Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Ola Solli Grønningsæter

Echo State Network Based Inverse Models for Feedforward Assisted Optimal Control of Electric Submersible Pumps

Master's thesis in Cybernetics and Robotics
Supervisor: Lars S. Imsland
Co-supervisor: Eduardo Camponogara
July 2023

NTNU
Norwegian University of Science and Technology

UFSC

Ola Solli Grønningsæter

Echo State Network Based Inverse Models for Feedforward Assisted Optimal Control of Electric Submersible Pumps

UNIVERSIDADE FEDERAL
DE SANTA CATARINA

Master's thesis in Cybernetics and Robotics
Supervisor: Lars S. Imsland
Co-supervisor: Eduardo Camponogara
July 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Acknowledgments

First and foremost, I want to thank my supervisors, Prof. Eduardo Camponogara (UFSC) and Prof. Lars S. Imsland (NTNU), for their extensive support and guidance throughout this project. I will always be grateful to them for the opportunity to write my dissertation here in Florianópolis, Brazil. It has been an experience for life. Additionally, I would like to thank Prof. Eric A. Antonelo (UFSC) for sharing his vast knowledge of echo state networks and artificial intelligence in general.

There are many people who have motivated me during this work. I would especially like to thank M. Iffan Hannanu, Bruno Pacheco, and Pedro Antunes for all the great academic, as well as non-academic, conversations during the many hours spent in the research lab at UFSC. It has been a pleasure to get to know you, and I am thankful for all the knowledge we have shared.

Furthermore, I want to give a special thanks to André Pelosini Del Bianco and Gabriel Diz Acosta. Thank you for all the memories, conversations, and for giving me the warmest welcome to Brazil. I am sure my stay would not have been the same without getting to know you, and I am looking forward to seeing you again in the future. Also, thank you to my Norwegian companions, Aurora S. Bjørlo and Maren F. Arnø, for the adventures we have shared here in Brazil. The three of you have motivated and inspired me during the whole time.

Lastly, I would like to thank my partner, Hanne B. Krogstie, as well as my mother, father, brother, and the rest of my family and friends, for their continuous support throughout my studies. It has always been a great source of motivation for me.

Ola S. Grønningsæter
Florianópolis, July 2023

Abstract

Handling disturbances is arguably one of the most important, yet challenging, aspects of control theory. Despite the existence of numerous methods and extensive theories on how to deal with disturbances, they are always present in any real-world process. Disturbances generally represent uncertainties in models, and it can be difficult to determine their origin. This demands looser constraints when optimizing the process control, affecting both profitability and efficiency. Many modern industries can be characterized by their insatiable demand for economic profitability and efficiency. This is one of many reasons why disturbance remains an important area of research within control theory.

In this dissertation, a method to enhance the performance of a nonlinear model predictive controller (NMPC) was investigated. The NMPC was utilized to control an electric submersible pump (ESP) subject to disturbances that were not accounted for in the model. To address this, an echo state network (ESN) was used to identify the inverse model of the disturbance-inflicted dynamical system. Ultimately, the obtained inverse model was used to synthesize a feedforward controller. This controller generated corrections to the current NMPC control input based on this input along with the control reference and the current system state.

The results obtained from the experiments demonstrate that introducing ESN-based feedforward control can indeed improve the performance of the NMPC. It was also observed that the ESN's ability to learn the inverse model depends on the particular noise used to excite the ESN output during the training phase. This became particularly evident after comparing two models trained with uniformly and normally distributed noise, with the uniform noise model yielding better overall performance.

Furthermore, it was demonstrated that the ESN can successfully identify a satisfactory inverse model for feedforward control even without prior knowledge of the actual system disturbances. In an experiment where the controller manipulated both ESP control inputs, the mean absolute error (MAE) was improved by 63.03% on average. This improvement was achieved regardless of the disturbance characteristics and steps in the control reference.

Sammendrag

Håndtering av forstyrrelser er antakelig en av de viktigste, men også mest utfordrende, områdene innen reguleringsteknikk. Til tross for at det finnes uttalelige metoder, samt store mengder teori på hvordan forstyrrelser kan håndteres, vil alle fysiske prosesser påvirkes av dette i praksis. Generelt representerer forstyrrelser usikkerheten i en modell, og det kan være vanskelig å avgjøre hvor disse stammer fra. Dette påvirker både lønnsomheten og effektiviteten til prosessen, da det er avgjørende for hvilke begrensninger som kan benyttes for å optimalisere kontrollbruken. Moderne industri kjennetegnes i stor grad av økonomisk lønnsomhet og effektivitet. Dette er en av mange grunner til at forstyrrelser fremdeles er, og vil forbli, et viktig forskningsområde innen reguleringsteknikk.

I denne masteroppgaven ble en metode for å forbedre ytelsen til en ulineær-modell-prediktiv-kontroller (NMPC) undersøkt. Kontrolleren ble brukt til å regulere en elektrisk nedsenkbar pumpe (ESP) som var utsatt for forstyrrelser som ikke inngikk i modellen. Videre ble den inverse modellen for dette forstyrrelsesutsatte systemet identifisert ved hjelp av et ekko-tilstands-nettverk (ESN). Ut fra denne ble det laget en foroverkopling som genererte korrigeringer til kontrollsignalene fra NMPCen basert på det nåværende NMPC-kontrollsignalet, kontrollreferansen, samt den nåværende systemtilstanden.

Resultatene fra eksperimentene viser at ESN-basert foroverkopling kan forbedre ytelsen til NMPCen. Det ble også observert at ESNens evne til å lære den inverse modellen avhenger av den spesifikke støyen som ble brukt for å eksitere ESN-utgangen under treningsfasen. Dette kom svært tydelig frem ved sammenligning av to modeller som var trent med henholdsvis uniformfordelt og normalfordelt støy. I dette eksperimentet ga modellen trent med uniformfordelt støy langt bedre ytelse totalt sett.

Videre ble det vist at en ESN kan identifisere en invers modell for foroverkopling også uten forkunnskap om de faktiske systemforstyrrelsene. I et av eksperimentene der kontrolleren manipulerte begge kontrollinngangene til ESPen, ble den gjennomsnittlige absolutte feilen (MAE) forbedret med 63.03% i snitt. Dette resultatet var verken påvirket av karakteristikken til forstyrrelsene eller enhetsstegene i reguleringsreferansen.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
BHP	Brake Horsepower
DAE	Differential Algebraic Equation
DMC	Dynamic Matrix Control
ESP	Electric Submersible Pump
GPC	Generalized Predictive Control
IAE	Integral Absolute Error
Ipopt	Interior Point Optimizer
LQR	Linear Quadratic Controller
LP	Linear Programming
LSTM	Long short-term memory
LTI	Linear time-invariant
MBC	Model-based Control
MAE	Mean Absolute Error
MIMO	Multiple-input, multiple-output
MISO	Multiple-input, single-output
MPC	Model Predictive Control
MSE	Mean Squared Error
NLP	Nonlinear Programming
NMPC	Nonlinear Model Predictive Control
NRMSE	Normalized Root Mean Squared Error
NN	Neural Network
ODE	Ordinary Differential Equation
QP	Quadratic Programming
RNN	Recursive Neural Network
SIMO	Single-input, multiple-output
SISO	Single-input, single-output
SQP	Sequential Quadratic Programming
VCF	Viscosity Correction Factor
VSD	Variable Speed Driver

1 Introduction

1.1 Background and Motivation

Over the last two years, artificial intelligence (AI) has received massive public attention. Especially after OpenAI released ChatGPT in late 2022, the public has generally become more aware of all the possibilities this technology possesses (De Angelis et al., 2023). This also applies to various companies and entire industries, and it may be argued in the future that the release of ChatGPT marks the beginning of a new paradigm. In any case, the enhanced public availability and awareness of AI might further accelerate its development the same way as with the internet after it was publicly released in the early 1990s (Leiner et al., 2009). Yet, the sudden hype and upswing for AI has also created skepticism among experts in the field. In the aftermath of the Cambridge Analytica scandal, it is evident that the current developments within AI technology go too fast for the authorities to keep up. This has made experts from both the tech industry and academia to publicly ask for a halt in the development pending the establishment of ethical guidelines for further development of AI (Vallance, 2023).

However, the massive potential that the development of AI carries for various industries should not be underexaggerated. Especially now at the beginning of the fourth industrial revolution which is, among other things, characterized by the ability to learn from process data (Zambrano et al., 2022). Operators and analysts can indeed learn a lot about a given process from recorded timeseries data. Nevertheless, it has been demonstrated several times in the literature that AI models are capable of identifying connections in operational data far beyond human interpretations.

System modeling is a tool of high importance in many disciplines that benefits greatly from the use of AI. Modern processes have become too complex in general for traditional system identification based on first principles (i.e., Newton's laws, mass balances, etc.). Instead, it has been developed specialized AIs capable of identifying system models purely from recorded process data. This modeling technique is frequently referred to as *black-box modeling* and was demonstrated in the preliminary work preceding this dissertation, (Grønningsæter, 2022). In that project, an Echo State Network (ESN) was utilized to identify the dynamical model of an electric submersible pump (ESP) based solely on simulated operational data.

The ESN is an artificial neural network (ANN) specialized to recognize complex dynamical systems from timeseries data. It was for instance used to learn the dynamical nonlinear behaviors for a downhole pressure estimation in (Antonelo, Camponogara, & Foss, 2017), and also used to model increasingly complex behavior from examples in (Antonelo & Schrauwen, 2015). Compared to other more conventional recursive neural networks (RNN), the ESN requires significantly shorter training times due to the utilization of linear regression. Considering this fact and the vast importance of process models in modern real-world applications, it remains a crucial subject for research.

Within the field of control theory, dynamical system models are of particular interest for model-based control (MBC). In (Jordanou et al., 2022), an ESN was used to identify

the system model for an ESP. The identified model was thereafter utilized in a nonlinear model predictive controller (NMPC) to control the same ESP. Despite the results indicating a good generalization of the system model, the NMPC proved too computationally expensive to achieve long-term predictions within the required deadlines. This suggests that more complex systems probably require even longer computational times as simulations of the ESP model itself are not among the most computationally demanding. Consequently, these models become unsatisfactory for optimal control applications as most controllers operate on strict deadlines.

This dissertation will investigate another approach involving system identification using ESN. The system of interest is still the ESP, but it will also be subject to disturbance. Inspired by the work conducted in (Jordanou, 2019), an ESN will be used to recognize the inverse system model. Subsequently, this model will be utilized in a feedforward controller intended to support an NMPC running with the nominal ESP system model. It is believed that this setup will overcome the computational time challenges while also effectively suppressing the disturbances that are not accounted for in the nominal model used in the NMPC.

If this approach proves successful, it will demonstrate the possibility of using AI to identify dynamical models capable of capturing the dynamics of unmeasurable disturbances. Obtaining insight like this into unobservable parameters is arguably the most valuable aspect of inverse modeling. In the context of control theory, unmeasurable disturbance has traditionally been handled through controller robustness. Increased robustness often leads to looser controller constraints which are usually less beneficial both economically and environmentally.

1.2 Research Objectives and Research Questions

This Master's dissertation documents the research conducted during my Master's project and serves as the concluding part of my Master's degree in Cybernetics and Robotics at the Norwegian University of Science and Technology (NTNU). The work was carried out in cooperation with the Federal University of Santa Catarina (UFSC) from March 1, 2023, until July 13, 2023. The primary objective of the project was to investigate if the performance of an NMPC controlling an ESP subject to disturbance could be improved by introducing feedforward control with ESN-based inverse models. This involves the implementation of

- a simulator for the ESP lifted well.
- an NMPC for the ESP simulator.
- an ESN to learn models for the inverse ESP subject to disturbance.
- feedforward control based on the inverse models.

Leading up to this Master's project, a preliminary project was carried out during the fall of 2022. This project resulted in a report available in (Grønningsæter, 2022) which documents, among other things, the implementation of a simulator for the ESP lifted well and the implementation of an ESN framework. A few pages of this dissertation are therefore extracted from the preliminary project. However, this will be highlighted in the relevant chapters.

After the implementations, different types of stochastic noise will be used to excite the ESN to learn different inverse models of the ESP subject to disturbance. Then, multiple controllers featuring the NMPC (using the nominal ESP model) and the ESN-based feedforward controllers will be synthesized and experimented upon. These experiments will attempt to answer the following research questions:

I. Is it possible to improve the performance of the NMPC using feedforward control based on inverse models provided by an ESN?

II. Will different distributions of the stochastic noise exciting the ESN affect the ability to learn the inverse model of the ESP subject to disturbance?

III. Can the NMPC also benefit from ESN-based feedforward control when the ESN has no prior knowledge of the actual disturbance affecting the ESP during the training phase?

IV. How will the ESN's ability to learn a sufficient inverse model for feedforward control be affected by the ESP using multiple control inputs?

1.3 Contributions

The work with this dissertation has led to the following contributions:

- A simulator for the ESP lifted well both with and without disturbance.
- An NMPC capable of controlling an ESP using either one or two control inputs.
- A control scheme utilizing ESN-based feedforward control that is capable of assisting an NMPC in controlling an ESP subject to disturbances.

1.4 Structure of the Dissertation

The dissertation consists of seven chapters and six appendices, documenting all aspects of the work conducted during the Master's project. This first introductory chapter is followed by:

Chapter 2 covers the essential theoretical concepts necessary for understanding the rest of the dissertation. The relevant literature is cited using APA-in-text citations, and a complete list of references is provided at the very end of the dissertation.

Chapter 3 documents the baseline implementation. These serve as the foundation for the ultimate system implementation.

Chapter 4 documents the enhanced implementation, where the components from Chapter 3 are combined. This chapter also details the experimental designs and explains how the hyperparameters for the ESN models were determined.

Chapter 5 provides a summary of the results obtained from the experiments detailed in Chapter 4.

Chapter 6 discusses the results and the experiments in detail.

Chapter 7 concludes the dissertation by addressing the main objective and the research questions from Section 1.2. The conclusions are based on the discussion in Chapter 6. In addition, some subjects of interest for future work are proposed.

Appendix A provides a overview of fundamental theory on dynamical systems. This appendix serves as a supplement for the unfamiliar reader.

Appendix B covers selected essential topics within classical control theory. This appendix serves as a supplement for the unfamiliar reader.

Appendix C contains the plots from the reverse step NMPC experiments discussed in Section 3.2.

Appendix D details how the hyperparameters for the normal model in Section 5.1 were determined.

Appendix E details how the hyperparameters for the model in Section 5.2 were determined.

Appendix F details how the hyperparameters for the model in Section 5.3 were determined.

2 Essential Theoretical Concepts

This chapter gives an introduction to existing theory on essential concepts that are important for the understanding of this dissertation. The following sections will cover:

Section 2.1: Model Predictive Control (MPC) which is an optimal control strategy utilizing a system model to find optimal control inputs based on the current state and feedback.

Section 2.2: Fundamentals on the ESP which is the system subject to control in this dissertation.

Section 2.3: Stochastic processes, more specifically white uniformly and normally distributed noise.

Section 2.4: Fundamentals on the ESN which is the neural network (NN) that will be used to approximate the inverse model of the ESN.

Section 2.5: Error metrics used to evaluate system performances in this dissertation, more specifically mean squared error (MSE), mean absolute error (MAE), and integral absolute error (IAE).

It is assumed that the reader is familiar with dynamical systems modeling and classical control theory. Complementary theory on these topics is available in Appendix A and B, respectively, for the unfamiliar reader.

2.1 Model Predictive Control

The large interest in MPC became evident in the late 1970s (Camacho & Bordons, 2007). Particularly after the publication of the first papers on dynamic matrix control (DMC) and generalized predictive control (GPC) (Morari & Lee, 1999). Although the applications for these two methods were very different, they are both regarded as first-generation MPC strategies, sharing many of the same underlying ideas (Seborg, Edgar, & Mellichamp, 2004). DMC was developed in the early 80s to handle multivariable constrained control problems within the oil and chemical industries. Within these industries it was typical to use deterministic time-domain models without considering disturbance at all (Morari & Lee, 1999). GPC, on the other hand, was developed in the late 80s for adaptive control where transfer functions and stochastic models were more common (Morari & Lee, 1999). However, in the literature, the attempts on understanding the DMC are usually considered as the initial MPC research (Morari & Lee, 1999).

MPC has ever since the early 80s been an important subject for research due to the large potential it carries within process control, economics (Camacho & Bordons, 2007), climate change (Kim et al., 2022), and many others. Over the same period, there has also been a massive increase in available computational power. Especially the developments over the last 20 years have together with MPC enabled control of systems that were previously unimaginable (Schwenzer, Ay, Bergs, & Abel, 2021). Another major advantage of the MPC is its intuitiveness, and it is relatively easy to tune compared

to other controllers. This makes it desirable even in the case where the operators have limited knowledge to control (Camacho & Bordons, 2007).

2.1.1 The MPC Principle

Recalling that both DMC and GPC are considered MPCs, it is clear that MPC does not refer to a single controller or strategy, but to multiple advanced control strategies. According to (Camacho & Bordons, 2007), there are three main ideas that all MPC strategies to some extent share:

1) Predict the future process output over a finite time horizon explicitly by utilizing a process model.

2) Finding an optimal control sequence from minimizing an objective function.

3) Using the receding horizon strategy which involves using only the first control element calculated in 2) for each timestep.

The different MPC strategies vary in the type of process model they use to predict future outputs and the objective function they aim to minimize (Camacho & Bordons, 2007). However, the main challenge with any MPC strategy is to derive the process model mathematically (Camacho & Bordons, 2007). This is because many industrial processes have become severely complex over the last decades. Thus, many process models are impossible to derive based on first principles and require that parts of or eventually that the entire model is estimated from process data (Hou & Wang, 2013). In the case where a sufficiently accurate model is available, model imperfections (including unmeasurable disturbances) will cause drift in the states. This is because each individual MPC optimization is a feedforward that depends on being initialized close to the actual process state (Hovd, 2022). Nevertheless, it is possible to achieve feedback if the MPC is run for every time sample using the current state as starting point (this will require feedback from the output). This controller is often referred to as the *MPC principle* and is illustrated in Figure 1.

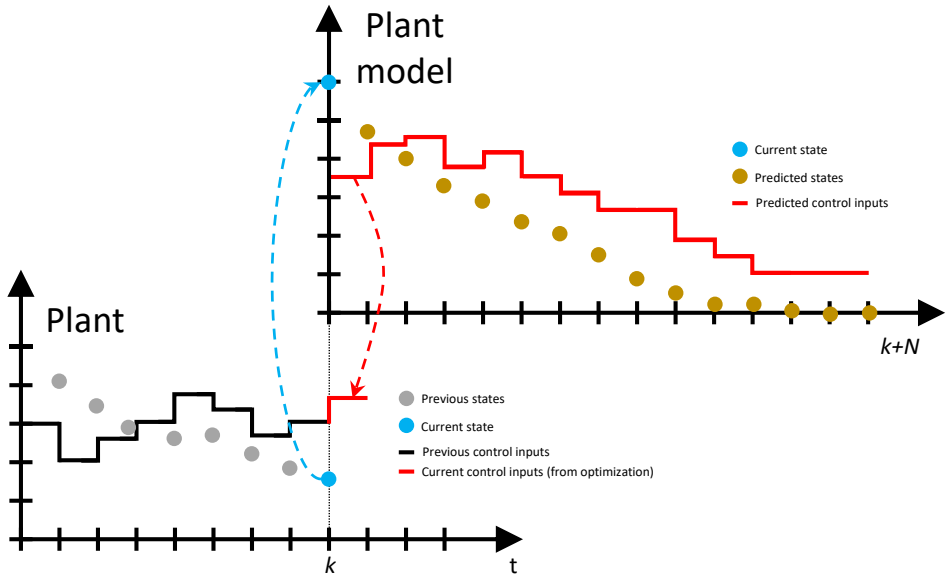


Figure 1: MPC principle, recreated from (Foss & Heirung, 2016).

Any stabilizable and detectable system can be stabilized and controlled with MPC by applying this principle if provided with a perfect model and an infinite horizon (Camacho & Bordons, 2007). However, this applies only in theory because there will always be some imperfections in any process model, and the computational demand for very long horizons would make the calculation infeasible. Yet, if a process model is available, imperfections can be handled to some extent by utilizing model update methods (Hovd, 2022). These are powerful, but advanced methods to suppress modeling imperfections including unmeasurable disturbances. The general idea is to design an observer such as the Kalman filter which is used to estimate the current state. This estimation is based on a weighting between the predicted and measured system output. Note that the observer will also require tuning and estimates of the stochastic properties of the disturbance.

Despite the challenge of deriving a process model, MPCs have proven to be reasonable strategies for industrial control (Foss & Heirung, 2016). The MPC has in many cases improved the performance of industrial processes massively through better tracking and increased disturbance rejection. This is because it enables tighter performance specifications compared to classical controllers such as the PID (Foss & Heirung, 2016). It is, however, typical to implement MPC on top of classical controllers forming a control hierarchy in modern industry. This setup ensures a stable system as systems with basic control in most cases are stable (Hovd, 2022). This is beneficial to many industries because they can implement the MPC on top of systems they already have invested in. Furthermore, the classical controllers will typically compute new control inputs every second while the MPC computes setpoints (in some cases even control parameters) for the classical controllers with a sampling time of minutes or more (Foss & Heirung, 2016). By being able to run the MPC on larger sampling times, it has become available for more industries. This is because it can be implemented without requiring upgrading process control computers. These computers are often old and expensive to upgrade. Consequently, they have poor computing power and are often

being used for communications, alarms etc. too (Camacho & Bordons, 2007).

2.1.2 Optimization Problems

Optimization refers in general mathematics to the minimization (or maximization) of some function with constrained variables (Nocedal & Wright, 2006). This function is usually referred to as the *objective* or *cost* function and is a scalar measuring the performance of the system being optimized. The objective is dependent on the problem and can for instance be time, profit, energy, or some combination of different entities. When optimizing an objective subject to some constrained variables, the goal is to find the variables that minimize (or maximize) the objective. A general optimization problem can be expressed as

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}) \quad \text{subject to} \begin{cases} c_i(\boldsymbol{x}) = 0, & i \in \mathcal{E} \\ c_i(\boldsymbol{x}) \geq 0, & i \in \mathcal{I} \end{cases} \tag{1}$$

where $\boldsymbol{x}$ is a vector with variables, f is a scalar objective function to be optimized and c_i are scalar constraint functions of $\boldsymbol{x}$ defining equalities ($\mathcal{E}$) and inequalities ($\mathcal{I}$) that $\boldsymbol{x}$ must satisfy (Nocedal & Wright, 2006).

Depending on the object function and the constraints, optimization problems can be divided into classes. These classes also imply which solving algorithm that should be used in order to solve the optimization problem. The three most general classes are:

- *Linear Programming (LP)* where both the objective function and the constraints are linear. These problems are always convex and can be solved efficiently using the SIMPLEX method or an interior point algorithm.
- *Quadratic Programming (QP)* where the objective function is quadratic and the constraints are linear. These problems are convex as long as the Hessian matrix is positive semidefinite and can be solved efficiently using an active set method. In the case where the Hessian is indefinite, the problem becomes a nonconvex QP which is generally harder to solve due to several stationary points and local minima.
- *Nonlinear Programming (NLP)* where either the objective function, one of the constraint functions, or all are nonlinear. These problems are in general nonconvex and can be very challenging to solve. Popular solvers for NLP problems are sequential quadratic programming (SQP) algorithms and the interior point optimizer (Ipopt).

When solving an optimization problem, the algorithm will search for an optimal solution within the *feasible* region. This region is defined by the constraints and all candidates within this region are contained inside the feasible set. In a case where no candidate fulfills all the constraints, this set will be empty which means the problem is infeasible (has no solution). Furthermore, LP and NLP problems can also be unbounded. This happens when the objective function f is able to become arbitrarily small (minimization) or large (maximization) without violating a constraint.

2.1.3 Objective Functions in Optimal Control

In the context of optimal control, the overall objective is to penalize deviations from a desired reference trajectory ($\boldsymbol{x}_{ref}$) and a desired input trajectory ($\boldsymbol{u}_{ref}$). Both trajectories are in the general case assumed to be given to the MPC from some external source. This leads to the following very general objective function formulation

$$\begin{aligned} J(\boldsymbol{x}, \boldsymbol{u}) = \sum_{k=1}^{\infty} &(\boldsymbol{x}_k - \boldsymbol{x}_{ref,k})^T Q (\boldsymbol{x}_k - \boldsymbol{x}_{ref,k}) \\ &+ (\boldsymbol{u}_k - \boldsymbol{u}_{ref,k})^T R (\boldsymbol{u}_k - \boldsymbol{u}_{ref,k}) \end{aligned} \tag{2}$$

where Q and R are symmetric and positive definite weighting matrices that can be used for tuning of the controller. However, since this formulation uses an infinite horizon, this objective function takes all future steps into account. This is a typical approach for unconstrained optimal controllers such as the linear quadratic controller (LQR). In comparison, the MPC also considers constraints and solves the optimization problem repeatedly. An infinite horizon is thus computationally infeasible as described initially. For this reason, the MPC uses a finite horizon which results in the following objective function:

$$J(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}}) = \sum_{k=1}^{N-1} \tilde{\boldsymbol{x}}_k^T Q \tilde{\boldsymbol{x}}_k + \tilde{\boldsymbol{u}}_k^T R \tilde{\boldsymbol{u}}_k \tag{3}$$

where $\tilde{\boldsymbol{x}}_k = (\boldsymbol{x}_k - \boldsymbol{x}_{ref,k})$ and $\tilde{\boldsymbol{x}}_k = (\boldsymbol{u}_k - \boldsymbol{u}_{ref,k})$, Q and R are the weighting matrices as previously described and N is the prediction horizon. In some applications, there might be necessary to append J with a final term given by $(\boldsymbol{x}_N - \boldsymbol{x}_{ref,N}) S (\boldsymbol{x}_N - \boldsymbol{x}_{ref,N})$ in order to ensure a feasible state at the end of the prediction. This is, however, outside the scope of this dissertation.

The MPC is tuned by selecting Q, R, and N. Typically, Q and R contain a weight for each state on the diagonal. Large elements in Q leads to more aggressive control since a small value in x will in the calculation of the cost get amplified. Thus, a large value will punish a deviation harder. In the opposite case, the deviation is ignored as a low value will drive the respective term closer to zero. R punishes the use of control, meaning that a large weight will lead to less aggressive control. This is because the optimizer will try to find control inputs close to the previous in order to minimize the difference. Both Q and R are typically chosen based on what behavior is desirable and by simulating different values. The prediction horizon N affects the performance and computational complexity of the controller. A low value of N corresponds to a short prediction giving poorer performance, but less complexity. A large value of N will give the opposite behavior. Generally, N is chosen as large as computationally possible and preferably larger than any time constants in the system. This way the finite prediction horizon acts as if it was infinite.

2.1.4 The Role of Constraints in MPC

One of the major advantages of applying MPC is the ability to achieve optimal control subject to some constraints. After all, any process is restricted by some physical limi-

tations. This can be the frequency range of a motor, a valve opening, etc. Today, also economic and environmental concerns are considered when designing an industrial process and the associated control system. Constraints for the general formulation given in (3) are typically given on the following form:

$$\underline{\boldsymbol{u}} \leq \boldsymbol{u}_{k+i} \leq \overline{\boldsymbol{u}}, \quad i \in [0, N-1] \tag{4}$$

$$\underline{\boldsymbol{x}} \leq \boldsymbol{x}_{k+i} \leq \overline{\boldsymbol{x}}, \quad i \in [0, N] \tag{5}$$

$$\underline{\Delta \boldsymbol{u}} \leq \Delta \boldsymbol{u}_{k+i} \leq \overline{\Delta \boldsymbol{u}}, \quad i \in [0, N-1] \tag{6}$$

where $\Delta \boldsymbol{u}_k = (\boldsymbol{u}_k - \boldsymbol{u}_{k-1})$ which is the same as constraining the change in control inputs, and $\underline{\cdot}$ and $\overline{\cdot}$ denote *min* and *max*, respectively.

Generally, there exist two types of constraints. These are *soft* and *hard* constraints. As the names suggest, soft constraints can to some degree be violated, while hard constraints can not. More specifically, soft constraints are modified constraints from introducing penalty functions in the optimization problem. This is achieved when introducing additional variables ensuring that the constraints are always feasible for sufficiently large values of the additional variables. The objective function is also modified by introducing a penalty term on the magnitude of the constraint violations making the introduced additional variables free variables in the optimization. This is often referred to as *relaxing* a constraint (Hovd, 2022). Note that a soft constraint must be possible to violate without saturating (or even destabilizing) the process. This can for instance be a constraint with the intention of minimizing climate gas emissions from the process. Hard constraints, on the other hand, will result in infeasibility if violated and is thus not relaxable. Hard constraints are most eminent in input constraints such as the limitations of an actuator.

2.2 Electric Submersible Pump

This chapter is mainly based on (Takács, 2009) and summarizes the theory on ESPs from (Grønningsæter, 2022). The reader is referred to these two sources for further details on this topic.

2.2.1 Artificial Lifting

Oil wells (reservoirs) are generally categorized as alive or dead based on how oil can be produced from them. An alive oil well has a sufficiently high natural pressure which ensures a proper flow of oil, water, and gas (production fluid) to the surface. These wells are often referred to as having "natural lift", and they are naturally suited for economical production. A dead oil well has lost its natural lift due to the fluid being extracted from the well over time. This will eventually cause a significant decrease in the natural pressure making the natural lift insufficient for economical production. However, it is possible to revive dead wells. Artificial lifting describes various methods and technologies that can be used for this purpose. One such technology is the ESP (Pavlov, Krishnamoorthy, Fjalestad, Aske, & Fredriksen, 2014) which is an installation based on pump lifting which, together with gas lifting, is considered one of the main methods to achieve artificial lift.

Pumping methods use pumps in the well bottom to increase the pressure in the reservoir to overcome the flowing pressure loss. In general, pumping methods are classified based on how the pump is driven and whether it uses rod or rodless pumping. The ESP is a rodless pumping method that utilizes a submerged electrical motor that drives a multistage centrifugal pump. The ESP is ideally suited to produce high liquid volumes and is one of the most efficient pumps when considering all depths. Regardless, it should be mentioned that gas lifting is still more efficient.

2.2.2 History and Challenges

The ESP was invented by Armiais Arutunoff in the late 1910s, and the first ESP was successfully operated in the El Dorado field, Kansas in 1926. Since then, there have been multiple improvements to the technology, and it is approximated that around 10% (Takács, 2009) of the world's oil supply is produced with ESPs today. The ESP is therefore considered by many as one of the most successful techniques for artificial lifting both on- and off-shore with high liquid volumes from medium depths. This is a result of the ESPs being both energy efficient and easy to install with low demand for maintenance as long as it is properly installed and operated.

There are, however, some disadvantages with the ESP. One of these is the difficulty of repairing faulty equipment in the field. In most cases, faulty equipment must be sent back to the manufacturer for repair which means downtime in the production since the installation can not be replaced during this period. To make matters worse, the repair in itself is expensive and comes in addition to the multi-million loss due to the downtime. Hence, it is desirable to make the lifetime of the ESP as long as possible and the need for maintenance as small as possible.

Statistics from *Centrilift* presented in (Pavlov et al., 2014), report that 23% of all ESP failures are caused by operator mistakes. Although this is a large improvement from the early days when around 80% of all failures were due to human errors, this is still causing enormous economical losses. The reason for these failures is the ESPs being run close to the operating limits because this is where optimal production points are located. There is in general little automation supporting the operation of the pumps. Many ESPs are driven by variable speed drivers (VSD) enabling operators to run the ESP motor at different speeds. They are also able to adjust the opening of the production choke at the top of the well affecting both the well and the operation of the ESP. Hence, it is hard to find an optimal operation point where the production is optimized and the lifetime is not significantly reduced. There are usually large teams with competence in the ESP monitoring multiple variables and doing analyses to support the operators. However, this is prone to human errors due to differences in experience, and the fact that most fields operate multiple ESPs making the task even more complex (Pavlov et al., 2014).

2.2.3 Components and Assumptions in the Conventional ESP

As mentioned in Subsection 2.2.1, ESPs are installations utilizing a submerged electrical motor (powered by electricity from the surface) that drives a multistage centrifugal pump. Generally, the ESP unit is submerged in well fluid and consists of a motor, a protector, a gas separator, and a pump. On the surface, one can usually find a junction box where surface and submerged electric cables are joined as well as a control unit. Although there have been multiple improvements to the ESP over the years, the conventional installation is still frequently used based on the following three assumptions:

1. Ideal pump conditions with only fluid entering the pump.
2. Produced fluid has a low viscosity (ideally close to the viscosity of water).
3. The motor is operating at a constant speed as it is powered with AC having a constant frequency. Also ensuring constant speed of the pump.

Even though these assumptions do not always hold, the conventional ESP installation has still proven both reliable and effective in various conditions. In special cases, where for example the gas production is too large and breaks the first assumption, the ESP installation can be augmented with special equipment. Installations different from the conventional are, however, outside the scope of this dissertation.

2.3 White Noise as a Stochastic Process

Noise as defined in Appendix B.2 can only be described in terms of probability (Brown & Hwang, 2012). This is because noise takes the form of a random process that can not be described using explicit mathematical functions such as sine waves, logarithms, etc. The foundations of this field were not laid before in the 1940s with the work of Wiener (Wiener, 1949) and Rice (Rice, 1944). Today they are regarded as pioneers within the field, and their work has proven crucial in many disciplines such as signal processing, which has played a major part in many modern technological breakthroughs.

One particularly important random process is *white noise*. This is a random process which by definition is stationary and has a constant spectral density function (Brown & Hwang, 2012). The noise is referred to as *white* with inspiration from optics meaning that it contains all frequencies. White noise will consequently have infinite variance making it physically unrealistic. Yet, it is physically valid to drive a system with white noise because all physical systems are bandlimited, yielding a process with finite variance (Brown & Hwang, 2012). A simpler interpretation of white noise is that it describes any sequence of uncorrelated samples with zero mean and finite variance. There are thus various types of white noise differing in the distribution from which the samples are drawn. This is commonly referred to as the noise having different probability density functions. The normal (also known as Gaussian) and uniform distributions are two particularly important distributions that will be introduced in the following two sections.

2.3.1 Normal White Noise

White noise is called *normal* when the samples are drawn from a normal (also known as a Gaussian) distribution. A random variable $\mathcal{X}$ is considered normal if it has the following probability density function (Brown & Hwang, 2012):

$$f_{\mathcal{X}}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{7}$$

where μ denotes the mean and σ denotes the standard deviation of the random variables. The normal distribution describes many naturally occurring random phenomena and is often encountered in applied probability (Walpole, Myers, Myers, & Ye, 2010). Due to being probably the most important distribution within statistics, normal random variables are often defined using the random variable $\mathcal{X}$ with the following notation:

$$\mathcal{X} \sim \mathcal{N}(\mu, \sigma^2) \tag{8}$$

Note that in the context of normal white noise, $\mu = 0$ by definition. The resulting probability density function from the density function $f_{\mathcal{X}}(x)$ is known as the *bell curve*. Figure 2 shows three bell curves with increasing standard deviation σ.

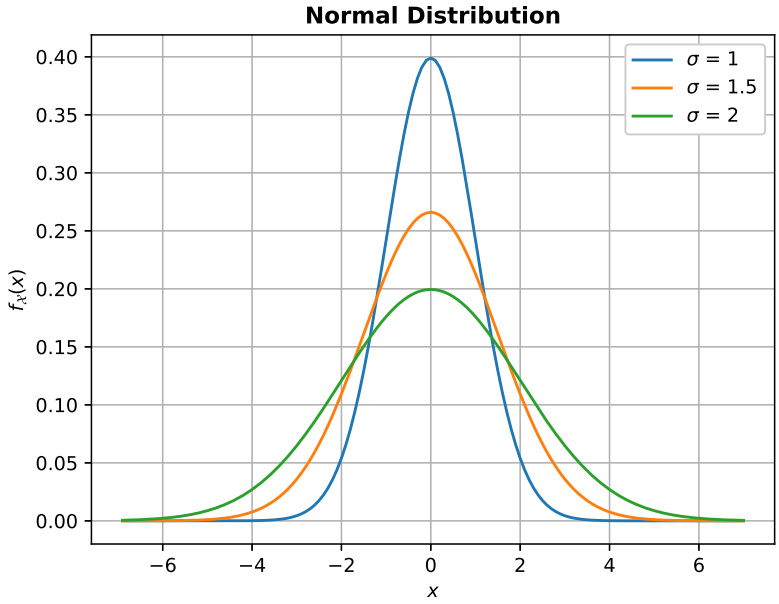


Figure 2: Bell curve plotted for different σ values, with zero mean (μ).

The area under the bell curve is always one and represents the probability of drawing a particular x from the entire interval. By integrating the bell curve over a subset, one can obtain the probability of drawing an x value from that particular subset. From further inspection of Figure 2, it is clear that the most probable values lie in the middle of the interval. Moreover, the probability of drawing a value away from the middle declines rapidly.

2.3.2 Uniform White Noise

An *uniform* distributions refer to a random variable $\mathcal{X}$ where all values between a maximum and minimum have the same probability of being drawn. The probability density function of this distribution is given by (Walpole et al., 2010):

$$f_{\mathcal{X}}(x) = \begin{cases} \frac{1}{B-A}, & A \leq x \leq B \\ 0, & else \end{cases} \tag{9}$$

where A and B denote the minimum and maximum of the interval. This distribution is one of the simplest probability distributions, and it does not occur as frequently as the normal distribution (Walpole et al., 2010). Figure 3 shows a uniform distribution plotted for the closed interval $[-1, 1]$.

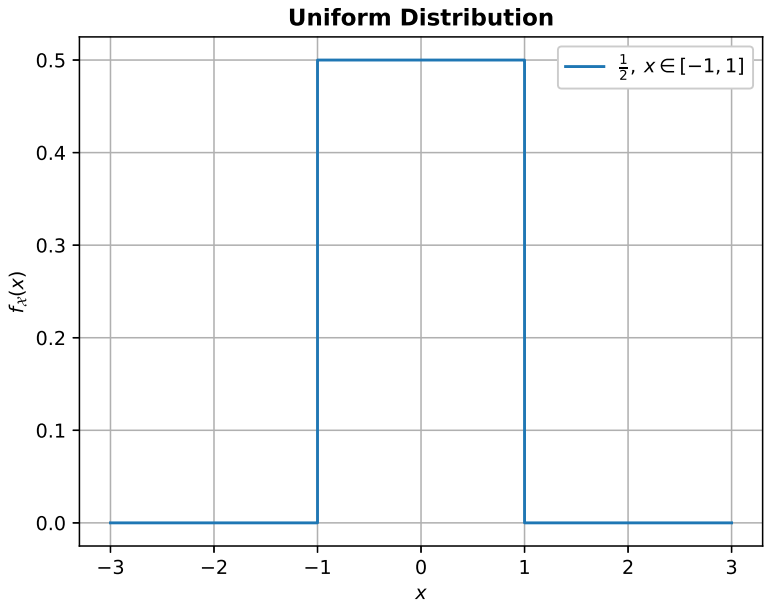


Figure 3: Uniform distribution for a random variable on the interval $[-1, 1]$.

As for the normal distribution, the area under the graph equals one. Hence, the probability of drawing an x value from a given subset is also here given by integrating over the particular subset. However, due to all values having the same probability, any uniform distribution takes the form of a rectangle. The probability of drawing a value within a given subset can thus be interpreted geometrically. Furthermore, the uniform distribution is usually defined by an interval and not the mean and standard deviation, which is the case for normal distributions. The mean μ and standard deviation σ^2 for an uniform distribution is given by (Walpole et al., 2010):

$$\mu = \frac{A + B}{2} \tag{10}$$

$$\sigma^2 = \frac{(B - A)^2}{12} \tag{11}$$

2.4 Echo State Networks

The introduction as well as Section 2.4.1 to 2.4.3 in this chapter has been extracted with some modifications from (Grønningsæter, 2022). The modifications are mostly to phrasings.

The ESN is an RNN based on reservoir computation instead of multiple hidden layers like more traditional neural networks. In reservoir computing, input and reservoir weights are randomly initialized entailing faster training and less computational cost compared to the RNNs where all weights are trained. This approach was proven eligible in (Schiller & Steil, 2005) where the authors showed that nearly all significant weight adaption happens in the output layer during the standard training of RNNs. The general ESN consists of a random input layer, the reservoir which acts as a single hidden layer, and an output layer that is trained. This is illustrated in Figure 4. ESNs can also be augmented with feedback to the reservoir. This is omitted in the figure because it is outside the scope of this dissertation.

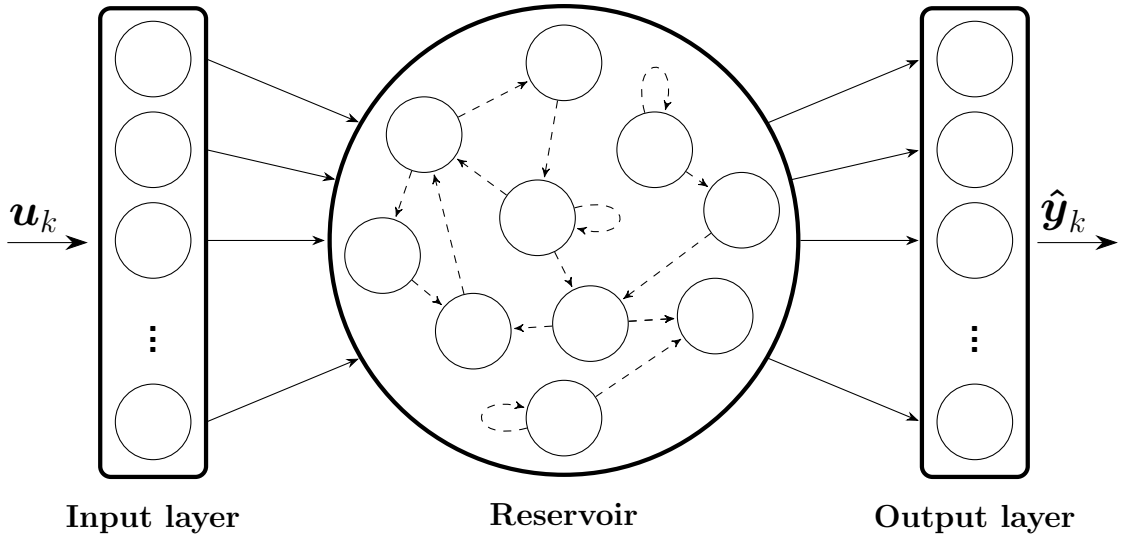


Figure 4: Schematic illustration of the general ESN, recreated from (Jaeger, 2007).

2.4.1 The Structure of the General ESN

ESNs can be considered specialized networks for fast learning of complex dynamical systems with low computational costs. This is not only due to the exploitation of reservoir computation but also the use of the supervised learning principle. The formal task of an ESN was formulated in (Jaeger, 2001) as:

Given a teacher I/O time series ($\boldsymbol{u}_{teach,k}$, $\boldsymbol{y}_{teach,k}$) for k=0,...,T, where the inputs come from a compact set $\boldsymbol{U}_{in}$ and the desired outputs $\boldsymbol{y}_{teach,k}$ from a compact set $\boldsymbol{U}_{out}$. A RNN whose output $\hat{\boldsymbol{y}}_k$ approximates $\boldsymbol{y}_{teach,k}$ is wanted.

The ESN can in the same way as RNNs be described in terms of a dynamical system. This system (without feedback) is given by the following equations:

$$\begin{aligned} \boldsymbol{x}_{k+1} &= (1-\alpha)\boldsymbol{x}_k + \alpha f(\boldsymbol{W}_r \boldsymbol{x}_k + \boldsymbol{W}_i \boldsymbol{u}_{k+1} + \boldsymbol{W}_b) \\ \boldsymbol{y}_{k+1} &= \boldsymbol{W}_o \boldsymbol{x}_{k+1} \end{aligned} \tag{12}$$

where $\boldsymbol{x}_k$, $\boldsymbol{u}_k$, $\boldsymbol{y}_k$ are the state of the reservoir neurons, the values of the input neurons and the values of the output neurons, respectively. $\boldsymbol{W}$ represents the weight matrix with the subscripts i, r, o and b being input, reservoir, output and bias. The remaining two properties, α and $f(\cdot)$ correspond to the leak rate and the nonlinear activation function (Lukoševičius, 2012). Furthermore, according to (Lukoševičius, 2012), the general method of reservoir computing introduced with ESNs is:

1. Define $\boldsymbol{W}_i$, $\boldsymbol{W}_r$ and α, and generate a random reservoir RNN.
2. Run the reservoir using the training input $\boldsymbol{u}_k$ and collect the corresponding reservoir activation states $\boldsymbol{x}_k$;
3. Compute $\boldsymbol{W}_o$ by minimizing the MSE between $\hat{\boldsymbol{y}}_{k+1}$ and $\boldsymbol{y}_k^{target}$.
4. Use the trained network on new input data $\boldsymbol{u}_k$ by computing $\hat{\boldsymbol{y}}_{k+1}$ using the computed $\boldsymbol{W}_o$.

2.4.2 Global ESN Reservoir Hyperparameters

When building an ESN, there are five hyperparameters that must be considered in order to optimize the reservoir of the ESN (Lukoševičius, 2012). These are reservoir size (N_x), sparsity, leak rate (α), spectral radius ($\rho(\boldsymbol{W}_r)$) and input scaling. Where the latter three are the most important for optimizing performance. The hyperparameter values are usually found through trial and error as there exist no analytic methods for this (Lukoševičius, 2012). Alternatively, it is also possible to conduct a grid search (Grimstad, 2022). The following sections are mainly based on (Lukoševičius, 2012) and will describe the hyperparameters in more detail.

Reservoir Size (N_x)

The reservoir size refers to the number of nodes in the reservoir. In general, one should use as many nodes as computationally affordable which makes it more likely to find a linear combination of reservoir signals $\boldsymbol{x}_k$ approximating $\boldsymbol{y}_k^{target}$. Yet, it can be cumbersome to determine the remaining hyperparameters if starting with a large reservoir. It is therefore recommended to start with a relatively small reservoir and determine the other hyperparameters before scaling up. Note that large reservoirs require appropriate regularization measures to avoid overfitting, and it is not always necessary to scale up the reservoir if a smaller reservoir yields satisfactory results.

Sparsity

Sparsity is relevant for both the input layer and the reservoir. In the input layer, it is recommended to make most of the values in $\boldsymbol{W}_i$ equal or as close to zero as possible. The performance of the ESN is generally not affected by sparsity in the reservoir. Hence, this hyperparameter is mostly used to speed up computations since it reduces the number of connections in the reservoir making the state matrix more sparse.

Spectral Radius ($\rho(\boldsymbol{W}_r)$)

The spectral radius is perhaps the most important hyperparameter because it scales the maximum absolute eigenvalue of the reservoir weight matrix. This is equivalent to scaling the width of the distribution of its nonzero elements. After $\boldsymbol{W}_r$ is generated randomly, it should be divided by $max(||eig(\boldsymbol{W}_r)||_1)$ ensuring unit spectral radius before it is scaled with the spectral radius. This is to ensure that the ESN satisfies the echo state property being that the reservoir is uniquely defined by the fading history of the input. Usually, the echo state property is ensured by choosing $\rho(\boldsymbol{W}_r) < 1$. For practical purposes, the spectral radius is selected to maximize the performance, and it determines how fast the influence of an input dies and the stability of the reservoir activation. Generally, the spectral radius should be greater in tasks requiring longer memory of the input.

Leak Rate (α)

The leak rate can be considered as the speed of the reservoir update dynamics. Reservoirs possess no time constant and use instead leaky nodes to slow down the dynamics in the system at hand. In (Osnes, 2020), the author describes the leak rate as *how much of the current state that will be "leaked" to the next.* This seems like a good interpretation considering its role in (12) and that its value ranges from zero to one.

Input Scaling

Input scaling determines the scaling of the input weight $\boldsymbol{W}_i$ affecting how nonlinear the reservoir responses are. Recalling that this is also affected by the spectral radius, these two parameters must be considered together. These two parameters regulate the amount of nonlinearity in $\boldsymbol{x}_k$ and the relative effect of the current input $\boldsymbol{u}_k$ on $\boldsymbol{x}_k$ opposed to history. In order to have few hyperparameters in the ESN, $\boldsymbol{W}_i$ should be scaled uniformly.

Bias Scaling

In (Grønningsæter, 2022), the bias weight $\boldsymbol{W}_b$ was fixed to one and included in the first column in $\boldsymbol{W}_i$. In this work, however, it is scaled separately to increase the performance of the reservoir as described in (Lukoševičius, 2012). In addition, it is also possible to further scale each column of $\boldsymbol{W}_i$ separately if the input sequence contributes differently to the task. This is not done in this dissertation because each scaling introduces a new hyperparameter that must be tuned.

2.4.3 Training of ESNs

In this paragraph, the training method of ESNs will be presented. Given a training set $\boldsymbol{u}_k$ and a target set $\boldsymbol{y}_{target,k}$ with $k = 1, ..., T$, and by including the bias weight in $\boldsymbol{W}_i$, (12) can, by using matrix notation, be rewritten as:

$$\boldsymbol{Y} = \boldsymbol{W}_o \boldsymbol{X}, \tag{13}$$

where $\boldsymbol{Y} \in \mathbb{R}^{N_y \times T}$ and $\boldsymbol{X} = [1 \quad \boldsymbol{u}_k \quad \boldsymbol{x}_k]^T \in \mathbb{R}^{(1+N_u \times N_x) \times T}$ are all outputs and states produced by the reservoir when presented all inputs $\boldsymbol{U} \in \mathbb{R}^{N_u \times T}$ (Lukoševičius, 2012). Finding the optimal $\boldsymbol{W}_o$ is equivalent to minimizing the MSE between the ESN output $\hat{\boldsymbol{y}}_k$ and the given target $\boldsymbol{y}_{target,k}$. Since $T \gg 1 + N_u + N_x$ in most cases, this is an overdetermined linear regression problem (Lukoševičius, 2012). One of the most common remedies when dealing with an overdetermined system is to use the ridge regression (also known as Tikhonov regularization):

$$\boldsymbol{W}_o = \boldsymbol{Y}_{target}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{X}^T + \beta\boldsymbol{I})^{-1} \tag{14}$$

where $\boldsymbol{Y}_{target} \in \mathbb{R}^{N_y \times T}$ is a matrix with every given $\boldsymbol{y}_{target,k}$ from $k = 1, ..., T$, β is the regularization coefficient and $\boldsymbol{I}$ is the identity matrix. This will avoid both overfitting and feedback instability (Lukoševičius, 2012) and solves the following optimization problem:

$$\boldsymbol{W}_o = \arg\min_{\boldsymbol{W}_o} = \frac{1}{N_y}\sum_{i=1}^{N_y}\left(\sum_{k=1}^{T}(\hat{y}_{i,k} - y_{target,i,k})^2 + \beta||\boldsymbol{w}_i^o||_2^2\right) \tag{15}$$

compared to the ordinary MSE, this object function also includes $\beta||\boldsymbol{w}_i^o||_2^2$ which is a regularization term that penalizes large $\boldsymbol{W}_o$ making a compromise between small training error and small output weights. β determines the relative importance and is a hyperparameter that must be tuned (Lukoševičius, 2012). There are multiple ways of determining β, with grid search being one of them (Grimstad, 2022). Note that (14) was transposed in the implementation in order to get it on the form $\boldsymbol{A}x = \boldsymbol{B}$ to exploit the *pytorch* linalg solver:

$$\boldsymbol{W}_o^T = (\boldsymbol{X}\boldsymbol{X}^T + \beta\boldsymbol{I})^{-1}\boldsymbol{X}\boldsymbol{Y}_{target}^T \tag{16}$$

NNs are traditionally trained by running through training data multiple times minimizing a cost function. The ESN differs from this because all data points are presented to the network once. This is often referred to as *one-shot training* (Lukoševičius, 2012). In most cases, this yields faster training of NNs. Moreover, when using ridge regression there are no limits to the amount of training data, and the time of the training procedure will be independent of the number of training points (Lukoševičius, 2012).

2.4.4 ESNs in Control Applications

Feedback controllers are often driven by estimates instead of the pure measurements of the system output (Chen, 1999). These estimators are often based on the inverse of the system model in order to predict the actual system state. However, in many applications, the system model is either unavailable or might be too complex to invert. To overcome these problems, the author of (Jaeger, 2008) proposes a black-box modeling framework utilizing the ESN to learn the inverse model of a system purely from observations of in- and output data. More specifically, this approach synthesizes a controller based on observations from random excitations of the ESN output(s). This idea was then used to make a feedback controller based on online learning of the inverse model in (Waegeman, Schrauwen, et al., 2012) and (Jordanou, 2019).

The original idea from (Jaeger, 2008) was to first train an ESN offline using the current and previous feedbacks y_k and $y_{k-\delta}$ as inputs and a random value $u_{k-\delta}$ as output. In this setup, δ denotes the delay in number of timesteps. This is shown schematically in Figure 5.

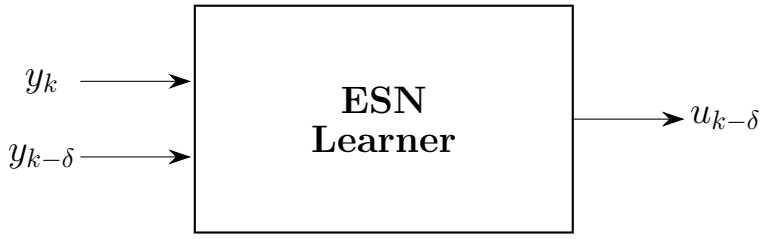


Figure 5: Schematic illustration of the learning phase of the ESN controller.

After the training phase, both the in- and output signals are shifted δ timesteps ahead. This results in the ESN taking $y_{k+\delta}$ and y_k as inputs producing u_k as output. The future input $y_{k+\delta}$ denotes the desired future input, which in the context of control, is the reference y^{ref}. A schematic illustration of the proposed ESN controller is shown in Figure 6.

Figure 6: Schematic illustration of the ESN controller.

The delayed feedback is required to enable the controller to use both the current and future (desired) feedback values as inputs. Furthermore, notice that the ESN-learner and ESN-controller illustrated in Figure 5 and 6 use the same model. After the training phase, the only difference between them is the origin of time of their inputs and output. In both (Waegeman et al., 2012) and (Jordanou, 2019), the general idea was to learn the network to produce the control output driving the system to the reference. They both managed to do so successfully, but both authors outline the importance of choosing δ carefully. The δ is proportional to the time constants of the system (Jordanou, 2019), and is therefore regarded as a crucial hyperparameter. Hence, finding a good δ is essential in order to find a good generalization of the inverse model. Generally, according to (Waegeman et al., 2012), systems with fast dynamics require a smaller δ, while slower dynamics require a larger value.

2.5 Error Metrics

Error metrics are necessary in order to evaluate the performance of any NN or controller. The MSE is a widely used error metric to evaluate the performance of an NN model on a test or validation set (Goodfellow, Bengio, & Courville, 2016). It is given by:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(\hat{y} - y)_i^2 \tag{17}$$

where N is the number of data points, $\hat{y}$ is the predicted value and y is the value target. From the equation it is evident that the MSE is really measuring the Euclidean distance between the prediction and the target:

$$MSE = \frac{1}{N}||\hat{y} - y||_2^2 \tag{18}$$

Lower MSE, therefore, implies better generalization as the MSE approaches zero when the predicted value $\hat{y}$ approaches the target value y. Generally, the MSE is a popular error metric for tuning hyperparameters. It should still be mentioned that the MSE is scale dependent, and also that it is vulnerable to outliers. This makes the MSE an undesirable metric for comparing NN models with different scaling.

In order to evaluate the performance of the controller, one can utilize the MAE (e_T) which is defined in (Hafner & Riedmiller, 2011). This metric is given by:

$$e_T = \frac{1}{N}\sum_{i=1}^{N}||x_i^{ref} - x_i||_1 \tag{19}$$

where N is the number of data points and x_i^{ref} and x_i are the state reference and state value at time step i. Although this metric is based on the $\mathcal{L}_1$ norm, it is based on the same intuition as the MSE since both error metrics average the distance between some reference and a calculated value over all data points. Hence, one could definitely apply the $\mathcal{L}_2$ norm instead. In fact, using MAE to evaluate the performance of controllers originates from using the same metric with the $\mathcal{L}_2$ norm as a term in cost functions for linear predictive control strategies (Camacho & Bordons, 2007). However, for the purpose of this dissertation, it will be used as defined in (19). Looking at the equation, it is evident that the MAE will grow with the time the system uses to converge to the reference. Thus, this error metric can also be used to indirectly evaluate the convergence rate. Another closely related error metric, that is often reported in similar studies, is the IAE. As the name suggests, this error metric measures the total area of the absolute error. The IAE is related to the average trajectory error e_T through the following relation:

$$IAE = N \cdot e_T \tag{20}$$

where N is the number of data points and e_T is the mean trajectory error. A drawback with the MAE and IAE is that neither of these metrics is able to evaluate the transient behavior (i.e. oscillations) of the controller (Jordanou, 2019). Therefore, a final error metric measuring the absolute variation of the control action is also defined:

$$\Delta u = \sum_{i=1}^{N} ||u_i - u_{i-1}||_1 \tag{21}$$

where N is the number of data points and u_i and u_{i-1} are the current and previous control input. The intuition behind this error metric is that large variations between control inputs are physically infeasible and can be damaging to the actuators. Hence, it is desired to follow the trajectory with as conservative use of control as possible.

3 Baseline Implementations

This chapter documents the baseline implementations in the project. These will serve as crucial and individual parts in the adequate system implementation. The following sections will cover:

Section 3.1: The mathematical model and implementation of the conventional ESP lifted well.

Section 3.2: Mathematical formulation and implementation of the NMPC.

3.1 Conventional ESP Lifted Well

This subchapter covers the mathematical model of the conventional ESP as well as the implementation of this. Section 3.1.1 and 3.1.2 are both extracted from (Grønningsæter, 2022). Nevertheless, the presentation of the equations in the former is rewritten to resolve some unclarities and misunderstandings from (Grønningsæter, 2022). The reader is referred to this source for experiments confirming the integrity of the model implementation.

3.1.1 Mathematical Model of the Conventional ESP Lifted Well

The ESP is, as described in Subsection 2.2.3, a complex system with multiple mechanical components. In addition, the well must also be considered in order to derive a mathematical model for optimizing operation and ultimately oil production. One such model is presented in (Pavlov et al., 2014) where it was used in an MPC and tested extensively at Equinor's R&D center in Porsgrunn, Norway. This model was originally developed in 2010 by the same authors in cooperation with Equinor (previously Statoil). In the same year, (Binder et al., 2015) enhanced this model to also include estimates of flow rates and viscosity in a similar well. This model, including the reported parameters and assumptions, is the model that will be used to implement the simulator in this project. An illustration of the model is presented in Figure 7. From inspection, it is evident that there are many variables and parameters that must be considered. All the reported numerical values are summarized in Table 1.

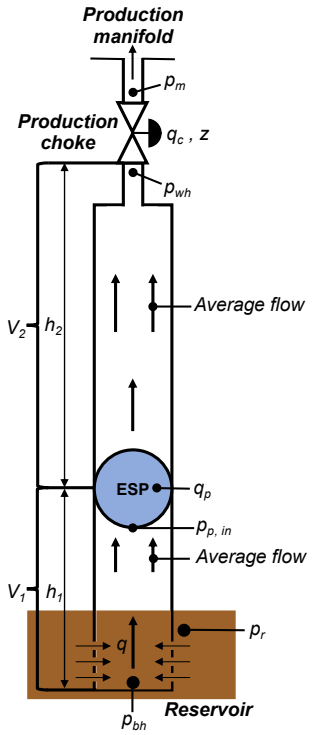


Figure 7: Mathematical ESP model, recreated from (Binder et al., 2015).

The resulting model from (Binder et al., 2015) is a third-order nonlinear model given by the following differential equations:

$$\dot{p}_{bh} = \frac{\beta_1}{V_1}(q_r - \hat{q}), \quad [Pa/s] \tag{22}$$

$$\dot{p}_{wh} = \frac{\beta_2}{V_2}(\hat{q} - q_c), \quad [Pa/s] \tag{23}$$

$$\dot{\hat{q}} = \frac{1}{M}(p_{bh} - p_{wh} - \rho g h_w - \Delta p_f + \Delta p_p), \quad [m^3/s^2] \tag{24}$$

where q_r, q_c, Δp_f, and Δp_p represent the inflow, outflow, friction, and pressure difference across the ESP, respectively. The remaining parameters are constants given in Table 1. Further details on these models are given in the following four subsections.

Inflow from the Reservoir into the Well (q_r)

The inflow rate from the reservoir into the well is given by the following equation:

$$q_r = PI(p_r - p_{bh}), \quad [m^3/s] \tag{25}$$

where PI is the well productivity index, p_r the reservoir pressure, and p_{bh} the bottomhole pressure. The two former parameters are constants given in Table 1, whereas the latter is the current ESP bottomhole pressure. This depends on the current inflow q_r and is found by solving (22).

Flow Rate through the Production Choke (q_c)

The flow rate through the production choke is given by the following equation:

$$q_c = C_c\sqrt{p_{wh} - p_m} \cdot z, \quad [m^3/s] \tag{26}$$

where C_c and p_m denote the choke valve constant and the production manifold pressure, respectively. These are constants given in Table 1. z denotes the valve opening which is a system control input. Lastly, p_{wh} is the current ESP wellhead pressure. This depends on the current outflow q_c and is found by solving (23).

Frictional Pressure Drop in the Well (Δp_f)

The frictional pressure drop in the well is given by the following equation:

$$\Delta p_f = F_1 + F_2, \quad [Pa] \tag{27}$$

where F_1 and F_2 denote the frictional pressure drop below and above the ESP, respectively. These values are given by:

$$F_i = 0.158 \cdot \frac{\rho L_i \hat{q}^2}{D_i A_i^2}\left(\frac{\mu}{\rho D_i \hat{q}}\right)^{\frac{1}{4}}, \quad [Pa] \tag{28}$$

where $\hat{q}$ represents the current flow through the ESP. This is found by solving (24). Furthermore, ρ and μ are the density and the viscosity of the produced fluid. An experimental value for the fluid density ρ is found in (Binder et al., 2015). This value is reported in Table 1. The fluid viscosity μ is reported as varying in the same source, and it was therefore chosen as 0.025 $Pa \cdot s$ for the purpose of the dissertation. This ensures a viscous fluid within the valid interval for the viscosity correction factors (VCF) included in (Binder et al., 2015). The VCFs are covered in the next subsection.

Pressure difference across the ESP (Δp_p)

The pressure difference across the ESP is given by the following equation:

$$\Delta p_p = \rho g H, \quad [Pa] \tag{29}$$

where ρ and g denote the density of the produced fluid and the gravitational acceleration, respectively. These parameters are constants given in Table 1. H is further given by the following equations:

$$H = C_H(\mu)H_0(q_0)\left(\frac{f}{f_0}\right)^2, \quad [m] \tag{30a}$$

$$q_0 = \frac{\hat{q}}{C_q(\mu)}\left(\frac{f_0}{f}\right), \quad [m^3/s] \tag{30b}$$

where f is the ESP motor frequency which is a system input, f_0 is the ESP characteristics reference frequency given in Table 1, and $\hat{q}$ is the current flow found from solving (24). $C_q(\mu)$, $C_H(\mu)$, and $C_P(\mu)$ denote the viscosity corrected flow rate, head, and

brake horsepower (BHP). The latter is an imperial measurement for the electric power consummation of the ESP motor. Moreover, $H_0(q)$ denotes the ESP head characteristics and is together with the VCFs assumed to be known for the particular ESP model. VCFs are usually obtained from published sources or empirically obtained, whereas the characteristics for the ESP are provided by the pump vendor (Binder et al., 2015). The VFCs and ESP characteristics are in (Binder et al., 2015) given by polynomials on the following form

$$P(x) = \sum_{i=0}^{4} c_i x^i \tag{31}$$

where the coefficients $(c_i, i = 1, ..., 4)$ are given in Table 2. These are the same as in the publication.

Variable	Description	Value	unit
Known constants			
g	Gravitational acceleration constant	9.81	m/s^2
C_c	Choke valve constant	$2 \cdot 10^{-5}$	$*$
A_1	Cross-section area of pipe below ESP	$8.107 \cdot 10^{-3}$	m^2
A_2	Cross-section area of pipe above ESP	$8.107 \cdot 10^{-3}$	m^2
D_1	Pipe diameter below ESP	$1.016 \cdot 10^{-1}$	m
D_2	Pipe diameter above ESP	$1.016 \cdot 10^{-1}$	m
h_1	Height from reservoir to ESP	200	m
h_w	Total vertical distance in well	1000	m
L_1	Length from reservoir to ESP	500	m
L_2	Length from ESP to choke	1200	m
V_1	Pipe volume below ESP	4.054	m^3
V_2	Pipe volume above ESP	9.729	m^3
ESP data			
f_0	ESP characteristics reference frequency	60	Hz
I_{np}	ESP motor nameplate current	65	A
P_{np}	ESP motor nameplate power	$1.625 \cdot 10^5$	W
Parameters from fluid analysis and well tests			
β_1	Bulk modulus below ESP	$1.5 \cdot 10^9$	Pa
β_2	Bulk modulus above ESP	$1.5 \cdot 10^9$	Pa
M	Fluid inertia parameter	$1.992 \cdot 10^8$	kg/m^4
ρ	Density of produced fluid	$9.5 \cdot 10^2$	kg/m^3
p_r	Reservoir pressure	$1.26 \cdot 10^7$	Pa
Unknown parameters			
PI	Well productivity index	$2.32 \cdot 10^{-9}$	$m^3/s/Pa$
μ	Viscosity of produced fluid	$2.5 \cdot 10^{-2}$	$Pa \cdot s$
H_0	ESP head characteristics	Varying	m
P_0	ESP BHP characteristics	Varying	W
q_0	Theoretical flow rate at reference frequency	Varying	m^3/s
C_H	VCF for head	Varying	–
C_P	VCF for brake horsepower of the ESP	Varying	–
C_q	VCF for ESP flow rate	Varying	–

Table 1: Model parameters used in the simulator. These values are obtained from (Binder et al., 2015).

	c_4	c_3	c_2	c_1	c_0
H_0	0	0	$-1.2454 \cdot 10^6$	$7.4959 \cdot 10^3$	$9.5970 \cdot 10^2$
P_0	0	$-2.3599 \cdot 10^9$	$-1.8082 \cdot 10^7$	$4.3346 \cdot 10^6$	$9.4355 \cdot 10^4$
C_q	2.7944	-6.8104	6.0032	-2.6266	1
C_H	0	0	0	-0.03	1
C_P	-4.4376	11.091	-9.9306	3.9042	1

Table 2: Coefficients used for VCFs and ESP characteristics. These values are obtained from (Binder et al., 2015).

3.1.2 Implementation of the ESP Lifted Well Simulator

The simulator implementation is inspired by (Osnes, 2020) and implemented using Python 3.10, *NumPy*, and *CasADI*. *NumPy* is an open-source framework for numerical computation in Python. This was mainly used for handling tensors inside the simulator. *CasADI* is also an open-source tool, but for nonlinear optimization and algorithmic differentiation (Andersson, Gillis, Horn, Rawlings, & Diehl, 2019). In the implementation, *CasADI* was utilized for solving the differential equations.

To simulate the ESP lifted well, the simulator takes the current state x and an input u and solves the initial value problem one time step ahead. The simulator thus requires an initial state x_0 which in this work was given by

$$x_0 = \begin{bmatrix} p_{bh,0} \\ p_{wh,0} \\ \hat{q}_0 \end{bmatrix} = \begin{bmatrix} 75 \cdot 10^5 \\ 30 \cdot 10^5 \\ 0.01 \end{bmatrix} \tag{32}$$

where the numerical values were inspired by (Osnes, 2020), but slightly changed after conducting some experiments. Note that both the initial pressures and the initial flow are given in Pa and m^3/s, respectively, whereas the later reported results are given in bar and m^3/h. The input u is a vector with the choke opening z in % and the ESP motor frequency f given in Hz:

$$u = \begin{bmatrix} z \\ f \end{bmatrix} \tag{33}$$

During the initial testing of the simulator, there were some issues with the *CasADI* integrator not converging due to encountering negative flow values. This is equivalent to the flow suddenly changing direction and is not consistent with the expected behavior of the ESP because it would require the motor or the head pressure to also change direction. Further investigation revealed that when solving (24), the flow value $\hat{q}$ became infinitesimally small leading to numerical instability. Hence, a minimum $\hat{q}$ value of 10^{-9} was enforced in the simulator, removing the instability and thus the convergence issues.

3.2 Nonlinear Model Predictive Control

The only difference between an NMPC and an MPC is that the system model of the former is nonlinear. This results in the optimization problem becoming an NLP. In the final implementation, the NMPC will control the ESP assisted by the ESN. It will also be used to generate training and validation data for the offline training of the ESN.

The controller was implemented in Python 3.10 using *NumPy* and the *Opti()* module from *CasADI* (Andersson et al., 2019). This module is an open-source framework for the formulation of optimization problems. The resulting problem was ultimately solved using *multiple shooting* and Ipopt. Multiple shooting means that the whole state and control trajectory are used as decision variables. For further details on this and other shooting methods, the reader is referred to (Imsland, 2007). Ipopt is a widely used open-source software for solving NLPs. For details on the solver, the reader is referred to (Wächter & Biegler, 2006)

3.2.1 Mathematical Formulation of the NMPC

The NLP solved each timestep in the NMPC minimizes the following quadratic objective function:

$$J(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}}) = \sum_{k=1}^{N-1} \tilde{\boldsymbol{x}}_k^T Q \tilde{\boldsymbol{x}}_k + \tilde{\boldsymbol{u}}_k^T R \tilde{\boldsymbol{u}}_k \tag{34}$$

where Q and R are weighting matrices, and $\tilde{\boldsymbol{x}} = \boldsymbol{x}_k - \boldsymbol{x}^{ref}$ and $\tilde{\boldsymbol{u}} = \boldsymbol{u}_k - \boldsymbol{u}_{k-1}$. The vectors $\boldsymbol{x}$ and $\boldsymbol{u}$ are further given by:

$$\boldsymbol{x} = \begin{bmatrix} p_{bh} \\ p_{wh} \\ q \end{bmatrix}, \quad \boldsymbol{u} = \begin{bmatrix} z \\ f \end{bmatrix} \tag{35}$$

The ESP model from Subsection 3.1.2 was implemented as a hard constraint, introducing the nonlinearity to the controller. Furthermore, the physical limitations of the model were inspired by those reported in (Binder, Kufoalor, Pavlov, & Johansen, 2014). These were implemented through the following inequality constraint:

$$\underline{\boldsymbol{x}} = \begin{bmatrix} 0 \text{ bar} \\ 1 \text{ bar} \\ 0 \ m^3/h \end{bmatrix} \leq \boldsymbol{x} \leq \begin{bmatrix} \infty \text{ bar} \\ 60 \text{ bar} \\ \infty \ m^3/h \end{bmatrix} = \overline{\boldsymbol{x}} \tag{36}$$

$$\underline{\boldsymbol{u}} = \begin{bmatrix} 10 \ \% \\ 35 \text{ Hz} \end{bmatrix} \leq \boldsymbol{u} \leq \begin{bmatrix} 100 \ \% \\ 65 \text{ Hz} \end{bmatrix} = \overline{\boldsymbol{u}} \tag{37}$$

It is reported in (Osnes, 2020) that z values $< 10\%$ result in the model being imprecise. Hence, the lower bound for the valve opening was set to 10%. In order to ensure feasibility when solving the NLP, the flow $\hat{q}$ from (Osnes, 2020) was also slacked.

The final NMPC formulation was given by:

$$\min_{\boldsymbol{x},\boldsymbol{u}} J(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}}) \text{ s.t.} \begin{cases} \boldsymbol{x}_0 = given \\ \boldsymbol{x}_{k+1} = f(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}}) \\ \underline{\boldsymbol{u}}_k \leq \boldsymbol{u}_k \leq \overline{\boldsymbol{u}}_k \\ \underline{\boldsymbol{x}}_k \leq \boldsymbol{x}_k \leq \overline{\boldsymbol{x}}_k \end{cases} \tag{38}$$

where f is the ESP model and J is the quadratic objective function (34). Assuming that $\boldsymbol{x}_0$ is given is a common assumption in the context of (N)MPC.

The behavior of the controller depends on the design of Q and R. Only the bottomhole pressure p_{bh} will be controlled in this dissertation. A weight q_1 was therefore introduced in the first diagonal element of Q. The two remaining diagonal elements were set to zero in order to prevent them from contributing to the objective function. The general Q matrix is thus given by:

$$Q = \begin{bmatrix} q_1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{39}$$

Generally, there are always time constants associated with actuators in real-world applications. It is reasonable to assume that it will take some time for the valve z to open and close. Therefore, the first diagonal element in R should be large to enforce a gradual change in the opening. The motor frequency f, on the other hand, is assumed to have a relatively lower weight. This is because ESP motors have exceptionally low inertia and an ability to utilize about ten times higher current densities because of fluid cooling (Takács, 2009). However, rapid changes that might increase the need for maintenance should also be avoided. The general R matrix is given by:

$$R = \begin{bmatrix} r_1 & 0 \\ 0 & r_2 \end{bmatrix} \tag{40}$$

Experiments to find suitable Q and R will be conducted in the following subsections.

3.2.2 Confirmation and Tuning of the NMPC

In this subsection, the implementation of the NMPC will be confirmed by conducting experiments on the controller using the nominal system model. Different values for the Q and R matrices will also be tested with an emphasis on finding values that ensure satisfactory behavior as well as feasible solutions to the NLP problem. Finally, an experiment where the valve opening z is fixed to 80% will be conducted. The purpose of this experiment is to determine the operational region for the setup where the ESP is controlled by only manipulating the motor frequency f. Finding this region will be crucial to avoid saturation when generating data for the ESN controller. Determining the operational region when using both control inputs is not necessary as this is already known from the experiments conducted in (Grønningsæter, 2022). All experiments will use a horizon of 10 samples (which corresponds to 1 second). The following initial guess will be used in the solver:

$$x_{init} = \begin{bmatrix} 75 \cdot 10^5 \\ 30 \cdot 10^5 \\ 0.01 \end{bmatrix}, \quad u_{init} = \begin{bmatrix} 0.5 \\ 45 \end{bmatrix} \tag{41}$$

These were decided arbitrarily within the feasible region to ensure that the Jacobi was well-defined. Figure 8 shows a schematic illustration of the system used in each of the conducted experiments.

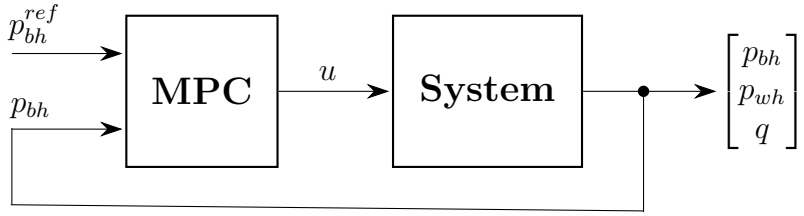


Figure 8: Schematic illustration of the system used in the experiments.

Experiment 1: Step Response Using Both Control Inputs

In this experiment, the response of the closed-loop system after a step in the state reference x^{ref} was investigated. The step yielded a change from 70 to 80 bar in the bottomhole pressure p_{bh}. These values were inspired by the resulting responses from control inputs within the normal operation rates reported in (Pavlov et al., 2014). A reference change of 10 bar is in itself unrealistic and not how the ESP should be operated.

However, the main objective of this dissertation is to design an optimal control system able to handle the output disturbance of the system. Hence, the controller should be able to bring the system to a distant reference without violating the physical limitations. At the same time, it should be able to achieve this without reducing the lifetime of the equipment significantly. Therefore, it was expected that a reference change of this magnitude would provide valuable insight into finding suitable weights for the R matrix.

During the testing of different values, it became evident that the q_1 weight had to be fairly small to maintain a well-defined Jacobi. Subsequently, this is to some extent obvious since the pressure is given in pascals, yielding values in a magnitude of millions. The value yielding the best response was $5 \cdot 10^{-8}$. Furthermore, the weights r_1 and r_2 were set to 10^5 and $3 \cdot 10^2$, respectively, due to the valve being incapable of the same rapid changes as the motor. These weights yielded the following Q and R matrices:

$$Q = \begin{bmatrix} 5 \cdot 10^{-8} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad R = \begin{bmatrix} 10^5 & 0 \\ 0 & 3 \cdot 10^2 \end{bmatrix} \tag{42}$$

The step response from using these weight matrices is shown in Figure 9.

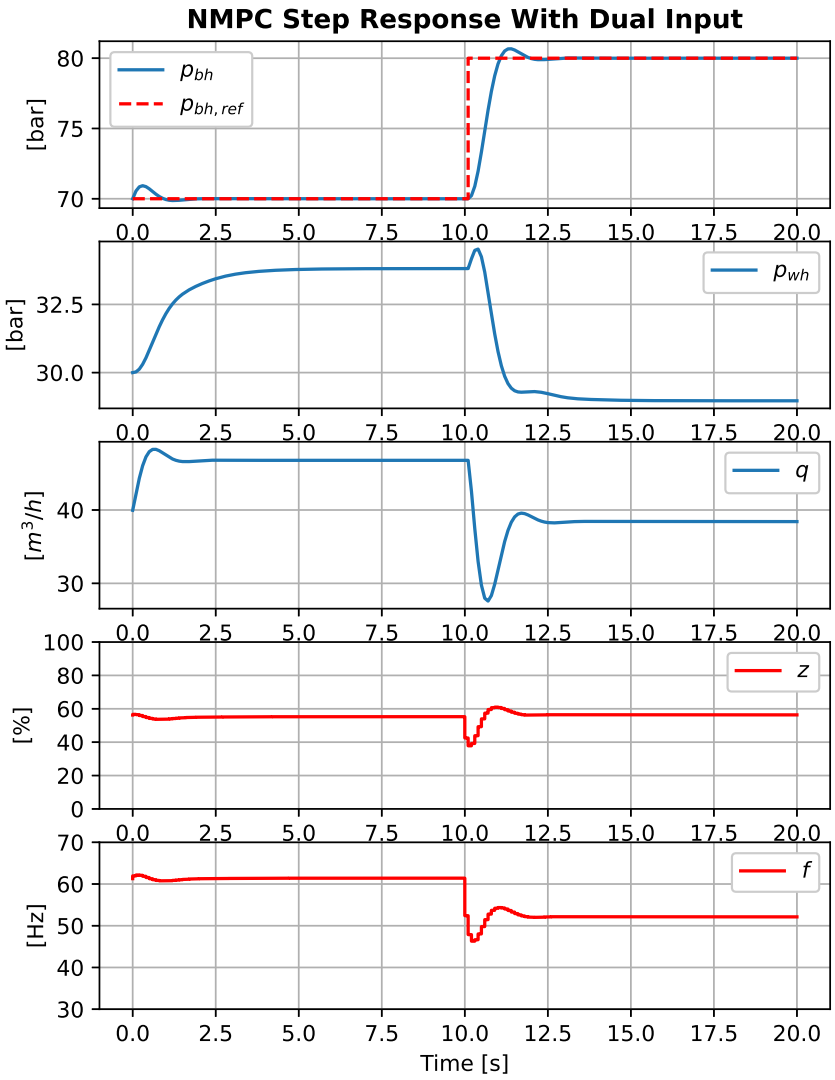


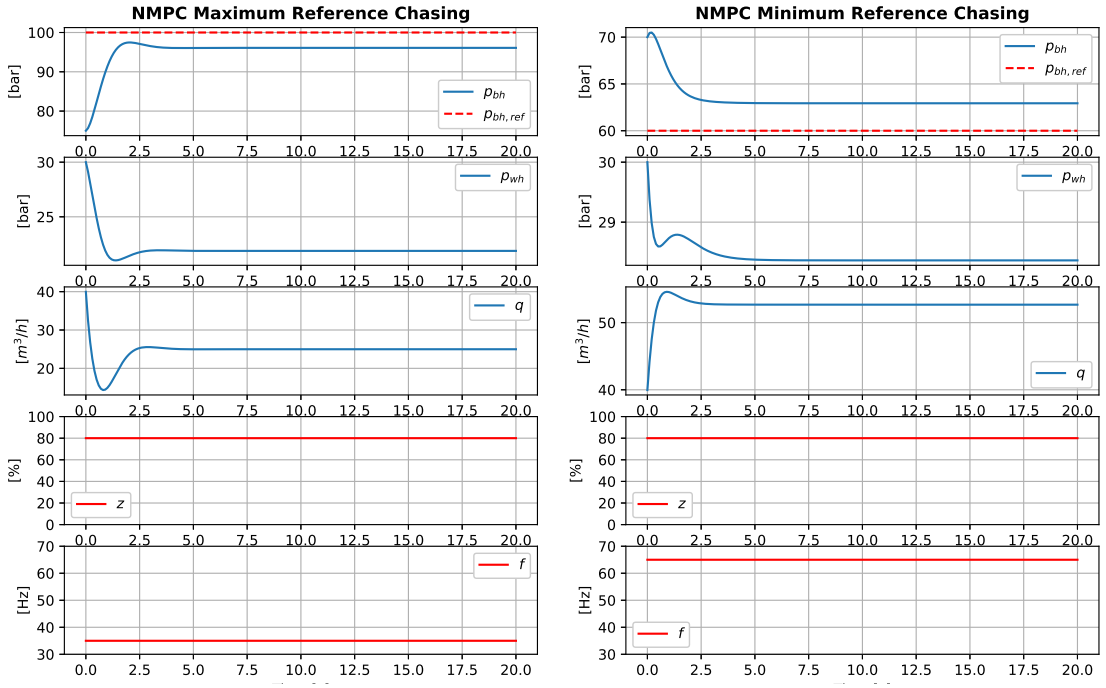
Figure 9: Step response from using both ESP control inputs and the weight matrices from (42).

It is evident from inspection of Figure 9 that the controller needs around a second to drive the system to the reference after the initial starting point. This is because the initial control input did not match the initial state. Furthermore, one can see a minor overshoot after the step in the reference. This is due to a trade-off between optimal control behavior and realistic changes in the control inputs. Further experimentation yielded that the overshoot could be removed completely if drastic changes were allowed in both inputs. Additionally, a reverse experiment with a step from 80 to 70 bar was also conducted. This yielded similar results, and the resulting plot is shown in Appendix C.1.

Experiment 2: Operational Region With f as Only Control Input

It is noteworthy that the r_1 in the resulting R matrix from the previous experiment is 99.7% larger than r_2. This suggests that the valve opening has a significant impact on

the control. Further inspection of Figure 9 supports this observation. Based on these observations, it was expected that fixing the valve opening z to 80% would limit the operational region of the controller. An experiment to investigate this and possibly decide the region was conducted by using reference chasing as described in (Osnes, 2020). The maximum and minimum of the region were found by running simulations using a reference x^{ref} of 100 and 60 bar, respectively. These values were found through a trial and error approach. Results from both experiments are shown in Figure 10.



(a) Chasing maximum reference.

(b) Chasing minimum reference.

Figure 10: Results from experiments using reference chasing to determine the operational region for a fixed valve opening z of 80%.

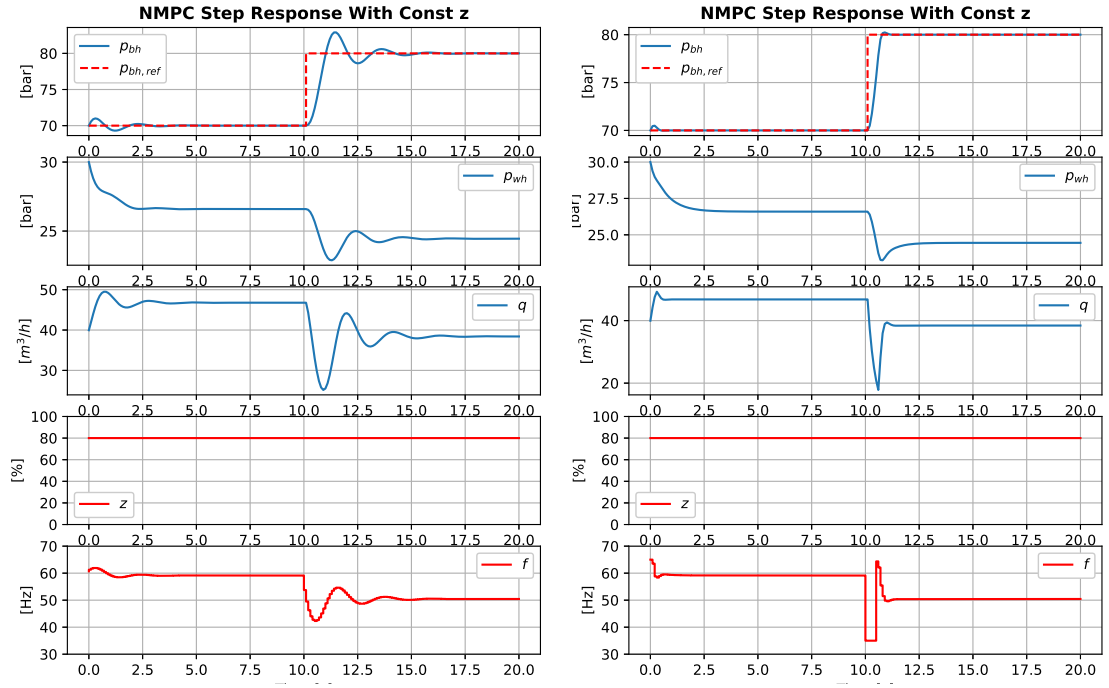
Inspection of Figure 10(a) yields a maximum bottomhole pressure p_{bh} of a little more than 95 bar. This happened when the controller saturated at 35 Hz. Similar behavior is also evident from inspecting Figure 10(b). This experiment yielded a minimum bottomhole pressure p_{bh} of a little less than 65 bar. This happened when the controller saturated at 65 Hz. The operational region for a fixed z of 80% was thus identified as $p_{bh} \in [65, 95]$.

Experiment 3: Step Response Using Only f as Control Input

In this final experiment, it was investigated if the Q and R matrices from (42) would result in satisfactory behavior also when the valve opening z was fixed. This experiment used the same step (from 70 to 80 bar) as in the first experiment.

When using $R = 3 \cdot 10^2$ (r_2 from the first experiment), the step yielded an oscillatory response. This is shown in Figure 11(a). Another experiment with $R = 1$ was also conducted to compare the impact of different R values. This yielded the step response

shown in 11(b). The Q matrix remained unchanged from (42) in all of the experiments to ensure consistency, but also feasibility in the NLP problem.



(a) NMPC step response with $R = 3 \cdot 10^2$.

(b) NMPC step response with $R = 1$.

Figure 11: Step responses from using only the motor frequency f as the control input. The plots show the responses from using a large (a) and a low (b) R weight on the motor frequency f.

Inspection of Figure 11(b) shows that $R = 1$ yielded a response with no oscillations. However, from further inspection, it is also clear that this requires the motor frequency to change from 60 Hz to 35 Hz in 0.1 seconds which is somewhat unrealistic. By comparing this response to the previous, it is evident that an R value yielding satisfactory behavior lies somewhere in between the tested R values.

After some trial and error, it was found that an $R = 60$ provided an acceptable response. This response contained fewer oscillations than $R = 3 \cdot 10^2$, and the controller used less aggressive changes in the motor frequency. The resulting step response is shown in Figure 12. A reverse step response yielded similar results. This is shown in Appendix C.2.

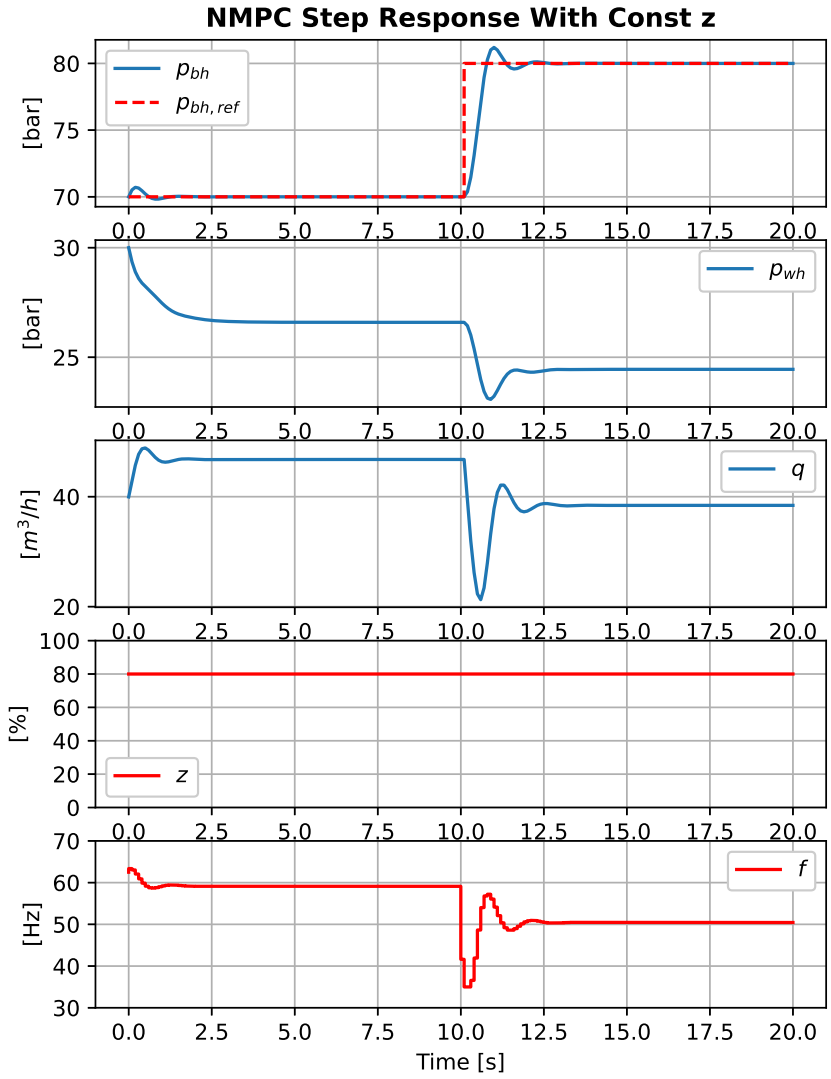


Figure 12: Step response from using only the motor frequency f as control input with $R = 60$.

4 Enhanced Implementation and Methodology

This chapter consists of three sections and documents the final controller implementation, the experimental designs, and the tuning of the ESN hyperparameters. In order to simplify descriptions and comparisons, the obtained ESN models will for the remainder of the dissertation be referred to in terms of the noise used to train them. More specifically, the three sections cover:

Section 4.1: The implementation of the ESN assisted optimal controller.

Section 4.2: The experimental designs.

Section 4.3: The strategy used for tuning the ESN hyperparameters.

4.1 Implementation of the Feedforward Assisted NMPC

The final controller will be used to control only one state, the bottomhole pressure p_{bh}, yet all states will be reported for completeness. In the proposed setup, the NMPC uses the nominal ESP model derived in (Binder et al., 2015). This model is also used in the ESP simulator, which, in addition, will experience a persistent disturbance affecting the reservoir pressure. Without any prior knowledge of the disturbance, the NMPC will be unable to counteract it efficiently. Furthermore, the system will be augmented with an ESN-based feedforward controller trained to produce a correctional control input u_2 based on the current reference p_{bh}^{ref}, the current state p_{bh}, and the current NMPC control input u_1. This control input will be added to the current NMPC control input u_1 to produce a corrected control input $u = u_1 + u_2$ that suppresses the disturbance in the reservoir pressure. Figure 13 provides a schematic illustration of the described implementation. The figure also includes a saturation function. This is to ensure the control input to the system is within the feasible range from (37).

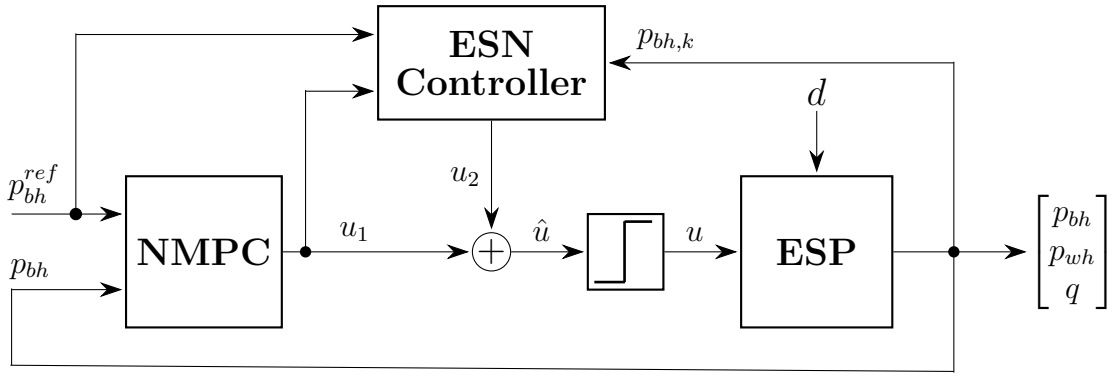


Figure 13: Schematic illustration of the final implementation of the controller.

The following subsection will cover the implementation and training of the ESN controller.

4.1.1 Implementation and Training of the ESN Controller

The baseline ESN framework implemented in (Grønningsæter, 2022) was utilized also in this work. This framework was implemented in Python 3.10 using mainly *PyTorch*, along with *NumPy* and *SciPy*. *PyTorch* is an open-source machine learning framework, but it neither offers any ESN models nor any general support for reservoir computations. However, with the support of *NumPy* and *SciPy*, it was used to handle the tensors in the calculations. Furthermore, all training data was normalized to the interval $[-1, 1]$ using the *MinMaxScaler()* from the *preprocessing* module offered in *scikit-learn (sklearn)*. *scikit-learn* is an open-source machine learning library for Python.

Generation of Training and Test Data

It was demonstrated in (Jaeger, 2008) that the reservoir can be trained offline through model exploration by using random values as output and the corresponding system responses as input. This strategy was adopted in this work for the ESN to learn the inverse model of the ESP affected by disturbance. More specifically, output values from a white stochastic process were used to excite the system, and training data was collected by recording this random value and the corresponding system responses. Both uniformly ($\mathcal{U}$) and normal ($\mathcal{N}$) distributed white noise were tested in this dissertation.

Due to the ESP model being nonlinear, steps with random magnitudes and different holding times were made in the system reference during training. This enables the ESN to explore dynamics from the entire operational area, as detailed in (Grønningsæter, 2022). In terms of the holding time, the reference was maintained for 50 samples in the first half of the training period and for 120 samples in the second half. This corresponds to 5 and 12 seconds, respectively, enabling the model to capture both faster and slower dynamics. Figure 14 shows a schematic illustration of the training loop used to generate training data. The test data for hyperparameter tuning was generated in the same way.

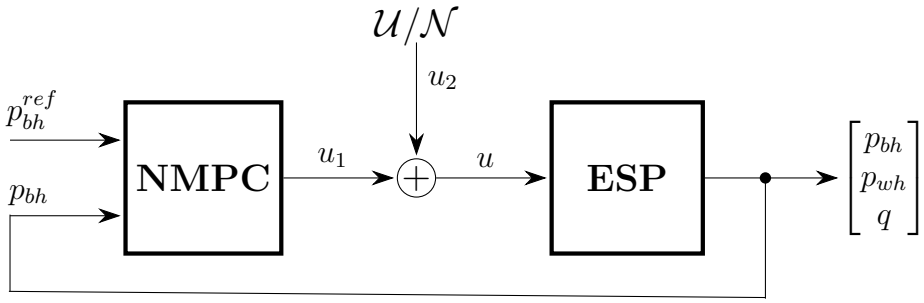
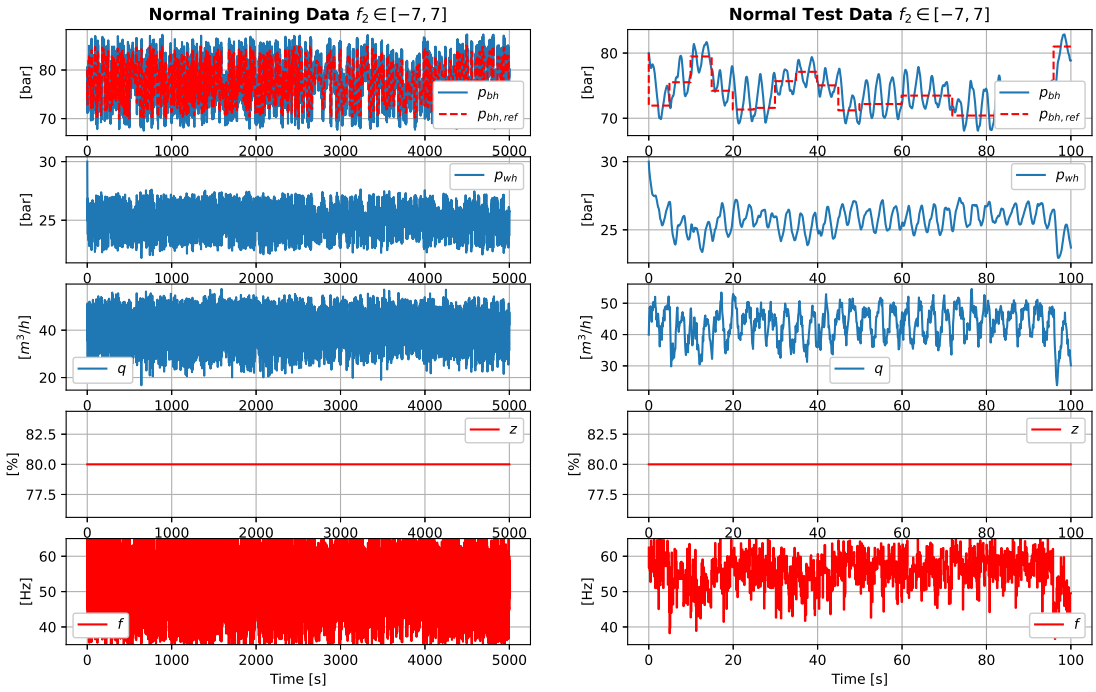


Figure 14: Schematic illustration of the training loop used to generate training and test data for the ESN.

The interval from which the u_2 values were drawn was found using a trial and error approach, starting with $u_2 \in [-7, 7]$. Initially, only the motor frequency f was considered while the valve opening z was kept fixed. To ensure that the system still respected (37), the interval for the NMPC was narrowed. Thus, for the aforementioned $u_2 \in [-7, 7]$, a $u_1 \in [42, 58]$ was selected as this yields $u_1 + u_2 = u \in [35, 65]$. These intervals turned

out to depend on the particular case and will be reported as a part of the results. The training and test sets used in this work contained 50 000 and 1 000 samples, respectively. Figure 15 shows training and test data generated from normally distributed white noise for the training and tuning phase of the ESN controller. Uniform training data appears similar and is therefore not shown.



(a) Training data (50 000 samples).

(b) Test data (1 000 samples).

Figure 15: Normally distributed training and test data generated for z fixed at 80% and $f_2 \in [-7, 7]$.

Even though the ESN is trained with a large number of training samples, it may still encounter outliers from partly explored regions. The probability of this declines with an increasing amount of training data, but there is yet no general method to ensure that the model indeed explores all regions sufficiently. Encountering such an area in the control phase can potentially produce saturated control inputs, which may cause instability in the overall control scheme. To avoid this, the hyperbolic tangent of the control output, $tanh(u_2)$, was used instead of u_2. This ensured that all values were within the training region since all data was initially scaled between -1 and 1.

Lastly, to generate data using the normal distribution, a mean μ of zero and a standard deviation σ covering the desired region were found. It can be challenging to find an exact σ with maximum and minimum values precisely within the closed interval. Instead, various bell curves were examined for the given interval, and a suitable σ was found through a qualitative interpretation of these plots. Then, a saturation function was applied to ensure that all values remained within the closed interval during the training.

4.2 Experimental Designs

Three experiments were conducted in an attempt to answer the research questions stated in Section 1.2. This section will detail how these experiments were designed and conducted.

Experiment 1: Comparison of a Uniform and Normal Model

Two models were trained in this experiment: one using uniform noise and the other using normal noise. In both cases, the ESP was exposed to the actual disturbance affecting the real system. This is equivalent to training the ESN with operational data recorded from an NMPC controlling an ESP subject to disturbance. Therefore, it is expected that the ESN will learn an inverse model specific to the particular system where the dynamics of the actual disturbance are incorporated.

During the training and simulations, the ESP was subject to a sinusoidal disturbance in the reservoir pressure. The disturbance is shown in Figure 16. It is known from Table 1 that the ESP model maintains a constant reservoir pressure of 126 bar. However, after applying the disturbance, this parameter will oscillate between 116 and 136 bar.

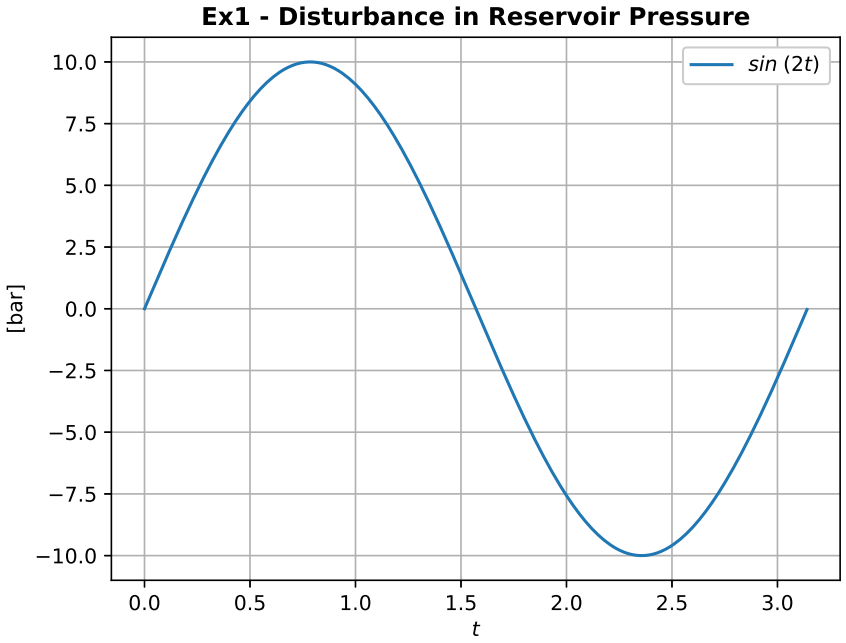


Figure 16: The disturbance affecting the ESP reservoir pressure in all experiments.

After the training phase, both models were used to synthesize two ESN controllers. Each controller was tested through two simulations. In both simulations, the controllers were supposed to track the bottomhole pressure reference p_{bh}^{ref} subject to the same disturbance as in the training phase. The first simulation involved a constant reference, while the second included a reference step of 3 bar halfway through the simulation time. The controllers' ability to suppress the disturbance was evaluated by comparing their performance to the performance of an NMPC controller without ESN

feedforward control. Finally, the performance of the controller using the uniform model was compared to the controller using the normal model.

Experiment 2: Training with Random System Disturbance

In contrast to the experiment comparing the uniform and the normal model, only a single uniform model was trained in this experiment. The choice of using the uniform model was based on the findings from the previous experiment, which will be further discussed in Chapter 6. Another important difference in this experiment is that the ESP will be exposed to random disturbance during the training phase. This is equivalent to training the ESN without any operational data providing prior knowledge of the actual disturbance. Consequently, the ESN is expected to learn how the system is generally affected by disturbances. The random disturbance was generated from drawing values from the uniform distribution $\mathcal{U}(-10, 10)$ every 0.5 seconds. This sampling time ensured that the disturbance could propagate through the system.

After completing the training phase, the model was utilized to synthesize an ESN controller. The resulting controller was evaluated through the same simulations as the previous experiment comparing the two models. To investigate the ESN's capability in suppressing disturbances with different characteristics, the simulations were conducted twice with two different disturbances affecting the reservoir pressure. The first disturbance was the same as when the two models were compared, while the second was a slightly more complex disturbance. It is shown in Figure 17.

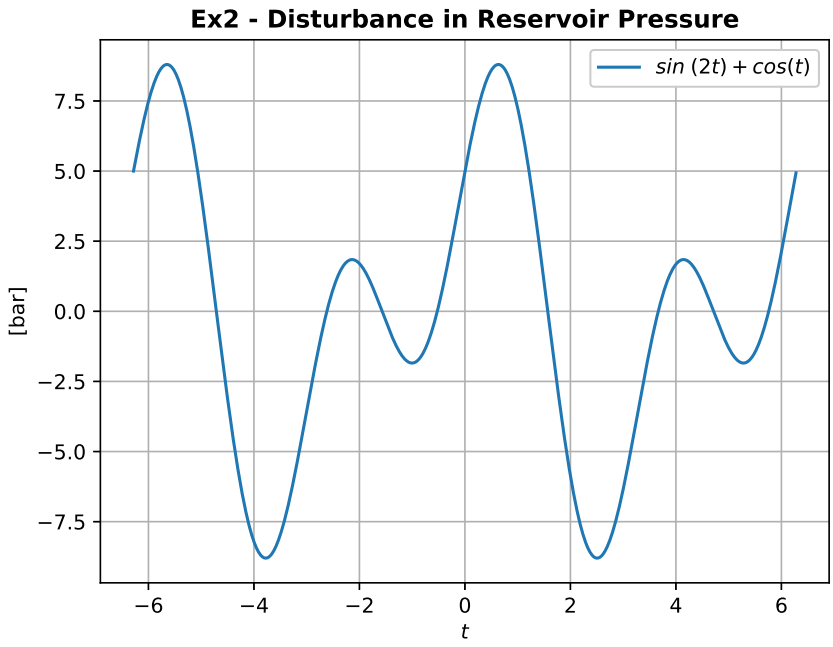


Figure 17: The disturbance affecting the ESP reservoir pressure in the second part of Experiment 2 and 3.

Experiment 3: Manipulation of Both Control Inputs

The uniform model was also selected for this final experiment. However, in this case, the controller was granted an additional degree of freedom by manipulating both the valve opening z and the motor frequency f. This resulted in the control input being expanded from scalars to vectors:

$$u_1 + u_2 = \begin{bmatrix} z_1 \\ f_1 \end{bmatrix} + \begin{bmatrix} z_2 \\ f_2 \end{bmatrix} = \begin{bmatrix} z \\ f \end{bmatrix} = u \tag{43}$$

Similarly to the previous experiment, the uniform model was exposed to random system disturbance during the training phase. Furthermore, the same set of experiments was also conducted in this experiment.

4.3 Hyperparameter Tuning

It is evident from Subsection 2.4.2 that there are many hyperparameters to consider in the context of the ESN. In addition to the hyperparameters considered in (Grønningsæter, 2022), two more hyperparameters must be considered in this work. These are the input delay δ and the bias scaling. This further complicates the search for the globally optimal set of hyperparameters. As described in Section 2.4, there is no general method for finding either a suboptimal or the globally optimal set of hyperparameters. Many of the strategies found in the literature are therefore ad-hoc strategies based on experience and grid searches.

The hyperparameters used for learning the ESP model in (Grønningsæter, 2022) were found using such an ad-hoc strategy. Each hyperparameter was determined by searching for the value yielding the lowest normalized root mean squared error (NRMSE) on a validation set while holding the other parameters constant. It is possible (and even expected) that there are better sets of hyperparameters than those reported. However, for the particular project objective, the reported values proved sufficient.

Since the test set in this work consists of system responses to random control values, it is not expected that there exists a direct correlation between minimizing the MSE and suppressing the system disturbance. This suggests that a good generalization of the system should be capable of recognizing the trend in the noise rather than making accurate predictions of it. The following three subsections will detail the strategies used to determine hyperparameters in this work.

Determining the Reservoir Size (N_x) and Spectral Radius ($\rho(\boldsymbol{W}_r)$)

Running a grid search for all the hyperparameters considered in this work would be computationally impractical. The reservoir size (N_x) and spectral radius ($\rho(\boldsymbol{W}_r)$) were therefore determined based on insight from (Lukoševičius, 2012) and the experience gained from (Grønningsæter, 2022). According to (Lukoševičius, 2012), the reservoir size should be selected as large as computationally feasible. Thus, a reservoir size of 500 nodes was utilized in all experiments. Furthermore, it is recommended to select the spectral radius as close to 1 as possible to ensure both the echo state property and long-lasting memory for the reservoir. Consequently, a spectral radius of 0.999 was chosen.

Determining the Input and Bias Scaling

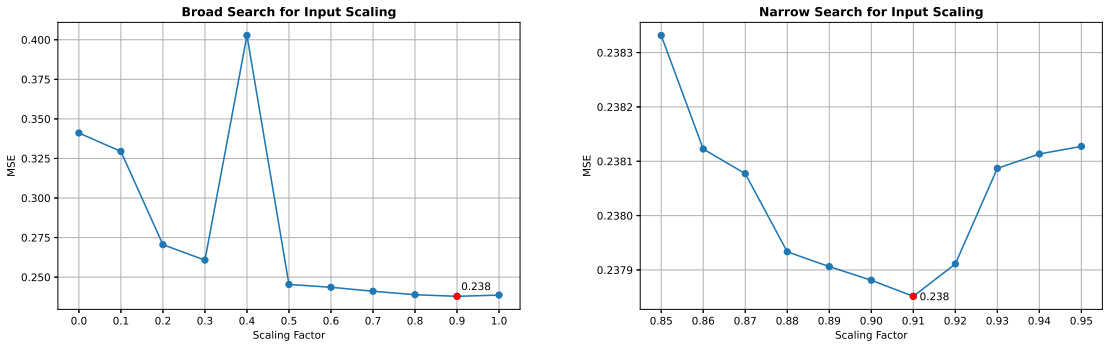
Within the reservoir, (Lukoševičius, 2012) states that the input scaling and the spectral radius together regulate the

i amount of nonlinearity in the reservoir representation.

ii the relative effect of the current input opposed to history.

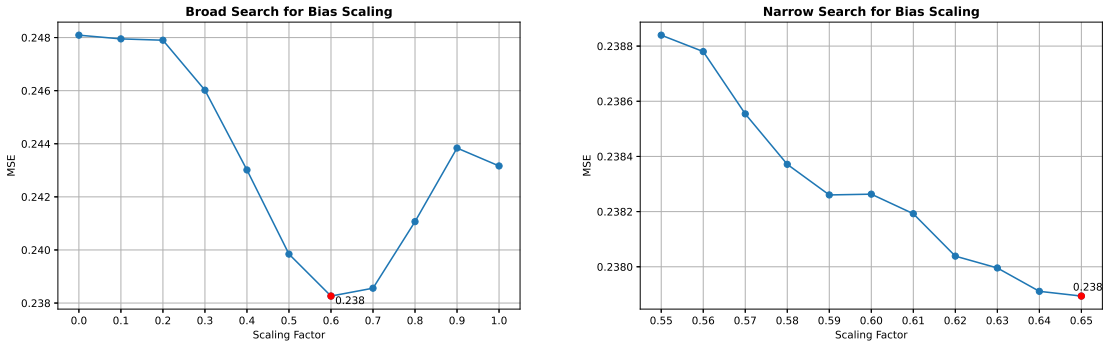
This also applies to some extent to the bias scaling. Hence, these two hyperparameters will primarily affect the aggressiveness of the controller as the spectral radius is fixed.

The model's ability to generalize to the white noise can be used as a metric for this. A lower MSE indicates that the model is capable of predicting values across a broader spectrum. Consequently, the scalings were determined by conducting similar searches to those conducted in (Grønningsæter, 2022). Figure 18 shows the search for the input scaling, and Figure 19 shows the search for the bias scaling, both for the uniform model in the experiment comparing the uniform and normal models. From inspection, it is evident that an input scaling of 0.91 and a bias scaling of 0.65 result in the lowest MSEs.



(a) Broad search. (b) Narrow search.

Figure 18: Broad (a) and narrow (b) search for the input scaling for the uniform model in Experiment 1.



(a) Broad search. (b) Narrow search.

Figure 19: Broad (a) and narrow (b) search for the bias scaling for the uniform model in Experiment 1.

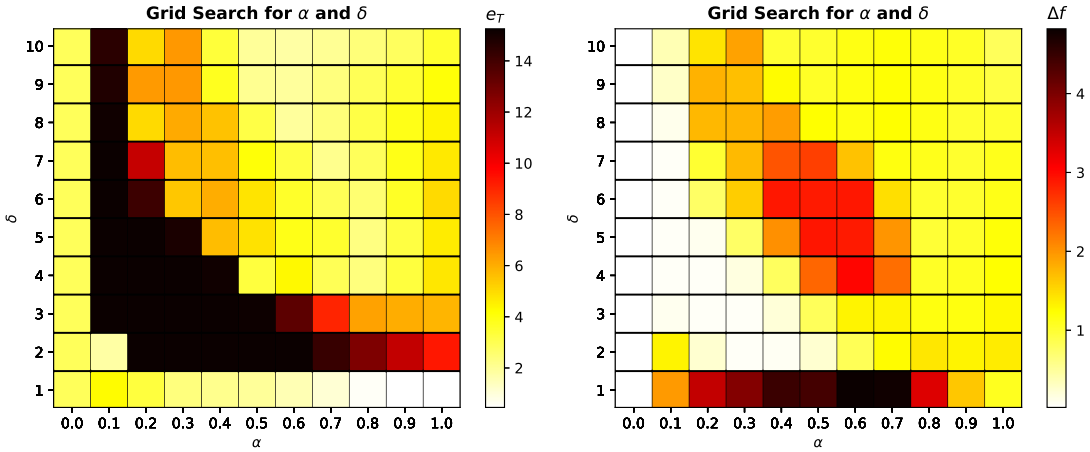
Prior to each search, satisfactory values for the remaining hyperparameters were determined through experimentation. These values were then held constant during the search process.

Determining the Delay (δ), Leak Rate (α), and Regularization Coefficient (β)

While experimenting to determine temporary values for the delay (δ), leak rate (α), and regularization coefficient (β) in the previous section, it became clear that these

hyperparameters play a crucial role in suppressing the system disturbance. Therefore, grid searches were conducted to find suitable values for these hyperparameters. For each combination of α and δ, a similar search to those conducted for the scalings was performed to determine the corresponding optimal β value. This was the value yielding the lowest MSE on the test set from the range $[10^{-8}, 10^{8}]$ with a step size of 10.

After finding the optimal β value, the performance of the corresponding (δ, α) pair was evaluated using MAE and Δf. The results from the grid search conducted for the uniform model from Experiment 1 are shown in Figure 20. It can be seen from the figures that the best candidates are $(0.9, 1)$ and $(1.0, 1)$. From further experimentation with these values, it was found that $(0.9, 1)$ with a β of 10^{-2} yielded the most satisfactory response.



(a) MAE (e_T) per grid element.

(b) Δf per grid element.

Figure 20: Grid search for (α, δ) for the uniform model in Experiment 1. Each set is measured in MAE (a) and Δf (b) per grid element.

The hyperparameter values found in this section are summarized in Table 3. Although sparsity was set to one and not further considered during the searches, it will still be reported. The strategy presented in this section was used to find the hyperparameters for all the other models too. However, these will be reported together with the results. Plots of the searches similar to those in this section will be attached as appendices.

Hyperparameter	**Value**
Reservoir size (N_x)	500
Spectral radius ($\rho(\mathbf{W})$)	0.999
Input scaling	0.91
Bias scaling	0.65
Leakage rate (α)	0.9
Delta (δ)	1
Regularization coefficient (β)	10^{-2}
(Sparsity)	1

Table 3: Resulting hyperparameters used in the uniform model from Experiment 1.

Figure 21 shows a snapshot of the uniform model tracking the test set using the hyperparameters from Table 3. This yielded an MSE of 0.2378 and demonstrates as expected that the model would not track the test set accurately, but still be able to recognize the tendency in the noise.

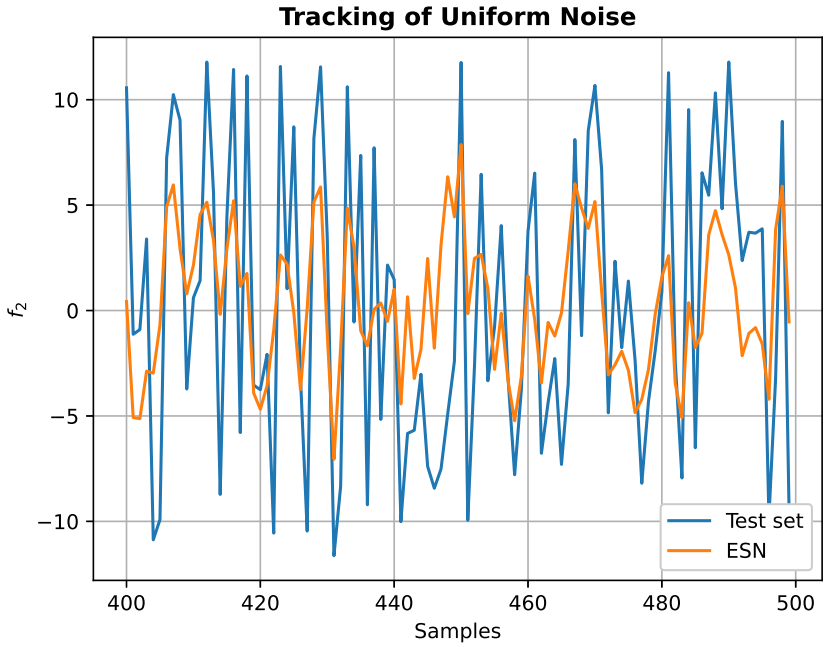


Figure 21: Uniform model in Experiment 1 tracking the test set using the hyperparameters from Table 3.

5 Results

This chapter reports the results obtained from the experiments described in Section 4.2. Each section will provide a brief description of how the results are presented. A thorough discussion of the results will not be given before Chapter 6. The three sections in this chapter cover:

Section 5.1: The results from the comparison of two models trained with uniform and normal noise.

Section 5.2: The results from the model trained with uniform noise and random system disturbance.

Section 5.3: The results from the model manipulating both ESP control inputs. This model was trained with uniform noise and random system disturbance.

5.1 Comparison of a Uniform and a Normal Model

Table 4 summarizes the hyperparameters, training settings, and control settings used in this experiment. The hyperparameters for the uniform model were determined as described in Section 4.3, while for the normal model, they were determined as described in Appendix D. In the following two subsections, the results from the simulations with and without a reference step are presented. To facilitate visual comparison for the reader, the results from each model will be presented side-by-side. As the valve opening z is fixed at 80 %, this control input will be omitted from all figures in this section.

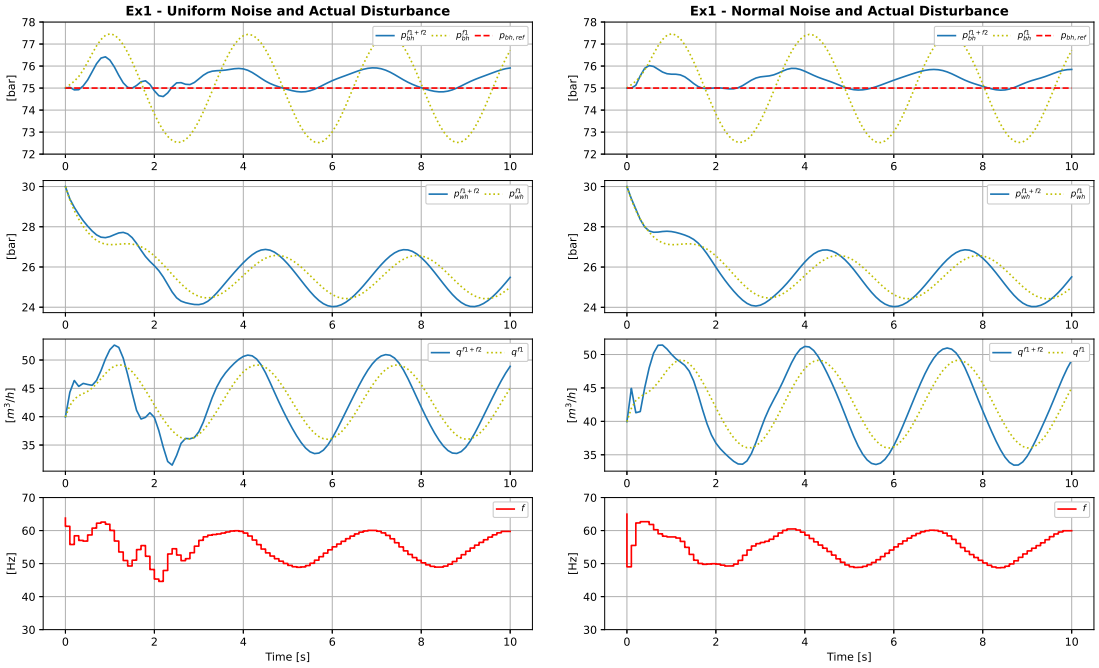
Hyperparameter	**Uniform model**	**Normal model**
Reservoir size (N_x)	500	500
Spectral radius ($\rho(\mathbf{W})$)	0.999	0.999
Input scaling	0.91	0.56
Bias scaling	0.65	0.65
Leak rate (α)	0.9	0.9
Delta (δ)	1	1
Regularization coefficient (β)	10^{-2}	10^{-7}
(Sparsity)	1	1
Training settings		
Motor frequency noise [Hz]	$\mathcal{U}(-12, 12)$	$sat(\mathcal{N}(0, 4))$[1]
System disturbance [bar]	$10sin(2t)$	$10sin(2t)$
Control setting		
Sampling rate [s/sample]	0.1	0.1
NMPC horizon [s]	0.5	0.5
NMPC control range [Hz]	$f_1 \in [47, 53]$	$f_1 \in [47, 53]$
ESN control range [Hz]	$f_2 \in [-12, 12]$	$f_2 \in [-12, 12]$

Table 4: Hyperparameters, training settings, and control settings for each model in the experiment comparing the uniform and normal model.

[1] $sat(\cdot)$ denotes the saturation function ensuring that the random values are inside the predetermined interval.

Tracking of a Constant Reference

The simulations of constant reference tracking are shown for both models in Figure 22. Table 5 provides the corresponding calculated errors.



(a) Uniform model. (b) Normal model.

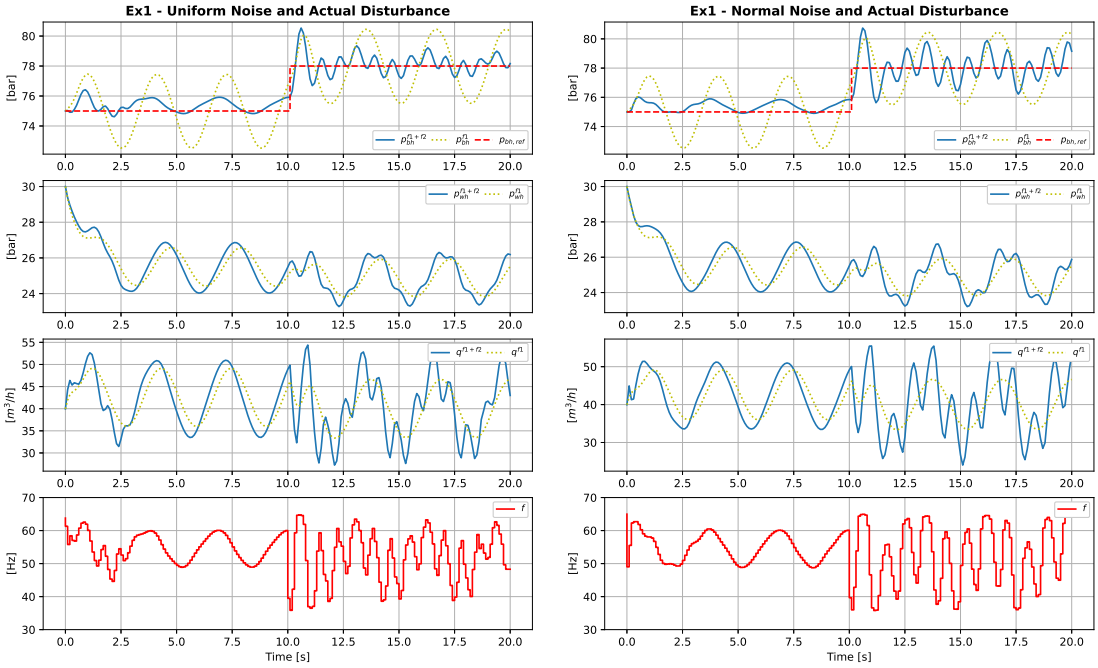
Figure 22: Simulations of constant reference tracking for the comparison of a uniform (a) and a normal (b) model.

Metric	NMPC	NMPC+ESN$_{\mathcal{U}}$	NMPC+ESN$_{\mathcal{N}}$
IAE [bar]	125.40	34.48	31.94
MAE [bar]	1.57	0.43	0.40
Δf [Hz]	-	1.16	1.01
Improvement	-	72.50%	74.53%

Table 5: Calculated errors from simulations of constant reference tracking for the comparison of a uniform and a normal model.

Tracking of a Reference with a Step

The simulations of reference tracking with steps are shown for both models in Figure 23. Table 6 provides the corresponding calculated errors.



(a) Uniform model.

(b) Normal model.

Figure 23: Simulations of reference tracking with a step for the comparison of a uniform (a) and a normal (b) model.

Metric	NMPC	NMPC+ESN$_{\mathcal{U}}$	NMPC+ESN$_{\mathcal{N}}$
IAE [bar]	281.37	90.78	121.79
MAE [bar]	1.56.	0.50	0.68
Δf [Hz]	-	2.86	2.96
Improvement	-	67.74%	56.71%

Table 6: Calculated errors from simulations of reference tracking with a step for the comparison of a uniform and a normal model.

5.2 Uniform Model Trained with Random System Disturbance

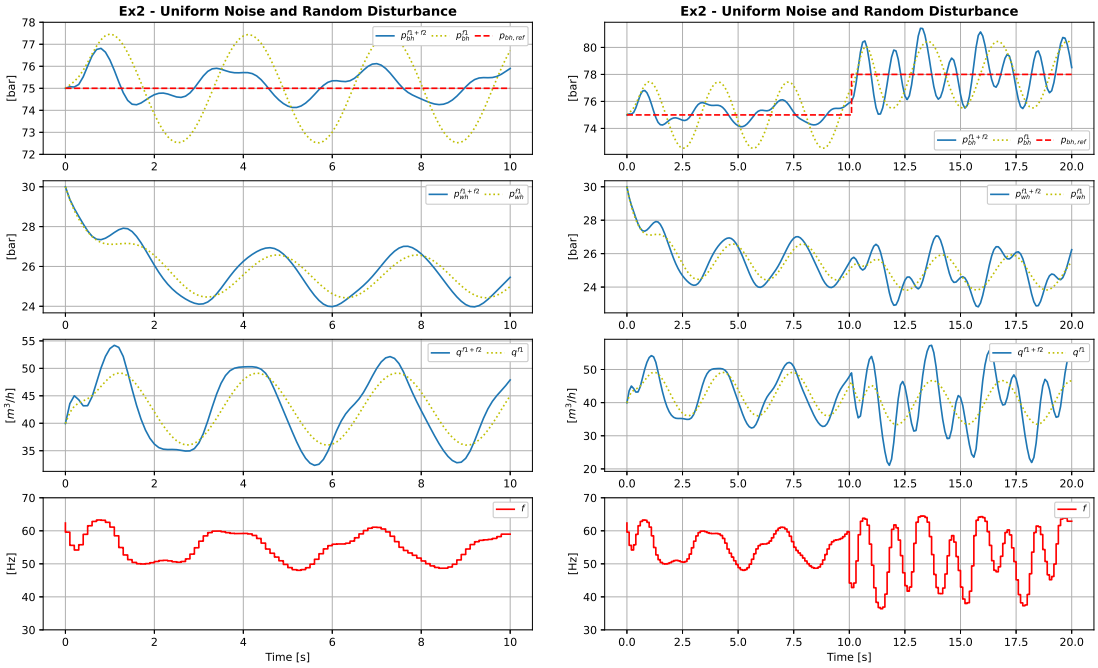
Table 7 summarizes the hyperparameters, training settings, and control settings used in this experiment. The hyperparameters for the uniform model utilized in this experiment were determined as described in Appendix E. In the following two subsections, the results from the simulations with the two different disturbances are presented. For each disturbance, the simulation with constant reference and the simulation with a step in the reference are presented side-by-side to make the sections more concise. As the valve opening z was fixed at 80 % also in this experiment, this control input will once again be omitted from all figures.

Hyperparameter	**Uniform model**
Reservoir size (N_x)	500
Spectral radius ($\rho(\mathbf{W})$)	0.999
Input scaling	0.59
Bias scaling	1.00
Leak rate (α)	0.6
Delta (δ)	1
Regularization coefficient (β)	1
(Sparsity)	1
Control setting	
Sampling rate [s/sample]	0.1
NMPC horizon [s]	0.5
NMPC control range [Hz]	$f_1 \in [46, 54]$
ESN control range [Hz]	$f_2 \in [-11, 11]$
Training setting	
Frequency noise [Hz]	$\mathcal{U}(-11, 11)$
System disturbance [bar]	$\mathcal{U}(-10, 10)$
Disturbance sampling time [s]	0.5

Table 7: Hyperparameters, training settings, and control settings for the uniform model trained with random system disturbance.

Simple Sinusoidal Disturbance

Two simulations of the system affected by a simple sinusoidal disturbance are shown in Figure 24. More specifically, Figure 24(a) and 24(b) show the simulation with a constant reference and the simulation with a step in the reference, respectively. Table 8 and 9 provide the corresponding calculated errors.



(a) Constant reference.

(b) Step of 3 bars in reference.

Figure 24: Simulations of the uniform model trained with random system disturbance subject to a simple sinusoidal disturbance.

Metric	NMPC	NMPC+ESN$_{\mathcal{U}}$
IAE [bar]	125.40	41.81
MAE [bar]	1.57	0.52
Δf [Hz]	-	0.90
Improvement	-	66.66%

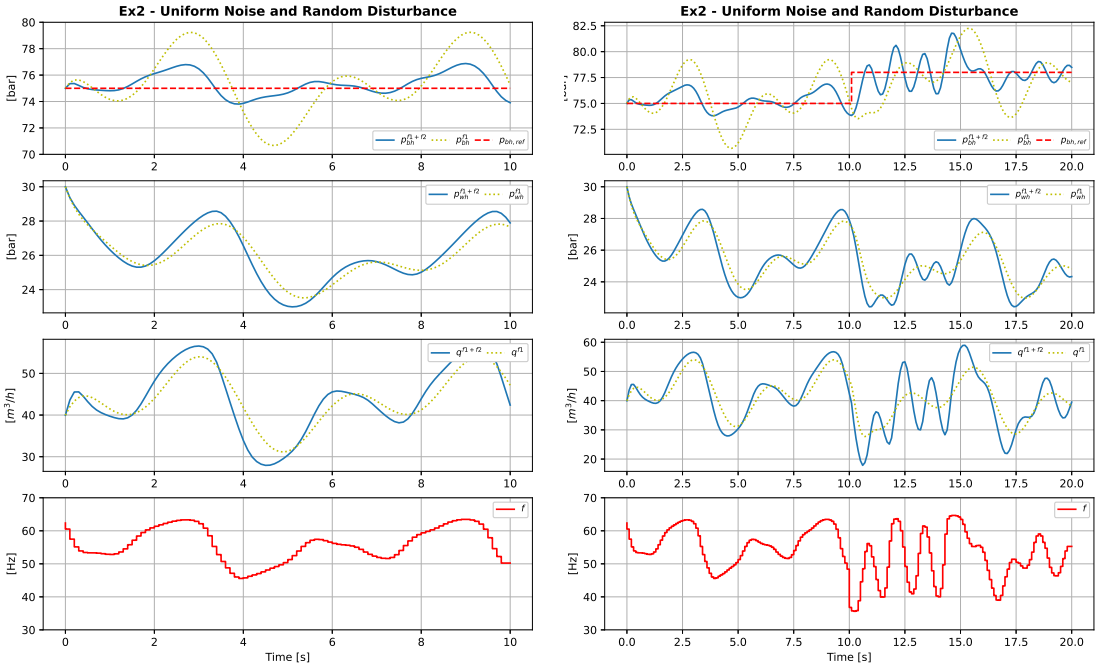
Table 8: Calculated errors from constant reference tracking for the uniform model (trained with random disturbance) subject to a simple sinusoidal disturbance.

Metric	NMPC	NMPC+ESN$_{\mathcal{U}}$
IAE [bar]	281.37	185.08
MAE [bar]	1.56	1.03
Δf [Hz]	-	2.13
Improvement	-	34.22%

Table 9: Calculated errors from reference tracking with a step for the uniform model (trained with random disturbance) subject to a simple sinusoidal disturbance.

Complex Sinusoidal Disturbance

Two simulations of the system affected by a complex sinusoidal disturbance are shown in Figure 25. More specifically, Figure 25(a) and 25(b) show the simulation with a constant reference and the simulation with a step in the reference, respectively. Table 10 and 11 provide the corresponding calculated errors.



(a) Constant reference.

(b) Step of 3 bars in reference.

Figure 25: Simulations of the uniform model trained with random system disturbance subject to a complex sinusoidal disturbance.

Metric	NMPC	NMPC+ESN$_{\mathcal{U}}$
IAE [bar]	173.06	63.01
MAE [bar]	2.16	0.79
Δf [Hz]	-	0.81
Improvement	-	63.59%

Table 10: Calculated errors from constant reference tracking for the uniform model (trained with random disturbance) subject to a complex sinusoidal disturbance.

Metric	NMPC	NMPC+ESN$_{\mathcal{U}}$
IAE [bar]	369.25	176.65
MAE [bar]	2.05	0.98
Δf [Hz]	-	1.60
Improvement	-	52.16%

Table 11: Calculated errors from reference tracking with a step for the uniform model (trained with random disturbance) subject to a complex sinusoidal disturbance.

5.3 Uniform Model Manipulating Both ESP Control Inputs

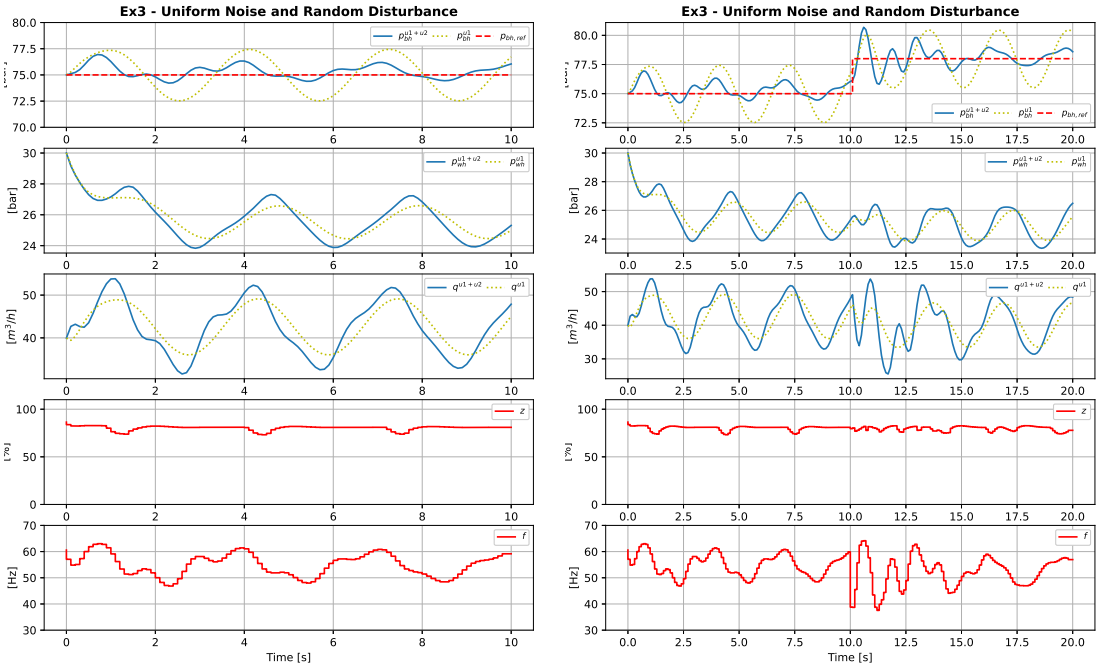
Table 12 summarizes the hyperparameters, training settings, and control settings used in this experiment. The hyperparameters for the uniform model utilized in this experiment were determined as described in Appendix F. In the following two subsections, the results from simulations with the two different disturbances are presented. For each disturbance, the simulation with constant reference and the simulation with a step in the reference are presented side-by-side to make the sections more concise.

Hyperparameter	**Uniform model**
Reservoir size (N_x)	500
Spectral radius ($\rho(\mathbf{W})$)	0.999
Input scaling	0.30
Bias scaling	1.00
Leak rate (α)	0.6
Delta (δ)	1
Regularization coefficient (β)	1
(Sparsity)	1
Training setting	
Motor frequency noise [Hz]	$\mathcal{U}(-13, 13)$
Valve opening noise [%]	$\mathcal{U}(-35, 35)$
System disturbance [bar]	$\mathcal{U}(-10, 10)$
Disturbance sampling time [s]	0.5
Control setting	
Sampling rate [s/sample]	0.1
NMPC horizon [s]	0.5
NMPC frequency control range [Hz]	$f_1 \in [48, 52]$
NMPC valve control range [%]	$z_1 \in [45, 65]$
ESN frequency control range [Hz]	$f_2 \in [-13, 13]$
ESN valve control range [%]	$z_2 \in [-35, 35]$

Table 12: Hyperparameters, training settings, and control settings for the uniform model manipulating both ESP control inputs.

Simple Sinusoidal Disturbance

Two simulations of the system affected by a simple sinusoidal disturbance are shown in Figure 26. More specifically, Figure 26(a) and 26(b) show the simulation with a constant reference and the simulation with a step in the reference, respectively. Table 13 and 14 provide the corresponding calculated errors.



(a) Constant reference.

(b) Step of 3 bars in reference.

Figure 26: Simulations of the uniform model manipulating both ESP control inputs subject to a simple sinusoidal disturbance.

Metric	NMPC	NMPC+ESN$_\mathcal{U}$
IAE [bar]	125.41	42.24
MAE [bar]	1.57	0.53
Δz [%]	-	0.60
Δf [Hz]	-	1.05
Improvement	-	66.32%

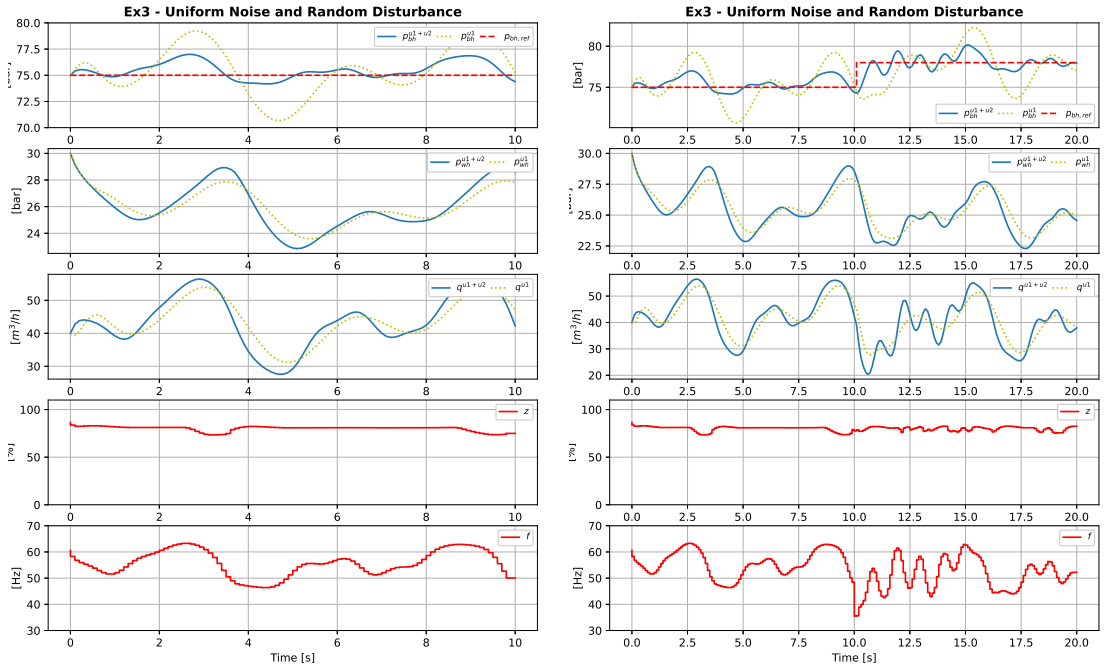
Table 13: Calculated errors from constant reference tracking for the uniform model (manipulating both ESP control inputs) subject to a simple sinusoidal disturbance.

Metric	NMPC	NMPC+ESN$_\mathcal{U}$
IAE [bar]	281.35	112.37
MAE [bar]	1.56	0.62
Δz [%]	-	0.82
Δf [Hz]	-	1.49
Improvement	-	60.06%

Table 14: Calculated errors from reference tracking with a step for the uniform model (manipulating both ESP control inputs) subject to a simple sinusoidal disturbance.

Complex Sinusoidal Disturbance

Two simulations of the system affected by a complex sinusoidal disturbance are shown in Figure 27. More specifically, Figure 27(a) and 27(b) show the simulation with a constant reference and the simulation with a step in the reference, respectively. Table 15 and 16 provide the corresponding calculated errors.



(a) Constant reference.

(b) Step of 3 bars in reference.

Figure 27: Simulations of the uniform model manipulating both ESP control inputs subject to a complex sinusoidal disturbance.

Metric	NMPC	NMPC+ESN$_{\mathcal{U}}$
IAE [bar]	173.03	63.56
MAE [bar]	2.16	0.79
Δz [%]	-	0.35
Δf [Hz]	-	0.81
Improvement	-	63.26%

Table 15: Calculated errors from constant reference tracking for uniform model (manipulating both ESP control inputs) subject to a complex sinusoidal disturbance.

Metric	NMPC	NMPC+ESN$_{\mathcal{U}}$
IAE [bar]	396.16	138.50
MAE [bar]	2.05	0.77
Δz [%]	-	0.69
Δf [Hz]	-	1.37
Improvement	-	62.48%

Table 16: Calculated errors from reference tracking with a step for uniform model (manipulating both ESP control inputs) subject to a complex sinusoidal disturbance.

6 Discussion

This chapter aims to provide a thorough discussion of the experimental results presented in Chapter 5. The results from each of the three experiments will be examined with a focus on their respective findings and the implications of these findings in relation to each other. All comparisons will be made using the MAE and the variation of control. The chapter is concluded with a more general discussion of the experiments and their findings within a broader context of the general objectives of this dissertation.

Section 6.1: Discussion of the results obtained from the comparison of a uniform and normal model.

Section 6.2: Discussion of the results obtained from simulating a uniform model trained with random system disturbance.

Section 6.3: Discussion of the results obtained from simulating a uniform model manipulating both ESP control inputs.

Section 6.4: General discussion of the experiments and the implications of the obtained results within a broader context.

6.1 Comparison of a Uniform and a Normal Model

Examination of the results in Chapter 5.1, summarized in Table 5 and 6, shows that the normal model performed slightly better than the uniform model when tracking a constant state reference. Compared to the NMPC without ESN assistance, the normal model yielded a 74.53% improvement of MAE, while the uniform model yielded an improvement of 72.50%. This was achieved using slightly less variation of control, with a $\Delta f = 1.01$ Hz compared to the $\Delta f = 1.16$ Hz used by the uniform model.

However, in the simulation with a 3 bar step in the state reference, the uniform model outperformed the normal model by nearly 10%. In this case, the uniform model improved the MAE by 67.74%, while the normal model yielded only an improvement of 56.71%. Moreover, the uniform model also used less variation of control, with a $\Delta f = 2.86$ Hz compared to the $\Delta f = 2.96$ Hz used by the normal model.

These results suggest that a controller using a uniform model is better suited for this control task. It is expected that the ESP will operate with different references, and the controller should therefore be able to track these also after a change in the reference. The reason why the uniform model is better suited can be further explained by considering the dynamics of the closed-loop system. Since the valve is fixed, the bottomhole pressure can only be reduced by increasing the motor frequency to extract more fluid. Conversely, the opposite applies when the pressure needs to be increased. Furthermore, in the presence of a harmonic disturbance like in this case, the controller should generate a harmonic variation in control inputs to counteract it. This behavior is evident in both simulations and can be seen from inspecting the control inputs in Figure 22.

To generate the harmonic variation of control inputs seen in this figure, the ESN must gradually increase and decrease the correctional inputs. It is also evident that all the correcting inputs lie strictly within the training interval, as the resulting control input never saturates. These values are generally more likely when training the model using normally distributed noise. With this distribution, the probability of drawing an extreme value is much smaller compared to drawing a value around the mean (which in this case is zero). Consequently, the controller is more likely to generate values that are close to each other. This is not the case with the uniform distribution, where the probability of drawing any value within the interval is the same. Eventually, this will result in a more aggressive controller, hence the larger Δf observed in the first simulation.

Inspection of Figure 23 shows that both controllers struggle with oscillations after the step. These oscillations are believed to be caused by the NMPC, as it was demonstrated in Figure 12 that manipulation of only the motor frequency f is insufficient to bring the system to a new state without oscillatory behavior. However, it is apparent that the normal model struggles more compared to the uniform model. A closer examination of the control inputs in Figure 23 reveals that the uniform model generates inputs that are larger relative to each other. Since both models utilize the same NMPC, it is clear that the uniform model handles the oscillations better due to being more aggressive. However, the observed difference in control inputs is not reflected in the Δf values. A reasonable explanation for this disparity is that the control inputs generated by the normal model are unable to dampen the oscillations, resulting in larger variations in control over time. This explanation is also supported by further inspection of Figure 23.

6.2 Uniform Model Trained with Random System Disturbance

Analysis of the results in Chapter 5.2, summarized in Table 8-11, reveals that the uniform model trained with random disturbance improved the MAE by 66.66% when tracking a constant reference while being affected by a standard sinusoidal disturbance. This improvement was achieved with a control input variation of $\Delta f = 0.90$ Hz. In comparison to the previous experiment, where the uniform and normal model were compared, this model yielded approximately 6% less improvement of MAE. Yet, it achieved this utilizing 0.26 Hz less variation of control. These results exceeded expectations since the model in this experiment has no prior knowledge of the actual disturbance affecting the system.

Although the model performed surprisingly well when tracking a constant reference, it only improved the MAE by 34.22% when a step was made in the reference. This improvement is only about 50% of what was achieved in the first experiment. In this case, however, the model utilized approximately 0.70 Hz less variation in control, resulting in a $\Delta f = 2.13$ Hz. A visual inspection of Figure 24(b) shows that this model suffers significantly from the oscillations after the step, even worse than the normal model from Figure 23. One possible explanation for this could be that the model utilizes an ESN control range that is one Hz smaller than the model in the comparison experiment. However, additional experiments conducted with a wider control range yielded even worse results. Another plausible explanation for the significantly worse performance is the absence of the actual disturbance during the training phase. Without this prior knowledge, the model might not be able to counteract the oscillations caused by the NMPC as discussed in Section 6.1.

On the other hand, it is likely that the training data contains similar events due to both the steps and the disturbance being random. Furthermore, during this experiment, it was discovered that the disturbance had to be within the bounds of the actual disturbance. This assumption is reasonable in order for the controller to be suitable for the actual range of the disturbance affecting the system. If the controller were trained with disturbances with lower amplitudes, it would overestimate the system disturbance and become too aggressive. Conversely, the controller would underestimate the system disturbance and become insufficiently aggressive. This is a common assumption within robust control theory, and it would be nearly impossible to achieve performance improvement without it. With this assumption, it is also even more likely that the model will encounter events similar to the step during the training phase.

The simulation with the more complex disturbance yielded similar results to the standard sinusoidal disturbance when it came to tracking a constant reference. In this case, the model improved the MAE by 63.59%, which is only 3% less than in the previous case. It achieved this with slightly less variation of control resulting in a $\Delta f = 0.81$. This suggests that the model is capable of improving the control performance regardless of the disturbance. Thus, random disturbance in the training phase indeed enables the model to learn a general inverse model.

It is perhaps even more surprising that the model in the simulation with a step was able

to improve the MAE by 52.16% with a $\Delta f = 1.60$ Hz. Since the inflicting disturbance is different in this simulation, it can not be directly compared to any of the previous experiments. Nevertheless, it is remarkable how the model handles the oscillations after the step when compared qualitatively to the previous experiments. Inspection of Figure 25(b) reveals that fewer oscillations are present after the step. These oscillations appear to be gone after about five seconds. At this point, it seems like the controller has returned to the same behavior as prior to the step. This suggests that the training data indeed contains similar events to the reference step, contrary to what the experiment with the standard sinusoidal disturbance implied. Moreover, it also indicates that the model struggles more with a sinusoidal disturbance than disturbances composed of different harmonic components.

6.3 Uniform Model Manipulating Both ESP Control Inputs

Investigation of the results in Chapter 5.3, summarized in Table 13-16, yields that the model benefits massively from manipulating the valve opening z in addition to the motor frequency f. The controller improved the MAE by 66.32% when tracking a constant reference while being subject to a standard sinusoidal disturbance. This was achieved using a variation of control of $\Delta z = 0.60\%$ and $\Delta f = 1.05$ Hz. These results are very similar to those obtained from the corresponding simulation with a fixed valve opening.

However, in the simulation with a step in the reference, this model proves superior with a 60.06% improvement of MAE. This improvement is nearly 26% higher than in the corresponding experiment with a fixed valve opening. Inspection of the variation of control reveals a $\Delta z = 0.82\%$ and $\Delta f = 1.49$ Hz, which is approximately 0.6 Hz less than in the fixed valve case. These results demonstrate the significant impact that the valve opening has on the ESP system. Especially if taking its entire control range of 90% into consideration, it is remarkable how an accumulative variation of barely 1% can result in such a major improvement.

The same conclusions can be drawn from inspecting the results obtained after conducting the same simulations with the complex sinusoidal disturbance. When tracking a constant reference, the controller achieved a 63.26% improvement with a $\Delta z = 0.35\%$ and a $\Delta f = 0.81$ Hz. This result is similar to the result from the corresponding simulation with the fixed valve opening. In the other simulation, with a step in the reference, the controller yielded an improvement of 62.48% with a $\Delta z = 0.69\%$ and a $\Delta f = 1.37$ Hz. This is about a 10% improvement compared to the corresponding fixed valve experiment. It can also be observed in these simulations that the variation of the valve opening reduces the variation of the motor frequency.

In contrast to the fixed valve experiments, this model efficiently counteracts the oscillations that occur after the reference step. This is evident from a visual comparison of Figure 26(b) and 27(b). In both figures, some additional oscillations are visible immediately after the step, but these disappear after about five seconds. Subsequently, the system returns to the same behavior prior to the step. These observations, combined with the improvement discussed in the previous paragraphs, demonstrate that the dual input controller is capable of both suppressing disturbance and track references with and without steps.

Furthermore, it is evident that manipulation of the valve opening is necessary to perform reference changes. One of the reasons for this becomes clear when comparing the wellhead pressures p_{wh} in Figure 24(a) and 26(a). In the former figure, the pressure in the wellhead changes harmonically. However, in the latter figure, the pressure changes take on a more triangular shape with linear slopes. The peaks of these triangular variations align with the reduction and increase of the valve opening, preventing decelerations of the pressure build-ups and reliefs. These observations highlight how even small adjustments in the valve opening allow effective regulation of the overall pressure in the ESP system. Considering the ESP model given in (22)-(24), it can be seen how these pressure changes propagate throughout the system. Generally, an in-

crease/decrease in the wellhead pressure leads to a corresponding decrease/increase in flow, ultimately increasing/decreasing the bottomhole pressure. This relationship was also demonstrated in (Grønningsæter, 2022).

It can be observed in Figure 26(b) and 27(b) that the valve opening varies more after the reference step. This once again highlights the importance of the valve opening in suppressing the additional oscillations caused by the NMPC. Moreover, it also demonstrates that relying solely on the motor is insufficient for achieving the required pressure regulations to effectively suppress these oscillations, without resorting to unrealistic control inputs that could lead to equipment saturation and damage. These findings align with the observations made in the previous experiments. However, in an application where the valve opening is fixed, it is possible that a larger change in the reference could be achieved by conducting multiple smaller changes. However, this approach was not tested in this dissertation due to the satisfactory results obtained in this experiment.

6.4 The Experiments and their Implications in a Broader Context

The results from the three conducted experiments demonstrate that the ESN can provide a sufficient inverse model for feedforward control. However, during the simulations, it was observed that the NMPC always saturated and that the ESN provided corrections to these saturated values. To what extent this would matter in a practical application is difficult to evaluate. Since there is no overlap between the NMPC and ESN control ranges, the hyperbolic tangent of the ESN value u_2 ensures that the final control input u never saturates. Moreover, the NMPC values are used as inputs to the ESN, meaning that the ESN should be able to account for the saturation observed in the NMPC. The main purpose of this study was, nevertheless, to investigate how the NMPC can be corrected in the absence of a sufficient model. Hence, it should not matter what values the NMPC produces as long as they end up inside the assigned control subinterval.

It should be emphasized that this work serves as a proof of concept, meaning that there most likely exist better configurations of the ESN that could provide even better models. Similarly, while the ESN is an ANN specialized for identifying higher-order relations in dynamical systems, there may exist other models better suited for this specific task. As demonstrated in (Grønningsæter, 2022), the long short-term memory (LSTM) produces equally good models as the ESN, though requiring longer training time. Therefore, it is expected that the LSTM would perform equally well as the ESN, as identifying an inverse model is just a matter of the direction in which the inputs are fed to the ANN.

The general idea investigated in this dissertation can potentially benefit the modern industry greatly. Every real-world process is non-static, meaning it will change over time. Yet, there are many models assuming staticity, which ultimately gives the model and thus the quality of the process optimization an expiration date. Unfortunately, it is often less beneficial to identify a new model as the expired model may still provide suboptimal results. It can also be difficult to predict the benefit of deriving a new model. In any case, it would probably be more economical to augment the system with an ESN-based feedforward controller similar to the one proposed in this work. This would not require a long downtime, and it is far less challenging than conducting an entire new identification of the overall system model.

While many conclusions about the potential benefits of this idea can be drawn from the experiments conducted in this work, there are still many aspects that require further research. It was, among other things, discovered that the time-shift δ and the leak rate α had the most significant impact on the overall control performance. These values were chosen based on their MAE performance during a five-second simulation. In comparison to the MAE, the Δu was practically disregarded. This was primarily due to the main objective of following the reference in this work, and also because the Δu proved to be far less volatile regardless of α and δ. Finding a more effective method of selecting these hyperparameters is therefore believed to be one of the more noteworthy ways of improving the overall control performance.

In retrospect, it might have been better to remove the $\tilde{u}$ from the cost function (34) and instead introduce it as inequality constraints in the optimization problem. These constraints can be derived from the limits reported by the manufacturer of the actuators in the particular system. Moreover, this approach could potentially facilitate the process of finding good hyperparameters since realistic behavior of the actuators is enforced within the optimization problem. In this case, it would not be necessary to consider the usage of control Δu during tuning.
Another crucial extension of the experiments would have been to test more sophisticated types of disturbances. Additionally, valuable insight could also have been acquired from simulations with disturbance affecting other system parameters than the reservoir pressure. However, the experiments showed that the model produced satisfactory results for periodical disturbances, even though it was trained with random system disturbances. This suggests that the specific type and characteristics of the disturbance may be irrelevant, as long as the model is sufficiently trained within the bounds of the actual system disturbance.

7 Conclusion and Further Work

Conclusion

The primary objective of this project was to investigate if the performance of an NMPC controlling an ESP subject to disturbance could be improved by introducing feedforward control with ESN-based inverse models. In Section 1.2, four research questions were formulated to shed light on various aspects of the primary objective. The process of finding answers to these questions resulted in a preliminary project, (Grønningsæter, 2022), as well as this Master's dissertation.

Through the work conducted in the preliminary project, a deeper and necessary understanding of the subject matter was acquired. Furthermore, important baseline components, including the ESP simulator and an ESN framework, were successfully implemented. These components were carried over to the more comprehensive work documented in this dissertation. In *this* work, an NMPC capable of handling both single and multiple inputs and outputs for the ESP was successfully implemented. Additionally, the ESN framework was enhanced to enable the learning of inverse models using time-shifted inputs. These components were ultimately combined into a control scheme that allowed for conducting experiments to address the aforementioned research questions:

I. The results obtained from each experiment (summarized in Chapter 5) indicate that the feedforward-assisted NMPC yields better MAE than the NMPC alone. This suggests that it is indeed possible to improve the performance of the NMPC by introducing feedforward control based on ESN-based inverse models.

II. A comparison of two controllers trained with uniformly and normally distributed noise performed equally well on suppressing a sinusoidal disturbance while tracking a constant reference. However, when a step was made in the reference, the uniform controller outperformed the normal controller by improving the MAE by 10% more. This implies that the choice of training noise clearly affects the ESN's ability to learn the inverse model.

III. The ESN was subject to random disturbance during the training phase in two out of the three conducted experiments. In both cases, uniform noise was used to excite the output. Simulations with the resulting models showed that the MAE was improved by more than 60% when tracking a constant reference while affected by two different sinusoidal disturbances. Although this is approximately 10% lower than the model trained with the actual disturbance, it demonstrates that the ESN can provide satisfactory results without prior knowledge of the actual disturbance. However, it should be noted that the bounds of the actual disturbance were assumed to be known.

IV. An even better performance was obtained by augmenting the controller to also manipulate the valve opening. This became particularly clear in the simulations involving a step in the reference. Unlike the previous controllers that manipulated only the motor frequency, this controller demonstrated satisfactory tracking both

before and after the reference step. These observations suggest that the use of both ESP control inputs does not impair ESN's ability to learn a sufficient inverse model.

With the insight acquired from the process of answering the research questions, it is possible to conclude on the primary objective. The performance of an NMPC controlling an ESP subject to disturbance can indeed be improved by introducing feedforward control with ESN-based inverse models.

Further Work

While there exist similar studies in the literature, this work primarily serves as a proof of concept. Therefore, numerous subjects of interest for future studies emerged throughout this project. Those found most interesting are listed below, along with a brief explanation of their relevance.

- Control the wellhead pressure p_{wh} and the flow $\hat{q}$ in addition to the bottomhole pressure p_{bh}. It can be seen in all the conducted experiments that both the wellhead pressure p_{wh} and the flow $\hat{q}$ suffer from oscillations. Exploring how additional control objectives may counteract these oscillations could be an interesting area for further research.

- Allow some overlap between the control ranges of the NMPC and the ESN. This would require better means to avoid saturation, but it might facilitate better cooperation between the controllers and thus further improvement of the MAE.

- Test other types of noise in the training phase. This can potentially give valuable insight into the role of the excitement noise for different tasks. It might also enable synthesis of controllers specialized for a particular application.

- Train the ESN with a disturbance having different bounds. This could perhaps make it more robust to disturbance with different amplitudes. It could eventually give further insight into the necessity of knowing the bounds of the disturbance.

References

Andersson, J. A. E., Gillis, J., Horn, G., Rawlings, J. B., & Diehl, M. (2019). CasADi – a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, *11*(1), 1–36. doi: https://doi.org/10.1007/s12532-018-0139-4

Antonelo, E. A., Camponogara, E., & Foss, B. (2017). Echo state networks for data-driven downhole pressure estimation in gas-lift oil wells. *Neural Networks*, *85*, 106-117. doi: https://doi.org/10.1016/j.neunet.2016.09.009

Antonelo, E. A., & Schrauwen, B. (2015). On learning navigation behaviors for small mobile robots with reservoir computing architectures. *IEEE Transactions on Neural Networks and Learning Systems*, *26*(4), 763-780. doi: https://doi.org/10.1109/tnnls.2014.2323247

Arrowsmith, D. K., Place, C. M., Place, C., et al. (1990). *An introduction to dynamical systems* (1. ed.). Cambridge university press. doi: https://doi.org/10.1002/zamm.19920721228

Balchen, J. G., Andresen, T., & Foss, B. A. (2016). *Reguleringsteknikk* [Control Theory] (6. ed.). Department of Engineering Cybernetics, Norwegian University of Science and Technology. Retrieved from https://folk.ntnu.no/tronda/regtek-kurs/bok-reguleringsteknikk.pdf

Banderchuk, A. C., Coutinho, D., & Camponogara, E. (2023). Combining robust control and machine learning for uncertain nonlinear systems subject to persistent disturbances. *arXiv preprint arXiv:2303.11890*, 1–7. doi: https://doi.org/10.48550/arXiv.2303.11890

Binder, B. J. T., Kufoalor, D. K. M., Pavlov, A., & Johansen, T. A. (2014). Embedded model predictive control for an electric submersible pump on a programmable logic controller. In *2014 IEEE conference on control applications (CCA)* (pp. 579–585). doi: https://doi.org/10.1109/CCA.2014.6981402

Binder, B. J. T., Pavlov, A., & Johansen, T. A. (2015). Estimation of flow rate and viscosity in a well with an electric submersible pump using moving horizon estimation. *IFAC-PapersOnLine*, *48*(6), 140–146. doi: https://doi.org/10.1016/j.ifacol.2015.08.022

Brown, R. G., & Hwang, P. Y. C. (2012). *Introduction to random signals and applied kalman filtering : with matlab exercises* (4. ed.). Wiley.

Camacho, E. F., & Bordons, C. (2007). *Model Predictive Control* (2. ed.). Springer. doi: https://doi.org/10.1007/978-0-85729-398-5

Chen, C.-T. (1999). *Linear system theory and design* (3. ed.). Oxford University Press.

De Angelis, L., Baglivo, F., Arzilli, G., Privitera, G. P., Ferragina, P., Tozzi, A. E., & Rizzo, C. (2023). Chatgpt and the rise of large language models: the new ai-driven infodemic threat in public health. *Frontiers in Public Health*, *11*, 1166120. doi: https://doi.org/10.3389/fpubh.2023.1166120

Foss, B., & Heirung, T. A. N. (2016). *Merging optimization and control*. Department of Engineering Cybernetics, Norwegian University of Science and Technology. Retrieved from https://folk.ntnu.no/bjarnean/Publications/OptimalControl.pdf

Glad, T., & Ljung, L. (2000). *Control Theory* (1. ed.). CRC press. doi: https://doi.org/10.1201/9781315274737

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. Retrieved from http://www.deeplearningbook.org

Gravdahl, J. T. (2019). *Kybernetikk Introduksjon: Innføring i dynamikk og reguleringsteknikk* [Introduction to Cybernetics: Introduction to dynamics and control theory]. Department of Engineering Cybernetics, Norwegian University of Science and Technology. Retrieved from https://folk.ntnu.no/tomgra/KybintroKompendium.pdf?id=ansatte/Gravdahl_Jan.Tommy/KybintroKompendium.pdf

Grimstad, B. (2022, September). *TTK28 modeling with neural networks - deep learning: Deep feed-forward networks* [PowerPoint slides]. Department of Engineering Cybernetics, Norwegian University of Science and Technology.

Gros, S. (2021). *Modelling and simulation - lecture notes for the NTNU/ITK course TTK4130*. Department of Engineering Cybernetics, Norwegian University of Science and Technology.

Grønningsæter, O. S. (2022). *Black-box modeling of an electric submersible pump lifted well using an echo state network* [Specialization report, TTK4550]. Department of Engineering Cybernetics, Norwegian University of Science and Technology.

Hafner, R., & Riedmiller, M. (2011). Reinforcement learning in feedback control: Challenges and benchmarks from technical process control. *Machine learning*, *84*, 137–169. doi: https://doi.org/10.1007/s10994-011-5235-x

Hou, Z.-S., & Wang, Z. (2013). From model-based control to data-driven control: Survey, classification and perspective. *Information sciences*, *235*, 3–35. doi: https://doi.org/10.1016/j.ins.2012.07.014

Hovd, M. (2009). *Lecture notes for the course advanced control of industrial processes*. Department of Engineering Cybernetics, Norwegian University of Science and Technology. Retrieved from https://folk.ntnu.no/skoge/presentation/OLD-2017-and-earlier/plantwide-course-brasil-july2011/Hovd-Kompendium-2010.pdf

Hovd, M. (2022, April). *TTK4210: Advanced control of industrial processes - week 8: MPC* [PowerPoint slides]. Department of Engineering Cybernetics, Norwegian University of Science and Technology.

Imsland, L. (2007). *Introduction to model predictive control*. Department of Engineering Cybernetics, Norwegian University of Science and Technology.

Jaeger, H. (2001). The "echo state" approach to analysing and training recurrent neural

networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, *148*(34). doi: https://doi.org/10.24406/publica-fhg-291111

Jaeger, H. (2007). Echo state network. *Scholarpedia*, *2*(9), 2330. doi: https://doi.org/10.4249/scholarpedia.2330

Jaeger, H. (2008). *Method for supervised teaching of a recurrent artificial neural network.* U.S. Patent No. 7,321,882. Washington, DC: U.S. Patent and Trademark Office.

Jordanou, J. P. (2019). *Echo state networks for online learning control and MPC of unknown dynamic systems: Applications in the control of oil wells* [Master's dissertation, Federal University of Santa Catarina (UFSC)]. RI UFSC. Retrieved from https://repositorio.ufsc.br/handle/123456789/214380

Jordanou, J. P., Osnes, I., Hernes, S. B., Camponogara, E., Antonelo, E. A., & Imsland, L. S. (2022). Nonlinear model predictive control of electrical submersible pumps based on echo state networks. *Advanced Engineering Informatics*, *52*, 101553. doi: https://doi.org/10.1016/j.aei.2022.101553

Kim, D., Wang, Z., Brugger, J., Blum, D., Wetter, M., Hong, T., & Piette, M. A. (2022). Site demonstration and performance evaluation of MPC for a large chiller plant with TES for renewable energy integration and grid decarbonization. *Applied energy*, *321*, 119343. doi: https://doi.org/10.1016/j.apenergy.2022.119343

Leiner, B. M., Cerf, V. G., Clark, D. D., Kahn, R. E., Kleinrock, L., Lynch, D. C., ... Wolff, S. (2009). A brief history of the internet. *ACM SIGCOMM computer communication review*, *39*(5), 22–31. doi: https://doi.org/10.1145/1629607.1629613

Liu, L., Tian, S., Xue, D., Zhang, T., & Chen, Y. (2019). Industrial feedforward control technology: a review. *Journal of Intelligent Manufacturing*, *30*(8), 2819–2833. doi: https://doi.org/10.1007/s10845-018-1399-6

Lukoševičius, M. (2012). A practical guide to applying echo state networks. In G. Montavon, G. B. Orr, & K.-R. Müller (Eds.), *Neural networks: Tricks of the trade* (2. ed.). Springer Berlin Heidelberg. doi: 10.1007/978-3-642-35289-8_36

Morari, M., & Lee, J. H. (1999). Model predictive control: past, present and future. *Computers & chemical engineering*, *23*(4–5), 667–682. doi: https://doi.org/10.1016/S0098-1354(98)00301-9

Nakamura, G., & Potthast, R. (2015). *Inverse modeling* (1. ed.). IOP Publishing. doi: https://doi.org/10.1088/978-0-7503-1218-9

Nelles, O. (2013). *Nonlinear system identification: from classical approaches to neural networks and fuzzy models* (1. ed.). Springer Berlin, Heidelberg. doi: https://doi.org/10.1007/978-3-662-04323-3

Nocedal, J., & Wright, S. J. (2006). *Numerical optimization* (2. ed.). Springer New York, NY. doi: https://doi.org/10.1007/978-0-387-40065-5

Osnes, I. (2020). *Recurrent neural networks and nonlinear model-based predictive control of an oil well with ESP* [Master's dissertation, Norwegian University of Science and Technology (NTNU)]. NTNU Open. Retrieved from https://hdl.handle.net/11250/2781004

Pavlov, A. K., Krishnamoorthy, D., Fjalestad, K., Aske, E. M. B., & Fredriksen, M. (2014). Modelling and model predictive control of oil wells with electric submersible pumps. *2014 IEEE Conference on Control Applications (CCA)*, 586–592. doi: https://doi.org/10.1109/CCA.2014.6981403

Rice, S. O. (1944). Mathematical analysis of random noise. *The Bell System Technical Journal*, *23*(3), 282–332. doi: https://doi.org/10.1002/j.1538-7305.1944.tb00874.x

Schiller, U. D., & Steil, J. J. (2005). Analyzing the weight dynamics of recurrent learning algorithms. *Neurocomputing*, *63*, 5-23. doi: 10.1016/j.neucom.2004.04.006

Schwenzer, M., Ay, M., Bergs, T., & Abel, D. (2021). Review on model predictive control: an engineering perspective. *International journal of advanced manufacturing technology*, *117*(5-6), 1327–1349. doi: https://doi.org/10.1007/s00170-021-07682-3

Seborg, D., Edgar, T. F., & Mellichamp, D. (2004). *Process dynamics and control* (2. ed.). Wiley India Pvt. Limited.

Takács, G. (2009). *Electrical submersible pumps manual : design, operations, and maintenance* (1. ed.). Gulf Professional Pub./Elsevier.

Vallance, C. (2023, March 30). Elon musk among experts urging a halt to ai training. *BBC*. Retrieved 2023-06-19, from https://www.bbc.com/news/technology-65110030

Wächter, A., & Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, *106*, 25–57. doi: https://doi.org/10.1007/s10107-004-0559-y

Waegeman, T., Schrauwen, B., et al. (2012). Feedback control by online learning an inverse model. *IEEE Transactions on Neural Networks and Learning Systems*, *23*(10), 1637–1648. doi: https://doi.org/10.1109/TNNLS.2012.2208655

Walpole, R., Myers, S., Myers, R., & Ye, K. (2010). *Probability & statistics for engineers and scientists* (9. ed.). Pearson Education.

Wiener, N. (1949). *Extrapolation, interpolation, and smoothing of stationary time series: with engineering applications* (Vols. 113, No.21, pp. 1043–54). MIT press Cambridge, MA.

Zambrano, V., Mueller-Roemer, J., Sandberg, M., Talasila, P., Zanin, D., Larsen, P. G., ... Stork, A. (2022). Industrial digitalization in the industry 4.0 era: Classification, reuse and authoring of digital models on digital twin platforms. *Array*, *14*, 100176. doi: https://doi.org/10.1016/j.array.2022.100176

Appendix A Dynamical Systems Theory

Dynamical systems are defined as systems with states evolving over time (Arrowsmith, Place, Place, et al., 1990). There are two types of dynamical systems based on whether the time is continuous or discrete (Arrowsmith et al., 1990). Many real-world processes are timeseries processes which in general can be represented as a dynamical system. Multiple disciplines and theories utilize dynamical systems theory to describe the systems being studied, with the main difference being the type of dynamics. For instance, control theory studies how to influence the behavior of a dynamical system, while chaos theory studies dynamical systems that are sensitive to initial conditions.

However, it is difficult to derive an exact model representing a real-world process because many real-world processes contain very complex dynamics (Jordanou, 2019). Most models, therefore, approximate these processes instead by omitting these complex dynamics. This will of course decrease the precision of the model, but not necessarily invalidate it. In some cases, it can even facilitate a deeper understanding of the overall system because simpler models allow simpler computations. As the authors of (Jordanou, 2019) argue, models should ideally be as simple as the application allows. The rest of this chapter is based on (Chen, 1999) and will give an introduction to existing and widely accepted fundamental theory of dynamical systems as well as how they can be represented.

A.1 Dynamical Systems Fundamentals

A *system* can be defined as the relationship between input and output, where the system must produce unique output for a given input (Chen, 1999). This is a widely accepted definition in the literature. Dynamical systems are typically categorized into four classes based on their number of inputs and outputs:

- SISO: Single-input, single-output
- SIMO: Single-input, multiple-output
- MISO: Multiple-input, single-output
- MIMO: Multiple-input, multiple-output

These systems are illustrated schematically in Figure 28

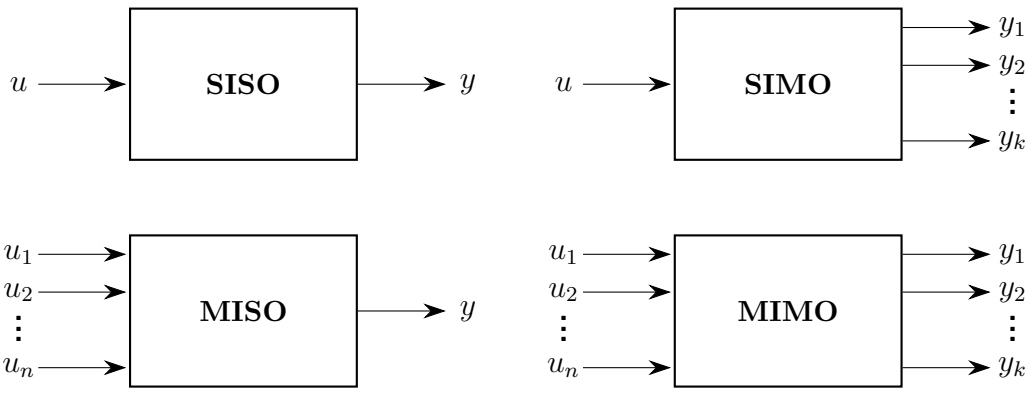


Figure 28: Schematic illustration of a SISO, SIMO, MISO, and MIMO system.

Furthermore, systems are commonly categorized based on the characteristics of their inputs and outputs and the dependency between these. The following sections will describe the most fundamental categorizations with a basis in the definitions given in (Chen, 1999).

Continuous-time and Discrete-time Systems

Continuous-time dynamical systems take continuous signals as inputs and produce continuous-time signals as outputs. These signals are continuous functions of time which in terms of mathematics means that the function $f(t)$ approaches the function value $f(a)$ for a given point a in the interval from all directions. Thus, given any point τ from the interval $t \in [-\infty, \infty]$, $f(t)$ is continuous if and only if

$$\min_{t \to \tau^+} f(t) = \min_{t \to \tau^-} f(t) = \min_{t \to \tau} f(t) = f(\tau) \tag{44}$$

This is a rigorous mathematical definition of continuity. A simpler but still sufficient interpretation of continuity is that a continuous function in a closed interval will contain infinitely many points regardless of the size of the interval. In the opposite case, where the closed interval has a finite number of points, the dynamical system is called *discrete*. The number of points in this interval remains finite even though the length of the interval approaches infinity.

Since time, in general, is continuous, it follows that all real-world processes are naturally continuous. Measurements, on the other hand, are discrete due to being an observation of a system at a given point in time (usually referred to as a sample). Depending on the application, a continuous or discrete model will be derived from collected samples. Using discrete time often proves sufficient as illustrated in Figure 29. Moreover, in order to simulate a dynamical system on a computer, a discrete-time model is required due to machine precision being finite.

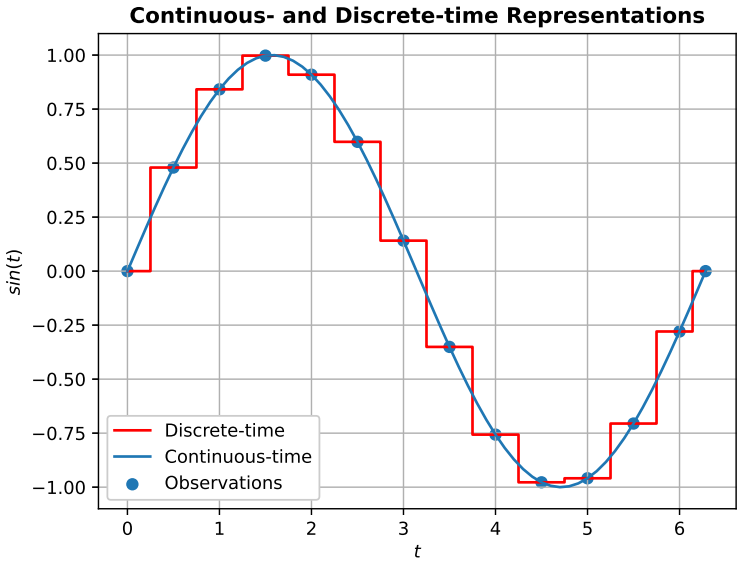


Figure 29: Illustration of a discrete-time sine wave approximating the corresponding continuous-time sine wave given a finite number of samples.

Causality and States

Causality describes the system's dependency on previous, current, and future inputs. If a system output only depends on the current input, it is considered *memoryless*. However, most systems of interest have memory, meaning they depend on the current as well as the past and/or future values. If the system's output depends on its current and previous inputs, but not its future inputs it is called a *causal* system. In the opposite case, it is called a noncausal system. This implies that a system is able to predict its future inputs. There exist no physical systems with this capability (Chen, 1999).

In theory, causal system outputs depend on all past inputs back to $-\infty$. This is clearly computationally infeasible and has led to the concept of system *states*. The author of (Chen, 1999) defines a state as *the information at time* τ *that together with the input* $u(t)$ *for* $t \geq \tau$ *uniquely determines the output* $y(t)$ *for all* $t \geq \tau$. In other words, one can consider states as functions summarizing the effect of all previous inputs. Utilization of states in dynamical systems will generally simplify and make calculations and modeling more efficient. If a dynamical system contains a finite number of state variables, it is called a *lumped* system. In the other case, it is called a *distributed* system.

Linear and Nonlinear Systems

Linearity and nonlinearity describe the relationship between a particular state x_k and the next state x_{k+1} in a system. Linear systems are a special case of nonlinear systems, and they are generally preferred due to their well-behaved nature and the large body of existing theory. A system is linear if and only if it possesses both the additivity and the homogeneity property. In mathematical terms, a function f is linear if and only if

$$\alpha f(x) + \beta f(y) = f(\alpha x + \beta y) \tag{45}$$

for any α, β, x and y. Real-world systems are, nevertheless, almost always nonlinear. Yet, nonlinearity can be more eminent in specific operational regions allowing linear approximations of other operational regions. Saturation is a well-known example of regional nonlinearity within control theory. The closed-loop system can be well-behaved until it saturates which introduces strong nonlinearity into the system.

Time-variant and Time-invariant Systems

Dynamical systems that produce different outputs if given the same input at different points of time, are called time-variant. These systems have time-dependent model variables in addition to their states. Time-invariant systems, on the other hand, will produce the same output for a given input regardless of time. Real-world processes are generally time-variant but at very different rates. A given process might be very slowly varying enabling it to be modeled as time-invariant which can potentially expedite calculations.

A.2 Modeling of Dynamical Systems

Dynamical systems usually refer to causal systems that may possess one or many of the properties discussed in Section A.1. These system representations can be used to model a real-world process where the inputs are variables to be manipulated, and the output can be measured. However, deriving a perfect model of a real-world process is for all practical matters impossible. This is because models should be thought of as approximations of real-world events and not replications. The quality of a model can be evaluated by comparing predicted values with measured values from the real process. Thus, a measure of the model quality is its precision. Fortunately, there are many applications not demanding exact precision meaning a less complex model is sufficient. As suggested by the author in (Jordanou, 2019), one should try to make as simple models as possible because this generally enhances the understanding of the problem and reduces the computational demand through simplifying calculations.

The rest of this section discusses model representations and is extracted from (Grønningsæter, 2022). System models are usually given as differential equations. Differential equations can be used to describe how states or variables vary over time (Gravdahl, 2019). These equations enable a dense description of system state-spaces, which also can be transformed into the complex domain resulting in transfer functions. Hence, differential equations are powerful mathematical tools when it comes to modeling physical systems. A state-space can be expressed using *ordinary* differential equations (ODE) which generally relate the output y of a system to the input u (Gros, 2021). These equations are typically given on the form:

$$\begin{aligned} \dot{x} &= f(x, u) \\ y &= h(x, u) \end{aligned} \tag{46}$$

where f and h can be either linear or nonlinear. Another benefit of using ODEs is that these systems are generally straightforward to express in discrete-time enabling them to be solved numerically. Yet, there is no guarantee that there will exist ODEs defining the entire state directly (Gros, 2021). Another type of differential equations, omitting this problem is differential algebraic equations (DAE). These are, simply put, systems of differential equations and algebraic equations and are typically given on the form:

$$\begin{aligned} \dot{x} &= f(x, y) \\ 0 &= g(x, y) \end{aligned} \tag{47}$$

The equations used in the ESP simulation in this dissertation are represented by DAEs.

A.3 State-space Representations and Transfer Functions

Linear time-invariant (LTI) dynamical systems can be represented using a state-space representation or through a transfer function. The former is a matrix representation of the dynamical system in the following form:

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{48}$$
$$y(t) = Cx(t) + Du(t) \tag{49}$$

where A, B, C and D are system matrices and x and u are the state and input vector, respectively. In addition, (48) and (49) may also be appended with another term describing how measurable system disturbances and measurement noises affect the system.

Any lumped LTI system can be represented with a state-space model since these properties make the systems a linear combination of their states and inputs. This representation also offers valuable insight into the system dynamics from calculating the Eigenvectors of A. Furthermore, given A, B, C, and D it is possible to investigate if the system is controllable and/or observable. The former indicates whether or not a state is steerable from the input. The latter indicates if a state can be estimated from the output. These concepts are also dual meaning that the matrix pair (A, B) is controllable if and only if (A^T, B^T) is observable and vice versa.

Transfer functions represent the system in the Laplace domain in terms of frequency instead of time. The transfer function can be found by taking the Laplace transform of the state-space representation. A given system can be represented with infinitely many state-spaces, but it can only have one transfer function. Hence, the state-space representation is considered a more general representation than the transfer function. The process of transforming a transfer function to a state-space representation is called a *realization* which is not a uniquely defined process unless it is *minimal*. This refers to realizing the state-space with the minimal number of states.

When the state-space representation is unavailable, one can also find the transfer function of a system by taking the Laplace transform of the system's impulse response. This response is obtained when exciting the system with the Dirac delta function $\delta(t)$. Note that this function is a mathematical abstraction of a function with infinite amplitude, no duration, and a total area of one. The input-output relation of linear SISO systems can be described through the zero-state response which is generally given by

$$y(t) = \int_{-\infty}^{\infty} g(t,\tau)u(\tau)d\tau \tag{50}$$

where $g(t,\tau)$ represents the impulse response, and $u(\tau) = \delta(t-\tau)$ is the impulse at time τ. Assuming that the system is LTI, (50) can be reduced to

$$y(t) = \int_0^t g(t-\tau)u(\tau)d\tau = \int_0^t g(\tau)u(t-\tau)d\tau \tag{51}$$

by further assuming that (51) is causal and taking the Laplace transformation one can obtain the following input-output relation in the Laplace domain

$$\frac{\hat{y}(s)}{\hat{u}(s)} = \hat{g}(s) = \int_0^\infty g(t)e^{-st}dt \tag{52}$$

where $g(t)$ is the impulse response and $\hat{g}(s)$ is the corresponding Laplace transform. Since ODEs describe the evolution of a system over time, the system's transfer function can be found by taking the Laplace transform of the ODE. This usually results in a rational $\hat{g(s)}$ on the following form

$$\hat{g}(s) = \frac{b(s)}{a(s)} = K\frac{(s-z_1)(s-z_2)...(s-z_m)}{(s-p_1)(s-p_2)...(s-p_n)} \tag{53}$$

where b and a are polynomials of s. The roots of these polynomials are referred to as poles and zeros, respectively. Their location provides qualitative information on the characteristics of the system response. Poles and zeros can together with the gain constant K, which is found by solving for $s = 0$, give the complete characteristics of a differential equation. Due to being relatively simple to derive and the large dynamical insight transfer functions offer, they are probably the most popular representation in control theory.

A.4 Dynamical Systems Inversion

Up until now, only *forward modeling* has been discussed. *Forward* in this context relates to the direction of time because observations of the causes are used to calculate the effects in the future. Hence, a forward model is in fact a mapping H from some state x to some measurement y (Nakamura & Potthast, 2015):

$$H : x \to y \tag{54}$$

Models in classical control theory are therefore forward models since some state is mapped to some output that can be measured. However, this requires that all model parameters are directly observable which is not always the case in many control applications (Chen, 1999). These parameters can instead be estimated using an *inverse model* which is the opposite of a forward model. Inverse models use the observations of the effects to calculate the cause. Continuing with the same notation, the general inverse problem is, according to (Nakamura & Potthast, 2015), equivalent to finding solutions to the following equation:

$$H(x) = y \iff F(x) := H(x) - y = 0 \tag{55}$$

where y is known. This can be generally very difficult due to system complexity and nonlinearity. Yet, these models are highly desirable in many applications because of their ability to provide information both directly and indirectly about unobservable parameters. Furthermore, the forward model is not necessarily available suggesting that one can just as well find the inverse model directly. Due to the importance of these models, there exist many methods to achieve this. Perhaps one of the most fundamental methods for approximating a model based on observations is the *least squares method*. This method is equivalent to the general inverse problem defined in

(55) (Nakamura & Potthast, 2015). The objective of the least squares method is to minimize the following cost function:

$$J(x) = ||H(x) - y||_2^2 \tag{56}$$

This allows for the inverse problem to be solved as a minimization problem where any optimization method may be applied. However, this formulation must be approached carefully since it does not offer any insight into the structure, instability, and non-uniqueness of the problem at hand (Nakamura & Potthast, 2015).

Inverse Black-Box Modeling Using Artificial Intelligence

The process of finding a dynamical model from observations is commonly referred to as *system identification*. There exist many methods for system identification, and a way to distinguish between different methods is to categorize them after the use of observed data. The author of (Nelles, 2013) categorizes system identification methods from white-box to black-box modeling. White-box modeling refers to having full insight into how the model parameters are derived, and these methods are based on first principles such as Newton's second law, mass balances, etc. Black-box modeling, on the other hand, refers to models derived purely from estimations. Black-box models offer little to no insight into how the model relates the inputs and outputs. Nor do these methods yield explicit differential equations for the dynamical system. Gray-box modeling refers to modeling techniques falling in between white- and black-box modeling techniques.

As demonstrated in (Jordanou, 2019), (Jordanou et al., 2022), and (Grønningsæter, 2022), supervised AI is a powerful tool for black-box modeling. In all three reports, they were able to successfully use an ESN to learn a nonlinear dynamical system. Details of the ESN are covered in Section 2.4. The latter was also able to use an LSTM to learn the same system with a high level of generalization. For further details on these results and the LSTM, the reader is referred to (Grønningsæter, 2022). As demonstrated in (Waegeman et al., 2012), (Jordanou, 2019), and (Banderchuk, Coutinho, & Camponogara, 2023), it is possible to use black-box modeling to also learn the inverse model of a dynamical system. This is further supported by investigating the general inverse problem (55). When parameters are approximated using black-box models there is no way for these methods to tell whether it is the cause or the effect that is given as input and output. This is illustrated in Figure 30.

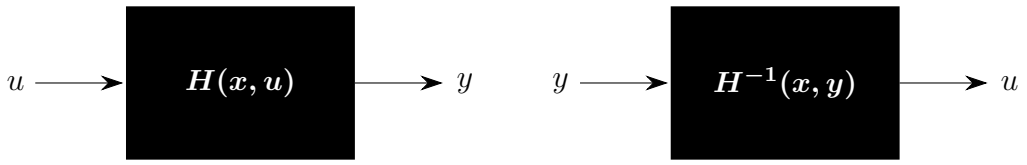


Figure 30: Black-box modeling of forward and inverse model.

Appendix B Classical Control Theory

Control theory is a field within applied mathematics regarding the control of dynamical systems. It lays the foundation for control engineering which is a branch in engineering cybernetics concerning control, communication, and database processing in technical systems (Balchen, Andresen, & Foss, 2016). The purpose of control engineering is the synthesis of control strategies and methods to regulate the behavior of a dynamical system to some predetermined objective (Balchen et al., 2016). This chapter will give a brief introduction to some selected topics within classical control theory considered relevant to the understanding of this dissertation. More specifically, feedback control, disturbance, noise, and feedforward control. For the sake of simplicity, all systems are assumed LTI. This is generally a common assumption in the literature.

B.1 Feedback Control

Feedback control is probably the most fundamental concept in control theory. The feedback loop is used to inform the controller about the deviation from the predetermined objective (commonly referred to as the system reference y^{ref}). Feedback control can also stabilize unstable systems if the controller is synthesized properly (Hovd, 2009). A system with feedback control is illustrated in Figure 31.

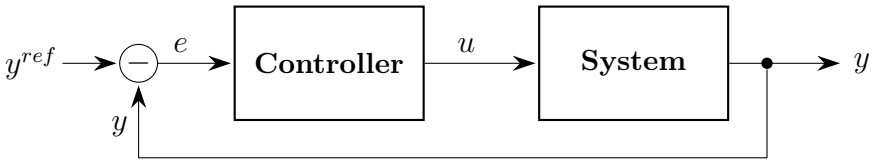


Figure 31: Schematic illustration of a general system with feedback control.

Every control system requires a feedback loop in order to correct deviations between the reference y^{ref} and the output y. Assuming that the controller and the system in Figure 31 are represented with the transfer functions $g_{ctrl}(s)$ and $g_{sys}(s)$, the output y can be expressed as

$$y = \frac{g_{ctrl}(s)g_{sys}(s)}{1 + g_{ctrl}(s)g_{sys}(s)}\, y^{ref} \tag{57}$$

which shows that the output also depends on the feedback. In a perfect world, feedback would be redundant. However, any control application is subject to uncertainty in the form of *disturbances* and *noise*. These concepts will be discussed in detail in Section B.2. In advanced control theory, also model errors can introduce uncertainty. These are the main reasons why feedback is necessary for almost any control application.

B.2 Disturbance and Noise in Control Applications

Disturbances and noise are always present in any control application. Both terms refer to unwanted signals affecting either the controlled variable or a measurement (Glad & Ljung, 2000), and it is common to see the terms being used interchangeably in the literature. One way of distinguishing disturbance and noise is to define them on the basis of their origins and effects. This way, disturbance can be defined as *an unwanted signal, originating from external factors, which affects the system output directly.* Disturbances are typically caused by parameter variations, model uncertainties, etc. (Glad & Ljung, 2000). It is also common to distinguish between measurable and unmeasurable disturbances. Noise, on the other hand, can be defined as *an unwanted signal, originating from any other factor, affecting the output, but not the control signal directly.* When introducing feedback to a control system, noise becomes capable of affecting the control output indirectly. Noises are typically caused by measurement errors in the sensors, random fluctuations, etc. (Glad & Ljung, 2000).

Consequently, there is a fundamental difference in where disturbance and noise enter a general system when using these definitions. Disturbance d will enter the system directly, while noise does not enter before the original system output z is measured, giving the system output $y = z + n$. This is illustrated in Figure 32 where u denotes the system input.

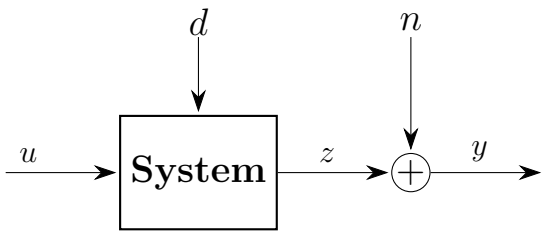


Figure 32: Illustration of disturbance and noise entrance in a general system.

B.3 Feedforward Control

Feedforward control is generally used to counteract measurable disturbances before they affect a system (Balchen et al., 2016). These controllers can effectively suppress measurable disturbances if a sufficiently accurate model of how the disturbance affects the system and a model of the system itself are available. A schematic illustration of a system with both a feedforward and feedback controller is shown in Figure 33. The feedback controller is included because feedforward control is insufficient to stabilize an unstable system by itself (Hovd, 2009). It is, therefore, normally utilized in combination with a feedback controller. Moreover, feedforward controllers can not make a closed-loop system unstable, and they are often useful to avoid saturation when large disturbances are present (Hovd, 2009).

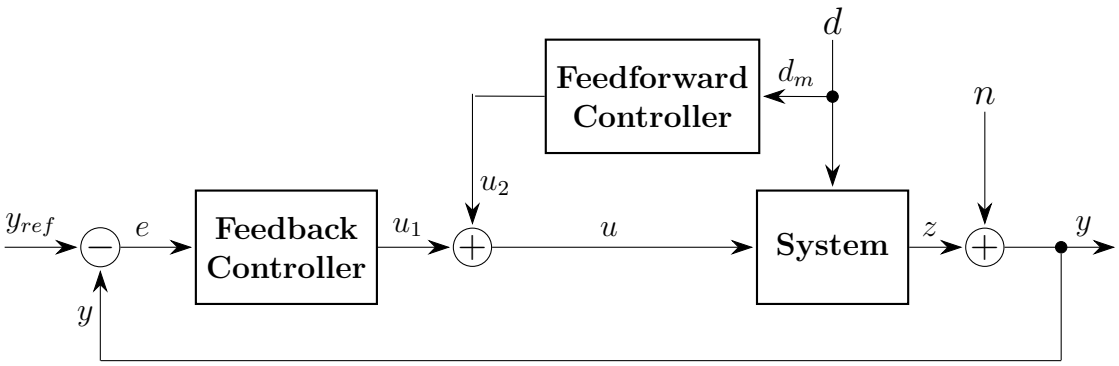


Figure 33: Schematic illustration of a general system with stabilizing feedback control and disturbance suppressing feedforward control.

Using the same notation as in Section B.1, the output y from u in the system from Figure 33 can be expressed as

$$y = g_{sys}(s)(u_1 + u_2 + g_d(s)d_m) \tag{58}$$

where $g_d(s)$ is the transfer function of the disturbance. By also omitting u_1 and solving for $y = 0$, the optimal feedforward controller can be found and is given by

$$u_2 = -g_{sys}^{-1}(s)u_1 g_d(s)d_m \tag{59}$$

This equation shows that the optimal feedforward controller depends on the inverse system transfer function $g_{sys}^{-1}(s)$. In many industrial applications, identifying the system transfer function can be challenging or even impossible. Obtaining its inverse is often even more challenging. To make matters worse, even if the transfer function is known, it can be non-invertible due to uncancellable zeros (Liu, Tian, Xue, Zhang, & Chen, 2019).

Further inspection of Figure 33 reveals that feedforward control can not suppress the measurement noise. This is because the measurement noise n does not affect the system output z, but rather the measured output y.
Controllers aiming to correct the system output with respect to the measurement noise should ideally be implemented in the feedback loop. However, it is generally impossible

to measure the measurement error for a particular sensor online. Measurement noise can yet be dealt with using state estimators (Chen, 1999). This method is outside the scope of this dissertation, but the general idea is to use a system model along with the statistical properties of the sensors to estimate the actual system output. Thus, the feedback value becomes an estimate instead of the actual measurement.

Appendix C Reversed NMPC Experiments

C.1 Reverse Step Response using Both Control Inputs

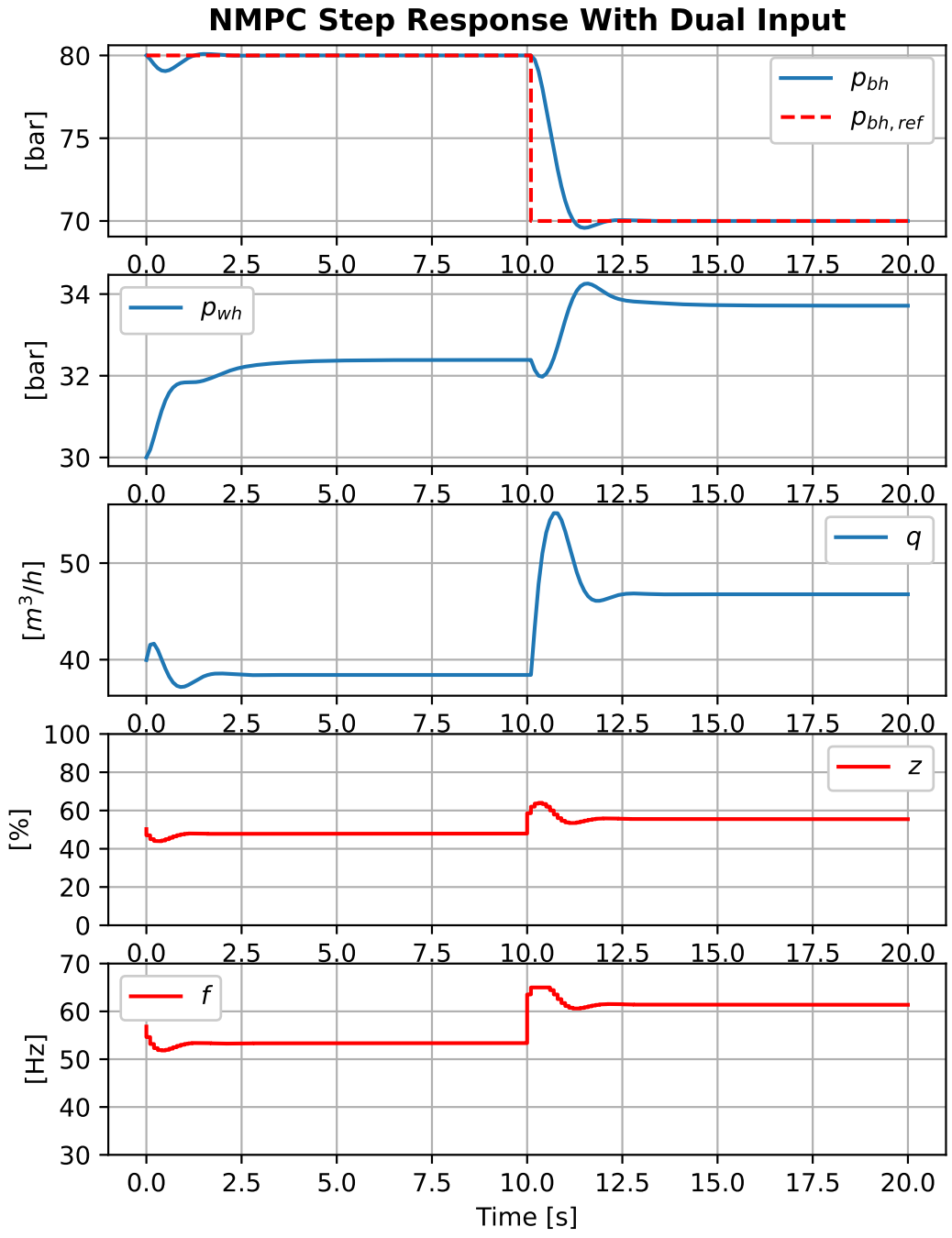


Figure 34: Reverse step response from using both ESP control inputs and the weight matrices from (42).

C.2 Reverse Step Response using Only f as Control Input

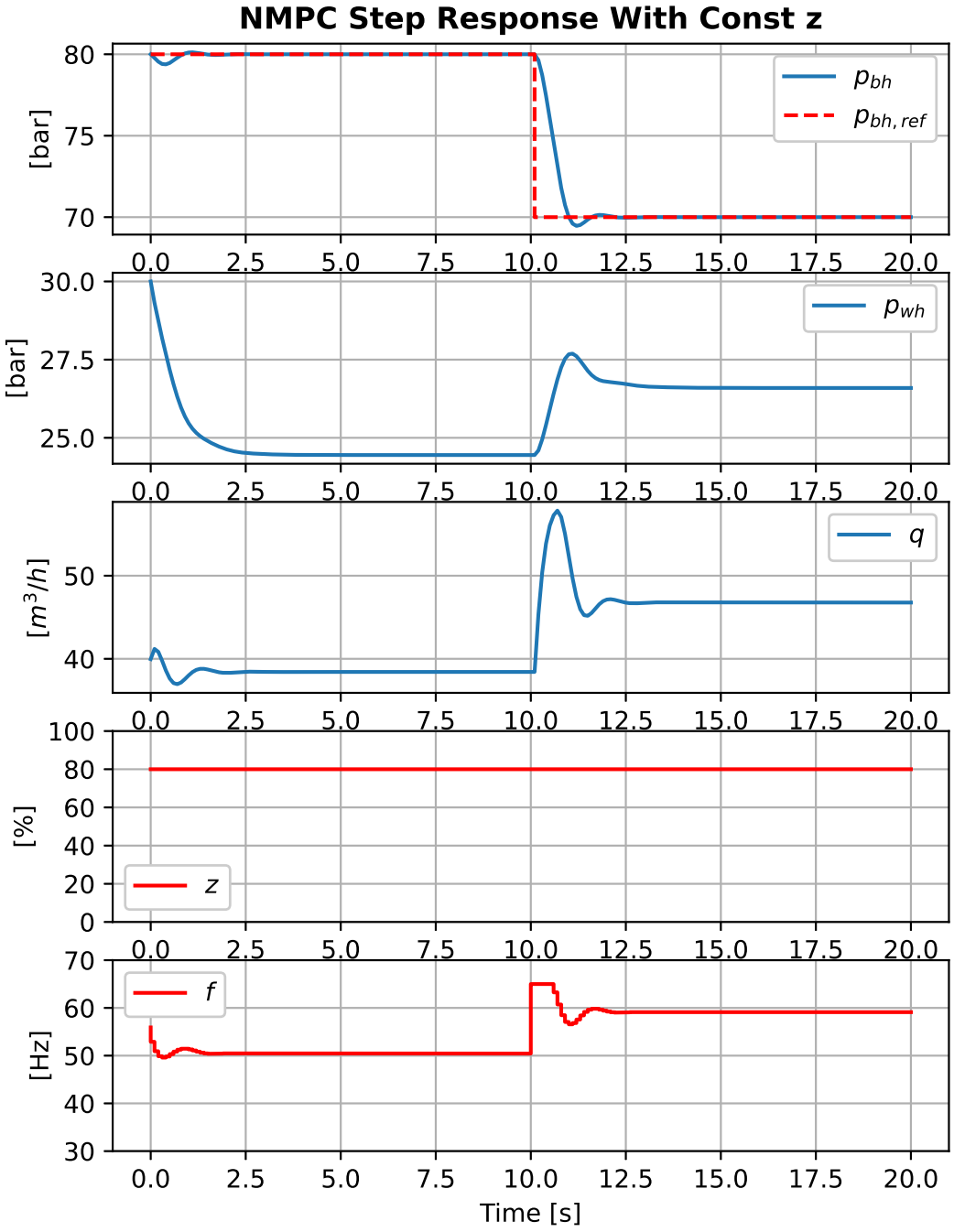


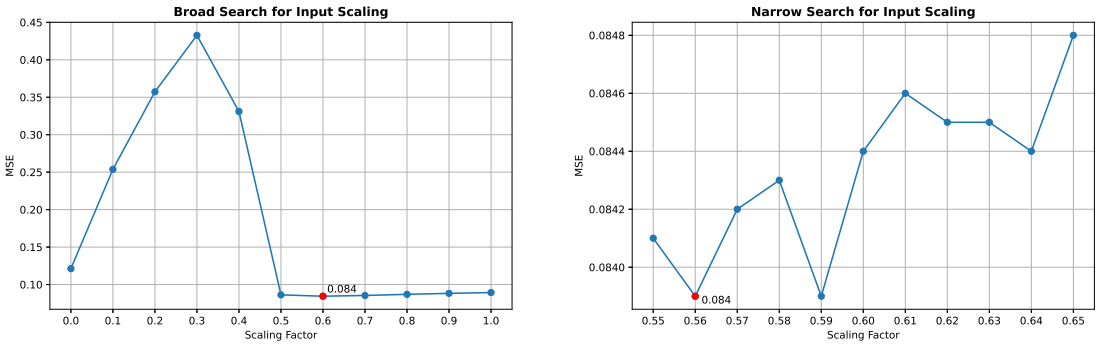
Figure 35: Reverse step response from using only the motor frequency f as control input with $R = 60$.

Appendix D Hyperparameters for the Normal Model in Experiment 1

The reservoir size (N_x) and spectral radius ($\rho(\boldsymbol{W}_r)$) were set to 500 and 0.999, respectively, for the same reasons explained in Section 4.3. All remaining parameters were determined using the same strategy as in the aforementioned section.

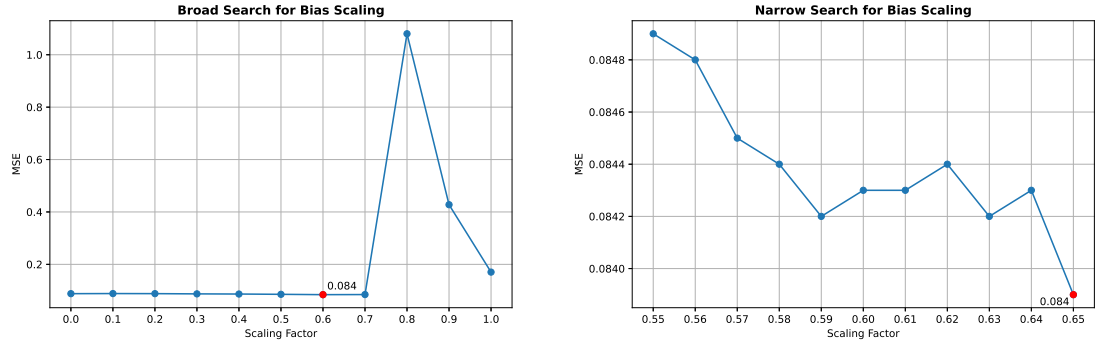
Search for Input and Bias Scaling

Figure 36 and 37 show the conducted searches to determine the input and bias scaling used in the normal model in Experiment 1. From inspection, it is evident that an input scaling of 0.56 and a bias scaling of 0.65 yield the lowest MSEs.



(a) Broad search. (b) Narrow search.

Figure 36: Broad (a) and narrow (b) search for the input scaling for the normal model in Experiment 1.

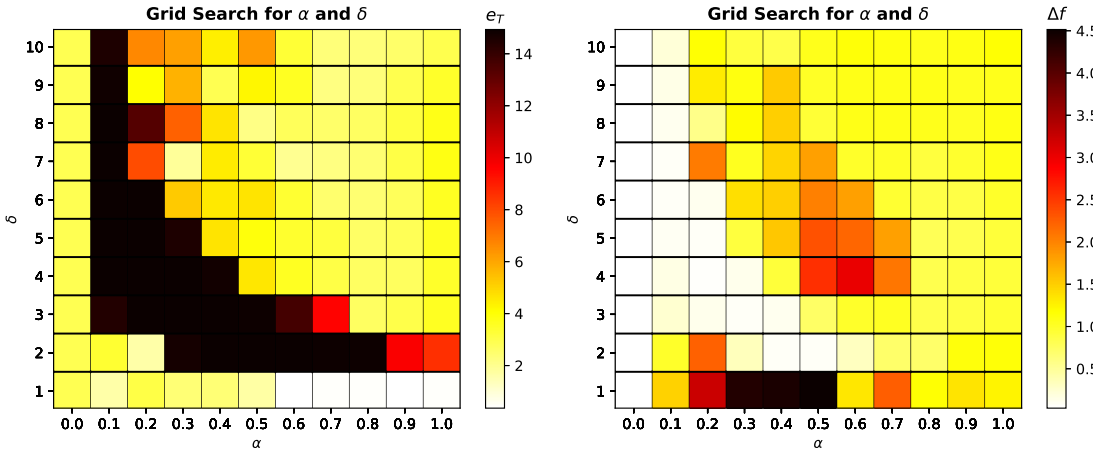


(a) Broad search. (b) Narrow search.

Figure 37: Broad (a) and narrow (b) search for the bias scaling for the normal model in Experiment 1.

Grid Search for δ, α, and β

Figure 38 shows the results from the grid search conducted to find δ, α, and β. Through inspection and further experimentation, it was found that using $(0.9, 1)$ with a β of 10^{-7} yielded the most satisfactory control performance.



(a) MAE (e_T) per grid element.

(b) Δf per grid element.

Figure 38: Grid search for (α, δ) for the normal model in Experiment 1. Each set is measured in MAE (a) and Δf (b) per grid element.

Tracking of Normal Distributed White Noise

The hyperparameter values obtained in this section are summarized in Table 4. Figure 39 shows a snapshot of the model tracking the test set using these values, resulting in an MSE of 0.0890. Visual inspection confirms that the model is capable of recognizing the tendency in the noise.

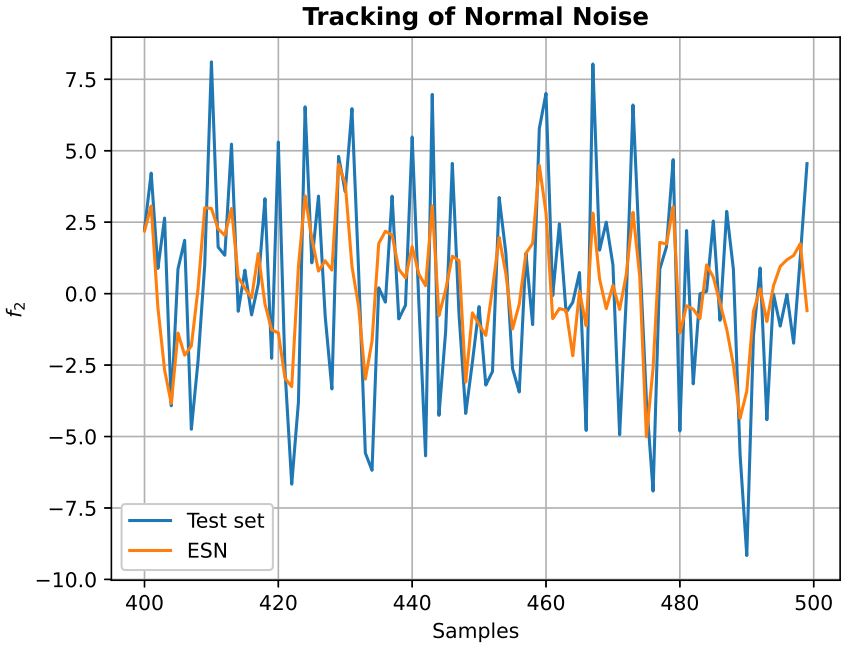


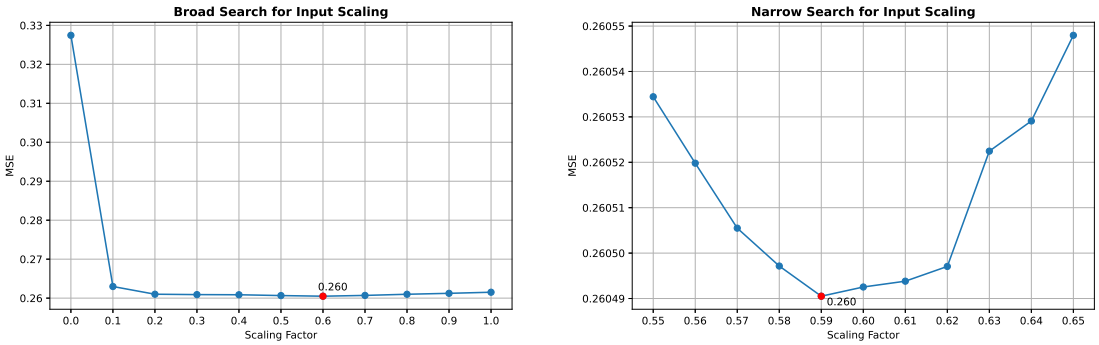
Figure 39: Uniform model in Experiment 1 tracking the test set using the hyperparameters from Table 4.

Appendix E Hyperparameters for the Model in Experiment 2

The reservoir size (N_x) and spectral radius ($\rho(\boldsymbol{W}_r)$) were set to 500 and 0.999, respectively, for the same reasons explained in Section 4.3. All remaining parameters were determined using the same strategy as in the aforementioned section.

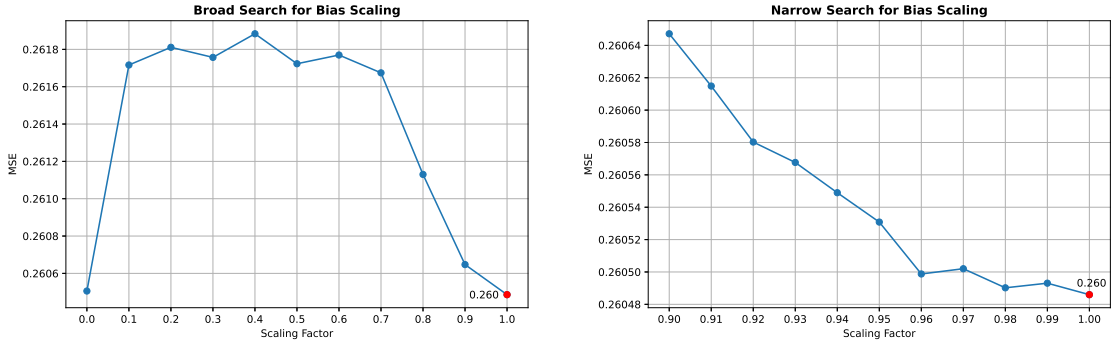
Search for Input and Bias Scaling

Figure 40 and 41 show the conducted searches to determine the input and bias scaling used in the model in Experiment 2. From inspection, it is evident that an input scaling of 0.59 and a bias scaling of 1.00 yield the lowest MSEs.



(a) Broad search. (b) Narrow search.

Figure 40: Broad (a) and narrow (b) search for the input scaling for the model in Experiment 2.

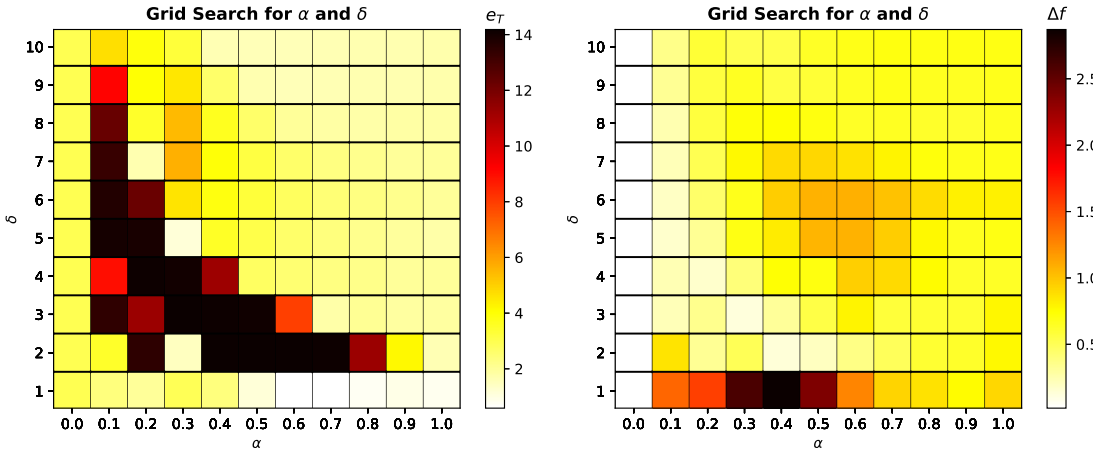


(a) Broad search. (b) Narrow search.

Figure 41: Broad (a) and narrow (b) search for the bias scaling for the model in Experiment 2.

Grid Search for δ, α, and β

Figure 42 shows the results from the grid search conducted to find δ, α, and β. Through inspection and further experimentation, it was found that using $(0.6, 1)$ with a β of 1 yielded the most satisfactory control performance.



(a) MAE (e_T) per grid element.

(b) Δf per grid element.

Figure 42: Grid search for (α, δ) for the model in Experiment 2. Each set is measured in MAE (a) and Δf (b) per grid element.

Tracking of Uniform Distributed White Noise

The hyperparameter values obtained in this section are summarized in Table 7. Figure 43 shows a snapshot of the model tracking the test set using these values, resulting in an MSE of 0.0890. Visual inspection confirms that the model is capable of recognizing the tendency in the noise.

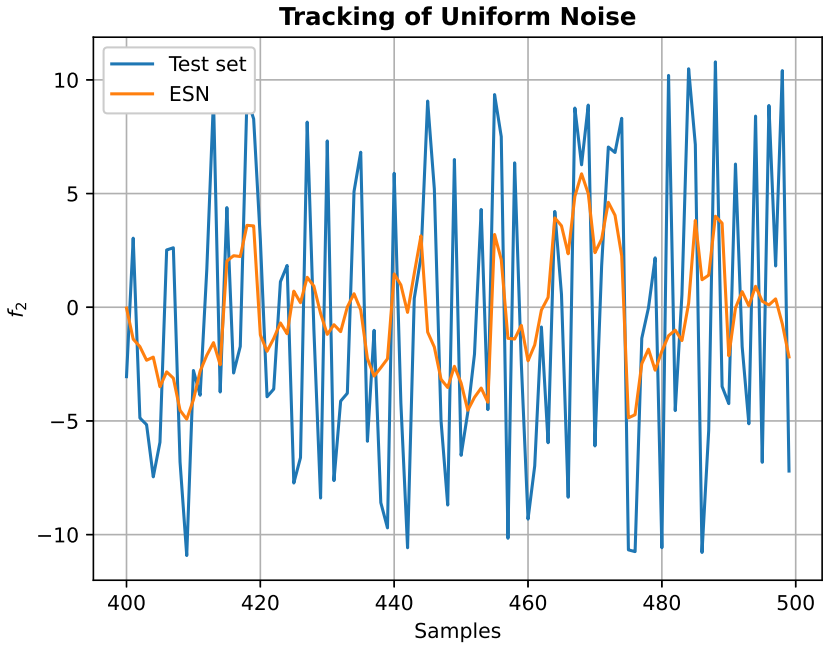


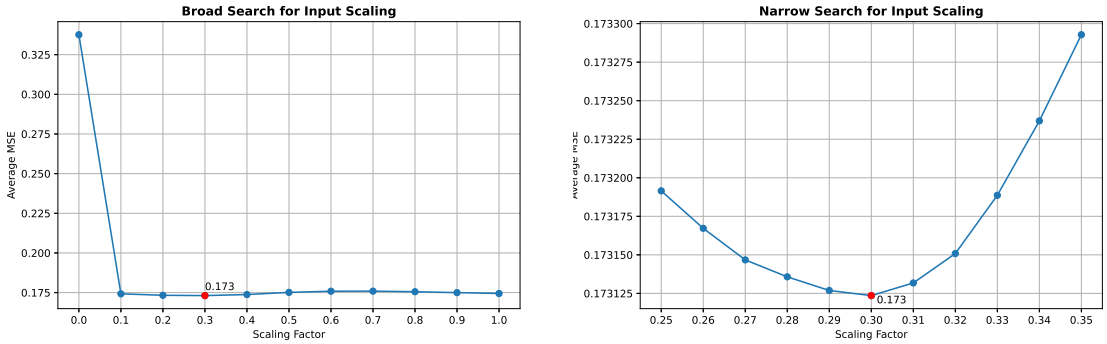
Figure 43: Uniform model in Experiment 2 tracking the test set using the hyperparameters from Table 7.

Appendix F Hyperparameters for the Model in Experiment 3

The reservoir size (N_x) and spectral radius ($\rho(\boldsymbol{W}_r)$) were set to 500 and 0.999, respectively, for the same reasons explained in Section 4.3. All remaining parameters were determined using the same strategy as in the aforementioned section.

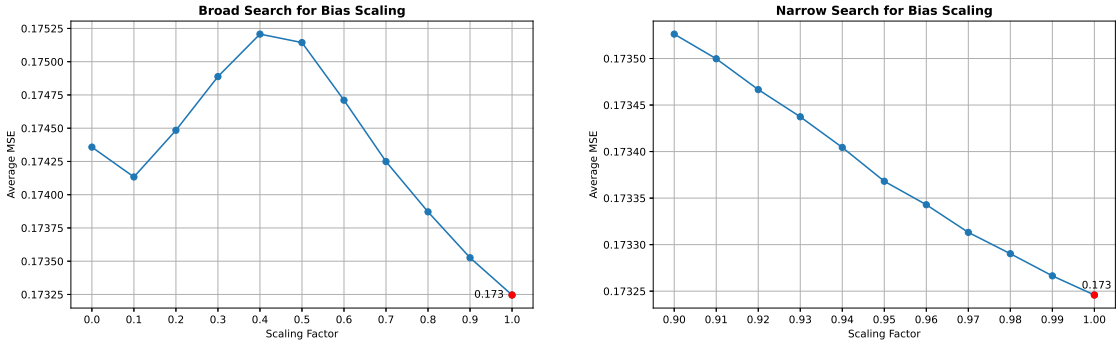
Search for Input and Bias Scaling

Figure 44 and 45 show the conducted searches to determine the input and bias scaling used in the model in Experiment 3. From inspection, it is evident that an input scaling of 0.30 and a bias scaling of 1.00 yield the lowest MSEs.



(a) Broad search. (b) Narrow search.

Figure 44: Broad (a) and narrow (b) search for the input scaling for the model in Experiment 3.



(a) Broad search. (b) Narrow search.

Figure 45: Broad (a) and narrow (b) search for the bias scaling for the model in Experiment 3.

Grid Search for δ, α, and β

Figure 46 and 47 show the results from the grid search conducted to find δ, α, and β. Through inspection and further experimentation, it was found that using $(0.6, 1)$ with a β of 1 yielded the most satisfactory control performance.

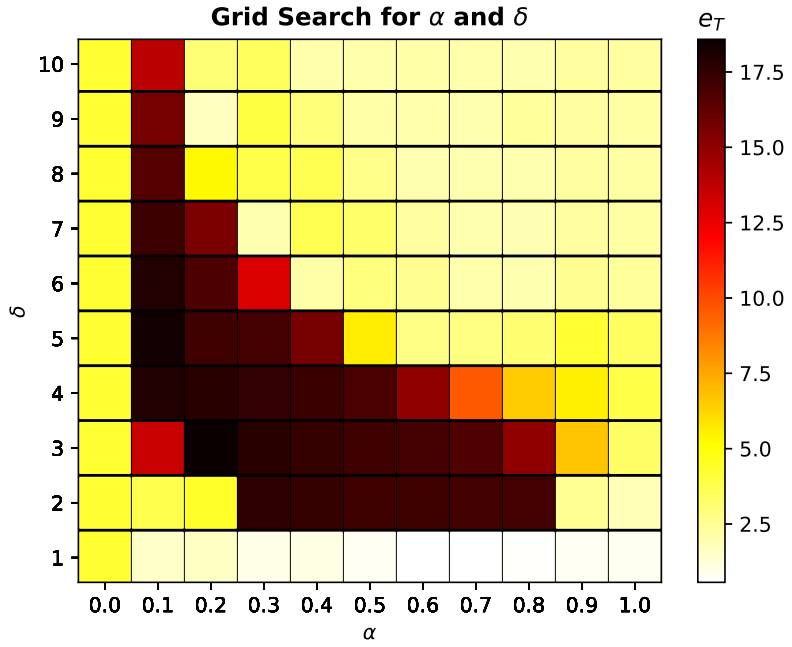
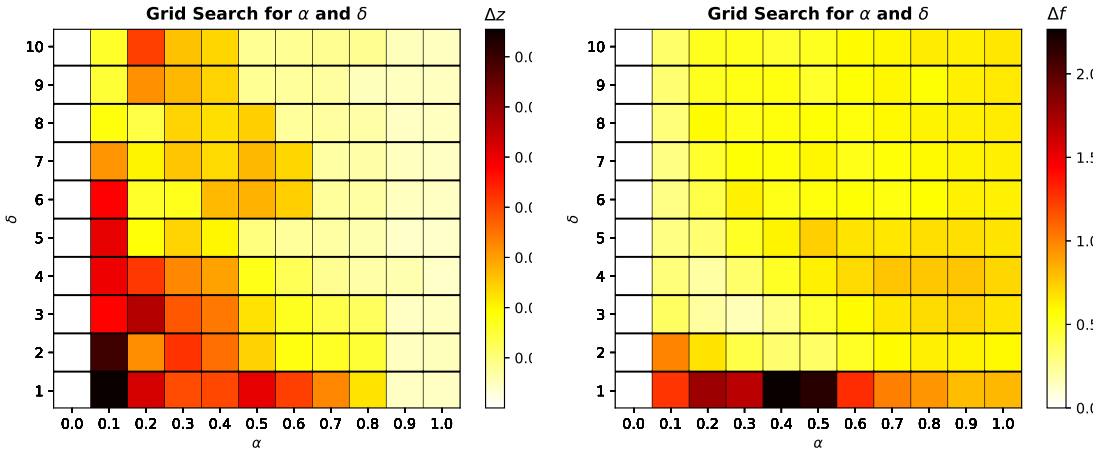


Figure 46: Grid search for (α, δ) for the model in Experiment 3. Each set is measured in MAE (e_T) per grid element.



(a) Δz per grid element.

(b) Δf per grid element.

Figure 47: Grid search for (α, δ) for the model in Experiment 3. Each set is measured in Δz (a) and Δf (b) per grid element.

Tracking of Uniform Distributed White Noise

The hyperparameter values obtained in this section are summarized in Table 12. Figure 48 shows snapshots of the model tracking the test set using these values. The trackings yielded an MSE of 0.08257 and 0.2625 for the valve opening and frequency, respectively, resulting in an average MSE of 0.1725. Visual inspection confirms that the model is capable of recognizing the tendency in each of the noises.

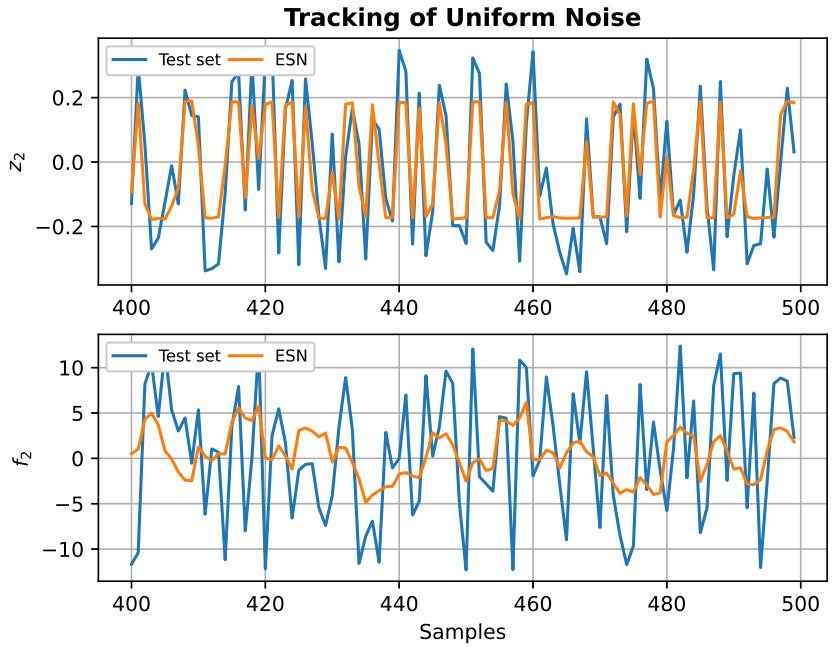
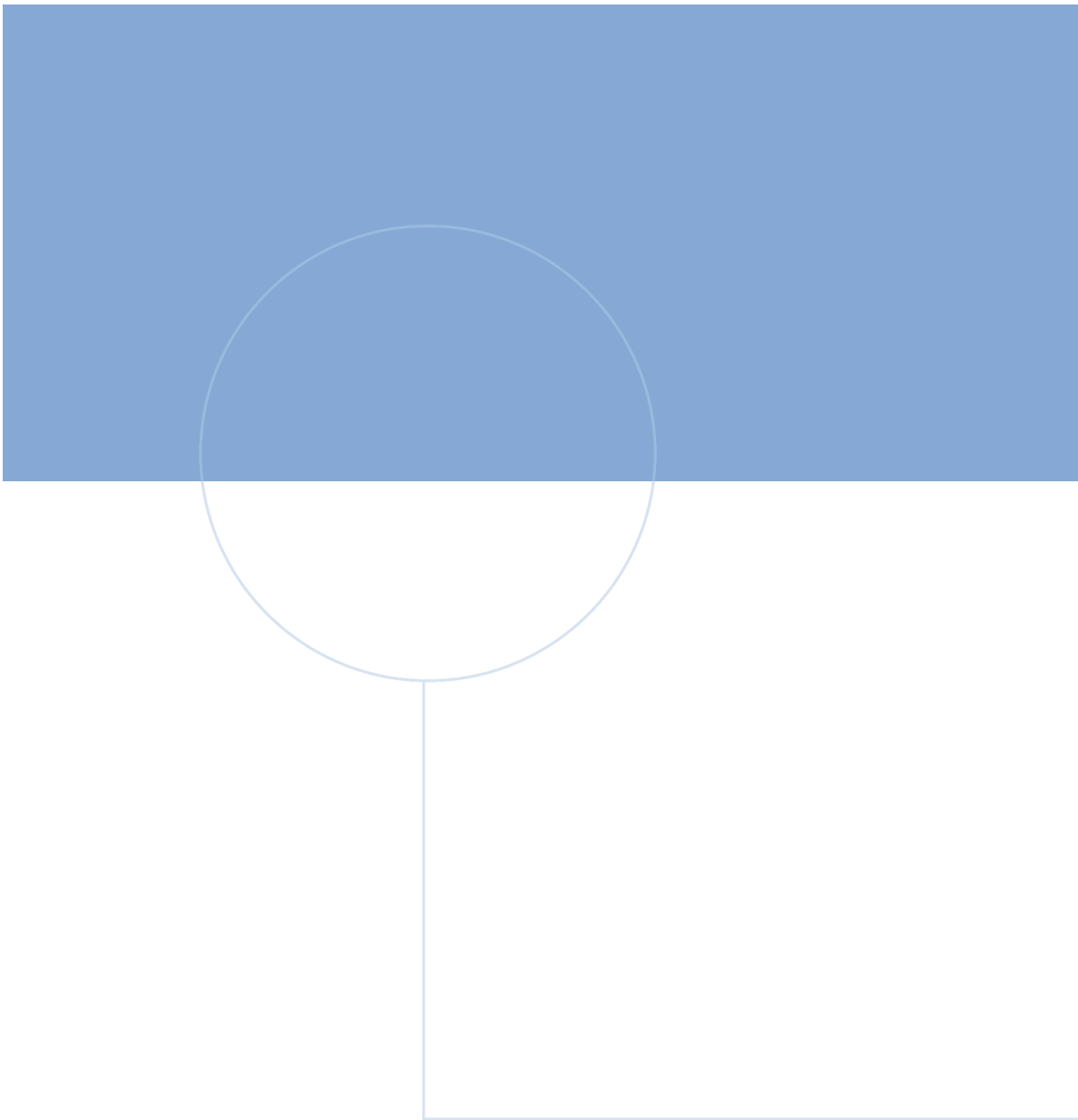


Figure 48: Uniform model in Experiment 3 tracking the test set using the hyperparameters from Table 12.



NTNU
Norwegian University of
Science and Technology

UFSC