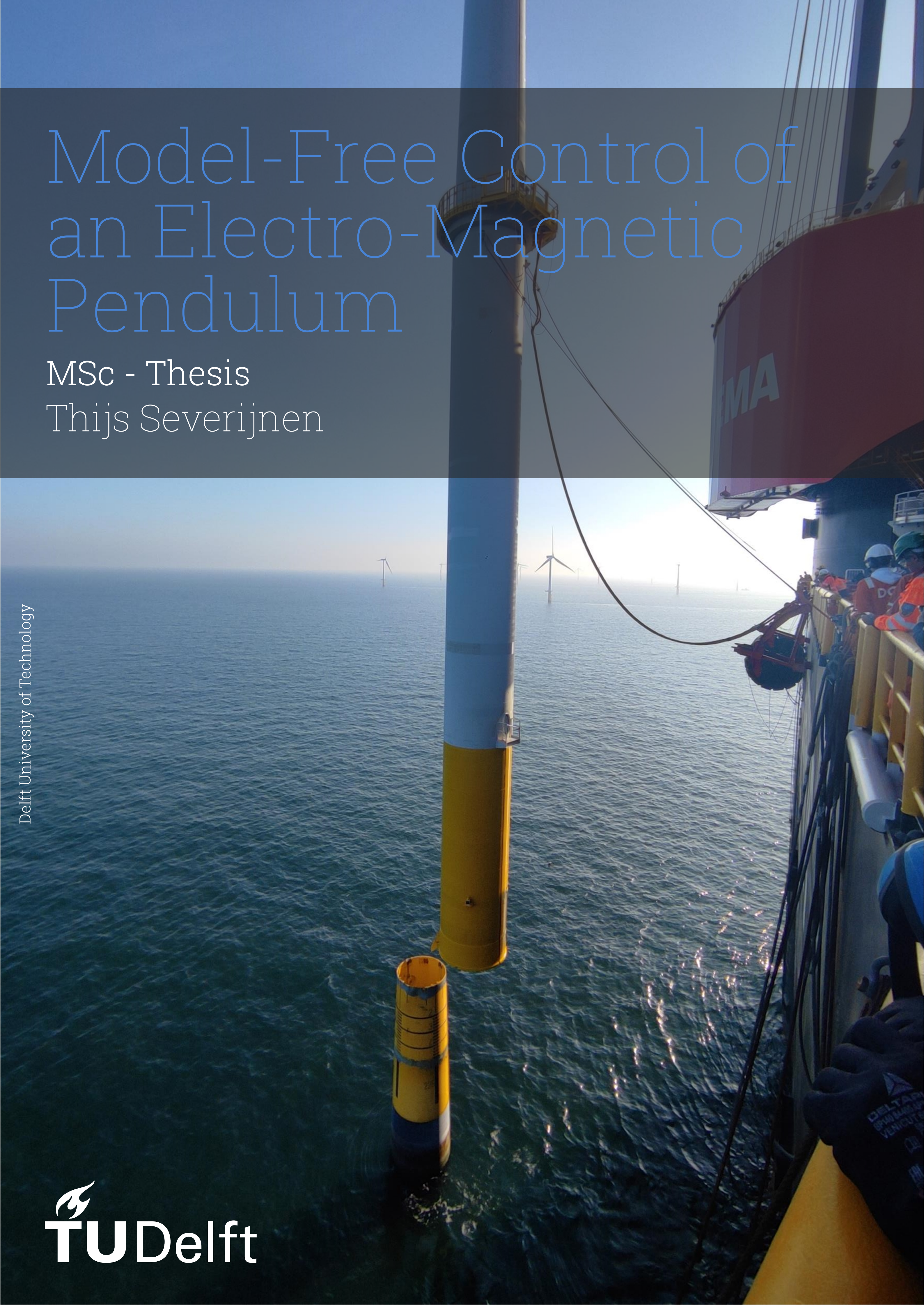


Model-Free Control of an Electro-Magnetic Pendulum

MSc - Thesis

Thijs Severijnen

Delft University of Technology

TU Delft

Model-Free Control of an Electro-Magnetic Pendulum

by

Thijs Severijnen

to obtain the degree of Master of Science
at the Delft University of Technology,
to be presented on the 15th of july 2024

Student number:	4954378	
Project duration:	September 15, 2023 – July 15, 2024	
Thesis committee:	Dr. ir. E. Lourens,	TU Delft, Chair
	Dr. Ir. P. Meijers,	TU Delft, Daily Supervisor
	Dr. Ir. J.O. Colomes Gene,	TU Delft
	Ir. P. Atzampou	TU Delft

Preface

Dear Reader,

In front of you is my master thesis on Model-free control of an electro-magnetic pendulum, this research is the culmination of six years of studying at the TU Delft. The subject of this thesis may initially not seem closely related to civil engineering, because it delves into neural networks and control theory. I must say this also presented quite a for me as well. However, this is precisely what I love about the MSc in Civil Engineering. The field is remarkably broad and far from being confined to static concrete calculations. Discovering new areas of research motivates me to keep learning and exploring.

Throughout my years of study and work in civil engineering, I have had the pleasure of diverse experiences in the field of civil engineering. Outside of the required curriculum for hydraulic structures that already encompasses a broad amount of topics. I got to further broaden my knowledge, through field measurements of lake waves in the Markerwadden during a week-long measurement campaig, by taking a variety of elective courses in computational hydraulics, computational dynamics, and even dredging, and through teaching structural mechanics to both bachelor's and master's students, occasionally in lecture halls with up to 80 students. Additionally, I got to spent four months as an intern site-superintendent, contributing to the construction of a 400-meter concrete viaduct using the Incremental Launching Method (ILM). This time taught me discipline and provided me with extensive insights into construction technology. Although, all these experiences might not seem directly related to this thesis and control theory or neural networks. They collectively contribute to my development towards a competent civil engineer and leave me very grateful.

I want to use this preface to also express gratitude to the people that helped me during this academic year. First of all my daily supervisor Peter, you were always available and went far beyond what is required from a supervisor. Without some key breakthroughs you offered, I could never have finished this research. I really enjoyed working together. Then of course all the members of my committee for their individual contributions. Eliz-Mari for keeping me on course and making me reflect on my own work. Panagiota for always being available when I needed help, and providing me with extensive feedback. Finally, Oriol Colomes Gene for agreeing to help asses my work.

I want to conclude with mentioning my friends, family, teammates and roommates, who supported me during all six years of my studies and helped me in numerous ways. Keeping me motivated for achieving my academic goals, but also for guiding me through my personal experiences and struggles.

Thijs Adriaan Constantijn Severijnen
Delft, July 2024

Abstract

This research contributes to a novel non-contact installation method for offshore wind turbines, utilizing electromagnetism for positional control. Floating wind installation requires efficient control methods for payload positioning. Current methods use tuggerlines that are physically attached and work through tensile attractive forces. In contrast, electromagnetic control offers a non-contact solution and allows for control through both attractive and repulsive forces.

This research explores the applicability of multiple data-driven control algorithms on a magnetically controlled pendulum system. These algorithms are evaluated for both positional control and motion attenuation under pivot-point excitation, using simulations of a numerical model of the electromagnetic pendulum. The model-free control algorithms considered in this study assumed that changes in the system's state are solely due to the control output. These methods aimed to optimize the control output based on either the state error or estimates of the state gradient with respect to the control output. However, this assumption was found to be valid only for systems dominated by the control response with limited dynamic behaviour over time, which was not the case for the electromagnetic pendulum system.

The study culminates in the development of a controller based on the online estimation of the magnetic interaction force. This controller, named Proportional-Derivative Force estimating neural network controller (PD-FeNN), is a neural-network based state-dependent PD-controller. The neural network is trained during an operational learning phase on estimations of the magnetic interaction force, which are derived using the model of a linear undamped pendulum. By incorporating the neural network, this approach eliminates the requirement of the previous state-of-the-art controller to manual model the magnetic interaction force based on experimental data.

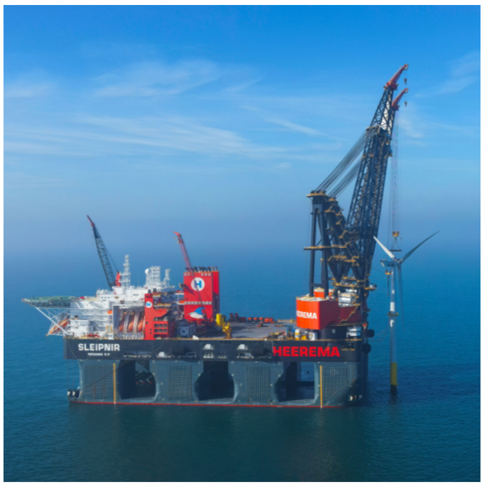
The PD-FeNN controller is tested on both numerical and physical models of the electromagnetically controlled pendulum. The results demonstrate efficient control across a wide range of excitation frequencies and amplitudes for both motion attenuation and positional control. As a successor to the modified PD controller, the PD-FeNN controller improves upon its predecessor by enabling positional control without requiring a model of the magnetic interaction. This advancement enhances the applicability of non-contact motion control for payloads in the offshore wind industry.

Contents

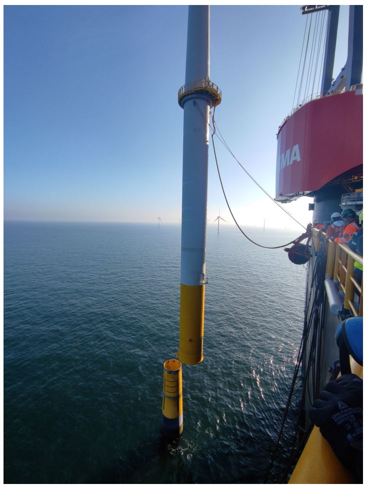
1 Introduction

1.1. Motivation for research

The growth in offshore wind projects is expected to accelerate in the upcoming years (IEA, 2022). This growth is caused by to the ongoing energy transition. Offshore wind development is seen as a vital part of this transition. The main advantages of offshore wind are the available space to build and the efficiency of offshore wind turbines (OWT). This efficiency is due to both the larger size of offshore wind turbines and the presence of stronger and more consistent winds offshore. The disadvantages of offshore wind are the expensive costs related to both construction and maintenance. Therefore, a lot of research is being done into improving installation methods of offshore wind turbines. Installation of offshore wind turbines often requires complicated heavy lifting operations. These lifting operations are only getting more complicated as installation is increasingly moving further offshore and towards floating turbines. This means semi-submersible vessels (SSV) will be required to install OWT's, instead of the current jack up vessels. These SSV's, however, encounter considerable movement of the crane due to wind and wave loading. This movement of the crane is undesired as it leads to uncontrolled movement of the suspended load. During mating of the wind turbine, connecting the wind turbine to its (floating) foundation (Jiang, 2021), this movement can cause set-down forces that damage the structural integrity of the OWT (Blom, 2022). To account for the ship movement and improve the capability and efficiency of these lifting operations, recent research has been done into stabilizing payloads using tugger lines and automated control algorithms, as performed in Figure 1.1b (Bron, 2022 and Ren et al., 2021).



(a) Floating Wind Turbine Installation (TU Delft, 2021)



(b) Tugger lines attachted to OWT (TU Delft, 2021)

Figure 1.1: Pictures from the FOX research project (TU Delft, 2021

Tugger lines, however, only work in tension and require the load to be connected to the ship. Therefore, a novel approach has been proposed. This approach uses magnetic interaction between the payload and an electromagnetic actuator to control the position of the payload (Meijers et al., 2023). This approach is capable of both attractive and repulsive forces, while also being a non-contact method. Recent research has been done by Atzampou et al. (2023)) and Meijers et al. (2023) into the behaviour and control of this system. This research has been focused on a schematised scale model of the lifting operation. This model is controlled through a modified PD controller with state-dependent gains to account for the non-linear magnetic force. The next step in this research is to control a more complicated system closer to the real life case. This increase in complexity leads to problems in the modelling of the dynamical system and the control response. The non-linearity of the electromagnet and unknown dynamics relating to environmental conditions make it difficult to model the entire system and find a model based solution.

This research is, therefore, about implementing a model-free controller for positional control of an electromagnetic pendulum. This means the payload will be controlled by an algorithm that has no knowledge of the dynamics. The control is purely based on the input and output of the dynamical system.

1.2. Report Outline

This report exists out of 10 chapters. The first chapters contains the problem definition and the methodology of this thesis. Chapter 2 introduces the current model description and controller of the scale model.

Chapter 3 dives into controlled dynamical systems and control theory. It introduces relevant concepts from control theory and categorizes control methods based on their use of data. Chapter 3 concludes by identifying promising control methods that are to be applied to the system of interest. Chapter 4 functions as a brief introduction to artificial neural networks and explains the required concepts relevant for understanding this research.

Chapters 5 to 8 form the main body of this thesis. The chapters describe the implementation of control methods applied to the system of interest. Chapter 8 is the most important chapter of the three, because it details the implementation and functioning of the proposed controller by this thesis. The chapters start of by introducing the assumptions, mathematical formula and block diagrams related to the control methods. This is followd by an implementation. The control method is applied first to the system of interest of the original reference paper to validate the implementation of the method. Second, the method is applied to the system of interest of this thesis and the results of (simulated) test cases are displayed. The chapters conclude by discussing the results and giving a conclusion on how well the control algorithm satisfied the requirements. Chapter 4 is about model-free adaptive control (MFAC). Chapter 5 details a neural network based PD controller (PD-NN). Chapter 6 implements a polynomial weighted output recurrent neural network PID controller (PID-PWORNN). Chapter 8 introduces a PD-Force estimating neural network controller (PD-FeNN). This is a PD controller with state-dependent gains calculated by a neural network trained on estimates of the magnetic interaction force.

Finally, Chapter 9 contains the conclusion of the research. It answers the research question, based on the results from the subquestions and presents the PD-FeNN controller as the final delivery of this research. Chapter 10 contains recommendations for further research, based on the findings of this thesis and the difficulties encountered during development of a model-free control algorithm for the system of interest.

1.3. Problem Definition

This section outlines the objective of the research, the associating research questions and the requirements for the model-free control algorithm.

1.3.1. Objective

The current state of the art controller requires an experimentally determined relation to function. If the system is changed this relation needs to be redetermined. This is unfavourable as it will require an operator, that is educated in the dynamics of the system, to perform experimental test. This makes the controller less general and thus less often applicable. Therefore, creating a controller that is more general and does not require experimentally determined relations or parameters is desired. The objective of this research is to achieve positional control of a suspended load by creating a model-free controller for an electromagnet actuator. By creating this controller, the research aims to further the knowledge on- and applicability of the novel method of positional payload control through electromagnetic actuators.

1.3.2. Research Questions

The research question help reach the objective of creating a model-free controller, and understand the behaviour and capabilities of such a controller. The following research questions are posed:

Main Research Question:
Which model-free control algorithms allow for positional control of a suspended load, while adapting control parameters based on measurement data, without requiring manual tuning?

The following subquestions are posed to aid in answering the main research question:

1. What is a model-free controller?
2. Which model-free controllers exist?
3. What are the most important limitations for model-free controllers?

1.3.3. Requirements proposed controller

This research does not only aim to find insights on model-free controllers, it also aims to present a model-free controller suited for the electromagnetic pendulum system. Therefore a set of requirements are formulated that this controller must satisfy. The following requirements for the new controller have been determined:

R1: The controller makes the payload follow a predefined reference trajectory with the allowance of a small error.
R2: The controller is model free .
R3: The controller adjusts its parameters based on measurement data.
R4: The controller does not require the manually tuning of parameters when system parameters are altered.

Furthermore, it is preferred that the controller runs on present hardware of the available scale model. To explore the behaviour of the proposed control further, the following subquestions will be answered to get more insights in the capabilities of the proposed controller:

4. What is the performance of the proposed controller?
5. How generalizable is this controller?

1.4. Methodology

The methodology of the thesis is based on 3 phases. First, a literature study will be performed that focuses on the field of control theory. This literature study culminates in a selection of promising control methods, that are expected to satisfy the requirements. Second, these promising methods will be implemented and applied to a numerical model of the system. The controller will be tested on 4 test cases stabilization after a disturbance, positional control, pivot-point motion attenuation, and positional control under pivot point excitation. Based on the performance during the numerical simulations a decision will be made on which control algorithm is the most promising. Third, this most promising controller will be implemented into the laboratory scale setup. After implementation, the performance is again assessed and possible required changes to the controller will be made. When the control performance is satisfactory on the scale model, the scale model will be slightly altered to asses the generalizability of the controller. This methodology is schematized in the flowchart given in Figure 1.2.

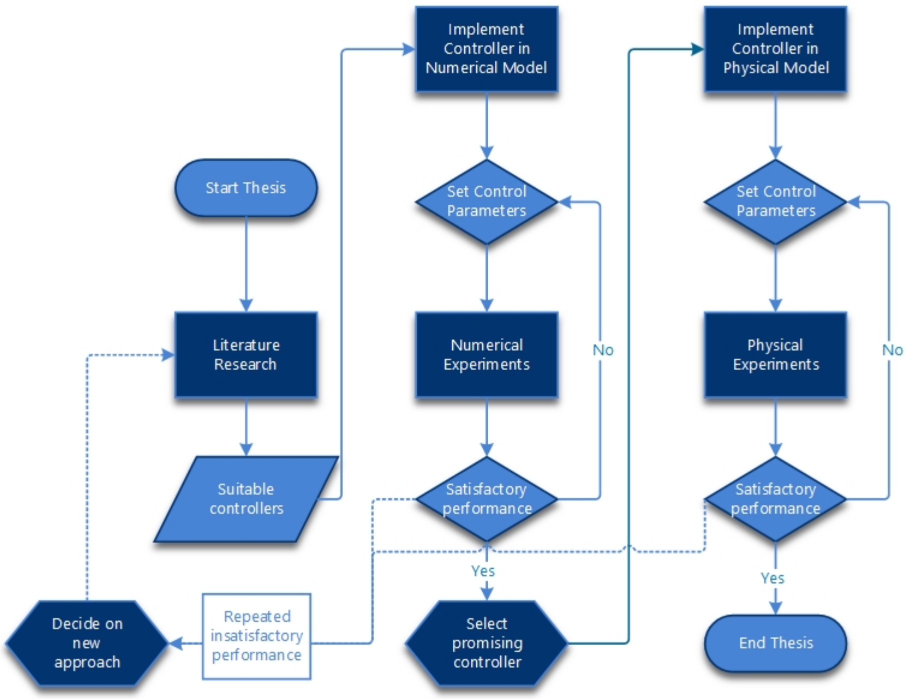


Figure 1.2: Flowchart methodology

1.4.1. Experimental setup

Both numerical and physical modelling will be used to examine the performance of the control algorithms. Numerical modelling is used, because it allows for quick assessment of the performance of the algorithms and implementation of the algorithms is relatively easy. The numerical models are not restricted by the hardware of the scale model and they can basically be done at any place and at any time. Physical experiments are still required to asses the real performance of the controller, because there are mismatches between the numerical model and the real system. The numerical model is both subjected to modelling errors as well as numerical errors, due to numeric integration. The aim of this research is to control the scale model, thus the controller must be applied to the scale model. The scale model is expected to be more difficult to control than the simulated system, because the states of the system are no longer available from calculation. Instead, the states of the system will need to be measured. Measuring introduces a measurement error and limits knowledge of the state to the measurement frequency.

2 System Description

This chapter details the properties of the scale model of the system. It explores the system modeling process and the design of the controller. Finally, it outlines the test cases used to evaluate the performance of the controllers.

2.1. Scale model

The scale model of the research from Atzampou et al. (2024) is used to physically model the system of interest, Figure 2.1.

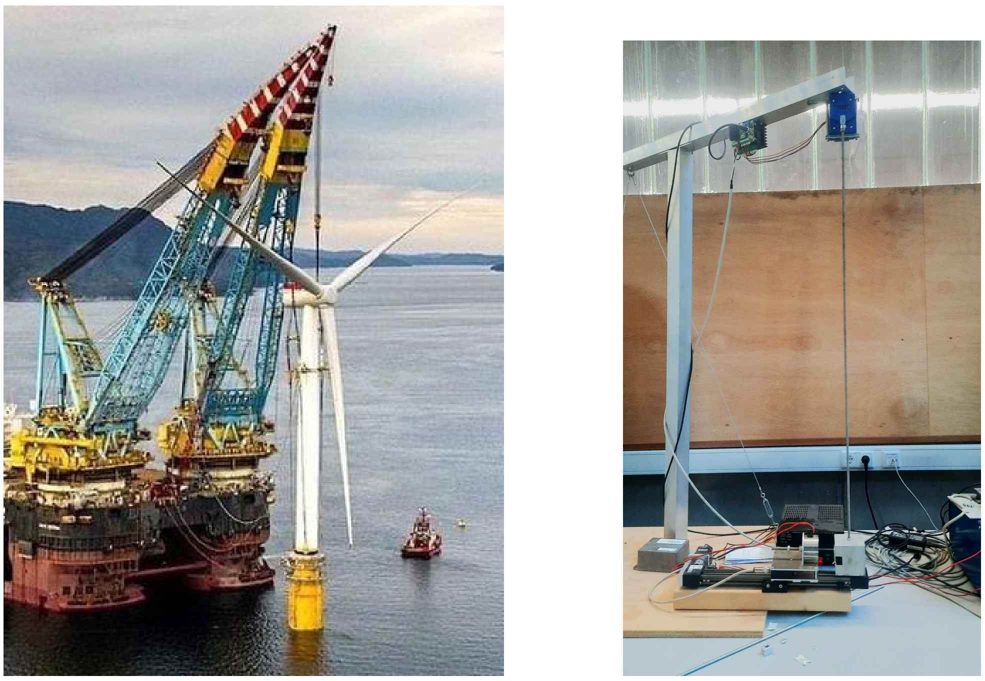

(a) System of interest (photo: Equinor, 2020) **(b)** Scale model

Figure 2.1: Physical modelling of the system of interest in a scale model

The scale model consists of a simple pendulum with a cubicle mass M connected to a rigid rod of total mass m and length ℓ rotating around a pivot point. The system is actuated by an electromagnet (Eclipse

Magnetics EM65-24V-DC) (EM) mounted a constant distance d apart from the cubicle mass on a belt positioning system. Across from the electromagnet, a permanent magnet (N52 Ø15x3 mm) (PM) is attached to the center of the cubicle mass. Such that the dipole moments of the magnets are aligned on the same axis. The system is externally excited through a horizontal prescribed motion of the pivot point h (Atzampou et al., 2023). An annotated picture of the setup can be found in Figure 2.2 and the dimensions of the relevant parts are given in Table 2.1.

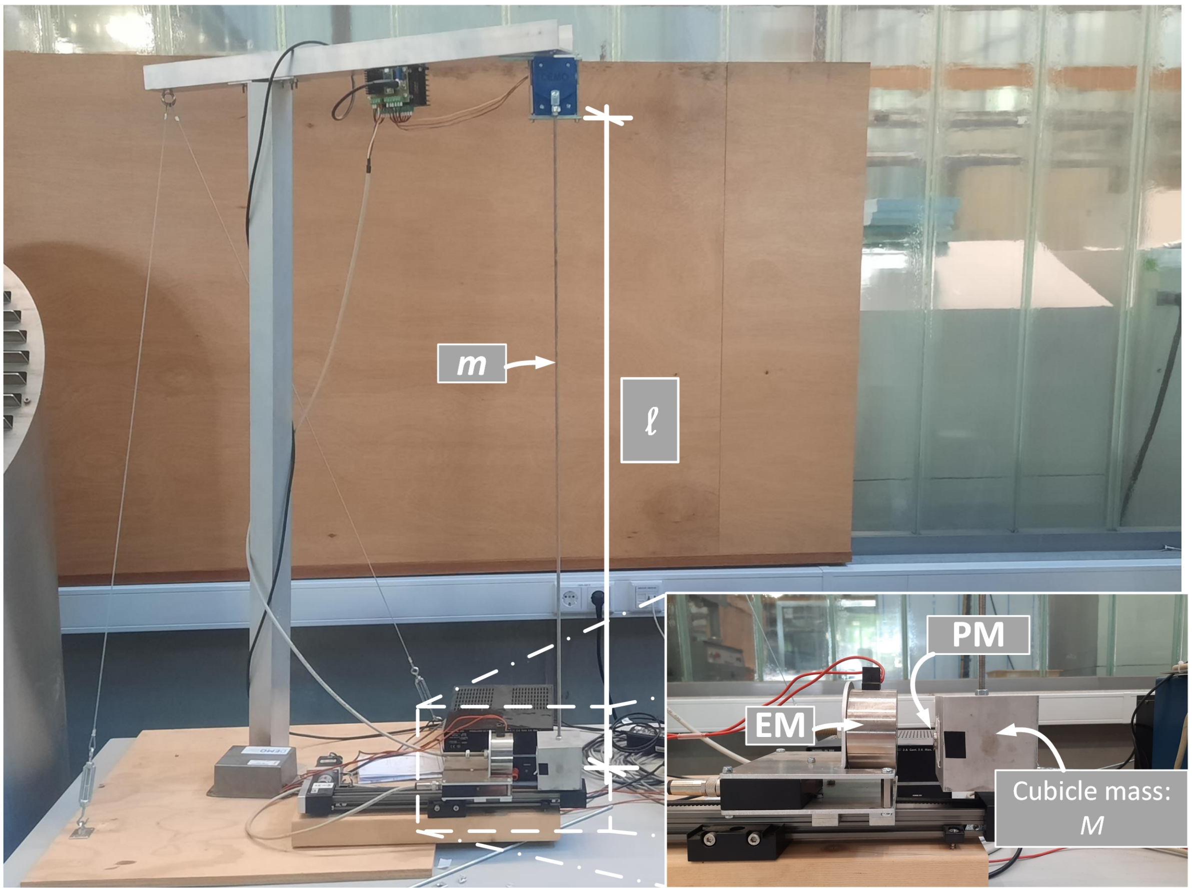


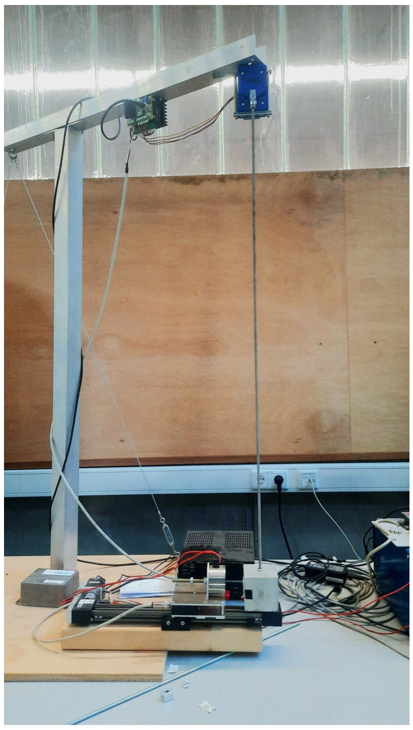
Figure 2.2: Scale model

ℓ [m]	M [kg]	m [kg]	Cube Size	PM [mm]	EM size [mm]
1.04	0.9382	0.1868	70 x 70 x 70	Diameter = 15	Diameter = 65
				Thickness = 3	Thickness = 35

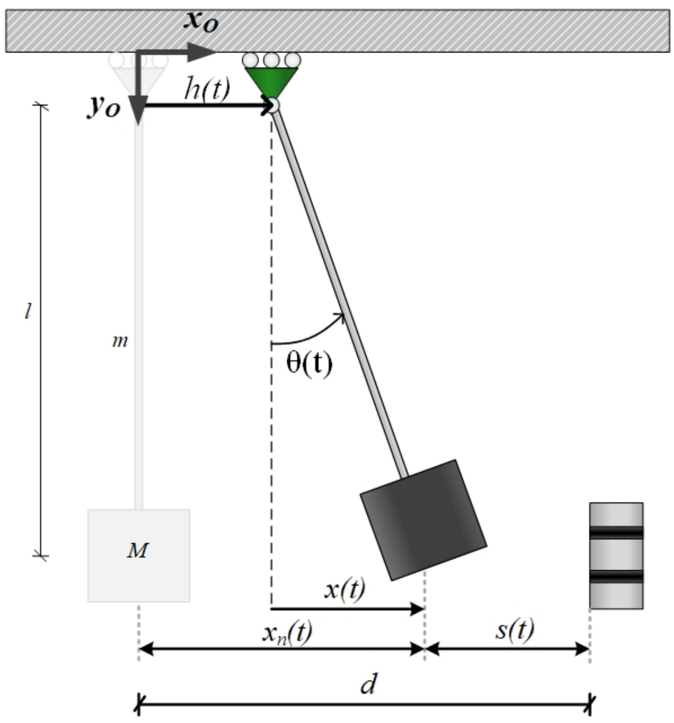
Table 2.1: Scale model dimensions Atzampou et al., 2024

2.2. Dynamical Model

One of the achievements of the research of Atzampou et al. (2024) was creating a dynamical representation of this scale model. The scale model was modelled as a 2D pendulum with 1 Degree of Freedom θ with a prescribed motion off the crane tip h. The rotation of the pendulum θ is related to the displacement of the bottom load through $x = l \cdot \sin(\theta)$, for small angles this approximates to $x \approx l\theta$. This linearization is valid when the motion of the load is small compared to the length of the wire ($l >> x$). For the scale model this is the case as $l \approx O(10^0)$ m and $x \approx O(10^{-2})$ m. The model is schematized in Figure 2.3.



(a) Scale model



(b) Dynamical Model

Figure 2.3: Modelling of mating operation

This schematization corresponds with the following dynamical model as can be derived using Euler-Lagrange (Atzampou et al., 2024):

$$\left(M + \frac{m}{3}\right)\ddot{x} + \left(M + \frac{m}{2}\right)\frac{g}{\ell}x = -\left(M + \frac{m}{2}\right)\ddot{h} + D + F_{magnet}, \tag{2.1}$$

in which dots indicate time derivatives, and:

x	[m]	displacement payload;
M	[kg]	mass aluminium block;
m	[kg]	mass steel rod;
ℓ	[m]	length steel rod;
h	[m]	displacement pivot point;
g	[m^2/s]	gravitational acceleration;
D	[N]	damping term;
F_{magnet}	[N]	force from electromagnet on the cubicle mass.

The terms D and F_{magnet} are still unknown. As these cannot simply be derived from the Euler-Lagrange equation.These terms must be estimated based on experimental data.

2.2.1. Damping estimation

Damping is the energy dissipation in the system, modelled as a force in the equation of motion. The dissipation in the scale model was found to be largely governed by the hinged connection point and could be modelled using the Coulomb's friction formula:

$$D = \frac{M_{fr}}{\ell} = -\mu\,\text{sgn}(\dot{x}), \tag{2.2}$$

where M_{fr} is the frictional moment at the hinge and μ is the kinetic friction. Coulomb friction is non-linear because of the signum function and states that the magnitude of friction is independent of the velocity, this leads to a linear decay in amplitude of the oscillations of the pendulum. The kinetic friction is dependent on many factors. Factors such as environmental conditions, lubrication, frequency and amplitude of the oscillations. This makes the simulation of friction with high accuracy a tedious task with experiments that involve little reproducibility. However, for the scale model a constant value for μ already produced satisfactory results. The constant was fitted to $\mu = 0.05 kgm/s^2$ based on the test data. The comparison between the experimental resulted and the model prediction is given in Figure 2.4 (Take note that Atzampou et al. (2024) used u to denote x).

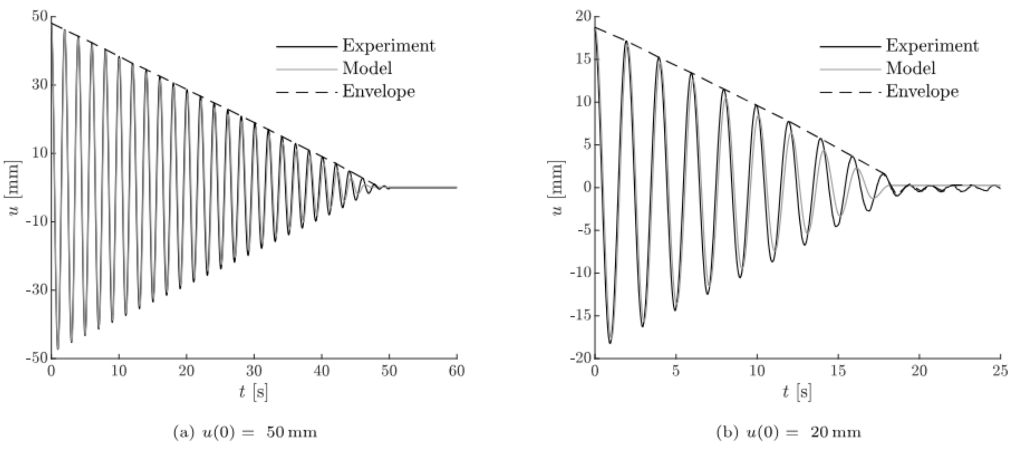


Figure 2.4: Damping calibration for two initial distances (Atzampou et al., 2024)

2.2.2. Modelling electromagnetic Force

The electromagnetic force exciting the system is caused by repulsion or attraction between the fixed polarity and strength permanent magnet (PM) and the varying polarity and strength electromagnet (EM) (Atzampou et al., 2024). By assuming the PM can be modelled as a magnetic dipole the force can be calculated as the gradient of the dot product of the dipole and the magnetic field created by the EM (Griffiths, 2017):

$$\boldsymbol{F} = \nabla(\boldsymbol{m} \cdot \boldsymbol{B}), \tag{2.3}$$

in which ∇ is the gradient operator, $\boldsymbol{m}$ is the dipole moment of the PM, and $\boldsymbol{B}$ is the magnetic field from the EM at the location of PM. The system is a SDOF system, thus only the gradient with respect to x_0 is relevant.

$$F = \frac{d}{dx_o}(m_{x_o} B_{x_o}) \tag{2.4}$$

It is considered difficult to model B_{x_0}, due to the geometry of the EM (Atzampou et al., 2024). Therefore, this term is based on experimental data. A line is fitted through collected data points from 3 experiments with varying voltage, Figure 2.5.

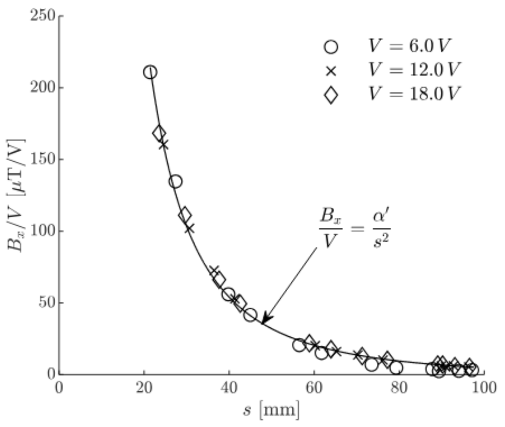


Figure 2.5: Determination magnetic field dependency on distance (Atzampou et al., 2024)

This fit gives an inverse squared relation for the magnetic field as a function of the distance s. Using this relation in the equation and absorbing all constants in the fitted parameter gives:

$$F = \frac{d}{dx_o}\left(m_{x_o}\frac{\alpha' I}{s^2}\right) = 2m_{x_o}\frac{\alpha' I}{s^3} = \frac{\alpha J}{s^3} = \frac{\alpha J}{d-x-h}, \tag{2.5}$$

in which, I is the current in the coil of the electromagnet and J the voltage as calculated by $J = RI$.

Because the electromagnet is an inductor, its magnetic field resist changes in voltage. Thus ,the voltage in the electromagnet will not always equal the control output voltage $u(t)$. Therefore, Atzampou et al. (2024) modelled the circuit as an resistor-inductor model:

$$\dot{J}(t) = \frac{1}{\tau}(u(t) - J(t)) \tag{2.6}$$

in which dots indicate time derivatives and τ is a time delay parameter in secodns s. By measuring the electromagnetic field while oscillating the control output u (Figure 2.6). This parameter can be fitted and for the scale model was found to be τ = 0.040 s.

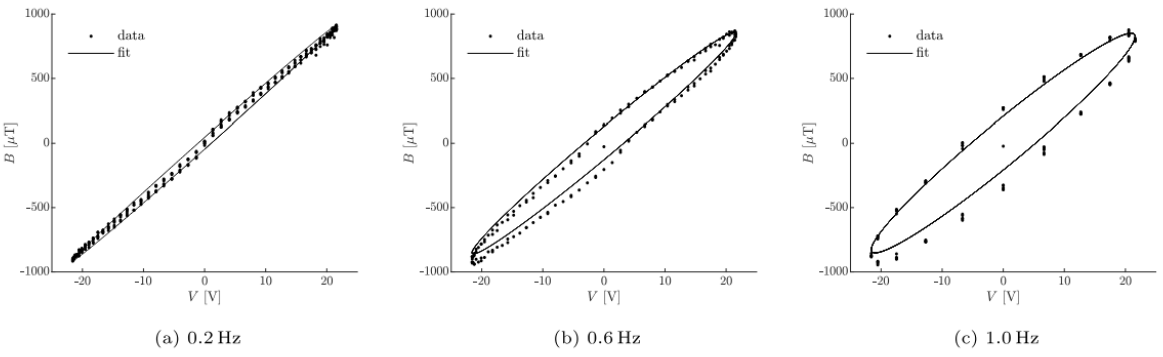


Figure 2.6: Determination of time constant τ (Atzampou et al., 2024)

The only term that is left to determine is the constant α. This parameter is estimated by measuring the static displacement of the pendulum under a constant voltage :

$$K_e x = \frac{\alpha J}{(d-x)^3} \longrightarrow \alpha \approx K_e x \cdot \frac{(d-x)^3}{J}. \tag{2.7}$$

For the scale model α was determined to be equal to $2.5810 \cdot 10^{-7}$ Nm^3/V. This value is scale model specific, a change in the electromagnet or permanent magnet, would require refitting of the parameter α.

2.2.3. Equations of Motion

Inserting the experimentally determined damping and magnetic interaction terms, as described above, into equation 2.1 completes the equations of motion. The full equations of motions from the dynamical model as found by Atzampou et al. (2024) are given below:

$$\left(M+\frac{m}{3}\right)\ddot{x}+\left(M+\frac{m}{2}\right)\frac{g}{\ell}x=-\left(M+\frac{m}{2}\right)\ddot{h}-\mu\,\mathrm{sgn}(\dot{x})+\frac{\alpha J}{s^3}, \tag{2.8a}$$

$$\dot{J}=\frac{1}{\tau}(u-J), \tag{2.8b}$$

in which dots indicate time derivatives, sgn stands for the signum function, and:

x	[m]	displacement pay-load;
M	[kg]	mass aluminium block;
m	[kg]	mass steel rod;
ℓ	[m]	length steel rod;
h	[m]	displacement pivot point;
g	[m^2/s]	gravitational acceleration;
μ	[kg m^2/s^2]	Coulomb friction constant;
α	[N m^3/V]	constant related to interaction magnetic force;
J	[V]	voltage in electromagnet;
s	[m]	distance between payload and electromagnet, defined as $s = d - h - x$;
τ	[s]	delay constant related to self inductance, experimentally determined;
u	[V]	control output voltage.

This model contains three parameters and a distance relationship, that were fitted based on collected experimental data from the scale model: μ, τ, α, and s^3. These values are only applicable for the scale model and are the motivation to seek a model-free solution.

To ease notation in the following sections, two quantities are introduced effective mass: $M_e = \left(M + \frac{m}{3}\right)$ and effective stiffness: $K_e = \left(M + \frac{m}{2}\right)\frac{g}{l}$. Using this notating the equations of motion simplify to:

$$M_e\ddot{x}+K_e x=-\left(M+\frac{m}{2}\right)\ddot{h}+\mu\,\mathrm{sgn}(\dot{x})+\frac{\alpha J}{s^3}, \tag{2.9a}$$

$$\dot{J}=\frac{1}{\tau}(u-J), \tag{2.9b}$$

2.3. State of the art Controller

Positional control of the system of interest is desired. The current state of the art controller for positional control was designed by Atzampou et al. (2024) in parallel with the modelling of the system. The controller was specifically designed for the scale model. The controller is a modified PD controller with state-dependent gains. This controller is based on the experimentally determined model for the electromagnetic interaction. The block diagram of the control algorithm is given in Figure 2.7 (Take not that Atzampou et al. (2024) used $c'(t)$ for control output and $u(t)$ for the displacement).

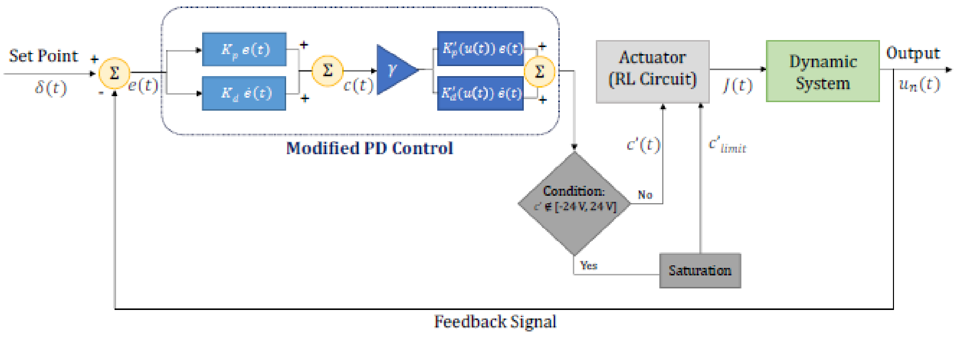


Figure 2.7: Block diagram modified PD control (Atzampou et al., 2023)

The control value can be calculated through:

$$u(t) = \left[\gamma K_p(\delta - x_n) + \gamma K_d(\dot{\delta} - \dot{x}_n)\right]_{u_{limit-}}^{u_{limit+}}, \tag{2.10}$$

in which γ is a state-dependent correction factor:

$$\gamma = s^3/\alpha = \frac{d - x - h}{\alpha}, \tag{2.11}$$

and K_p [-] and K_d [-] are constant gains and u_{limit+} and u_{limit-} are the saturation levels of the actuator. These limits exists due to physical limitations of the scale model. The maximum voltage the battery can provide is 24 Volts, thus u_{limit+} = 24 V and u_{limit-} = −24 V. Saturation is a form of non-linearity in the control output as can seen by plotting the output as a function of the error Figure 2.8. Saturation limits the capabilities of the controller, because it limits the amount of force that can be exerted on the dynamical system. A saturated controller means the control algorithm calculates a larger control output than can be provided thus the control value becomes limited by the the saturation limit.

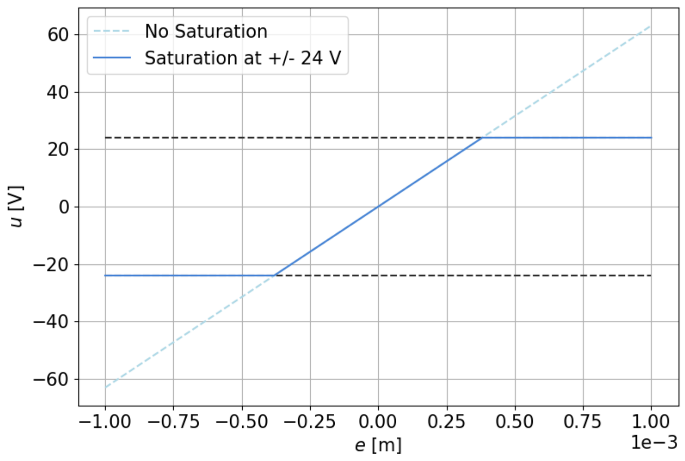
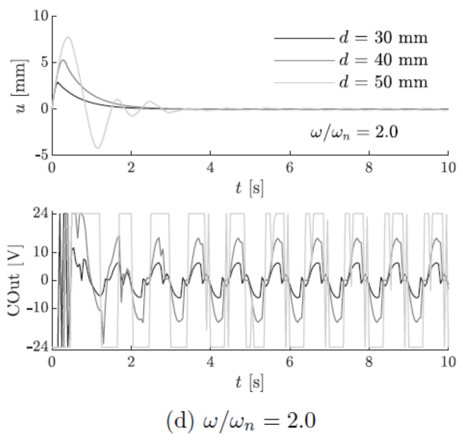
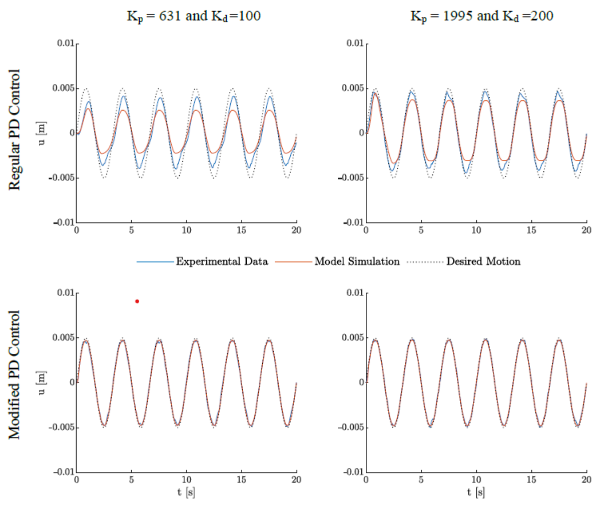


Figure 2.8: Non-linearity due to saturation

The current controller has experimentally been proven to be able to attenuate pivot point motion $h(t)$ (Figure 2.9a) and perform positional control in the absence of disturbances $\delta(t)$ (Figure 2.9b). Providing, the required voltage for control remains within the saturation limits of the controller. This controller has however the downside that it is setup specific. It depends on a correct determination of the magnetic interaction, through the use of the $\frac{\alpha}{s^3}$ relationship from Section 2.2.2. Determining this relationship is a difficult process and might not be possible for more complex dynamical systems.



(a) Pivot point motion attenuation



(b) Positional control

Figure 2.9: Capabilities of current state of the art controller (Atzampou et al., 2023)

2.3.1. Numerical modelling

The numerical model is based on the equations of motion determined by the research of Atzampou et al. (2024) and presented in equation 2.9b. These equations are transformed into non-linear space-state notation:

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t) = f(\boldsymbol{x}(t), u(t)) \Rightarrow \frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} x(t) \\ \dot{x}(t) \\ J(t) \end{bmatrix} = \begin{bmatrix} \dot{x}(t) \\ \frac{1}{M_e}\left(-K_e x(t) - \mu\,\mathrm{sgn}(\dot{x}(t)) - (M + \frac{m}{2})h(t) + \frac{\alpha}{s(t)^3}J(t)\right) \\ \frac{1}{\tau}\left(u(t) - J(t)\right) \end{bmatrix}, \tag{2.12}$$

in which $\boldsymbol{x}$ is the state vector $[x(t), \dot{x}(t), J(t)]$ and $f(\boldsymbol{x}(t), u(t))$ a non-linear function of both the state and the control output $u(t)$. For the numerical experiments, these equations are solved numerically, through the method of direct integration using a RK-4(5) scheme. This is implemented in Python with the use of the SciPy and NumPy packages. The parameters of the dynamical system are based on the values from the scale model. To asses the generalizability of the controller and its capabilities to adjust based on measurement data, the parameters can be modified to represent a change in the physical system.

Next to the full non-linear system, a discrete linear system is used. The discrete linear system is used for the implementation of control methods based on an iteration number k. As implementing these methods into an integration scheme using variable time stepping is considered difficult and only beneficial if the control algorithm has already been proven to control the linear system. The linear system is discretized using central differences and looks like:

$$x(k+1) = 2x(k) - x(k-1) + \frac{\Delta t^2}{M_e}\left(-K_e x(k) + a u(k) + F(k)\right), \tag{2.13}$$

in which:

$x(k)$	[m]	displacement of payload;
M_e	[kg]	equivalent mass;
K_e	[N/m]	equivalent stiffness;
Δt	[s]	time step;
a	[-]	linear control coefficient;
$F(k)$	[N]	external force;

more information on the derivation can be found in Appendix A.

2.3.2. Test cases

To asses and compare the control algorithms, four test cases are selected. As mentioned in Section 1.4, these are initial displacement stabilization, positional control, pivot-point motion attenuation, and positional control under pivot point excitation. These test cases are quantified based on the experiments of Atzampou et al. (2024). For test case 1, the initial displacement is set to 10 mm, the desired trajectory and the pivot-point motion are set to a constant zero. Test case 2, positional control, has zero initial conditions, a desired trajectory of 1.5 times the natural frequency with an amplitude of 4 mm, and no pivot-point motion. Test case 3, pivot-point motion attenuation, has zero initial conditions a desired trajectory of constant zero and an oscillation of the pivot-point motion in the natural frequency of the system with an amplitude of 5 mm. Test case 4 has zero initial conditions, a desired trajectory in the natural frequency with an amplitude of 5 mm and a pivot-point motion based on the Jonswap spectrum (Atzampou et al., 2024). The time series of this motion is given in Figure 2.10.

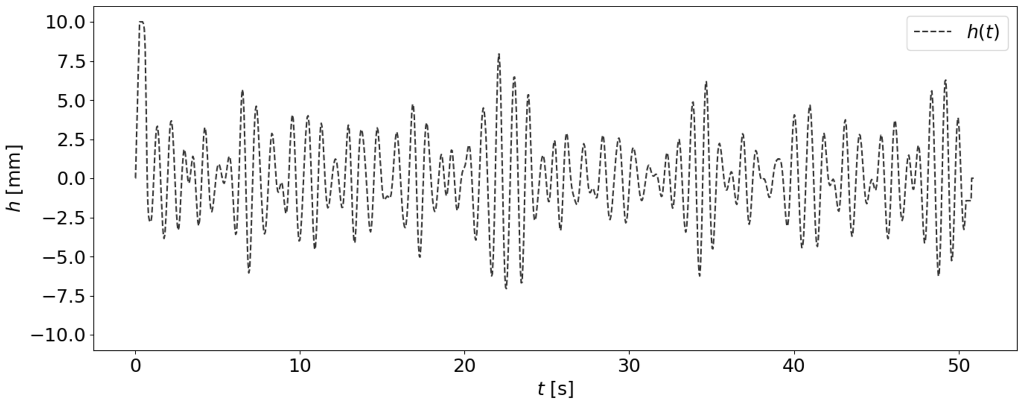


Figure 2.10: Pivot-point motion time series of test case 4

The amplitude of the desired trajectory are chosen such that the current state of the art controller is capable of positional control for the test case. Larger amplitudes are not feasible due to the saturation limits of the controller. Because the frequencies of the desired trajectory and pivot-point motion are a factor of the natural frequency, the natural frequency needs to be calculated. This can be done by taking the square root of the effective stiffness over the effective mass and gives:

$$\omega_n = \sqrt{\frac{K_e}{M_e}} \approx 3.12 \ \text{rad/s} \longrightarrow f_n \approx 0.50 \ \text{Hz}. \tag{2.14}$$

All test cases are executed with a constant distance $d = 40$ mm, between the electromagnet and the permanent magnet in resting position. The test cases are given in an overview in Table 2.2

	Test Case 1	Test Case 2	Test Case 3	Test Case 4
$x(0)$ [mm]	10	0	0	0
$\delta(t) : f$ [Hz]	0	0.75	0	0.50
$\delta(t) : A$ [mm]	0	4	0	5
$h(t) : f$ [Hz]	0	0	0.50	multiple
$h(t) : A$ [mm]	0	0	5	multiple

Table 2.2: Overview test cases

3
Control Theory

This chapter contains relevant knowledge about dynamical systems and the most important concepts from control theory. It discusses the mathematical representation of dynamical systems in space-state notation. It dives into different types of control algorithms grouping them based on their use of dynamical models and data. The data-driven models are then further split into indirect and direct controllers depending on the utilisation of system identification. This chapters concludes with presenting the most promising methods based on literature.

3.1. Controlled dynamical systems

To determine the applicability of control algorithms, it is important to investigate what type of dynamical system is aimed to be controlled, especially the presence of non linearity in the system.

Dynamical systems in control theory are often represented using state-space notation instead of equations of motion. The most basic system that can be controlled and also the one most found in literature is the linear time invariant system (LTI). This system can be written as:

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t) = \mathbf{A}\boldsymbol{x}(t) + \mathbf{B}\boldsymbol{u}(t), \tag{3.1a}$$

$$\boldsymbol{y}(t) = \mathbf{C}\boldsymbol{x}(t) + \mathbf{D}\boldsymbol{u}(t), \tag{3.1b}$$

in which $\boldsymbol{x}$ is the state vector describing the current state of the system, $\mathbf{A}$ is the system matrix this describes the development of the system into the future, $\mathbf{B}$ is the input matrix and describes how the control value influences the state of the system, $\boldsymbol{y}$ is the output vector and contains the measurements taken from the system, $\mathbf{C}$ is the observation matrix selecting which states are measured and the $\mathbf{D}$ matrix is the feedforward matrix describing how the control value influences the measurements. For the linear representation of the system of interest this looks like:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} x \\ \dot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \frac{-K_e}{M} & 0 \end{bmatrix}\begin{bmatrix} x \\ \dot{x}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ a \end{bmatrix} u(t), \tag{3.2a}$$

$$y = \begin{bmatrix} 1 \\ 0 \end{bmatrix}\begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u. \tag{3.2b}$$

These above representation use continuous time t. Control is, however, often executed at discrete times with a certain frequency. Therefore, many systems and control algorithms are notated in discrete time k and use a sequence of values to represent a signal instead of a continuous function.

For these purely linear systems (equation 3.2), the literature is rich and extensive. Many control algorithms exist and can be used for instance Linear quadratic regulatros (LQR) or Proportional-Integral-Derivative controllers (PID) (Åstrom and Murray, 2009).

It is also possible to have a non-linear dynamical system with a linear response to the control output. These systems can be controlled by linearizing around operational points and then applying many linear controllers or by a model-free approach that is able to approximate the non-linear dynamics (Fliess and Join, 2013a). These systems can be represented as follows:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x} = f(\mathbf{x}(t)) + \mathbf{B}\boldsymbol{u}, \tag{3.3}$$

in which $f(\boldsymbol{x}(t))$ is some arbitrary function of $\boldsymbol{x}$ and t and the response to control is still represented as a matrix vector product with $\mathbf{B}$ being the input matrix. The observation vector y for non-linear systems can be represented as some function $g(\cdot)$ of both the state and control output: $y(t) = g(x(t), u(t))$

Finally, also the response to the input can be non linear. In this case there is a non-linear system with non-linear response to controller output. This is represented as:

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x}(t) = f(\mathbf{x}(t), \mathbf{u}(t)) \tag{3.4}$$

in which $f(\boldsymbol{x}(t), \boldsymbol{u}(t))$ is an arbitrary function of both the state, as well as the controller output. The system of interest as described in Chapter 2 is a non-linear system with non-linear response to controller output. The non linearity in the system is only present in the damping and is relatively small. The response to control output as described in Chapter 2 is non linear , because of both state dependency as well as time delay. Note the scale model is only a simplified representation of the real application, which might include even more important parameters or features that are necessary to describe the real system. Currently these features are unknown. These could be dependencies on payload geometry or external wind forcing. To create a controller that is applicable irrespective of the environmental condition of which no current model is available, an approach is necessary that does not need a model description of these dynamics. This leads us to the field of Data Driven Control (DDC) where control is achieved through only the in and output data of the system.

3.2. Control algorithms

There are two main approaches to achieve control of a dynamical system, model based control and data driven control (Z.-S. Hou and Wang, 2013). These approaches are not mutually exclusive and can also be combined within a controller, a so called hybrid approach. Model-based control relies on full knowledge of the dynamics and initial state of the system. Using mathematics a(n) (optimal) control strategy is calculated beforehand and fed as input to the system. Model-based control is not suitable if relevant parts of the dynamics are unknown, because the calculated response will not align with reality. Model-based control can be done purely based on the initial state and the model of the system. This is called open-loop control, Figure 3.1. However, it can also use the output vector $\boldsymbol{y}$ in it's calculation making it a closed-loop controller, also called feedback controller.

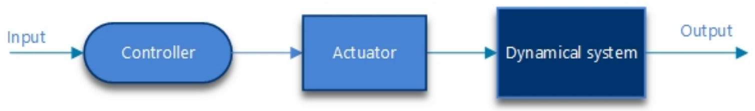


Figure 3.1: Block diagram open loop control

Data-driven control is control based only on the input and output data of the system and is able to overcome the gap of knowledge on the dynamics by using data from the system. Data-driven control is inherently closed loop as the output of the system is used as input for the controller, Figure 3.2.

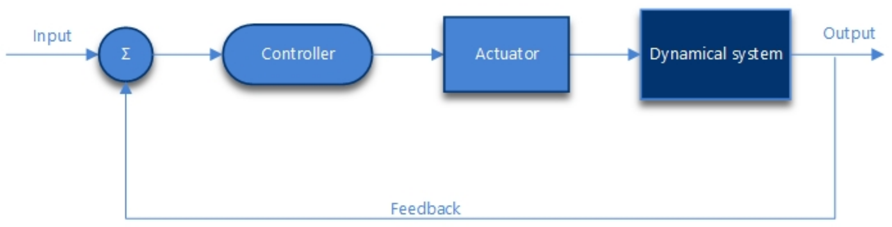


Figure 3.2: Block diagram closed loop control

3.2.1. Data-driven control (DDC)

Due to the aforementioned limits in model-based control and the increase in computing power and data collection over the past decades recent control research is focused on using system input and output data to design controllers. The advantage of using data is the ability to control systems with unknown dynamics and non-linear dynamics. By treating the system as a black box there is the danger to lose interpretability of the physics and the control. However, this might not be required for a functional controller. There exist two main approaches for data-driven control: indirect-control methods and direct-control methods. These methods can use two types of available data: offline data and online data.

Offline data is data collected before operation of the controller and online data is collected during operation of the controller. Offline approaches allow for the use of large amounts of data and computing time to be used for development and tuning of the controller. However, it does require this data of the system to be available before operation which might not always be the case. A downside of using only offline data is, that after this controller has been put in operation, the controller is not able to adjust anymore. Contrary an approach with **online data** allows for adjustments during operation and does not require any pre-available collected data. However, depending on the method this can mean that initial performance in poor, because of limited data available and slow convergence (Z.-S. Hou and Wang, 2013)

Indirect-control methods are methods were data is used to identify the dynamical system, so called system identification. Depending on which underlying technique is applied this system can be linear or non linear. The control law is then calculated based on this identified model. This method is visualised in Figure 3.3. This method can be based on both online or offline data. System Identification can be done with a variety of methods: DMDc, SINDy, Volterra serie, autoregressive models (ARX, ARMA, NARX and NARMAX) (Steve Brunton, 2018).

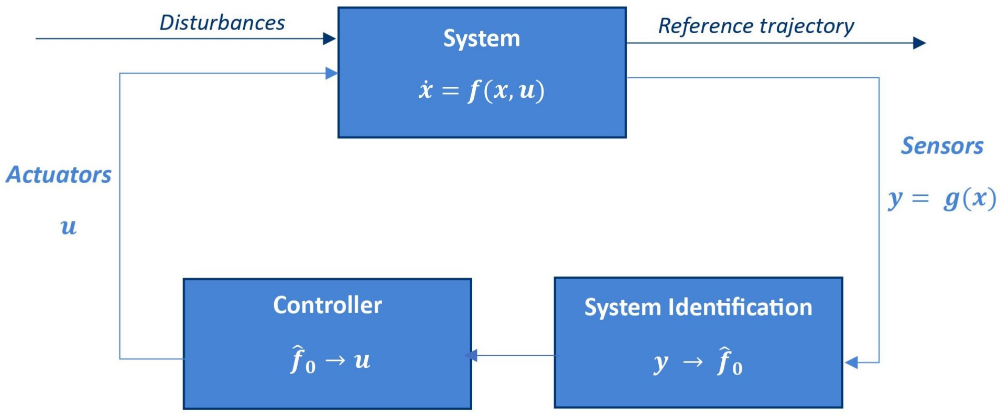


Figure 3.3: Indirect control block schematization

Direct-control methods use output data directly to compute the control values, as visualised in Figure 3.4. There is no effort made to model or describe the dynamical system. Thus, this class of controllers are considered model-free controllers. This also means it is difficult to interpret the result of the control and asses how optimal the control is. Some prominent examples are: VRFT, IFT, PID, MFAC, SPSA (Z.-S. Hou and Wang, 2013), DEePc (van Wingerden et al., 2022) and Q-Learning (Steve Brunton, 2022). Depending on the method these are suitable for both linear and non-linear systems respectively.

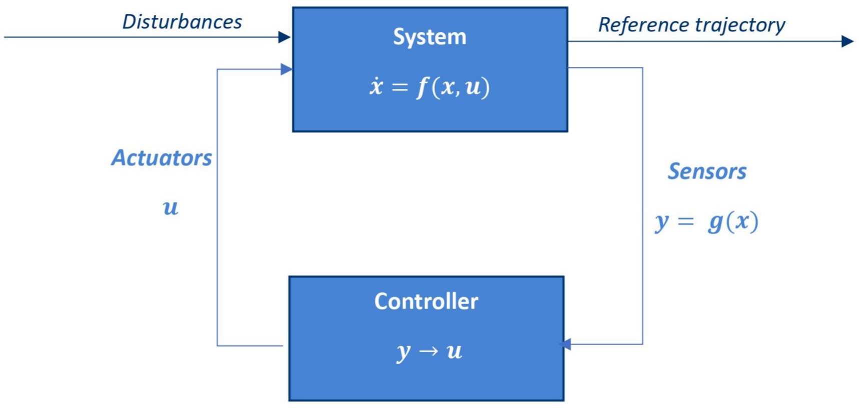


Figure 3.4: Direct control block schematization

3.2.2. Proportional-Integral-Derivative controller (PID)

To understand the further sections and to get an illustration of a direct data-driven controller, PID control is discussed in this section. PID control is a data-driven direct controller that controls the system based on a linear combination of the proportional error (P), integrated error (I) and derivative of the error (D). Error is defined as the difference between the desired and current state: $e = \delta - x$. It's the most prevalent type of control algorithm in industry (Z.-S. Hou and Wang, 2013) and can also be found in P, PI, PD form. The algorithm can be written in discrete form for discrete time systems as:

$$u(k) = K_p e(k) + K_I \sum_{i=0}^{k} e(i) + K_d \left(e(k) - e(k-1)\right), \tag{3.5}$$

and it can be written for continous time as:

$$u(t) = K_p e(t) + K_I \int_0^t e(\tau) d\tau + K_d \frac{\mathrm{d}e(t)}{\mathrm{d}t}, \tag{3.6}$$

in which $u(\cdot)$ the control output, $e(\cdot)$ the control error, and K_p, K_I, K_d the constant gains of the controller. An PID controller can also be written in incremental form, in which the starting point is the previous control output and this value is incremented based on the errors in the current time-step, for discrete time:

$$u(k) = u(k-1) + K_p(e(k) - e(k-1)) + K_I e(k) + K_d \left(e(k) - 2e(k-1) + e(k-2)\right). \tag{3.7}$$

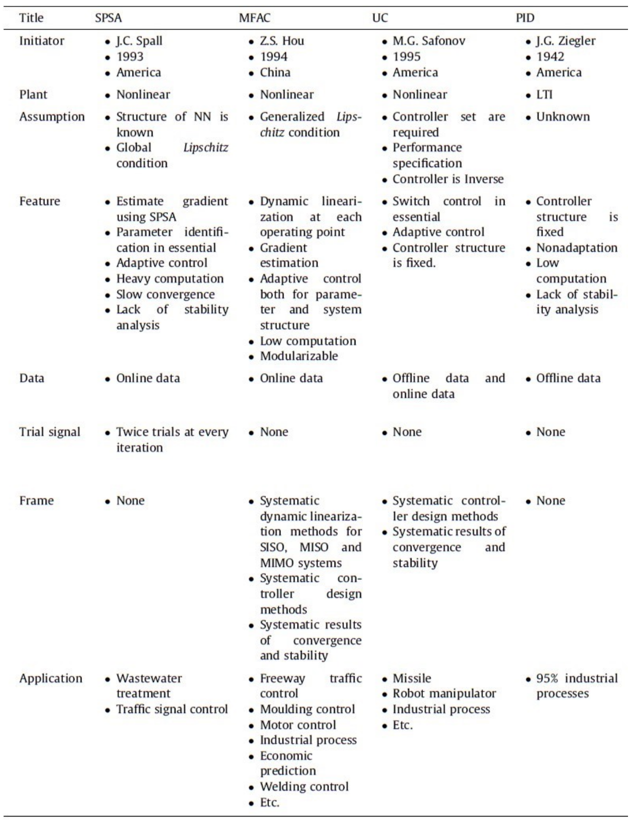
The gains of the PID controller (K_p, K_i, K_d) need to be manually tuned for the controller to work properly. Improper tuning can lead to overshooting and sustained errors. Tuning can be done either through an heuristic method, as the Ziegler-Nichols method or a more model based approach, as extremum seeking (N. J. Killingsworth and M. Krstic, 2006). As can be deduced from equation 3.5 the controller is a model-free controller using online data. The equation does not contain knowledge of the system model and it only requires data points from the last 3 available time steps.The conventional PID controller, however, is only suitable for linear time invariant systems, because the control output is linear dependent on the error.

3.3. Selection Controller method

An overview of common data-driven methods is given in Figure 3.5 and is taken from the research of Z.-S. Hou and Wang (2013). Applying the requirements for the controller to this overview in Figure 3.5 eliminates most data-driven control methods, based on the fact that the system of interest is non linear: the PID controller (PID), Iterative Feedback Tuning (IFT) and Virtual Reference Feedback Tuning (VRFT) can be eliminated. Incremental Learning Control is only suitable for a process that always has the same initial conditions, so is also not suitable. Unfalsified Control (UC) requires a set of available controllers while none are available. Lazy Learning (LL) is a form of system identification and must be combined with a controller (for instance PID, Pan et al., 2006 or minimum-variance and pole-placement or gradient based controller, Bontempi et al., 2010). SPSA has slow convergance and heavy computation and is therefore also not preferred. MFAC looks very promising based on this overview, however take note that this overview is made by Zhongsheng Hou whose PhD and further research has mainly focused on the design of MFAC (Scholar, n.d.). Taking this into account it cannot be underestimated how beneficiary the method can be as it does not require learning and is model free. Therefore, this method is seen as a suitable method to obtain the required control.

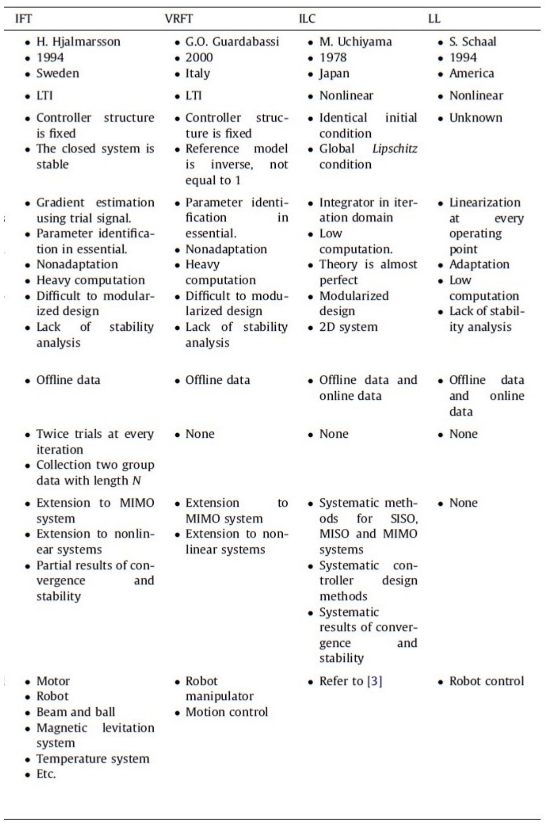
As given in Figure 3.5, and explained in Section 3.2.2, the classic PID controller is unable to control a non-linear system. However, as shown by the previous research of Atzampou et al. (2024), by introducing state-dependent gains, control could be achieved. Thus, PID control can be modified to also control non-linear systems. Some of these methods are *intelligent PID control* (Fliess and Join, 2013b), *Fuzzy PID* (Nguyen and Ahn, 2006), *Genetic algorithm based PID* (Zhang et al., 2009) and *NN based PID* (Dong et al., 2012). The advantages of using a PID controller are: simplicity, interpretability of gains, ability to compare with previous research and the available information due to it being the standard control algorithm for many industries (Z.-S. Hou and Wang, 2013). From all the improved PID controllers mentioned is a neural network (NN) based PID considered the most promising. This PID controller adjusts its gains based on the output of a neural network, thus creating in essence a modified PD controller similar to Atzampou et al. (2024). The other PID improvement methods are based on offline data and do not allow for online learning. These methods will thus not change parameter values, if the systems is disturbed or changed during operation.

In conclusion, this research focuses on finding a model-free controller and is thus limited to data-driven direct controllers. A considerable amount of these controllers are not suitable as they only apply to linear systems and the system of interest is identified to be non-linear. Other controllers have specific limitations, as for example slow converges or a requirement to have an available set of controllers. Therefore, only PID-NN (Chapter 6, 7, and 8) and MFAC (Chapter 5) are further explored.



Title	SPSA	MFAC	UC	PID
Initiator	• J.C. Spall • 1993 • America	• Z.S. Hou • 1994 • China	• M.G. Safonov • 1995 • America	• J.G. Ziegler • 1942 • America
Plant	• Nonlinear	• Nonlinear	• Nonlinear	• LTI
Assumption	• Structure of NN is known • Global *Lipschitz* condition	• Generalized *Lipschitz* condition	• Controller set are required • Performance specification • Controller is Inverse	• Unknown
Feature	• Estimate gradient using SPSA • Parameter identification in essential • Adaptive control • Heavy computation • Slow convergence • Lack of stability analysis	• Dynamic linearization at each operating point • Gradient estimation • Adaptive control both for parameter and system structure • Low computation • Modularizable	• Switch control in essential • Adaptive control • Controller structure is fixed.	• Controller structure is fixed • Nonadaptation • Low computation • Lack of stability analysis
Data	• Online data	• Online data	• Offline data and online data	• Offline data
Trial signal	• Twice trials at every iteration	• None	• None	• None
Frame	• None	• Systematic dynamic linearization methods for SISO, MISO and MIMO systems • Systematic controller design methods • Systematic results of convergence and stability	• Systematic controller design methods • Systematic results of convergence and stability	• None
Application	• Wastewater treatment • Traffic signal control	• Freeway traffic control • Moulding control • Motor control • Industrial process • Economic prediction • Welding control • Etc.	• Missile • Robot manipulator • Industrial process • Etc.	• 95% industrial processes

(a) Overview part 1



IFT	VRFT	ILC	LL
• H. Hjalmarsson • 1994 • Sweden	• G.O. Guardabassi • 2000 • Italy	• M. Uchiyama • 1978 • Japan	• S. Schaal • 1994 • America
• LTI	• LTI	• Nonlinear	• Nonlinear
• Controller structure is fixed • The closed system is stable	• Controller structure is fixed • Reference model is inverse, not equal to 1	• Identical initial condition • Global *Lipschitz* condition	• Unknown
• Gradient estimation using trial signal. • Parameter identification in essential. • Nonadaptation • Heavy computation • Difficult to modularized design • Lack of stability analysis	• Parameter identification in essential. • Nonadaptation • Heavy computation • Difficult to modularized design • Lack of stability analysis	• Integrator in iteration domain • Low computation. • Theory is almost perfect • Modularized design • 2D system	• Linearization at every operating point • Adaptation • Low computation • Lack of stability analysis
• Offline data	• Offline data	• Offline data and online data	• Offline data and online data
• Twice trials at every iteration • Collection two group data with length *N*	• None	• None	• None
• Extension to MIMO system • Extension to nonlinear systems • Partial results of convergence and stability	• Extension to MIMO system • Extension to nonlinear systems	• Systematic methods for SISO, MISO and MIMO systems • Systematic controller design methods • Systematic results of convergence and stability	• None
• Motor • Robot • Beam and ball • Magnetic levitation system • Temperature system • Etc.	• Robot manipulator • Motion control	• Refer to [3]	• Robot control

(b) Overview part 2

Figure 3.5: Overview of DDC methods (Z.-S. Hou and Wang, 2013)

4 Artifical Neural Networks

This chapter contains information on artificial neural networks. Artifical Neural Networks are a form of artificial intelligence which were initially inspired by the human brain (Roberts, n.d.). Knowledge of neural networks is required to understand the implementation of the PD-NN controller. Neural networks are used for classification, prediction, clustering and association. Classification is concerned with labelling (visual) data. Prediction or regression/function approximation is about finding a relationship between input variables and predicting some output variable. This chapter constrains itself to feedforward neural networks used for prediction as these are the most relevant for this research. Feedforward references the flow of information in the network. For a feedforward network, the information travels unidirectional and goes form the input layer towards the output layer. This chapter is structured as follows: structure and notation of neural networks, activation functions, network initialization, cost functions and finally optimization and the backpropagation algorithm.

4.1. Structure

Neural networks are computational models composed of interconnected nodes, or neurons, organized into layers. These layers are an input layer, one or multiple hidden layers and an output layer (Figure 4.1). Neurons can be connected in a multitude of ways most common is a dense connection in which each node of a layer is connected with each node in the next layer, another notable layer is the convolution layer, which is primarily used for visual input data. A general structure of a densely connected feedforward neural network is given in Figure 4.1.

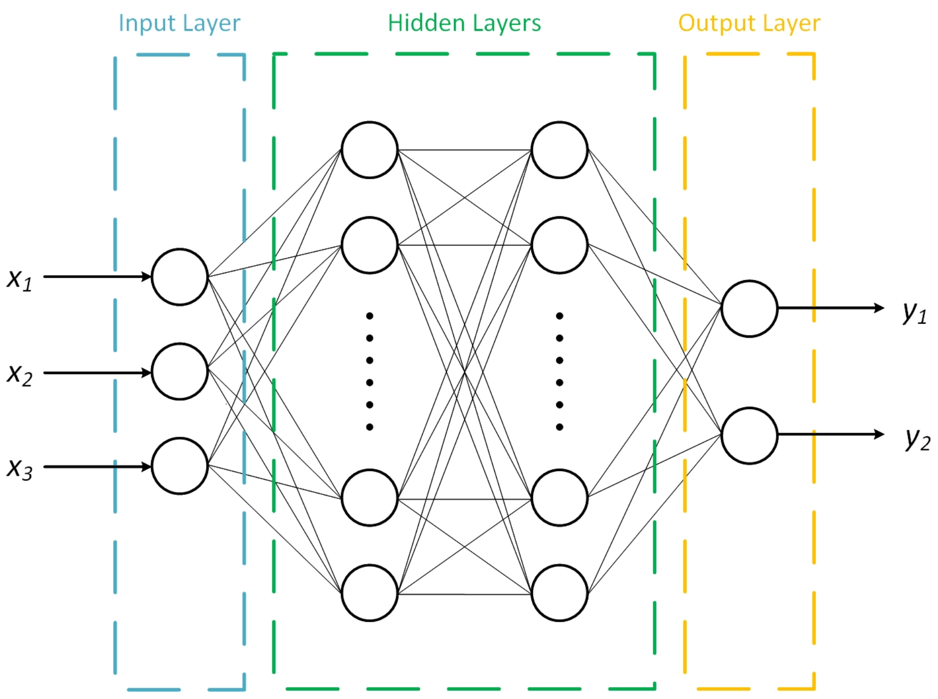


Figure 4.1: Structure of feedforward neural network

Each link between nodes has a weight and each node has a bias. To perform a feedforward operation and calculate the value of a node in the next layer the following steps are taken. The value of all inputs nodes are multiplied with the weight of their respective link to the node being calculated. The bias of the node is added and all values are added together. This value is called the weighted input, the final nodal value is calculated by feeding the weighted input through an activation function. Activation functions are further explained in the next section.

To be able to use mathematical formulas to describe the network some notation needs to be introduced. The value of a neuron is notated with the symbol a, the weight of a link with w, the bias of a node with b and the activation function is denoted as $\sigma(\cdot)$. Indices work as follows: w^l_{jk} means the weight connecting the k^{th} neuron from the previous layer to the j^{th} neuron in the l^{th} layer.

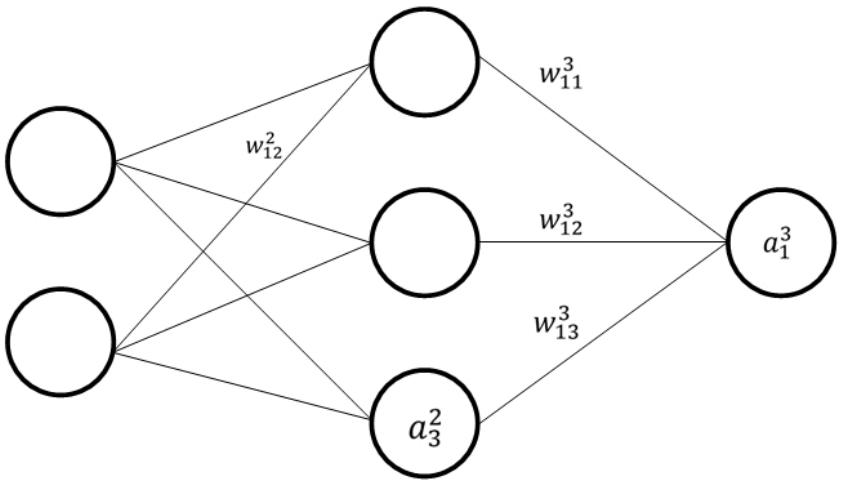


Figure 4.2: Example notation neural network

Calculating the output node from the example network of Figure 4.2 gives:

$$a_1^3 = \sigma(w_{11}^3 \cdot a_1^2 + b_1^3 + w_{12}^3 \cdot a_2^2 + b_1^3 + w_{13}^3 \cdot a_3^2 + b_1^3), \tag{4.1}$$

calculating any weight can be represented as follows:

$$a_j^l = \sigma(\sum_k w_{jk}^l \cdot a_j^l + b_j^l). \tag{4.2}$$

Instead of representing the calculation of nodal values individually, the notation can also be done in matrix vector form. This shortens notation and implementing the network using matrices and vectors can also greatly increases performance, as it allows for the use of optimized algorithms (Stack Exchange, 2016). If you collect all the weights between two layers in a matrix w^l, the nodal values in a vector a^l and biases in a vector b^l. The formula for calculating all values in a layer becomes one calculation:

$$a^l = \sigma(w^l a^{l-1} + b^l). \tag{4.3}$$

4.2. Activation Functions

The activation function is the function wrapped around the weighted input of a node. The choice of this function is important as it determines the range of the output, speed of convergences and the overall performance . The simplest form is a linear activation function:

$$\sigma(x) = x, \tag{4.4}$$

a linear activation functions means the nodal value is equal to the weighted input. If a linear activation function is applied, the model can only approximate linear functions and the neural network becomes a linear regression model. Through the use of non-linear activation functions a neural network can approximate any continuous function (deep-mind, 2023). Some of the most common activation functions are: Sigmoid, Tanh, ReLU, and Leaky ReLU.

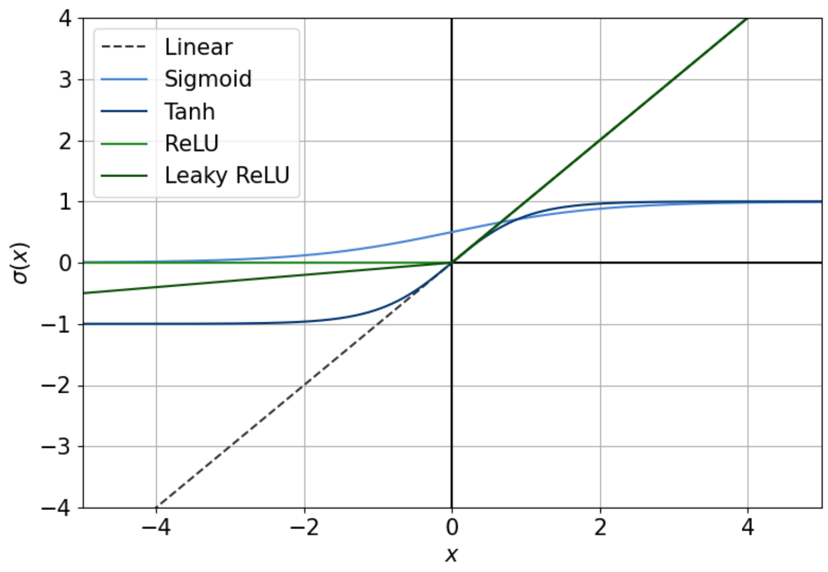


Figure 4.3: Overview activation functions

Sigmoid,

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \tag{4.5}$$

outputs values between 0 and 1 and is primarily used for binary classification problems. The disadvantages of this activation function are the time consuming cost of computing exponents and the large sections of the function with a small derivative making gradient descent very slow for large positive or negative numbers. This phenomena is called saturation. The function is also not zero centered which can reduce the efficiency of the weight update (Brodtman, 2021).

Hyperbolic tangent (Tanh),

$$\sigma(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \tag{4.6}$$

the hyperbolic tangent outputs values between -1 and 1. This allows for both positive and negative outputs and makes the function zero centered, which is better for the weight update. The function still suffers from saturation for large positive and negative inputs and also requires calculation of the exponent function (Brodtman, 2021).

Rectified Linear Unit (ReLU),

$$f(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases} = \max(0, x). \tag{4.7}$$

The ReLU activation function is relatively recent and has relevant advantages compared to the activation functions discussed so far. It is computationally faster, as it contains no exponentials. It does not suffer from gradient saturation, making it more robust to gradient vansihing. The function can introduce sparsity into the model making it less complex and memory intensive. Sparsity means not every layer is fully conenected, this is caused by neurons turning *off* if the input is smaller than 0. This is also a downside of ReLU, since the removal of links can be undesired. If this occurs, the neuron is considered a dead neuron. ReLU is not zero-centered and is sensitive to initialization (Brodtman, 2021).

Leaky ReLU,

$$f(x) = \begin{cases} \alpha \cdot x & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}, \tag{4.8}$$

in which:
α coefficient determining the slope of the negative part.
Leaky ReLU is very similar to ReLU. The difference lies in having aslope for negative input values instead of being zero. This means no dead neurons occur. The output has a range between (-∞, ∞) making Leaky ReLU almost zero centered (Brodtman, 2021).

4.3. Initialization

The values for the weights and bias need an initial value as starting point. Setting the initial value is important, because the initial values can influence the outcome of the model, and determine if convergence is even possible. Historically this has been done by initializing the weights to random floating point numbers, between [-0.3, 0.3] or [0, 1] or [-1, 1]. This tends to work quite well (Brownlee, 2021). It is important to avoid setting all weights to zeros. This can lead to the model being stuck, because the derivatives will also be zero and the weights will not be updated. If two nodes have the same activation function setting them to the same initial weight must be avoided, because if the learning process is deterministic, this will always lead to them being updated the exact same way (Brownlee, 2022).

Over the past decade better initialization functions have been developed such as *Normalized Xavier weight* and *He weight initialization* (Brownlee, 2021). Normalized Xavier weight initialization is based on the number of inputs and output neurons of a layer. It is calculated by a random number from a uniform distribution (U) with bounds based on the number of in- and output neurons:

$$w = U\left[-\frac{\sqrt{6}}{\sqrt{n+m}}, \frac{\sqrt{6}}{\sqrt{n+m}}\right], \tag{4.9}$$

in which n is the number of input neurons and m the number of output neurons. Using the same approach He et al. (2015) got a different result for layers using the ReLU function, because ReLU is not symmetric around 0. This alternative initialization for ReLU layers is called: *He initialization*. This initialization uses a random number from a Gaussian / Normal distribution (G) with a mean equal to 0 and a standard deviation of $\sqrt{\frac{2}{n}}$ with n being the number of input neurons.

$$w = G\left[0, \sqrt{\frac{2}{n}}\right] \tag{4.10}$$

4.4. Cost Function

To asses the performance of a neural network and to allow for learning of the network, a metric for the performance is required. This metric is called the cost function or loss function. The distinction between these two terms is that a loss function is defined on a single data point, while a cost function is defined over an entire dataset (Rajan, 2020). The cost function compares the predicted values by the network with the actual real values. The larger the result of the cost function the worse the network did, the closer to zero the better. Common types of cost functions for regression problems are:

- Mean Absolute Error (MEA)
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)

Mean Absolute Error,
this is the summation of the absolute difference between the real value and the predicted value:

$$C = \frac{1}{n}\sum_{i}^{n}|y_i - \hat{y}_i|, \tag{4.11}$$

the advantage of this cost function is that outliers are not heavily penalized. The disadvantage is that the derivative is not defined at 0. This means optimization algorithms using derivatives do not work using this cost function.

Mean Squared Error,

$$C = \frac{1}{n}\sum_{i}^{n}(y_i - \hat{y}_i)^2, \tag{4.12}$$

this cost function heavily penalizes large mistakes and is, thus, really sensitive to outliers. Caution is required, when applying this cost function, if you expect outliers based on some measurement errors in your input or actual values.

Root Mean Squared Error,

$$C = \sqrt{\frac{1}{n}\sum_{i}^{n}(y_i - \hat{y}_i)^2}, \tag{4.13}$$

this function is very similar to MSE, however it is slightly less sensitive to outliers, because of the square root decreasing the effect of outliers (Rajan, 2020).

4.5. Optimizing the Network

As briefly introduced in the previous section, The aim of a neural network is to minimize the cost function:

$$\min_{w,b} C(\cdot). \tag{4.14}$$

The goal is to find the weights and bias that lead to the minimum cost. To achieve this an optimization algorithm is used. A multitude of optimization algorithms exist, this section outlines the basic gradient descent algorithms, and the more advanced Adam optimizer and RMSprop.

Gradient descent (GD) is the most basic algorithm for optimization. It is based on the derivatives/gradients of the cost function with respect to each weight and bias for the entire dataset. By repeatedly updating the weights and biases into the direction of the gradient, a minimum will be found eventually. This is visualized for a single weight in Figure 4.4. As the gradient decreases the step size decreases together with the decrease in the cost function.

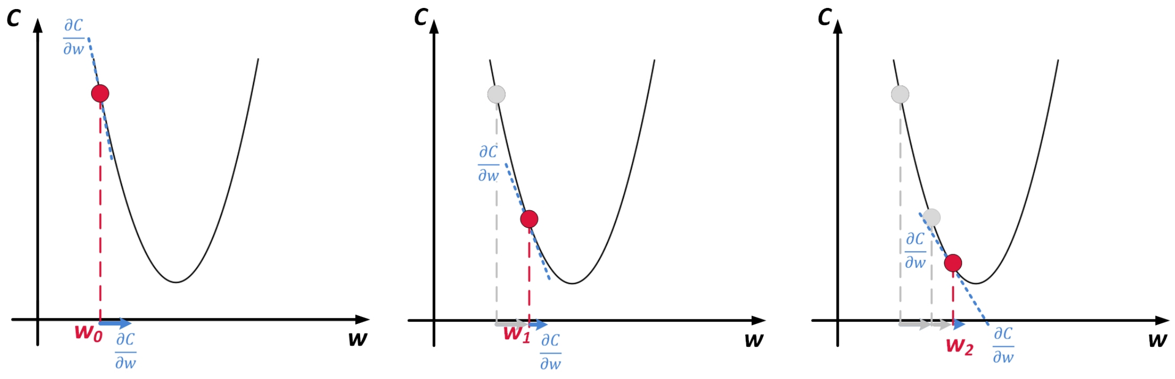

C
$\frac{\partial C}{\partial w}$
w_0
$\frac{\partial C}{\partial w}$
w
C
$\frac{\partial C}{\partial w}$
w_1
$\frac{\partial C}{\partial w}$
w
C
$\frac{\partial C}{\partial w}$
w_2
$\frac{\partial C}{\partial w}$
w

Figure 4.4: Gradient descent visualization

Mathematically written this gives:

$$\theta_t = \theta_{t-1} - h\nabla_\theta C, \tag{4.15}$$

in which:

θ	trainable variable that needs to be optimized;
h	learning rate;
C	cost function that needs to be minimized.

The advantages of this algorithm is the ease of understanding, computing, and implementing. However, in this basic form it has quite some disadvantages. The algorithm can get stuck on a local minimum, a lot of memory is required to calculate gradients over the entire dataset, and convergence can take very long if the dataset is very large (Doshi, 2020).

Stochastic gradient descent (SGD) tackles the problem of large datasets requiring a lot of time for convergence and a lot of memory. With this algorithm the trainable parameters are updated for each data point in the dataset:

$$\theta_t = \theta_{t-1} - h\nabla_\theta C(x_i). \tag{4.16}$$

Downsides of stochastic gradient descent are a high variance in model parameters, and the algorithm will keep updating even upon achieving a global minimum. To avoid this issue, and achieve the same result as gradien descent a decaying learning rate should be used (Doshi, 2020).

Mini-batched gradient descent is right in the middle between GD and SGD. The trainable parameters are updated per mini batch,a group of data points of certain size typically 32. This allows it to have the best properties of both algorithms.

However, gradient descent still has some inherent disadvantages, that cannot be solved with batch size only. The algorithm can still get stuck at local minima and it can not differentiate its learning rate per parameter. Even tough, some parameters might require different learning rates. Furthermore the learning rate needs to be tuned correctly, otherwise convergence might occur very slowly or not occur at all (Doshi, 2020).

Root mean square propagation (RMSprop) improves on mini-batched learning by dealing with the vanishing and exploding gradient problem. The vanishing and exploding gradient problem references the tendency of the gradient to either explode or vanish when propagating through the network. RMSprop applies a moving average of squared gradients to normalize the gradients. This balances out the step size of the algorithm. If the gradient is large the step size is reduced and if the gradient is small the step size is increased. In other words the learning rate varies for each variable to ensure roughly equal step sizes (Sanghvirajit, 2024):

$$E[g^2]_t = \beta \cdot E[g^2]_{t-1} + (1-\beta) \cdot (\nabla_\theta C)^2, \tag{4.17a}$$

$$\theta_t = \theta_{t-1} - \frac{h}{\sqrt{E[g^2]_t}} \nabla_\theta C, \tag{4.17b}$$

in which:

β	moving average parameter (default ≈ 0.9);
$E[g^2]$	squared moving average of gradient.

Adaptive moment estimation (Adam) has been made to combine the advantages of the RMSprop (online non-stationary settings) and the AdaGrad method (sparse gradients) (Kingma and Ba, 2014). Adam computes adaptive learning rates from the first and second moment of the gradients:

$$g_t = \nabla_\theta C, \tag{4.18a}$$

$$m_t = \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t, \tag{4.18b}$$

$$v_t = \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2, \tag{4.18c}$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \tag{4.18d}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}, \tag{4.18e}$$

$$\theta_t = \theta_{t-1} - \frac{h}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t, \tag{4.18f}$$

(4.18g)

in which:

β_1	first moment parameter (default: 0.9);
β_2	second moment parameter (default: 0.999);
$\hat{m}_t$	first moment (mean);
$\hat{v}_t$	second moment (uncentered variance);
ϵ	parameter (default: 10^{-8}).

4.6. Backpropagation

All algorithms given in the previous section are based on the gradient of the cost function with respect to the trainable variables. Backpropagation is an algorithm that allows the efficient computation of these gradients. Backpropagation makes two assumptions about the used cost function.

Assumption 1: The cost function can be written as an average.

Assumption 2: The cost function can be written as a function of the outputs from the neural network.

All cost functions introduced in section 4.4 comply with the first assumption and this is thus a valid assumption. Assumption 2 is specifically for control applications often not satisfied. This is further discussed in Section 6.3.

Two extra definitions are required to shorten notation and help explain back propagation: First z^l, the notation for weighted input: $z^l \equiv w^l a^{l-1} + b^l$. Second δ^l, the error of layer l, $\Lambda^l \equiv \frac{\partial C}{\partial z^l}$. Backpropagation is based on four fundamental equations (Nielsen, 2015). From the definition of the error, the first equation can be derived using the chain rule. The error in the output layer is:

$$\Lambda_j^L = \frac{\partial C}{\partial a_j^l} \sigma'(z_j^L). \tag{4.19}$$

This equation can be rewritten in matrix vector form using the elementwise multiplication operator $\odot$:

$$\Lambda^L = \nabla_a C \odot \sigma'(z^L). \tag{4.20}$$

The second equation is for expressing the error in a layer in terms of the next layer:

$$\Lambda^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l), \tag{4.21}$$

by multiplying the error of the next layer by the transpose of the weight matrix, the errors are moved *backwards* through the network. Applying this equation for each layer, the error can be backpropagated all the way back to the first hidden layer. The next step is to relate the calculated errors to the gradients of the weights and biases. The third equation is for the gradient of the bias, and is straightforward:

$$\frac{\partial C}{\partial b_j^l} = \Lambda_j^l, \tag{4.22}$$

the error of the node is exactly equal to the rate of change of the bias. Calculating the gradient of the weight is similar but requires multiplying the error by the nodal value of the previous layer. Equation four is:

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \Lambda_j^l. \tag{4.23}$$

To summarize the algorithm, first of equation 4.20 is used to calculate the error in the output layer. This error is then backpropagated through the network by repeated use of 4.21. Next, these errors are related to the derivative of the bias by equation 4.22 and to the derivative of the weight by equation 4.23. After computation of all the gradients an optimization algorithm from for instance Section 4.5 can be applied to improve the neural network.

5 Model-free adaptive control (MFAC)

In this chapter, Model-free adaptive control (MFAC) is discussed. MFAC is selected as a promising model-free controller, that might be able to satisfy the requirements and control the system of interest. Model-free adaptive control is a direct data-driven control scheme, developed by Z. Hou and Jin (2008). The scheme is able to control non-linear systems with one input, purely based on input and output data of the system. The controller adapts based on recent system state and input changes. The structure of this chapter is as follows: first the assumptions related to the controller are introduced; second the mathematical formula and implementation are presented, then the test cases are simulated, and finally the chapter concludes with discussion of the results and conclusions.

5.1. MFAC Assumptions

MFAC is founded on three assumption about the system that is being controlled. These assumptions lead to one important theorem. The three assumptions are:

Assumption 1: The system is observable and controllable, meaning that for some bounded expected system output signal $\delta(k+1)$ there exists a bounded control input signal in time instant k, that will make the output $x(k+1)$ equal to this reference value.

Assumption 2: The partial derivative with respect to control output $u(k)$ is continuous.

Assumption 3: The system is generalized Lipschitz. That means it satisfies the following condition: $|\Delta x(k+1)| \leq b||\Delta U(k)||$ for any k, where b is a positive constant, $\Delta U(k) = [\Delta u(k), ..., \Delta u(k-L+1)]^T$ and L is a positive integer.

These assumptions lead to Theorem 1 (Z. Hou and Jin, 2008):

Theorem 1 *For the nonlinear system satisfying the assumptions. There must exist a $\Phi(k)$, called pseudo-partial-derivative (PPD) vector when $||\Delta U(k)|| \neq 0$,*

$$\Delta x(k+1) = \Phi^T(k)\Delta U(k), \tag{5.1}$$

where $\Phi(k) = [\phi_1(k), ..., \phi_L(k)]^T$ and $||\Phi(k)|| \leq b$.

5.2. Implementation

To calculate the best control output the following cost function must be optimized:

$$R\big(u(k)\big) = \Big(x(k+1) - \delta(k+1)\Big)^2 + \lambda \cdot \Big(u(k) - u(k-1)\Big)^2, \tag{5.2}$$

where λ [-] is a weighting constant. The cost function can be understood as a squared error function with an additional cost for rapid changes in control output values. The parameter λ determines this cost related to rapid changes in control output. Equation 5.1 can be rewritten in terms of the next time step as follows:

$$x(k+1) = x(k) + \Phi^T(k)\Delta U(k), \tag{5.3}$$

substituting this equation into the cost function, setting the cost function to equal zero and differentiating with respect to the control output $u(k)$ gives the control law:

$$\frac{\partial R}{\partial u} = 0, \tag{5.4}$$

$$u(k) = u(k-1) + \rho_k \phi_1(k) \frac{\big(\delta(k+1) - x(k)\big) - \sum_{i=2}^{L} \phi_i(k)\Delta u(k-i+1)}{\lambda + \phi_1^2(k)}, \tag{5.5}$$

the introduced ρ is a step-size constant series, to make the algorithm more general applicable (Z. Hou and Jin, 2008), which usually can be set as $\rho_k \in (0,2)$.

Next to the control output $u(k)$, also the PPD $\Phi(k)$ needs to be determined. It is assumed that the unknown parameter vector PPD is a slowly time-varying vector, that can be estimated using conventional time-varying parameter estimation algorithms (Z. Hou and Jin, 2008). These are for example, modified projection algorithm (Z. Hou and Huang, 1997), the least-square algorithm with a time-varying forgetting factor (Goodwin and Sin, 1984) or the leakage recursive least-square algorithm (Tan et al., 2001). For this implementation, the modified projection algorithm is used:

$$\hat{\Phi}(k+1) = \hat{\Phi}(k) + \eta_k \Delta U(k) \frac{\Delta x(k+1) - \Delta U^T(k)\hat{\Phi}(k)}{\mu + ||\Delta U(k)||^2}, \tag{5.6}$$

which is a result of optimizing the equation (Zhongsheng Hou and Wenhu Huang, 1997):

$$R\big(\Phi(k+1)\big) = \Big(\delta(k+1) - x(k) - \Phi(k+1)\Delta U(k)\Big)^2 + \mu \cdot \Big(\Phi(k+1) - \hat{\Phi}(k)\Big)^2, \tag{5.7}$$

in this expression $\hat{\Phi}(k)$ is the approximation for the real $\Phi(k)$. The μ parameter is added to function both as an avoidance of dividing by zero and as a penalty for rapid change of the PPD. The introduced η is similar to ρ, a step-size constant series, to make the algorithm more general applicable, which usually can be set as $\eta_k \in (0,2)$.

The entire control algorithm is presented below:

$$u(k) = u(k-1) + \rho_k \phi_1(k) \frac{(\delta(k+1) - x(k)) - \sum_{i=2}^{L} \phi_i(k)\Delta u(k-i+1)}{\lambda + \phi_1^2(k)}, \tag{5.8a}$$

$$\hat{\Phi}(k+1) = \hat{\Phi}(k) + \eta_k \Delta U(k) \frac{\Delta x(k+1) - \Delta U^T(k)\hat{\Phi}(k)}{\mu + ||\Delta U(k)||^2}, \tag{5.8b}$$

$$\hat{\phi}_1(k+1) = \hat{\phi}_1(1), \qquad \text{if } |\hat{\phi}_1(k+1)| \leq \varepsilon, \tag{5.8c}$$

from these equations it can be deduced that MFAC is indeed a direct controller. It only uses input and output data ($x(\cdot)$, $\Delta U(k)$) to calculate the control output u. Model information is not required. The structure of MFAC is schematized below in Figure 5.1.

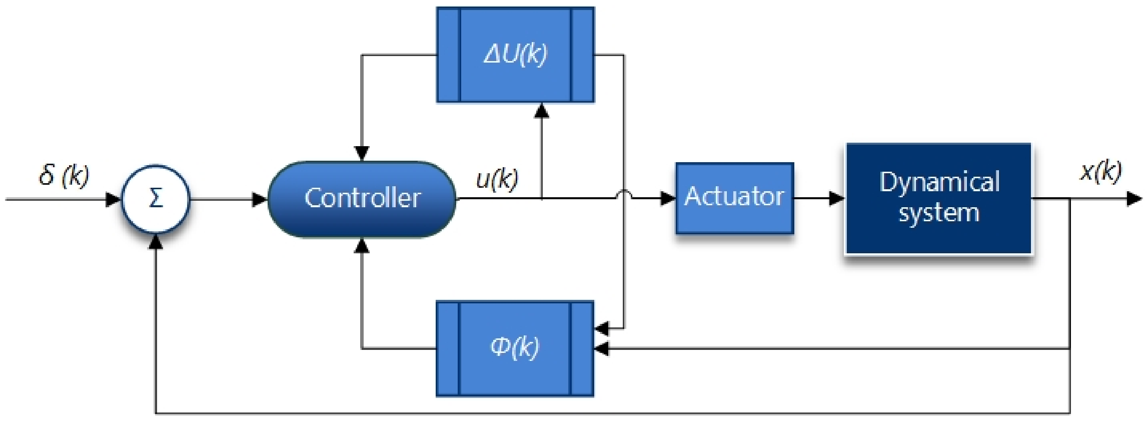


Figure 5.1: MFAC block scheme

5.3. Test case simulation

To asses the performance of MFAC, numerical simulations are done. Firstly, the implementation of the algorithm is verified, by letting the controller control the dynamical system from a reference paper of Yang et al. (2021). Secondly, the system of interest is simulated for test case 1.

This system used for verification is according to Yang et al. (2021) a classical single-input-single-output discrete non-linear system. The system is described through the following equation:

$$\begin{aligned} x(k+1) = & \frac{2.5x(k)x(k-1)}{1+x^2(k)+x^2(k-1)} + 1.2u(k) + 0.09u(k)u(k-1) + 1.6u(k-2) \\ & + 0.7\sin\Big(0.5\big(x(k)+x(k-1)\big)\Big) \cdot \cos\Big(0.5\big(x(k)+x(k-1)\big)\Big), \end{aligned} \tag{5.9}$$

the desired output is a sinusoïd:

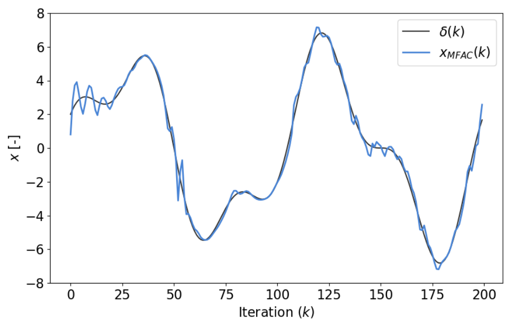
$$\delta(k) = 5\sin\left(\frac{k\pi}{50}\right) + 2\cos\left(\frac{k\pi}{20}\right), \tag{5.10}$$

the parameters and initial conditions from Table 5.1 are used for the simulation. The paper does not link these quantities to physical quantities, as such these quantities have no units.

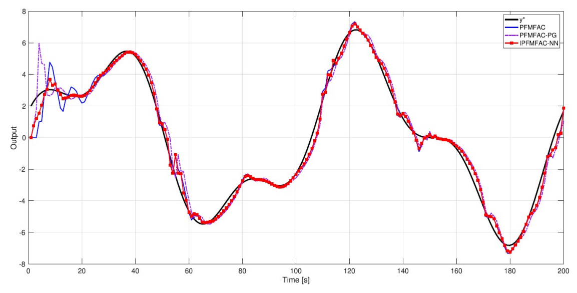
Quanitity	Value
$x(1)$	0
$u(1)$	0
μ	1
ϵ	10^{-5}
L	3
$\hat{\Phi}_{p,L}(k), k=1$	[1, 0, 0]
λ	0.5
ρ	0.5
η	0.05

Table 5.1: Initialization values based on the simulation of Yang et al. (2021)

The result of the simulation is shown in Figure 5.2 below.



(a) Simulation results of classical dynamical system



(b) Results of paper from Yang et al. (2021)

Figure 5.2: Simulation results compared to results from Yang et al. (2021)

As evidenced by Figure 5.2, the result of the simulation is very similar to the paper from Yang et al. (2021). Slight differences have likely occurred due to the fact the paper employs neural networks to tune the MFAC parameters ρ and λ instead of the constant values used in the simulation. From the result of this simulation, it can be concluded the controller is indeed able to control a non-linear dynamical systems

The next step is to apply this control algorithm to the system of interest of this paper. For implementation the dynamical model must be rewritten into discrete form, to simplify this conversion the control relation is made linear. The conversion is extensively done in appendix A and the result is given below:

$$x(k+1) = 2x(k) - x(k-1) + \frac{\Delta t^2}{M_e}\Big(-K_e x(k) + au(k) + F(k)\Big). \tag{5.11}$$

The values of the constants are given in Table 5.2.

Quantity	Value	Unit
M_e	1.082	kg
K_e	10.525	N/m
Δk	0.001	s
a	0.2	-

Table 5.2: simulation quantities

Test case 1 is simulated to asses the performance of MFAC on the linearized discrete system. The initialization values of the MFAC parameters are based on the simulation from Yang et al. (2021). Except for the initial condition of the system, as it would not require any control, if the initial condition for displacement would start at zero.

Quanitity	Value	Unit
x(1)	0.01	m
u(1)	0	N
μ	1	-
ϵ	10^{-5}	-
L	3	-
$\hat{\Phi}_{p,L}(k), k=1$	[1, 0, 0]	-
λ	0.5	-
ρ	0.5	-
η	0.05	-

Table 5.3: Initialization MFAC for simulation system of interest

The discretization step is set to 0.0001 s, because this is the control update frequency in the state of the art controller of the scale set up. The total duration for the simulation is set to 4 seconds or 4000 iterations. The result of the simulation is given below in Figure 5.3.

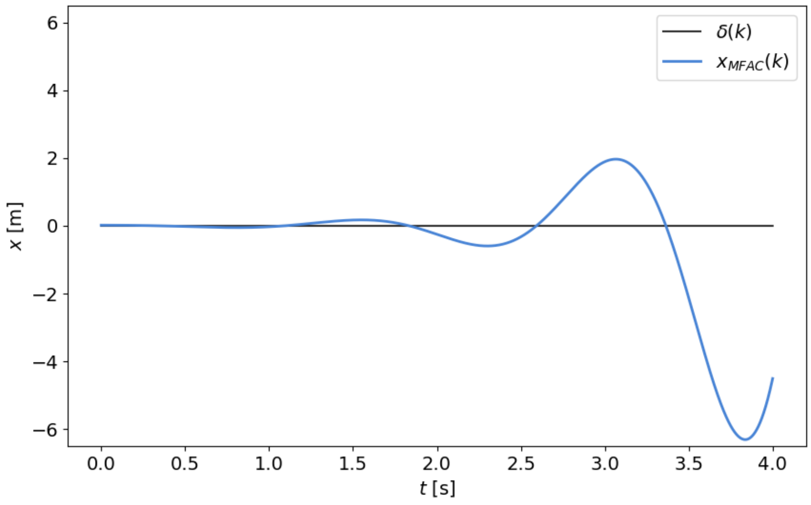


Figure 5.3: Test case 1 simulation of MFAC on the linearized and discretized system of interest

The controller is unable to control the linear system for test case 1, as seen in Figure 5.3. The system is unstable. Because test case 1 is the simplest test case no further simulations are performed. The next section will dive into why MFAC is unable to control the system of interest.

5.4. Discussion

To understand the behaviour of the controlled system a deeper look into the algorithm is required. First off, comparing the mathematical model MFAC uses to predict the next time step with the actual discrete model that is simulated gives:

$$x(k+1) = x(k) + \Phi^T(k)\Delta U(k), \tag{5.12}$$

$$x(k+1) = 2x(k) - x(k-1) - \Delta t^2 \frac{K_e}{M_e} x(k) + \Delta t^2 \frac{a}{M_e} u(k), \tag{5.13}$$

slightly rewriting the system of interest:

$$x(k+1) = x(k) + \left[\Phi^T(k)\Delta U(k)\right], \tag{5.14}$$

$$x(k+1) = x(k) + \left[(1 - \Delta t^2 \frac{K_e}{M_e})x(k) - x(k-1) + \Delta t^2 \frac{a}{M_e} u(k)\right], \tag{5.15}$$

the parts between square brackets should correspond to the same value, for MFAC to predict the correct next time step. However, during simulation this is not the case, the predicted state changes do not coincide with the actual state changes. This is caused by MFAC assuming state changes are predominately caused by control output. The state changes over time as caused by the **A** matrix are neglected. Instead the system is controlled by only estimating the **B** matrix. This approach would be valid if the control response dominates the time response, as is the case for the non-linear system of Yang et al. (2021). This is, however, certainly not the case for the electromagnetic pendulum.

P-controller comparison From the control output formula of the MFAC algorithm, it can be deduced, it primarily works as an incremental P controller with an adaptive gain. If the equation 5.5 is slightly rewritten one obtains equation 5.16. The formula for the P controller is given in equation 5.17 .

$$u(k) = u(k-1) + \frac{\rho_k \phi_1(k)}{\lambda + \phi_1^2(k)}(\delta(k+1) - x(k)) - \frac{\rho_k \phi_1}{\lambda + \phi_1^2(k)} \sum_{i=2}^{L} \phi_i(k)\Delta u(k-i+1) \tag{5.16}$$

$$u(k) = u(k-1) + K_p(k)(\delta(k) - x(k)) \tag{5.17}$$

For $L = 1$ MFAC is a P-controller with an adaptive gain, in which $K_p(k) = \frac{\rho_k \phi_1(k)}{\lambda + \phi_1^2(k)}$. For $L \geq 2$ the controller has a correction for past behaviour of the system as predicted by the model assumption of MFAC. This behaviour is visible in the simulation of test case 1. By plotting both the control output and the displacement along the same time axis, Figure 5.4, it can be concluded that the maxima of the control output coincide with zero state error. If the current error is positive the control output is increased if the error is negative the control output is decreased. This behaviour leads to overshooting. This overshooting combined with the inertia of the system leads to unstable behaviour.

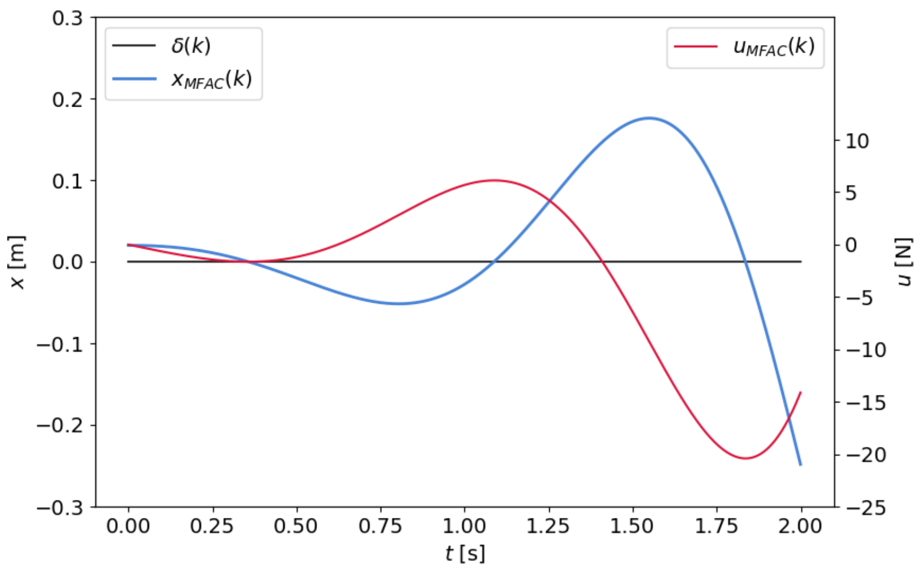
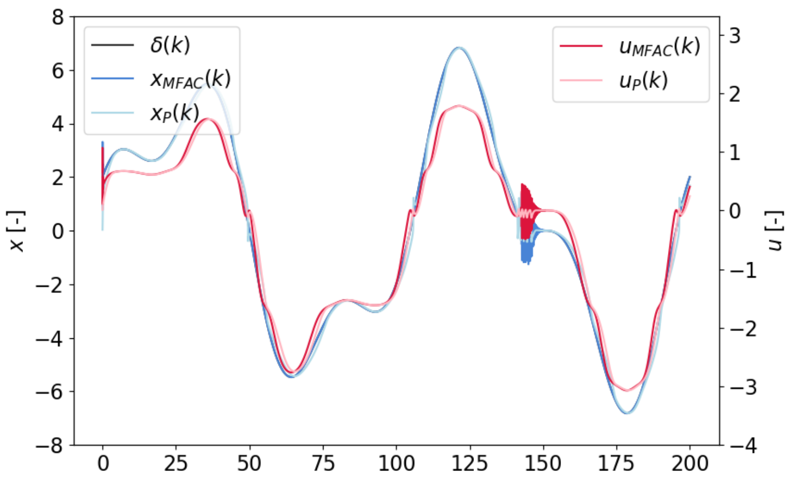
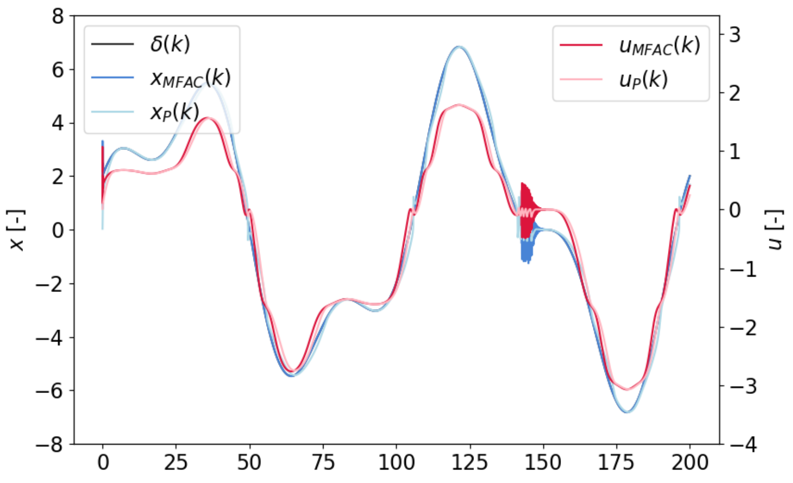


Figure 5.4: Test case 1 displacement and control output

Comparing the performance of a P-controller with a MFAC controller on both systems gives similar results, Figure 5.5. Both controllers are able to control the system from the paper by Yang et al. (2021), Figure 5.5a. The only difference between the P-controller and the MFAC controller is that P-controller requires a higher control frequency to achieve the same performance. In this case the desired desired motion was slowed down by a factor 100, mimicking a 100 Hz control frequency. In the simulation of the system of interest. The unstable behaviour for the system of interest is displayed by both controllers in Figure 5.5b.



(a) System of equation 5.9 ($K_p = 0.005$)



(b) System of interest ($K_p = 0.4$)

Figure 5.5: Comparison between MFAC controller and P-controller with control frequency $f = 100$ Hz

5.5. Conclusion

The controlled system of interest displays persistent unstable behaviour, when controlled by model-free adaptive control. MFAC is shown to primarily behave as an incremental P-controller with adaptive gain. This type of controller is shown to not be suited for the system of interest. MFAC assumes the state changes are caused by the control output only and that the system dynamics can be neglected. This assumption is not valid for the electromagnetic pendulum. Thus, this control approach is not further investigated.

6 Neural network enhanced PD

In Chapter 5, it was concluded that MFAC is not a suitable controller for the electromagnetic pendulum. Consequently, this research now focuses on the second identified controller, the PID-NN. Since the current state of the art controller does not utilize an integral term, this implementation will also exclude it. This chapter explores the integration of a PD controller with a neural network to address the non linearity in the system's response to control output. The chapter is structured as follows: first, the implementation of the controller is discussed; second, the test cases are simulated; and finally, the results are discussed, leading to a conclusion on the applicability of this controller.

6.1. Implementation PD-NN

The neural network enhanced PD applied in this research uses the same gains and structure as a classical PD. The difference is that the gains are not constant and instead made adaptive by the output of a neural network.

The structure of the neural network of the controller is as follows: the neural network has 1 input node, containing $s(t)$, in the input layer, 3 hidden layers with each 30 nodes and 2 output nodes K_p and K_d. All layers are fully connected and use the Leaky ReLU activation function. The structure of the network is visualized in Figure 6.1. The amount of nodes and layers are determined through a heuristic approach. The important takeaway from te design of the structure is that it requires sufficient nodes and layers to approximate the magnetic interaction. However, it should not have too many as this would require a lot of computing time and could lead to an exploding or vanishing gradient.

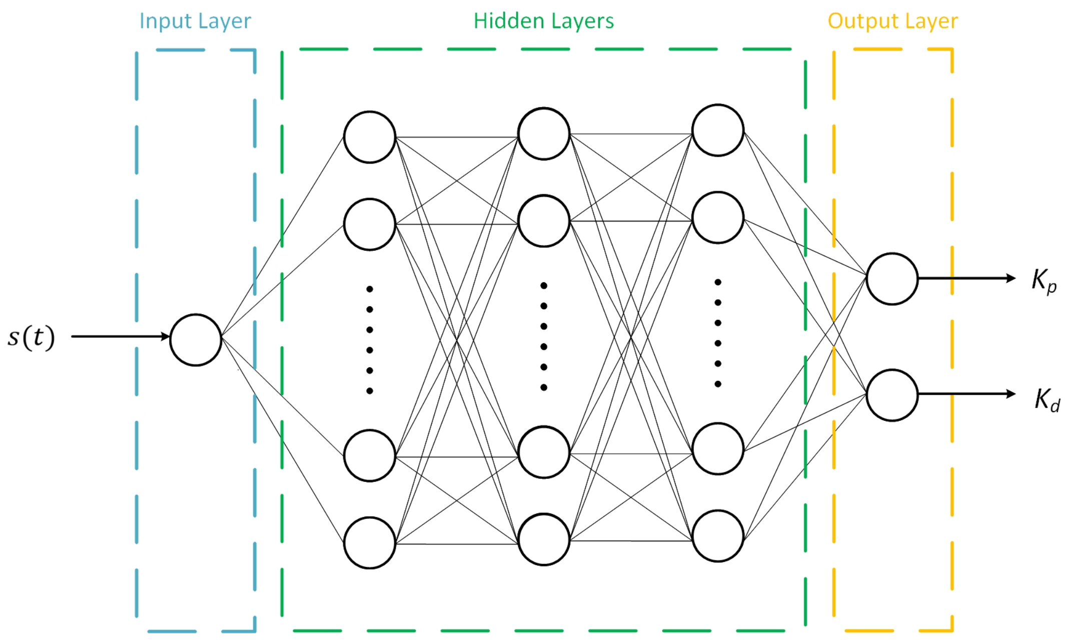


Figure 6.1: Structure of neural network

The mean squared error cost function is used to calculate the cost of the neural network:

$$C = \frac{1}{N}\sum_{i=0}^{N}((K'_{p,i} - \hat{K}_{p,i})^2 + (K'_{d,i} - \hat{K}_{d,i})^2), \tag{6.1}$$

, to compute the MSE true values of the gains are required. These true values are calculated by using the modelled relation of $\frac{\alpha}{s^3}$ from Atzampou et al. (2024). This makes the neural network mimic the model based controller, without explicitly telling the network the relation. Learning is done online using the Adam optimizer. The input node is the distance $s(t)$ between the electromagnet and permanent magnet in meters. The model approach from Section 2 is only dependent on the separation $s(t)$, thus the network is expected to find a relation between the separation and the required gains, based on this one input. The control output can then be calculated as follows:

$$u(t) = K_p(s(t))e(t) + K_d(s(t))\frac{\mathrm{d}e(t)}{\mathrm{d}t}. \tag{6.2}$$

The controller is visualized in the block scheme of Figure 6.2

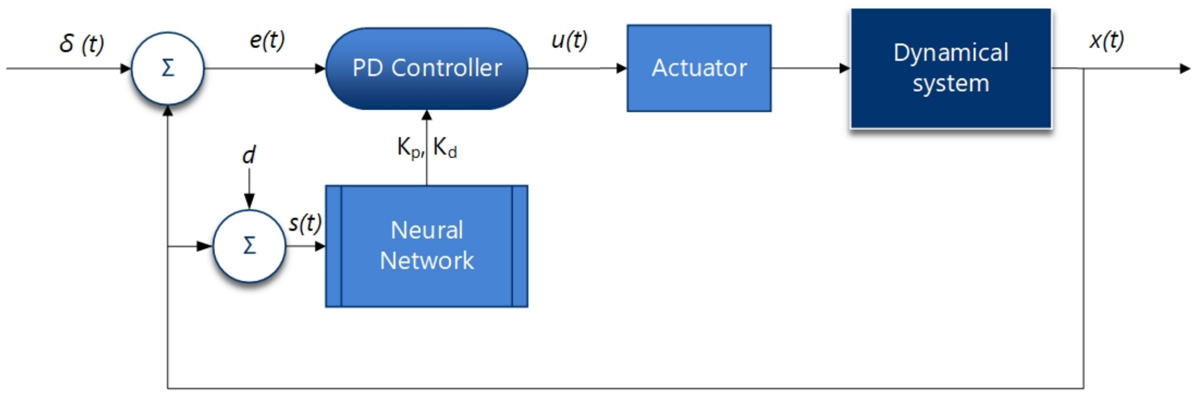


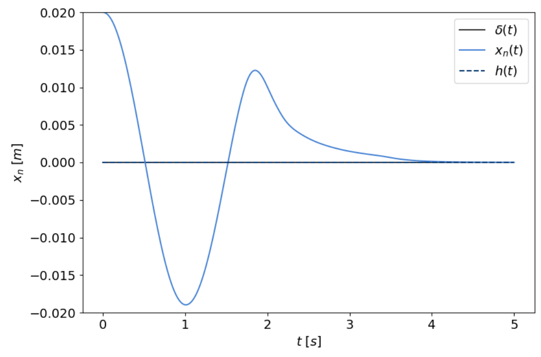
Figure 6.2: PD-NN block scheme

6.2. Test case simulation

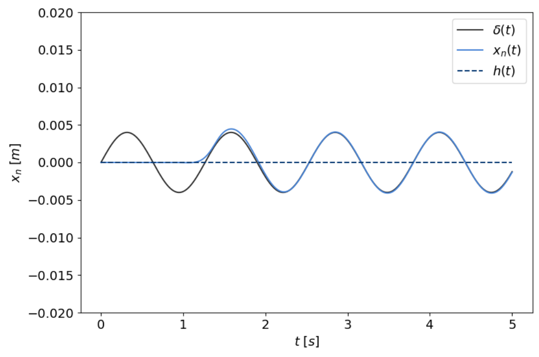
To asses the performance of this proposed control scheme the controller is implemented into the numerical simulation. The simulation is done for all four test cases according to Section 2.3.2. There are two slight deviations. Test case 1 is simulated with a twice as large dispclament and test case 4 is a combination of the other 3 test cases instead of the Jonswap based pivot-point motion. The altered test cases are given in Table 6.1. The results are displayed in Figure 6.3 .

	Test case 1*	Test case 2	Test case 3	Test case 4*
$x(0)$ [mm]	20	0	0	20
$\delta(t) : f$ [Hz]	0	0.75	0	0.75
$\delta(t) : A$ [mm]	0	4	0	4
$h(t) : f$ [Hz]	0	0	0.50	0.50
$h(t) : A$ [mm]	0	0	5	5

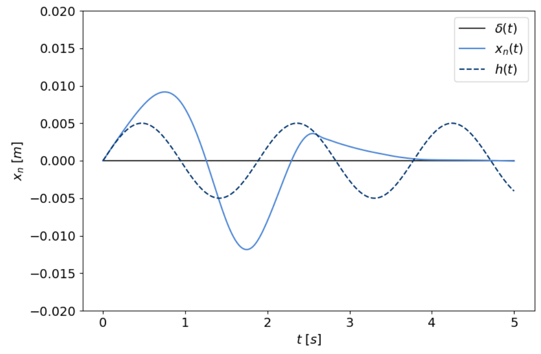
Table 6.1: Overview test cases



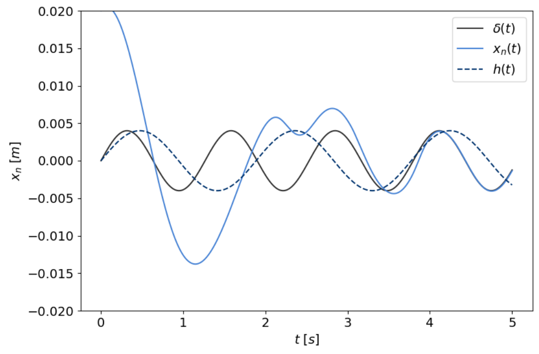
(a) Simulation test case 1*



(b) Simulation test case 2



(c) Simulation test case 3



(d) Simulation test case 4*

Figure 6.3: Results test case simulation PD-NN

6.3. Discussion

From the simulation results, it can be concluded that the controller gives a satisfactory performance. In all 4 simulation cases the system is stable and the system follows the desired trajectory. Initially, the performance is poor as the controller has not yet learned the correct gain, but after some time it has learned the correct relation and positional control is achieved. However, this network uses too much information and is, therefore, not fully model free. The cost function cannot be evaluated without using model information. In this implementation $K'_{p/d,i}$ are based on model information. Since the controller needs to be model-free and general applicable, assuming a dataset of known gains or desired control outputs is available, is clearly false. There is no prerequisite knowledge of the correct control output. Therefore, the cost function should not be based on the control output error, but on the state error as according to Hernández-Alvarado et al. (2016), Patrikar and Provence (1996) and Ponce et al. (2004):

$$C = \frac{1}{N}\sum_{i=0}^{N}(\delta_i - x_i)^2", \tag{6.3}$$

this cost function does however not allow for easy optimizing. To optimize the neural network using any optimizer the gradients of the cost function with respect to the weights need to be calculated. Remember from section 4.6 this is done using the four equations from backpropagation. The first equation that needs to be applied is calculating the errors in the output layer using:

$$\Lambda^{x,L} = \frac{\partial C}{\partial a_j^L}\sigma'(z_j^L), \tag{6.4}$$

substituting the cost function into the expression for one data point gives:

$$\Delta^{x,L} = \frac{\partial(\delta_i - x_i)^2}{\partial a_j^L}\sigma'(z_j^L), \tag{6.5}$$

applying the chain rule to the derivative gives:

$$\Delta^{x,L} = 2(\delta_i - x_i)\frac{\partial(\delta_i - x_i)}{\partial a_j^L}\sigma'(z_j^L), \tag{6.6}$$

the derivative of the desired state towards a_j^L is 0 as the desired state is independent of the neural network:

$$\Delta^{x,L} = -2(\delta_i - x_i)\frac{\partial x_i}{\partial a_j^L}\sigma'(z_j^L), \tag{6.7}$$

the state x is a function of the control output u and the control output u is a function of the output nodes. Applying the chain rule again gives:

$$\Delta^{x,L} = -2(\delta_i - x_i)\frac{\partial x_i}{\partial u}\frac{\partial u}{\partial a_j^L}\sigma'(z_j^L). \tag{6.8}$$

The issue with the remaining formulation is the term $\frac{\partial x_i}{\partial u}$ this term is unknown. The algorithm is model free, meaning the relation between the state of the system and control is unknown and the gradient with respect to the control output is also unknown. This violates assumption 2 (Section 4.6) of the back propagation. Methods tend to differ on how they deal with this challenge. Stochastic approximation leads to SPSA, using the least-squares algorithm gives MFAC and iterative optimzation can yield IFT (Z.-S. Hou and Wang, 2013). According to Ponce et al. (2004) the magnitude of this term can be also adopted into the learning rate. This way only the sign of the term needs to be known. The sign of the gradient is often relatively straightforward to determine and does not require modelling the system. For the system of interest an increase in control output leads to an increase in displacement thus is the sign positive. This means only the gradient $\frac{\partial u}{\partial a_j^L}$ requires computation.

6.4. Conclusion

PD-NN is able to achieve positional control over the system of interest, if the network is optimized based on control output error. However, optimizing based on control output error requires model information or control information. Therefore, this control scheme is not model free. Instead the optimization should be done based on the gradient of the state error e. Calculating the gradient with respect to the state error requires knowledge of the gradient of the state with respect to the control output: $\frac{\partial x}{\partial a^L} = \frac{\partial x}{\partial u}\frac{\partial u}{\partial a^L}$. Model-free control algorithms, however, have no knowledge of the gradient of the state with respect to the control output, because this would require a model description. The crux for any DDC approach lies in how to calculate this gradient $\frac{\partial x}{\partial u}$. For PID controllers using neural networks to tune the gains, the magnitude of this term can be absorbed in the learning rate as done by Ponce et al. (2004), requiring only knowledge of the sign of the gradient to optimize the network.

7 PWORNN

This chapter details the implementation of a PID controller based on a polynominal weighted output recurrent neural network (PWORNN) (Hanna et al., 2022). In the previous chapter, the conclusion was reached that the cost function should use the state error to tune the weights instead of the control output error. This PWORNN implementation uses the state error and optimizes using a proposed Lyupanov stability criterion. Polynominal weighted output means the network used is not a typical neural network. The last layer of the network is a *pi unit*, the inputs are multiplied in the node instead of summated as usual in a neural network. This differs from self-organizing PNN's (Oh et al., 2003), that are also called polynomial networks. Recurrent means in this case the network uses its own output as input.

7.1. Controller Structure

The implementation of the algorithm is done according to the paper of Hanna et al. (2022). The controller is an incremental adaptive PID controller. The adaptive gains are calculated by the PWORNN. For discrete time the algorithm can be represented as:

$$u(k) = u(k-1) + K_p(k) \cdot (e(k) - e(k-1)) + K_I \cdot e(k) + K_d \cdot (e(k) - 2e(k-1) + e(k-2)) \tag{7.1}$$

The block scheme of the controller is given in Figure 7.1.

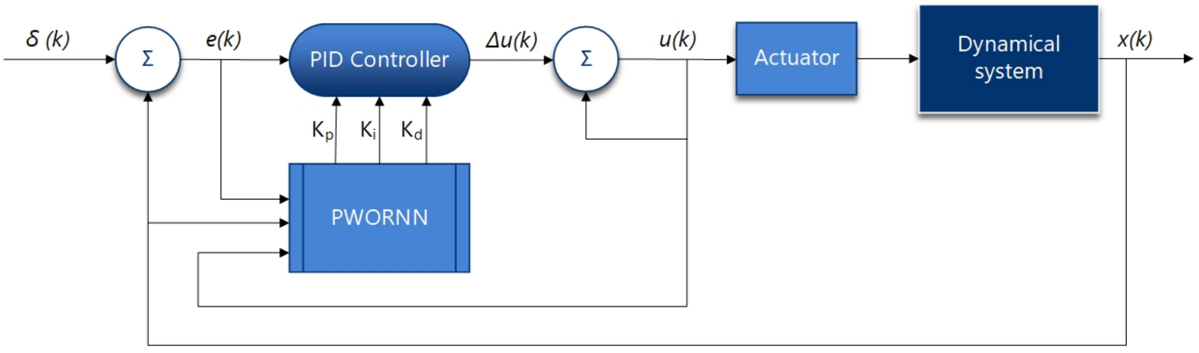


Figure 7.1: Block Diagram of PWORNN based PID controller

The structure of the PWORNN is a 3-2-3 structure, according to Figure 7.2 .

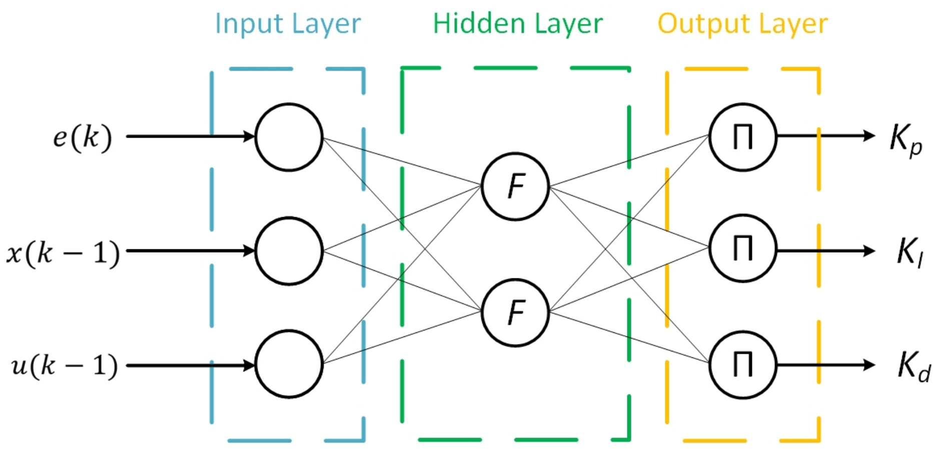


Figure 7.2: Structure PWORNN

The input layer is vector $X_i(k) = [e(k), x(k-1), u(k-1)]$. It exist of the current error, the previous state and the previous input. The hidden layer has two nodes of the same value, it is the summation of the inputs (e.g. weights = 1 and bias = 0). This layer is not updated and the values of the weight shall thus remain constant, throughout the simulation. This saves computational time and memory space. The value of the hidden node is called F and is formulated as follows:

$$F_j(k) = \sum_{i=1}^{3} X_i(k). \tag{7.2}$$

Finally, the value of the output neurons is calculated by multiplication of the inputs of the output layer:

$$K_p(k) = \prod_{j=1}^{M} w_{j1}(k)F_j(k), \tag{7.3a}$$

$$K_i(k) = \prod_{j=1}^{M} w_{j2}(k)F_j(k), \tag{7.3b}$$

$$K_d(k) = \prod_{j=1}^{M} w_{j3}(k)F_j(k), \tag{7.3c}$$

$$\tag{7.3d}$$

for this implementation $M = 2$.

7.2. Optimization

As outlined in section 6.3, the difficult part of optimizing the network is to go from the state error towards updating the weights of the network. Because the weights influence the state error indirectly through the control output, and the influence of the control output on the state is unknown. This implementation follows the paper of Hanna et al. (2022) and optimization is done based on the state error and the weight update rule is derived through the use of a Lyupanov function:

$$V_L(k) = ae^2(k) + 2be(k)\boldsymbol{w}(k) + c\boldsymbol{w}^2(k), \tag{7.4}$$

in which the constants a, b, c can be arbitrarily chosen, given they satisfy two conditions: $a > 0$ and $ac - b^2 > 0$. $e(k)$ is the state error at iteration k and $\boldsymbol{w}(k)$ is the vector with the weights of the output layer. The stability criterion states that the system must satisfy is:

$$\Delta V_L(k) = V_L(k+1) - V_L(k) \leq 0, \tag{7.5}$$

from this definition the weight update rule can be determined. The weight update is each iteration step, which corresponds to stochastic gradient descent:

$$w(k+1) = w(k) + h\Delta w(k), \tag{7.6}$$

where h is a constant learning rate. This contrasts slightly with the reference paper, as it implemented an adaptive learning rate. The adaptive learning rate is based on an upper bound calculation. This ensures the learning rate stays between zero and this upper bound for each time step. This adaptive ability is, however, not required, if the learning rate is sufficiently small. The trade off is that convergence will be slower. $\Delta w(k)$ is the weight update and is calculated based on the stability criterion from equation 7.5:

$$\Delta w(k) = -\frac{1}{c}(L_1 + \frac{\partial e(k)}{\partial w(k)} L_2), \tag{7.7}$$

in which L_1 and L_2 are used for readability and are actually:

$$\begin{aligned} L_1 &= (2b\Delta e(k) + 2be(k) + 2cw(k)), \\ L_2 &= (2a\Delta e(k) + 2ae(k) + 2bw(k)). \end{aligned} \tag{7.8}$$

Similar to equation 6.5 in Section 6.3, this rule still contains the gradient of the state error to the network weights $\frac{\partial e}{\partial w}$. Expanding the derivative using the chain rule gives: $\frac{\partial e}{\partial w} = \frac{\partial - x}{\partial w} = -\frac{\partial x}{\partial u}\frac{\partial u}{\partial w}$. The derivative $\frac{\partial x}{\partial u}$ is unknown, because there is no model. The magnitude is considered irrelevant for this approach, as it can be absorbed in the learning rate h. However, the sign of the derivative is important (Patrikar and Provence, 1996). This sign is (+) positive, because increasing the the control input (e.g. increasing voltage) leads to increased displacement. Thus, the weight update rule is:

$$w(k+1) = w(k) - h \cdot \frac{1}{c}\left(L_1 - \frac{\partial u(k)}{\partial w(k)} L_2\right). \tag{7.9}$$

7.3. Test case simulation

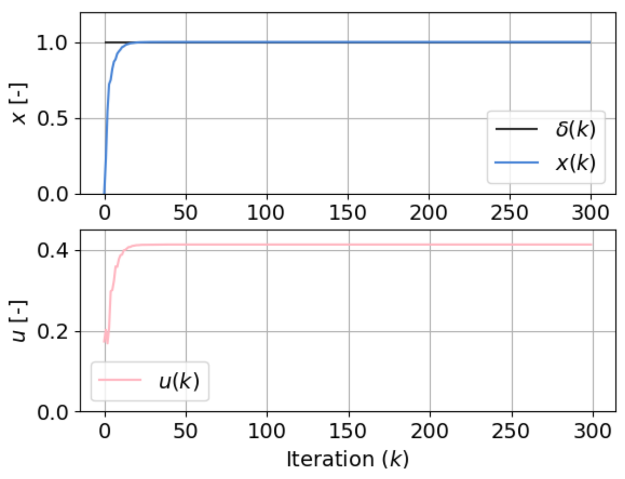
This controller is tested numerically to asses the performance. First, the coded implementation of the controller is checked by comparing the simulation results to the results presented in the original paper from Hanna et al. (2022). The original paper controls the following dynamical system:

$$x_p(k+1) = \frac{x(k)}{1 + x^2(k) + x^2(k-1)} + \frac{0.1}{1 + \mathrm{e}^{-\left(x(k)+x(k-1)\right)}} + u(k) + 0.4u(k-1), \tag{7.10}$$

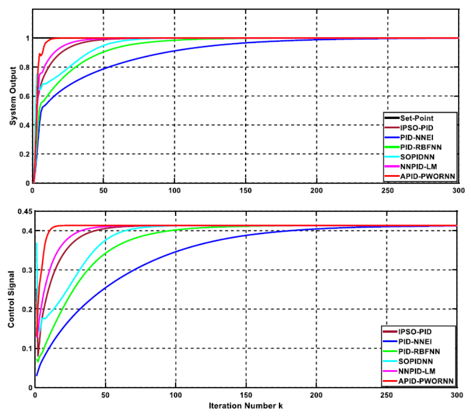
in which:

$x(k)$	state of the system;
$u(k)$	control output.

The initial condition is set to 0 ($x(0) = 0$) and the desired state is 1 ($\delta = 1$). This test case is, thus, a unit-step response. The result of this simulation is given in Figure 7.4. This result is very similar to the one presented in the reference paper (Hanna et al., 2022) given by the red line in Figure 7.3b. However, due to the random initialization of the weights, it is not exactly identical or repeatable.



(a) Simulation result PWORNN



(b) Results reference paper Hanna et al. (2022)

Figure 7.3: Result comparison between simulation and reference paper

Figure 7.3b proves that this controller is able to control a non-linear system. Thus, the controller is applied to the system of interest. For this simulation, the discrete linear model will be used from appendix A, because this control algorithm is based on iteration number and not time. The parameters of the equation are given in Table 7.1 and the equation is repeated below:

$$x(k+1) = 2x(k) - x(k-1) + \frac{\Delta t^2}{M_e}\Big(-K_e x(k) + a u(k) + F(k)\Big). \tag{7.11}$$

	Value	Unit
M	1.082	kg
K	10.525	N/m
Δt	0.001	s
α	0.2	-

Table 7.1: Simulation quantities

For test case 1, the initial condition is $x(0) = 0.01$ m and desired trajectory is $\delta(k) = 0$ m. For the first time step also the PWORNN needs to be initialized and requires three input values. This causes issues, because $x(k) = -e(k)$, u must be initialized as non zero to avoid $F = 0$. If the sum of the inputs start at 0 the controller will never start as the gains will remain at zero. To combat this u is initialized at 0.1.

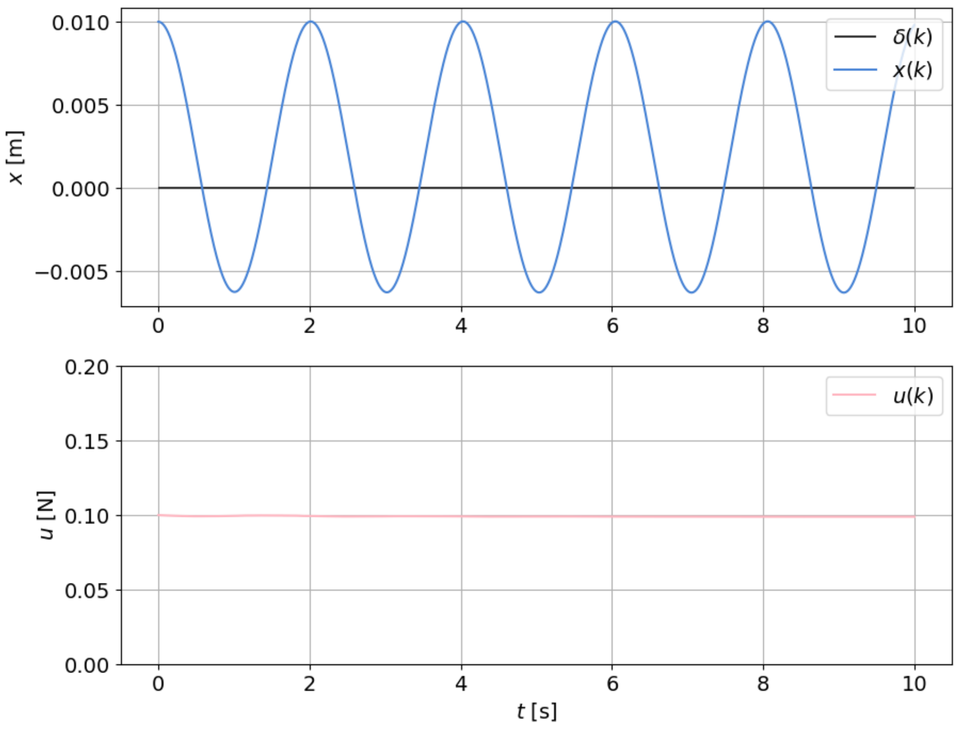


Figure 7.4: Test case 1 simulation PID-PWORNN, system of interest

As seen from Figure 7.4 , this approach does not yield meaningful results. The control value hardly changes staying almost constant. This can be understood by looking at the time series of the gains, Figure 7.5.

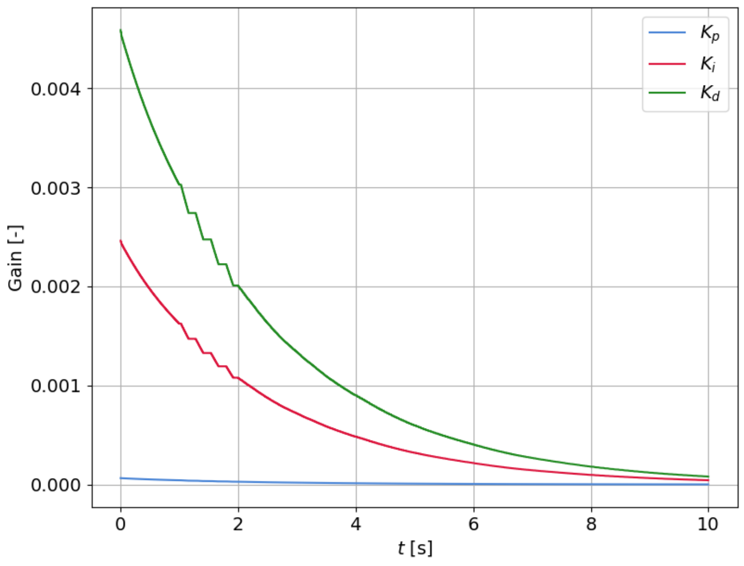


Figure 7.5: Evolution of PID gains during simulation

7.4. Discussion

From the plot in Figure 7.5, it can be deduced that the algorithm is not learning any behaviour. Every time step the gains are decreased. This eventually leads to zero gains and the output value is no longer incremented irrespective of error. The trend to zero is caused by the formula for L_1. The weights are large compared with the error and especially compared to the derivative of the error. Therefore, most time steps the weight is simply reduced by the learning rate times itself. This will always lead to decreasing weights and thus decreasing gains. The PID is implemented as an incremental PID, the previous input is the starting point of the new input, thus the control value will become constant for small gains. Furthermore, updating is also proportional to the weights themselves. Once the weight becomes small the updating value also becomes small and the algorithm becomes stuck. Only when the errors are large the controller will increase the gains. In this case large means compared to the weights. This controller shows good performance for the system of the original paper, because the test case of the paper is a unit step requiring a constant control value from the controller. This algorithm is made to find this control value by incrementing the gains based on state error. However, the system of interest is a quickly oscillating system that requires a time varying control output the PID-PWORNN is unable to achieve this.

7.5. Conclusion

This PID-PWORNN method does not show satisfying control for the system of interest. The performance is initialization sensitive and the control output convergences to a constant value irrespective of the state of the system. Therefore, it is concluded this algorithm is unable to learn meaningful relationship between the state error and the control output. Thus the method of using a modified weight updating rule based on state error, and absorbing the magnitude of the gradient $\frac{\partial x}{\partial u}$ in the learning rate is not suitable for the system of interest. As presented in the previous Chapter, the crux lies in optimizing the network without knowing model information. This chapter shows, that an incremental PID controller with gains optimized based on the state error does not lead to satisfactory control. Therefore, it is concluded that using available model information to create a differentiable cost function, while not requiring knowledge of experimentally determined parameters of the system, is the next focus of this research.

8 PD-Force estimating neural network

The previous chapter concluded that optimizing a network based on state error did not yield the desired results. Consequently, requirement 2 is slightly relaxed for the control algorithm in this chapter. Instead of implementing a model-free controller, a controller is proposed that leverages some model information, however, it avoids using experimentally determined parameters. This proposed controller is called PD-Force estimating neural network controller (PD-FeNN). It utilizes the equation of motion of a pendulum to estimate the interaction force of the magnets, incorporating these estimates into the cost function to train the neural network. This approach allows for optimization with known gradients. The objective of the training is to teach the neural network the relationship between the state of the pendulum and the magnetic interaction force. The outputs from the trained network are then used to generate state-dependent gains.

8.1. Assumption and Estimation

This controller is based on two assumptions:

Assumption 1: The system can be described by the equations of a controlled linear unforced undamped pendulum.
Assumption 2: The magnetic interaction force scales linear with the voltage.

If the assumptions hold the following equation should describe the physical system:

$$M\ddot{x} + Kx = \gamma(\cdot)u, \tag{8.1}$$

$\gamma(\cdot)$ is an arbitrary function that gives the magnetic force exerted on the pendulum system, when multiplied by the output voltage. The function can take any input. This implementation of the proposed controller further assumes this function only depends on s. Based on the previous research from Atzampou et al. (2024) and the estimation explained in Section 2.2.2. The output of the γ function can be estimated by rewriting equation 8.1 to have the interaction term on the left hand side:

$$\gamma(s) = \frac{M_e\ddot{x} + K_e x}{u}, \tag{8.2}$$

calculating the value of γ requires knowing all values on the right hand side. This requires measuring the position and acceleration of the system. This can be achieved through sensors that measure position x only. Using a Savitzky-Golay filter (Savitzky and Golay, 1964), the acceleration can be calculated from the time series of the position x:

$$\hat{\ddot{x}}_i = \frac{-126x_{i-4} + 371x_{i-3} + 151x_{i-2} - 211x_{i-1} - 370x_i - 211x_{i+1} + 151x_{i+2} + 371x_{i+3} - 126x_{i+4}}{1716\Delta t^2}, \tag{8.3}$$

in which $\hat{\ddot{x}}_i$ is the estimated acceleration in m^2/s for data point i and Δt is the measurement frequency in seconds. Furthermore, knowledge on the mass and length of the pendulum is required, to calculate the effective mass M_e and effective stiffness K_e.

8.2. Control algorithm

The control algorithm is a state-dependent PD controller. The state dependency comes from the non-linear relation of the magnetic interaction. By placing the inverse of the magnetic interaction in front of the gains as shown by Atzampou et al. (2023) control of the system of interest can be achieved. The difference compared to the modified controller from Atzampou et al. (2024) is that the interaction term is predicted by a neural network instead of being based on a mathematical model developed beforehand. The control algorithm can be expressed as follows:

$$u(t) = \frac{1}{\hat{\gamma}\big(s(t)\big)} K_p e(t) + \frac{1}{\hat{\gamma}\big(s(t)\big)} K_d \dot{e}(t), \tag{8.4}$$

in which $\hat{\gamma}\big(s(t)\big)$ is the output of the neural network approximating the function $\gamma\big(s(t)\big)$. The controller is visualized in a block scheme in Figure 8.1 .

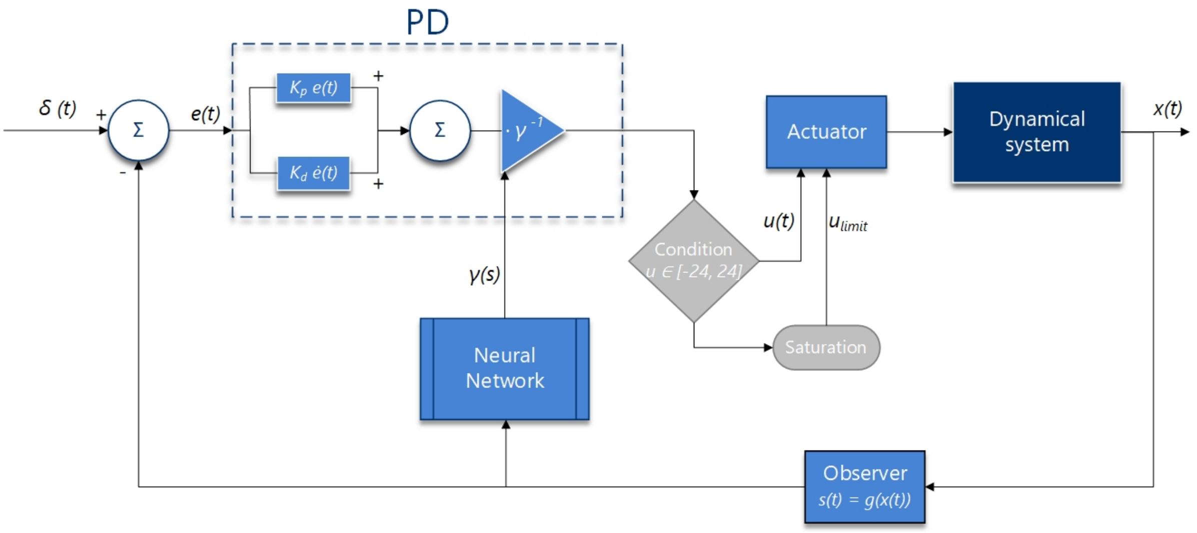


Figure 8.1: Block scheme PD-NN controller

8.3. Neural network

The neural network is used to approximate the function $\gamma(\cdot)$ and find a relation between the state of the system and the estimated interaction force.The network exists out of the following layers:

The **Input layer** contain the input of the function $\gamma(\cdot)$, so all the parameters that influence the magnetic interaction between the electromagnet and the pendulum. As mentioned in Section 8.1 it is assumed only the distance s between the electromagnet and pendulum is relevant . In Section 8.7.3 it will be futher explored whether only using the distance s as input is reasonable.
The 5 **Hidden Layers** contain 20 nodes and are densely connected with a Leaky ReLU activation function. The **Output Layer** is 1 node that gives the estimate for the magnetic interaction $\hat{\gamma}$. The structure is visualized in Figure 8.2. The amount of hidden layers and nodes have, similar to Chapter 6, been determined through a trial and error heuristic approach. The upper limit of the amount of layers and nodes is governed by the computing time of a feedforward pass in the microprocessor from the scale model, requiring it to remain below 100 μs .

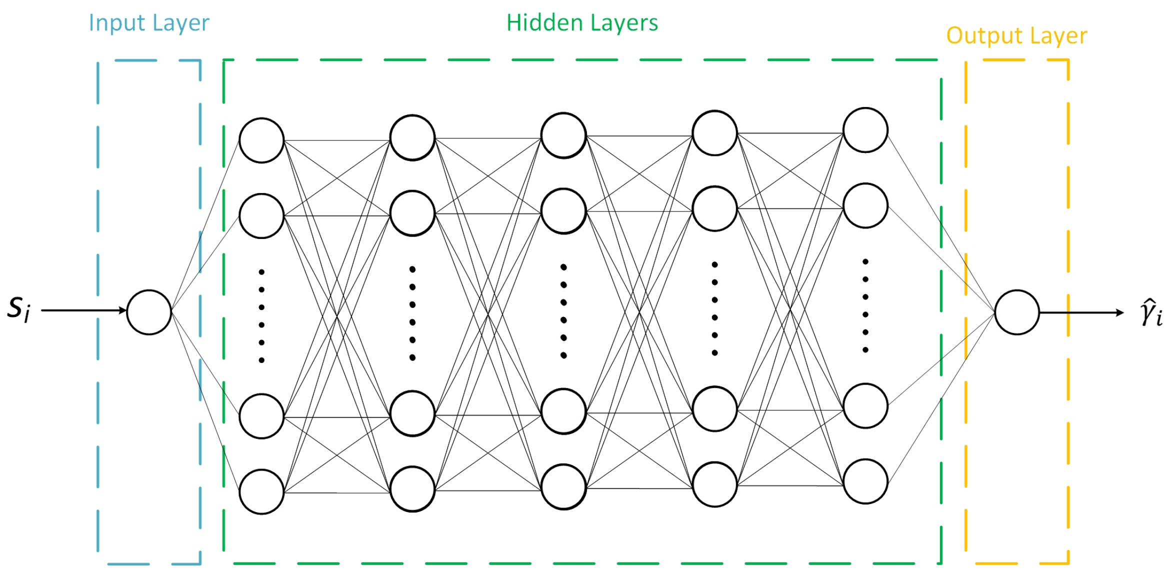


Figure 8.2: Structure Neural Network

Optimization is done by the Adam optimizer and the cost function of the network is Root mean squared error. This cost function is chosen to lessen the impact of outlier data points on the prediction results.

$$C = \frac{1}{N}\sum_{i=1}^{N}\sqrt{(\gamma_i - \hat{\gamma}_i)^2}, \tag{8.5}$$

in which γ_i is the estimated output, calculated with the following equation:

$$\gamma_i = \frac{M\hat{\ddot{x}}_i + Kx_i}{u_i}. \tag{8.6}$$

8.4. Learning Phase

Initially the controller has no knowledge of the interaction and is, therefore, unable to control the system. A *learning phase* is required during which data points of the interaction $\gamma(\cdot)$ can be collected. This data is consequently used to train the neural network. This phase must be long enough to collect sufficient data points for learning and these data points must be spread over the entire operational range, because the neural network is poor at extrapolation (Section 8.7.2).

8.4.1. Time Delay

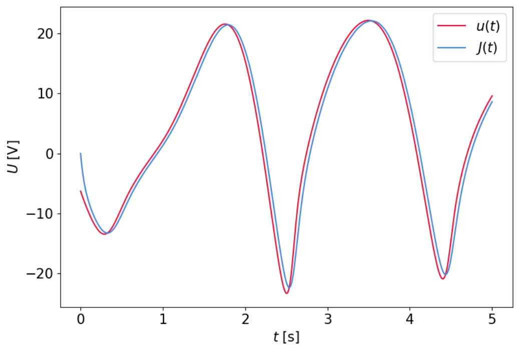
Assumption 1 from Section 8.1 is not fully satisfied by the system of interest. Remember, the modelled equations of motion of the system from Section 2.2.3 are:

$$M_e x + K_e x = -(M + \frac{m}{2})\ddot{h} + \mu\,\text{sgn}(\dot{x}) + \frac{\alpha J}{s^3} \tag{8.7a}$$

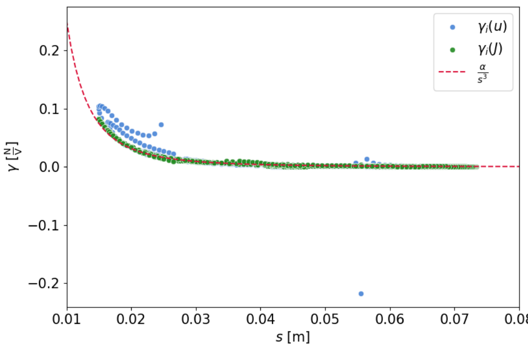
$$\dot{J} = \frac{1}{\tau}(u'(t) - J) \tag{8.7b}$$

Control output changes do not influence the system immediately, because changes in the magnetic field are resisted. This occurs due to the coil of the electromagnet being subject to self inductance (Khan Academy, n.d.). This is modelled by the addition of a time delay between control output u and voltage in the electromagnet J in equation 8.7b. Thus estimation should be done based on voltage J instead of

u. However, J is only known during simulation and not measured in the scale model. The difference between using the control output u and the voltage J can be seen in Figure 8.3b. This figure shows the estimations of γ for both methods during a simulation with an oscillating control output, Figure 8.3a. The Figure 8.3 is created by simulating the system of interest using the non-linear state-space representation and varying the control output value in time. The green dots are estimates calculated with the voltage in the electromagnet J and the blue dots are estimates calculated with the control output u.



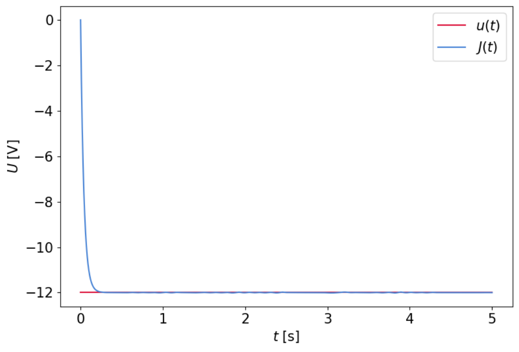
(a) Time series control output and voltage



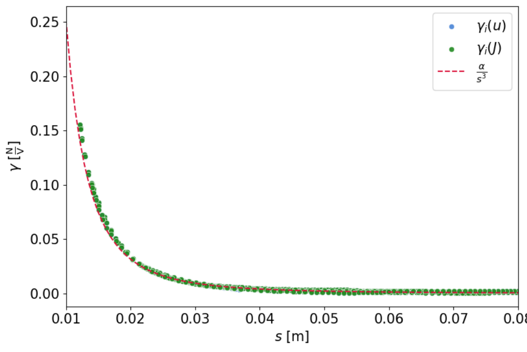
(b) Estimates magnetic interaction

Figure 8.3: Oscillating voltage during learning phase

From Figure 8.3, it can be deduced that the time delay effects the estimation. Errors in the estimations lead to worse predictions by the neural network, because it uses the estimates as data points to evaluate the cost function, and optimize its parameters. This problem can be dealt with by decreasing the rate of change of the voltage, by setting the control output to a constant value. This nullifies the effect of self inductance and the approximation of $u_i \approx J_i$ becomes valid. This is visualized in Figure 8.4, where the control output is set to a constant repelling value. Both the green and blue dots are now equally close to the model line, they originate from. The time delay no longer effects the estimation accuracy. Thus, it is important to keep the rate of change of the control output small.



(a) Time series control output and voltage



(b) Estimates magnetic interaction

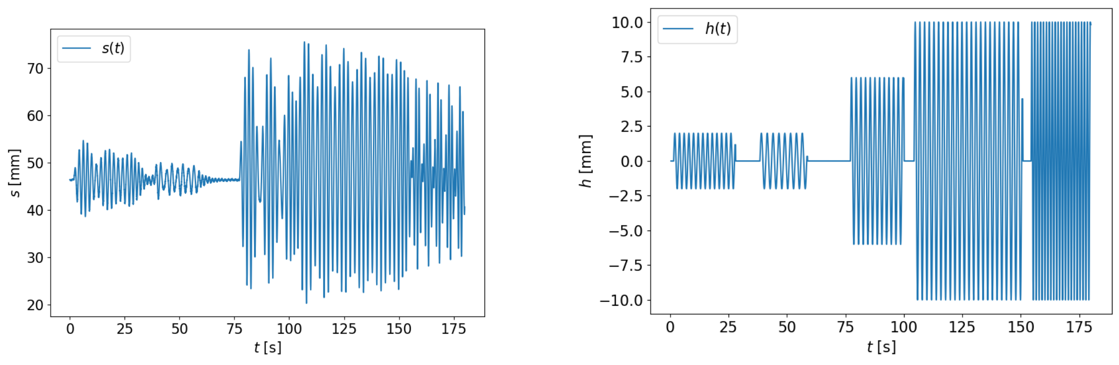
Figure 8.4: Constant voltage during learning phase

8.4.2. Offline learning

The learning phase of the network can be done online during operation of the pendulum or offline based on data collected from previous operations of the pendulum. This does not influence control performance after training. The network will learn the same relation, granted sufficient data points are collected, while taking into account time delay and the operational range. In the practical implementation, there is a difference. The scale model employs a Teensy 3.6 micro-controller to calculate the control output and direct the electromagnet. Implementing an offline trained network on a micro controller is relatively easily, while implementing online learning, involving backpropagation and weight updating is very difficult. On top of this an online implementation requires more available memory and will increase the computing time, potentially requiring different hardware, and limiting the control frequency. Therefore, all training for the scale model has been done offline prior to operation. This requires the micro controller to only do feed-forward passes, which given a sufficiently small network can be completed within 100 μs.

8.4.3. Learning phase implementation

During the learning phase the magnet is repelled by a constant negative voltage, ensuring no collision occurs between electromagnet and payload. This control voltage must be greater than zero to ensure proper estimation. If u_i is close to zero estimates of γ_i become very large and inaccurate. Applying a greater force is also beneficial because the effect of the electromagnet becomes greater compared to the measurement error. This increases the signal to noise ratio and improves the estimations. During the learning phase, the payload is excited through prescribed motion of the pivot pointn for instance a harmonic wave with an amplitude of 10 mm and frequency of 0.5 Hz, $h(t) = 0.01\sin(2\pi 0.5t)$. The exact excitation, however, does not matter. It is mainly important the learning phase collects datapoints throughout the entire operational range as extrapolation by the network is poor. The controller is trained on a learning phase with different pivot point excitation's over a period of 180 seconds, the time series of this learning phase with changing pivot excitation can be found in Figure 8.5 , displaying both the time series of the pivot point, as well as the time series of the separation between the magnets.



(a) Separation between permanent magnet and electromagnet

(b) Pivot-point motion

Figure 8.5: Time series learning phase

From the time series of the separation and the time series of the control output $u(t) = -18$ V, estimations of the magnetic interaction can be made according to equation 8.2. These estimations are subsequently used as data points to train the neural network. The network is trained for 10 epochs with a batch size of 32 data points. The value of the cost function per epoch is presented in Figure 8.6 for 10 different initializations. The cost functions rapidly decrease from the first to the second epoch, after which they remain relatively constant. Convergence of the network is achieved fast, because it is a single dimensional problem. The 10 epochs are used to have a conservative approach. All initializations lead to approximately the same final cost value. This means the initializiation does not effect the performance of the controller. The model from Atzampou et al. (2024), the data points and the fitted relationship from the neural network are plotted in Figure 8.7 .

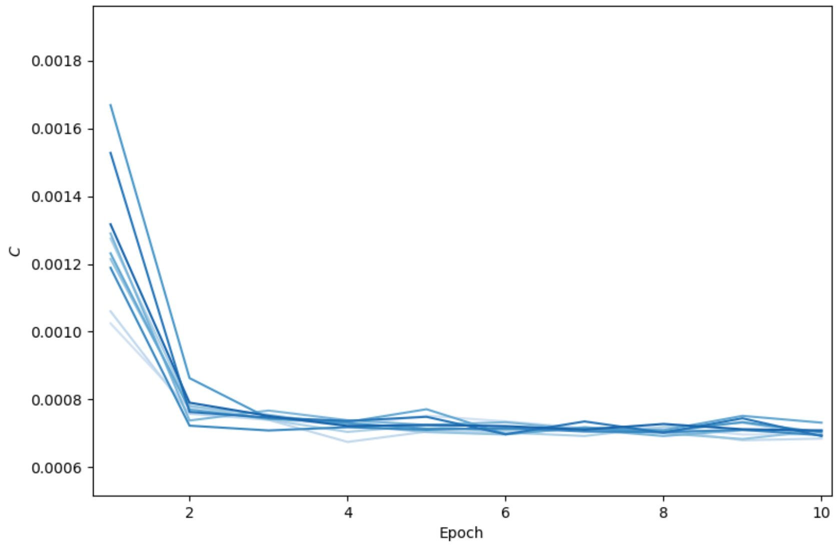


Figure 8.6: Cost per Epoch

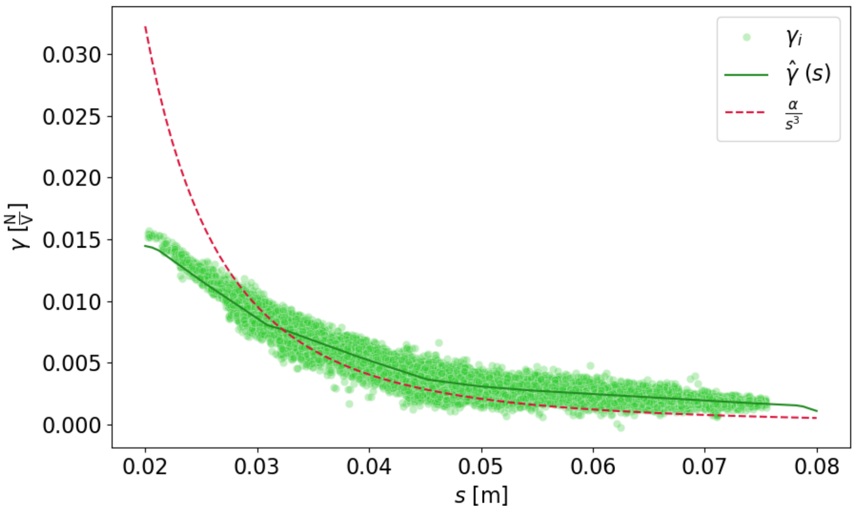


Figure 8.7: Learned relation by neural network

8.5. Test case experiments

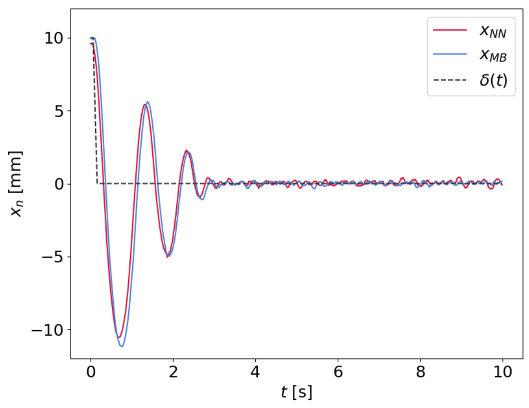
The PD-FeNN controller is assessed by experiments on the scale model with the 4 identified test cases. The test cases are repeated in Table 8.1. The positional control is compared to the modified PD controller, to asses performance. The control methods are compared based on filtered error, step, cumulative error and root mean squared error.

- Root mean squared error: RMSE = $\sqrt{\frac{1}{N}\sum_{i=0}^{N}(e_i)^2}$
- Cumulative error: $e_{cum}(t_n) = \sum_{i=0}^{n} e(t_i)$
- Filtered error: $e_f(t_n) = \frac{\sum_{i=0}^{25} e(t_{n-12+i})}{25}$

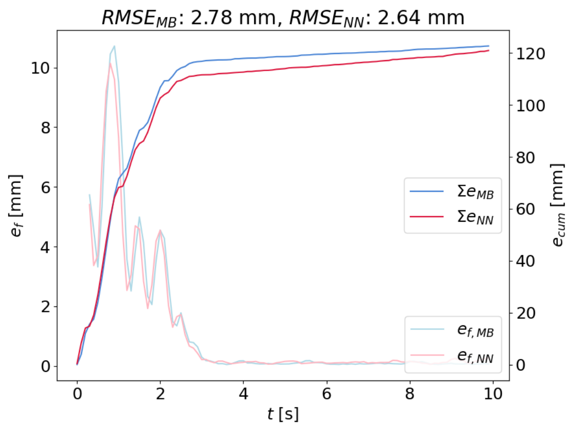
The errors are calculated for time values that have a measured error. The measurement is of course not continuous. The frequency is 100 Hz thus t_n stands for a time value with a measurement and t_{n+1} is the measurement one time step later. The root mean squared error is useful, because it gives a single quantitative value for the performance in a test case. The filtered error is useful to identify and compare, the error at each time step. The cumulative error is the integral of this error and displays the development of these errors and aids in comparing the error values. The results of the test cases are displayed in Figure 8.8 to Figure 8.13. The MB subscript stands for model based and corresponds to the modified PD controller of Atzampou et al. (2024). The NN subscript stands for neural network and corresponds to the proposed PD-FeNN controller.

	Test Case 1	Test Case 2	Test Case 3	Test Case 4
$x(0)$ [mm]	10	0	0	0
$\delta(t) : f$ [Hz]	0	0.75	0	0.50
$\delta(t) : A$ [mm]	0	4	0	5
$h(t) : f$ [Hz]	0	0	0.50	multiple
$h(t) : A$ [mm]	0	0	5	multiple

Table 8.1: Overview test cases

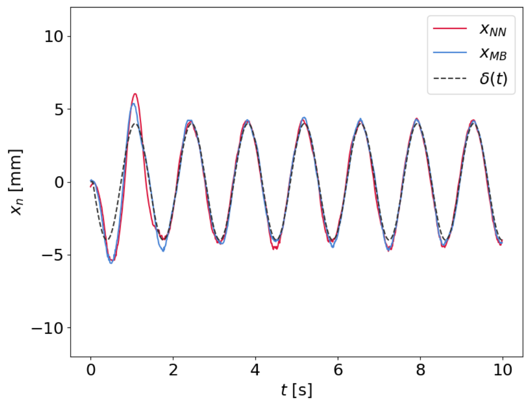


(a) Time series displacement

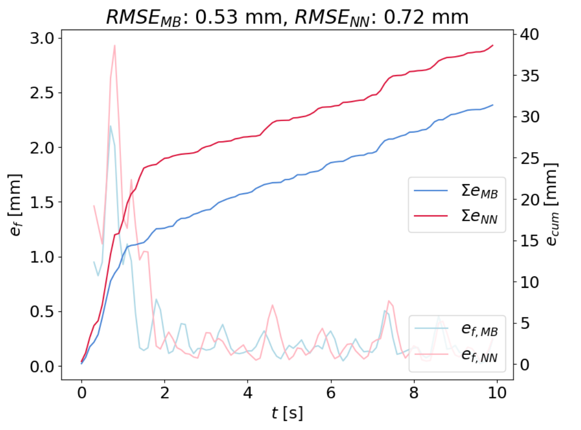


(b) Time series error

Figure 8.8: Test Case 1

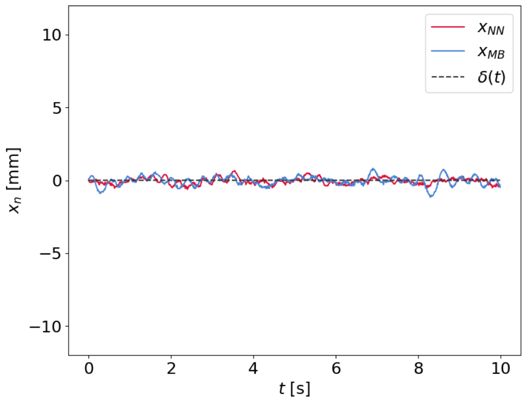


(a) Time series displacement

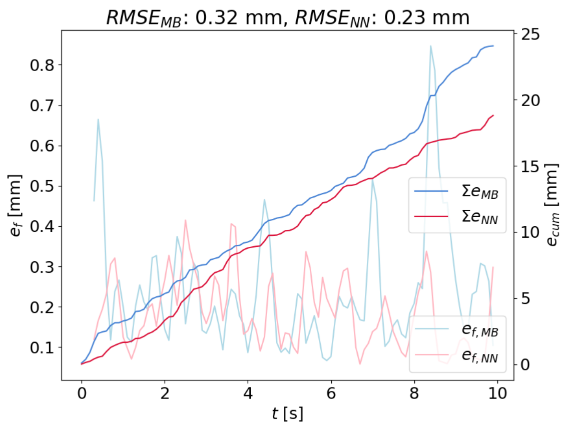


(b) Time series error

Figure 8.9: Test Case 2

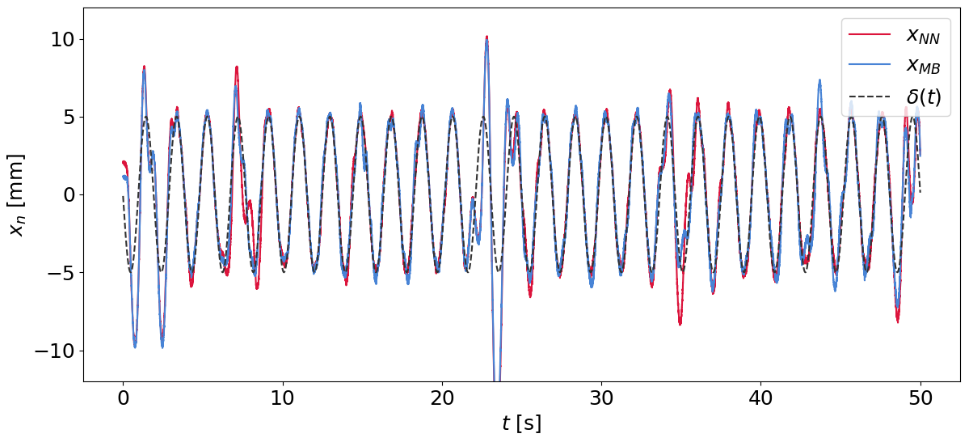


(a) Time series displacement

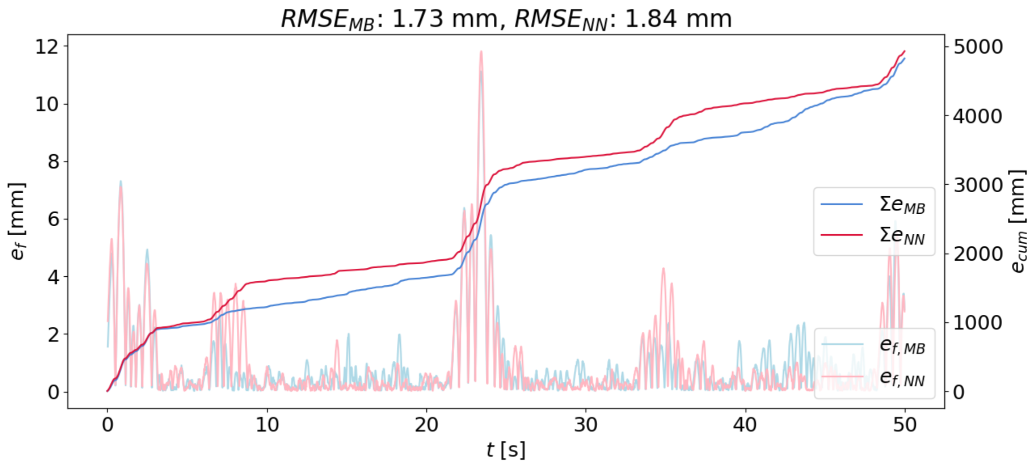


(b) Time series error

Figure 8.10: Test Case 3



(a) Time series displacement



(b) Time series error

Figure 8.11: Test Case 4

The control performance of both controllers is similar for all test cases. The RMSE only differs around ±0.1 mm and the RMSE does not exceed 3 mm for any test case. Furthermore, the majority of this error is caused by transient effects caused by a mismatch between the initial conditions and desired trajectory, the steady-state error is much smaller. The transient effect is clearly shown in Figure 8.8 and 8.9, because the mismatch between the initial conditions and the desired states is the largest for these test cases. The error during these test cases is large initially as seen by the steep slope of the cumulative error. After some time the transient effects pass and the slope of the cumulative error decreases. In test case 3, Figure 8.10, there is no transient response. The slope of the cumulative error remains almost constant and the steady-state RMSE is approximately 0.3 mm.

8.6. Scale model modification

The advantage of the proposed method is, that the control algorithm is not dependent on manual calibration and does not require prerequisite knowledge on the electro- or permanent magnet. The control algorithm is independent of the properties of these components in the scale model.

8.6.1. Changing permanent magnet

The permanent magnet (PM1) is replaced by a larger permanent magnet (PM2) in the scale model. They are shown side by side in Figure 8.12. From PM2 it is known, it is a stronger magnet than PM1, therefore, the α factor in the model description is expected to be greater. In other words the estimated interaction points will have a greater value for the same distance.

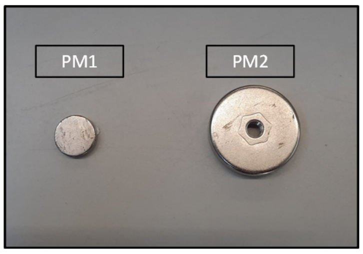
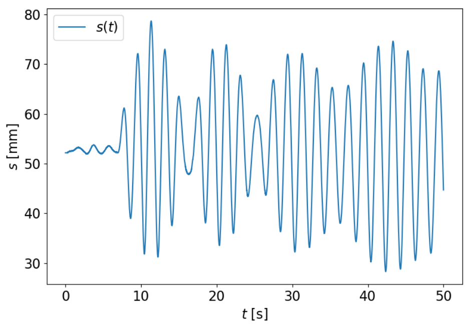


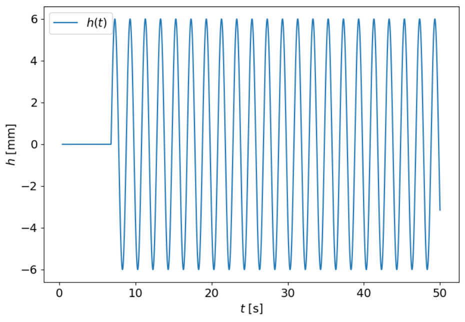
Figure 8.12: Permanent magnet 1 and permanent magnet 2

8.6.2. Learning Phase

As discussed in Section 8.4, the controller needs an initial learning phase to train the neural network before operating. The learning phase implemented for PM2 is 50 seconds with a sinusoidal pivot-point excitation. The amplitude is 6 mm and the frequency 0.50 Hz: $h(t) = 0.006\sin(2\pi 0.50t)$ (Figure 8.13b). The control output is set constant to $u(t) = -14.4$ V. The corresponding time series for the displacement is given in Figure 8.13a.



(a) Separation time series



(b) Pivot-point motion time series

Figure 8.13: Learning phase PM2

Applying equation 8.2 to the time series gives the estimates for the magnetic interaction. Feeding these data points to the network results in the fit presented in Figure 8.14.

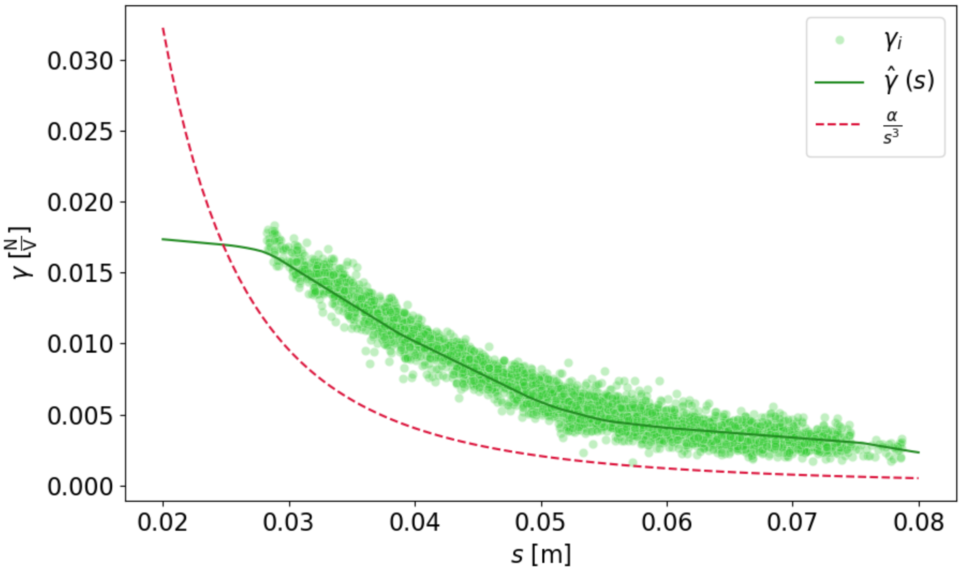


Figure 8.14: Learned relation by neural network

8.6.3. Model comparison

Figure 8.15 shows three different relations for the magnet interaction. First, the modelled $\frac{\alpha}{s^3}$ relation from the previous work of Atzampou et al. (2023) with $\alpha = 2.581 \cdot 10^{-7}$ Nm^3/V. Second, the fit of the neural network trained on the data from permanent magnet 1. Third, the neural network fit from the learning phase of permanent magnet 2.

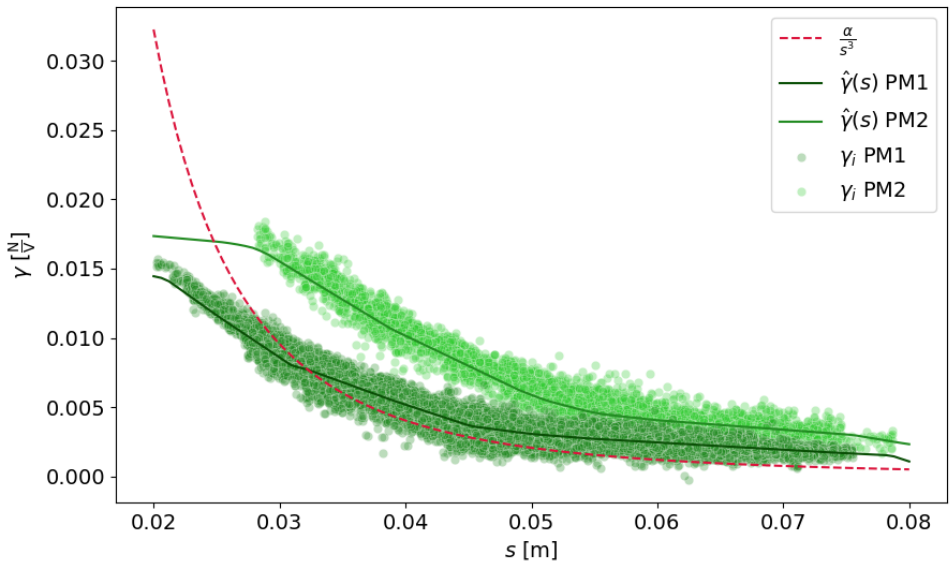
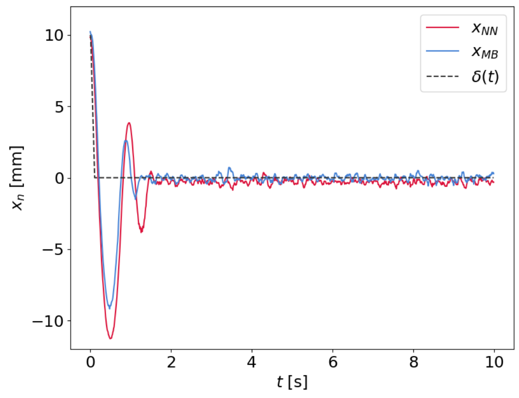
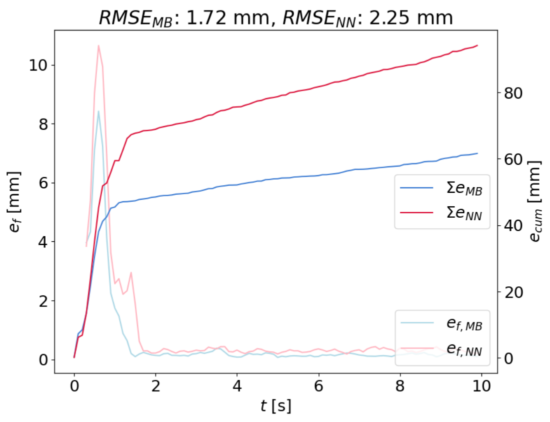


Figure 8.15: Magnetic interaction models

Figure 8.15 verifies that PM2 is indeed stronger than PM1. Furthermore, during the learning phase the separation distance remains greater compared to PM1, because the repelling force is larger. This is exemplified by no data points being present between $s = 0.02 - 0.03$ m for the PM2. This new relation is expected to achieve a better control performance than the modified PD controller, because the modified PD controller uses a model calibrated for PM1 and not PM2. To asses this the four test cases are performed again. This time with the PM2 mounted on the scale model. The result of the test cases can be seen in Figures 8.16 to Figure 8.19.

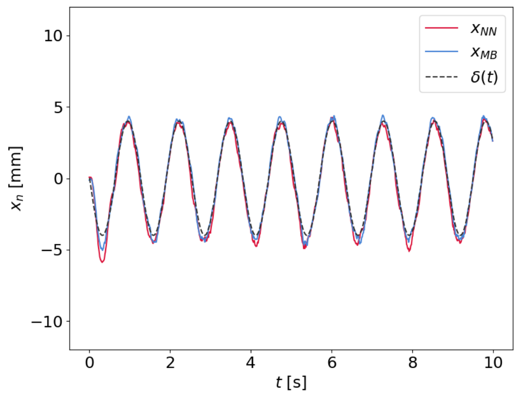


(a) Displacement time series

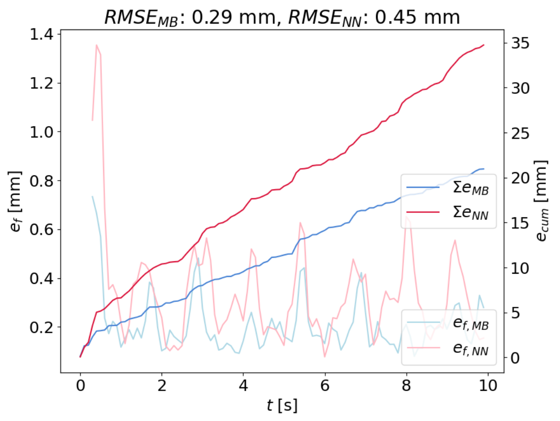


(b) Error time series

Figure 8.16: Test case 1, PM2

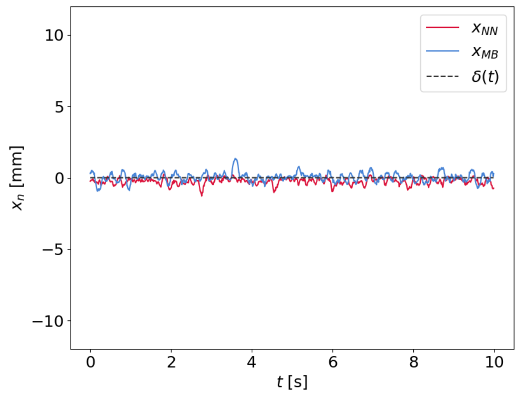


(a) Displacement time series

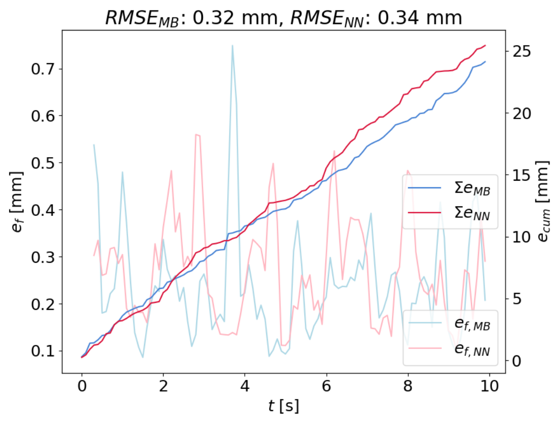


(b) Error time series

Figure 8.17: Test case 2, PM2

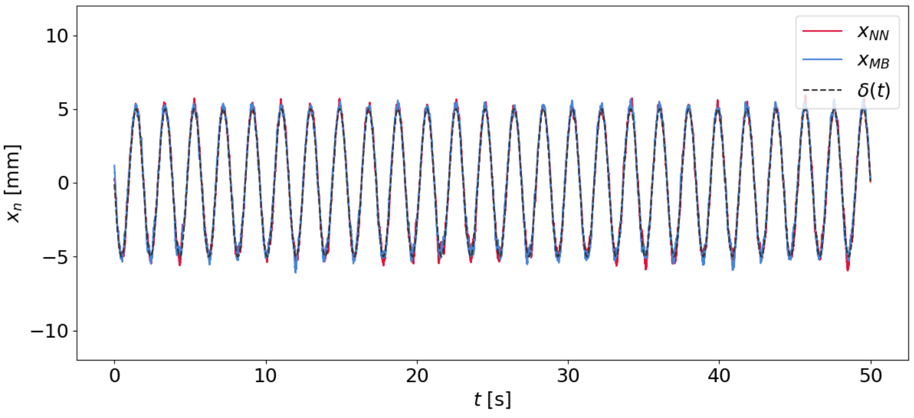


(a) Displacement time series

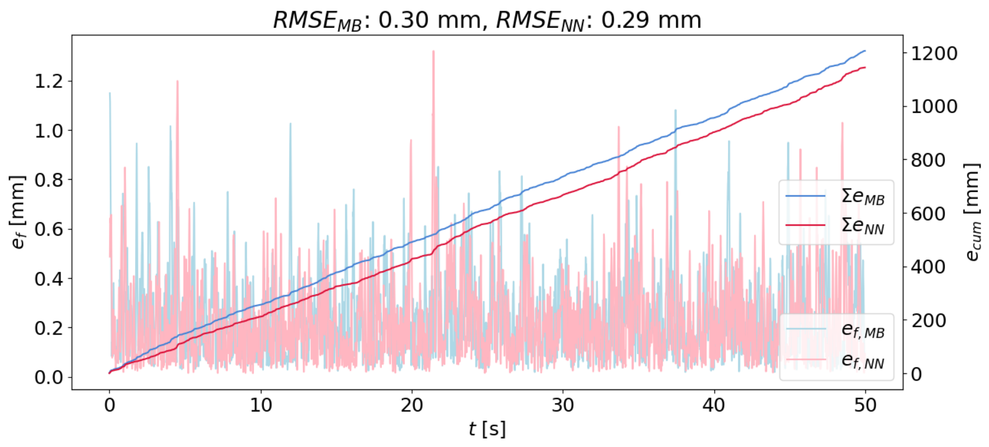


(b) Error time series

Figure 8.18: Test case 3, PM2



(a) Displacement time series



(b) Error time series

Figure 8.19: Test case 4, PM2

From the above Figures, it can be deduced that the control performance of the neural network gain scheduled PD is similar to the modified PD controller. A significant difference in performance was hypothesised, because the magnetic interaction relation is better approximated by the network than the model description, as shown in Figure 8.14. However, the test cases show no significant difference in control performance.

This is due to the order of the relation still being of the same order a cubic decay with distance. The difference is in the constant value α. However, due to the nature of the PD controller, the controller gains compensate for this change in the constant α. For example, if the real alpha α_{real} is twice as large as the modelled alpha value α_{modelled} . Then control is calculated using the modelled $\alpha_{modelled}$, according to equation 8.9. Substituting equation 8.8 into equation 8.9 gives equation 8.10. In which clearly can be seen that a modelling error in the α is similar to a correctly modelled α with different gains.

$$\alpha_{\text{real}} = 2 \cdot \alpha_{\text{modelled}} \tag{8.8}$$

$$u(t) = \frac{\alpha_{\text{modelled}}}{s^3}(K_p e + K_d \dot{e}) \tag{8.9}$$

$$u(t) = \frac{\alpha_{\text{real}}}{s^3}(\frac{K_p}{2} e + \frac{K_d}{2} \dot{e}) \tag{8.10}$$

Furthermore, the control output value is often the saturation limit. If this occurs both methods give exactly the same output, the saturation limit. Using the time series of the displacement of test case 1 and calculating the control output based on this time series for both controllers gives the time series of Figure 8.20. This Figure clearly shows that for the transient response the control is saturated and the control outputs are almost equal.

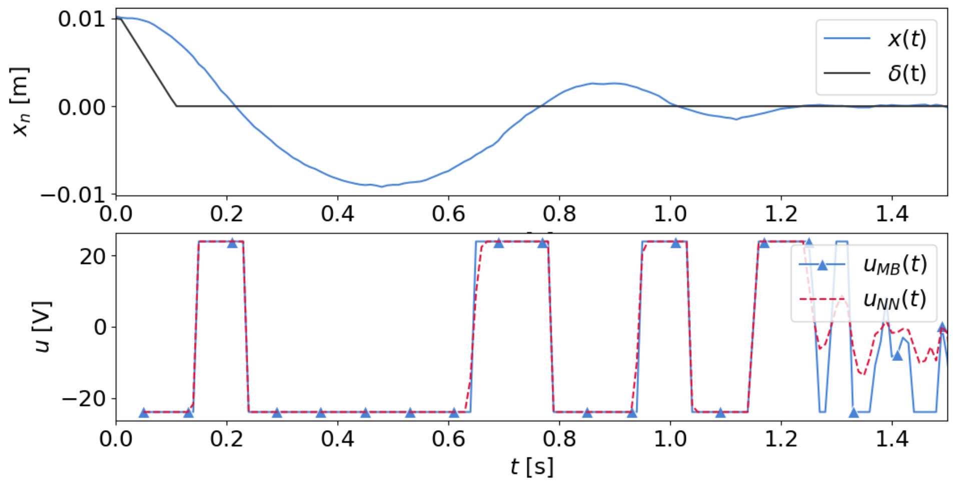


Figure 8.20: Simulated control output for modified PD and PD-FeNN

Significant differences in performance are expected to occur, if the power law of the separation relation changes and the control value is not saturated. This can for instance occur when multiple permanent magnets are placed on the payload and the separation is smaller.

8.7. Discussion

As mentioned in the introduction of the chapter the proposed PID-FeNN controller is not model free, however, it doesnot use experimentally determined parameters. This is still an improvement from the current state of the art controller of Atzampou et al. (2024), because it is more adaptable to changes in scale model and does not require manual fitting based on experimental data. The current controller, however, still has some limitations that are elaborated on in this section.

8.7.1. External Forcing

In section 8.1 **Assumption 1** reads: The system can be described by the equations of a controlled linear unforced undamped pendulum. The assumption that the pendulum is linear is a valid assumption, because displacements are small compared to the length of the pendulum as explained in Section 2. This is the case for both the physical model and the evntual practical application. The assumption of unforced is, however, not applicable for the practical application. In the physical model, there is no external forcing, because there is for example no wind disturbance inside the laboratory. In the practical application windforces will undoubtedly affect the system and cause oscillations of the payload. This is challenging, because this will influence the estimation of the magnetic interaction term. This is shown in equation 8.11 by the added unknown force term. This can be tackled by either knowing the external forcing (for instance through modelling of the windforces combined with real time measurements) or by neglecting it, which is valid if the external forcing is small compared to the inertia, restoring force and magnetic interaction. The same holds for assuming an undamped system, however, since the damping is expected to be small this can be circumvented by neglecting it.

$$M\ddot{x} + Kx = \gamma(\cdot)u \Rightarrow M\ddot{x} + Kx = \gamma(\cdot)u + F_{ext} \tag{8.11a}$$

$$\gamma_i = \frac{M\hat{\ddot{x}}_i + Kx_i}{u_i} \Rightarrow \gamma_i = \frac{M\hat{\ddot{x}}_i + Kx_i - F_{ext}}{u_i} \tag{8.11b}$$

8.7.2. Extrapolation

The proposed controller is only able to control the payload within separation values contained in the dataset used to train the network, because feedforward neural networks with ReLU activation functions tend to converge to linear functions when extrapolating (Xu et al., 2021). This behaviour of the network is shown in Figure 8.21. The predictions for the interaction make a bend just outside the domain of the dataset and further away from the dataset the predicted relation tends to a linear relation. This linear relation outside the training-set domain does not have any physical meaning and even becomes negative, due to the use of the Leaky ReLU activation function. These negative values could be tackled by using the ReLU function. This is, however, not done as during implementation this occasionally lead to dead neurons after initialization, which in turn lead to no convergence.

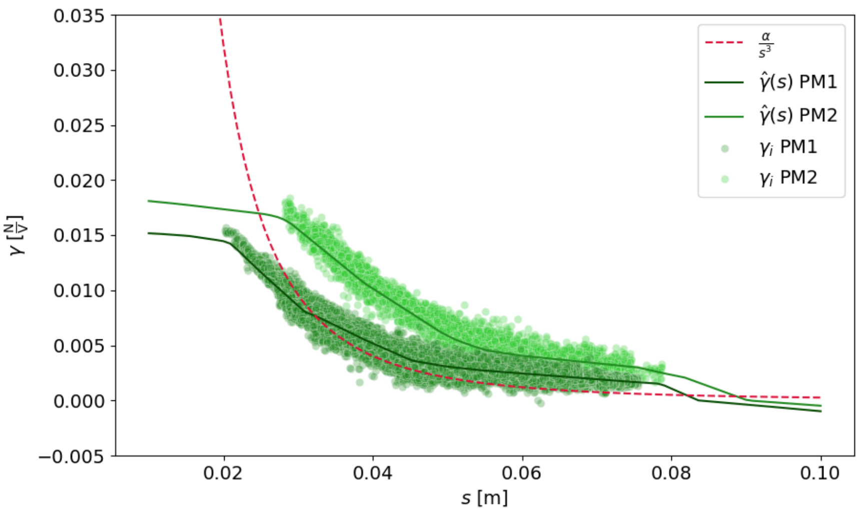
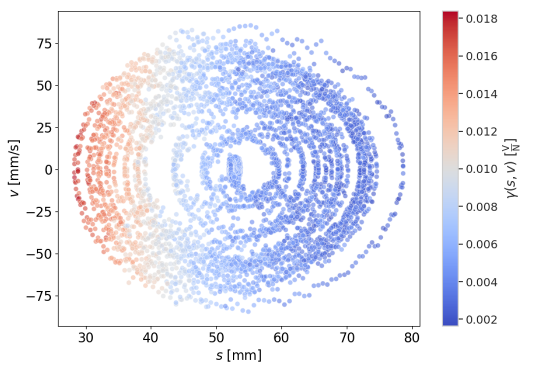


Figure 8.21: Prediction values outside of the domain of the dataset

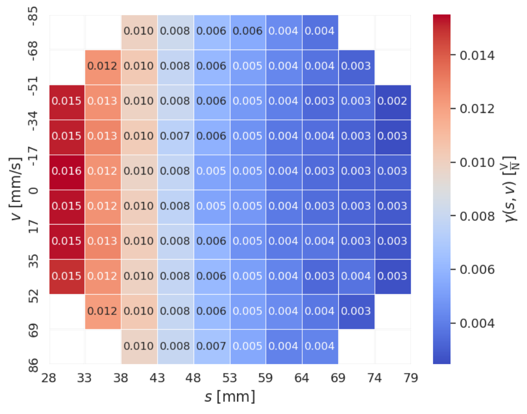
8.7.3. Interaction Dependencies

During this research it has been assumed that the magnetic interaction force is only a function of the distance between the electromagnet and the permanent magnet. For the 2D-dynamical system this means it is only dependent on s and not on velocity $\dot{x}$. This has been based on previous research from Atzampou et al. (2024) and is explained in Section 2.2.2.

To verify that velocity does not influence the magnetic interaction. The velocity is plotted against the distance, with the magnetic interaction term γ visualized with color. This both done per data point in a scatterplot, Figure 8.22a, and averaged out in a heatmap , Figure 8.22b. The value of gamma γ only changes along the s axis and not when moving along the v axis. Thus indeed the magnetic interaction is independent of velocity of the payload.



(a) Scatterplot of magnetic interaction data points for s and v



(b) Heatmap of magnetic interaction for s and v

As proven, only the separation s determines the magnetic interaction for the 2D pendulum. However, a 3D pendulum with a rigid body payload has more degrees of freedom. This means also the displacement in y direction and the rotation of the payload will influence the magnetic interaction, by changing the distance between the magnets and disaligning the magnetic dipoles. This could be tackled by the PD-FeNN by adding more nodes in the input layer and collect a wider range of data, that also includes lateral displacement and rotation.

9
Conclusion

The objective of this research was to design a model-free control algorithm for an electromagnetic actuator to achieve positional control of a suspended load. The accompanying research question was: "Which model-free control algorithms allow for positional control of a suspended load, while adapting control parameters based on measurement data, without requiring manual tuning?". To answer this question three subquestions were posed, based on which the question is answered.

1. *What is a model-free controller?*

A model-free controller is in this thesis defined as: a controller that controls a dynamical system without requiring a model description of the system. It does not use information from the equations of motion or state-space equations to calculate control output or determine the structure of the controller. This type of controller is called a direct data-driven controller. Differing from indirect data-driven controllers by not identifying the system being controlled based on data. Instead direct data-driven controllers calculate the control output directly from measured input and output data.

2. *Which model-free control algorithms exist?*

A multitude of direct data-driven control algorithms are identified through literature research. An overview is given that contained: Simultaneous perturbation stochastic approximation (SPSA), Proportional–integral–derivative controller (PID), Model-free adaptive control (MFAC), Unclassified control (UC), Iterative feedback tuning (IFT), Virtual reference . feedback tuning (VRFT), Incremental learning controller (ILC), and Lazy learning (LL). Outside of this reference, additional methods are identified such as: Q-learning, Data enabled predictive control (DEePC), and intelligent PID's (iPID). This is an extensive list of algorithms, however, the field of data driven control is even larger and should not be limited to the algorithms discussed in this thesis.

3. *What are the most important limitations for model free algorithms?*

The first important limitation is that a large section of data-driven methods are only suitable for linear time invariant systems (LTI) . Examples of controllers only suitable for LTI systems are: PID, IFT, and VRFT. The second limitation is that many control algorithms require offline data to be properly tuned before operation and do not adjust online during operation, such as ILC. The third and most important limitation is the assumption of many methods that the state and time dependence of the system can be neglected. They assume the change of the state of the system can be represented as a function of the control output only. This is valid if the control output dominates compared to the dynamics of the system. The subsequent aim of such a controller is to estimate the gradient of the state with respect to control output and find some optimal control value that will reach the desired state. This approach is not suitable for the electromagnetic pendulum, because the dynamics dominate the control instead of the other way around. The important takeaway is that model-free controllers are not independent of the system being controlled, although, they do not rely on a model description of the system. Knowledge on the behaviour and type of system certainly is important, because the system that is aimed to be controlled must satisfy certain assumptions for a specific controller to function. An example of these assumptions are linearity or being dominated by the control output.

Thsese subquestions lead to the answer of the research question:

Which model-free control algorithms allow for positional control of a suspended load, while adapting control parameters based on measurement data, without requiring manual tuning?

Given the aforementioned limitations of model-free control algorithms, this research found no model-free controllers that are capable of positional control of the suspended load. The researched controllers were invalidated, because they either assumed a linear system or assumed the dynamics to be negligible compared to the response to the control output. To still improve on the state of the art controller, the requirement of model-free is relaxed and a controller is designed and proposed by this thesis.

The main motivation for a model-free controller was the difficulty of modelling the response to the control output. The proposed controller, therefore, is still model free with respect to the response to control output. However, it uses the model of a linear undamped pendulum for the dynamics. The function relating the response to control output is approximated by a neural network. The accompanying control law is a modified PD controller. The PD controller uses the inverse of the output of the neural network to create state dependent gains and take into account the non linearity of the magnetic interaction. This controller is named: PD-FeNN.

The requirements for the proposed controller were determined to be:

R1: The controller makes the payload follow a predefined reference trajectory with the allowance of a small error.
R2: The controller is model-free .
R3: The controller adjusts its parameters based on measurement data.
R4: The controller does not require the manually tuning of parameters when system parameters are altered.

The proposed controller satisfies requirement 1, because it is shown through multiple test cases to be capable of positional control. The proposed controller does not completely satisfy requirement 2, because it uses model information of the dynamics. However, relaxing this requirement is supported by the findings of this research. Requirement 3 is satisfied through updating the trainable parameters of the neural network, the controller adjusts the state dependency of the gains. The controller satisfies requirement 4, because it does not need tuning of parameters when system parameters alter. However, it does require knowledge of the inertia and restoring force of the system.

The final 2 subquestions are about the properties of the developed controller.

4: What is the performance of the proposed control algorithm

The performance of the proposed control is assessed using the root mean squared error metric and by comparison with the modified PD controller from Atzampou et al. (2024). The proposed controller and modified PD controller have an equal RMSE for all test cases, given a range of ±0.2 mm. This range is caused by possible misalignment in initialization and measurement error. The RMSE is for both controller primarily dominated by the transient response and the transient response is thus comparable. The steady-state RMSE is for both controllers approximately 0.30 mm.

5: How generalizable is the proposed controller?

The proposed controller uses model information from the dynamics of the system. It is specifically designed for systems described by the model information used. The system must satisfy assumption 1: "The system can be described by the equations of a linear unforced undamped pendulum" and assumption 2: "The system is controlled by a magnetic interaction force that scales linear with voltage.". The control is generalizable to any system that satisfies the assumptions and is thus not dependent on the value of the physical parameters. This approach can also be further generalized and expanded to 3D pendulums having three inputs into the neural network x-, y-distance and axial rotation.

10 Recommendations

In this chapter the recommendations for future work and research are discussed. The control performance of the available controllers of the scale model are very adequate. The main recommendation is, therefore, to focus on increasing complexity in the scale model and taking steps towards getting this novel approach to be a valid market solution for the installation of offshore wind turbines. The recommendations are split in 3 main directions: direction 1: practical application, direction 2: model-based controller of complex scale model, and direction 3: implementation of model-free approaches not adequately research in this thesis.

10.1. Practical application

The current research has mainly been focused on achieving control over a scalel model of the mating operation using electromagnets. However, outside of a scale comparison in "Motion Control of a Pendulum via Magnetic Interaction" by Atzampou et al. (2024), no research has been done towards implementation of this method. I recommend to do research into the economical case and the feasibility of the approach, given a control algorithm can be designed. Research topics of interest could be the magnitude of the force required for control, the required size and strength of the (electro)magnets, and feasible separation of the magnets in practice. Conclusions from this research can lead to further research on implementation of the method: mounting magnets on the ships, operation of these magnets and how shipmovement effect magnet positioning. The merit of this research direction is based on two benefits. The first is that it can prove or disprove the economic and perhaps potential safety benefits of the innovation. If proven to reduce the installation cost, this would be a solid ground for more funding and research into the innovation. Second, it can shed light on practical problems. These can subsequently already be accounted for during development of the control algorithm and help create better scale models. This allows for an interplay between practical implementation and theoretical development.

10.2. Model Approach

The proposed controller and modified PD controller perform well for the scale model, however, this model is quite simplified compared to the practical application. Modelling the wind turbine tower as a point mass is quite crude, modeling as a rigid body and including the degree of freedom of rotation, would already be an improvement, Figure 10.1. Furthermore, the current scale model is 2D, while the practical application is 3D introducing more degrees of freedom. I would advise to deal with this by incrementally increasing the complexity of the scale model and accompanying this, develop a model for the dynamics. This approach is preferred over aiming to find a model-free controller for the complex system. Because the more system knowledge is available, the easier designing and tuning

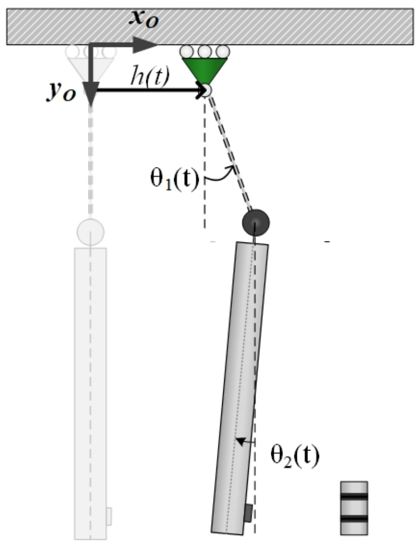


Figure 10.1: Model representation of the suspended load as a rigid body

a controller becomes. Model based controllers have the advantage of interpretability. If terms have physical meaning, they can be understood and modified. Data-driven or model-free algorithms behave more like black boxes that often lack this interpretability, leading to trial and error approaches.

10.3. Model-free approach

The final recommended direction for further research is into model-free controllers, who were not properly investigated in this thesis. These model-free control methods were not investigated, because they were deemed to difficult to understand and/or implement given the time frame and scope of this research. This direction should only be taken if modelling becomes infeasible and not vice-versa. It is also recommended to ensure a good prerequisite knowledge and understanding of controlled systems and mathematics when investigating these methods. Two model-free methods are identified, that could be capable of controlling the system: DEePc and reinforcement learning methods such as Deep Q-Networks (DQN).

DEePc stands for Data enabled predictive control and is a direct data-driven control framework. It uses past input and output data to make future decisions without an explicit parametric model (van Wingerden et al., 2022). It is similar to Model Predictive Cotnrol (MPC) and stands as the data driven counterpart. The algorithm originates form the fundamental lemma that all future input-output trajectories of a linear system are parameterized by a sufficiently excited past input-output trajectory (van Wingerden et al., 2022). In principal DEePC only applies to Linear Time Invariant Systems (LTI) . However, if the non linear system can be written in linear parameter varying form (LPV) this method can also be applied to non linear system as done by Verhoek et al. (2021). Downsides of this approach are mainly the large complexity of the control method and the requirement that the system can be written in LPV form, which might not be the case in the practical application or for a more complex scale model.

Reinforcement learning (RL) is a model-free framework for solving optimal control problems stated as Markov decision processes (MDPs) (Buşoniu et al., 2018). A Markov decision process is a process with discrete time and discrete state space that has the Markov property. This property is that the next state only depends on the current state and not on past states (Rocca, 2021). Reinforcement learning is based on an agent trying to find an action in the current state that maximizes a numerical reward value. This reward value is calculated based on the changes in the environment, Figure 10.2. This can be done gradient based or by policy iteration (gradient free) (Steve Brunton, 2022).

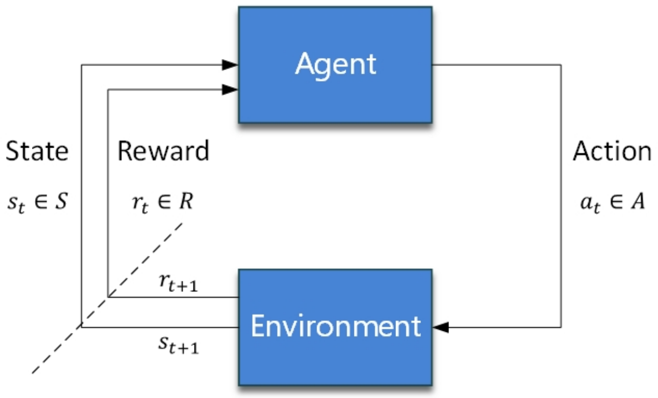


Figure 10.2: Agent, environment interaction in RL

Possible downside of this approach is the vast amount of data required to train these controllers. Currently the main use of reinforcement learning is in computer simulation, however, simulation can not be done without an available model. If there is no model available, experimental data must be used for training. Accumulating this data to have information on every possible action-state pair is, however, expected to be very difficult. Furthermore, reinforcement learning is mainly meant for discrete systems, while the system of interest is continuous with both a continuous state space and a continuous action space. Application of reinforcement learning would require these spaces to be discretized. Finally, the time delay between control output and electromagnetic field change could invalidate the Markov property of only the current state determining the next state.

References

Åstrom, K. J., & Murray, R. (2009, February). *Feedback Systems* (v2.10b). PRINCETON UNIVERSITY PRESS.

Atzampou, P., Meijers, P., Tsouvalas, A., & Metrikine, A. (2023). Magnetic control of a simple pendulum with a moving pivot point.

Atzampou, P., Meijers, P. C., Tsouvalas, A., & Metrikine, A. V. (2024). Contactless control of suspended loads for offshore installations: Proof of concept using magnetic interaction. *Journal of Sound and Vibration*, *575*, 118246. https://doi.org/10.1016/j.jsv.2024.118246

Blom, R. (2022). Allowable Wind Turbine Set-down Impact. Retrieved June 20, 2024, from https://repository.tudelft.nl/islandora/object/uuid%3Aa3e14327-0d64-46b7-9730-0d1803cd7e9c

Bontempi, G., Birattari, M., & Bersini, H. (2010). Lazy learning for modeling and control design. *International Journal of Control*, *72*, 643–658. https://doi.org/10.1080/002071799220830

Brodtman, Z. (2021, November). The Importance and Reasoning behind Activation Functions. Retrieved March 7, 2024, from https://towardsdatascience.com/the-importance-and-reasoning-behind-activation-functions-4dc00e74db41

Bron, T. (2022). Model-free Positioning Control of a Payload by use of Wires.

Brownlee, J. (2021, February). Weight Initialization for Deep Learning Neural Networks. Retrieved March 5, 2024, from https://machinelearningmastery.com/weight-initialization-for-deep-learning-neural-networks/

Brownlee, J. (2022, August). Why Initialize a Neural Network with Random Weights? Retrieved March 5, 2024, from https://machinelearningmastery.com/why-initialize-a-neural-network-with-random-weights/

Buşoniu, L., de Bruin, T., Tolić, D., Kober, J., & Palunko, I. (2018). Reinforcement learning for control: Performance, stability, and deep approximators. *Annual Reviews in Control*, *46*, 8–28. https://doi.org/10.1016/j.arcontrol.2018.09.005

deep-mind. (2023, March). The Universal Approximation Theorem –. Retrieved July 1, 2024, from https://www.deep-mind.org/2023/03/26/the-universal-approximation-theorem/

Dong, E., Guo, S., Lin, X., Li, X., & Wang, Y. (2012). A neural network-based self-tuning pid controller of an autonomous underwater vehicle. *2012 IEEE International Conference on Mechatronics and Automation*, 898–903. https://doi.org/10.1109/ICMA.2012.6283262

Doshi, S. (2020, August). Various Optimization Algorithms For Training Neural Network. Retrieved May 1, 2024, from https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6

Equinor. (2020). Mating: Saipem 7000 lifts Equinor's Hywind turbines onto its floating spar foundation. https://www.upstreamonline.com/energy-transition/oil-and-gas-contracting-stalwarts-find-new-path-in-renewables/2-1-908780

Fliess, M., & Join, C. (2013a). Model-free control. *International Journal of Control*, *86*(12), 2228–2252. https://doi.org/10.1080/00207179.2013.810345

Fliess, M., & Join, C. (2013b). Model-free control. *International Journal of Control*, *86*. https://doi.org/10.1080/00207179.2013.810345

Goodwin, G. C., & Sin, K. S. (1984). *Adaptive filtering prediction and control*. Prentice-Hall Englewood Cliffs, N.J. http://www.freading.com/ebooks/details/r:download/MDAwMDE4LTgzNTg4NjE=

Griffiths, D. J. (2017, June). Introduction to Electrodynamics [ISBN: 9781108333511 Publisher: Cambridge University Press]. https://doi.org/10.1017/9781108333511

Hanna, Y., Khater, A., El-Nagar, A., & el Bardini, M. (2022). Polynomial Recurrent Neural Network-Based Adaptive PID Controller With Stable Learning Algorithm. *Neural Processing Letters*, *55*, 1–26. https://doi.org/10.1007/s11063-022-10989-1

He, K., Zhang, X., Ren, S., & Sun, J. (2015, February). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification [arXiv:1502.01852 [cs]]. https://doi.org/10.48550/arXiv.1502.01852

Hernández-Alvarado, R., García-Valdovinos, L. G., Salgado-Jiménez, T., Gómez-Espinosa, A., & Fonseca-Navarro, F. (2016). Neural Network-Based Self-Tuning PID Control for Underwater Vehicles [Number: 9 Publisher: Multidisciplinary Digital Publishing Institute]. *Sensors*, *16*(9), 1429. https://doi.org/10.3390/s16091429

Hou, Z., & Huang, W. (1997). The model-free learning adaptive control of a class of siso nonlinear systems. *Proceedings of the 1997 American Control Conference (Cat. No.97CH36041)*, *1*, 343–344 vol.1. https://doi.org/10.1109/ACC.1997.611815

Hou, Z., & Jin, S. (2008). Model-free adaptive control for a class of nonlinear discrete-time systems based on the partial form linearization [17th IFAC World Congress]. *IFAC Proceedings Volumes*, *41*(2), 3509–3514. https://doi.org/https://doi.org/10.3182/20080706-5-KR-1001.00593

Hou, Z.-S., & Wang, Z. (2013). From model-based control to data-driven control: Survey, classification and perspective [Data-based Control, Decision, Scheduling and Fault Diagnostics]. *Information Sciences*, *235*, 3–35. https://doi.org/https://doi.org/10.1016/j.ins.2012.07.014

IEA. (2022). Renewables 2022. https://www.iea.org/reports/offshore-wind-outlook-2019

Jiang, Z. (2021). Installation of offshore wind turbines: A technical review. *Renewable and Sustainable Energy Reviews*, *139*, 110576. https://doi.org/10.1016/j.rser.2020.110576

Khan Academy. (n.d.). What are inductors? (self-inductance) (video). Retrieved July 4, 2024, from https://www.khanacademy.org/science/electromagnetism/x4352f0cb3cc997f5:how-wireless-charging-and-transformers-work/x4352f0cb3cc997f5:inertia-of-charging/v/what-are-inductors-self-inductance

Kingma, D., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*.

Meijers, P., Atzampou, P., & Metrikine, A. (2023). Experimental and numerical study of a magnetic pendulum.

N. J. Killingsworth & M. Krstic. (2006). PID tuning using extremum seeking: Online, model-free performance optimization. *IEEE Control Systems Magazine*, *26*(1), 70–79. https://doi.org/10.1109/MCS.2006.1580155

Nguyen, B. K., & Ahn, K. K. (2006). Position control of shape memory alloy actuators by using self tuning fuzzy pid controller. *International Journal of Control, Automation and Systems*, *4*, 1–5. https://doi.org/10.1109/ICIEA.2006.257198

Nielsen, M. A. (2015). Neural Networks and Deep Learning [Publisher: Determination Press]. Retrieved February 20, 2024, from http://neuralnetworksanddeeplearning.com

Oh, S.-K., Pedrycz, W., & Park, B.-J. (2003). Polynomial neural networks architecture: Analysis and design. *Computers & Electrical Engineering*, *29*(6), 703–725. https://doi.org/10.1016/S0045-7906(02)00045-9

Pan, T., Li, S., & Cai, W. (2006). Lazy Learning-Based Online Identification and Adaptive PID Control: A case study for CSTR process. *Industrial & Engineering Chemistry Research*, *46*(2), 472–480. https://doi.org/10.1021/ie0608713

Patrikar, A., & Provence, J. (1996). Nonlinear system identification and adaptive control using polynomial networks. *Mathematical and Computer Modelling*, *23*(1), 159–173. https://doi.org/10.1016/0895-7177(95)00225-1

Ponce, A. N., Behar, A. A., Hernández, A. O., & Sitar, V. R. (2004). Neural Networks for Self-tuning Control Systems [Number: 1]. *Acta Polytechnica*, *44*(1). https://doi.org/10.14311/514

Rajan, S. (2020, July). Cost functions for Regression and its Optimization Techniques in Machine Learning. Retrieved April 30, 2024, from https://towardsdatascience.com/cost-functions-of-regression-and-its-optimization-techniques-in-machine-learning-2f5931cd33f1

Ren, Z., Verma, A., Ataei, B., Halse, K., & Hildre, H. (2021). Model-free anti-swing control of complex-shaped payload with offshore floating cranes and a large number of lift wires. https://doi.org/10.1016/j.oceaneng.2021.108868

Roberts, E. (n.d.). Neural Networks - History. Retrieved February 20, 2024, from https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/History/history1.html

Rocca, J. (2021, March). Introduction to Markov chains. Retrieved July 7, 2024, from https://towardsdatascience.com/brief-introduction-to-markov-chains-2c8cab9c98ab

Sanghvirajit. (2024, February). A Complete Guide to Adam and RMSprop Optimizer. Retrieved May 1, 2024, from https://medium.com/analytics-vidhya/a-complete-guide-to-adam-and-rmsprop-optimizer-75f4502d83be

Savitzky, A., & Golay, M. J. E. (1964). Smoothing and Differentiation of Data by Simplified Least Squares Procedures. [Publisher: American Chemical Society]. *Analytical Chemistry*, *36*(8), 1627–1639. https://doi.org/10.1021/ac60214a047

Scholar, G. (n.d.). Overview works of Prof. Zhongsheng Hou. https://scholar.google.com/citations?view_op=list_works&hl=en&hl=en&user=z0Pg-AwAAAAJ

Stack Exchange. (2016, March). Answer to "Why does expressing calculations as matrix multiplications make them faster?". Retrieved July 1, 2024, from https://softwareengineering.stackexchange.com/a/312448

Steve Brunton. (2018, June). Data-Driven Control: Linear System Identification. Retrieved June 27, 2024, from https://www.youtube.com/watch?v=6F2YVsT9dOs

Steve Brunton. (2022, January). Q-Learning: Model Free Reinforcement Learning and Temporal Difference Learning. Retrieved June 27, 2024, from https://www.youtube.com/watch?v=0iqz4tcKN58

Tan, K. K., Lee, T. H., Huang, S. N., & Leu, F. M. (2001). Adaptive-Predictive Control of a Class of SISO Nonlinear Systems. *Dynamics and Control*, *11*(2), 151–174. https://doi.org/10.1023/A:1012583811904

TensorFlow. (n.d.). TensorFlow Lite Guide. Retrieved May 23, 2024, from https://www.tensorflow.org/lite/guide

TU Delft. (2021). TU Delft on board the world largest crane vessel for exploring future Offshore Wind Turbines. Retrieved June 20, 2024, from https://www.tudelft.nl/2021/3me/oktober/tu-delft-on-board-the-world-largest-crane-vessel-for-exploring-future-offshore-wind-turbines

van Wingerden, J.-W., Mulders, S. P., Dinkla, R., Oomen, T., & Verhaegen, M. (2022). Data-enabled predictive control with instrumental variables: The direct equivalence with subspace predictive control [ISSN: 2576-2370]. *2022 IEEE 61st Conference on Decision and Control (CDC)*, 2111–2116. https://doi.org/10.1109/CDC51059.2022.9992824

Verhoek, C., Abbas, H. S., Tóth, R., & Haesaert, S. (2021). Data-Driven Predictive Control for Linear Parameter-Varying Systems*. *IFAC-PapersOnLine*, *54*(8), 101–108. https://doi.org/10.1016/j.ifacol.2021.08.588

Xu, K., Zhang, M., Li, J., Du, S. S., Kawarabayashi, K.-i., & Jegelka, S. (2021, March). How Neural Networks Extrapolate: From Feedforward to Graph Neural Networks [arXiv:2009.11848 [cs, stat]]. https://doi.org/10.48550/arXiv.2009.11848

Yang, Y., Chen, C., & Lu, J. (2021). An Improved Partial-Form MFAC Design for Discrete-Time Nonlinear Systems with Neural Networks. *IEEE Access*, *PP*, 1–1. https://doi.org/10.1109/ACCESS.2021.3065311

Zhang, J., Zhuang, J., Du, H., & Wang, S. (2009). Self-organizing genetic algorithm based tuning of pid controllers. *Information Sciences*, *179*(7), 1007–1018. https://doi.org/https://doi.org/10.1016/j.ins.2008.11.038

Zhongsheng Hou & Wenhu Huang. (1997). The model-free learning adaptive control of a class of SISO nonlinear systems [Journal Abbreviation: Proceedings of the 1997 American Control Conference (Cat. No.97CH36041)]. *Proceedings of the 1997 American Control Conference (Cat. No.97CH36041)*, *1*, 343–344 vol.1. https://doi.org/10.1109/ACC.1997.611815

A
Discrete form

The microprocessor of the scale model works with discrete values. Only an analog processor is in theory able to output an continuous signal. Many control algorithms are, therefore, written in a discrete form. To simulate these control algorithms a discrete model of the system is required.

A.1. Discretizing with finite differences

To simplify discretization and ensure stability of the numerical method applied the equation of motion is first linearized. The damping term $\mu \operatorname{sgn} \dot{x}$ is neglected and the control output is considered to effect the system linearly through a constant linear coëfficient a:

$$M_e \ddot{x}(t) + K_e x(t) = -(M + \frac{m}{2})\ddot{h}(t) + au(t). \tag{A.1}$$

Excitation through pivot-point motion will be represented as a time dependent external force:

$$F(t) = -(M + \frac{m}{2})\ddot{h}(t), \tag{A.2}$$

This leads to the following representation of the system

$$M_e \ddot{x}(t) + \ K_e x(t) = F(t) + au(t), \tag{A.3}$$

dividing by M_e gives the canonical form:

$$\ddot{x}(t) + \frac{K_e}{M_e} x(t) = \frac{F(t)}{M_e} + \frac{a}{M_e} u(t). \tag{A.4}$$

This equation is discretized using central differences $\left(\frac{d^2y}{dx^2} \approx \frac{y(k+1)-2y(k)+y(k-1)}{\Delta t^x}\right)$):

$$\frac{x(k+1) - 2x(k) + x(k-1)}{\Delta t^2} + \frac{K_e}{M_e} x(k) = \frac{F(k)}{M_e} + \frac{a}{M_e} u(k), \tag{A.5}$$

rewriting to place the next time step on the left hand side gives:

$$x(k+1) = 2x(k) - x(k-1) + \frac{\Delta t^2}{M_e}\Big(-K_e x(k) + au(k) + F(k)\Big) \tag{A.6}$$

This equation A.6 is the linear discrete form of the dynamical model and is used to simulate test cases with discrete control algorithms.

B
Programming Code

This thesis contained a lot of coding, for simulation, control of the scale model, and displaying results. The code used for simulation and control will be shared on a public GitHub. As of writing this can be found ***here***. However, I expect it might migrate to another place, in which case you will have to take up contact to acces it. The remainder of this appendix elaborates on the implementation of the simulations and control algorithms.

B.1. Numerical Model

There are two numerical models of the system. One model is the discrete linear model used to simulate the MFAC and PWORNN control. This linear model uses central differences to solve the initial value problem. The other model is the full non-linear model and uses Runga-Kutta 4(5) as a solver.

B.1.1. Linear Model

The linear model was implemented to have a first assessment of control algorithms. The advantages of the linear model is that it could be solved by a implemention of central differences of my self, given small enough time steps. Instead of having to use the SciPy library. This allowed for more control over the calculation of the control values and storing of the results.

B.1.2. Non-Linear Model

The non-linear model is a full implementation of the modelled equations of motions of the system (equation 2.8). Initially a solver from the SciPy package: scipy.integrate.solve_ivp is used. This is an implementation of the RK4(5) algorithm. Later to allow for online learning a custom Forward Euler implementation has been made. The non-linear model is used to simulate the behaviour of the system and extract data points for offline learning of the PID-FeNN controller from chapter 8.

B.2. Neural networks

TensorFlow is used to create and set up the neural networks used in this thesis. This library is chosen for the ability to convert models to TensorFlow Lite models that can be run on microprocessors. Four iterations of networks have been made. First iteration: offline learning mimicking model based controller ($C = \frac{1}{N}\sum_{i=1}^{N}(u_i - \hat{u}_i)^2$). Second iteration: online learning mimicking model based controller ($C = \frac{1}{N}\sum_{i=1}^{N}(u_i - \hat{u}_i)^2$). Third iteration online learning, but cost function based on state error ($C = \frac{1}{N}\sum_{i=1}^{N}(\delta_i - x_i)^2$). This implementation did not allow for computing gradients using the TensorFlow function. This would have required manual implementation of backpropagation algorithm as explained in section 4.6.The fourth implementation: online learning of interaction, using estimation of the interaction term to calculate the loss value ($C = \frac{1}{N}\sum_{i=1}^{N}(\gamma_i - \hat{\gamma}_i)^2$). This implementation was successful and lead to good results.

B.2.1. TensorFlow model conversion

This section is dedicated to the steps required to convert a trained TensorFlow model into a C-Array so it can be used for feedforward (inference) on a micro-controller. After training the algorithm using the online implementation from section B.1.2 or based on data collected according to section 8.4. The model can be saved using:

```
tf.saved_model.save(model, model_path)
```

This stores the network with trained weights in a folder specified by model_path. This folder can however not be loaded into a microcontroller. The network needs to be converted to a '.tflite file'. This can be by creating a instance of the TFLiteConverter class from the tf.lite library loading in the folder where you saved the trained model.. Then wrtiting the converted model to a file with the '.tflite' extension.

```
converter = tf.lite.TFLiteConverter.from_saved_model(model_path)
tflite_model = converter.convert()

# Save the model.
with open(tflite_path, 'wb') as f:
    f.write(tflite_model)
```

This type is optimized for on-device machine learning. It supports multiple language including C++ and can be run on number of devices including microcontrollers (TensorFlow, n.d.). To run the network on a microcontroller the .tflite file must first be converted to a C-byte array. After which it can be stored in read-only memory on the controller and be used for inference using the C++ library of tensorflow micro. To achieve the conversion one can use the -xxd function from linux or write a custom python code.

B.2.2. Microcontroller implementation

This section is dedicated to the steps required to run inference of a neural network on a microcontroller and specifically the Teensy 3.6. After converting the model to a C-byte array, this C-byte array must be added to the Arduino project folder of your microcontroller. After this you will also need the micro repository from TensorFlow on github as part of your project. You need the error reporter and microinterpreter class to create an instance of the microinterpreter. This class allows you to do a forward pass. Creating a class and for the microinterpret can be quite challenging, since TensorFlow built in hardware specific checks, thus, I recommend using a wrapper from GitHub that gets rid of these checks.