Master of Science Thesis in Electrical Engineering
Department of Electrical Engineering, Linköping University, 2023

Neural Network Based Control Design for a Unicycle System

Axel Ek

Master of Science Thesis in Electrical Engineering

Neural Network Based Control Design for a Unicycle System

Axel Ek

LiTH-ISY-EX–23/5552–SE

Supervisor: **Abhishek Dhar**
ISY, Linköpings universitet
Filip Petersson
Combine

Examiner: **Daniel Axehill**
ISY, Linköpings universitet

Division of Automatic Control
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

Abstract

Physics-based models of dynamical systems can take a lot of time and be hard to derive, and there will always be some effect that was not added to the calculations, like aerodynamic-, gyroscopic- or frictional- effects. Calculating all these effects takes time and a lot of knowledge of the system dynamics. There are many different ways to implement different methods to speed up the process of creating the control policy. What is the simplest way to create a control policy and how does it control the system?

Neural networks is a promising approach, where there are two different methods. First, by using the mathematical structure of a neural network a model of the system can be derived, and then a simple control policy is used. Second, Reinforcement learning is where the control policy is learned. These two are compared to a baseline model where the model of the system is derived from the physical description of the system.

First, the model is calculated by the system dynamics with classical mechanics that describes the mathematical description of a physics-based system. Then the machine-learning approach of using a neural network to learn and describe the system is implemented. Lastly, the Reinforcement learning method is made and compared to the other models.

The models had all their own differences in performance. The controllers based on the physics-based model were good in a small region around the equilibrium and it took a long time to derive. The neural network models were more general and easier to implement but were more unstable, they showed the problems with data collection for training the model, here several approaches could be used to improve the model and patch the problems seen. Lastly, the reinforcement controller worked well but from a control theory perspective, it is very hard to prove the stability of the controller.

Acknowledgments

I would like to thank my two supervisors, Abhishek Dhar at LiU and Filip Peterson at Combine. Both have been great support and always helped when needed. Further, I would like to thank Tobias Olsson at Combine who was able to step in as an extra supervisor when needed and made sure that I felt welcome at Combine together with the rest of the company. Lastly, I would like to thank my examiner Daniel Axehill who has made sure that I have been on the right path to a good thesis.

Linköping, 2022
Axel Ek

Contents

Notation

Mathematics

Notation	Meaning
$\mathbb{R}$	The set of real numbers

Abbreviations

Abbreviations	Meaning
LQR	Linear Quadratic Regression (Regulator)
NN	Neural Network
PID	Proportional, Integral and Derivative controller
DOF	Degrees of Freedom
MSE	Mean Squared Error
MPC	Model Predictive Controller
PPO	Proximal Policy Optimization
TRPO	Trust Region Policy Optimization
ROA	Region of Attraction
RF	Reinforcement Learning
OSPQ	Operator Splitting Quadratic Program

1

Introduction

1.1 Background

Physics-based models of dynamical systems for control policies can be hard and take a lot of time to derive. And there will always be some effect that was not added to the calculations, like aerodynamic-, gyroscopic- or frictional- effects that might affect the system dynamics. To limit these errors different tactics were used when designing the control policies are made to be able to handle them, Glad and Ljung [4].

For systems where the mechanical description of the system is used to create the control policy, if the description of the system is poor, the complexity of the control policy needs to be higher. To solve this problem, the description of the model needs to be improved or the complexity of the control policy is raised.

One way to speed up this process is Machine Learning (ML). ML is a field that has been used to improve many similar problems in recent years. There are broadly two methods to use ML on control systems: learning the control policy with reinforcement learning and learning the system model with some ML or Deep Learning (DL) algorithm. For the first strategy, there are many off-the-shelf methods, with different probabilities of stably controlling the system. For example, Proximal Policy Optimization (PPO) is a reinforcement learning method that is commonly used for learning control of arbitrary systems, Jonsdottir and Petersson [6], Schulman et al. [15]. The second method involves learning the system model with ML and using it for controlling the system. The advantage of this method is that popular control theory methods can be used to guarantee desirable performance for dynamical systems. This is the method that will be focused on in this thesis.

To validate the efficacy of the strategy proposed in this thesis, a unicycle system is chosen as a testbed. The unicycle is a good way to benchmark different control methods, as it can be seen as a combination of an inverted pendulum and a reaction wheel, Siradjuddin et al. [16], Lee et al. [7]. The experimental studies in this thesis are done using a unicycle provided by Combine.

1.2 Problem formulation

The objective of this thesis is to develop a system model using neural networks which learn the system models utilizing the input/output data from the system. As presented in Section 1.3 the previous works have been focusing on the performance of the system model. To improve on these works the problem becomes, how the models perform when applied with control policies and do they differ in performance from the physics-based models. Further, the closed-loop performance obtained with said neural network-based approach is compared with the classical approach of controlling the system by utilizing a physics-based model of the system. The unicycle system considered in this thesis is seen in Figure 1.1. It is a balancing robot that has an unstable equilibrium in the upright position. They are all tested in simulation on a non-linear description of the model, and if they showed good behavior there taken out on the real-world unicycle.

When designing good control algorithms for a system a sufficiently rich knowledge about the physics of the system is needed to be able to predict the motions correctly, which is essential to derive a control policy to guarantee the desired behavior of the system. The questions this thesis aims to answer can be summarized as follows:

- How does a generic controller work on the system?
- Is it possible to improve the performance with a neural network based learned model?
- Can a model predictive controller (MPC) further improve the performance with operational constraints being considered explicitly?
- What is the performance difference between the control policies derived from the different models?

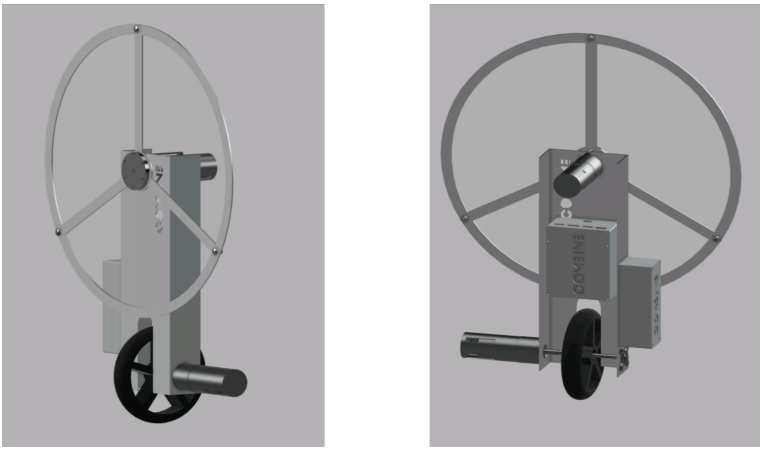


Figure 1.1: *The unicycle*

1.3 Related work

The usage of neural networks to find the model of a dynamic system has been used for a long time. Sjöberg et al. [17] presented the theory in 1994 where they combined the theory of neural networks and regression models to create a black box model of the system. The models they presented were such as NNARX, NNFIR, and NNARMAX. The models were evaluated on known data from a hydraulic actuation for oil pressure. They present the possible improvement possibilities of collecting the data from the real-world system, starting the training from an initial guess to try to solve the problem by finding local optima for the model.

Ogunmolu et al. [11] presented the model performance of the three different nonlinear structures of NN 1) multilayer perceptrons, 2) recurrent neural networks, and 3) convolutional neural networks when these were used for system identification. Their results showed at best a model fit for the multilayer perceptions of 99.9905% on the glass furnace dataset from the DaISy file server.

The model for the Unicycle has been derived in Jonsdottir and Petersson [6] where

the comparison between a physics-based LQR controller and a Reinforcement learning (RL) controller was analyzed and presented. They presented some good results for the RL where the controller managed to stabilize the system in a good way in simulation and managed to stabilize in the real world with reasonable performance. Even if the controller showed some undesired behavior with high accelerations and maximizing the motors. They were not able to understand where this behavior originated and showed one of the drawbacks of using RL. Which is that it is hard to understand where the behavior originated and therefore hard to remove or add behavior that is or is not desired.

For nonlinear models Gautam [3] presented a NN based system identification approach to control an inverted pendulum. It used a three-layer network to create the nonlinear model for the system and a detuned controller to stabilize the system. It was shown that the nonlinear model performed well when controlling the system.

1.4 Delimitations

The focus of this thesis is to be able to balance the unicycle, the position of the unicycle is neglected as long as it balances close to the equilibrium.

1.5 Thesis outline

In Chapter 2 the model of the system is obtained through Lagrangian analysis. The necessary theory to derive this model is presented and described. Thereafter a description of how to derive an LQR controller is presented. This ties together by an analysis of how the LQR controller is able to control the system both in simulation and real-world tests.

In Chapter 3 a description of how a NN can be used to learn the model of the system is made. This model is used to derive an LQR controller for the system, which is tested through simulation. In Chapter 4 an MPC controller is designed from the four different models produced in the chapters before.

The main part of the thesis is finished with the discussion in Chapter 6 where the results are discussed and why the results ended up as they did. Lastly, in Chapter 7 the conclusions and future work are discussed.

2

Control using physics-based model

In this chapter, the physics-based model of the unicycle is analytically derived. The two main methods to derive the model are the Newtonian approach and the Lagrangian approach. In this thesis, the Lagrange model will be used. Thereafter the theory to derive an LQR controller is presented and lastly the results, obtained through simulating the LQR on the model are presented.

2.1 Mathematical description

To describe the systems' model the rotation coordinates and the connection between the coordinate systems need to be described.

2.1.1 Rotation coordinates

The rotational description for an arbitrary vector $\mathbf{p^0}$ between two arbitrary frames $\mathbf{F^0}$ and $\mathbf{F^1}$ can be described as follows. Consider vector $\mathbf{p^0}$ in frame $\mathbf{F^0}$ as:

$$\mathbf{p}^0 = \begin{bmatrix} p_x^0 \\ p_y^0 \\ p_z^0 \end{bmatrix} \tag{2.1}$$

Where p_x^0, p_y^0, and p_z^0 is the scalar components of the arbitrary vector $\mathbf{p^0}$. Then $\mathbf{p^0}$ in frame $\mathbf{F^1}$ can be described as:

$$\boldsymbol{p}^1 = \mathbf{R_0^1} \begin{bmatrix} p_x^0 \\ p_y^0 \\ p_z^0 \end{bmatrix} \tag{2.2}$$

Where $\mathbf{R_0^1}$ is the rotation matrix from frame $\mathbf{F^0}$ to frame $\mathbf{F^1}$.
The rotation matrices that are of interest for the physics-based description in this thesis are the elementary rotations along the $x-$, $y-$ and $z-$ axes, they are:

$$\mathbf{R_x}(\varphi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi \\ 0 & -\sin\varphi & \cos\varphi \end{bmatrix} \tag{2.3}$$

$$\mathbf{R_y}(\theta) = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2.4}$$

$$\mathbf{R_z}(\psi) = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

Where φ is the rotation around the x-axis, θ is the rotation around the y-axis and ψ is the rotation around the z-axis. These three rotation matrices can be combined and create complex rotational descriptions between different frames.

2.2 Description of the unicycle model characteristics

To describe the position of the unicycle a state that describes the current position of the unicycle is denoted by **s**. To describe this model of the system, the Degrees Of Freedom (DOF) of the system need to be calculated.

The unicycle can be seen as three connected bodies the *wheel*, the *body*, and the *disc*. These bodies can be described with four frames, the world frame, the wheel frame, the body frame, and the disc frame. The world frame which is the space fixed coordinate system, is denoted by $[x_0, y_0, z_0]$. The wheel frame is the coordinate system connected from the center of mass of the wheel and is not affected by the rotation of the wheel, this frame is denoted by $[x_w, y_w, z_w]$. The body frame is the coordinate system from the center of mass of the body, this frame is denoted by $[x_b, y_b, z_b]$. Lastly, the disc frame is the coordinate system from the center of mass of the disc, this frame is denoted by $[x_d, y_d, z_d]$.

The DOF of the system can then be described as: i) the roll, which is the rotation angle of the system around the x_0-axis, ii) the pitch of the system which is the rotation angle around the y_w-axis, iii) the movement of the wheel along the x_0-axis creates the third DOF and iv) the rotation of the disc around the x_d-axis.

Thus the unicycle has four degrees of freedom, the roll, which is denoted by φ, the pitch, which is denoted by θ, the rotation of the wheel, which is denoted by α_w and the rotation of the disc which is denoted by α_d. To be able to describe the system behavior only these four states are needed.

To describe the positions of the system, the positions of the origins of the frames

can be expressed by disconnecting the three bodies and expressing them in the world frame.

The description and values of the constants used in the model can be found in the chapter appendix.

To describe the position of the wheel only its position to the world frame is needed and therefore described by:

$$\mathbf{p}_w^0 = \begin{bmatrix} -r_w \alpha_w \\ -r_w \sin\varphi \\ r_w \cos\varphi \end{bmatrix} \tag{2.6}$$

Where r_w is the radius of the wheel. The body is connected to the wheel and to describe the rotation frame $\mathbf{R_w^0}$ is given by:

$$\mathbf{R_w^0} = \mathbf{R_x^T} \tag{2.7}$$

Where $\mathbf{R_x}$ is the rotation frame described in Equation 2.3. Then the origin of the body frame can be expressed by:

$$\mathbf{p}_b^0 = \mathbf{p}_w^0 + \mathbf{R_w^0 p}_b^w = \begin{bmatrix} -r_w \alpha_w + L_{wb} \sin\theta \\ -r_w \sin\varphi + L_{wb} \sin\varphi \cos\theta \\ r_w \cos\varphi + L_{wb} \sin\varphi \cos\theta \end{bmatrix} \tag{2.8}$$

Where L_{wb} is the length between the center of mass of the wheel and body. In the same way, the rotation matrix $\mathbf{R_d^w}$ between the body and the disc is described by:

$$\mathbf{R_d^w} = \mathbf{R_y^T} \tag{2.9}$$

Where $\mathbf{R_y}$ is the rotation frame described in Equation 2.4. then the origin of the disc frame can be expressed by:

$$\mathbf{p}_d^0 = \mathbf{p}_w^0 + \mathbf{R_w^0 p}_d^w = \begin{bmatrix} -r_w \alpha_w + L_{wd} \sin\theta \\ -r_w \sin\varphi + L_{wd} \sin\varphi \cos\theta \\ r_w \cos\varphi + L_{wd} \sin\varphi \cos\theta \end{bmatrix} \tag{2.10}$$

Where L_{wd} is the length between the center of mass of the wheel and the disc.

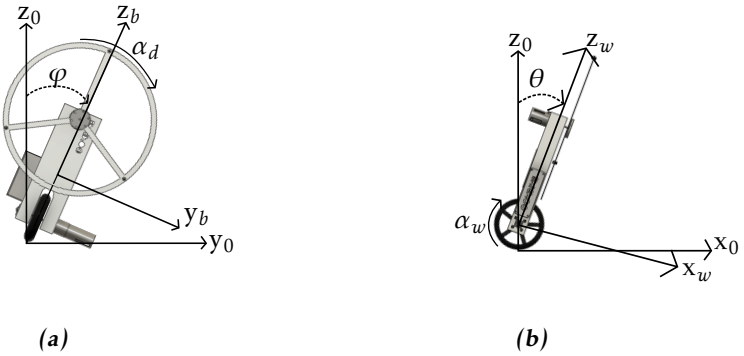


Figure 2.1: *Description of the coordinate systems.*

2.3 Lagrange dynamics

To describe the Lagrangian the essential states are the DOF for the system. In Section 2.2 it was shown that there are four DOF of the system, which are $\alpha_w, \alpha_d, \varphi, \theta$. So the state vector of the system can be described as follows,

$$\mathbf{q(t)} = \begin{bmatrix} \alpha_w(t) \\ \alpha_d(t) \\ \varphi(t) \\ \theta(t) \end{bmatrix} \in \mathbb{R}^4 \tag{2.11}$$

2.3.1 Lagrangian

The Lagrangian is a function that combines the energies of the system to derive its dynamical model, and is defined as:

$$\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}) = T(\mathbf{q}, \dot{\mathbf{q}}) - U(\mathbf{q}) \tag{2.12}$$

Where $T(\mathbf{q}, \dot{\mathbf{q}})$ is the kinetic energy and $U(\mathbf{q})$ is the potential energy of the concerned system.

2.3.2 Potential energy

The potential energy of the system can be described by:

$$U(\mathbf{q}) = g \begin{bmatrix} m_w & m_b & m_d \end{bmatrix} \begin{bmatrix} \mathbf{p_w^0} \\ \mathbf{p_b^0} \\ \mathbf{p_d^0} \end{bmatrix} \tag{2.13}$$

Where g is the acceleration due to gravity, m_i is the mass of body i and $\mathbf{p_i^0}$ is the position of the body in the world frame, where $i = w, b, d$

2.3.3 Kinetic energy

The kinetic energy can be split in two parts, translational $T_t(\dot{\mathbf{q}}, \mathbf{q})$ and rotational energy $T_r(\dot{\mathbf{q}}, \mathbf{q})$. The transitional energy, i.e the energy created by the motion of the bodies of the system and can be described as follows:

$$T_t = \frac{1}{2} m_w \dot{\mathbf{p}}_\mathbf{w}^\mathbf{0}(\mathbf{q}) \dot{\mathbf{p}}_\mathbf{w}^\mathbf{0}(\mathbf{q})^T + \frac{1}{2} m_b \dot{\mathbf{p}}_\mathbf{b}^\mathbf{0}(\mathbf{q}) \dot{\mathbf{p}}_\mathbf{b}^\mathbf{0}(\mathbf{q})^T + \frac{1}{2} m_d \dot{\mathbf{p}}_\mathbf{d}^\mathbf{0}(\mathbf{q}) \dot{\mathbf{p}}_\mathbf{d}^\mathbf{0}(\mathbf{q})^T \tag{2.14}$$

Where $\dot{\mathbf{p}}_\mathbf{i}^\mathbf{0}(\mathbf{q})$ is the time derivative of the position of body i.

The rotational energy for the bodies are described as:

$$T_{ri} = \frac{1}{2} \omega_\mathbf{i}^\mathbf{i} \mathbf{I_i} \omega_\mathbf{i}^\mathbf{i} \qquad , i = w, b, d \tag{2.15}$$

Where $\mathbf{I_i}$ is the inertia matrix of the body i and ω_i^i is the angular velocity of the body in its own frame. The angular velocities are described as following.

For the wheel, the rotations of the body is the spin of the wheel α_w, and the roll around the x_0-axis $\dot{\varphi}$.

$$\omega_w^w = \mathbf{R_x} \begin{bmatrix} \dot{\varphi} \\ 0 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ \dot{\alpha}_w \\ 0 \end{bmatrix} \tag{2.16}$$

The pitch $\dot{\theta}$ and roll $\dot{\varphi}$ of the system gives the rotational velocity of the body ω_b^b.

$$\omega_b^b = \mathbf{R_x R_y} \begin{bmatrix} \dot{\varphi} \\ 0 \\ 0 \end{bmatrix} + \mathbf{R_y} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} \tag{2.17}$$

The disc follows the rotation of the body with the angular velocity of the disc $\dot{\alpha}_d$, whose angular velocity ω_d^d is defined as:

$$\omega_d^d = \omega_b^b + \begin{bmatrix} \dot{\alpha}_d \\ 0 \\ 0 \end{bmatrix} \tag{2.18}$$

Using (2.15)-(2.18), the total rotational energy is obtained as follows:

$$T_r(\mathbf{q}, \dot{\mathbf{q}}) = T_{rw} + T_{rb} + T_{rd} \tag{2.19}$$

The total kinetic energy is then defined as:

$$T(\mathbf{q}, \dot{\mathbf{q}}) = T_r + T_t \tag{2.20}$$

Where T_r is described in (2.19) and T_t in (2.14).

2.3.4 External forces

The last part that is needed to create the Lagrangian equation is the external forces on the system, $\mathbf{Q}$. The external forces are limited to the torque of the motors, these can be described as:

$$\mathbf{Q} = \begin{bmatrix} \tau_w \\ \tau_d \\ -\tau_d \\ \tau_w \end{bmatrix} \tag{2.21}$$

Where τ_d is the torque of the motor controlling the disc and τ_d is the torque of the motor controlling the wheel.

According to Kirchoffs voltage law Sahin et al. [13] the torque of the motor can be described as:

$$\tau(t) = K_m i(t) \tag{2.22}$$

where K_m is the torque constant of the motor and $i(t)$ is the armature current

$$-u(t) + R_a i(t) + u_m(t) = 0 \Leftrightarrow i(t) = \frac{u(t) - u_m(t)}{R_a} \tag{2.23}$$

Where R_a is the armature resistance, u is the input voltage, u_m is the reverse emf voltage which is dependent of the angular velocity of the motor and can be described as $u_m = K_u \omega(t)$, where K_u is the motor velocity constant. This together with (2.23), the torque of the motor can be described as:

$$\tau_m = K_m \frac{u_m(t) - K_u \dot{\alpha}_m(t)}{R_a} \tag{2.24}$$

So the external forces can be written as:

$$\mathbf{Q} = \begin{bmatrix} K_w \frac{u_w(t) - K_{u,w} \dot{\alpha}_w(t)}{R_a} \\ K_d \frac{u_d(t) - K_{u,d} \dot{\alpha}_d(t)}{R_a} \\ -K_d \frac{u_d(t) - K_{u,d} \dot{\alpha}_d(t)}{R_a} \\ K_w \frac{u_w(t) - K_{u,w} \dot{\alpha}_w(t)}{R_a} \end{bmatrix} \tag{2.25}$$

2.3.5 Lagrange equation

The Lagrangian function associated with the considered system can be described as:

$$\frac{d}{dt} \frac{\partial \mathcal{L}(\dot{\mathbf{q}}, \mathbf{q})}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}(\dot{\mathbf{q}}, \mathbf{q})}{\partial q_i} + \mathbf{Q}(\dot{\mathbf{q}}, \mathbf{u}) = 0, \quad i = 1...M \tag{2.26}$$

This creates a nonlinear description of the system. Equation (2.26) can be simplified if (2.12) is inserted in (2.26). Then (2.26) can be written as:

$$\frac{\partial}{\partial \dot{\mathbf{q}}} \frac{\partial T(\dot{\mathbf{q}}, \mathbf{q})}{\partial \dot{\mathbf{q}}} \ddot{\mathbf{q}} - \frac{\partial}{\partial \mathbf{q}} \frac{\partial T(\dot{\mathbf{q}}, \mathbf{q})}{\partial \dot{\mathbf{q}}} \dot{\mathbf{q}} - \frac{\partial \mathcal{L}(\dot{\mathbf{q}}, \mathbf{q})}{\partial \mathbf{q}} \dot{\mathbf{q}} + \mathbf{Q}(\dot{\mathbf{q}}, \mathbf{u}) = 0 \tag{2.27}$$

The Lagrangian can then be solved for $\ddot{\mathbf{q}}$ which gives a describing function of the system as:

$$\mathbf{M} \cdot \ddot{\mathbf{q}} = \mathbf{f}(\dot{\mathbf{q}}, \mathbf{q}, \mathbf{u}) \tag{2.28}$$

Where $\mathbf{M}$ is the mass matrix and $\mathbf{f}$ is the matrix of the functions, dependent on velocity, position and inputs of the system. Equation (2.28) is simplified to get the acceleration of the system as:

$$\ddot{\mathbf{q}} = \mathbf{M}^{-1}(\dot{\mathbf{q}}, \mathbf{q}, \mathbf{u}) \cdot \mathbf{f}(\dot{\mathbf{q}}, \mathbf{q}, \mathbf{u}) \tag{2.29}$$

From this description of the acceleration, the velocity and position states can be derived with initial value solvers and the full description of the system is obtained.

2.4 Simulation environment

To analyze the closed-loop performance of the unicycle system, a simulation environment was made. This simulation environment is using the full nonlinear description of the system as presented in Section 2.3. The initial value solver used in the simulation is the Runge-Kutta 45.

2.5 LQR control theory

Linear Quadratic Regulator (LQR) is a full state feedback method where the system is controlled by suitably designing an optimal closed-loop gain.

2.5.1 Linearization

The first step to being able to synthesize an LQR controller is to linearize the nonlinear model. The dynamics of the nonlinear system can be described as:

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \qquad \mathbf{x} \in \mathbb{R}^n, \mathbf{u} \in \mathbb{R}^m \tag{2.30}$$

$$\mathbf{y}(t) = \mathbf{h}(\mathbf{x}(t), \mathbf{u}(t)), \qquad \mathbf{y} \in \mathbb{R}^p \tag{2.31}$$

For an LQR controller the linearization is performed around the equilibrium point $\mathbf{x}_{eq}, \mathbf{u}_{eq}$. At this point the system satisfies $\dot{\mathbf{x}} = 0$, which implies:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) = 0 \tag{2.32}$$

The linear dynamics can be seen as a valid approximation of the system model, in a neighborhood of the equilibrium point provided f and h are differentiable. Expanding f and h around the equilibrium with 1^{st} order Taylor expansion, the linearized dynamics can be described as:

$$\frac{d}{dt}\mathbf{x} = A\left(\mathbf{x}(t) - \mathbf{x}_{eq}\right)\Big) + B\left(u(t) - u_{eq}\right) \tag{2.33}$$

$$\mathbf{y} = C\left(\mathbf{x}(t) - \mathbf{x}_{eq}\right) + D\left(u(t) - u_{eq}\right) \tag{2.34}$$

Where:

$$A = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\substack{x=x_{eq} \\ u=u_{eq}}} \tag{2.35}$$

$$B = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{\substack{x=x_{eq} \\ u=u_{eq}}} \tag{2.36}$$

$$C = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\substack{x=x_{eq} \\ u=u_{eq}}} \tag{2.37}$$

$$D = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{u}}\right|_{\substack{x=x_{eq} \\ u=u_{eq}}} \tag{2.38}$$

2.5.2 Infinite-horizon discrete time LQR

After discretization Equation (2.33) can be expressed as:

$$\mathbf{x}_{k+1} = A\mathbf{x_k} + B\mathbf{u_k} \tag{2.39}$$

A performance index J is considered as follows:

$$J = \sum_{k=0}^{\infty} \left(\mathbf{x_k^T} Q \mathbf{x_k} + \mathbf{u_k^T} R \mathbf{u_k}\right) \tag{2.40}$$

Where Q and R are matrices that are given by trial and error.

Then the optimal feedback is:

$$\mathbf{u_k} = -K\mathbf{x_k} \tag{2.41}$$

Where

$$K = \left(R + B^T P B\right)^{-1} \left(B^T P A + N^T\right) \tag{2.42}$$

And P is the positive definite solution to the discrete time algebraic Riccati equation:

$$P = A^T P A - A^T P B \left(R + B^T P B\right)^{-1} B^T P A + Q \tag{2.43}$$

By combining (2.39) and (2.40), the close-loop system can be written as:

$$\mathbf{x}_{k+1} = (A - BK)\,\mathbf{x_k} \tag{2.44}$$

Equation 2.44 has guaranteed phase and gain margins and is asymptotically stable.

2.5.3 Controllability

A system is controllable if the controllability matrix $\mathcal{S}(A, B)$, is of full rank Glad and Ljung [4], where $\mathcal{S}(A, B)$ is defined as

$$\mathcal{S}(A, B) = \begin{bmatrix} B & AB & A^2B & \dots & A^{n-1}B \end{bmatrix} \tag{2.45}$$

With the A and B matrix given by the linearization in Section 2.5.1. The systems' controllability matrix has a rank of 6, which is equal to the system state dimension. Therefore the system is controllable.

2.6 LQR control

To design the LQR controller for the unicycle system, the nonlinear model is linerized as described in Section 2.5.1. The state vector is defined as:

$$\mathbf{x} = \begin{bmatrix} \dot{\alpha}_w \\ \dot{\alpha}_d \\ \dot{\varphi} \\ \dot{\theta} \\ \varphi \\ \theta \end{bmatrix} \tag{2.46}$$

The linearized system matrices are obtained as follows:

$$A = \begin{bmatrix} -23.189 & 0.001 & 0 & 0 & 0.100 & 129.730 \\ 0.006 & -2.018 & 0 & 0 & -21.228 & -0.057 \\ -0.006 & 0.132 & 0 & 0 & 21.228 & 0.057 \\ -7.920 & 0.001 & 0 & 0 & 0.057 & 73.969 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{2.47}$$

$$B = \begin{bmatrix} 21.653 & -0.002 \\ -0.005 & 7.697 \\ 0.005 & -0.503 \\ 7.395 & -0.001 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{2.48}$$

The system matrices in (2.47) - (2.48) give the continuous-time linearized system model. To implement the LQR, described in Section 2.5.2, the continuous-time model is discretized. The system matrices for the discrete-time linearized system are given as follows:

$$A_d = \begin{bmatrix} 7.928 \cdot 10^{-1} & 5.547 \cdot 10^{-6} & 4.676 \cdot 10^{-6} & 6.016 \cdot 10^{-3} & 9.010 \cdot 10^{-4} & 1.159 \cdot 10^{0} \\ 5.437 \cdot 10^{-5} & 9.800 \cdot 10^{-1} & -1.054 \cdot 10^{-3} & -2.728 \cdot 10^{-6} & -2.102 \cdot 10^{-1} & -5.324 \cdot 10^{-4} \\ -5.500 \cdot 10^{-5} & 1.308 \cdot 10^{-3} & 1.001 \cdot 10^{0} & 2.750 \cdot 10^{-6} & 2.122 \cdot 10^{-1} & 5.385 \cdot 10^{-4} \\ -7.077 \cdot 10^{-2} & 3.316 \cdot 10^{-6} & 2.750 \cdot 10^{-6} & 1.003 \cdot 10^{+0} & 5.385 \cdot 10^{-4} & 6.929 \cdot 10^{-1} \\ -2.854 \cdot 10^{-7} & 6.562 \cdot 10^{-6} & 1.000 \cdot 10^{-2} & 9.267 \cdot 10^{-9} & 1.001 \cdot 10^{0} & 2.750 \cdot 10^{-6} \\ -3.673 \cdot 10^{-4} & 1.699 \cdot 10^{-8} & 9.267 \cdot 10^{-9} & 1.001 \cdot 10^{-2} & 2.750 \cdot 10^{-6} & 1.003 \cdot 10^{0} \end{bmatrix} \tag{2.49}$$

$$B_d = \begin{bmatrix} 1.934 \cdot 10^{-1} & -2.115 \cdot 10^{-5} \\ -5.077 \cdot 10^{-5} & 7.620 \cdot 10^{-2} \\ 5.136 \cdot 10^{-5} & -4.989 \cdot 10^{-3} \\ 6.608 \cdot 10^{-2} & -1.264 \cdot 10^{-5} \\ 2.665 \cdot 10^{-7} & -2.502 \cdot 10^{-5} \\ 3.429 \cdot 10^{-4} & -6.480 \cdot 10^{-8} \end{bmatrix} \tag{2.50}$$

From these system matrices, an LQR controller can be synthesized as described in Section 2.5.2.

2.6.1 Penalty matrices

The choice of R and Q can be made differently depending on how the system is desired to work. In this thesis three main Matrices will be used, these have been chosen for their performance where they have shown the desired behavior on at least one of the models used in this thesis.

The matrices are the following:

$$Q_1 = \begin{bmatrix} 0.001 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.51a}$$

$$R_1 = \begin{bmatrix} 0.12 & 0 \\ 0 & 0.12 \end{bmatrix} \tag{2.51b}$$

$$Q_2 = \begin{bmatrix} 0.0001 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.0001 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.52a}$$

$$R_2 = \begin{bmatrix} 0.12 & 0 \\ 0 & 120 \end{bmatrix} \tag{2.52b}$$

$$Q_3 = \begin{bmatrix} 0.00001 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.00001 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0001 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0001 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \end{bmatrix} \tag{2.53a}$$

$$R_3 = \begin{bmatrix} 0.000001 & 0 \\ 0 & 1000000 \end{bmatrix} \tag{2.53b}$$

2.6.2 Control gain

With these control matrices and the linerized state space model of the system used in (2.42) the control gain K is obtained as:

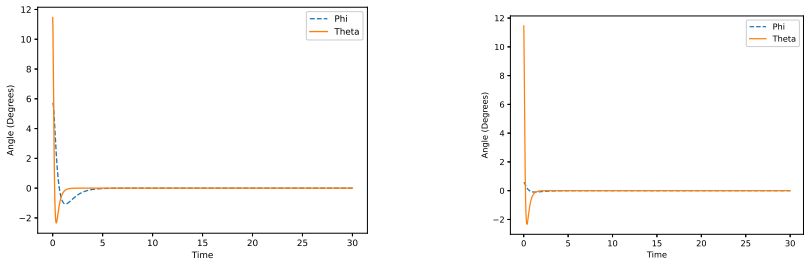
$$K_1 = \begin{bmatrix} -2.07 & -5.64 \cdot 10^{-5} & -1.62 \cdot 10^{-3} & 7.00 & -1.20 \cdot 10^{-2} & 4.65 \cdot 10^{1} \\ 2.40 \cdot 10^{-4} & -5.27 \cdot 10^{-1} & -2.82 \cdot 10^{1} & -5.00 \cdot 10^{-3} & -1.33 \cdot 10^{2} & -1.43 \cdot 10^{-2} \end{bmatrix} \tag{2.54}$$

$$K_2 = \begin{bmatrix} -2.068 \cdot 10^{0} & 2.531 \cdot 10^{0} & 1.260 \cdot 10^{2} & 7.677 \cdot 10^{0} & 5.750 \cdot 10^{2} & 4.813 \cdot 10^{1} \\ -1.104 \cdot 10^{-3} & -5.113 \cdot 10^{-1} & -2.546 \cdot 10^{1} & 2.199 \cdot 10^{-2} & -1.161 \cdot 10^{2} & 6.241 \cdot 10^{-2} \end{bmatrix} \tag{2.55}$$

$$K_3 = \begin{bmatrix} -9.564 \cdot 10^{-1} & 5.138 \cdot 10^{3} & 2.559 \cdot 10^{5} & -1.775 \cdot 10^{2} & 1.167 \cdot 10^{6} & -3.377 \cdot 10^{2} \\ -1.586 \cdot 10^{-6} & -3.233 \cdot 10^{-2} & -1.610 \cdot 10^{0} & 1.228 \cdot 10^{-3} & -7.345 \cdot 10^{0} & 4.788 \cdot 10^{-3} \end{bmatrix} \tag{2.56}$$

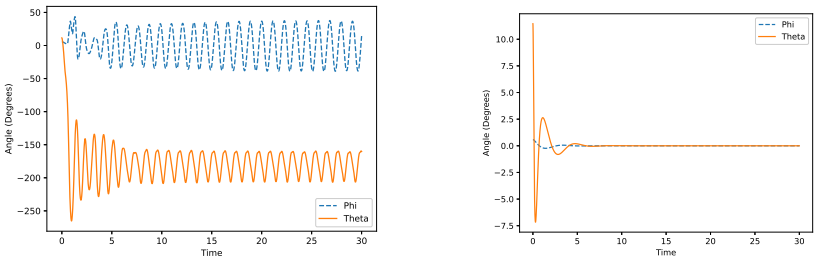
2.6.3 Simulation

The different control gain stabilize the systems with different characteristics. The following plots show the reaction of the system with the starting point as $\theta = 0.2^{rad} \approx 11.5°$ and $\varphi = 0.1^{rad} \approx 5.7°$ in plot a) and for plot b) $\theta = 0.02^{rad} \approx 1.15°$ and $\varphi = 0.01^{rad} \approx 0.6°$



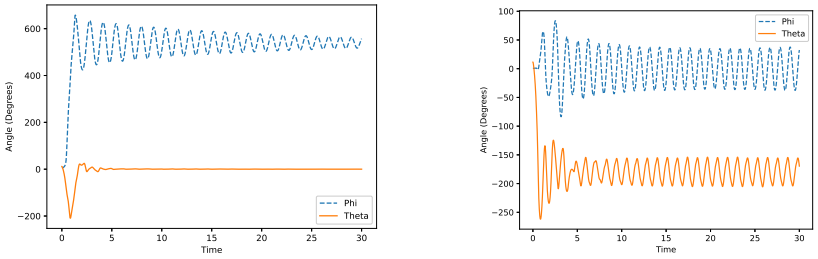
(a) *(b)*

Figure 2.2: *Balancing the unicycle system with LQR with control gain K_1*



(a) *(b)*

Figure 2.3: *Balancing the unicycle system with LQR with control gain K_2*



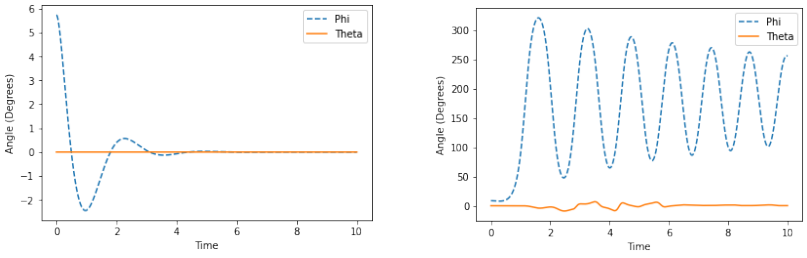
(a) *(b)*

Figure 2.4: *Balancing the unicycle system with LQR with control gain K_3*

The best control gain is K_1 and with it the penalty matrices of Q_1 and R_1.

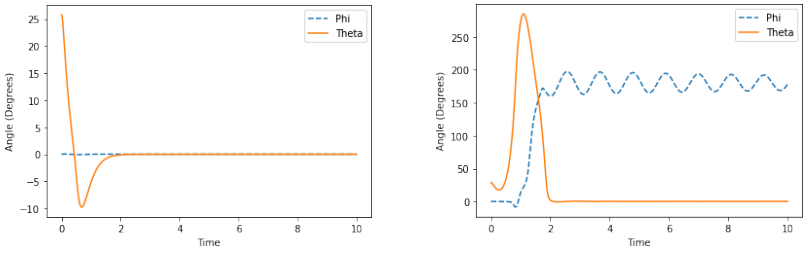
2.6.4 Characterizing region of attraction neighborhood

The linearized model of the system is only valid within a neighborhood around the equilibrium point. One way to find this area is to find initial states, from which the unicycle can stabilize itself. From these tests, the region of attraction (ROA) can be characterized. By trial and error the ROA when K_1 is used is found to be $|\theta| \leq 0.45^{rad} \approx 26°$ and $|\varphi| \leq 0.1^{rad} \approx 6°$. The process to find these angles is presented in figures 2.5-2.7.



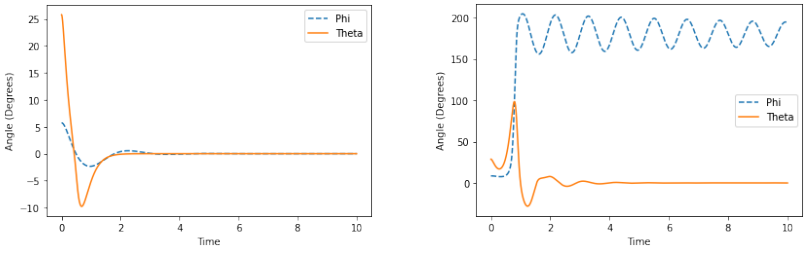
(a) *(b)*

Figure 2.5: *Balance with LQR in simulation with initial conditions to find max φ*



(a) *(b)*

Figure 2.6: *Balance with LQR in simulation with initial conditions to find max θ*



(a) *(b)*

Figure 2.7: *LQR with start just inside (a) and just outside (b) the stable zone.*

The same process were used to find the ROA when K_2 and K_3 are used as the control matrices. For K_2 the ROA is for $\varphi = 0.1^{rad} \approx 5.7°$ and $\theta = 0.45^{rad} \approx 25°$

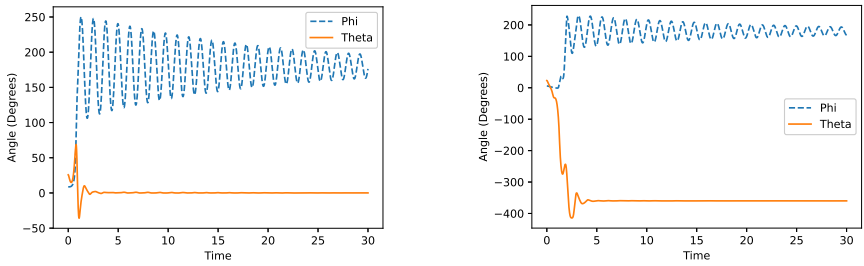


Figure 2.8: *Stable and unstable starting points with K_2 as control gain*

For K_3 the ROA is for $\varphi = 0^{rad} \approx 0°$ and $\theta = 0^{rad} \approx 0°$

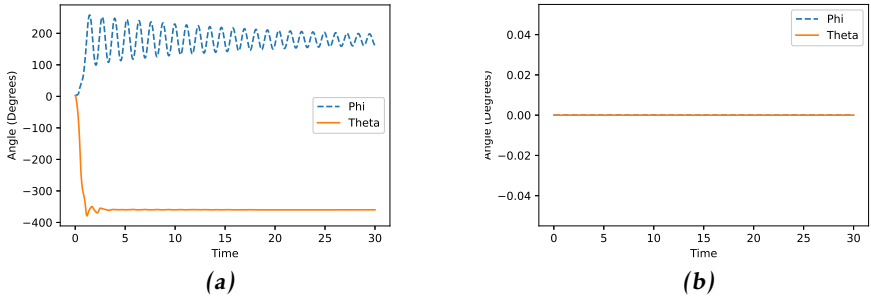


Figure 2.9: *Stable and unstable starting points with K_3 as control gain*

2.7 Disturbance reaction

To test the disturbance reactions of the control matrices the control input was set to a fixed value for 0.1 seconds, this value was slowly raised until the controller did not manage to stabilize after.

When K_1 is used to control the system the system behavior is shown in Figure 2.10

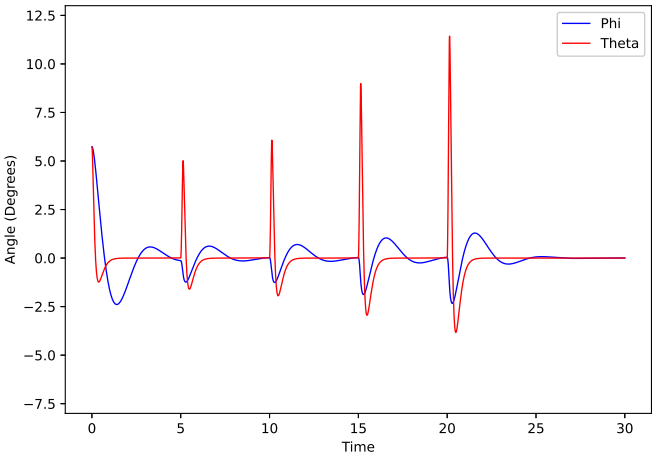


Figure 2.10: *LQR control with K_1 with disturbances every 5 seconds*

The disturbances the system manages to control are when it is inside the ROA presented in Section 2.6.4

2.7.1 Real world test

The most promising controller according to the behaviors in the simulation and ROA is K_1. This LQR controller is applied on the real-world unicycle and it manages to stand. It oscillates around θ but manages to get back to equilibrium.

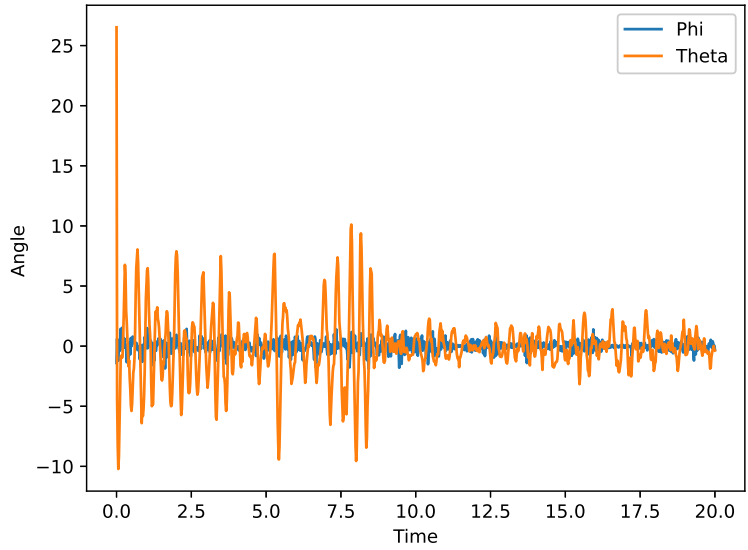


Figure 2.11: *Balance with LQR in the real world*

2.8 Summary

An LQR controller is synthesized from the Lagrangian nonlinear description of the unicycle, which is shown to stabilize the unicycle system for both the tests in simulation and the tests on the real-world unicycle system.

The controller works reasonably well, even though the pitch and roll of the system are highly correlated. There is room for improving the controller with the help of different filters and system models. These filters can for example dampen the reaction of the controller, as seen in Figure 2.11. The θ angle oscillates around the equilibrium but never manages to stabilize itself.

3

Learning-based model

In this chapter, the approach of learning the system model with a neural network is formulated. The model learned using the neural network is then used to derive an LQR controller. A short analysis of how the system performs, with the derived LQR, concludes the chapter.

3.1 System Identification

System identification deals with building a mathematical model of a dynamical system based on the observed data of the system with statistical methods, Ljung [8]. System Identification is a black or gray box system, which is a system that can be viewed in terms of its inputs and outputs and the terms in the models have no direct or not complete physical interpretation, Ljung and Glad [9]. Thus when learning the model with a neural network is classified as a system identification method.

3.2 Neural Network

A neural network is a mathematical structure consisting of suitably designed nodes with weighted connections amongst themselves. The network tries to learn a system model by updating the connection weights, by utilizing the input/output data of the concerned system, Bishop [1].

3.2.1 Basic structure

A neural network is created by a number of input and output nodes that are connected. All the connections are assigned a weight that tries to map the input to

the output. For example, a neural network that has one input and one output node takes the input value and multiplies it with the weight between the nodes, and presents it as the output. This output is then compared with the actual output of the system (obtained input/output database of the system) and the weight of the connection is suitably adjusted so that errors between the obtained output and the actual output are reduced. See Figure 3.1 for a visual example.

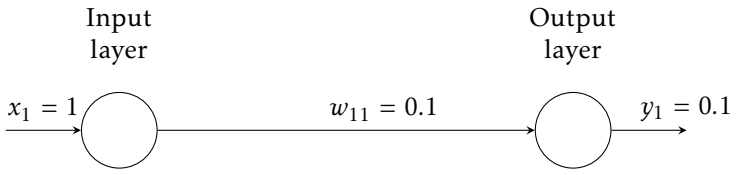


Figure 3.1: *Neural network with one input and one output*

This NN can be described in mathematical terms as:

$$w_{11} \cdot x_1 = y_1 \tag{3.1}$$

If the network is expanded with more inputs and outputs, a larger network of weights will connect the inputs with the outputs where the output is the sum of the inputs multiplied by the weights connected to the outputs and create a linear mapping between them, as seen in Figure 3.2.

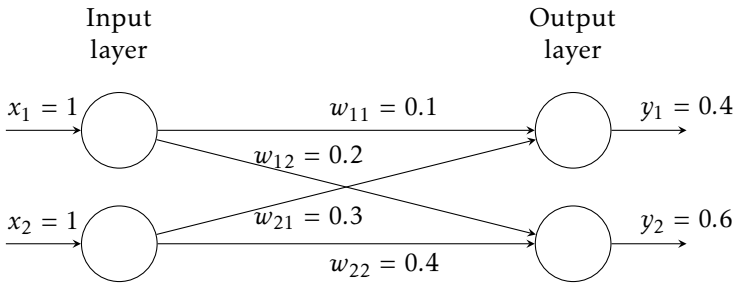


Figure 3.2: *Neural network*

The extended general NNs with j inputs and i outputs output can mathematically be described as:

$$y_i = \sum_j w_{ij} \cdot x_j \tag{3.2}$$

Which can be represented in compact form as follows:

$$\begin{bmatrix} y_1 \\ \vdots \\ y_i \end{bmatrix} = \begin{bmatrix} w_{11} & \dots & w_{1j} \\ \vdots & \ddots & \vdots \\ w_{i1} & \dots & w_{ij} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_j \end{bmatrix} \tag{3.3}$$

3.2.2 Activation function

In neural networks, the activation function passes the values after a node through a function. So for example after the output layer a function is added, so the output would be $g(w_{11}x_1 + \cdots + w_{84}x_8)$ where g is the activation function. In this thesis, the linear activation function is used. The linear activation function sets the function g as follows:

$$g(x) = x \tag{3.4}$$

3.2.3 Loss function

To quantify the performance of the neural network, a loss function is used. This function is a big part of the backpropagation algorithm (discussed below) and depending on what loss function is used the backpropagation will work differently. The most common loss function for regression problems is MSE - Mean Squared Error, which is described by the difference between the model prediction and the true measured value of the state squared, Blom et al. [2]. The loss function is denoted by $\varepsilon(j)$ and the equation for MSE is described by:

$$\varepsilon(j) = (y_j - z_j)^2 \tag{3.5}$$

Where y_j is the true measured value and z_j is the model predicted value.

3.2.4 Backpropagation

To train the NN, i.e. update its weights, backpropagation is used where the change of the weight between nodes i and j, Δw_{ij} is calculated by:

$$\Delta w_{ij} = -\eta \sum_{j=1}^{J} \frac{\partial \varepsilon(j)}{\partial w_{ij}} \tag{3.6}$$

Where η is the learning rate, j is the output node, J is the total number of output nodes, and ε is the value of the loss functions for the measurement of the output node.

3.2.5 Student-Teacher learning

Student-Teacher methods are used to help the network that is going to be deployed to start training from a point where it has been given an advantage and therefore has a higher chance to find the global optima of the system. This can be realized in several different ways. One is to let a network train in a simulated environment where it has been given privileged knowledge, like its global position. This network is called the teacher network. This network is later used to help train the network that is going to be deployed, the student network, to find the best model.

The method of warm starting the training of a network is often used to help give

the network a good guess as a starting point and is classified as a student-teacher method.

3.3 Neural network to system description

The system can be described as in (2.39), if we substitute $A\mathbf{x_k} + B\mathbf{u_k}$ with $\mathbf{F}(\dot{\mathbf{q}}, \mathbf{q}, \mathbf{u}) = \mathbf{F} \cdot \begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \\ \mathbf{u} \end{bmatrix}$ we get $\ddot{\mathbf{q}} = \mathbf{F} \cdot \begin{bmatrix} \mathbf{q} \\ \dot{\mathbf{q}} \\ \mathbf{u} \end{bmatrix}$. Then $\mathbf{F}$ is a $n \times m$ matrix where n is the number of states in $\dot{\mathbf{q}}, \mathbf{q}$ and m is the number of states in $\ddot{\mathbf{q}}$. Then F can be learned with a NN and then rewritten back to state space form as $\dot{\mathbf{x}} = \mathbf{A} \cdot \mathbf{x} + \mathbf{B} \cdot \mathbf{u}$.

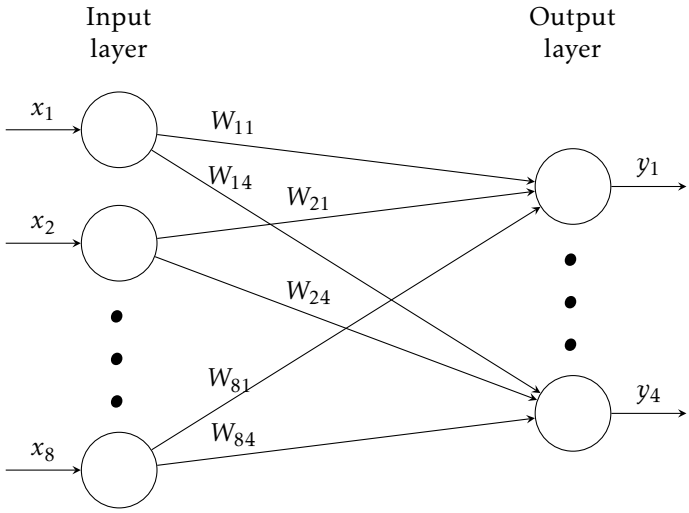


Figure 3.3: *Neural network*

F and therefore A and B can be derived from the structure of the NN. Where the weights between the input nodes and output nodes can be structured as a matrix. As can be seen in Figure 3.3 creates F as:

$$F = \begin{bmatrix} W_{11} & W_{21} & W_{31} & W_{41} \\ W_{12} & W_{22} & W_{32} & W_{42} \\ W_{13} & W_{23} & W_{33} & W_{43} \\ W_{14} & W_{24} & W_{34} & W_{44} \\ W_{15} & W_{25} & W_{35} & W_{45} \\ W_{16} & W_{26} & W_{36} & W_{46} \\ W_{17} & W_{27} & W_{37} & W_{47} \\ W_{18} & W_{28} & W_{38} & W_{48} \end{bmatrix} \tag{3.7}$$

$$F^T \cdot \begin{bmatrix} \dot{\alpha}_w \\ \dot{\alpha}_d \\ \dot{\varphi} \\ \dot{\theta} \\ \varphi \\ \theta \\ u_w \\ u_d \end{bmatrix} = \begin{bmatrix} \ddot{\alpha}_w \\ \ddot{\alpha}_d \\ \ddot{\varphi} \\ \ddot{\theta} \end{bmatrix} \tag{3.8}$$

In this example the system is described by $\mathbf{x} = \begin{bmatrix} \dot{\alpha}_w \\ \dot{\alpha}_d \\ \dot{\varphi} \\ \dot{\theta} \\ \varphi \\ \theta \end{bmatrix}$.

To get the system matrices first the weights corresponding to **u** are taken out and create B as:

$$B = \begin{bmatrix} W_{17} & W_{27} & W_{37} & W_{47} \\ W_{18} & W_{28} & W_{38} & W_{48} \end{bmatrix}^T = \begin{bmatrix} W_{17} & W_{18} \\ W_{27} & W_{28} \\ W_{37} & W_{38} \\ W_{47} & W_{48} \end{bmatrix} \tag{3.9}$$

And the A matrix then becomes:

$$A = \begin{bmatrix} W_{11} & W_{21} & W_{31} & W_{41} & 0 & 0 \\ W_{12} & W_{22} & W_{32} & W_{42} & 0 & 0 \\ W_{13} & W_{23} & W_{33} & W_{43} & 1 & 0 \\ W_{14} & W_{24} & W_{34} & W_{44} & 0 & 1 \\ W_{15} & W_{25} & W_{35} & W_{45} & 0 & 0 \\ W_{16} & W_{26} & W_{36} & W_{46} & 0 & 0 \end{bmatrix}^T = \begin{bmatrix} W_{11} & W_{12} & W_{13} & W_{14} & W_{15} & W_{16} \\ W_{21} & W_{22} & W_{23} & W_{24} & W_{25} & W_{26} \\ W_{31} & W_{32} & W_{33} & W_{34} & W_{35} & W_{36} \\ W_{41} & W_{42} & W_{43} & W_{44} & W_{45} & W_{46} \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{3.10}$$

The complete system can finally be described as:

$$\begin{bmatrix} \ddot{\alpha}_w \\ \ddot{\alpha}_d \\ \ddot{\varphi} \\ \ddot{\theta} \\ \dot{\varphi} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} W_{11} & W_{12} & W_{13} & W_{14} & W_{15} & W_{16} \\ W_{21} & W_{22} & W_{23} & W_{24} & W_{25} & W_{26} \\ W_{31} & W_{32} & W_{33} & W_{34} & W_{35} & W_{36} \\ W_{41} & W_{42} & W_{43} & W_{44} & W_{45} & W_{46} \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \dot{\alpha}_w \\ \dot{\alpha}_d \\ \dot{\varphi} \\ \dot{\theta} \\ \varphi \\ \theta \end{bmatrix} + \begin{bmatrix} W_{17} & W_{18} \\ W_{27} & W_{28} \\ W_{37} & W_{38} \\ W_{47} & W_{48} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} u_w \\ u_d \end{bmatrix} \tag{3.11}$$

3.4 Dataset

The dataset used to train the neural network to learn the model of the system was created by running the unicycle in the real world. During the data collection, the LQR designed in Chapter 2 was used to stabilize the Unicycle. To get the dataset

as diverse as possible firstly the controller ran with the standard LQR controller from Chapter 2, where the systems model around the equilibrium is in focus. Sequentially a white noise was added to the input to obtain diverseness in the collected data. The standard deviation of the white noise was slowly increased and when the unicycle was not able to stand, the white noise was introduced only at every tenth time step.

This ended up with around 400 000 data points that could be used for learning the system behavior.

3.5 Training

To find the best training approach for the models several different approaches were tested and the final training method includes the following structure.

3.5.1 Loss function

The loss function used was MSE. This method returns the difference between the true measured value and the predicted value squared.

$$\varepsilon(j) = (y_j - z_j)^2 \tag{3.12}$$

This gives the backpropagating term as:

$$\frac{\partial \varepsilon(j)}{\partial z_j} = -2(y_j - z_j) \tag{3.13}$$

Where y_j is the true measured value and z_j is the model predicted value.

3.5.2 Epochs

To get as much as possible from the dataset, the model was trained on $\sim 80\%$ of the data and the rest of the data was used for evaluation and testing. The training data was then split into five batches, which were iterated for each epoch of training. Up to 100 epochs were used to produce the models.

3.6 Models

To try to find the best model to describe the system two different model structures were tested.

3.6.1 Model structure

The linearized model can be created in different ways. To create a linearized model the A matrix of the system model needs to be able to be described with a quadratic matrix. The first model is the same as used in Chapter 2 the A matrix was described as a 6×6 matrix where

$$\begin{bmatrix} \ddot{\alpha} \\ \ddot{\alpha} \\ \ddot{\varphi} \\ \ddot{\theta} \\ \dot{\varphi} \\ \dot{\theta} \end{bmatrix} = A_1 \begin{bmatrix} \dot{\alpha}_w \\ \dot{\alpha}_d \\ \dot{\varphi} \\ \dot{\theta} \\ \varphi \\ \theta \end{bmatrix} + B_1 \begin{bmatrix} u_w \\ u_d \end{bmatrix} \tag{3.14}$$

Where the two last rows in A_1 are

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

and

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

since they describe $\dot{\varphi}$ and $\dot{\theta}$ directly, as seen in Section 3.3.

The second models structure is describing the velocity and creates the A matrix as 4×4 which is described as the following system:

$$\begin{bmatrix} \dot{\alpha}_w \\ \dot{\alpha}_d \\ \dot{\varphi} \\ \dot{\theta} \end{bmatrix} = A_2 \begin{bmatrix} \alpha_w \\ \alpha_d \\ \varphi \\ \theta \end{bmatrix} + B_2 \begin{bmatrix} u_w \\ u_d \end{bmatrix} \tag{3.15}$$

The theory derived in Section 3.3 is used to create the matrices for the second model structure. The difference is for F which will be a 6×6 matrix and a square matrix and thus no extra rows will be added.

3.6.2 System matrices

The first model structure gave the system matrices shown in (3.16)

$$A_1 = \begin{bmatrix} 7.429 \cdot 10^{-4} & -2.423 \cdot 10^{-3} & -2.745 \cdot 10^{-2} & 6.024 \cdot 10^{-2} & -1.002 \cdot 10^{-1} & -1.535 \cdot 10^{0} \\ 2.415 \cdot 10^{-3} & -3.298 \cdot 10^{-2} & -1.318 \cdot 10^{0} & -8.788 \cdot 10^{-3} & 2.041 \cdot 10^{0} & 2.164 \cdot 10^{-1} \\ -2.012 \cdot 10^{-3} & -4.371 \cdot 10^{-3} & -1.402 \cdot 10^{-1} & 1.091 \cdot 10^{-2} & -1.092 \cdot 10^{0} & 2.720 \cdot 10^{-2} \\ -3.994 \cdot 10^{-2} & 6.373 \cdot 10^{-3} & 2.919 \cdot 10^{-1} & 1.991 \cdot 10^{-1} & 1.161 \cdot 10^{0} & 2.068 \cdot 10^{-1} \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{3.16a}$$

$$B_1 = \begin{bmatrix} 1.486 \cdot 10^{-2} & 4.013 \cdot 10^{-3} \\ -6.443 \cdot 10^{-3} & 1.235 \cdot 10^{-1} \\ 2.490 \cdot 10^{-4} & 4.566 \cdot 10^{-3} \\ 2.818 \cdot 10^{-2} & -5.868 \cdot 10^{-3} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{3.16b}$$

The second model approach gave the system matrices shown in (3.17)

$$A_2 = \begin{bmatrix} -7.136 \cdot 10^{-4} & -1.112 \cdot 10^{-3} & 1.615 \cdot 10^{-4} & -1.153 \cdot 10^{-4} \\ -7.267 \cdot 10^{-5} & 1.054 \cdot 10^{-3} & -2.264 \cdot 10^{-4} & -1.405 \cdot 10^{-4} \\ -4.864 \cdot 10^{-1} & -1.802 \cdot 10^{2} & -5.750 \cdot 10^{-1} & 1.373 \cdot 10^{-2} \\ 2.374 \cdot 10^{1} & -1.806 \cdot 10^{0} & 3.909 \cdot 10^{-2} & 2.783 \cdot 10^{0} \end{bmatrix} \tag{3.17a}$$

$$B_2 = \begin{bmatrix} 5.354 \cdot 10^{-1} & -4.287 \cdot 10^{-3} \\ 2.980 \cdot 10^{-3} & 1.585 \cdot 10^{+0} \\ -5.123 \cdot 10^{-4} & -5.393 \cdot 10^{-3} \\ -1.017 \cdot 10^{-3} & 5.955 \cdot 10^{-3} \end{bmatrix} \tag{3.17b}$$

3.6.3 Student Network

As described in section 3.2.5 the learning of the model can be started from a privileged starting point. This was done by setting the start weights as the weights derived in section 2.5, and the state space models presented in equations (2.47) and (2.48). When this is applied the NN models start predicting according to the model that was produced in Chapter 2 and then starts to converge from that model towards the model that the dataset produced. This gives the NN a higher chance to converge towards the true model, where the true model is how the system is behaving with all the effects that were not added to the calculations in Chapter 2. Since only the first model approach was produced with the Lagrangian in Chapter 2. This is the only NN model that can be used with student-teacher learning.

When this approach was applied the following system matrices were produced:

$$A_{sn} = \begin{bmatrix} 8.133 \cdot 10^{-2} & 3.163 \cdot 10^{-3} & 3.111 \cdot 10^{-1} & -1.269 \cdot 10^{-1} & 9.454 \cdot 10^{-1} & -3.060 \cdot 10^{0} \\ 3.162 \cdot 10^{-3} & -1.025 \cdot 10^{-2} & -6.929 \cdot 10^{-1} & 2.252 \cdot 10^{-2} & -6.179 \cdot 10^{-1} & 3.488 \cdot 10^{-2} \\ -2.194 \cdot 10^{-3} & -6.440 \cdot 10^{-4} & 7.585 \cdot 10^{-2} & 4.336 \cdot 10^{-3} & -1.358 \cdot 10^{-1} & 1.090 \cdot 10^{-2} \\ -2.730 \cdot 10^{-2} & -4.241 \cdot 10^{-4} & 4.831 \cdot 10^{-2} & 1.480 \cdot 10^{-1} & -8.748 \cdot 10^{-2} & 1.527 \cdot 10^{-1} \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{3.18a}$$

$$B_{sn} = \begin{bmatrix} -4.048 \cdot 10^{-2} & -2.213 \cdot 10^{-3} \\ -4.293 \cdot 10^{-3} & 2.199 \cdot 10^{-2} \\ 2.456 \cdot 10^{-3} & 7.625 \cdot 10^{-4} \\ 1.756 \cdot 10^{-2} & 1.529 \cdot 10^{-3} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{3.18b}$$

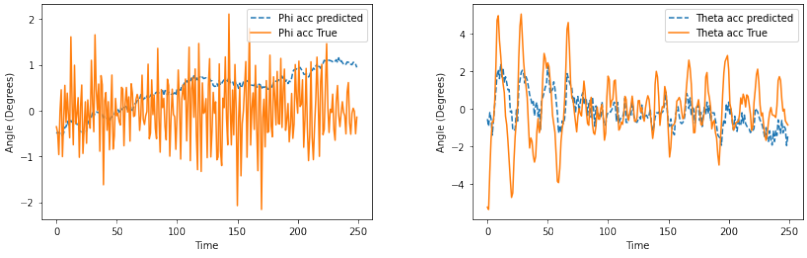
3.6.4 Model validation

For the second model, the rank of the controllability matrix is 4 which is a full rank and therefore controllable according to the controllability theorem.

For the first and the student-based model, the rank of the corresponding controllability matrices is 6 which is of full rank, and therefore the model is controllable.

To validate the performances of the models they were used to predict the motions of the unicycle from a shorter real world test run. The same test data were used for all models except for the second model which measures the velocity of the states compared to the acceleration in the other models.

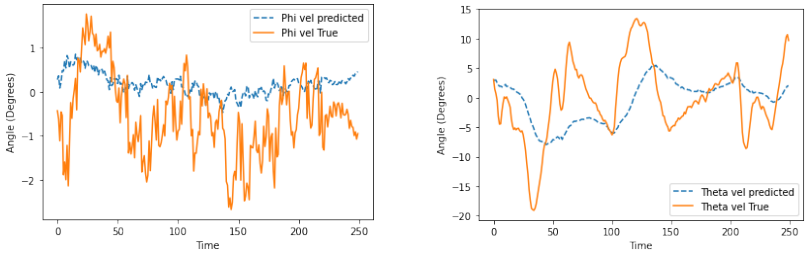
For the first model the measurement of $\ddot{\varphi}$ and $\ddot{\theta}$ is shown in Figure 3.4. The measurement of $\ddot{\varphi}$, seen in Figure 3.4a, begins performing well but drifts away towards the end of the test. For the measurement of $\ddot{\theta}$, see Figure 3.4b, it follows the reactions of the test data but it does not spike as high.



(a) *(b)*

Figure 3.4: *True and measured values of model 1*

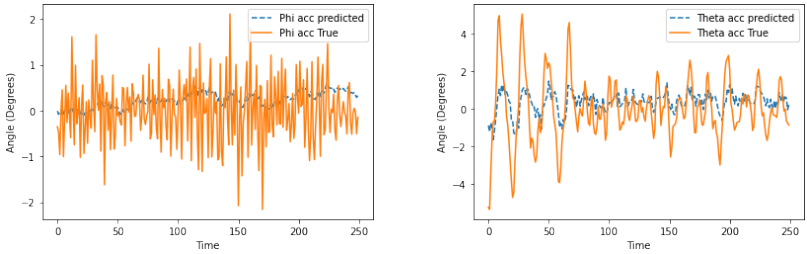
The second model is not predicting the changes in $\dot{\varphi}$ and $\dot{\theta}$ well. It tries to follow but does not manage to react to any of the changes in the system.



(a) *(b)*

Figure 3.5: *True and measured values of model 2*

For the student network model measurements in Figure 3.6 its measurements are similar to model three for both $\ddot{\varphi}$ and $\ddot{\theta}$. But a small improvement of reacting on the changes in accelerations of φ where it seems to follow the true value better.



(a) *(b)*

Figure 3.6: *True and measured values of the student network*

3.7 Simulation

The same simulation environment that was created by the nonlinear model in Section 2.3 was used, with the same solver Runge-Kutta 45. The simulation gives an understanding if the model is promising or not before being applied in the real world. For all simulations, the same nonlinear model was used.

3.8 LQR control

The system matrices from the different models were used to create the LQR controllers which were applied in the simulation environment. The simulation angle

of φ is shifted by π for all models. This result is discussed in Chapter 7.

3.8.1 Model 1

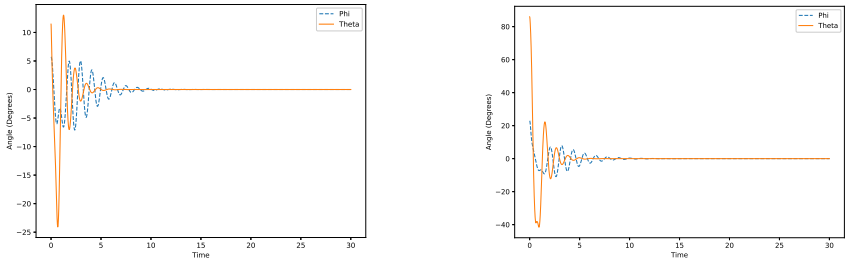
Then the derived K_d matrices becomes:

$$K_1 = \begin{bmatrix} -3.015 \cdot 10^{0} & 2.030 \cdot 10^{-1} & 4.102 \cdot 10^{1} & 4.950 \cdot 10^{1} & 1.969 \cdot 10^{1} & 2.376 \cdot 10^{1} \\ 2.024 \cdot 10^{-1} & -3.025 \cdot 10^{-4} & -2.475 \cdot 10^{0} & -3.284 \cdot 10^{0} & ,-1.243 \cdot 10^{0} & -1.595 \cdot 10^{0} \end{bmatrix} \tag{3.19}$$

$$K_2 = \begin{bmatrix} -3.031 \cdot 10^{0} & 1.957 \cdot 10^{-1} & 4.139 \cdot 10^{1} & 4.974 \cdot 10^{1} & ,1.882 \cdot 10^{1} & 2.390 \cdot 10^{1} \\ 1.964 \cdot 10^{-4} & 7.774 \cdot 10^{-6} & 9.283 \cdot 10^{-3} & -3.406 \cdot 10^{-3} & 2.010 \cdot 10^{-3} & -1.562 \cdot 10^{-3} \end{bmatrix} \tag{3.20}$$

$$K_3 = \begin{bmatrix} -2.471 \cdot 10^{+2} & 1.142 \cdot 10^{+1} & 1.930 \cdot 10^{+2} & 9.770 \cdot 10^{+2} & 4.109 \cdot 10^{+1} & 8.836 \cdot 10^{+3} \\ -3.099 \cdot 10^{-9} & 1.575 \cdot 10^{-0} & 2.515 \cdot 10^{-9} & 1.493 \cdot 10^{-9} & 6.144 \cdot 10^{-1} & -1.504 \cdot 10^{-9} \end{bmatrix} \tag{3.21}$$

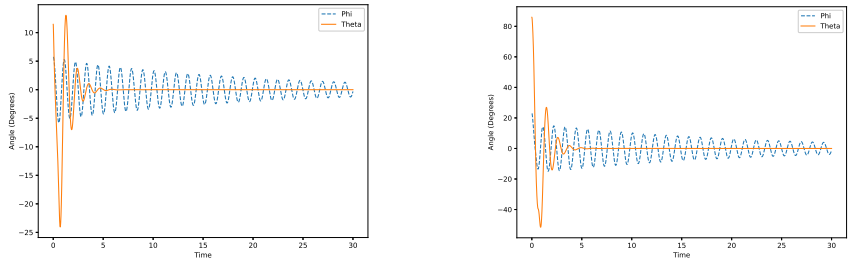
When tested in the simulation we get the following results:



(a) *(b)*

Figure 3.7: *Balancing the unicycle system with LQR with control gain* K_1

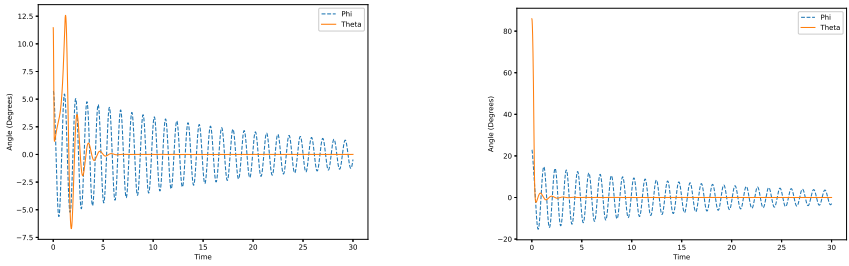
The first controller manages to stabilize within a reasonable time frame, even though the oscillation, in the beginning, is a bit higher.



(a) *(b)*

Figure 3.8: *Balancing the unicycle system with LQR with control gain K_2*

The second controller is not as fast to stabilize as the first but manages to stabilize in the end.



(a) *(b)*

Figure 3.9: *Balancing the unicycle system with LQR with control gain K_3*

The third controller has the worst behavior with high oscillation for φ, but in some cases manages to stabilize θ quickly.

3.8.2 Model 2

Since the Q and R matrices that are used for the other models are not working on model 2, separate matrices are produced and used. The Q and R matrices that

were found to perform the best in the simulation were:

$$Q = \begin{bmatrix} 0.0001 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix} \tag{3.22a}$$

$$R = \begin{bmatrix} 0.12 & 0 \\ 0 & 1.2 \end{bmatrix} \tag{3.22b}$$

Then the derived K_d matrix, with the choice of Q and R as above, is obtained as follows:

$$K_d = \begin{bmatrix} 4.633 \cdot 10^{0} & -9.381 \cdot 10^{-1} & 6.039 \cdot 10^{-3} & 5.433 \cdot 10^{-1} \\ -1.546 \cdot 10^{-2} & -3.591 \cdot 10^{-1} & -1.318 \cdot 10^{-3} & -1.754 \cdot 10^{-3} \end{bmatrix} \tag{3.23}$$

When tested in the simulation we get the following result:

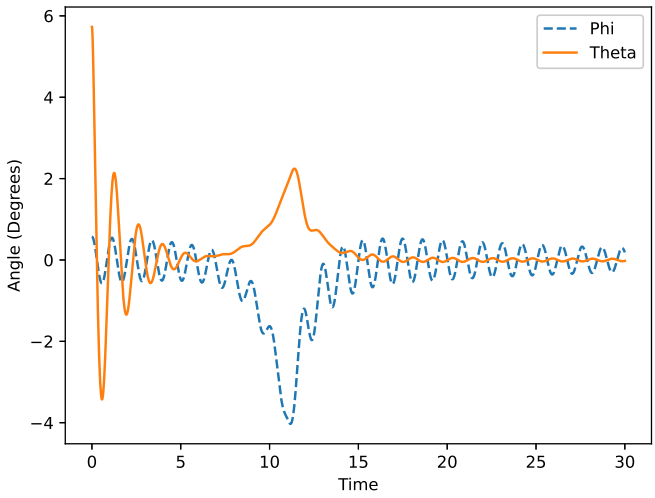


Figure 3.10: *LQR controller based on the second model*

The controller behaves undesirably, but manages to stabilize.

3.8.3 Model student network

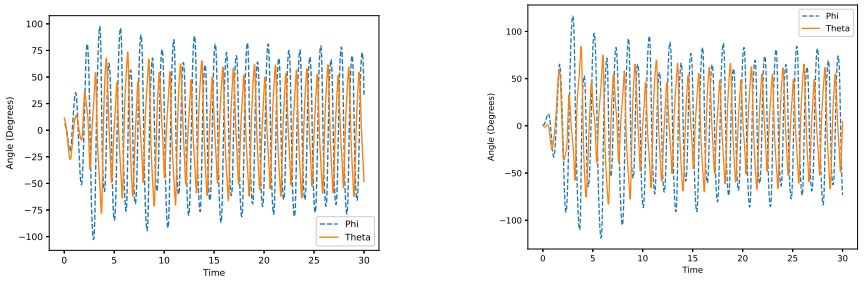
The Penalty matrices used are the same as presented in (2.51), (2.52) and (2.53). Then the derived K matrices are the following:

$$K_1 = \begin{bmatrix} -2.08 & 1.61 \cdot 10^{-4} & 3.53 \cdot 10^{-3} & 8.706 & 3.44 \cdot 10^{-2} & 5.37 \cdot 10^{1} \\ -6.85 \cdot 10^{-4} & -5.27 \cdot 10^{-1} & -2.77 \cdot 10^{1} & 1.79 \cdot 10^{-2} & -1.26 \cdot 10^{2} & 4.76 \cdot 10^{-2} \end{bmatrix} \tag{3.24}$$

$$K_2 = \begin{bmatrix} -3.800 \cdot 10^{0} & 2.068 \cdot 10^{-1} & -1.320 \cdot 10^{2} & ,9.351 \cdot 10^{1} & 2.593 \cdot 10^{1} & 3.945 \cdot 10^{1} \\ 6.020 \cdot 10^{-5} & 1.200 \cdot 10^{-5} & 6.312 \cdot 10^{-2} & ,-5.574 \cdot 10^{-3} & 3.183 \cdot 10^{-3} & -1.752 \cdot 10^{-3} \end{bmatrix} \tag{3.25}$$

$$K_3 = \begin{bmatrix} -2.235 \cdot 10^{2} & 4.590 \cdot 10^{1} & -3.677 \cdot 10^{4} & 5.714 \cdot 10^{3} & 2.635 \cdot 10^{3} & 8.547 \cdot 10^{3} \\ 1.231 \cdot 10^{-9} & -2.884 \cdot 10^{-0} & 7.684 \cdot 10^{-6} & -1.072 \cdot 10^{-6} & 3.779 \cdot 10^{-8} & 5.024 \cdot 10^{-8} \end{bmatrix} \tag{3.26}$$

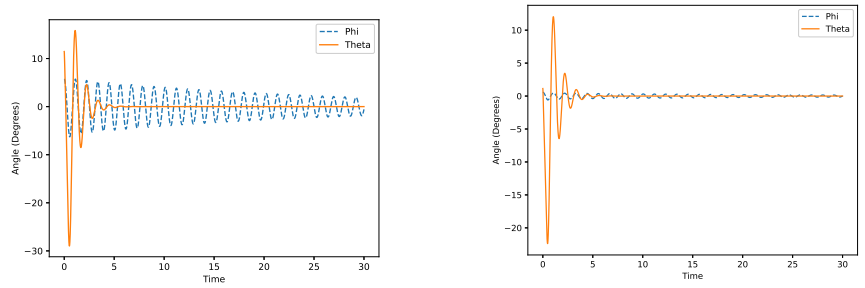
When tested in the simulation we get the following results:



(a) *(b)*

Figure 3.11: *Balancing the unicycle system with LQR with control gain K_1*

The first controller does not stabilize at all, even when staring almost at the equilibrium.



(a) *(b)*

Figure 3.12: *Balancing the unicycle system with LQR with control gain K_2*

The second controller manages to stabilize the system but has a big undesired oscillation in the beginning.

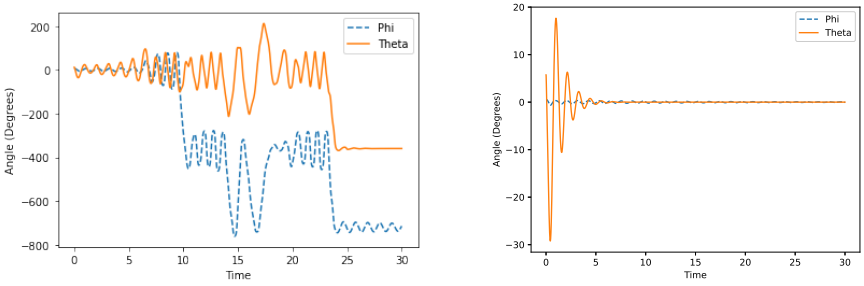


Figure 3.13: *Balancing the unicycle system with LQR with control gain K_3*

The third controller has a hard time stabilizing φ but when starting φ close to the equilibrium it manages to stabilize.

The controllers have different reactions, the first does not manage to stabilize, the second stabilizes but has high oscillation in the beginning, and the third only manages to stabilize if the start position is close to the equilibrium.

3.9 Characterizing region of attraction neighborhood

The linearized model of the system is only valid within a neighborhood around the equilibrium point. One way to find this area is to find the initial states, from which the unicycle can stabilize itself. From these tests, the ROA can be characterized. The ROA for the different models and control matrices are found by trial and error.

3.9.1 Model 1

For K_1 the ROA is for $\varphi = 3.2^{rad} \approx 183°$ and $\theta = 0.65^{rad} \approx 37°$

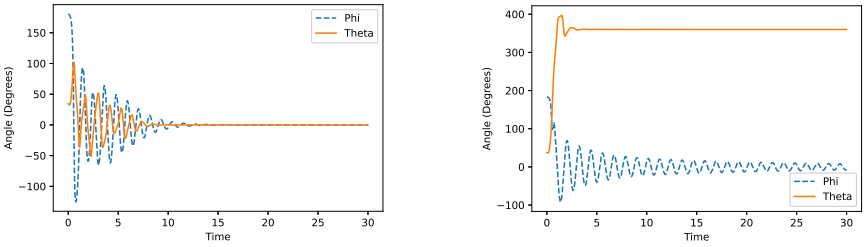


Figure 3.14: *Stable and unstable starting points with control gain K_1*

For K_2 the ROA is for $\varphi = 0.6^{rad} \approx 34°$ and $\theta = 0.25^{rad} \approx 14°$

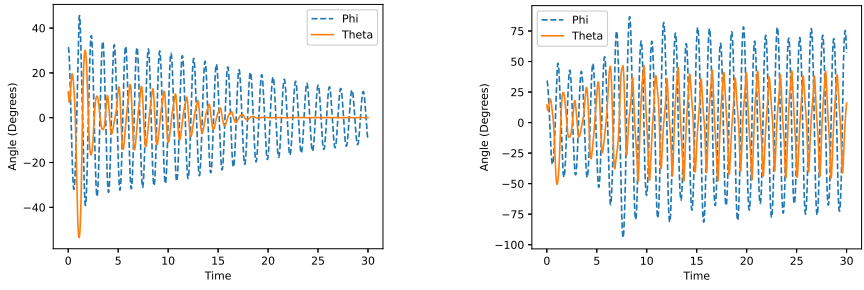


Figure 3.15: *Stable and unstable starting points with control gain K_2*

For K_3 the ROA is for $\varphi = 0.95^{rad} \approx 54°$ and $\theta = 2.4^{rad} \approx 137°$

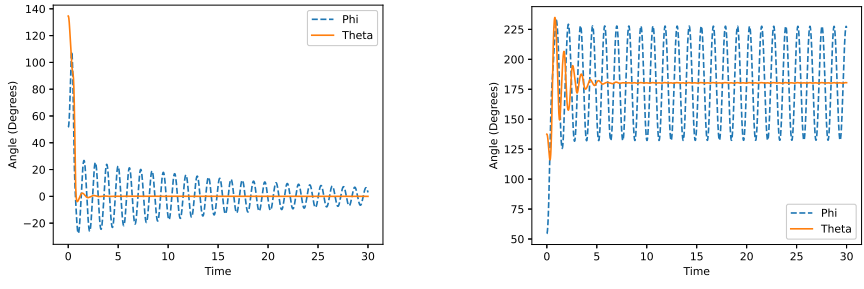


Figure 3.16: *Stable and unstable starting points with control gain K_3*

3.9.2 Model 2

For model 2 the LQR controller the ROA is small and hard to analyze. The behavior is undesired for all starting points but is stabilizing as seen in Figure 3.17.

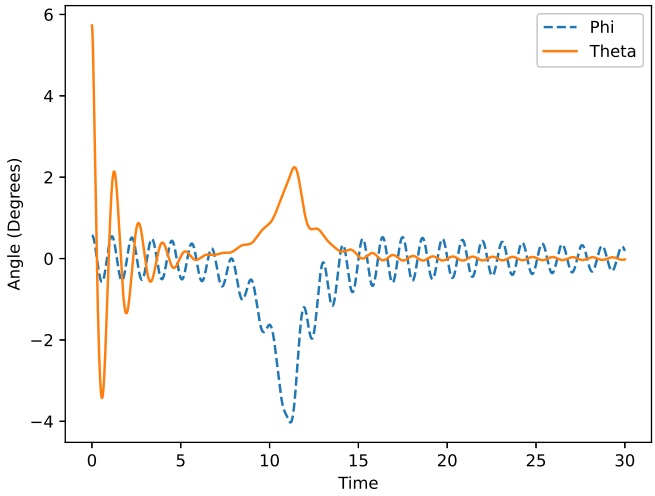
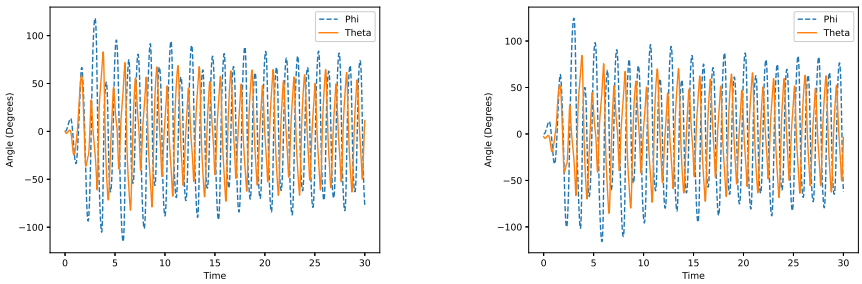


Figure 3.17: *LQR controller based on the second model*

3.9.3 Student model

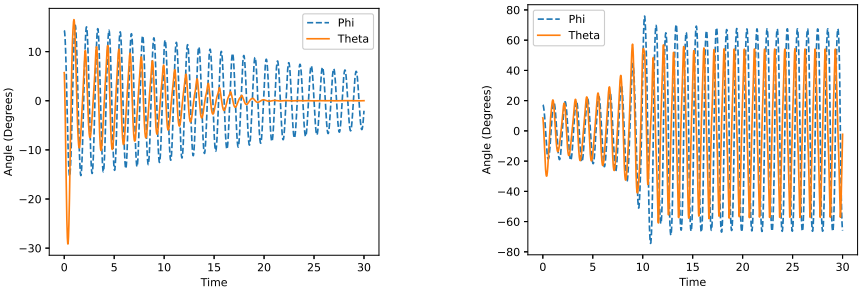
For K_1 the system is not stable even if started from the equilibrium.



(a) ***(b)***

Figure 3.18: *Balancing the unicycle system with LQR with control gain K_1*

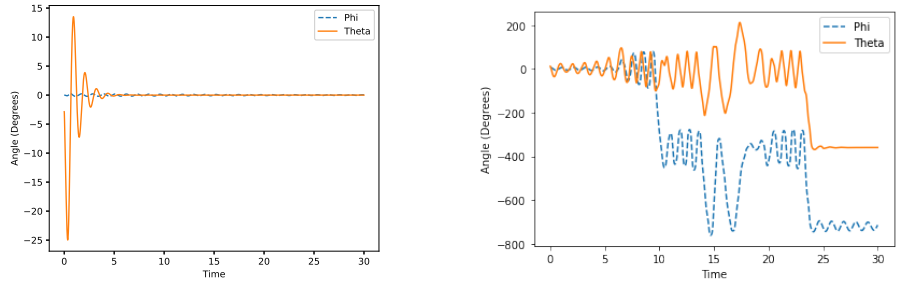
For K_2 the ROA is for $\varphi = 0.3^{rad} \approx 17°$ and $\theta = 0.15^{rad} \approx 8°$



(a) *(b)*

Figure 3.19: *Balancing the unicycle system with LQR with control gain K_2*

For K_3 the ROA is for $\varphi = 0.1^{rad} \approx 6°$ and $\theta = 0.3^{rad} \approx 17°$



(a) *(b)*

Figure 3.20: *Balancing the unicycle system with LQR with control gain K_3*

3.10 Disturbance reaction

To test the disturbance reactions of the control matrices the control input was set to a fixed value for 0.1 seconds, this value was slowly raised until the controller did not manage to stabilize.

3.10.1 Model 1

The disturbance tests for model 1. Only K_1 is used as a control gain for the disturbance tests since it was the gain that showed the best performance in the step tests.

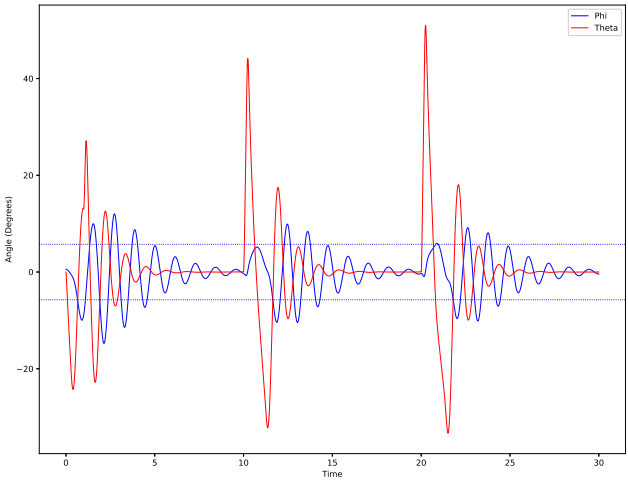


Figure 3.21: *Disturbance tests with LQR controller based on model 1*

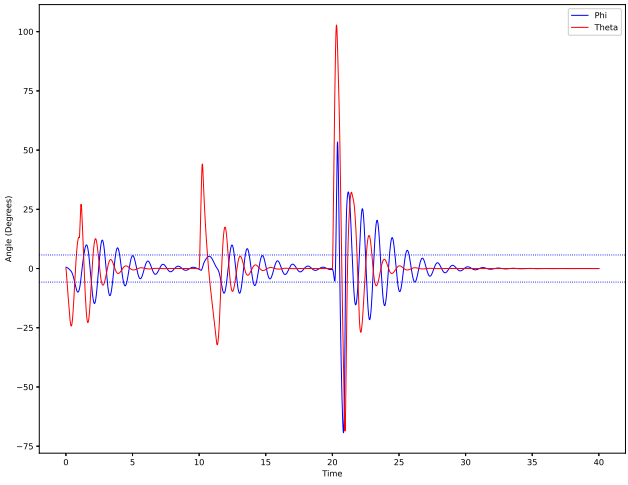


Figure 3.22: *Disturbance tests with LQR controller based on model 1 with larger disturbance*

The LQR controller for model 1 manages to stabilize after all disturbances in the test.

3.10.2 Model 2

Disturbance tests with LQR controller based on model 2

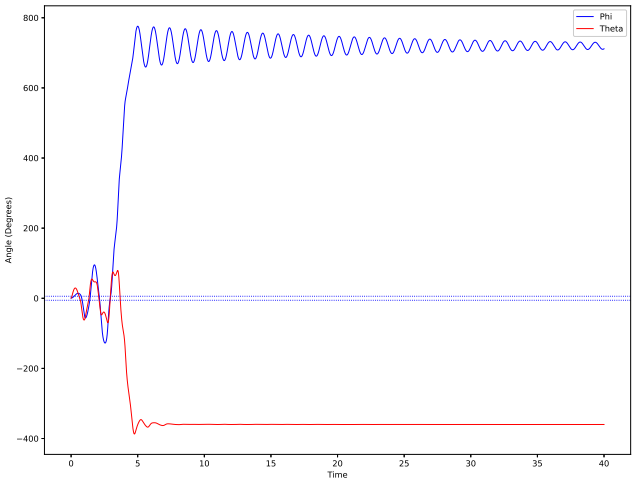


Figure 3.23: *Disturbance tests with LQR controller based on model 2*

The LQR controller for model 2 does not manage to stabilize after the first disturbances in the test.

3.10.3 Student model

Disturbance tests with LQR controller based on the student model. Only K_1 is used as a controller gain for the disturbance tests since it was the gain that showed the best performance in the step tests.

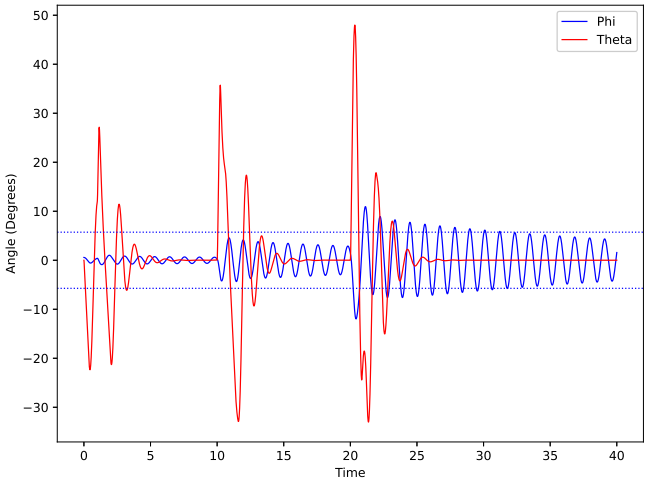


Figure 3.24: *Disturbance tests with LQR controller based on the student model*

The LQR controller for the student model manages to stabilize after all disturbances in the test.

3.11 Comparison

The best model is model 1. This model looks promising and as seen in Figure 3.7 and Figure 3.22 it manages to control the system in a desirable way. The other models have high oscillations and even though those models manage to stabilize in the end the behavior makes them less likely to be able to stabilize in the real world.

3.12 Summary

The learned models are able to control the system when the offset for φ is added, why this offset is needed is discussed in Chapter 7. The controller derived from model 1 with control gain K_1 is performing the best in the simulated environment. The other models are harder to use for control, they all stabilize but take a long time to do so. The student model is able to get to the equilibrium reasonably fast but with too high overshoots to be applied in the real world. An attempt was made to apply this controller on the real-world unicycle but it never was able to control the system to stay at equilibrium.

The control gain K_1 is performing the best for both model 1 and the student

model. For model 2 the behavior is undesired for most starting points and has some effects that are not desired even if it is stable.

4

MPC controller

To compare the models from Chapter 2 and 3 a model predictive controller (MPC) was created based on the models and the performance compared. An MPC controller works by predicting a finite number of future states and optimizing the input to the system to try to minimize the performance measure Raković and Levine [12].

4.1 State prediction

The state space model considered in this chapter is as follows Lorenzen et al. [10].

$$x_{k+1} = Ax_k + Bu_k \tag{4.1}$$

The model in (4.1) can be used for predicting future states. In the following state predictions for 4 consecutive time instants are shown.

$$x_{k+1} = Ax_k + Bu_k \tag{4.2a}$$
$$x_{k+2} = Ax_{k+1} + Bu_{k+1} \tag{4.2b}$$
$$x_{k+3} = Ax_{k+2} + Bu_{k+2} \tag{4.2c}$$
$$x_{k+4} = Ax_{k+3} + Bu_{k+3} \tag{4.2d}$$

By combining the equations in (4.2) the prediction equations can be rewritten as follows:

$$x_{k+1} = Ax_k + Bu_k \tag{4.3a}$$
$$x_{k+2} = A\left[Ax_k + Bu_k\right] + Bu_{k+1} \tag{4.3b}$$
$$x_{k+3} = A\left[A\left[Ax_k + Bu_k\right] + Bu_{k+1}\right] + Bu_{k+2} \tag{4.3c}$$
$$x_{k+4} = A\left[A\left[A\left[Ax_k + Bu_k\right] + Bu_{k+1}\right] + Bu_{k+2}\right] + Bu_{k+3} \tag{4.3d}$$

This can be simplified to:

$$x_{k+1} = Ax_k + Bu_k \tag{4.4a}$$
$$x_{k+2} = A^2x_k + ABu_k + Bu_{k+1} \tag{4.4b}$$
$$x_{k+3} = A^3x_k + A^2Bu_k + ABu_{k+1} + Bu_{k+2} \tag{4.4c}$$
$$x_{k+4} = A^4x_k + A^3Bu_k + A^2Bu_{k+1} + ABu_{k+2} + Bu_{k+3} \tag{4.4d}$$

This equation can be generalized as follows:

$$x_{k+n} = A^n x_k + A^{n-1}Bu_k + A^{n-2}Bu_{k+1} + \cdots + ABu_{k+n-2} + Bu_{k+n-1} \tag{4.5}$$

Utilizing (4.5), the state prediction equations can be presented in compact form as follows:

$$\mathbf{X}_{k+1} = \underbrace{\begin{bmatrix} A \\ A^2 \\ \vdots \\ A^n \end{bmatrix}}_{P} \mathbf{X}_k + \underbrace{\begin{bmatrix} B & 0 & \dots & 0 \\ AB & B & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{n-1}B & A^{n-2}B & \dots & B \end{bmatrix}}_{H} \underbrace{\begin{bmatrix} u_k \\ u_{k+1} \\ \vdots \\ u_{k+n-1} \end{bmatrix}}_{\bar{u}} \tag{4.6}$$

Where:

$$\mathbf{X_{k+1}} = \begin{bmatrix} x_{k+1} \\ x_{k+2} \\ \vdots \\ x_{k+n} \end{bmatrix} \tag{4.7}$$

And:

$$\mathbf{X_k} = \begin{bmatrix} x_k \\ x_{k+1} \\ \vdots \\ x_{k+n-1} \end{bmatrix} \tag{4.8}$$

Where $\overrightarrow{\mathbf{X}}_{k+1}$ is the vector of the predicted states.

4.2 Constraints

When choosing optimal control, constraints on the variables are set. For the unicycle, those constraints are as follows. For the motor voltages:

$$-12 \leq u_d(k) \leq 12 \tag{4.9a}$$
$$-12 \leq u_w(k) \leq 12 \tag{4.9b}$$

For the states, the constraints are set to reasonable stability for the system.

$$|\varphi| \leq 0.6^{rad} = \varphi_{max} \tag{4.10a}$$
$$|\theta| \leq 0.6^{rad} = \theta_{max} \tag{4.10b}$$

Since α_w and α_d are not affecting the controllability of the system, it is not relevant to constrain them.

4.3 Control synthesis

The controller is synthesized by the following quadratic program:

$$\min_{\mathbf{U}} \sum_{j=0}^{n} x_{k+j|k}^T Q x_{k+j|k} + u_{k+j|k}^T R u_{k+j|k} \tag{4.11a}$$

$$\text{subject to} \quad x_{k+j|k} = A x_{k+j-1|k} + B u_{k+j-1|k} \tag{4.11b}$$

$$|\theta| \leq \theta_{max} \tag{4.11c}$$

$$|\varphi| \leq \varphi_{max} \tag{4.11d}$$

$$|u_d| \leq 12 \tag{4.11e}$$

$$|u_w| \leq 12 \tag{4.11f}$$

Where $Q \geq 0$ and $R \geq 0$ are penalty matrices made when implementing the policy and the sequence of predicted control actions, $\mathbf{U}$ is defined as follows.

$$\mathbf{U} = \begin{bmatrix} u_{k|k} \\ u_{k+1|k} \\ \vdots \\ u_{k+n-1|k} \end{bmatrix} \tag{4.12}$$

This quadratic programming problem is solved at every timestep for the horizon of n timesteps forward. The solution is the optimal feedback at the current timestep k. The solver used to solve the problem was Operator Splitting Quadratic Program (OSQP).

4.4 End penalty

To force the MPC controller to find the optimal control policy an end-penalty can be used. The end penalty can be applied with several different methods, Zanon and Faulwasser [19]. For this thesis, a longer time horizon was used to ensure stable closed-loop satisfaction.

4.5 Simulation

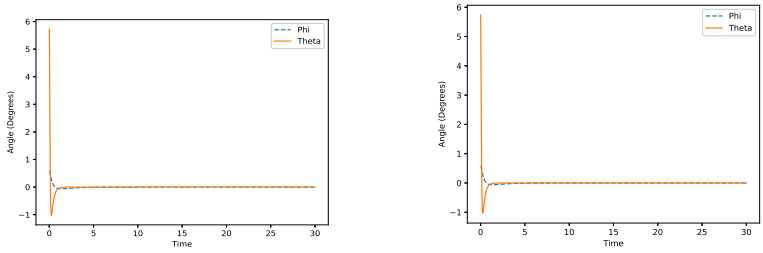
To simulate the MPC controller the simulation environment based on the nonlinear model that was described in Section 2.4 was used.

For each timestep in the simulation, the optimization problem presented in (4.11) was solved with the horizon n. When the optimal $\mathbf{U}$ was found for the n timesteps, the first element from the sequence of predicted control action is given as input to the system and with the new measured state the entire process is repeated for every time instant.

4.6 Physics-based model

For the physics-based model, the discrete state space model derived in Section 3.8 was applied to (4.11). The Q and R matrices are selected as presented in Section 2.6.1. The horizon was selected as 15 steps.

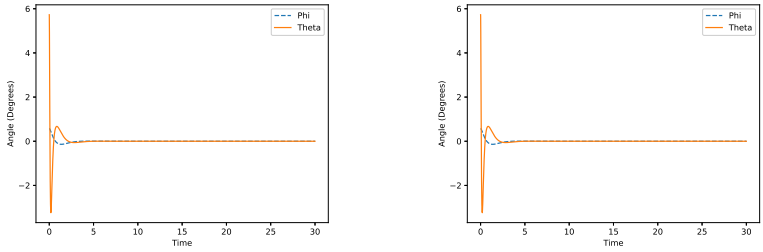
This creates the following plots when starting from $\varphi = 0.01^{rad}$ and $\theta = 0.1^{rad}$



(a) MPC controller *(b) LQR controller*

Figure 4.1: *Balancing the unicycle system with control matrices Q_1 and R_1*

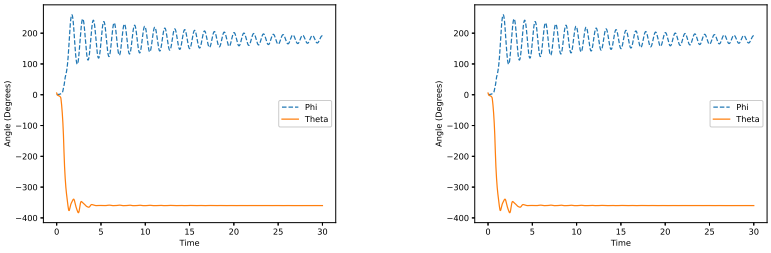
The first controller is coinciding with the LQR controller, stabilizing the system quickly.



(a) MPC controller *(b) LQR controller*

Figure 4.2: *Balancing the unicycle system with control matrices Q_2 and R_2*

The second controller is coinciding with the LQR controller, stabilizing the system quickly.



(a) MPC controller *(b) LQR controller*

Figure 4.3: *Balancing the unicycle system with control matrices Q_3 and R_3*

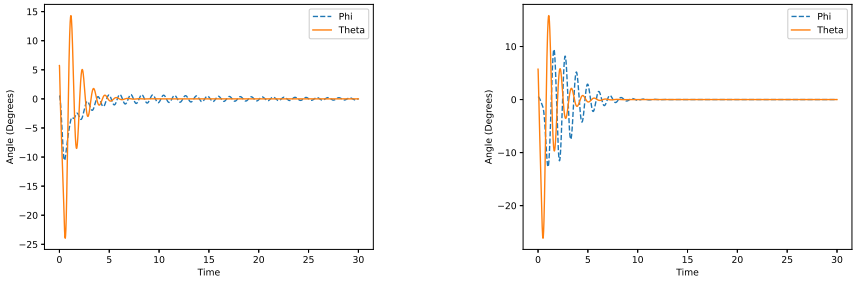
The third controller is not able to stabilize the system, even when starting close to the equilibrium.

4.7 Learned models

This section shows the results of applying the MPC controller based on the different learned models and penalty matrices.

4.7.1 Model 1

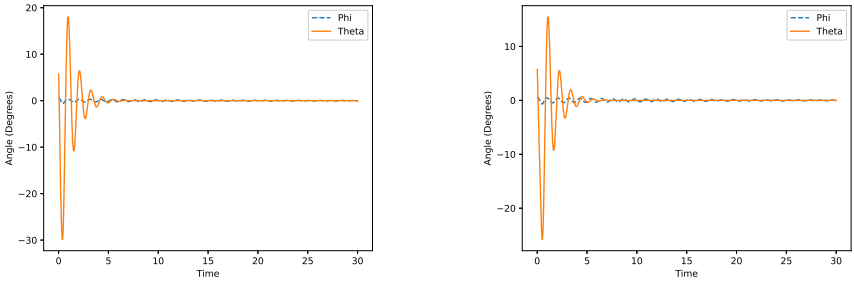
For the learned model, the state space description of the system from the student model is presented in (3.16). The Q and R matrices are selected as presented in Section 2.6.1. The start placement of the unicycle was set to $\varphi = 0.01^{rad}$ and $\theta = 0.1^{rad}$ and a horizon of 150 timesteps as the end penalty. This setup gave the following results:



(a) MPC controller *(b) LQR controller*

Figure 4.4: *Comparison between the MPC controller and LQR controller with control matrices Q_1 and R_1*

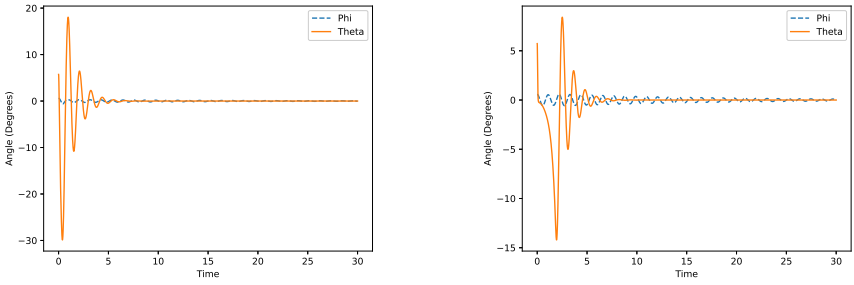
The first controller is able to stabilize the system, it does it quicker than the LQR controller and with a lower oscillation.



(a) MPC controller *(b) LQR controller*

Figure 4.5: *Comparison between the MPC controller and LQR controller with control matrices Q_2 and R_2*

The second controller stabilizes quickly with some oscillations for the θ angle. The controller behaves similar to the LQR controller.



(a) MPC controller *(b) LQR controller*

Figure 4.6: *Comparison between the MPC controller and LQR controller with control matrices Q_3 and R_3*

The third controller is stabilizing the system similar to the second controller. The controller is stabilizing quicker than the LQR controller.

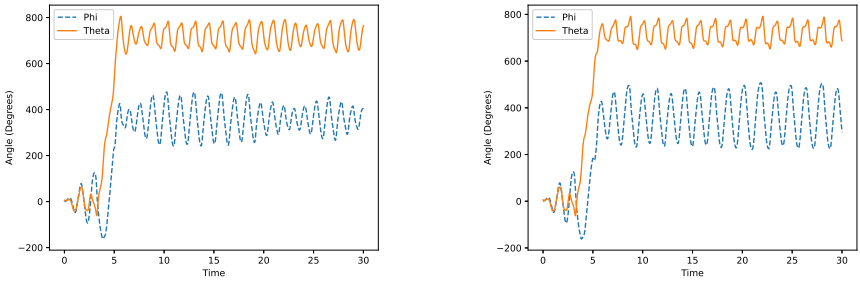
4.7.2 Model 2

The second model could not solve the MPC controllers optimization problem, the controller diverged quickly to a point where the MPC could not find a solution back to the equilibrium. Thus it coincides with the LQR controller from Chapter

3. Therefore model 2 will not be discussed in this chapter, for more see Chapter 6.

4.7.3 Student model

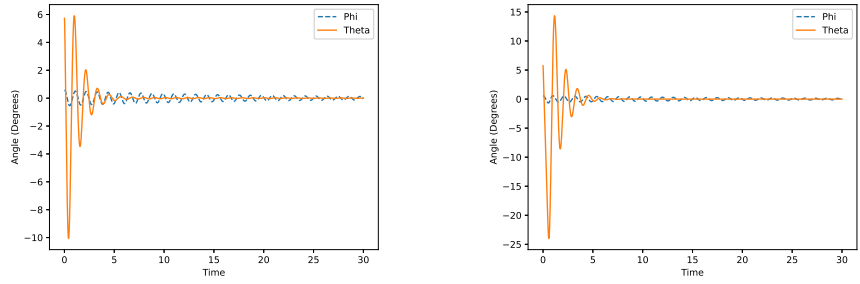
For the learned model, the state space description of the system from the student model is presented in (3.18). The Q and R matrices are selected as presented in Section 2.6.1. The start placement of the unicycle was set to $\varphi = 0.01^{rad}$ and $\theta = 0.1^{rad}$ and a horizon of 150 timesteps as the end penalty. This setup gave the following results compared to the LQR controller: The first controller is not able



(a) MPC controller *(b) LQR controller*

Figure 4.7: *Comparison between the MPC controller and LQR controller with control matrices Q_1 and R_1*

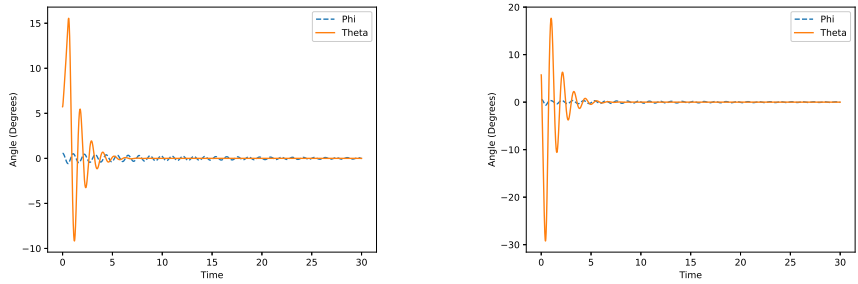
to stabilize the system.



(a) MPC controller *(b) LQR controller*

Figure 4.8: *Comparison between the MPC controller and LQR controller with control matrices Q_2 and R_2*

The second controller is stabilizing the system quicker and with lower oscillation than the LQR controller.



(a) MPC controller *(b) LQR controller*

Figure 4.9: *Comparison between the MPC controller and LQR controller with control matrices Q_3 and R_3*

The third controller is stabilizing the system quicker and with lower oscillation than the LQR controller.

4.8 Summary

The MPC controller works well for controlling θ but is not as good at controlling φ for all models in the simulation. The best controller is the Student model with control matrices (2.52). For all models, there is a small oscillation in φ when it is close to tracking the desired set-point, this is discussed in Chapter 6.

5

Control by reinforcement learning

The second method of learning a control method is reinforcement learning (RL) To compare this method with the previous models and controllers an off-the-shelf method was used.

5.1 Reinforcement learning

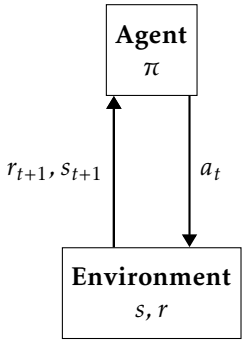
Using reinforcement learning to learn control policies has become popular over the later years. The methods have become more sophisticated and easier to implement which makes it a desirable option. Off-the-shelf methods have become easy to implement and the knowledge needed to use them is low, many thanks to companies like Open-Ai.

RL works by letting an agent discover the world around it by enabling it to take an action a_t at time step t and at the next time step $t + 1$ giving it a reward r_{t+1} and the state s_{t+1} of the agent according to the action taken.
There are several methods that can be used in RL based on the actions that are taken and how the policy is decided. In this thesis, the policy gradient method is used.

5.1.1 Policy gradient method

The policy gradient method is a technique to optimize the policies with respect to the reward. This creates a faster gradient descent toward the optimal policy for the system. The agent learns the policy directly from a parameterized function with respect to Θ instead of the value function of the action space. Where Θ represents the unknown parameter of the system.

$$\pi(a|x;\Theta) = Pr\{A(t_i) = a|S(t_i) = x, \Theta(t_i) = \Theta\} \tag{5.1}$$



Where $\pi(a|x;\Theta)$ is the probability that action a is taken at timestep t_i, given that the agent is at state x at time step t_i with parameter Θ.

This gradient method learns the policy parameter from the gradient of the scalar performance measure, $L(\Theta)$. Θ is updated by finding the best $\Delta L(\Theta)$ which gives the best return to the policy Sutton and Barto [18].

These methods are especially good for big action spaces, where the possible actions can go towards infinity, since the policy gradient method learns the probability distribution statistics.

5.1.2 Reward

The reward r_{t+1} is the reward the agent gets after taking the action a_t at timestep t. This creates the feedback the agent gets from the environment when taking an action. The reward functions can look very different and depend on how the designer of the system wants the agent to behave. This is the primary feedback the designer can give to the agent.

5.1.3 Environment

The environment has the same importance as the data had in Chapter 3. Because it is the environment that decides what happens to the agent after it has made its action. The environment used to train the policy is based on the nonlinear model that was created in Chapter 2.

5.2 Proximal Policy Optimization

Proximal Policy Optimization (PPO) is a policy gradient method that Open AI created, PPO is a general RL method that combines the stability of Trust Region Policy Optimization (TRPO) with a simpler and faster solver method, Schulman et al. [14]. This method has been proven to work well in many applications, Jin

and Wang [5] presented how PPO can be used to find the optimal path for mobile robots and Jonsdottir and Petersson [6] presented how the method works for unicycles.

5.2.1 Observation and action spaces

To let the system understand which real-world scenarios are the observation and action spaces are set to mirror the real-world setup. The output is the motor voltages for the agent, the unicycle, which corresponds to the action space of:

$$u_d \in [-12, 12] \tag{5.2a}$$
$$u_w \in [-12, 12] \tag{5.2b}$$

Similarly, the observation space is the states the agent can reach. The observation space is set by reason and testing to limit the agent to enter states from which it can not stabilize. These are set to:

$$\dot{\alpha}_w \in [-84, 84] \tag{5.3a}$$
$$\dot{\alpha}_d \in [-450, 450] \tag{5.3b}$$
$$\dot{\varphi} \in [-100, 100] \tag{5.3c}$$
$$\dot{\theta} \in [-100, 100] \tag{5.3d}$$
$$\varphi \in [-25, 25] \tag{5.3e}$$
$$\theta \in [-25, 25] \tag{5.3f}$$

If the agent reaches any of the limits of the observation space the states are reset to their origin and the training restarts.

5.2.2 Policy method

The policy gradient method used in PPO is defined as follows.

$$L^{PPO}(\Theta) = \hat{E}_t \left(\log \pi_\Theta(a_t|s_t) \hat{A}_t \right) \tag{5.4}$$

Where $\hat{E}_t(...)$ is the expectation function over a finite batch of samples, $\hat{A}_t$ is the estimator of the advantage function at timestep t and π_Θ is a stochastic policy Schulman et al. [14].

5.2.3 Reward

The reward is the agents' input on if it is performing correctly or badly, the feedback from the environment. By changing the reward the agents' behavior can differ and undesirable behavior can be created.

The design of the reward function is by trial and error, the reward function used for this thesis is based on the difference between the worst case of the state $\bar{x}$ and the current state x:

$$\sum_{i=1}^{6}\left(1-\left(\frac{x_{i,k}}{\bar{x}_i}\right)^2\right) \tag{5.5}$$

Where i is for the 6 different states. When the agent leaves the observation space a reward of -1 is given.

5.2.4 Simulation

The training of the agent is made in a simulated world created by a toolkit called *Gym* created by Open AI. This toolkit is used to create the environment where the agent is trained and the PPO models' structure. The testing of the control method is made in this environment.

5.2.5 Training

The training consisted of $1 \cdot 10^7$ timesteps where the agent was let to learn its policy. The training was made with a randomized starting point from where it had to learn how to stabilize and with a constant learning rate of 0.004.

The starting states are set to a random state within the observation space and every time the agent goes outside the observation space it is given a reward of -1 and the agent is set to a random starting state to start making actions again.

5.3 Controller performance

The resulting controller performance is shown in Figure 5.1 and 5.2.

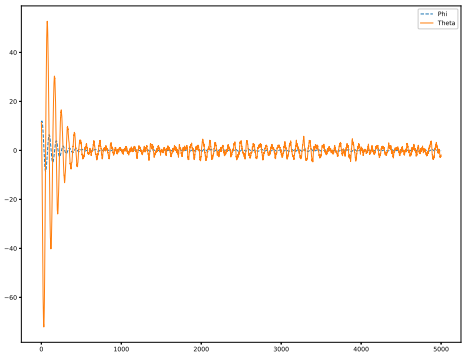


Figure 5.1: *RL controller*

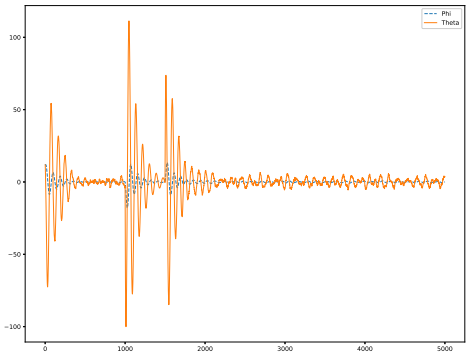


Figure 5.2: *RL controller step test*

When compared to the other control policies it has the same problem of understanding and controlling θ, but manages to control φ reasonably. The biggest difference between the RF controller and the others is the oscillation behavior. The other controllers have a sinusoidal oscillation compared to the RF controller which is harder to describe mathematically.

5.4 Summary

The RL controller performs well, but the behavior is harder to understand and describe. It has the same problem of understanding the θ angle as the other controllers but manages to stabilize.

6 Discussion

The controllers tested in this thesis had varied behaviors from very good to not working. In this chapter, the differences and reasons why they behaved as they did will be discussed.

6.1 Control based on the physics-based model

The controllers created from the physics-based model worked well and controlled the system in most cases within a reasonable ROA.

6.1.1 LQR controller

The physics-based LQR controller controls the simulated system stably and with low oscillations. Depending on which penalty matrices are used, the ROA of φ varies. However, when applied to the real-world system, it is θ that is harder for the controller to stabilize. The problems in θ that are not seen in the simulations are believed to arise from how the world affects the system. One example is the gyroscopic and aerodynamic effects created by the disc. This is created by the discs' high speed.

The benefit of the physics-based model is that if we have a good knowledge of mechanical systems, a model that can describe the system in a way that makes it possible to control the system can be constructed quite fast. But for many systems, this can be done faster and better with a simple PID controller. An attempt was made to implement a PID for the unicycle, but a simple PID was not able to control the unicycle. The theory why it did not work is that the states are too coupled. Since there are similar unicycles that can be controlled by PIDs but they

have more weight on the disc than the one used in this thesis and because of that, they do not need to accelerate the disc as fast as the unicycle used in this thesis.

6.1.2 MPC controller

The physics-based MPC controller controls the system in the same way as the LQR controller. It is observed that the performance with the MPC coincides with that generated by the LQR controller.

The controller using (2.53) becomes very good at controlling θ and manages to stabilize it directly, but it has a hard time stabilizing φ. This is because of how the penalty matrices are defined, (2.53) gives a large focus on controlling θ with the effect that it is not as focused on φ. The controller is in general a lot better than the LQR controller with the same penalty matrices that were not able to stabilize from very small starting angles.

6.2 Control based on the learned model

The controllers created by the learning-based models worked with satisfactory performance. Some behaviors were caused by flaws in the model which are discussed more in the following subsections.

All the controllers have a small acceleration around the equilibrium and do not fully stabilize in the time frame tested. This is believed to happen due to having an imperfect model since they are not able to describe φ correctly and it takes a very long time to stabilize fully.

6.2.1 LQR controller

The different models created very different results when used for the LQR controller. In general, the controllers were good at controlling θ and performed worse for φ, this is believed to be a consequence of the opposite problem for the physics-based model where it was good at controlling φ and worse for θ. Because the physics-based model controller was used to collect the data, the data collection became more diverse for the movement in the θ angle. This illustrates one possible direction for improvement when collecting the dataset to train the models. One solution could have been to add a greater distribution of noise in u_w when collecting the data.

A second possible deviation for improvement is that the models created had to be offset with π radians for φ. This is believed to come from when the model is trained, it has no frame of reference and therefore sets its φ reference vector "towards the ground". So the models believe that it is supposed to stabilize around $\varphi = \pi$ and is behaving like is expected except for the oscillations that are produced by the controller.

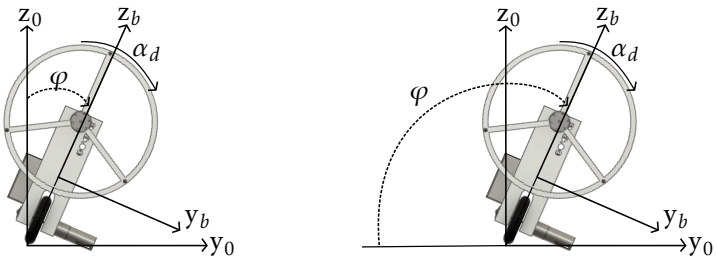


Figure 6.1: *The difference in how the models think φ is defined.*

One of the main drawbacks of learning-based models is that when errors like the one encountered with the φ offset occur, it is hard to troubleshoot and fix the error. To be able to do this fast, a high knowledge base is required. In my case, this error was found late, which made it hard to find a good solution to correct this error in the models.

The collection of the data is another drawback of learning, the models can only be as good as the data that it is given to learn. To be able to learn better a larger dataset with more white noise in the controller is needed, but to do this a highly controlled environment is needed.

6.2.2 MPC controller

The MPC controllers created from the learned system models are in most cases improving the LQR controller created by the same system description. They are either better at controlling φ, as seen in Figure 4.4, or faster at finding the equilibrium, as seen in all controllers created by the student model. The two controllers that are not improved are the controllers created by Model 1 with control matrices (2.51) and (2.52), where the initial oscillation is bigger but is as fast to stabilize as the LQR model. This is believed to come from the model behavior. Model 1 has a behavior with high oscillations, so it has not learned how the system works correctly from the input and is expecting the system to have lower reactions from the input it suggests. This is exaggerated when it is used in the MPC controller since it gives a large weight to the model and therefore applies a higher input than when the other models are used.

The student models work better and are improving from the LQR controller, it manages to stabilize quicker and with lower oscillations. Here it is possible to see how a learned model with an MPC controller can work and it is potential. The MPC can take the model and understand how the system is going to react and

plan accordingly. it is still far from the performance of the physics-based model and shows the same behavior as Model 1 discussed above, it does not have the full understanding of how the system will react to the input.

Model 1 shows another problem with the dataset, as it did not have enough data to understand how the system behaved with larger inputs. It looks to give a larger input than necessary or hold that input for too long, so it will overshoot the equilibrium.

6.3 Control based on reinforcement learning

The RL controller shows the same behavior as the NN-created models where it has a lot harder time controlling θ. This is believed to come from the lack of data on the system behavior for θ. This is the same problem believed to create the oscillations for the learning-based model, where more noise in θ was needed when collecting data.

The second drawback is that the behavior is harder to describe mathematically and it is hard to know how the controller will behave in the scenarios it has not been tested in. So the model needs to be tested more, to say that it can be stabilized. It can be seen from the test applied in this thesis that it looks promising.

6.4 The different penalty matrices

The reason why the three different matrices were used was their performance between the different models. Matrix nr.1, (2.51), were performing the best for the physical model, matrix nr.2, (2.52), performed the best for the student model, and matrix nr.3, (2.53), performed the best for Model 1 in the initial tests. The different matrices also showed the different behaviors from the different models, that either needed to be reduced or raised.

6.5 System identification

The NN approach for the system identification problem was chosen because of its simplicity and how easy it is to scale to a larger NN. Since this model structure can make a good model of the dynamics of the system with limited data, and create a linear description of the model which when used to create a control policy works desirably.

The reason why the one-layer NN was chosen was that the multi-layer NN is well documented and known for its performance and since this thesis is focused on trying to make the simplest model to control the system the best and the one-layer NN was a good way to show that.

6.6 Results

All the models and controllers have their pros and cons. The best controller and model derived in this thesis is the baseline controller made from the physics-based model and an LQR controller. The MPC controller shows high potential but its problems to control φ show that it is not better than the control matrices that were tested. There might be some control matrices that can stabilize φ better but they were not found for this thesis.

The big drawback of ML models was found with the offset for φ, this shows how it is hard to calculate how the ML model will work and understand the problem.

7

Conclusions

Even if the learned models were not able to become good enough to control the unicycle in the real world the models show good promise. The best learned model was able to become close to the physics-based model. With more time I think the learning models can outperform the physics-based model. Especially if the student network learning approach is used.

7.1 Answers to the problem formulation

How does a generic controller work on the system?
The generic controller works well for the system, it is able to stabilize the system both in simulation and in the real world.

Is it possible to improve the model and controller with a neural network?
It is not shown in this thesis that it is possible to improve the physics-based model with the one learned with a neural network. Although it has shown potential, more work needs to be done to be able to improve the model for the control systems.

If the LQR controller and MPC controller are compared it can be seen that in many cases the MPC controller is only slightly better. This shows the performance of the learned model, even with an MPC controller that is trying to optimize the controller of the system, when given a strong computer it had a compute time of up to 1 hour for a 30 second test. The simple LQR controller on the other hand is still able to understand the dynamics of the system in a way that is close to the MPC with a computation time of 5 seconds.

Can a model predictive controller further improve the performance with operational constraints being considered implicitly?
It can, but only for the right models. The difference in the performance of the MPC is shown by the different models. With a good model, the MPC controller works better however, it will also enhance the faults in the models as seen in the behaviors of model 1.

What is the performance difference between the control policies derived from the different models?
The performance differs from how the different models understand the system. This is reflected more or less depending on the controller, the MPC controller works better if the model is better but it also shows if the model has any problems. The LQR controller can reduce these problems since it is a simpler controller and is not as directly coupled with the model.

7.2 Future improvements

The result of this thesis opens up future improvements, to get the learned model to perform better than the physics-based model.

The first point to improve on is the dataset for the learned model. There are several methods to improve the dataset i) Bigger dataset where the unicycle has been left to balance for longer, this will let the NN have more data points to learn the system model. ii) To do this the most efficient way a wireless connection between the unicycle and a laptop should be made. Then the unicycle can run for longer by itself. For the data collected in this thesis, the connection between the unicycle and the laptop was via wire. iii) More white noise could be added to the controller when collecting the data. When the data was collected the environment was not optimal, I had to control both the unicycle and, the computer collecting the data. This limited the possibility to push the unicycles limits. During more controlled runs more white noise could be added and create a more diverse dataset. iv) Collect data from the simulated environment. By collecting data from the simulated environment there is a possibility to get data at points where it is hard to get the real-world unicycle safely. To be able to do this a fuller understanding of the system can be given to the learning model.

The methods for learning the model that is presented in this thesis are only for linear descriptions of the system. By applying non-linear activation functions to the NN, a nonlinear description of the system could also be realized.

Appendix

A

Constants for mathematical description

The inertia constants for the components are

Component	Disc	Wheel	Body
I_{xx}	$5.792 \cdot 10^{-3}$	$3.612 \cdot 10^{-4}$	$4.008 \cdot 10^{-2}$
I_{xy}	-0.000	-0.000	$-1.286 \cdot 10^{-4}$
I_{xz}	$2.140 \cdot 10^{-7}$	0	$2.148 \cdot 10^{-3}$
I_{yx}	$-7.957 \cdot 10^{-9}$	-0	$-1.286 \cdot 10^{-4}$
I_{yy}	$6.897 \cdot 10^{-3}$	$6.888 \cdot 10^{-4}$	$3.025 \cdot 10^{-2}$
I_{yz}	$1.580 \cdot 10^{-6}$	0	$6.148 \cdot 10^{-3}$
I_{zx}	$2.140 \cdot 10^{-7}$	0	$2.148 \cdot 10^{-3}$
I_{zy}	$1.580 \cdot 10^{-6}$	0	$6.211 \cdot 10^{-3}$
I_{zz}	$6.891 \cdot 10^{-3}$	$3.612 \cdot 10^{-4}$	$1.183 \cdot 10^{-2}$

Table A.1: *The inertia of the unicycle*

The constants used to describe the system are:

Constant	Description	Value
r_w	Radius of the wheel	0.072m
l_{wb}	Length between the center of mas for the wheel and body	0.1292 [m]
l_{wd}	Length between the center of mas for the wheel and disc	0.3006 [m]
m_w	Mass of the wheel	0.272987 [kg]
m_b	Mass of the body	0.25746 [kg]
m_d	Mass of the disc	2.168873 [kg]
K_w	Torque constant of the wheel motor	1.0709 [Vs]
K_d	Torque constant of the disc motor	0.2622 [Vs]
$K_{a,w}$	Armature resistance of the disc motor	4.8 [Ω]
$K_{a,d}$	Armature resistance of the disc motor	0.6 [Ω]

Table A.2: *The constants of the physics-based model of the unicycle*

Bibliography

[1] Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer New York, NY, 2006. ISBN 9781493938438.

[2] Gunnar Blom, Jan Enger, Gunnar Englund, Jan Grandell, and Lars Holst. *Sannolikhetsteori och statistikteori med tillämpningar.* Studentlitteratur AB, 2017. ISBN 9789144123561.

[3] Pooja Gautam. System identification of nonlinear inverted pendulum using artificial neural network. *2016 International Conference on Recent Advances and Innovations in Engineering (ICRAIE), Recent Advances and Innovations in Engineering (ICRAIE), 2016 International Conference on*, pages 1 – 5, 2016. ISSN 978-1-5090-2807-8.

[4] Torkel Glad and Lennart Ljung. *Reglerteori : flervariabla och olinjära metoder.* Studentlitteratur, 2003. ISBN 9144030037.

[5] Xin Jin and Zhengxiao Wang. Proximal policy optimization based dynamic path planning algorithm for mobile robots. In *Electronics letters*, volume 58, pages 13 – 15, 2022. URL https://login.e.bibl.liu.se/login?url=https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,uid&db=edsbl&AN=vdc.100145682935.0x000001&lang=sv&site=eds-live&scope=site.

[6] Maria Helga Jonsdottir and Filip Petersson. Deep learning and regular control methods, 2019. URL https://hdl.handle.net/20.500.12380/300526.

[7] Jaeoh Lee, Seongik Han, and Jangmyung Lee. Decoupled dynamic control for pitch and roll axes of the unicycle robot. *IEEE Transactions on Industrial Electronics*, 60(9):3814–3822, 2013. doi: 10.1109/TIE.2012.2208431.

[8] Lennart Ljung. *System identification : theory for the user.* Prentice-Hall information and system sciences series. Prentice Hall, 1999. ISBN 0136566952.

[9] Lennart Ljung and Torkel Glad. *Modeling and identification of dynamic systems.* Studentlitteratur, 2016. ISBN 9789144116884.

[10] M. Lorenzen, F. Dabbene, R. Tempo, and F. Allgower. Constraint-tightening and stability in stochastic model predictive control. *IEEE Transactions on Automatic Control, Automatic Control, IEEE Transactions on, IEEE Trans. Automat. Contr*, 62(7):3165 – 3177, 2017. ISSN 0018-9286. URL https://login.e.bibl.liu.se/login?url=https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,uid&db=edseee&AN=edseee.7733074&lang=sv&site=eds-live&scope=site.

[11] Olalekan Ogunmolu, Xuejun Gu, Steve Jiang, and Nicholas Gans. Nonlinear systems identification using deep dynamic neural networks. 2016.

[12] Saša V. Raković and William S. Levine. *Handbook of model predictive control*. Control Engineering. Springer International Publishing, 2019. ISBN 9783319774893. URL https://login.e.bibl.liu.se/login?url=https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,uid&db=cat00115a&AN=lkp.977327&lang=sv&site=eds-live&scope=site.

[13] Murat Sahin, Halil Bulbul, and Ilhami Colak. Position control of a DC motor used in solar panels with artificial neural network. pages 487–492, 12 2012. ISBN 978-1-4673-4651-1. doi: 10.1109/ICMLA.2012.216.

[14] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. 2017.

[15] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. 2017.

[16] Indrazno Siradjuddin, Budhy Setiawan, Ahmad Fahmi, Zakiyah Amalia, and ROHADI Erfan. State space control using LQR method for a cart-inverted pendulum linearized model. 17:119–126, 02 2017.

[17] Jonas Sjöberg, Håkan Hjalmarsson, and Lennart Ljung. *Neural networks in system identification*. LiTH-ISY-R: 1622. Univ., 1994. ISBN 1400-3902.

[18] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning : an introduction*. Adaptive computation and machine learning. The MIT Press, 2018. ISBN 9780262039246. URL https://login.e.bibl.liu.se/login?url=https://search.ebscohost.com/login.aspx?direct=true&AuthType=ip,uid&db=cat00115a&AN=lkp.1079275&lang=sv&site=eds-live&scope=site.

[19] Mario Zanon and Timm Faulwasser. Economic mpc without terminal constraints: Gradient-correcting end penalties enforce asymptotic stability. *Journal of Process Control*, 63:1–14, 2018. ISSN 0959-1524. doi: https://doi.org/10.1016/j.jprocont.2017.12.005. URL https://www.sciencedirect.com/science/article/pii/S0959152417302275.

Index