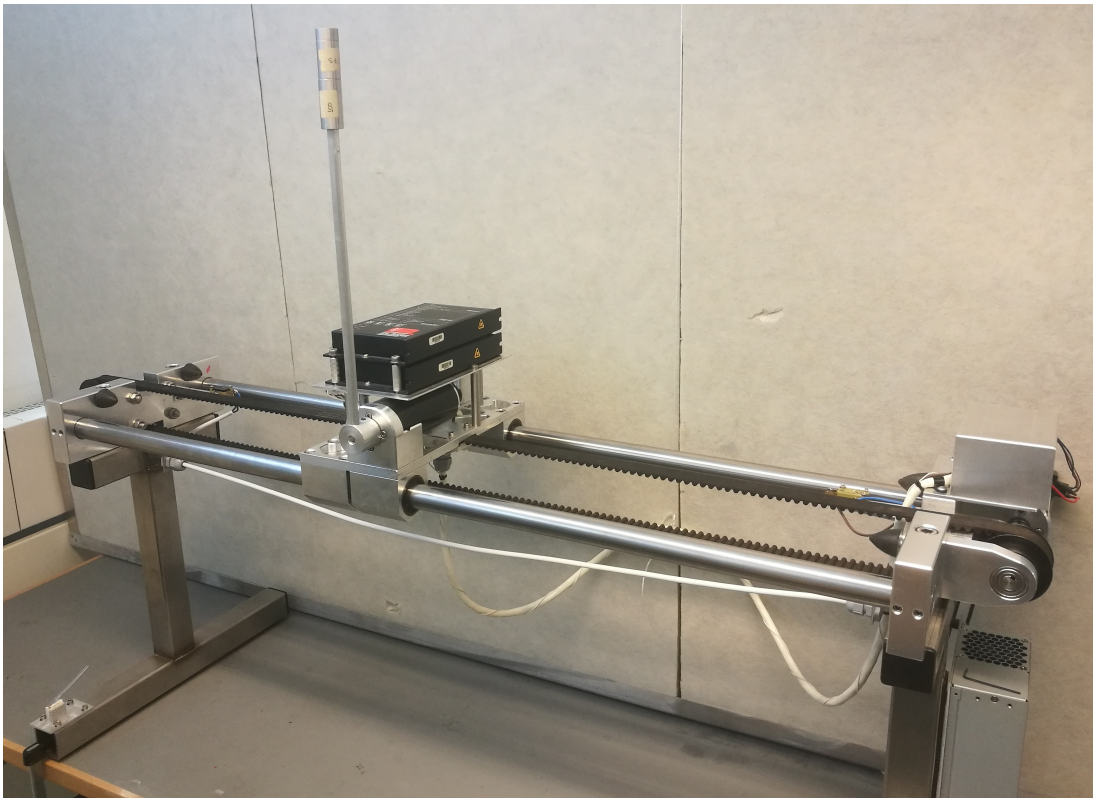
Non-linear Control and Machine Learning on an Inverted Pendulum on a Cart



Group 1032
Master Thesis
Control & Automation
Aalborg University
Spring 2018

AALBORG UNIVERSITY
STUDENT REPORT

Second year of study
Automation and Control
Fredrik Bajers vej 7
9220 Aalborg
www.aau.dk

Title:
Non-linear Control and Machine Learning on an Inverted Pendulum on a Cart

Project period:
Master Thesis,
Spring semester 2018

Project group:
1032

Participants:
Jesper Hove Jørgensen
Jonas Ørndrup Nielsen

Supervisors:
Jan Dimon Bendtsen
John-Josef Leth

Printed copies: 5

Number of pages: 111

Date of hand in 7/6-2018

This thesis covers the development of non-linear control and machine learning control with application to an inverted pendulum on a cart.
First, a dynamical model of the system is derived, which mimics the real system satisfactorily. A program structure is implemented, allowing for user interaction and shifts between controllers. Furthermore an extended Kalman filter is developed to estimate the non-measurable states.
The non-linear controller is designed as a sliding mode controller, which is able to balance the pendulum well, robust to parameter uncertainties, with good disturbance rejection and the capability of reference tracking. However with a steady state error.
The machine learning control is based on the reinforcement learning methods Q-learning and SARSA. Several parameter combinations and function approximation methods have been attempted for these methods, where the first converges to solutions working in simulation. Problems exist when applying these to the real system, where balancing only is possible for a couple of seconds. The methods tried are thereby deemed unfitting for control of the setup available in this thesis.

Jesper H. Jørgensen

Jonas Ørndrup

Preface

This thesis has been created by Jesper Hove Jørgensen and Jonas Ørndrup Nielsen. The thesis is made on the 4^{rd} semester as the master thesis of the education, Control and Automation at Aalborg University. The thesis is made in cooperation with Aalborg University. Aalborg University has contributed with materials for the thesis.

The thesis is intended for people with a background knowledge corresponding to a 4^{rd} semester master student at Control and Automation, Aalborg University. The thesis has been made using the programming languages C++ and MATLAB. The MATLAB scripts used in the thesis are available as digital attachment, as well as the C++ code running on the test setup. All graphical elements in the thesis are made by the group, otherwise, a reference to the source is stated in the figure text.

Sources are indicated with [index] and can be found in the bibliography list at the given index. After the source, the relevant chapter, section, page number or equation reference is indicated with (place), if appropriate.

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
AAU	Aalborg University
AC	Actor-critic
AI	Artificial intelligence
ANN	Artificial neural network
ARM	Advanced RISC Machines
DAC	Digital to analog converter
DC	Direct current
DQN	Deep Q-network
EEPROM	Electrically erasable programmable read-only memory
EKF	Extended Kalman filter
E.g.	Exempli gratia
FA	Function approximation
FPU	Floating-point unit
GD	Gradient descent
GP	Genetic programming
IC	Integrated circuit
Iff	If and only if
I.e.	Id est
I.I.D.	Independent and identically distributed
I/O	Input/output
IPC	Inverted pendulum on a cart
KF	Kalman filter
LIP	Linear in the parameters
LQR	Linear-quadratic regulator
MA	Moving average
MC	Monte Carlo
MCU	Microcontroller unit
MDP	Markov decision process
ML	Machine learning
MLC	Machine learning control
MSE	Mean squared error
NID	Normally identically distributed
NN	Neural network
ODE	Ordinary differential equation
PID	Proportional-integral-derivative
PILCO	Probabilistic Inference for Learning COntrol
RHP	Right half plane
RL	Reinforcement learning

RNN	Recurrent neural network
SARSA	State-action-reward-state-action
SDE	Stochastic differential equation
SGD	Stochastic gradient descent
SMC	Sliding mode control
SRAM	Static random-access memory
TD	Temporal difference

Symbols

Symbol	Description	Units
A	Area of frontal surface	m^2
a_k	Action at time k	1
B_v	Viscous friction coefficient	$\frac{Ns}{m}$
$B_{v,c}$	Viscous friction coefficient of cart	$\frac{Ns}{m}$
$B_{v,p}$	Viscous friction coefficient of pendulum	$\frac{Ns}{rad}$
$B_{v,p,kalman}$	Viscous friction coefficient of pendulum for Kalman filter	$\frac{Ns}{rad}$
bit_{DAC}	DAC bit value	1
C_D	Drag coefficient	1
d_{pulley}	Diameter of the pulley attached to the motor	m
F	Force applied to cart	N
F_a	Force applied from motor	N
F_B	Friction force	N
F_{B_c}	Friction force of cart	N
F_{B_p}	Friction force of pendulum	N
F_c	Coulomb friction force	N
F_d	Aerodynamic drag force	N
F_f	Combined friction force	N
F_k	Jacobian of system equation	1
F_{leftover}	Leftover force	N
F_{net}	Total net force in system	N
F_s	Static friction force	N
F_v	Viscous friction force	N
$\mathcal{F}_s$	Static friction coefficient	N
$\mathcal{F}_c$	Coulomb friction coefficient	N
$\mathcal{F}_{c,c}$	Coulomb friction coefficient of cart	N
$\mathcal{F}_{c,c_{neg}}$	Coulomb friction coefficient of cart in negative direction	N
$\mathcal{F}_{c,c_{pos}}$	Coulomb friction coefficient of cart in positive direction	N
$\mathcal{F}_{c,p}$	Coulomb friction coefficient of pendulum	N
g	Gravitational constant	$\frac{m}{s^2}$
H	Discrete output matrix	1
H_k	Jacobian of output equation at time k	1
I	Identity matrix	1
$I_{attachment}$	Inertia of motor shaft attachment	kgm^2
I_{bob}	Inertia of the bob	kgm^2

i_a	Applied current	A
$\bar{i}_a$	Current operating point	A
K	Kalman gain	1
K_p	Proportional control gain	1
K_t	Motor constant	$\frac{\text{Nm}}{\text{A}}$
k	Discrete time index, or in chapter 5 the sliding manifold gains	1
k_{tanh}	Constant for tanh approximation	1
l	Length from pendulum pivot point to middle of bob	m
l_{cart}	Length of the cart	m
l_{rail}	Length of the rail	m
l_{stick}	Length of the pendulum stick	m
M	Inertia matrix	1
$m_{attachment}$	Mass of motor shaft attachment	kg
m_b	Mass of the bob	kg
m_c	Mass of the cart	kg
P	Error covariance matrix	1
Q	External forces in chapter 3, process noise covariance in chapter 4	N or 1
Q_{lin}	Process noise covariance for linear Kalman filter	1
q	Generalized coordinates	1
$\dot{q}$	Derivative of the generalized coordinates	1
R	Output noise covariance	1
R_{lin}	Output noise covariance for linear Kalman filter	1
r	Radius of pulley	m
r_k	Reward at time k	1
s	Sliding manifold	1
T_s	Sampling time of the system	s
u	Plant input set by controller	A
u_k	Discrete control input at time k	A
$u_{neg,fric}$	Friction compensation for negative direction	A
$u_{pos,fric}$	Friction compensation for positive direction	A
u^*	Compensated control input	A
$V_{attachment}$	Volume of motor shaft attachment	m^3
v	Velocity	$\frac{\text{m}}{\text{s}}$
v_k	Output noise of discrete system at time k	1
v_{ref}	Velocity reference	$\frac{m}{s}$
w	Weight vector	1
w_k	Process noise at time index k	1
$\mathbf{x}$	State vector	1
$\dot{\mathbf{x}}$	Derivative of state vector	1
x	Cart position	m
x_k	Discrete state vector at time k	1
x_p	x-position of the pendulum	m
$\dot{x}$	Cart velocity	$\frac{\text{m}}{\text{s}}$
$\dot{x}_p$	Velocity of the x-position of the pendulum	$\frac{\text{m}}{\text{s}}$
$\hat{x}_k$	State estimation at time k	1

$\bar{x}$	State operating point	1
y	Output vector	1
y_k	Discrete output vector at time k	1
y_p	y-position of the pendulum	m
$\dot{y}_p$	Velocity of the y-position of the pendulum	$\frac{\text{m}}{\text{s}}$
$\hat{y}_k$	Output estimation at time k	1
$\tilde{y}_k$	Output error at time k	1
α	Learning rate or step size	1
β_0	Sliding mode control constant	1
Γ	Discrete time input matrix	1
γ	Discount factor of future rewards	1
ϵ	Saturation variable in chapter 5, random action probability in chapter 6	1
η	Reduced order transformed states	1
$\dot{\eta}$	Derivative of the reduced order transformed states	1
θ	Pendulum angle	rad
θ_q	Quantified pendulum angle	rad
$\dot{\theta}$	Pendulum angular velocity	$\frac{\text{rad}}{\text{s}}$
μ_s	Surface friction coefficient	1
ξ	Transformed input states	1
ρ	Density of medium	$\frac{\text{kg}}{\text{m}^3}$
$\rho_{attachment}$	Density of motor shaft attachment	$\frac{\text{kg}}{\text{m}^3}$
τ_m	Motor torque	Nm
Φ	Discrete time system matrix	1

1 Introduction

This thesis is made out of interest of the group and does not investigate a non-solved real-world problem. The goal of this thesis is for the group to gain experience with Kalman filtration and two control methods found interesting.

This thesis is twofold. Firstly, advanced model-based control is investigated, in terms of non-linear control as well as Kalman filtration, as experience with this is desired in the group. Secondly, machine learning, ML, in a control context is investigated, as more and more examples of ML in the control context is seen. With increasing and cheaper processing power, ML might expand further in the future and become the modern way of doing control [17]. The ability to let the system learn how to control by itself based on data and trials, without the use of classical control involving modelling and determination of a closed control expression, is found fascinating by the group. Thereby the last part of this thesis is to gain experience with machine learning control, MLC.

The interest of these two subjects thereby serves as the foundation for this thesis, which is to obtain knowledge and experience in the field of machine learning control in relation to classic control. It is desired to obtain experience with advanced classic control on a system and insight and experience into ML control, which at the start of the thesis was an unknown field for the group.

The test platform for the thesis is the inverted pendulum on a cart, IPC. This is a classic platform for benchmarks of control systems, as it is a non-linear system, which is naturally unstable and furthermore is under-actuated, meaning only one actuator for two degrees of freedom. All these make it a challenging control task [35]. An IPC test setup is available in the AAU control and automation laboratory and as the system is fairly straightforward to approach and test on, this has been chosen as the test platform for this master thesis.

In chapter 2 the test setup is presented. This is followed by a modelling chapter, 3, where the system is modelled in order to construct a simulation environment and have a model for the model based control. After this, the control structure for the thesis is presented, along with friction compensation and state estimation in chapter 4. Then the two control strategies are designed, firstly the non-linear control in chapter 5 and secondly the ML based control in chapter 6. It has however not been possible to achieve any good results with the chosen ML approach, however it is still presented. The thesis is finished off with conclusion as well as future work.

2 System description

This chapter describes the test setup placed in the AAU control and automation laboratory, which is used in this thesis. First the mechanical setup is presented followed by the electrical setup, such that the foundation for the rest of the thesis is clear.

2.1 Mechanical setup

The IPC test setup consists of a cart, which can move horizontally on rails, as it can be seen in figure 2.1. The pendulum is attached to the cart and is in this thesis freely rotating. However, the pendulum is in reality attached directly to the shaft of a DC motor, acting as the pivot point of the pendulum, in order to make the test setup more generically utilizable in different projects. At the end of the pendulum stick, different weights can be attached, as further described later. The weight at the end of a pendulum is called a bob, which is the reference used for the rest of this thesis.

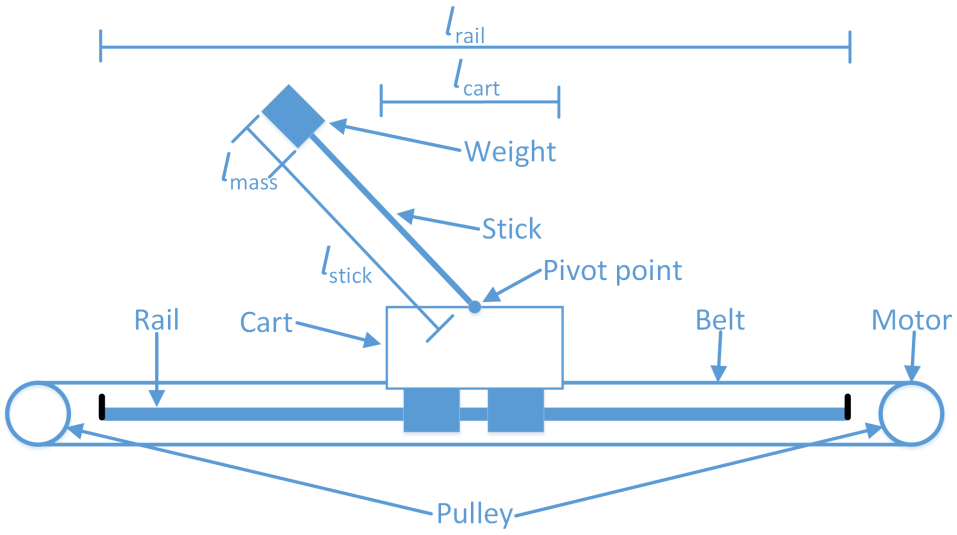


Figure 2.1. The IPC test setup at AAU automation and control laboratory.

The control input is acting on the cart to move it back and forth. This force is delivered from the motor through a pulley to the drive belt, which is attached to the cart. The control task is to stabilize the pendulum in an upright position using the horizontal force from the motor to manipulate the cart position and angle of the pendulum.

In table 2.1 the measured lengths related to figure 2.1 are presented. All measurements are done with a tape measure.

Parameter	Length [m]	Description
l_{rail}	0,89	The length of the rail between end stops.
l_{cart}	0,12	The length of the cart.
l_{stick}	0,282	The length of the stick between pendulum pivot point and bob.
d_{pulley}	0,056	The diameter of the pulley attached to the motor.

Table 2.1. Measured parameters for the IPC test setup.

The length of the rail is as stated 89 cm between the end stops. However, 3,2, cm before each end stop, contacts are placed, such that the power to the motor controllers are disabled before the end stops are reached. Given the cart length, this yields an effective travel length of 70,6 cm for the cart.

The bob is modular, such that different weights can be attached. This is done by screwing weights into each other, such that the length and thereby the mass of the weight is increased or decreased. The different weights and their lengths can be seen in table 2.2.

Weight [g]	Length [m$\cdot 10^{-2}$]
26	1,1
50	2,1
75	3,1
100	4,1

Table 2.2. Measured parameters for weight modules for the bob for the IPC test setup.

It is chosen to have a nominal bob weight of 100g+75g+26g=201g, as this allows for introducing parameter changes in both directions, but still having a relatively high weight to allow for the assumption of a massless stick, as utilized in the modelling as further described in section 3.2. This yields a nominal bob length of $8,3$ cm, and given the assumption that the center of mass of the bob is in the middle of its length, the nominal pendulum length, l, is 0,3235 m.

With the mechanical part of the test setup described, the electrical part will be described in the following section.

2.2 Electrical setup

The combination-motor used in the test setup is a 413267 from Maxon Motor, which consists of a Maxon 370356 DC motor [38] and a HEDS 5540 optical quadrature encoder [49] attached directly to the motor shaft. As seen in figure 2.1, the motor acting on the cart is attached to the right pulley and a motor is placed at the pivot point of the pendulum. The motor attached to the pendulum will be disabled such that it rotates freely in this thesis and its encoder is used to measure the pendulum angle.

The motor is controlled through a Maxon ADS 50/10 motor controller [37], which controls the current through the motor. The motor controller is 500 W with a 50 V, 12 A PSU, which is more

than sufficient for the 200 W Maxon motor. The reference input to the motor controller's current controller is given as a voltage between ±10 V. The full connection diagram can be seen in figure 2.2.

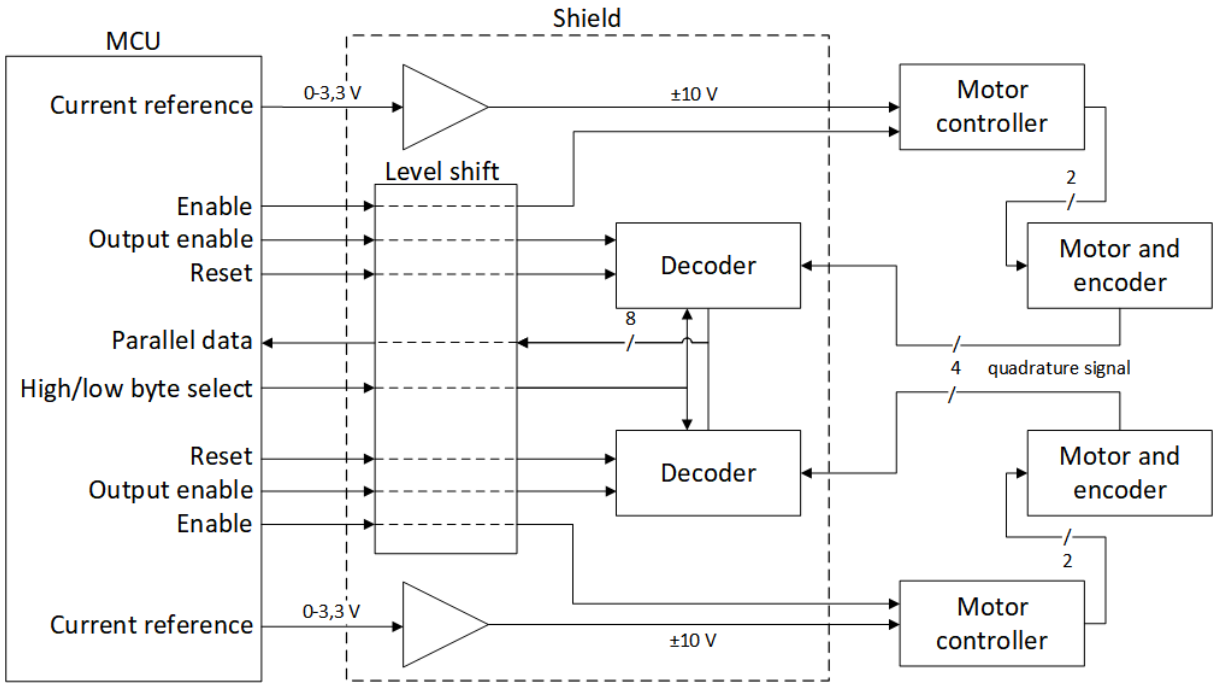


Figure 2.2. The IPC test setup at AAU automation and control laboratory.

The HEDS 5540 optical quadrature encoders are each interfaced through a HCTL-2021-PLC counter/decoder in order to avoid the MCU handling all the ticks from the encoder. This decoder/counter, analyses the signal from the encoder and acts as a counter of the number of ticks from the encoder, since the last reset of the decoder IC [32]. The reset can be performed from the MCU. The 16 bit counter is read through an 8 bit parallel data bus, high and low byte one at the time. The two decoders share the same parallel bus and is thereby time domain multiplexed and controlled from the MCU, by the output enable for each IC.

The encoder/decoder combination on the test setup delivers 2000 counts per revolution of the motor. This corresponds to the following quantization for the pendulum angle, as there are 2π radians on a turn.

$$\Delta\theta = \frac{2\pi}{2000} = \pi \cdot 10^{-3} \frac{\text{rad}}{\text{count}} \tag{2.1}$$

By measuring the total counts over the full length of rail, and a rail length of $l_{rail} - l_{cart} = 0,89\text{ m} - 0,12\text{ m} = 0,77\text{ m}$, the quantization for the cart position is as follows.

$$\Delta x = \frac{0,77}{8753} = 8,80 \cdot 10^{-5} \frac{\text{m}}{\text{count}} \tag{2.2}$$

These resolutions are deemed sufficient for the thesis.

The setup was originally controlled by an Arduino Due microcontroller unit, MCU. However, it has been chosen to upgrade from the Arduino Due to a Teensy 3,6 MCU for this thesis. This is done on behalf of the experience from previous groups, where an extended Kalman filter, EKF, running on the Arduino Due has been utilizing more than half of the available processing power [11]. This is deemed mainly due to the Arduinos lack of a floating-point unit, FPU. As this thesis intends to run ML on the Arduino, as well as an EKF, it has been chosen to upgrade to the Teensy 3,6, which has a FPU as well as in general being a faster and more capable board, as seen in table 2.3.

	Arduino Due	**Teensy 3,6**
Processor	32 bit ARM Cortex M3 @84MHz	32 bit ARM Cortex M4F @180MHz[1]
FPU	No	Yes
Flash Memory	512 KB	1 MB
SRAM	96 KB	256 KB
EEPROM	0	4 KB
I/O pins	54 (two 12 bit DACs)	62 (two 12 bit DACs)

Table 2.3. Comparison of Arduino Due [4] and Teensy 3,6 [47].

The Teensy is Arduino compatible and can be programmed through the Arduino IDE, which makes the system operate as with the original Arduino Due. The Teensy 3,6 is soldered to an Arduino Due prototype shield as a breakout board, such that easy back configuration to an Arduino Due is possible if desired. More documentation for the upgrade can be found in appendix A, as well as the full pin-mapping.

Both the Arduino Due and the Teensy 3,6 are 3,3 V units and as the motor controller needs ±10 V to set the current reference, an amplification is made on the shield in the test setup to amplify the 12 bit 0-3,3 V value from the MCU to the desired ±10 V, as seen in figure 2.2. A test has been conducted, to determine how the ADC value from the MCU corresponds to the current in the motor, which can be seen in figure 2.3. The test has been conducted with the cart fixed and the current is measured with a current probe and an oscilloscope.

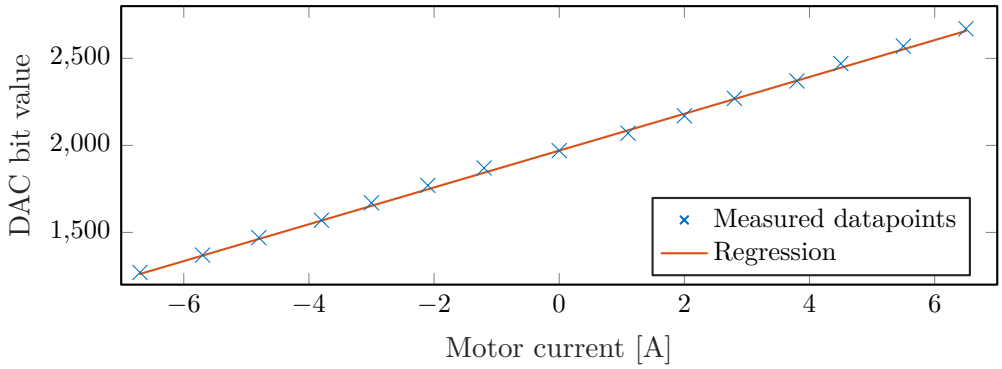


Figure 2.3. Linear regression of amperes to DAC bit value.

[1]The Teensy 3,6 can be overclocked to 240 MHz directly from the Arduino IDE if needed.

A linear regression is made, such that a conversion function from current reference, i_{ref}, to DAC value which corresponds to this current, can be implemented in the MCU, such that the interface is the wanted current directly. The regression is seen in the following equation.

$$bit_{DAC} = 105,78 I_{ref} + 1970 \tag{2.3}$$

This means, that the quantization of the current between DAC values is $\frac{1}{105,78} = 9,5\text{mA}$, which is more than sufficient.

The sampling frequency for the thesis has been chosen based on the fact, that most IPC projects and papers online uses between 100 and 200 Hz for such a setup, with good results. Furthermore, the previous groups at AAU working with this test setup have been using 200 Hz. It is chosen to use 150 Hz in this thesis, as it is in the middle of the range others use and is deemed a good compromise between MCU calculation capabilities and the problem with encoders when sampling fast, as further described in section 4.3. Given a cart velocity of 1 m/s this sampling frequency corresponds to a movement of $6,67 \cdot 10^{-3}$ m between samples and with an angular velocity of 2π/s it yields $4,2 \cdot 10^{-2}$ radians between samples. These are both deemed acceptable, as the velocities used for the calculation are high and still yields a sufficient precision.

With the test setup described, the modelling of the IPC is described in the following chapter.

3 Modelling of inverted pendulum on a cart

This chapter will describe the modelling of the inverted pendulum on a cart and present the parameter estimation as well as model validation for the described test setup in chapter 2, placed in the Control and Automation laboratory at AAU.

3.1 Newtonian vs Lagrangian mechanics

When modelling a mechanical system and thereby also the pendulum on a cart, there are two overall and common approaches which are used. One modelling method is by using the forces in the system e.g. Newtonian mechanics and the other are energy-based methods e.g. Lagrangian mechanics. Both approaches can be used and will result in the same equations of motion.

When using classic mechanics such as Newtonian mechanics, the dynamic equations for a mechanical system are derived from the forces acting upon the mechanical parts, which are individually represented as free body diagrams. The relationship between mass, force, position, velocity, and acceleration is defined from Newtown's three laws and is utilized to setup dynamic equations for the individual parts. This is normally done in the direction of the rectangular coordinate system, in where the system is viewed, as this is the most straightforward way of utilizing Newton's laws. The free body diagrams will contain the constrained forces for the system, as in this case the tension through the pendulum stick. By manipulation and substitution, the constrained forces can be removed from the equations, and the equations of motion are obtained. This is often referred to as the direct method, as the forces acting on the system is used directly [22].

However, when these constrained forces are not of interest or when dealing with nonrectangular coordinate systems, it is cleaner and less tedious to utilize an energy-based method for modelling the system, such as Lagrange mechanics. This method is called an indirect method, as it does not concern forces in the system, and thereby the constrained forces are not to be dealt with. Instead, it utilizes energy considerations and is described in terms of generalized coordinates, which can be chosen freely, as long as they are independent. The number of generalized coordinates needed, is equal to the degrees of freedoms in the system, which in this case is two [22] [52].

Because of this, the IPC in this thesis is modelled by use of Lagrangian mechanics, which is further described and used in the following section.

3.2 Modelling of the inverted pendulum on a cart

The equation used in the derivation of the equations of motion by use of Lagrange, comes from Hamilton's principle of stationary action. This states, that the trajectory of a mechanical system from its state at time a to its state at time b is given by the stationary point of the action functional, I, seen in equation 3.1 [52].

$$I(t,q) = \int_a^b L(t,q,\dot{q})dt \tag{3.1}$$

The function L is the Lagrangian function, which is defined as the difference between the kinetic energy, T, and potential energy, U, as a function of the generalized coordinates $q = [q_1 \ \cdots \ q_N]^T$ and its time derivatives $\dot{q} = [\dot{q}_1 \ \cdots \ \dot{q}_N]^T$,

$$L(t,q,\dot{q}) = T(t,q,\dot{q}) - U(t,q). \tag{3.2}$$

It is thereby known, that given the potential and kinetic energy of the system can be described as a function of the generalized coordinates, then the system trajectory is given as the stationary point of the action functional. A functional is a function which takes a function as input, in this case the trajectory of the system, and returns a real number. In terms of the action function, it is the integral of an energy potential over the path given as input. The trajectory for the mechanical system is by the Hamiltonian principle of stationary action given as the trajectory which minimizes this energy potential in the system.

A necessary condition for a stationary point of a functional is given from calculus of variations, and is given by the Euler-Lagrange equation,

$$0 = \frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q}. \tag{3.3}$$

Thereby, by expressing the kinetic and potential energy in terms of the generalized coordinates and defining the Lagrangian, the Euler-Lagrange function can be utilized to find the system trajectory and thereby the equations of motion. This is done in the following for the IPC, first for the system without external forces and frictions, as this is not included in the above. Afterwards, the equations of motion are expanded to include the external control force from the motor, as well as frictions.

The IPC can be seen in figure 3.1, where necessary variables for the modelling are included.

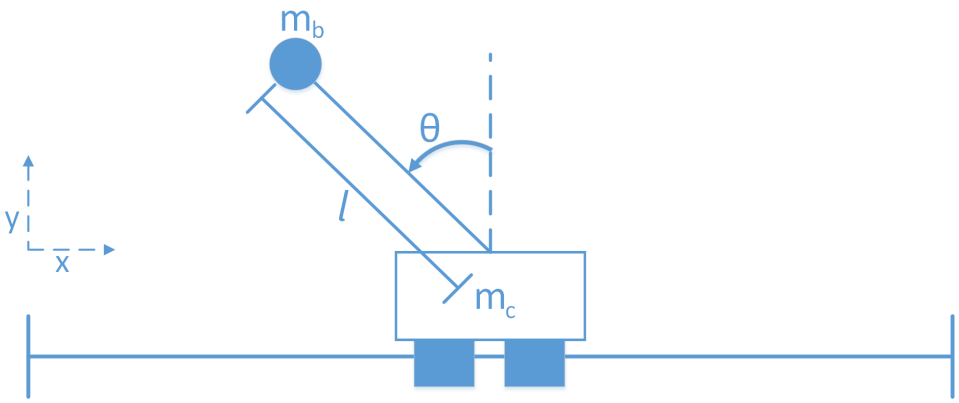


Figure 3.1. Pendulum on a cart.

The mass of the cart is defined as m_c, the bob is placed at the end of the pendulum stick and its weight is denoted m_b. The length, l, is the length from the rotational joint of the stick on the cart, to the center of mass of the bob, which is assumed in the middle of the bob, on behalf of its physical shape. The inertia of the pendulum comes from the bob, the stick, the attachment of the stick to the motor and the motor itself. In equations 3.4 and 3.5, rough estimates for the inertias are shown in order to determine which are necessary to include in the modelling. The motor inertia is $5{,}42{\cdot}10^{-5}$ kgm^2 [38]. The attachment of the stick to the motor shaft is made from a machined solid aluminum cylinder and its volume is hereunder calculated as un-machined based in its outer dimensions, introducing some conservatism. The stick itself is made from a thin aluminum tube and is thereby not considered, due to its very low weight.

$$I_{attachment} = \frac{1}{2} m_{attachment}\ r^2 = \frac{1}{2} \rho_{aluminum} V_{attachment}\ r^2 = 4,5 \cdot 10^{-18}\ \text{kgm}^2 \tag{3.4}$$

$$I_{bob} = m_b l^2 = 0,21 \cdot 10^{-2}\ \text{kgm}^2 \tag{3.5}$$

It can be seen, that the only significant contribution to the inertia comes from the bob, and thereby the rest is deemed negligible and the stick is deemed massless.

To obtain the Lagrangian, the expressions for the kinetic and potential energy is derived in the following. First the kinetic energy is expressed, which for the cart, is given as in equation 3.6. The generalized coordinates are defined as $q = [x\ \theta]^T$, as these are independent and minimal for describing the system fully.

$$T_c = \frac{1}{2} m_c \dot{x}^2. \tag{3.6}$$

In order to describe the pendulum, the positions of the cart and pendulum are defined. The cart's position is given by the x-coordinate, and the coordinate system is placed such as its non-changing y-position are at $y = 0$, as seen in figure 3.1. The pendulums position is given in x- and y-coordinates as

$$\begin{bmatrix} x_p \\ y_p \end{bmatrix} = \begin{bmatrix} x - l\sin(\theta) + \frac{l_{cart}}{2} \\ l\cos(\theta) \end{bmatrix}. \tag{3.7}$$

To express the kinetic energy for the pendulum, T_p, the pendulum velocity is found given by contributions in both the x- and y-directions, as the pendulum swings. The velocity components are found by differentiating equation 3.7.

$$\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \end{bmatrix} = \begin{bmatrix} \dot{x} - l\cos(\theta)\dot{\theta} \\ -l\sin(\theta)\dot{\theta} \end{bmatrix} \tag{3.8}$$

The kinetic energy for the pendulum is thereby defined as,

$$\begin{aligned} T_p &= \frac{1}{2}m_b(\dot{x}_p^2 + \dot{y}_p^2) = \frac{1}{2}m_b\left(\left(\dot{x} - l\cos(\theta)\dot{\theta}\right)^2 + \left(-l\sin(\theta)\dot{\theta}\right)^2\right) \\ &= \frac{1}{2}m_b\dot{x}^2 - m_b l\cos(\theta)\dot{\theta}\dot{x} + \frac{1}{2}m_b l^2\cos^2(\theta)\dot{\theta}^2 + \frac{1}{2}m_b l^2\sin^2(\theta)\dot{\theta}^2 \\ &= \frac{1}{2}m_b\dot{x}^2 - m_b l\cos(\theta)\dot{\theta}\dot{x} + \frac{1}{2}m_b l^2\dot{\theta}^2. \end{aligned} \tag{3.9}$$

The last step follows from the fact, that $\cos^2(\theta) + \sin^2(\theta) = 1$. The total kinetic energy is under the massless stick assumption as previously described, given by equation 3.6 and 3.9 yielding

$$T = T_c + T_p. \tag{3.10}$$

The potential energy for the system is given by the gravitational constant, g. The height of the cart will never change as it only moves vertically and thereby no change in potential energy will occur. Because of this, the potential energy of the cart is set to 0, as a reference potential. The reference potential can be set arbitrarily, as it disappears when applying the Euler-Lagrange equation. From this, the potential energy for the system can be expressed as equation 3.11 to 3.13.

$$U_c = 0 \tag{3.11}$$

$$U_p = glm_b\cos(\theta) \tag{3.12}$$

$$U = U_c + U_p \tag{3.13}$$

The Lagrangian is thereby found as,

$$L = T - U = \frac{1}{2}(m_c + m_b)\dot{x}^2 + \frac{1}{2}m_b l^2\dot{\theta}^2 - m_b l\cos(\theta)\dot{\theta}\dot{x} - glm_b\cos(\theta). \tag{3.14}$$

The Euler-Lagrange equation 3.3 can now be utilized to find the equations of motion from equation 3.14. The partial derivatives are taken in the equation 3.15.

$$\begin{aligned} \frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} &= \frac{d}{dt}\begin{bmatrix}(m_c + m_b)\dot{x} - m_b l\cos(\theta)\dot{\theta} \\ m_b l^2\dot{\theta} - m_b l\cos(\theta)\dot{x}\end{bmatrix} - \begin{bmatrix}0 \\ m_b l\sin(\theta)\dot{\theta}\dot{x} + glm_b\sin(\theta)\end{bmatrix} \\ &= \begin{bmatrix}(m_c + m_b)\ddot{x} - m_b l\cos(\theta)\ddot{\theta} + m_b l\sin(\theta)\dot{\theta}^2 \\ m_b l^2\ddot{\theta} - m_b l\cos(\theta)\ddot{x} + m_b l\sin(\theta)\dot{\theta}\dot{x}\end{bmatrix} - \begin{bmatrix}0 \\ m_b l\sin(\theta)\dot{\theta}\dot{x} + glm_b\sin(\theta)\end{bmatrix} \\ &= \begin{bmatrix}(m_c + m_b)\ddot{x} - m_b l\cos(\theta)\ddot{\theta} + m_b l\sin(\theta)\dot{\theta}^2 \\ m_b l^2\ddot{\theta} - m_b l\cos(\theta)\ddot{x} - glm_b\sin(\theta)\end{bmatrix} = 0 \end{aligned} \tag{3.15}$$

The dynamic equation, 3.15, is thereby describing the equations of motion for the system, given no external forces and frictions in the system. The equations have been verified in simulation

in MATLAB[1], where it is concluded, that both the pendulum and cart position is a harmonic oscillator, when the system is simulated from an initial position in the pendulum angle, as seen in figure 3.2.

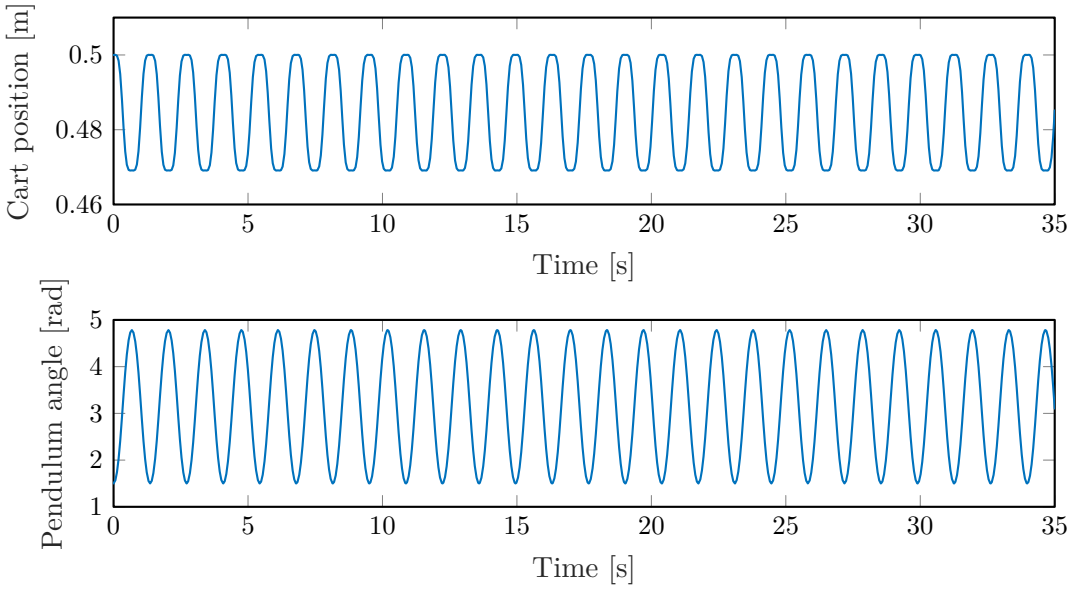


Figure 3.2. Simulation of equations of motion showing a harmonic oscillator.

The nominal bob weight and pendulum length are used, and the weight of the cart is set to 4 kg. The simulation is as expected, and the equations of motion are deemed correct, from where friction and external forces are introduced into the system in the following section.

3.3 Modeling of external forces

This section describes how the Euler-Lagrange equation can be expanded such that external forces acting on the mechanical system is included. This is used, such that the external control force and the friction force affecting the IPC, can be included in the equations of motion, equation 3.15, found in the previous section.

If an external force is acting on the system, the force needs to be described in the directions of the generalized coordinates, such a description of the forces is called generalized forces, Q, and can be incorporated into the Euler-Lagrange equation as shown in equation 3.16. However, this can be expanded to include forces, which is dependent on the velocity of the generalized coordinates as well, by use of the Lagrange-d'Alembert Principle, such that Q can be dependent on both the generalized coordinates and their velocities [52].

[1]It has by the group been observed, that in order to obtain this result, it is necessary to increase the precision of the ODE solver from the default 10^{-3} relative error tolerance and default 10^{-6} absolute error tolerance, in order to obtain a simulation showing a harmonic oscillator. For the rest of the modelling chapter, simulations are made with a relative error tolerance of 10^{-6} and an absolute error tolerance of 10^{-8}.

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = Q \tag{3.16}$$

The external forces are for the IPC system straightforward to describe as generalized forces and can thereby be incorporated into the model in the above mentioned way. The forces present for the IPC in the test setup are the applied control force, F_a, from the motor acting on the cart, and the friction forces in the cart and the pendulum.

When an object is moving through a medium, in this case air, an aerodynamic drag force will be present. The aerodynamic drag force is, as the friction force, acting in the opposite direction of the movement and is often given as equation 3.17 [10].

$$F_d = \frac{1}{2} C_D \rho A v^2 \tag{3.17}$$

Where C_D is the drag coefficient of the object, depending on e.g. the shape, ρ is the density of the medium the object is moving in, A is the frontal surface area of the object and v is the velocity.

It can be seen, that the drag force is proportional to the squared velocity, and in [36] the torque from drag is proven to be proportional to the squared angular velocity for the pendulum as well. Given the assumption that C_D and A does not change over time, the drag is acting on both the pendulum and the cart, in the same manner as the viscous friction, as described in section 3.5. Because of this, if any considerable drag is present in the system, it can be seen as a part of the viscous friction constant when this is estimated and is thereby not dealt with separately. The aerodynamical coefficients for the pendulum are not changing over time, and this is also deemed the case for the cart, as the drag from the pendulum position is minimal compared to the drag of the cart itself. However, as the focus of this thesis is stabilization of the pendulum, the speed is relatively slow of both the pendulum and cart and the drag can thereby be considered small.

The applied control force from the motor acts on the cart through the belt and pulley, as further described in section 3.4, and is denoted F_a. The control force acts directly on the cart in the direction of the generalized x-coordinate, such that the applied force, F, on the system is defined as,

$$F = \begin{bmatrix} F_a \\ 0 \end{bmatrix}. \tag{3.18}$$

The friction force, F_B, acts on both the pendulum and the cart in the direction of generalized coordinates and is given from the velocity of the generalized coordinates, namely the velocity of the cart and the angular velocity of the pendulum, as described further in section 3.5. These can thereby also be put into the system as external forces, where F_{B_c} is the friction force acting on the cart and F_{B_p} is the friction force acting on the pendulum,

$$F_B(\dot{x}, \dot{\theta}) = \begin{bmatrix} F_{B_c}(\dot{x}) \\ F_{B_p}(\dot{\theta}) \end{bmatrix}. \tag{3.19}$$

The applied control force, F, is defined in the positive x-direction and the frictions force, F_B will

act in the opposite direction of the velocity, such that the combined external forces are

$$Q = F - F_B. \tag{3.20}$$

This yields the following combined equation of motion using equation 3.16 to expand the equations of motion, in equation 3.15, into

$$\begin{bmatrix} (m_c + m_b)\ddot{x} - m_b l\cos(\theta)\ddot{\theta} + m_b l\sin(\theta)\dot{\theta}^2 \\ m_b l^2\ddot{\theta} - m_b l\cos(\theta)\ddot{x} - g l m_b \sin(\theta) \end{bmatrix} = F - F_B. \tag{3.21}$$

The equation is put on matrix form:

$$\underbrace{\begin{bmatrix} (m_c + m_b) & -m_b l\cos(\theta) \\ -m_b l\cos(\theta) & m_b l^2 \end{bmatrix}}_{M} \begin{bmatrix} \ddot{x} \\ \ddot{\theta} \end{bmatrix} + \begin{bmatrix} m_b l\sin(\theta)\dot{\theta}^2 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ -g l m_b \sin(\theta) \end{bmatrix} = F - F_B \tag{3.22}$$

It can be noted, that this formula fits with the generic formula for a M-link robot [26] (page 24), which is given as

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = u. \tag{3.23}$$

Here $M(q)$ is an inertia matrix, the term $C(q,\dot{q})\dot{q}$ describes the centrifugal and Coriolis forces, where terms proportional to the rotational velocity is Coriolis forces and the terms proportional to squared rotational forces are centrifugal forces. From here it can be seen, that no Coriolis forces are present in the IPC, which makes sense, as no rotating coordinate systems are used. The vector $g(q)$ describes the gravitational force and u is the vector of actuation forces/toques minus the frictions in the system. Based on this, equation 3.22 is deemed looking correct.

Equation 3.22 is solved for $\ddot{q}$ by taking the inverse of M, which always is possible since its determinant always will be different from zero and thereby always will be nonsingular, as $\sin^2(\theta) \geq 0$.

$$\det(M) = (m_c + m_b)m_b l^2 - m_b^2 l^2\cos^2(\theta) = m_b l^2\left(m_c + m_b\sin^2(\theta)\right) \geq m_c m_b l^2 > 0 \tag{3.24}$$

Thereby the second order derivatives can be isolated, by use of the expression for the determinant in equation 3.24, yielding the following:

$$
\begin{aligned}
\begin{bmatrix}\ddot{x}\\ \ddot{\theta}\end{bmatrix} =& M^{-1}\left(F - F_B - \begin{bmatrix} m_b l \sin(\theta)\dot{\theta}^2 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ -g l m_b \sin(\theta) \end{bmatrix}\right) \\
=& \frac{1}{\det M}\begin{bmatrix} m_b l^2 & m_b l\cos(\theta) \\ m_b l \cos(\theta) & (m_c + m_b) \end{bmatrix}\left(\begin{bmatrix} F_a \\ 0 \end{bmatrix} - F_B - \begin{bmatrix} m_b l \sin(\theta)\dot{\theta}^2 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ -g l m_b \sin(\theta) \end{bmatrix}\right) \\
=& \begin{bmatrix} \left(m_c + m_b \sin^2(\theta)\right)^{-1} & \cos(\theta)\left(l\left(m_c + m_b\sin^2(\theta)\right)\right)^{-1} \\ \cos(\theta)\left(l\left(m_c + m_b\sin^2(\theta)\right)\right)^{-1} & (m_c+m_b)\left(m_b l^2\left(m_c + m_b\sin^2(\theta)\right)\right)^{-1} \end{bmatrix} \\
& \left(\begin{bmatrix} F_a \\ 0 \end{bmatrix} - F_B - \begin{bmatrix} m_b l \sin(\theta)\dot{\theta}^2 \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ -g l m_b \sin(\theta) \end{bmatrix}\right) \\
=& \underbrace{-\begin{bmatrix} \left(m_c + m_b \sin^2(\theta)\right)^{-1} & \cos(\theta)\left(l\left(m_c + m_b\sin^2(\theta)\right)\right)^{-1} \\ \cos(\theta)\left(l\left(m_c + m_b\sin^2(\theta)\right)\right)^{-1} & (m_c+m_b)\left(m_b l^2\left(m_c + m_b\sin^2(\theta)\right)\right)^{-1} \end{bmatrix}}_{f_1(\theta)} F_B \\
& \underbrace{-\begin{bmatrix} \frac{m_b l \sin(\theta)\dot{\theta}^2 - g m_b \sin(\theta)\cos(\theta)}{m_c + m_b \sin^2(\theta)} \\ \frac{m_b l \sin(\theta)\cos(\theta)\dot{\theta}^2 - g\sin(\theta)(m_c+m_b)}{l\left(m_c + m_b\sin^2(\theta)\right)} \end{bmatrix}}_{f_2(\theta,\dot{\theta})} + \underbrace{\begin{bmatrix} \left(m_c + m_b \sin^2(\theta)\right)^{-1} \\ \cos(\theta)\left(l\left(m_c + m_b\sin^2(\theta)\right)\right)^{-1} \end{bmatrix}}_{g_1(\theta)} F_a
\end{aligned}
\tag{3.25}
$$

The achieved dynamical equation is however a second order differential equation, which is not desired. A new state vector is therefore defined in order to obtain a system of first order derivatives, such that it can be presented as a non-linear state space system,

$$
\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x \\ \theta \\ \dot{x} \\ \dot{\theta} \end{bmatrix}. \tag{3.26}
$$

This yields the first order differential equations for the system dynamics

$$
\dot{\mathbf{x}} = \underbrace{\begin{bmatrix} x_3 \\ x_4 \\ f_1(\mathbf{x})F_B + f_2(\mathbf{x}) \end{bmatrix}}_{f(\mathbf{x})} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ g_1(\mathbf{x}) \end{bmatrix}}_{g(\mathbf{x})} F_a. \tag{3.27}
$$

The non-linear state space system is thereby represented on standard non-linear form with the state development described by $f(\mathbf{x})$, and $g(\mathbf{x})$ describes the input dynamics, where F_a is the control input. It can furthermore be seen, that $g(x)$ will always be non-zero, ensuring that the system always is affectable by the control force.

Thereby the external forces from the motor and frictions have been included in the model. A simulation where an external force, F_a, is applied to the system is seen in figure 3.3.

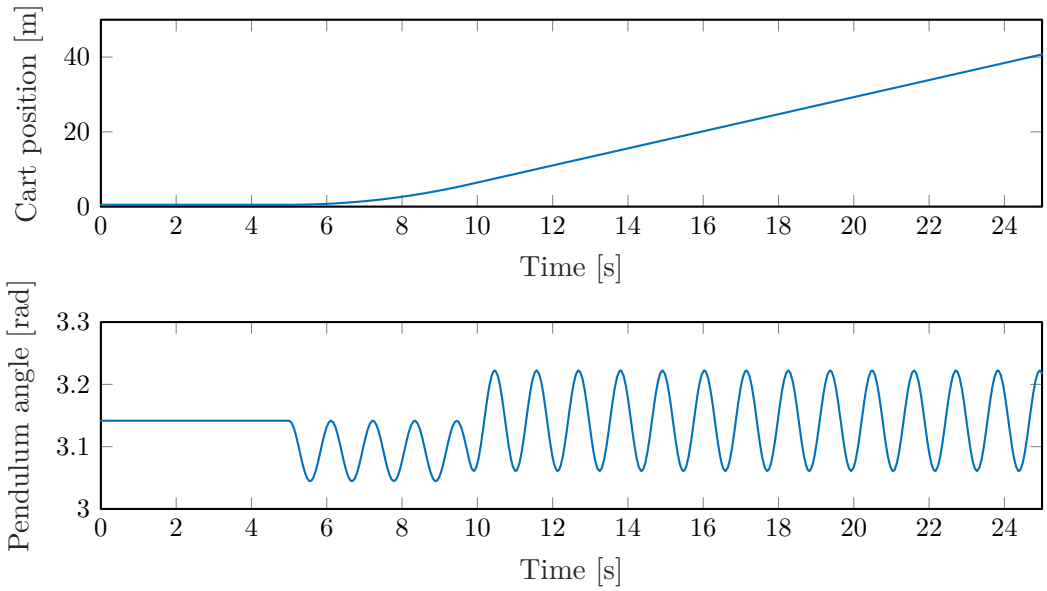


Figure 3.3. Simulation of equations of motion with an external force of 2 N is applied from time 5 to 9.8 seconds, and $m_c = 4$ kg and the rest of the parameters are the nominal values from chapter 2.1.

It can be seen, that the cart accelerates with constant acceleration in the time period where the force is applied and from there continues with an almost constant velocity, only affected by the pendulum swing. The pendulum angle swings undamped around an angle related to the applied control force, and after 9,8 seconds, its swings undamped around π, which is straight down. From this, it has been concluded that the system reacts correctly to the external motor force.

The damping force, F_B, is verified by checking whether it enters the system correctly, through simulation. The frictions present in the modelled system is described in section 3.5, however for this simulation one of the most common and simple frictions is used. This is the viscous friction, which is defined as

$$F_B = \begin{bmatrix} B_{v,c}\dot{x} \\ B_{v,p}\dot{\theta} \end{bmatrix}. \tag{3.28}$$

The simulation is done by having an initial angle of the pendulum at $1,5$ rad, and letting it swing. This can be seen in figure 3.4.

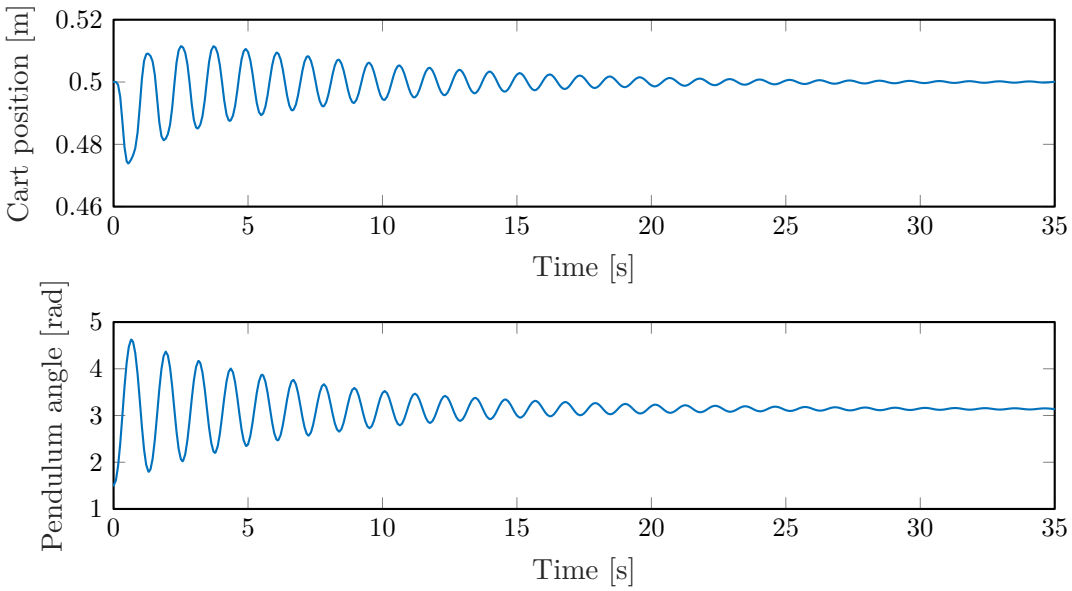


Figure 3.4. Simulation of equations of motion with an initial pendulum angle and viscous friction. The friction coefficients are $F_{v,c} = 3$ N/(m/s) and $F_{v,p} = 5$ mNm/(rad/s), and $m_c = 4$ kg and the rest of the parameters are the nominal values from chapter 2.1.

It can be seen that the pendulum angle, is damped with an exponential rate, as it should, for a damped harmonic oscillator. The cart also reacts appropriately, as it is damped similarly.

Based on this, the model with external actuator force and friction is deemed correct, and the relation between the force, F_a, and the controlled motor current is described in the next section.

3.4 Modelling of motor, pulley and belt

In this section, the motor, pulley, and belt are modelled in order to find a relation between the input set by the MCU and the force applied to the cart, as described and modelled in section 3.3.

As mentioned in section 2.2, the DC motor is current controlled from the MCU. This means that the electrical circuit-equivalent in the motor can be discarded, as the voltage is controlled by the internal current controller, in order to obtain the specified current. Because of this, the current to torque is simply a direct relation, given by the motor constant,

$$\tau_m = K_t i_a. \tag{3.29}$$

Where:

τ_m	is the motor torque	[Nm]
K_t	is the motor constant	[Nm/A]
i_a	is the applied current	[A]

Thereby the torque is directly determined by the current applied to the motor and it is thereby necessary to determine the speed and steady state error of the current controller in the motor

driver. In order to do this, step responses are measured, which can be seen in figure 3.5. It can be seen, that the motor controller is sufficiently fast, as it settles within 1 ms, which is sufficiently faster than the sampling frequency of the system of 150 Hz. The test has been conducted with the cart fixed in a standstill and the current is measured with a current probe and an oscilloscope. The small steady state error might be due to the magnetic field current probe being sensitive to position and angle to the wire, as well as an offset is present before the step, where the motor controller is disabled. The steady state error is thereby not deemed a problem. It is assumed that the 50 V available to the 500W motor controller is enough, in order for the motor driver to keep the current even at the highest velocity obtained.

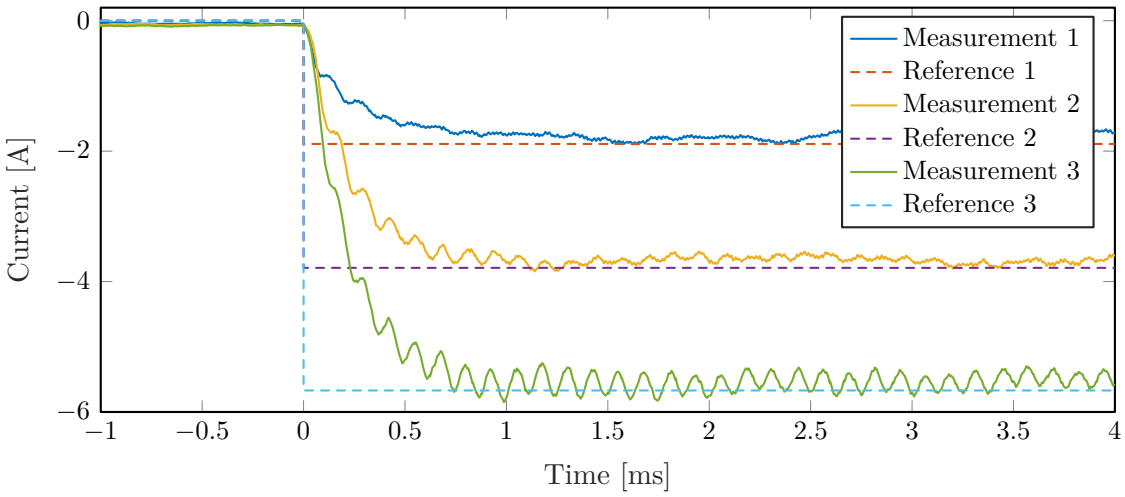


Figure 3.5. Step responses of the current to the DC motor. The data is filtered through a first order low-pass filter with a time constant of 0,2 ms.

On the figure oscillations at 7.5KHz is present, this can be due to aliasing, even though the sampling frequency is 500KHz, which is 10 times the PWM frequency of the motor controller. This has however not been investigated further as it is not deemed a problem in that frequency.

Thereby the relation between the motor current and the torque can be extended to describe the applied force on the cart from the requested current from the MCU. The cart and the motor are connected directly through the pulley and the belt, as shown in figure 2.1. Mechanically, both the motor, pulley, and belt will contribute with an inertia and a friction to the system. However, this inertia and friction can be combined with the inertia and friction of the cart itself, such that the estimated parameters for weight and friction for the cart, includes the frictions and inertias of the motor, pulley, and belt [24] (section 5,2). This is chosen, as this limits the number of parameters to be estimated and represents the same dynamics. Thereby, the description from the motor torque to the applied force is simply given as the following, where r is the radius of the pulley:

$$F_a = \frac{1}{r}\tau_m = \frac{K_t}{r} i_a \tag{3.30}$$

The belt is assumed non-elastic and no-slipping since this otherwise would lead to more compli-

cated modelling. This is however deemed a fair assumption, as it is a toothed drive belt, specially made for purposes as this.

Thereby the relation from the current, which is set by the Arduino, to the force applied to the cart is found. In the following section, the frictions present in the setup will be discussed and modeled.

3.5 Frictions in the setup

In a mechanical system, frictions will be present, which will be discussed in this section. The friction force, F_f, is a force acting in the opposite way of the movement or direction of applied force if the object is at a standstill. Friction is a complicated process and will in this thesis only be dealt with by the most normal and simple approximations, to limit the complexity, as this is often sufficient for control purposes of a system like the IPC [42].

Simplified friction can be described by three terms, the viscous, static and Coulomb friction. The viscous friction describes the friction force, F_v, as a linear relation to the velocity, v, between the sliding surfaces [42],

$$F_v(t) = B_v v(t). \tag{3.31}$$

The friction coefficient, B_v, is defined from the material and surface roughness, for the sliding objects and can be seen illustrated in figure 3.6a.

The static friction force, F_s, acting on the system, describes how the surfaces stick together in standstill, such that a force, $\mathcal{F}_s$, is needed, in order for the objects to start sliding, this can be seen on figure 3.6b. In reality, the static friction impacts moving objects, whose velocity are close to zero and not only in standstill, this is however not considered in this thesis, as it is normally not considered in simpler friction modelling. The description used in this thesis is shown in the following equation [42].

$$F_s(t) = \begin{cases} 0 & \text{if } v(t) \neq 0 \\ F_{net}(t) & \text{if } v(t) = 0 \ \wedge \ |F_{net}(t)| \leq \mathcal{F}_s \\ \mathcal{F}_s \ \text{sign}\left(F_{net}(t)\right) & \text{if } v(t) = 0 \ \wedge \ |F_{net}(t)| > \mathcal{F}_s \end{cases} \tag{3.32}$$

The force term F_{net} is the total net force in the system, meaning in the case of the cart, it is both the applied control force and the force from the pendulum acting on the cart.

It has in the parameter estimation, section 3.6, been concluded, that the static friction for the IPC test setup, is negligible. Due to this, it is not considered further in this thesis and is not a part of the final friction model.

The Coulomb friction acting on the system, F_c, is defined by the force pressing the two sliding surfaces together as well as the surface material and roughness. The Coulomb friction to overcome in order for the system to move, $\mathcal{F}_c$, is given from a friction coefficient, μ_s and the normal force between the two surfaces, N, as seen in equation 3.33. In principle, the normal force for the cart

will be time-varying, due to the rotation of the pendulum. This is however deemed negligible, as the pendulum has relative small angle deviation given it in this thesis is stabilized.

$$\mathcal{F}_c = \mu_s N \tag{3.33}$$

The expression for the Coulomb friction acting on the system, F_c, is described as [42]

$$F_c(t) = \begin{cases} \mathcal{F}_c \operatorname{sign}(v(t)) & \text{if } v(t) \neq 0 \\ F_{net}(t) & \text{if } v(t) = 0 \ \wedge \ |F_{net}(t)| \leq \mathcal{F}_c \\ \mathcal{F}_c \operatorname{sign}(F_{net}(t)) & \text{if } v(t) = 0 \ \wedge \ |F_{net}(t)| > \mathcal{F}_c \end{cases}. \tag{3.34}$$

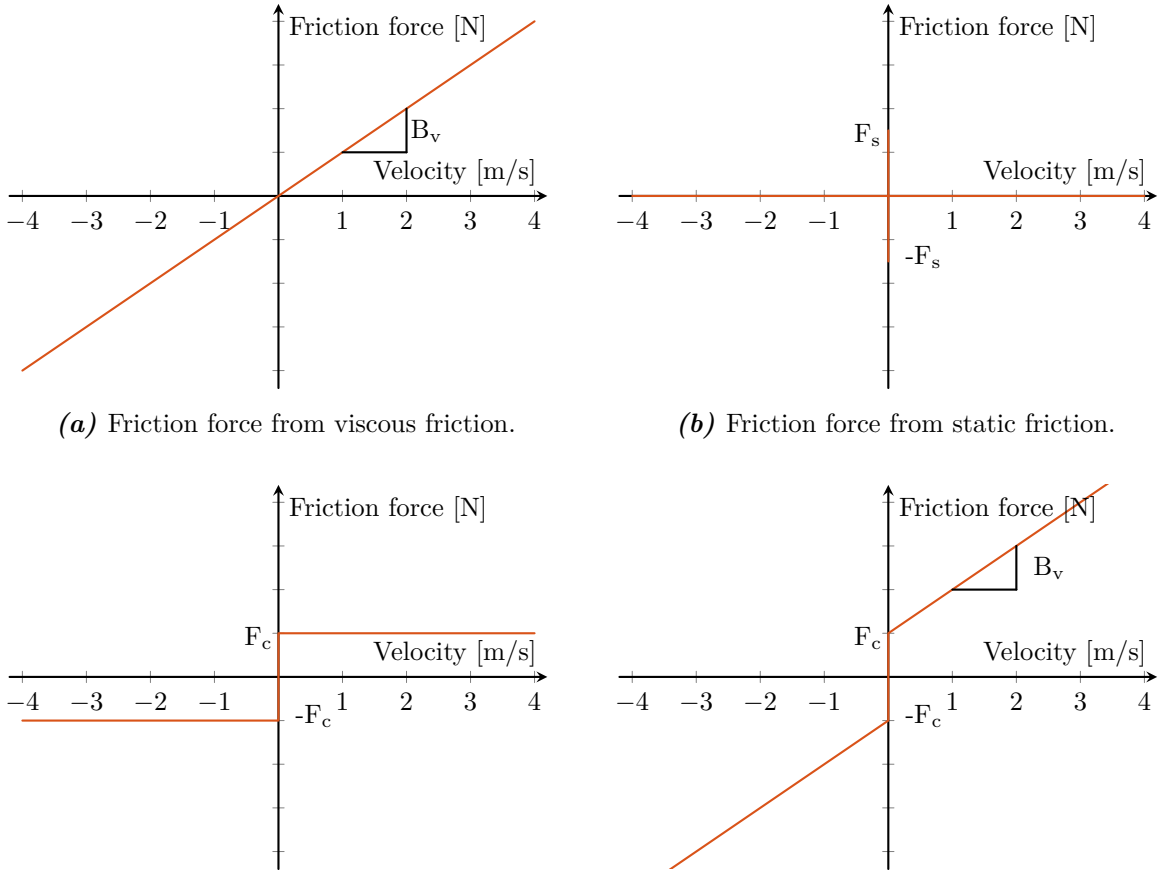
A simplified version of the Coulomb friction is often used, where the ramping behavior in relation to the applied force in standstill is not described. This approximation is given in terms of the velocity instead of the force and is thereby easier to incorporate in the modelling. The simplification is as follows and shown in figure 3.6c [2],

$$F_c(t) = \mathcal{F}_c \operatorname{sign}(v(t)), \tag{3.35}$$

where sign$(\cdot)$ is the signum function defined as

$$\operatorname{sign}(v(t)) = \begin{cases} 1 & \text{if } v(t) > 0 \\ 0 & \text{if } v(t) = 0 \\ -1 & \text{if } v(t) < 0 \end{cases}. \tag{3.36}$$

The total friction acting on the system, is the sum of the above described friction forces, excluding the static friction as mentioned, and can be seen in figure 3.6d.



(a) Friction force from viscous friction.

(b) Friction force from static friction.

(c) Friction force from Coulomb friction.

(d) The total friction force.

Figure 3.6. Illustrations of the friction forces acting on the mechanical system.

The approximation of the Coulomb friction in equation 3.35 however brings some problems, when the system has zero velocity, as no Coulomb friction will then be present, and the system will start moving regardless whether the applied force exceeds the Coulomb friction or not. This discontinuity in zero brings problems, which can be avoided by an continuous approximation [2], where the sign function is replaced by a tanh function,

$$F_c(t) = \mathcal{F}_c \tanh\left(k_{tanh} v(t)\right). \tag{3.37}$$

By varying the constant k_{tanh}, it can be determined how fast the tanh function changes from approximately -1 to 1, as it can be seen in figure 3.7.

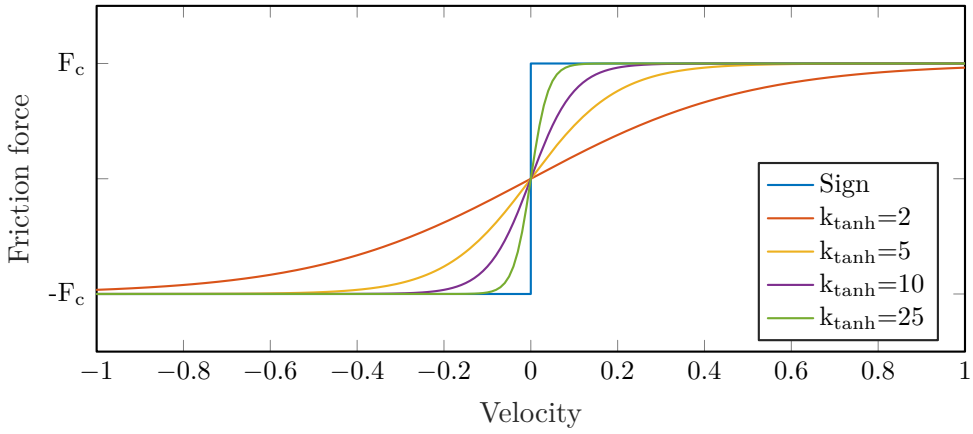


Figure 3.7. Comparison of Coulomb friction approximated with sign, equation 3.35, and tanh, equation 3.37.

It can be seen, that the bigger the constant, the faster the change is and the closer the tanh approximation is to the sign function, however, this also means a larger derivative close to zero, and thereby numerical problems. Because of this, the constant k_{tanh} is determined to be 250, as this means a velocity of 1 cm/s will yield approximately 0.99 of the tanh function, which is deemed sufficient and have not shown numerical problems or noticeable extra simulation time.

However, the problem with no friction present at zero velocity is still present. Furthermore, the model will start to drift with a small velocity, given an actuation force on the system, even though the actuation force is smaller than the Coulomb friction, due to the tanh function not reaching ± 1 before a certain velocity [2]. This can cause problems for systems with smaller control forces and operation close to 0 velocity if the position is of importance. However, given a large value of k_{tanh}, it is deemed not a problem for this thesis, and this approximation for the Coulomb friction is used.

This means, that the final description for the friction in the system, given by the Coulomb and viscous friction, is

$$F_f = B_v v(t) + \mathcal{F}_c \tanh\left(k_{tanh} v(t)\right). \tag{3.38}$$

The final model for the IPC is thereby as summarized in equation 3.39 to 3.43, from equation 3.25 to 3.27, equation 3.30 and equation 3.38.

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{\theta} \\ \ddot{x} \\ \ddot{\theta} \end{bmatrix} = \underbrace{\begin{bmatrix} \dot{x} \\ \dot{\theta} \\ f_1(\theta)\ F_B(\dot{x}, \dot{\theta}) + f_2(\theta, \dot{\theta}) \end{bmatrix}}_{f(\mathbf{x})} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ g_1(\theta) \end{bmatrix}}_{g(\mathbf{x})} i_a, \tag{3.39}$$

where

$$f_1(\theta) = -\begin{bmatrix} \left(m_c + m_b \sin^2(\theta)\right)^{-1} & \cos(\theta)\left(l\left(m_c + m_b \sin^2(\theta)\right)\right)^{-1} \\ \cos(\theta)\left(l\left(m_c + m_b \sin^2(\theta)\right)\right)^{-1} & (m_c + m_b)\left(m_b l^2\left(m_c + m_b \sin^2(\theta)\right)\right)^{-1} \end{bmatrix}, \tag{3.40}$$

$$F_B(\dot{x}, \dot{\theta}) = \begin{bmatrix} B_{v,c}\dot{x} + \mathcal{F}_{c,c} \tanh\left(k_{tanh}\dot{x}\right) \\ B_{v,p}\dot{\theta} + \mathcal{F}_{c,p} \tanh\left(k_{tanh}\dot{\theta}\right) \end{bmatrix}, \tag{3.41}$$

$$f_2(\theta, \dot{\theta}) = -\begin{bmatrix} \frac{m_b l \sin(\theta)\dot{\theta}^2 - g m_b \sin(\theta)\cos(\theta)}{m_c + m_b \sin^2(\theta)} \\ \frac{m_b l \sin(\theta)\cos(\theta)\dot{\theta}^2 - g \sin(\theta)(m_c + m_b)}{l\left(m_c + m_b \sin^2(\theta)\right)} \end{bmatrix}, \tag{3.42}$$

$$g_1(\theta) = \frac{K_t}{r}\begin{bmatrix} \left(m_c + m_b \sin^2(\theta)\right)^{-1} \\ \cos(\theta)\left(l\left(m_c + m_b \sin^2(\theta)\right)\right)^{-1} \end{bmatrix}. \tag{3.43}$$

For the system, the measurements through the encoders describe the two states, x and θ. Therefore the output equation is given as

$$y = \begin{bmatrix} x \\ \theta \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}}_{h(\mathbf{x})} \mathbf{x}, \tag{3.44}$$

however with quantization noise as further described in section 4.3.

The parameters B_v, $\mathcal{F}_c$ and m_c are estimated in the following section in order to have a model fitting the test setup.

3.6 Parameter estimation

This section will describe the parameter estimation of the test setup in the laboratory. The parameter estimation is carried out in several parts in order to limit the number of parameters to estimate at a time. This is done by estimating the parameters for the pendulum and the cart separately and afterward validating and adjusting the parameters for obtaining a combined model.

Several estimations have been performed, and it has turned out that the estimations needed to be done iteratively in order to get a good fit. The full procedure can be seen in appendix B, whereas only a summary of the methods used and final results will be presented here.

For the cart, three parameters need to be estimated, which are the mass of the cart, m_c, the viscous friction, $B_{v,c}$, and the Coulomb friction, $\mathcal{F}_{c,c}$. Firstly, the two friction coefficients are estimated by having the cart moving at a constant velocity and then using the force applied at that velocity. This is done several times and the model is regressed, such that the gradient is the viscous friction coefficient and the Coulomb friction is the crossing with $\dot{x} = 0$. Here the two friction coefficients are found to be direction dependent.

Afterward, the mass of the cart is estimated. This is done by applying a force, and based on the dynamic equations for the acceleration, estimating the cart mass, with the MATLAB function *greyest* from the System Identification Toolbox. These estimations have given results varying 1,3 kg and thereby the mean has been used. However, this could indicate a uncertainty on the cart mass, which the controller should be able to handle.

A sub-validation test is then performed, where it is found necessary to adjust the estimated parameters in order to make the model fit. The validation can be seen in figure 3.8, where the fitted parameters can be seen in table 3.1.

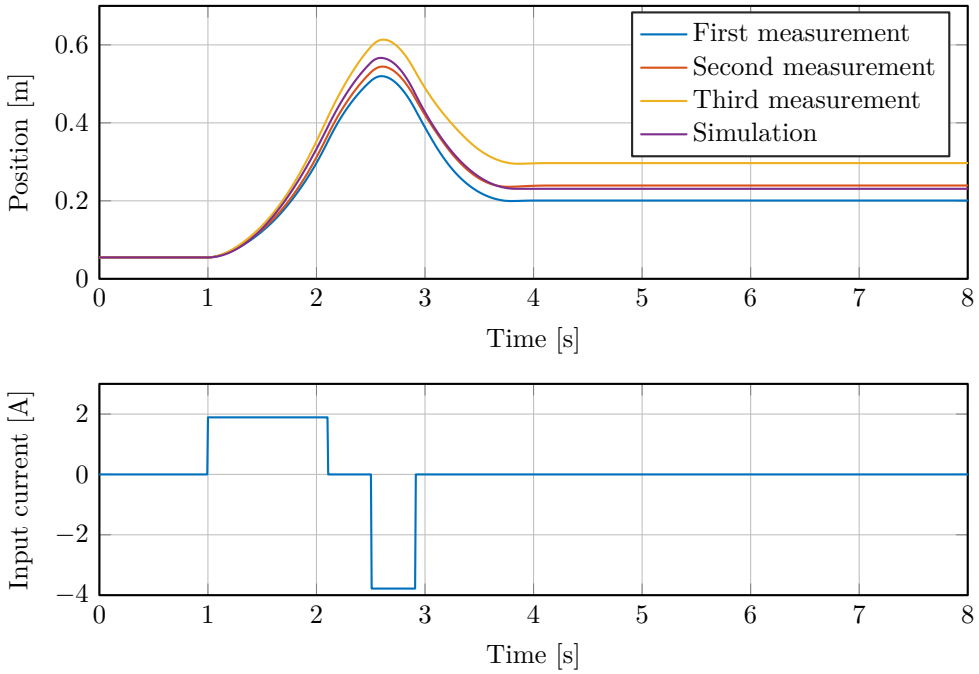


Figure 3.8. Validation of cart parameters.

It can here be seen that the system differs between measurements, where the simulated model fits well between the measurements. It has in general been noted, that at lower velocities, the consistency of the measurements is low, as seen in figure B.6. This is expected due to varying frictions along the rail, from where different initial positions of the cart, will yield different end positions, as it has been subject to different frictions along the test.

For the pendulum, the two frictions coefficient parameters, $B_{v,p}$ and $\mathcal{F}_{c,p}$, are to be estimated. Firstly, the Coulomb friction is estimated by lifting the pendulum to the highest angles, where it does not swing down. The gravitational pull in the pendulum at these angles equals the Coulomb friction force.

The viscous friction is estimated by letting the pendulum swing and estimating the coefficient through an exponential decay, as an underdamped harmonic oscillator.

When doing a sub-validation test for the pendulum it is observed, that it again is necessary to adjust the parameters to fit the response of the pendulum. This fitting is expected to be necessary, as the individual estimation methods do not take both frictions forces into account. The sub-validation with corrected parameters for the pendulum can be seen in figure 3.9 and the related parameters can be seen in table 3.1 as well.

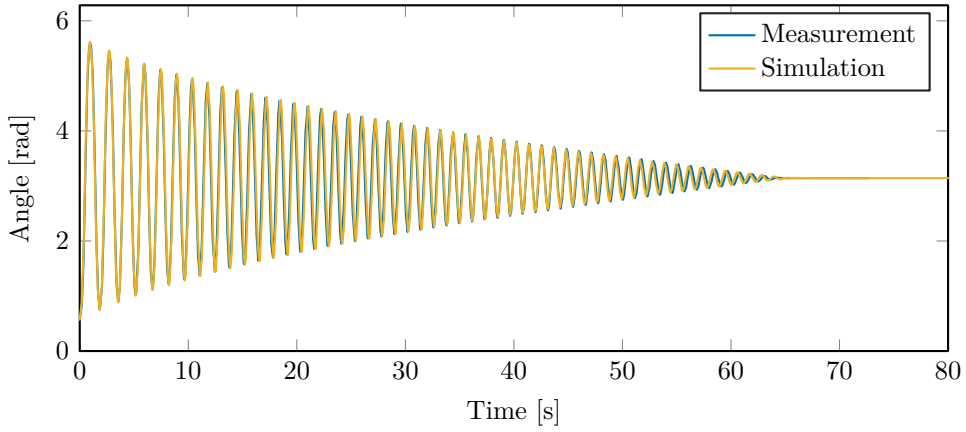


Figure 3.9. Verification of the adjusted model for the pendulum parameters.

The estimated parameters for both the cart and pendulum can be seen in table 3.1.

Parameter	$B_{v,c}$	$\mathcal{F}_{c,c}$	m_c	$B_{v,p}$	$\mathcal{F}_{c,p}$
Value	$1.937\ \frac{\text{N}}{\text{m/s}}$ if $\dot{x} > 0$, $1.422\ \frac{\text{N}}{\text{m/s}}$ if $\dot{x} < 0$	3.021 N if $\dot{x} > 0$, 2.746 N if $\dot{x} < 0$	5.273 kg	$0.0004\ \frac{\text{Nm}}{\text{rad/s}}$	0.004 Nm

Table 3.1. Final estimated parameters for the model of the IPC system.

Thereby the parameters for the system has been estimated, and combining them with the coefficients in table 3.2, they compromise all parameters for the state space system, in equations 3.39 to 3.43.

Parameter	m_b	l	K_t	r
Value	0,201 kg	0,3235 m	$0{,}0934\ \frac{\text{Nm}}{\text{A}}$	0,028 m

Table 3.2. Measured parameters for the system.

3.7 Model validation

In this section, the full model will be verified, meaning both the cart and the pendulum, as a final verification of the sub-verifications in the previous section.

The verification is done by use of two tests. The first verification test is by use of an applied control force. The pendulum has an initial condition in the equilibrium point straight down and the system is actuated with a control pulse as seen in figure 3.10.

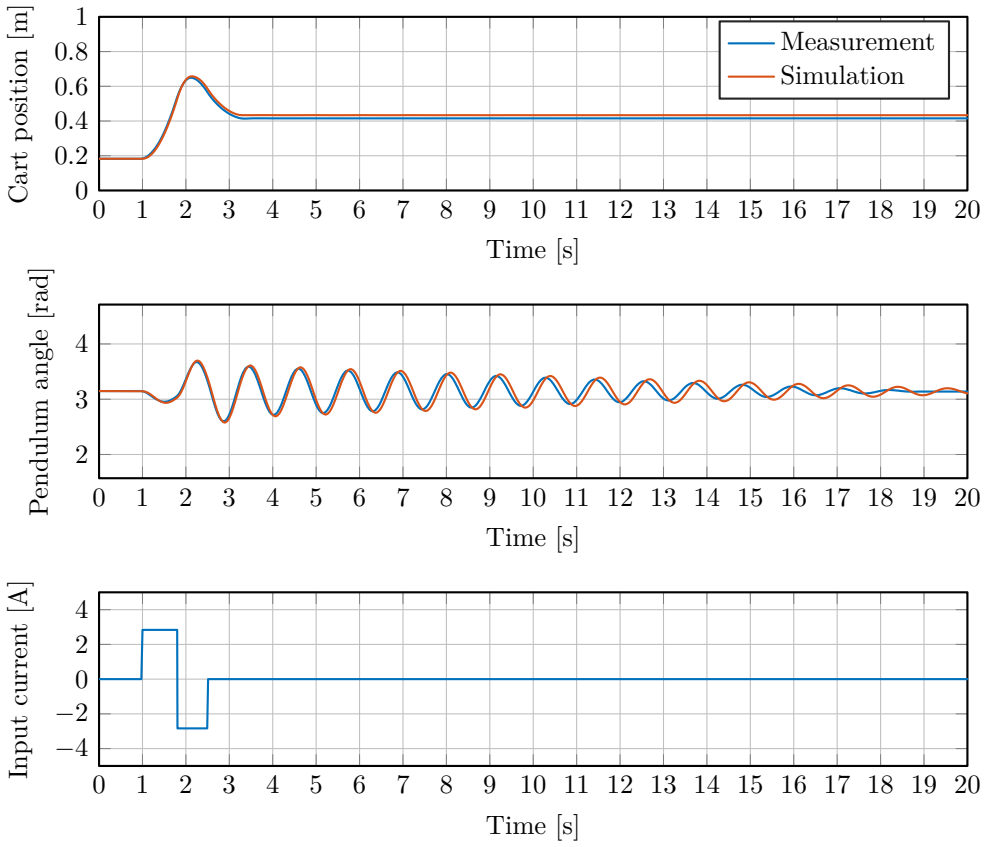


Figure 3.10. Validation of modelled system, given pulse input.

It can be observed that the simulated model fits the system response quite well, even though the phase of the pendulum drifts a little during the test. This is however deemed acceptable for the simulation of the system, as the drift is minor and after a long time.

The second test is from an initial pendulum angle, without a control force from the motor. This can be seen in figure 3.11.

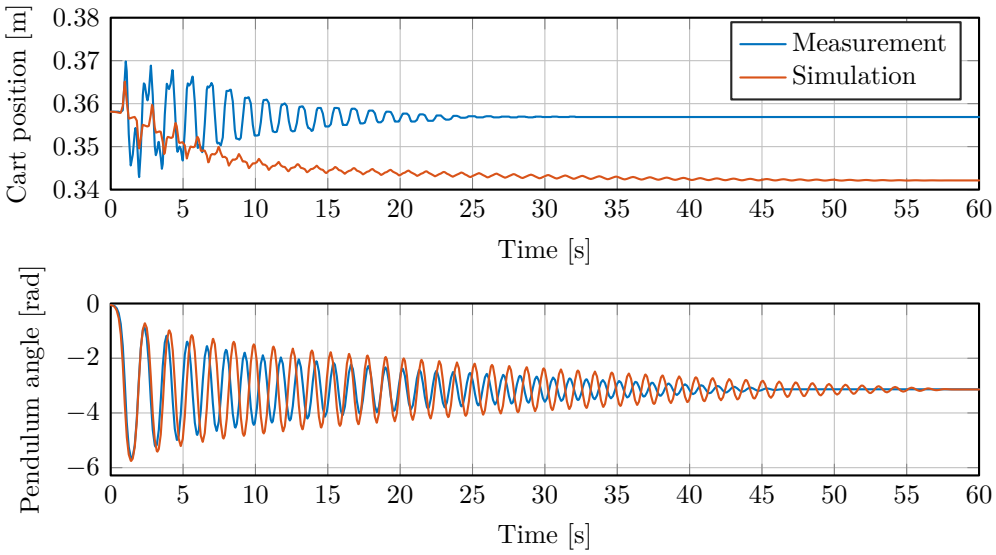


Figure 3.11. Validation of modelled system given no control inputs, from an initial angle of the pendulum.

Here it can be observed, that the model does not fit as well. For the cart position, it can be seen that the form of the simulation for the approximately first 5 seconds fits well. However, after this the measurements seem to get flat in the peaks, which the simulation does not. It is expected that the measurements react in this manner, contrary to the simulation, due to unmodelled static friction in the cart. Furthermore, the different form and amplitude of the cart movement alters the fit of the pendulum swing, as it can be seen that both the amplitude of the pendulum angle and the period of the swing, mismatches.

It can also be seen that the modelled cart drifts ≈1,5 cm to one side, which the system does not. This is due to the fact, that the system has varying frictions along the rail, which is confirmed by redoing the test at different initial cart position. This can be seen in figure 3.12, where the initial positions have been offset to zero, in order to easily visualize the drift.

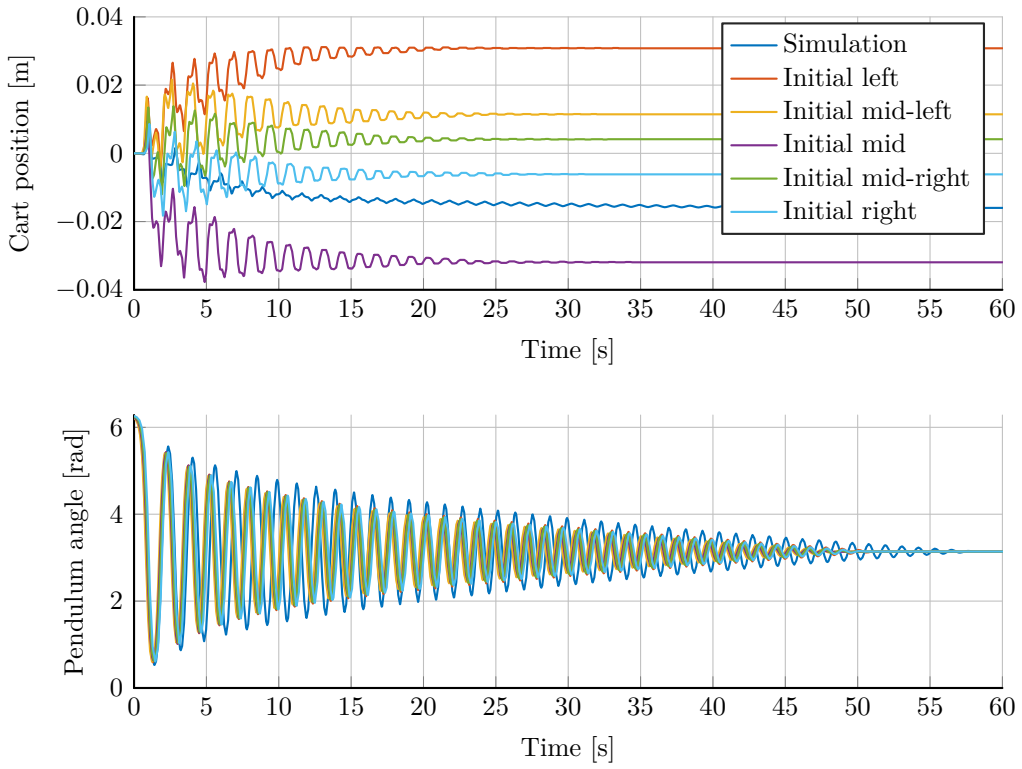


Figure 3.12. Drift of cart, given different starting positions on rail. All initial positions have been offset to start at the same place graphically.

It can here be seen, that the drift both varies in size and in direction, depending on initial position. The simulation drifts to the side because it has different frictions depending on the direction of movement. It can also be seen that the system some places drifts more than others and that the simulation is somewhere between those.

However as the fit in figure 3.10 is good, the model is deemed fitting for this thesis because the use-case of the model is control, where inputs are applied and not open loop fitment from an initial angle. However, the oscillation test still fits acceptably especially in the first couple of seconds and the model and parameters are thereby verified.

With the model verified and deemed sufficiently fitting for controller design, the next section will cover the implementation of the simulation setup.

3.8 Implementation of simulation environment

In this section, a brief introduction to the implemented simulation environment is given.

A simulation of the IPC has been implemented in MATLAB, based on the equations of motion in equation 3.39 to 3.43 with the parameters presented in table 3.1 and 3.2. The simulation is made by use of an ODE45 solver. In order to have a simulation environment as close to the real setup as possible, the encoders have been implemented based on the simulated state vector. The count from the decoder ICs is implemented, such that the cart position and pendulum angle is

represented as a counter similar to the real IPC test setup, as described in section 2.2, and with the same quantization size. This allows for testing of algorithms, as close to reality as possible in the simulation environment. As encoders are used, no classic Gaussian measurement noise is added to the measurements in terms of the encoder counters, as this is not present in an encoder sensor setup. This is further described in section 4.3.

In section 3.2 it has been pointed out in a footnote, that it has been necessary to increase the precision of the solver, in order to obtain the simulation result of a harmonic oscillator. The relative error tolerance were raised from the default 10^{-3} to 10^{-6} and the absolute error tolerance from the default 10^{-6} to 10^{-8}. This is however changed for a maximum allowed time step for the ODE solver, as this yields the iterations of the ODE function much closer to the sampling frequency, which is necessary for calculating the correct encoder count to each time, as well as ensuring the correct sampling time of the controller and state estimator, when this is to be implemented in the simulation. The *MaxStep* option has thereby been limited to $T_s/4$.

An animation based on the simulated state vector has been developed, in order to give a representation of the behavior of the system, which is easy to relate to, compared to normal graphs. An instance of the animation can be seen in figure 3.13 and the function to generate and play the animation can be found as the digital attachment. The animation simply needs the state vector as well as the sampling time.

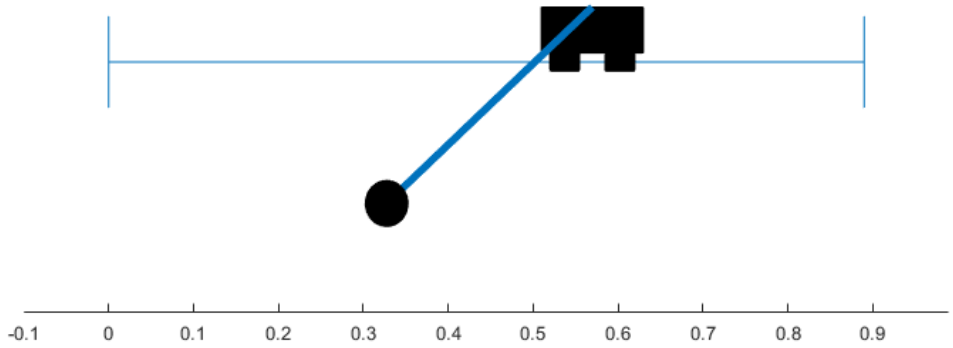


Figure 3.13. An instance from the developed animation based on the simulated state vector.

4 Friction compensation, overall structure and state estimation

This chapter will first describe the friction compensation in the cart, in order to simplify the model for the controller design. Afterward the overall control structure and concept of the thesis will be presented. At the end, the state estimation needed for the non-linear controller is described, designed and tested.

4.1 Coulomb friction compensation

In this section, the compensation for Coulomb friction in the cart will be described. It is desired to compensate for the Coulomb friction in the cart in order to slightly simplify the non-linear model.

Friction often causes big problems in control designs for mechanical systems as it can cause large tracking errors [8]. Many different approaches for friction compensation is developed depending on the necessity of precision in the control system, involving both advanced friction models and adaptive schemes. In section 3.5, the friction is modelled to a deemed sufficient extent for this thesis, which is fairly standard, and thereby the friction compensation is also made relatively simple, as it is not intended to be perfect, but assist the normal control.

It is desired to compensate for the Coulomb friction, as it is discontinuous in zero velocity and is thereby approximated with a tanh function in order to obtain close to the same behavior, but with a continuous function. However, this is very non-linear and instead of dealing with this term in the controller design, it is desired to compensate for it.

Since the Coulomb friction in the cart enters the system directly in the direction as the applied control force, it is possible to compensate for the Coulomb friction in the cart through the control force. This will then make it possible to remove the Coulomb friction from the model used for the controller and estimation design. This can be done by estimating the friction force and then adding the opposite of that friction to the control force, which cancels the friction [25].

This can be approximated in a feedforward manner, based on the control signal, such that extra actuation is added after the controller, in order to cancel the Coulomb friction, as seen in figure 4.1 [24].

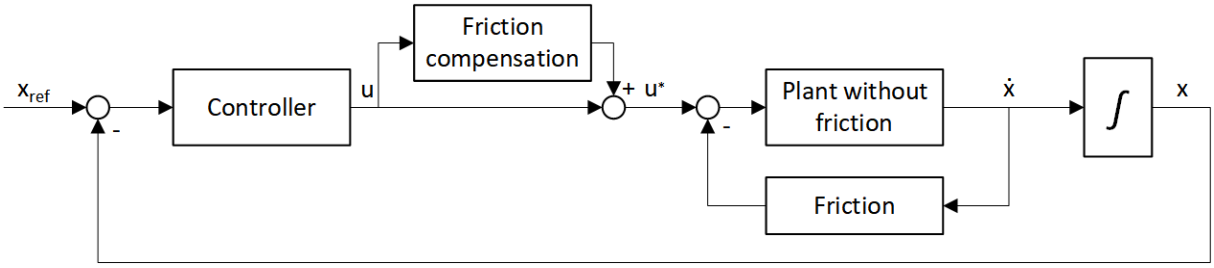


Figure 4.1. Friction compensation through feedforward of the control signal.

The friction compensation will then look like

$$u^*(t) = \begin{cases} u - u_{neg,fric} & \text{if } u < 0 \\ 0 & \text{if } u = 0 \\ u + u_{pos,fric} & \text{if } u > 0 \end{cases}, \tag{4.1}$$

where

$$u_{neg,fric} = \frac{r}{K_t}\mathcal{F}_{c,c_{neg}}, \tag{4.2}$$

$$u_{pos,fric} = \frac{r}{K_t}\mathcal{F}_{c,c_{pos}}. \tag{4.3}$$

This feedforward compensation, however only works as intended, as long as the movement of the system is in the same direction as the actuation. That is because, when actuation direction changes during movement, the compensation fails during the timespan the cart continues to move in the same direction due to the inertia. Thereby the sign of this compensation changes, regardless of the direction of system movement and will thereby not be theoretically correct as the Coulomb friction depends on the movement of the cart and not the control signal. This can cause problems for systems, whose input direction changes often, as it does for stabilization of the inverted pendulum.

The friction is as described in equation 3.38 dependent on the velocity and thereby the friction compensation can be made as a feedback dependent on the velocity, in order to resolve the above problem. The concept of this can be seen in figure 4.2 [25].

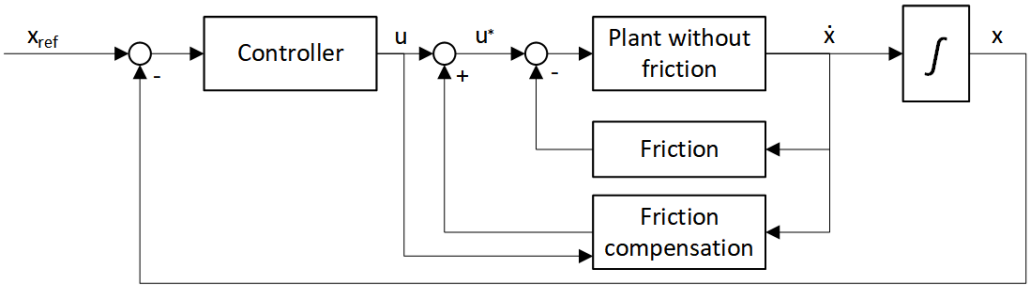


Figure 4.2. Friction compensation through velocity feedback.

The compensation is thereby

$$u^*(t) = \begin{cases} u - u_{neg,fric} & \text{if } \dot{x} < 0 \vee (\dot{x} = 0 \wedge \ u(t) < 0) \\ 0 & \text{if } \dot{x} = 0 \wedge \ u(t) = 0 \\ u + u_{pos,fric} & \text{if } \dot{x} > 0 \vee (\dot{x} = 0 \wedge \ u(t) > 0) \end{cases} . \tag{4.4}$$

The original control signal from the controller is needed in order to handle the zero-velocity situation.

An investigation has been carried out to test whether the inertia in the system is significant enough to justify the use of the more advanced feedback-based compensation. This can be seen in appendix C, where several tests have been carried out to see the timespan before the velocity changes sign, given a square wave input. It is in the appendix deemed, that the inertia yields significant enough delays for movement direction changes given the expected actuation range, that the compensation in equation 4.4 should be used. Furthermore, the chosen non-linear controller is a sliding mode controller, which will have a rapidly changing control signal due to its switching behavior, even though the movement of the cart is not changing according to this. This can be seen in chapter 5, which also implies the later compensation would be the best choice.

This compensation is implemented on the system and works as intended. The needed cart velocity for the compensation comes from the state estimation designed in section 4.3, as this is also needed for the controller. In the following section, the overall control and program structure is presented.

4.2 Overall controller and program structure

This section gives an overview of the control and program structure of the thesis, such that the individual parts can be designed in the following sections and chapters.

This thesis will consist of two different types of controllers, a non-linear controller and a machine learning controller, MLC, which it should be possible to switch between in order to easily demonstrate and compare them.

The interface is designed in order to handle user commands and switch to the desired controller as well as handling resets of the necessary variables in the code, both under startup and when shifting between controllers. The user can also set the reference position for the cart through the interface. The structure can be seen in figure 4.3.

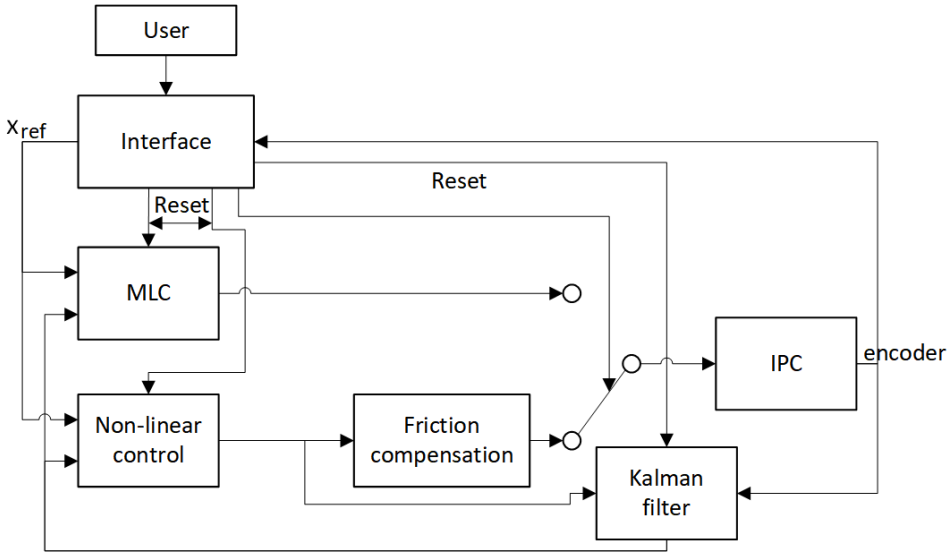


Figure 4.3. The overall control and program structure and interface.

The friction compensation is implemented for the non-linear controller and utilizes the estimated velocity as described in section 4.1. The state estimation is in terms of a Kalman filter, as described in section 4.3, and gets the uncompensated control signal, such that it can be designed without the Coulomb friction on the cart. Intentionally, it was wanted to feed the raw encoder signals directly to the MLC, as well as no friction compensation, in order to see how universal it is and how it compares to the classical model based control. However, due to time constraints it has been necessary to use the Kalman filter for getting full state information for the MLC approach as well, in order to try it on the real IPC. This is further described in chapter 6.

Multiple different versions of the MLC and non-linear controllers can be implemented in parallel, e.g. controllers with different tuning.

As seen in figure 4.3, it is wanted to be able to set the cart reference position, x_{ref}, through the user interface, such that the balancing position of the pendulum can be changed by the user.

The code executed on the MCU has the flow structure seen in the flowchart in figure 4.4. At startup the MCU's I/O pins are set up and the needed variables and objects in the code are created and initialized. After this, the code waits for a reset from the user, where the cart should be placed all the way to the left of the rail and the pendulum hanging straight down, from where the counters in the decoder IC's are reset for having a known reference [1].

[1] To ensure the pendulum is hanging straight down it might be necessary to pound the table a couple of times to overcome the Coulomb friction.

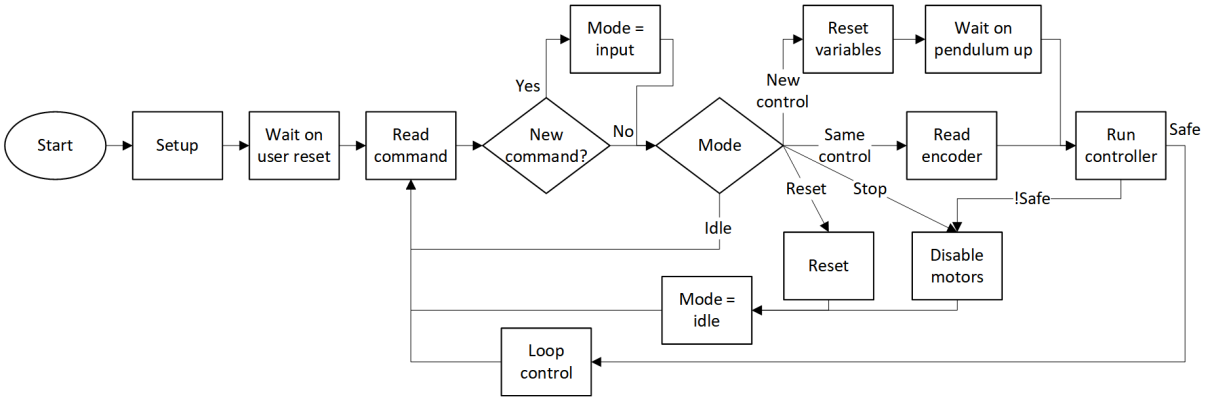


Figure 4.4. Overall flow chart for the MCU.

As multiple controllers need to be implemented as previously described, it is chosen to separate the code by use of modes. The user can determine the mode of the program through the user interface. The startup mode is idle, where the MCU waits on a new user command. This is either a choice of controller, reset in order to reset the decoder ICs as well as the internal variables if wanted after the initial reset, or stop of the motors. When a new controller is chosen, the code waits on the pendulum manually being lifted into $\pm 0,15$ rad $\approx \pm 8{,}6°$ and ± 10 cm from the middle of the rail, before the controller starts to stabilize the pendulum. This is because no swing up controller is designed in this thesis. The pendulum should be lifted in the clockwise direction, as this will give an upright angle of zero, as also defined in section 3.2. Furthermore, a safety feature is implemented on the pendulum angle when the controllers are active, such that if the pendulum angle gets bigger than $\pm 0,6$ rad $\approx \pm 34{,}6°$, the motor is disabled and the state set to idle, as either something is wrong or such a big disturbance has been introduced into the system, that it is deemed inappropriate to try to catch the pendulum given the limited rail length. In the end, busy waiting is implemented in order to obtain the correct loop time for the control system.

With the structure described, the state estimation is designed in the following section and the two controllers are designed in the following two chapters.

4.3 Velocity estimation based on encoders

In this section, the state estimation problem of the IPC is described, as well as the chosen estimator defined. The estimation is used both for the friction compensation as well as for both controllers as described in section 4.2.

As described in section 2.2, the position and angle sensors are two quadrature encoders, which utilizes decoding ICs, such that the encoders from the MCU perspective, acts as a counting encoder. This is also implemented in the simulation as described in section 3.8. The use of encoders introduces a quantization error in the measurement, which equals the half of the quantization step, under the assumption that the middle of an encoder step is precisely at the reference position, which for the cart is 0 and π for the pendulum. The quantization is illustrated in figure 4.5, where the error of half a quantization step in the ideal case can be seen. The quantization steps are in equation 2.1 found to be $\pi \cdot 10^{-3}$ rad/count for the pendulum angle and $8,80 \cdot 10^{-5}$ m/count for the cart position in equation 2.2.

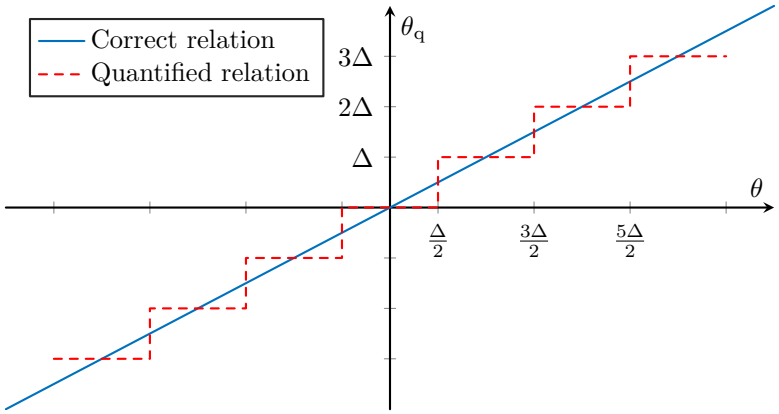


Figure 4.5. The quantitation error between the real angle, θ, and the quantified angle, θ_q, is the difference between the straight line and the quantization function.

The decoder ICs' counters are reset in order to have a reference in the system, however it is in reality not known how whether the quantization is in the middle as in figure 4.5 or is offset, which in the worst case will increase the error up to the quantization step size itself. This can cause a bias in the calculated position in relation to the real one, which especially for the pendulum angle can affect the system performance. The offset is ideally $\frac{\Delta}{2}$, but can vary in the interval $[0, \Delta)$. In the simulation, the ideal offset has been used. The quantization function for figure 4.5 is seen in equation 4.5. It should be noted, that the count from the decoding ICs in the IPC relates the quantified angle as $count \cdot \Delta = \theta_q$, as seen in figure 4.5.

$$\theta_q = \text{floor}\left(\frac{\theta + \frac{\Delta}{2}}{\Delta}\right)\Delta \tag{4.5}$$

As both the angular velocity for the pendulum as well as the velocity of the cart are needed, these need to be estimated based on the encoder information representing the position. The most classic approach for this is to use the derivative approximation as shown in equation 4.6.

$$\dot{f}(t) = \frac{df(t)}{dt} = \lim_{\Delta t \to 0} \frac{f(t) - f(t - \Delta t)}{\Delta t} \approx \frac{f_k - f_{k-1}}{T_s} = \dot{f}_k \tag{4.6}$$

Here f_k denotes the function f to the time instance $k \cdot T_s$. However, this approach is problematic as noise normally is present, which causes big derivatives. Given that encoders are used, no normal Gaussian measurement noise is present in the system, however the quantization still causes problems with this approach, as an error between the real and the measured is still present as previously described. This is often referred to as quantization noise. Utilizing the approach of equation 4.6 will result in a quantization of the estimated velocity as well, as shown in equation 4.7, where the information about 2000 count per turn is utilized [43]. This method is called frequency measurement.

$$\dot{\theta} = \frac{d\theta}{dt} \approx \frac{2\pi \cdot (\text{count}_k - \text{count}_{k-1})}{T_s \cdot 2000} \tag{4.7}$$

This means, that the quantization step of the velocity is depended on the sampling frequency, the faster sampling, the bigger quantization and thereby the bigger the risk of not getting encoder pulses between samples. Due to this, and the reasons mentioned in section 2.2, the sampling frequency for this thesis is chosen to 150 HZ. This yields a quantification of the velocities as seen in equations 4.8, given that only one encoder count is changed between samples and under the assumption of constant velocity between sampling times.

$$\Delta\dot{\theta} = \frac{\Delta\theta}{T_s} = \frac{\pi \cdot 10^{-3}}{6,667 \cdot 10^{-3}} \approx 0,471 \frac{\text{rad}}{\text{s}} \tag{4.8a}$$

$$\Delta\dot{x} = \frac{\Delta x}{T_s} = \frac{8.8 \cdot 10^{-5}}{6,667 \cdot 10^{-3}} \approx 0,0132 \frac{\text{m}}{\text{s}} \tag{4.8b}$$

It can be seen, that the quantization and thereby also the error is independent from the velocity of the system. This also means, that the relative error increases as the velocity gets smaller, as it can be seen in figure 4.6, where the pendulum is simulated from an initial angle. This is a problem, as this thesis focuses on stabilization of the pendulum and thereby operates at low velocities. It is clear to see, that the above mentioned quantization is not acceptable and needs to be improved in order to get acceptable estimation at low velocities.

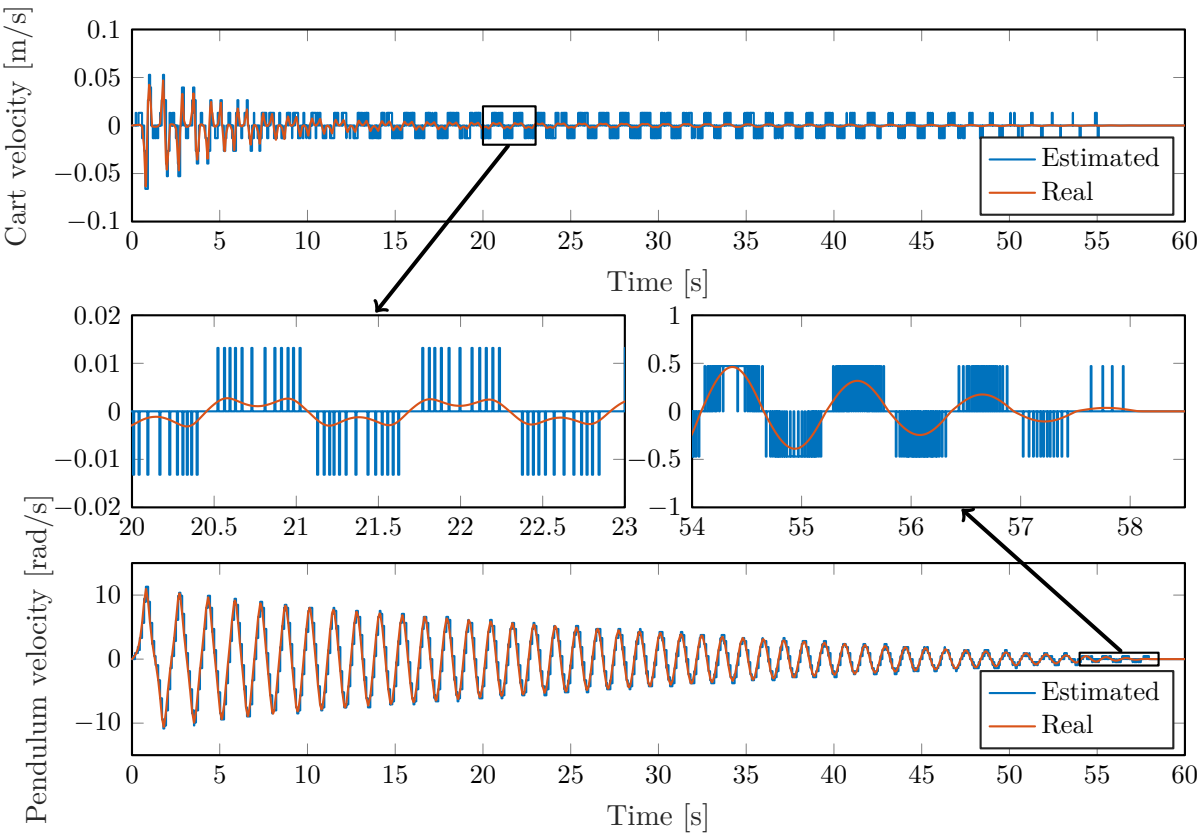


Figure 4.6. Simulation of open loop initial position from pendulum angle 0,1 radian, showing the quantization of the velocity.

Multiple methods for overcoming the quantization problem exists. The easiest is to low-pass filter the velocities estimated on behalf of the previously mentioned method, however such filters introduce delays in the estimation, where the higher the filter order the better smoothing, but the larger delay. This could e.g. be based on a moving average, MA, filter.

Other methods include time stamping of the encoder event, such that the period time of the encoder pulses can be determined more precise, this is referred to as period measurement [43]. This however suffers from problems with imperfection in the encoders such as varying transition locations and phase errors between the two encoder channels. In the test setup, the interrupt pins for this purpose from the quadrature decoder ICs on the shield are not routed to the MCU and this method can thereby not be utilized without modifying the test setup.

The chosen method for estimating the velocities are thereby by using a state observer as a proper and well-fitting dynamic model is known and thereby the control force can be taken into consideration in the state estimation as well. As state estimator the Kalman filter is chosen, as this is the optimal linear estimator for a non-deterministic dynamic system subject to Gaussian noise. In this case, the quantification is treated as measurement noise, which has shown good results for velocity estimation based on encoders [11] [43]. This is discussed in the following section, as well as an extension of the linear Kalman filter, KF, to a non-linear version for dealing with the non-linearities in the IPC.

4.4 State estimation by extended Kalman filter

The obtained model for the IPC in equation 3.39 to 3.43, describes the system as an ordinary differential equation, ODE. However, as described in section 4.3, the encoders introduce noise, v. Moreover, process model errors, w, can be introduced as process noise, such that the system is described in terms of stochastic differential equations, SDE. The standard form for a discrete linear SDE is,

$$x_{k+1} = \Phi x_k + \Gamma u_k + w_k, \tag{4.9}$$

$$y_k = H x_k + v_k, \tag{4.10}$$

where the subscript denotes the discrete time instance $t = k \cdot T_s$, Φ, Γ and H describes the discrete dynamics of the system, and v_k and w_k are independent and identically distributed (i.i.d.) processes. The two noise processes fulfill

$$w \in \text{NID}(0, Q), \tag{4.11}$$

$$v \in \text{NID}(0, R), \tag{4.12}$$

$$E\left(w_k v_l^T\right) = 0, \tag{4.13}$$

where the parameters Q and R are the covariance matrices for the two noise processes, which both are normally identically distributed, NID, and uncorrelated with each other.

The linear Kalman filter, KF, is the optimal state estimator for such a system, meaning it is unbiased and the one yielding the lowest means squared error [27], MSE,

$$\text{MSE} = E\left(\left(x_k - \hat{x}_k\right)^T M \left(x_k - \hat{x}_k\right)\right), M > 0. \tag{4.14}$$

The recursive algorithm for the KF with initial conditions

$$x_0 \in N\left(\hat{x}_{0|-1}, P_{0|-1}\right) \tag{4.15}$$

can be seen in equation 4.16 to 4.21 [27] (slide 23).

$$\tilde{y}_{k|k-1} = y_k - \hat{y}_{k|k-1} = y_k - H\hat{x}_{k|k-1} \tag{4.16}$$

$$K_k = P_{k|k-1}H^T\left(HP_{k|k-1}H^T + R\right)^{-1} \tag{4.17}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k\tilde{y}_{k|k-1} \tag{4.18}$$

$$P_{k|k} = \left(I - K_kH\right)P_{k|k-1}\left(I - K_kH\right)^T + K_kRK_k^T \tag{4.19}$$

$$\hat{x}_{k+1|k} = \Phi\hat{x}_{k|k} + \Gamma u_k \tag{4.20}$$

$$P_{k+1|k} = \Phi P_{k|k}\Phi^T + Q \tag{4.21}$$

The notation $_{k|k-1}$ denotes the prediction to time k based on the measurements and the control input from time $k-1$ and before. The first four equations are called the update step, which are based on the received measurements y_k. The last two equations are called the prediction step, where the input, u_k, is used in order to predict the state at the next time instance.

First, the output error, $\tilde{y}$, is calculated based on the measurements and the state prediction from the last run-through of the KF. Then the Kalman Gain, K, is calculated based on the error covariance, P, and the covariance of the measurement noise. The Kalman gain dictates how much the measurements are trusted in the update step of the state estimate in equation 4.18. The last update step updates the error covariance matrix based on the predicted covariance matrix propagated through the output matrix, H, the Kalman gain and the measurement noise covariance. In the prediction steps, the state is predicted based on the noiseless state equation and the error covariance is predicted through the system matrix and the process covariance.

The obtained model for the IPC is however non-linear and the KF does thereby not directly comply with this system. Given that this thesis focuses on stabilization of the IPC, the angle of the pendulum should be small and thereby small angle approximation could be used as well as a first order Taylor expansion for the rest of the non-linear terms. However, as the model includes $\sin^2(\theta)$, this limits the range where the small angle approximation is valid as well as a first order approximation of the Coulomb friction for the pendulum, which includes the *tanh* function, will act like a viscous friction. Furthermore, the squared angular velocities will be linearized in 0 rad/s, meaning they will not contribute, even though the angular velocity could get quite high under actuation when stabilizing the pendulum and at the same time applying reference changes in the cart position. The system has been linearized in $\bar{x} = \begin{bmatrix}0 & \pi & 0 & 0\end{bmatrix}^T$, $\bar{i}_a = 0$, and is shown

in appendix D.1. It is linearized for the pendulum hanging down, for these open loop tests, but if to be implemented on the system as an estimator for the closed loop controller, the linearization needs to be moved to the pendulum pointing up. As a part of the tuning of the KF in the simulation, the pendulum frictions have been adjusted, because the linearized Coulomb friction acts as a very large viscous friction, which thereby does not fit the model. The adjusted viscous friction coefficient is thereby

$$B_{v,p,kalman} = 0,0404 \quad \frac{\text{Nm}}{\text{rad/s}}. \tag{4.22}$$

The tuned covariance matrices are

$$\begin{aligned} Q_{lin} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 500 & 0 \\ 0 & 0 & 0 & 300 \end{bmatrix}, \\ R_{lin} &= \begin{bmatrix} 2 & 0 \\ 0 & 40 \end{bmatrix}. \end{aligned} \tag{4.23}$$

The simulation can be seen in figure 4.8, where it can be seen that the KF is estimating the overall tendency, however not that well, especially the estimation of the angular velocity is off, as it can be seen on the residual in figure 4.9. The velocity estimates are with the chosen parameters slightly delayed and the especially the angular velocity is having problems with amplitude. It is not possible to obtain good fitment both under actuation and without actuation and thereby a compromise, with focus on the actuation, has been made. The only way found to improve in both scenarios is by increasing the process noise variance of the velocities. However, this introduces more noise in the estimations and the estimates tend towards the quantization as for the direct derivative, which makes sense, as the measurements are trusted more. This also shows in the end, where the estimate is significantly affected for every new encoder pulse. These problems are expected mainly to be because of the friction problem.

Initially, the KF showed quite poor results, and thereby an extended Kalman filter, EKF, were designed to improve the estimate, which is shown in the following. An error in the KF was later detected such that the shown performance was obtained, however the EKF still shows significant improvements and is thereby kept and used in the report. This will lead to a significant increase in computations, which however is not deemed a problem due to the floating-point unit support of the MCU.

The EKF is therefore used, which is the direct non-linear expansion of the KF, [28]. Thereby the EKF is not limited to working in a small signal range as the KF would have been and can utilize the non-linear model. The difference between the two is thereby, that when predicting the state and output, the non-linear functions are used directly. For propagating the covariance matrices and in the Kalman gain, an online linearization at each time sample is utilized as an approximation. This is done by means of the Jacobians, to get the gradient at the specific time sample. The EKF is based on a system on the form,

$$x_{k+1} = f_k(x_k, u_k) + \omega_k, \tag{4.24}$$

$$y_k = h_k(x_k) + v_k. \tag{4.25}$$

The EKF with initial condition

$$x_0 \in N\left(\hat{x}_{0|-1}, P_{0|-1}\right) \tag{4.26}$$

is thereby given as the following [28] (slide 7),

$$\tilde{y}_{k|k-1} = y_k - h_k\left(\hat{x}_{k|k-1}\right), \tag{4.27}$$
$$K_k = P_{k|k-1}H_k^T\left(H_k P_{k|k-1} H_k^T + R\right)^{-1}, \tag{4.28}$$
$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k\tilde{y}_{k|k-1}, \tag{4.29}$$
$$P_{k|k} = \left(I - K_k H_k\right)P_{k|k-1}\left(I - K_k H_k\right)^T + K_k R K_k^T, \tag{4.30}$$
$$\hat{x}_{k+1|k} = f_k(x_k, u_k), \tag{4.31}$$
$$P_{k+1|k} = F_k P_{k|k} F_k^T + Q, \tag{4.32}$$

where H_k is the Jacobian of the output equation, in equation 4.28 to the newest predicted state, $h\left(\hat{x}_{k|k-1}\right)$, and in equation 4.30 to the newest state update $h\left(\hat{x}_{k|k}\right)$. F_k is the Jacobian of the state equation, $f\left(\hat{x}_{k|k}, u_k\right)$ using the most recent state estimate.

It should be noted, that the EKF is not an optimal estimator for non-linear systems but is however known to in general give good results for slightly non-linear systems, as the IPC [11] (chapter 5), [43]. Another note is that the EKF is not guaranteed to be stable due to the linearization, but several examples of EKFs for IPC setups have been made, with good results. Furthermore, the initial values of the EKF can have an impact on the performance, as far away initial values can make the EKF diverge due to the linearization. However, as described in section 4.2 the cart position and angle is fairly precisely known when the filter is enabled, due to the controller and filter not starting before the pendulum is lifted inside an angle close to 0.

As the model derived in chapter 3 is continuous and the EKF is implemented as a part of a discrete time control system, the model is to be discretized. This is done with the Forward Euler method as seen in the following equation, where T_s is the sampling time:

$$x_{k+1} = \mathbf{x}_k + T_s\left(f\left(\mathbf{x}_k\right) + g(\mathbf{x}_k)u_k\right) \tag{4.33}$$
$$h_k\left(\mathbf{x}_k\right) = h\left(\mathbf{x}_k\right) \tag{4.34}$$

The Jacobians are found analytically and their derivation can be seen in appendix D.2. With those Jacobians, the EKF is implemented in simulation in MATLAB. In figure 4.8 a simulation with a step input in figure 4.7 for both the KF along with the EKF is seen. In the tests shown below, both for the simulation and follow-up test, the compensation is included, however the added current is not to be seen from the state estimators point of view, and thereby not shown in the graphs.

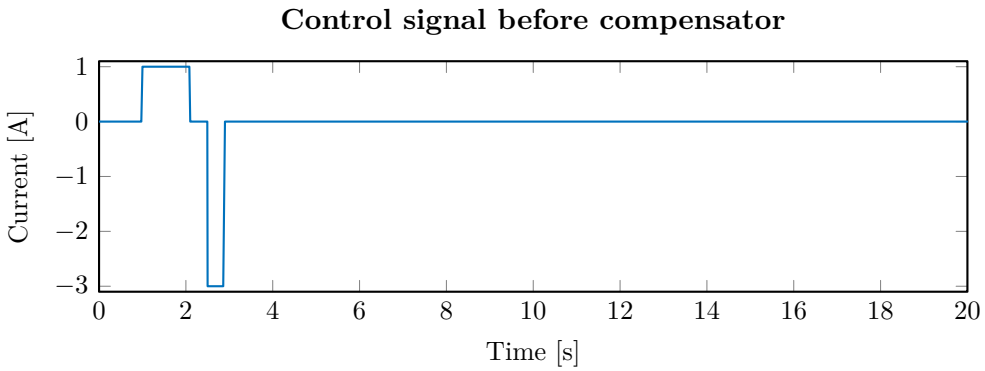


Figure 4.7. Step applied for state estimation simulation.

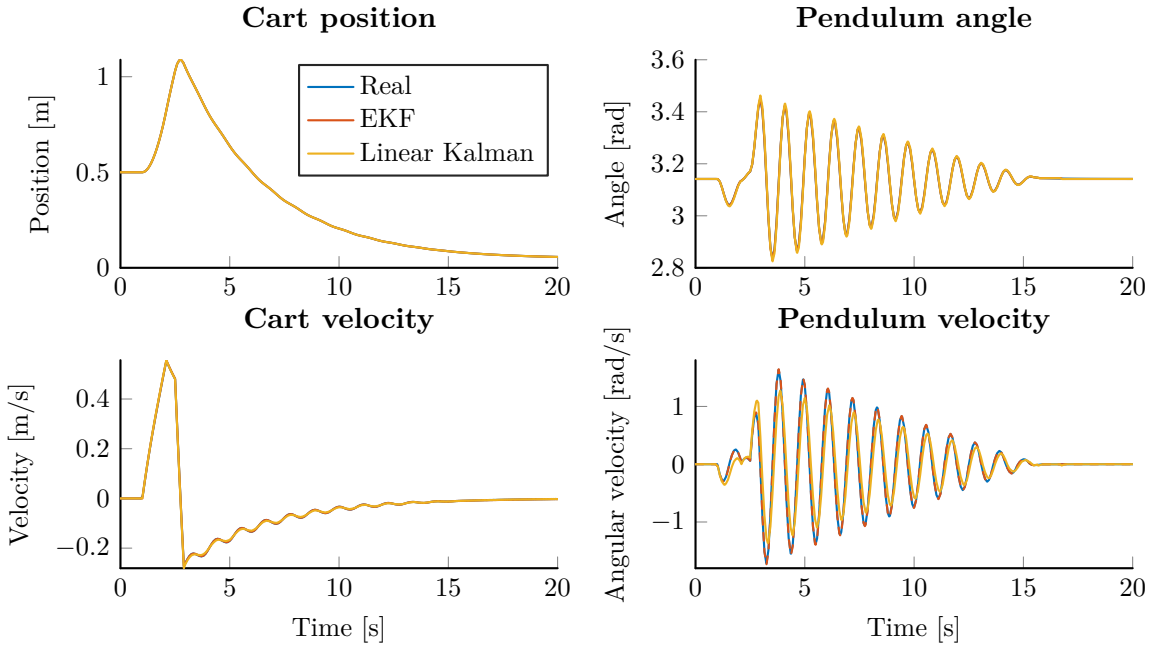


Figure 4.8. Simulation of both the linear and extended KFs, where the input can be seen in figure 4.7.

In order to further visualize the precision of, and the difference between, the two estimators, their residuals are plotted in figure 4.9.

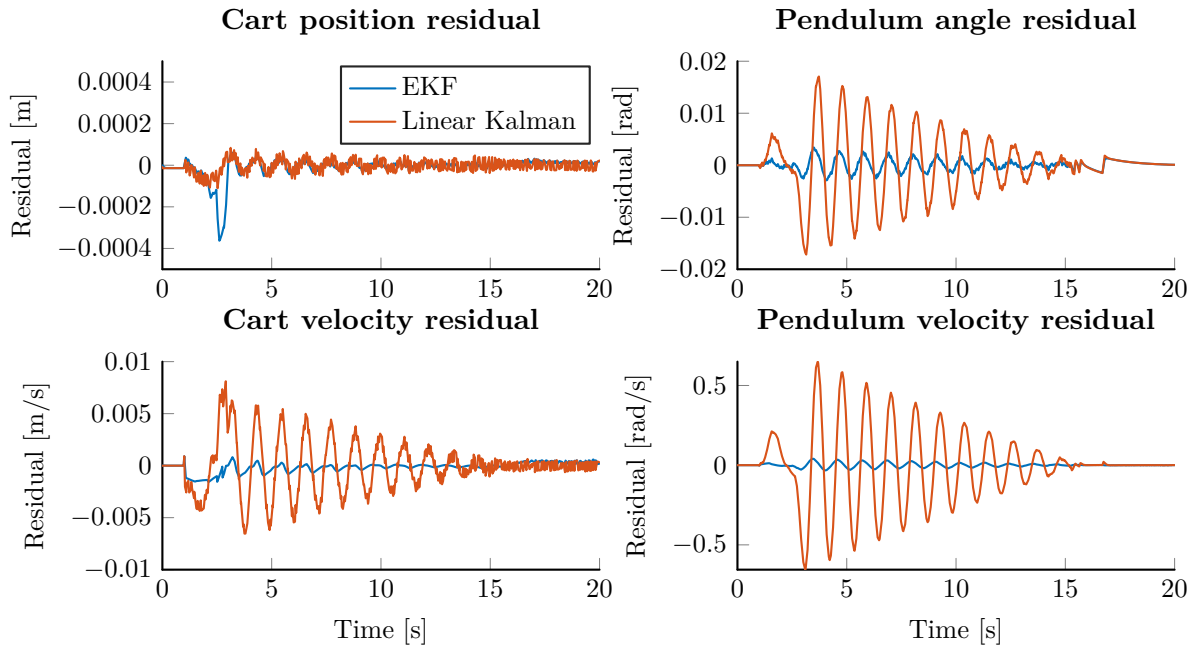


Figure 4.9. Residual of both the linear and extended KFs from the simulation in figure 4.8.

It can be seen, that the EKF shows a large improvement over the KF and fits almost exactly with the simulated states, without having the problem with a tradeoff between fitment of the angular velocity for either the actuated or the non-actuated time. Neither the estimate is delayed as the one from the KF. The EKF is thereby deemed appropriate as a state estimator for the non-linear control. The covariance matrices obtained from the tuning in the simulation are

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 25 \end{bmatrix}, \qquad R = \begin{bmatrix} 2 & 0 \\ 0 & 5 \end{bmatrix}. \tag{4.35}$$

These covariance matrices are substantially lower than for the KF, which is due to that the EKF includes the Coulomb friction in the pendulum and thereby can trust the model more, as well as an in general better fitting model.

The EKF filter is implemented in MATLAB as a function, which is called with the current measurement and the last control input as argument and returns the state estimate. The implementation is opposite order of the shown in equation 4.27 to 4.32, as the prediction step, i.e. the last two, is moved up as the first part, based on the previous state estimate and input. This is done for ease as the filter otherwise had to be split in two, in order to have the current control input, which is used for the prediction step. The error covariance matrix and last state estimate are saved as local variables in the function, to use at next run. From this function, C++ code is generated directly from the MATLAB code, by using the *MATLAB Coder* toolbox. The code is

generated as headers and C++ files, meaning it can be included as custom libraries in the MCU code and the EKF is executed as a function call like the MATLAB function. A test is shown in figure 4.11, where the system is excited with the control input seen in figure 4.10. In this test, the covariance matrices have been tuned in order to give a good estimate and filtering on the physical IPC setup.

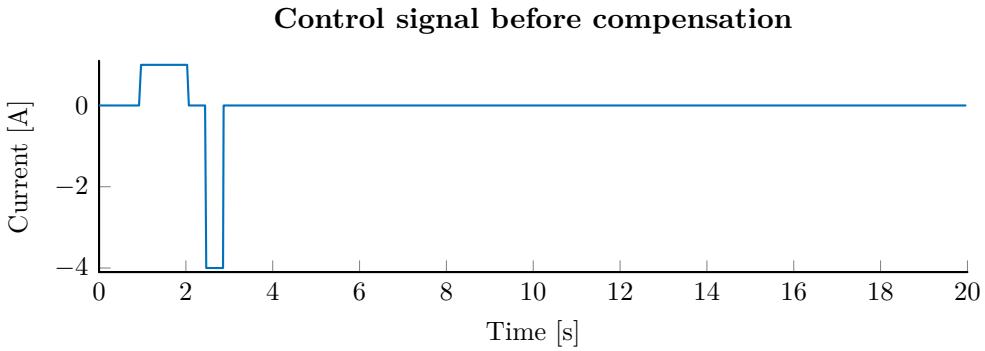


Figure 4.10. Step applied for state estimation test.

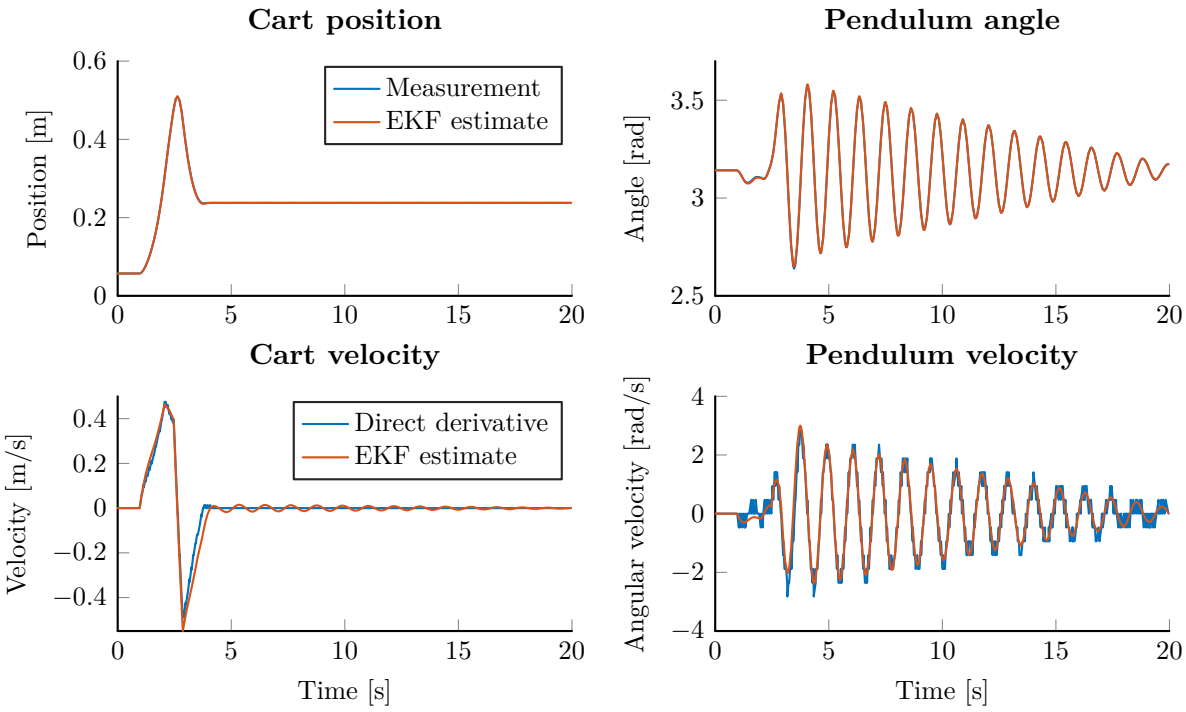


Figure 4.11. Test of the EKF, compared to the direct derivatives, where the input can be seen in figure 4.10.

In figure 4.11 it can be seen that the velocity estimations follows the direct estimations just without the quantization, of the direct derivative. It can however also be seen, that the cart velocity ends up oscillating a bit from ≈4 seconds, which the direct estimation does not. This is expected to be due to the design of the EKF did not include the Coulomb friction in the cart, and thereby

should move with the pendulum according to its model. This is seen as the pendulum is free swinging without actuation and with a quite large angle, and is thereby not deemed a problem under stabilization of the pendulum. The EKF is deemed a good estimator and the covariance matrices implemented are thereby

$$\begin{aligned} Q &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 50 \end{bmatrix}, \\ R &= \begin{bmatrix} 5 & 0 \\ 0 & 10 \end{bmatrix}. \end{aligned} \tag{4.36}$$

It can be seen, that the process variance in the velocity has been increased significantly, which complies with the fact, that the real system is not 100 % as modelled due to uncertainties and thereby more focus should be placed on the measurement to correct, than necessary in the simulation.

4.5 Conclusion

Throughout this chapter, the overall structure of the software running on the MCU is described, as well as a feedback compensation algorithm for the Coulomb friction in the cart is designed and tested. The compensation is deemed working as intended and thereby the Coulomb friction in the cart can be removed from the model for design state estimation and non-linear controller.

The state estimator has been designed and tested on the IPC and is deemed working as intended with satisfactory results. The final state estimator is an extended Kalman filter, as not satisfactory results were obtained with a linear Kalman filter, due to the non-linear Coulomb friction present in the pendulum.

5 Non-linear control of inverted pendulum on a cart

This chapter will describe the design of the non-linear controller for stabilizing the IPC. The non-linear control technique chosen for this thesis is the sliding mode control, SMC, due to its deemed good performance and especially robustness towards uncertainties, as described further through the chapter.

The ideal sliding mode controller differs from most other controllers, in that the sliding mode control law is not continuous in time, but varies between different continuous control laws depending on the state of the system. This is called a variable structure control method.

The concept of sliding mode control is shown in figure 5.1 and consists of two phases. First, the control forces the system onto the chosen sliding manifold, $s = 0$. This is called the reaching phase. The manifold is reached in finite time and by use of the variable structure control, the system is kept on the sliding manifold forever hereafter, which is called the sliding phase. As the system is confined to the manifold, it is constrained to move as a reduced order system defined by the manifold design. The manifold is designed such that the origin is an asymptotically stable equilibrium point. Hereby, the system slides along the manifold into the origin and the design of the manifold determines the reduced order dynamics of the system and thereby its behavior.

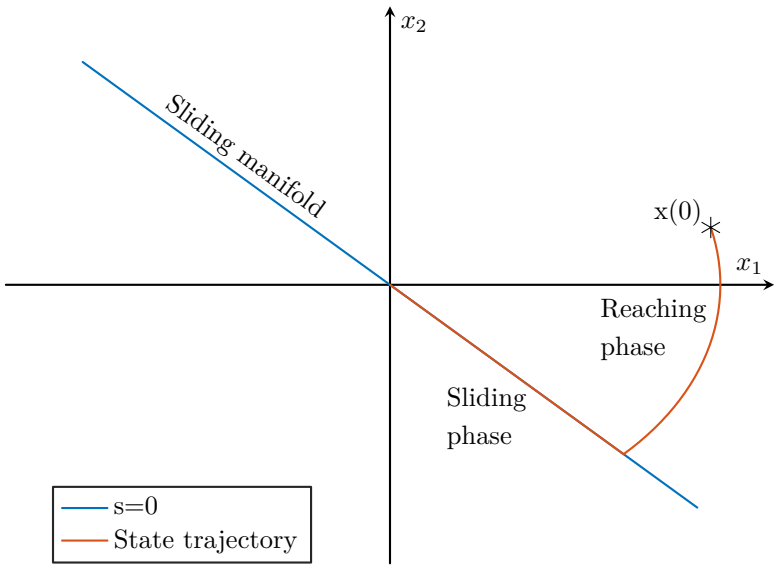


Figure 5.1. The concept of sliding mode control, here shown for a system with two states.

This chapter is based upon [26] (section 14.1) unless another source is stated.

5.1 Design of sliding mode controller

This section describes the design of the sliding mode controller. The design is based upon the obtained model for the IPC from equation 3.39 to 3.43 with the parameters presented in table 3.1 and 3.2, where the Coulomb friction in the cart has been removed from the model, due to the implemented compensation, as described in section 4.1. The model with the cart Coulomb friction removed, but still including Coulomb friction in the pendulum, is presented in equation 5.1.

$$\begin{aligned}\dot{x} = \begin{bmatrix}\dot{x}\\ \dot{\theta}\\ \ddot{x}\\ \ddot{\theta}\end{bmatrix} &= \begin{bmatrix} x_3 \\ x_4 \\ -\frac{B_{v,c}x_3+m_b l\sin(x_2)x_4^2}{m_c+m_b\sin^2(x_2)} - \frac{\cos(x_2)(B_{v,p}x_4+\mathcal{F}_{c,p}\tanh(k_{tanh}x_4)-m_b lg\sin(x_2))}{l\left(m_c+m_b\sin^2(x_2)\right)} \\ -\frac{\cos(x_2)\left(B_{v,c}x_3+m_b l\sin(x_2)x_4^2\right)}{l\left(m_c+m_b\sin^2(x_2)\right)} - \frac{(m_c+m_b)(B_{v,p}x_4+\mathcal{F}_{c,p}\tanh(k_{tanh}x_4)-m_b lg\sin(x_2))}{m_b l^2\left(m_c+m_b\sin^2(x_2)\right)} \end{bmatrix} \\ &+ \frac{K_t}{r}\begin{bmatrix} 0 \\ 0 \\ \left(m_c+m_b\sin^2(x_2)\right)^{-1} \\ cos(x_2)\left(l\left(m_c+m_b\sin^2(x_2)\right)\right)^{-1}\end{bmatrix} i_a \\ &= \begin{bmatrix} x_3 \\ x_4 \\ -f_{c1}(x)-f_{c2}(x) \\ f_p(x)\end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g_c(x) \\ g_p(x)\end{bmatrix} i_a = f(x)+g(x)i_a.\end{aligned} \tag{5.1}$$

The design of the SMC is based upon a system model in regular form

$$\dot{\eta} = f_a(\eta,\xi), \tag{5.2}$$

$$\dot{\xi} = f_b(\eta,\xi) + g_b(\eta,\xi)u, \tag{5.3}$$

with $u \in \mathbb{R}^p$, $\eta \in \mathbb{R}^{n-p}$ and $\xi \in \mathbb{R}^p$, and $g_b(\eta,\xi)$ is nonsingular. As $p = 1$ for the IPC, $g_b(\eta,\xi)$ needs to be 1×1, which is not the case in the form of equation 5.1. Due to this, the system needs to be transformed into another form, which is obtained, by use of a state transformation, $T(x) = \begin{bmatrix}\eta & \xi\end{bmatrix}^T$, such that

$$\begin{bmatrix}\dot{\eta}\\ \dot{\xi}\end{bmatrix} = \frac{\partial T(x)}{\partial x}f(x) + \frac{\partial T(x)}{\partial x}g(x)u. \tag{5.4}$$

As stated, the goal of the transformation is to obtain $\frac{\partial T(x)}{\partial x}g(x) = \begin{bmatrix}0 & 0 & 0 & g_\xi\end{bmatrix}^T$ such that $g_b(\eta,\xi) = g_\xi$, which is ensured by defining the last two columns of the Jacobian of the transformation as follows,

$$\frac{\partial T(x)}{\partial x} = \begin{bmatrix} T_{11} & T_{12} & 0 & 0 \\ T_{21} & T_{22} & 0 & 0 \\ T_{31} & T_{32} & \frac{\cos(x_2)}{l} & -1 \\ T_{41} & T_{42} & 0 & 1\end{bmatrix}. \tag{5.5}$$

The rest of the transformation matrix is chosen in order to obtain as simple transformed states as possible, but it still being a valid transformation, as later defined. This means T_{11} to T_{22} are chosen as identity and $\begin{bmatrix} T_{41} & T_{42} \end{bmatrix} = \begin{bmatrix} 0 & 0 \end{bmatrix}$ to keep the same states. The last two elements T_{31} and T_{32} are given from the transformed states, which on behalf of equation 5.5 are

$$\begin{bmatrix} \eta_1 \\ \eta_2 \\ \eta_3 \\ \xi \end{bmatrix} = T(x) = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \frac{\cos(x_2)}{l} - x_4 \\ x_4 \end{bmatrix}. \tag{5.6}$$

The last elements are thereby defined from differentiation and the final transformation matrix is

$$\frac{\partial T(x)}{\partial x} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -x_3 \frac{\sin(x_2)}{l} & \frac{\cos(x_2)}{l} & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{5.7}$$

The transformed system dynamics are thereby

$$\frac{T(x)}{\partial x} f(x) = \begin{bmatrix} x_3 \\ x_4 \\ -x_3 x_4 \frac{\sin(x_2)}{l} + \frac{f_{c2}}{l} \left(\frac{1 + \frac{m_c}{m_b} - \cos^2(x_2)}{\cos(x_2)} \right) \\ f_p \end{bmatrix} \tag{5.8}$$

and

$$g_\xi = g_p. \tag{5.9}$$

In order for the transformation to be valid, it needs to be a diffeomorphism for the domain of interest, D. For a transformation $z = T(x)$ to be a diffeomorphism, it needs to have an inverse map $x = T^{-1}(z)$ for all $z \in T(D)$. Both the transformation itself and its inverse transformation also needs to be continuously differentiable, as the derivatives of the state vectors, x and z should be continuous [26] (page 508). This can be checked with the inverse function theorem, which states, that if the Jacobian, J_t, of the transformation in a given point, a, is nonsingular, meaning $\text{rank}\left(J_t(a)\right) = \dim\left(J_t(a)\right)$, then the transformation fulfills the above mentioned requirements in a neighborhood of a and $T(a)$ [3] (page 372). As the rank of the Jacobian in equation 5.7 is only reduced in $x_2 = \frac{\pi}{2}$ and multiples of this, it can be concluded, that J_t is non-singular for $D = \left\{ x \in \mathbb{R}^4 : |x_2| < \frac{\pi}{2} \right\}$. As described in section 4.2, the controller is switched off due to safety before it exceeds the boundary of D and the transformation is thereby a diffeomorphism for the desired operation space including the origin as needed. The sliding mode controller can thereby be designed based on the transformed system, such that the origin is asymptotically stable.

As described in section 4.2, it is desired to introduce a reference position for the cart, x_{ref}, in order to control the cart to a specified position. This can be introduced into the transformation without

any problems, as the system can be stabilized regardless of the cart position without affecting the stabilization of the rest of the system. Because of this, the first state in the transformation is changed from x_1 to $x_1 - x_{ref}$ such that the state is the position error, which is okay as x_1 is not used in any of the other states. Thereby, the origin of the transformed system will in reality be $(x_{ref}, 0, 0, 0)$ as desired and the transformation is not changed on behalf of this. The final transformed coordinates are thereby

$$\begin{bmatrix} \eta_1 \\ \eta_2 \\ \eta_3 \\ \xi \end{bmatrix} = T(x) = \begin{bmatrix} x_1 - x_{ref} \\ x_2 \\ x_3 \frac{\cos(x_2)}{l} - x_4 \\ x_4 \end{bmatrix}. \tag{5.10}$$

The first step in the design of the SMC is to design the sliding manifold according to

$$s = \xi - \Phi(\eta) = 0 \tag{5.11}$$

where $\Phi(\eta)$ is a control function to be defined, such that when the system is on the manifold, $s = 0$, the reduced order system

$$\dot{\eta} = f_a(\eta, \xi) = f_a(\eta, \Phi(\eta)) \tag{5.12}$$

has the origin as an asymptotically stable equilibrium point. To do this, the system from equation 5.8 is written in terms of the transformed coordinates, from equation 5.10,

$$\begin{aligned} \dot{\eta} = f_a(\eta, \xi) &= \begin{bmatrix} x_3 \\ x_4 \\ -x_3 x_4 \frac{\sin(x_2)}{l} + \frac{f_{c2}}{l}\left(\frac{1+\frac{m_c}{m_b}-\cos^2(x_2)}{\cos(x_2)}\right) \end{bmatrix} \\ &= \begin{bmatrix} \frac{(\eta_3+\xi)l}{\cos(\eta_2)} \\ \xi \\ \frac{(\eta_3+\xi)\xi\sin(\eta_2)}{\cos(\eta_2)} + \frac{B_{v,p}\xi + \mathcal{F}_{c,p}\tanh(k_{tanh}\xi) - m_b l g \sin(\eta_2)}{l^2\left(m_c + m_b \sin^2(\eta_2)\right)}\left(1 + \frac{m_c}{m_b} - \cos^2(\eta_2)\right) \end{bmatrix}, \end{aligned} \tag{5.13}$$

utilizing from equation 5.10

$$\eta_3 = x_3 \frac{\cos(x_2)}{l} - x_4 \Rightarrow x_3 = \frac{(\eta_3 + x_4)\, l}{\cos(x_2)} = \frac{(\eta_3 + \xi)\, l}{\cos(\eta_2)}. \tag{5.14}$$

The stabilizing controller, $\Phi(\eta)$, for this system thereby needs to be designed. In order to get a starting point, a linear state feedback is designed,

$$\Phi(\eta) = -k\eta, \tag{5.15}$$

based upon a linearization of the plant. The nonlinear plant in equation 5.13 is therefore linearized into a standard linear system

$$\dot{\eta} = A\eta + B\xi, \tag{5.16}$$

by use of a first order Taylor approximation at the origin, using the same procedure as for the Kalman filter in appendix D.1. The linearized model matrices are

$$A = \frac{\partial f_a(\eta,\xi)}{\partial \eta}\Big|_{\eta=\xi=0} = \begin{bmatrix} 0 & 0 & l \\ 0 & 0 & 0 \\ 0 & -\frac{g}{l} & 0 \end{bmatrix}, \tag{5.17}$$

$$B = \frac{\partial f_a(\eta,\xi)}{\partial \xi}\Big|_{\eta=\xi=0} = \begin{bmatrix} l \\ 1 \\ \frac{B_{v.p,lin}}{l^2 m_b} \end{bmatrix}. \tag{5.18}$$

Due to the same problems with the pendulum Coulomb friction as for the linearization of the Kalman filter, the Coulomb friction is discarded in this linearization and the adjusted viscous friction from the Kalman filter in equation 4.22 is utilized instead.

The system is controllable, and the feedback controller is designed by use of the MATLAB function *place* where the closed loop poles are placed at $\begin{bmatrix} -3 & -4 & -6 \end{bmatrix}^T$. This pole placement is chosen, as it gives a sufficiently fast response for the system, as a simulation from initial values $\eta(0) = \begin{bmatrix} 0,1 & 0,1 & 0 \end{bmatrix}^T$ yields a settling time of approximately 1,1 s for the pendulum angle and 1,3 s for the cart position, as seen in figure 5.2. The obtained gains are

$$k = \begin{bmatrix} -7,34 & 19,08 & -1,93 \end{bmatrix}. \tag{5.19}$$

The designed feedback controller is tested on the non-linear system, including Coulomb friction in the pendulum, from equation 5.13, which is seen in figure 5.2 and shows good results. It can be seen that the non-linear system does not settle at the origin as for the linear model, but keeps oscillating slightly around it. This is due to the Coulomb friction in the pendulum, as very small movements are needed in the end to reach pendulum angle zero, which is not possible with the Coulomb friction present.

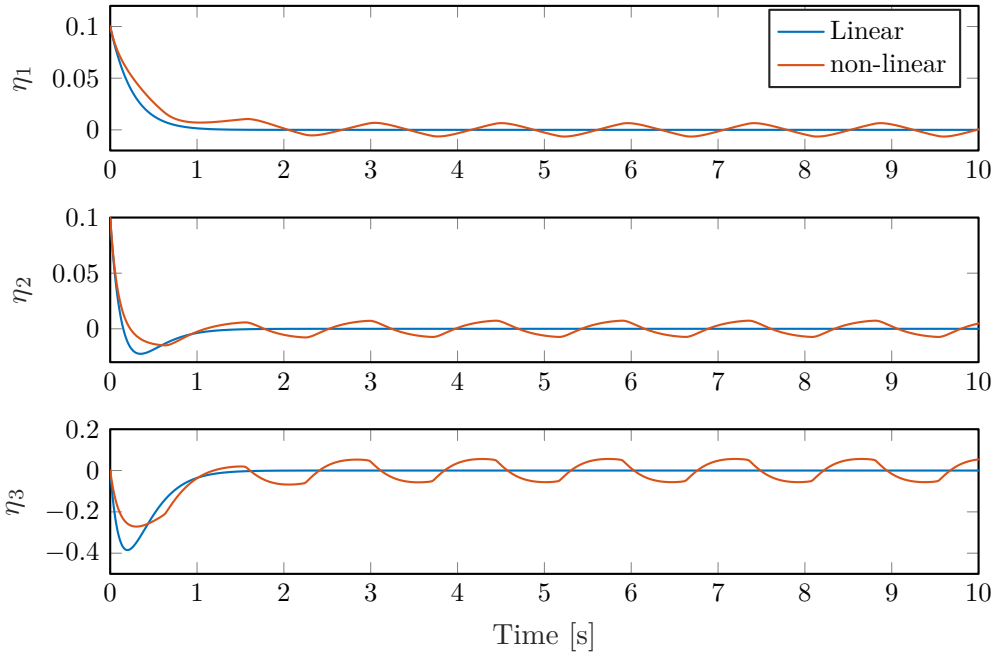


Figure 5.2. Linear and non-linear simulation of reduced order system with the designed linear feedback.

The sliding manifold from equation 5.11 is thereby by equation 5.15

$$s = \xi - \Phi(\eta) = \xi + k\eta = \xi + k_1\eta_1 + k_2\eta_2 + k_3\eta_3, \tag{5.20}$$

which in the original coordinates corresponds to

$$s = x_4 + k_1\left(x_1 - x_{ref}\right) + k_2x_2 + k_3\left(\frac{x_3\cos(x_2)}{l} - x_4\right). \tag{5.21}$$

The control law is designed based on the derivative of the manifold, such that the vector field is always pointing towards the manifold, in order to make $s = 0$ attractive and maintain the state there. Based on equation 5.20,

$$\begin{aligned}\dot{s} &= \dot{\xi} + k_1\dot{\eta}_1 + k_2\dot{\eta}_2 + k_3\dot{\eta}_3 \\ &= f_p + g_p i_a + k_1x_3 + k_2x_4 + k_3\left(-x_3x_4\frac{\sin(x_2)}{l} + \frac{f_{c2}}{l}\left(\frac{1 + \frac{m_c}{m_b} - \cos^2(x_2)}{\cos(x_2)}\right)\right) \\ &= \Omega(x) + g_p(x)i_a,\end{aligned} \tag{5.22}$$

by inserting the transformed system described in the original coordinates. In order to make $s = 0$ attractive and maintain it there, it is required that

$$\dot{s} \begin{cases} < 0 & \text{if } s > 0 \\ = 0 & \text{if } s = 0 \\ > 0 & \text{if } s < 0 \end{cases}, \tag{5.23}$$

which can be ensured by defining the control law as

$$i_a = -g_p^{-1}(x)\beta(x)\text{sign}(s) \tag{5.24}$$

as long as

$$0 < |\Omega(x)| < \beta(x) \tag{5.25}$$

with sign$(\cdot)$ being the signum function as defined in equation 3.36. Thereby $\beta(x)$ needs to overcome the system itself defined by $\Omega(x)$, but the condition also needs to be ensured for all parameter uncertainties and disturbances in the system, in order to ensure convergence to the manifold and uphold it there. The larger the $\beta(x)$, the larger uncertainties and disturbances can be handled at the cost of larger actuation. Thereby, the amount of uncertainties and disturbances, which can be handled, are limited by the desired actuation level and only an upper bound is needed to ensure this. Two types of different approaches for designing $\beta(x)$ are tried in this thesis, as further described in section 5.2.

The manifold is proven to be an asymptotically equilibrium by Lyapunov stability theory. The Lyapunov stability theorem states, that the origin is an asymptotically stable equilibrium point for a system, if there exists a function $V(x)$, such that the following hold

$$V(0) = 0, \tag{5.26}$$
$$V(x) \text{ is positive definite}, \tag{5.27}$$
$$\dot{V}(x) \text{ is negative definite}, \tag{5.28}$$

for the whole domain of interest [26] (page 114). A Lyapunov function for the system in equation 5.22 is $V(s) = \frac{1}{2}s^2$, which clearly upholds equation 5.26 and 5.27 and by

$$\begin{aligned} \dot{V}(s) = s\dot{s} &= s\left(\Omega(x) + g_p(x)i_a\right) \\ &= s\left(\Omega(x) - g_p(x)g_p^{-1}(x)\beta(x)\text{sign}(s)\right) \\ &= s\left(\Omega(x) - \beta(x)\text{sign}(s)\right) = \begin{cases} < 0 & \text{if } s \neq 0 \\ 0 & \text{if } s = 0 \end{cases}, \end{aligned} \tag{5.29}$$

condition 5.28 as well, as long as $0 < |\Omega(x)| < \beta(x)$. Thereby $s = 0$ is an asymptotically stable equilibrium and when reached, the system dynamics are constrained to the reduced order dynamics and asymptotically approaches the origin. As the system on the manifold is constrained

to a reduced order system independent of $f_b(\eta, \xi)$ and $g_b(\eta, \xi)$ in equation 5.3, it is also independent from uncertainties in these terms, which makes this control technique very robust. Furthermore, only an upper bound on the uncertainties are needed for the reaching phase, in order to ensure this. Uncertainties in these two terms are defined as matched uncertainties, and uncertainties affecting the reduced order system are called unmatched uncertainties. The SMC is not robust against unmatched uncertainties by itself, however the feedback defining the manifold can be designed in order to obtain robust behavior if needed.

The theoretical SMC as previously described, is based upon the signum function, in order to reach the sliding manifold and slide directly on the manifold hereafter. This works on perfect modelled system dynamics and in continuous time. However, this is not possible to obtain in reality, due to e.g. unmodelled high frequency actuator dynamics and discrete controller implementation. If the signum function is utilized, the system trajectory will pass through the manifold, before the controller changes control input due to the discrete implementation or the physical actuation dynamics, causing the sign of the manifold to change and thereby the manifold to be reached from the other side and so on. This phenomenon is called chattering and is shown in figure 5.3.

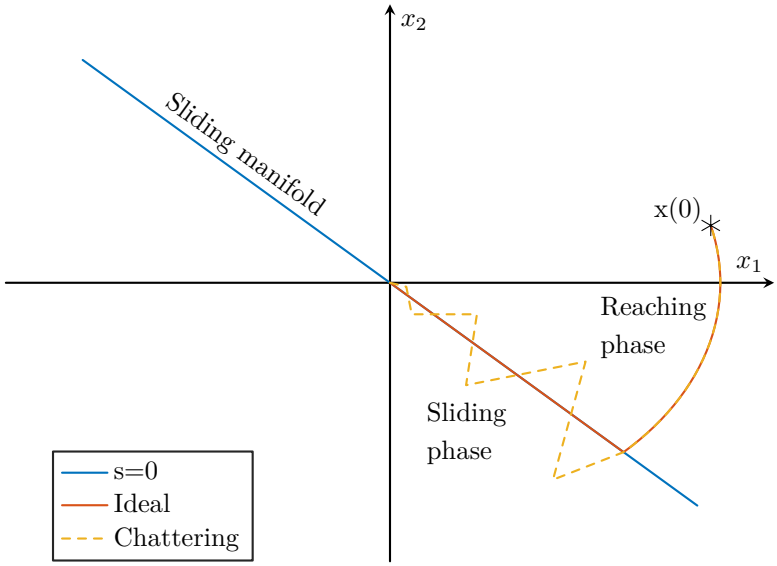


Figure 5.3. The concept of chattering in sliding mode control, here shown for a system with two states.

The simplest solution to this problem is to change the signum function to a relaxed continuous approximation in terms of the saturation function, where the slope in the saturation function is defined by ϵ,

$$\text{sgn}(s) \rightarrow \text{sat}\left(\frac{s}{\epsilon}\right). \tag{5.30}$$

As long as the function does not saturate, this acts as a P-controller in s, to keep the system on the manifold and thereby the hard switching behavior causing chattering can be reduced. The larger the ϵ, the less chattering. However, this introduces an area around the manifold called the

boundary layer, which the system will reach and be kept inside, but the system will not go to $s = 0$ and stay there as in the perfect case. As epsilon increases, so does the boundary layer, and the continuous approximation gets further away from the previously described theory and thereby also loses some of its robustness. As epsilon approaches zero, the continuous approximation converges to the discontinuous one. Thus, it is desired to keep ϵ as small as possible.

5.2 Tuning and results

As described in the previous section, two SMCs are designed, based on the same manifold, but with two different expressions for $\beta(x)$. The first controller utilizes a constant beta value and is first hand-tuned to obtain sufficient performance in simulation, yielding the control parameters

$$\beta(x) = \beta_0 = 7, \tag{5.31}$$

$$\epsilon = 0,04, \tag{5.32}$$

based on the designed manifold gains from equation 5.19. A simulation from initial values $\begin{bmatrix}\eta(0) & \xi(0)\end{bmatrix}^T = \begin{bmatrix}0,3 & 0,1 & 0 & 0\end{bmatrix}^T$ is seen in figure 5.4. The simulation is performed with encoder quantification and through the EKF implemented in the simulation in order to obtain a realistic impression of the expected performance.

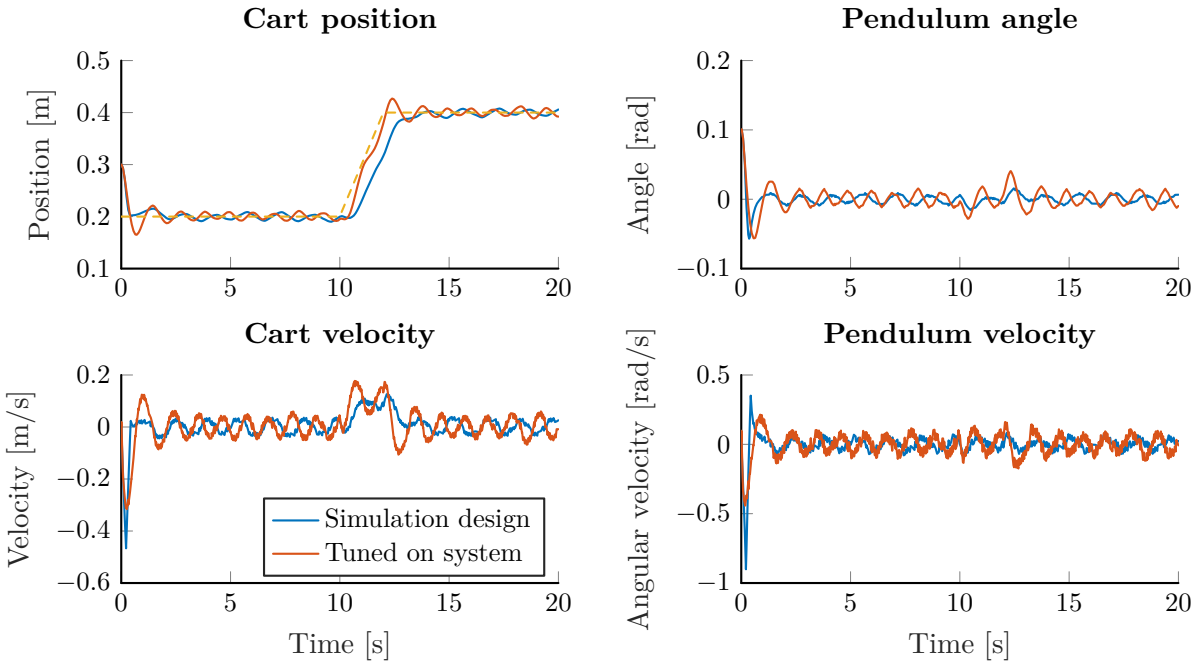


Figure 5.4. Simulation of both the first designed controller in simulation and the controller with tuned parameters for the physical system. Dashed yellow line is the reference.

It should be mentioned that the designed SMC is not designed for tracking of changing references, as this is not considered in this thesis. If this was intended, the derivative of the reference, as well as the differentiated error, should be included in the design e.g. by use of higher order SMC. Thereby changes in the reference in this thesis, act as a disturbance and too large reference

changes force the system off the manifold. Because of this, changes in the reference are made with a slope if 10 cm/s in order to minimize the disturbance and keep the system near the manifold.

Implementation of the controller on the IPC system showed tracking problems for the cart reference, even though the system for most of the time keeps inside the boundary layer. The system thereby drifts on the manifold, as the manifold can be kept close to zero without the cart position error being close to zero. Because of this, the k_1 gain for the cart position error in the manifold is increased, in order to put more weight on the cart position error. The manifold as well as β_0 and ϵ are tuned to ensure good performance on the real IPC system when stabilizing and following the reference, however also for physical disturbances introduced by pushing cart or the pendulum. This results in the following adjusted controller parameters, which represent a tradeoff between actuation magnitude and disturbance rejection:

$$k = \begin{bmatrix} -30 & 29 & -2,2 \end{bmatrix}, \tag{5.33}$$

$$\beta(x) = \beta_0 = 10, \tag{5.34}$$

$$\epsilon = 0,035. \tag{5.35}$$

A simulation based on these control parameters is seen in figure 5.4 in order to compare the initially designed controller with the hand-tuned for the physical system. Here it can be seen, that the latter in general is more aggressive, as expected since both β and the manifold has increased gains. The tuned controller also yields a less damped system which is especially seen in the beginning and after the reference change when the system has to settle. However, a great improvement in the reference following is seen in the reference change, as the position error has increased gain.

In figure 5.5 the performance of the tuned SMC can be seen on the physical IPC setup, which shows satisfactory performance, as well as a controller based on varying β value, which is designed later in this section. It should be noted, that all graphs showing measurements on the physical setup, has been cut in the beginning, due to slightly different initial positions, as well as disturbances introduced when supporting the pendulum by hand as the controller starts. Thereby the settling time for this is cut off, to create more comparable graphs.

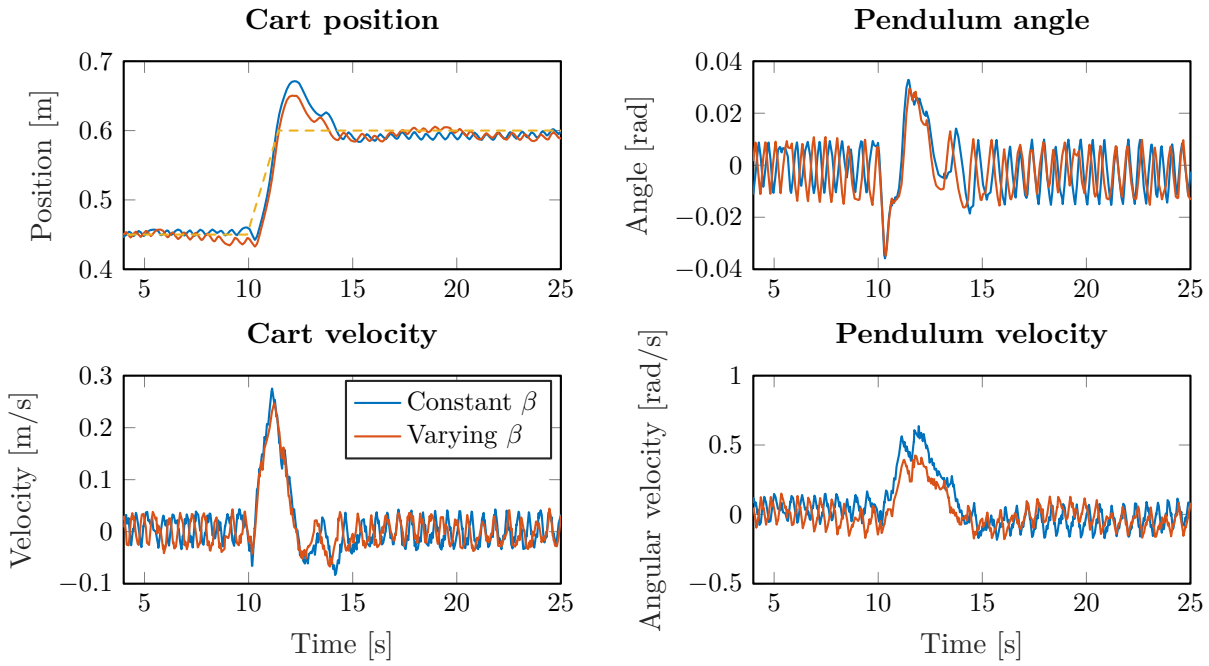


Figure 5.5. Tests on the real IPC of the two designed controllers. Dashed yellow line is the reference.

As described, the friction changes along the rail, which can easily be seen on the physical setup, as a steady state error is present in the cart position, which is varying both in sign and magnitude depending on the position on the rail. The general best performance has been found, by lowering the Coulomb friction compensation, such that the compensation is equal in both directions, and the lowest of the two frictions are used in both directions. The cart position used in figure 5.5 is one yielding a small steady state error, however other cart positions causes larger steady state errors. If the steady state error were to be eliminated, an integral term needs to be implemented into the SMC.

Disturbances in the form of pushing the pendulum and cart can be introduced into the system, which is handled as intended, as long as the system is not pushed off the manifold. The system is pushed off the manifold when equation 5.25 is not satisfied due to the disturbance. In order to be able to handle larger disturbances, $\beta(x)$ thereby needs to be bigger. If only a β_0 is used as in the previously described controller, this would mean larger actuation when the pendulum is balancing undisturbed, which is not desired. Therefore, a second controller is designed, where the β is dependent on the states, such that the actuation increases when the pendulum is not close to its equilibrium point in origin. The tuned controller is

$$\beta(x) = \frac{|\Omega(x)|}{3} + 9 \tag{5.36}$$

$$\epsilon = 0,055 \tag{5.37}$$

with the same manifold gain from equation 5.33. When the pendulum is balancing undisturbed,

$|\Omega(x)| \leq 3$ which means $\beta(x)$ in this case is approximately the same as for the previous controller. However, when large disturbances are introduced into the system, $|\Omega(x)|$ can approach 30, from where $\beta(x) \approx 20$ and thereby much more aggressive actuation is used, and larger disturbances can be handled. Due to the stronger actuation, it has been necessary to increase ϵ to avoid too intense chattering under large disturbances. This gives a decrease in actuation when close to the origin, but still acceptable performance. Comparison of the two controllers' actuation during the test shown in figure 5.5 can be seen in figure 5.6. Here it can be seen that the controller with varying β uses significant smaller actuation when close to origin, as well as being capable of handling larger disturbances, on behalf of a bit worse reference tracking, which can be seen in figure 5.5.

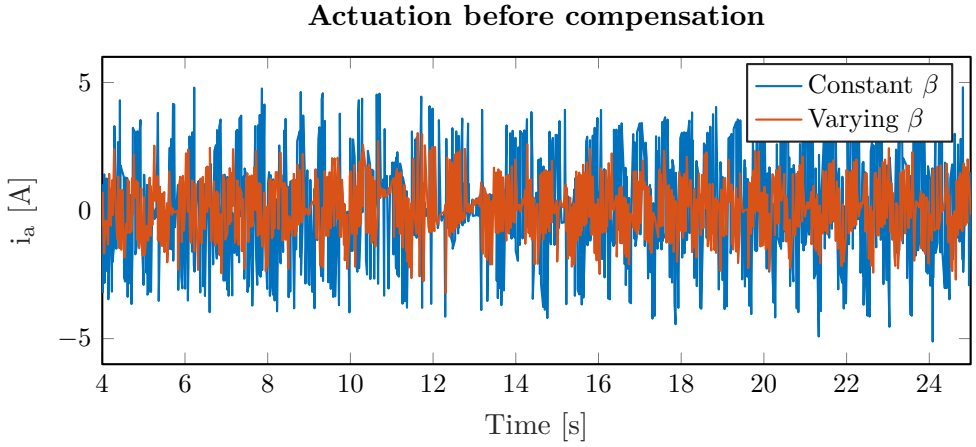


Figure 5.6. Actuation during the test in figure 5.5 of the two designed controllers.

Hereby two SMCs have been designed. One capable of handling larger disturbances due to its varying $\beta(x)$, at the cost of bigger actuation and a bit of chattering under large disturbances, compared to the one with constant β value.

No graph of the mentioned disturbances is shown, as the disturbances are introduced into the system by hand, and thereby two disturbances are not comparable. The controllers are deemed to have good disturbance rejection capabilities, the latter a bit better. In order to test their robustness towards parameter uncertainties, two other tests are performed for both controllers, where the bob weight is changed. This parameter affects both the input system and the reduced order system. The controller should be robust against the matched part of the uncertainty, however the SMC is not robust against the unmatched part of the uncertainty. The bob weight is changed ±50 g from the nominal weight of 201 g for both of the designed controllers, and the results can be seen in figure 5.7 and 5.8.

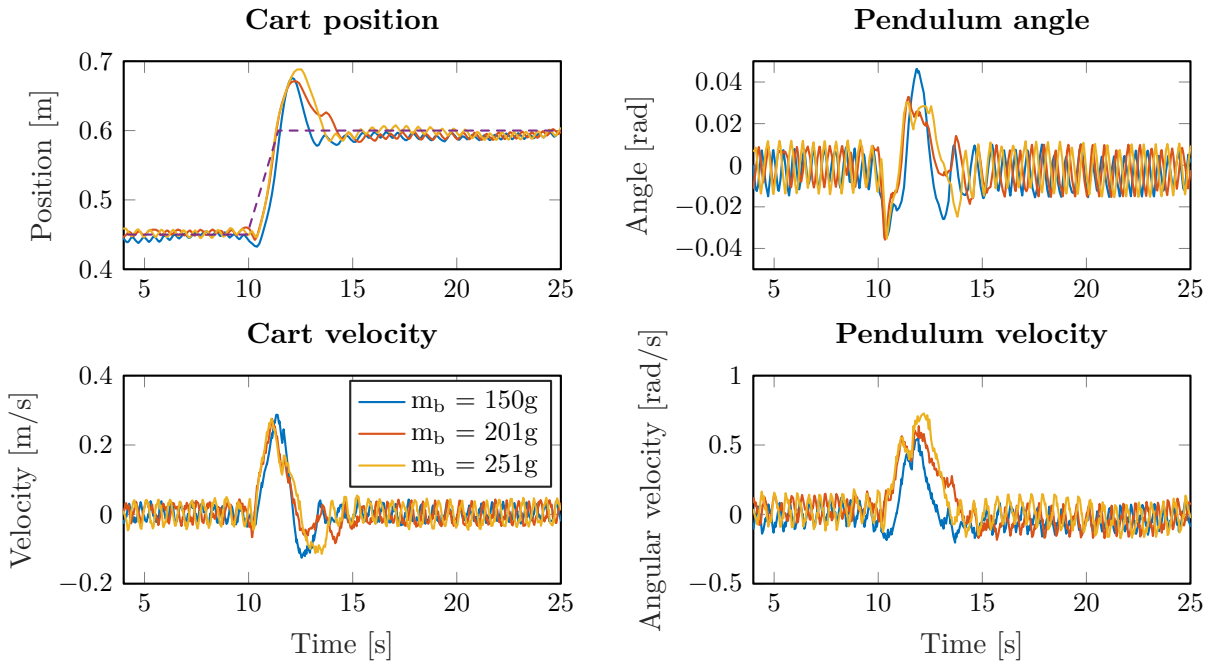


Figure 5.7. Robustness test against changing bob weight for the controller with constant β.

It can be seen in figure 5.7, that the ±25 % change in bob weight has no significant impact on the performance on the controller with constant β value, which indicates robustness against this parameter uncertainty.

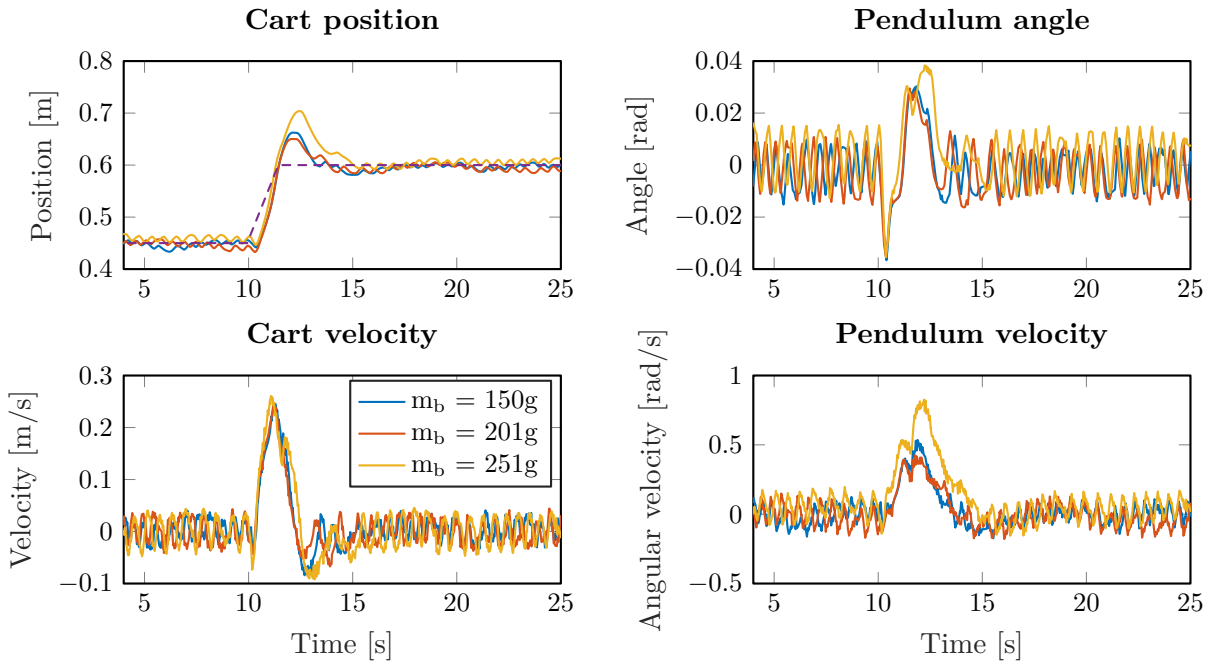


Figure 5.8. Robustness test against changing bob weight for the controller with varying β.

Figure 5.8 also shows evidence of good robustness for the controller with varying β value and stabilizes the pendulum with all 3 weights. A small decrease in performance can be seen for the increased bob weight. This is due to the smaller actuation compared to the other controller, from where the increased weight increases the settling time for the system. That is however not seen as a problem.

Another heuristic test has be made, where extra friction is added to the cart, by pressing a finger against the pulley at the end of the rail, which the controller is capable of handling. This is expected, as this is a matched uncertainty. This is however undocumented in the report, due to it not being reproducible such that the tests are comparable.

5.3 Conclusion

In this chapter, two sliding mode controllers, SMCs, have been designed, both showing good performance in terms of stabilizing the pendulum under the varying parameters in terms of the varying friction along the rail. The first SMC is designed with constant β value, which means slightly more actuation when close to origin, compared to the second controller which has state dependent β value. Due to this, the first controller has slightly better cart position reference tracking, however the second controller can handle larger disturbances. In general, both controllers have problems with steady state errors in the cart position, which also varies depending on the reference, due to the varying friction along the rail. Both controllers have been demonstrated to be robust against parameter uncertainties, e.g. ± 25 % bob weight and is thereby deemed successful.

6 Machine learning control of inverted pendulum on a cart

Throughout this chapter, it is investigated how machine learning can be used for controller design. This is chosen in this thesis, in order to get experience with more modern controller design opportunities, instead of the classic model-based controller design. Machine learning in general is defined as a computer "learning", meaning increasing its performance, by use of data and/or interactions with the environment. Machine learning in general, but also in a control context is of increasing interest in recent years, due to the increasing computer performance and big data.

Through this chapter, a quick overview of machine learning control, MLC, is given, from where reinforcement learning, RL, is chosen for this thesis. First, the RL algorithms are described and design-trials in simulation are made. Partly successful designs have been achieved in simulation, however it does not work on the real IPC setup. The chapter describes the theory and the foundation for the tried methods, as well as what is tried, even though no successful controller has been obtained.

6.1 MLC overview

MLC is a very big topic and a vast amount of methods and papers are available and still rapidly expanding, utilizing different approaches and expanding the theoretical foundation for obtaining controllers. MLC is a data-driven regression problem to find a control law i.e. a mapping from sensor signals to actuation commands which optimizes a goal function. As no knowledge of MLC, nor ML in general, is present in the group, a brief overview of some found MLC methods and concepts are described through this section. This will serve as the foundation for the choice of MLC approach.

6.1.1 Genetic programming

Genetic programming, GP, is a model-free MLC approach based upon genetic algorithms and has successfully been applied to especially hard to predict systems such as turbulent flows [19] (chapter 5 and 6), where the obtained non-linear controller is capable of outperforming the normal linear controllers, due to the very non-linear nature of the process. It has also successfully been applied to a physical inverted pendulum on a cart problem [15].

Genetic programming is an evolutionary algorithm, which is inspired by natural selection. A population of individuals evolves over generations in order to find the best individual. A generation consists of individuals, where each individual is a controller with a specific structure and parameters. The individuals are constructed as an expression tree structure, where the root is the control output, and the leaves are the sensor inputs or constants. Every branch is a mathematical operation chosen from a pool of allowed mathematical operations. An example of an individual represented as an expression tree structure can be seen in figure 6.1. The subscript $_{1,2}$ indicates

it is individual one in generation two. The controller equals the control law

$$u_{1,2} = \cos(s_1) + \tanh(0,4s_1) - 10,2s_2. \tag{6.1}$$

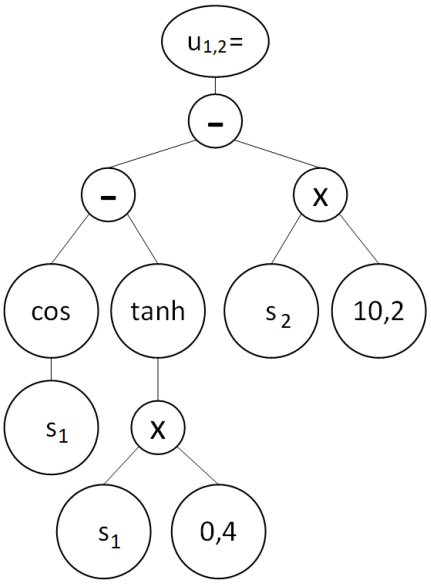


Figure 6.1. An expression tree corresponding to the control law in equation 6.1, which is an example of individual one in generation two.

The initial generation is generated randomly, and each controller is evaluated on the system and their performance is determined according to a defined cost function which is wanted to be minimized. The lower the evaluated cost of the individual, the higher its chance to advance to the next generation. A set of rules determines who advances and how the next generation is created, where this is the basis rules:

- **Elitism**: The N_e individuals with the lowest cost are copied directly to the next generation.
- **Replication**: All leftover individuals are assigned a probability of advancing to the next generation, based on their evaluated cost. The lower the cost, the higher the probability.
- **Mutation**: Four different mutations normally exist. The first is *cut and grow* where a randomly chosen subtree in an individual is replaced with another randomly generated subtree. *Shrink* replaces a randomly chosen subtree with a randomly chosen leaf (sensor or constant). *Hoist* replaces the whole tree with one of its own subtrees. *Reparametrization* replaces approximately 50 % of the constants by a randomly chosen new constant.
- **Crossover**: Two selected individuals exchange one of their randomly chosen subtrees with each other.

Elitism ensures the best individuals always stay in the population and replication stabilizes the convergence process, as it guarantees that a part of the population stays in the vicinity of the

found local minima, such that the area can be further exploited. Mutation and crossover ensure exploration and exploitation respectively [19].

In this way, the controller is optimized in both control parameters and structure over generations. The GP is tuned e.g. by means of the number of individuals in each generation, number of generations, individual's probability for advancing to next generation using the different previously described advancement methods, tree depths and leaf density. Even though GP is not guaranteed to converge to the global minima, GP has been successful in multiple applications.

6.1.2 Artificial neural networks

Artificial neural networks, ANNs, often just referred to as neural networks, NNs, are used in a lot of varieties in ML in general as well as MLC concepts. Due to this, the brief concept of a NN is explained in the following, before some of the MLC approaches based on NNs are described. The simplest NN is a feedforward NN and is thereby the foundation for the following explanation.

An artificial neural network can be used to approximate vector-valued functions and is based upon the biological neural network that constitutes animal brains [31] (chapter 1). An example of a feedforward ANN can be seen in figure 6.2. As the name implies, the data only travels forward through the network. A network consists of an input layer with one input node per input of the function to approximate. The input nodes do no computations on the inputs, but distributes the input data to the next layer in the network. The last layer in the network is the output layer, consisting of the number of neurons, corresponding to the number of outputs of the function to approximate. The network can be expanded by hidden layers consisting of neurons. The number of hidden layers and the number of neurons in each hidden layer are design parameters, depending on the target function to approximate. The network is named by the layers of neurons, and the one shown in figure 6.2 is thereby a two-layer NN. In general networks with more than one hidden layer are referred to as deep NNs.

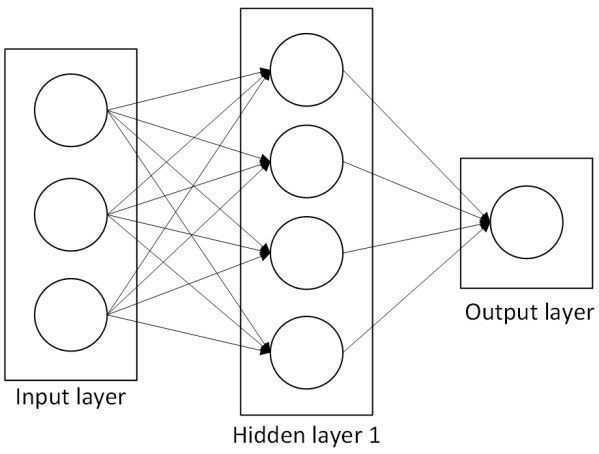


Figure 6.2. The concept of a two-layer feedforward artificial neural network.

The output of a neuron, $y(t)$, is calculated based on its n inputs, $\begin{bmatrix} x_1(t) & x_2(t) & \cdots & x_n(t) \end{bmatrix}^T$ linearly weighted by the n weights $\begin{bmatrix} w_1(t) & w_2(t) & \cdots & w_n(t) \end{bmatrix}^T$ and through a non-linear function,

$\sigma(\cdot)$, called the activation function. The output, $y(t)$, is thereby evaluated by

$$y(t) = \sigma\left(w(t)^T x(t) + w_0(t)\right), \tag{6.2}$$

where $w_0(t)$ is the bias of the neuron. The activation function is typically a sigmoid function

$$\sigma(z) = \frac{1}{1+e^{-z}}, \tag{6.3}$$

but can also be linear or other functions suitable for the approximation. Thereby the output of the neuron with a sigmoid activation function lies in $\sigma(z) \in \{0; 1\}$ for $z \in \{-\infty; \infty\}$. By utilizing this, a NN can by the universal approximation theorem approximate any smooth function, $f(x)$, mapping from $\mathbb{R}^n \to \mathbb{R}^m$ on a compact set $S \in \mathbb{R}^n$ with a sufficiently small approximation error, for some sufficiently large number of neurons in the hidden layer [23]. This is possible utilizing only one hidden layer, however, determining the weights to obtain this is in general difficult. Training of feedforward NNs is often done by the backpropagation algorithm, which uses gradient descent to adjust the weights such that the error between the NN output and the wanted output is minimized. This is done by backpropagating the output error into the layers, such that the respective weight can be adjusted. This function approximation property of NNs has made them very popular in many applications, including MLC. Some of the use cases of NNs for MLC are described in the following subsections.

NN to approximate process model

The nature of a NN is supervised learning, meaning it approximates an unknown vector-valued function, which for control purposes can be the input/output relation of the plant to control. This can be utilized to set up plant models based on collected input/output data constituting e.g. an advanced simulation model. Based on this simulation model, controllers can be verified and/or designed on a complex model. This is however not wanted for this thesis, as it is desired to try to design the controller directly based on MLC.

In order to utilize this model approximation capability for a controller, multiple methods are used to approximate either the full or a part of the process model. Utilizing a NN for this online makes the controller adaptive, as the NN adjust its weight as parameters in the system changes. One way of utilizing NN for control is by estimating the whole model and using the inverse of the model as an inner loop in order to cancel the plant dynamics. In this way, the NN works as a feedback linearization. An outer loop is then in charge of the reference tracking, but assisted by the adaptive inner loop [30]. In a quite similar fashion, inverse NN models of the plant can be used as additive feedforward from the reference in parallel with a normal controller, in order to increase the performance of the control system and/or allow for simpler control design [16]. NNs can also be used for predictive control, such that the NN when describing the physical plant well enough, can be used to predict the response of the plant and thereby can be used in the optimization process for determining the best control [51]. However predictive control is in general not well suited for fast systems as the IPC, due to the computational effort at each sample time.

Self-tuning PID-like controller based on a NN

In this approach, a self-tuning PID-like controller is designed, based on a NN [13]. In order to obtain the integral and derivative functionality of the PID controller, a recurrent NN is utilized. In contrast to a feedforward NN, a recurrent NN has data looping backward internally in the NN, by use of a unit delay. The number of input nodes corresponds to the number of tracking errors in the system and the output consists of a single neuron whose output is the control signal. The concept can be seen in figure 6.3, where all activation functions, f, are linear.

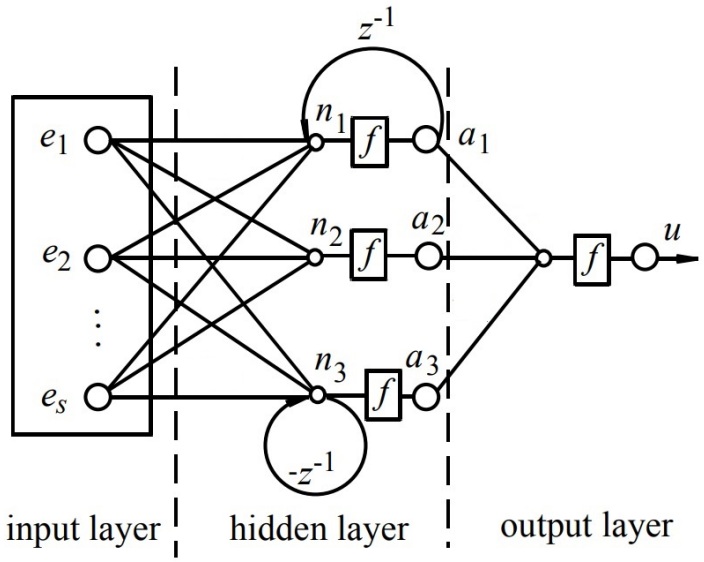


Figure 6.3. The concept a PID-like NN controller [13].

The network consists of three neurons in the hidden layer. The output of the first neuron, a_1, corresponds to the integral mode, which is achieved by feeding back a delayed version of the output to the weighted sum of the input. The second neuron corresponds to the proportional part and the third neuron is the derivative-like part, obtained by subtracting a delayed version of the last linear combination.

The network is then trained based on gradient descent for minimizing the tracking error and have been showing good results in comparison with LQR for a simulated double inverted pendulum on a cart [13].

NN to approximate non-linear optimal control

In [7] it is shown how a NN can be used to approximate the solution to the nonlinear optimal controller, minimizing the standard quadratic cost function, as a non-linear extension of the LQR. The NN weight-updating algorithm is derived based on calculus of variations. For the simulated double inverse pendulum on a cart, the NN in combination with a normal LQR controller shows significantly better performance than pure LQR control, when the system deviated from the linearization point. However, in order to obtain the weights, the Jacobians of the plant model are needed, which means a model both for the updating of the NN weights, as well as for the design of the LQR controller is needed. It is however mentioned, that an extra NN called an ”emulator”

can be used to approximate the plant behavior and thereby can be used to approximate the Jacobians. In [1] it is similarly shown, how a NN can be utilized to approximate the solution to the generalized Hamiltonian-Jacobi-Bellman equation in order to approximate the solution for a non-linear version of optimal control.

6.1.3 Reinforcement learning

Reinforcement learning, RL, is a form of machine learning which has been a subject of much research within the last couple of decades due to promising results. This section is mainly based on [48] unless otherwise stated.

RL is a ML technique which deals with maximizing a cumulative reward through actions affecting an environment. The concept is inspired by behaviorist psychology, as if a person is given a good reward for doing something, he/she will be more inclined to repeat that action. In a similar fashion, the reward can be a punishment in the case of a bad decision. The optimization problem is set up through a Markov decision process, MDP, environment, which is a discrete time stochastic control process. RL is classified as approximate dynamic programming, since it works without a process model and just converges to some local optimum over time. RL is neither classified as supervised nor unsupervised learning, because it does not get input-output data, but does get feedback in form of a reward, r.

The basic RL setup can be seen in figure 6.4, where the agent is the unit containing the RL algorithm (i.e. the controller). The agent applies an action, a_k, based on the current state and gets a new state, x_{k+1}, and a reward, r_{k+1}, for going from the previous state and into the new state. The agent chooses the action based on its learned policy, which depends on both design and method used for the RL.

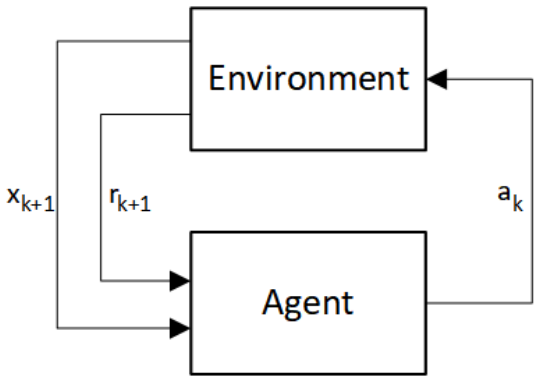


Figure 6.4. The reinforcement learning concept.

Several different methods of RL exists, with different amounts of complexity. Some methods rely on optimizing the value-function describing the cumulative reward for starting in each state, which is called value iteration. The policy to follow is then the one which yields the biggest cumulative reward. Other optimize the policy directly, which is called policy iteration, and lastly some try to optimize both e.g. actor-critic methods. A general thing, which is common for most simpler approaches is, that they rely on discrete state and action spaces. Despite this being an obvious flaw with regards to the continuous nature of the IPC's states, several have made IPC

examples in simulation [12, 21, 44]. However, these are mostly without frictions and thereby not true representatives of the reality.

More complex RL algorithms have in recent years gained a lot of ground, within the ML and AI branch. This is due to the expansion of RL, where it combines other ML techniques such as NNs or deep NNs allowing handling of larger state and action spaces, as well as optimizing the convergence rate of the algorithms. This has especially expanded due to the Google-owned company DeepMind who developed their AI AlphaGO capable of playing the game Go, which is one of the pioneering projects in RL [46]. Finally, RL has been combined with probabilistic estimation of process models and determining the best controller for this, based on the reinforcement obtained. An example of this is e.g. the PILCO algorithm [18], which has shown good results not only in simulation, but also on real systems, by others on a real IPC setup [29].

6.2 Choice of MLC

Based on the brief overview of MLC methods, a method will be chosen for this thesis. As this part of the thesis deals with getting experience with MLC, the choice is mainly based on interests, as it is expected that all the previous mentioned methods can prove successful results. It is desired by the group to work with a model-free approach, as it is found the most fascinating to test and gain experience with for controller designs. This thereby limits the choice to be between GP, RL and the model-free NN approaches.

GP seems promising, however the implementation could be hard and is thereby not chosen due to the limited time available for this part of the thesis. NN in a model-free MLC concept is a well-researched field, with many having applied it to real systems. However, the concept of RL is found fascinating by the group, that only a reward is used to learn the controller. RL is a newer topic and a lot of development is happening within this field. This also means that fewer applications to real systems exist. It has however recently been labeled as the most promising approach to achieve artificial intelligence in terms of control in the future [20].

Therefore, due to personal interests in the group and due to its vast difference from any methods known by the group, it is desired to try the RL approach and gain experience within this field. Thereby the MLC will be designed based on RL in the following sections. First, an overview of some of the main approaches in RL is given in the following section.

6.3 Reinforcement learning theory

The chosen branch of MLC is RL, which however is a quite broad topic with many different methods available [48]. Thereby in the first part of this section, a short description of different selected RL methods are presented, followed by a choice for the thesis. Throughout the following sections the descriptions and theory will be based on [48] unless otherwise stated.

In general, RL is separated into three classes of algorithms, namely value iteration, policy iteration and policy search. In the value iteration case, the algorithm searches for the optimal value function, describing the maximal return for every state-action pair, by iterative improvement of the approximation based on the obtained rewards. The optimal value function is then used to find the optimal policy to follow. In the policy iteration algorithms, the policy is evaluated and

its value function is found, which forms the foundation of deriving a new better policy. Policy search algorithms search directly in the policy space by use of optimization algorithms, but this is in general not used much in RL, due to lack of effective algorithms [9].

Different methods exist for performing the above mentioned iterations. One of the methods is Monte Carlo, MC, which requires that the learning problem is episodic, meaning control starts over after some time, while learning continuous. MC methods trains after each episode has ended and utilizes all the data at the same time. E.g. in policy iteration methods, it is used for policy evaluation, where the MC is estimating the value function for a given state-action pair, (x, a), of the initial state as the sum of rewards in that episode. Given sufficient time and enough visits to all state-action pairs, the algorithm should reach some local optimum for the action-value function. The action-value function describes the expected return of each individual state-action pair. A drawback of this method is that despite having long episodes of training, only the value of the initial state-action pair gets assigned a value and thereby require a lot of episodes for convergence.

Another category of methods is temporal difference, TD, which updates its estimate of the value function at each sample. This also means that problems do not need to be episodic, as was the case for the MC methods. Both the MC and the TD methods rely on a discrete state-action space, but methods are known within TD for approximating the value function with continuous states. TD methods typically converge much faster than MC methods.

A third option is actor-critic, AC, methods, which is a policy iteration method. It consists of two entities, an actor, which goal is to improve the current policy, and a critic, which evaluates the current policy and gives corrections to the actor. In this way, the policy improvement and evaluation step are split over two entities. This method can use both continuous states and actions if approximations are used, however requires more design in the sense that both an actor and critic need to be designed.

Due to the simplicity and its online nature, it is chosen to try to apply a TD method, as it seems to be the most used method for RL online for a wide variety of problems, and thereby is a good starting point in RL. The IPC is often used as starting point in RL simulations. The theory will be described in the following section.

6.3.1 RL using temporal difference

The two methods Q-learning and SARSA within TD have been chosen to try, since they are the two most widely used approaches and thereby a good starting point in RL. Therefore, in the following the theory for these two will be presented, by starting with the general formulations for the problem to solve and TD. It should be noted, that these methods require full state observations, which in a Markow decision process, MDP, sense means it should be fully observable. In a model-free context this could be achieved with e.g. a NN used as an estimator as well as expanding the theory from fully observable MDPs to partially observable MDPs, however due to time constraints this has not been developed and for testing purposes, the EKF is utilized as state observer.

RL builds on a MDP which contains a set of discrete states, $\mathcal{X}$, a set of possible actions, $\mathcal{A}$, and a state transition function, $x_{k+1} = f_t(x_k, a_k) : \mathcal{X} \times \mathcal{A} \to \mathcal{X}$, describing each action's effect in each state. In order to have a MDP setup, the Markov property needs to hold, which means that the effects of an action in the current state is independent of the history. Furthermore in RL, a reward

function, $r_{k+1} = R(x_k, a_k) : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$, describes the immediate reward for taking action a_k in state x_k and going into state x_{k+1}. For the model-free approach desired in the MLC part of this thesis, the transition description, $f_t(x_k, a_k)$, and the reward function, $R(x_k, a_k)$, are unknown. The reward function is a design parameter used to affect what the optimal learned policy is.

The problem of RL is to find a policy, $a_k = \pi(x_k)$, to follow, which maximizes the discounted cumulative reward, R_k, called the return, for starting in state x_k and along the following trajectory. The discounted cumulative reward can also be defined based on the policy, such that it describes the return given a state x_k and following the policy π,

$$R_\pi(x_k) = \sum_{j=0}^{\infty} \gamma^j r_{k+j+1} = \sum_{j=0}^{\infty} \gamma^j R(x_{k+j}, \pi(x_{k+j})), \tag{6.4}$$

where $\gamma \in (0, 1]$ is the discount factor and r_k is the immediate reward at time k and the transition is constrained by f_t. The γ factor is a design parameter, where a high value means the estimate is long sighted and a low value gives a short sighted weight of future rewards. This also gives an opportunity to take into account the increasing uncertainty of future rewards.

In order to determine the optimal policy, the action-value function, $Q_\pi(x_k, a_k) : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$, is defined, which describes the expected return when starting in a given state, x_k, and taking action a_k and from there following policy π. This function is referred to as the Q-function, and can therefore be used as a prediction of the return when starting in x_k and taking action a_k, based on equation 6.4,

$$Q_\pi(x_k, a_k) = R_\pi(x_k) = \sum_{j=0}^{\infty} \gamma^j r_{k+j+1} = r_{k+1} + \sum_{j=1}^{\infty} \gamma^j r_{j+k+1} \tag{6.5}$$

$$= r_{k+1} + \gamma \sum_{j=1}^{\infty} \gamma^{j-1} r_{j+k+1} = r_{k+1} + \gamma R_\pi(x_{k+1}) \tag{6.6}$$

$$= r_{k+1} + \gamma Q_\pi(x_{k+1}, a_{k+1}). \tag{6.7}$$

Equation 6.7 is known as the Bellman equation and is essential to RL, as it describes how the action-value function can be described as a recursive relation, which makes it possible to update the Q-values by use of its own successors. This is in RL refereed to as bootstrapping, which gives the agent the capability of learning while interacting with the environment.

In this discrete action-sense, the action-value function is set up in a table or matrix description, with a value in each entry for each state-action combination. The drawback of this is, that if a very large state-action space is used i.e. for precision, then it could be limited by computational power or memory on the system. This means that smaller state-action spaces are preferred despite the loss of precision.

The action-value function can iteratively be approximated in the model-free setup by use of temporal difference, TD, with

$$Q_\pi(x_k, a_k) \leftarrow Q_\pi(x_k, a_k) + \alpha \left[R_\pi - Q_\pi(x_k, a_k) \right], \tag{6.8}$$

where R_π is the TD target, which is the goal to approximate and α is the learning rate describing how much the value should be updated per time step according to the error between the obtained R_π and the estimated $Q_\pi(x_k, a_k)$. Equation 6.8 can be expanded with the Bellman equation in 6.7, such that $Q_{\pi_k}(x_k, a_k)$ can be approximated with

$$Q_\pi(x_k, a_k) \leftarrow Q_\pi(x_k, a_k) + \alpha[\underbrace{r_{k+1} + \gamma Q_\pi(x_{k+1}, a_{k+1})}_{\text{TD target}} - Q_\pi(x_k, a_k)]. \tag{6.9}$$

In this way, the action-value function can be updated based on other approximated action-value functions, using the sampling approach of TD.

As the action-value function deals with describing future rewards, it is desired to find the policy, which maximizes the return. With foundation in dynamic programming, it is known [14], that when a Q-function has been determined based on a policy, π_k, a new and at least as good policy can be obtained, π_{k+1}, which is greedy with respect to $Q_\pi(x, a)$

$$\pi_{k+1}(x) = \arg\max_{a \in \mathcal{A}} Q_{\pi_k}(x, a). \tag{6.10}$$

The term greedy means that the policy always takes the optimal action, with respect to the obtained action-value function. In this way, the policy can be improved, as the Q-function is approximated better. The policy is first evaluated and then improved, which is called policy iteration. Utilizing this, the policy evaluation and the policy improvement can be combined into one algorithm, which both evaluates and updates the Q-function, as well as acting optimal according to the obtained Q-function. An example of this is the SARSA algorithm described later in this section.

The future discounted rewards, $Q_\pi(x_{k+1}, a_{k+1})$, can be evaluated by multiple approaches, as the next state is obtained as feedback. One approach is to observe the next state and then update the estimate with the action, which gives the maximizing value

$$Q_\pi(x_{k+1}, a_{k+1}) = \max_{a^* \in \mathcal{A}} Q_\pi(x_{k+1}, a^*). \tag{6.11}$$

In this way, the Q-values are always updated with the one which gives the maximum reward. This is an application of the Bellman optimality operator, which states the optimal policy is given as the action maximizing the future rewards

$$Q^*(x_k, a_k) = \max_{a_k, a_{k+1} \in \mathcal{A}} \left(R(x_k, a_k) + \gamma Q^*(x_{k+1}, a_{k+1}) \right). \tag{6.12}$$

This maximization is easy to carry out since it simply is a search of the discrete values in the action-value function and finds the action which has the biggest return in the given state. The actually applied action is afterwards chosen based on the new estimate of the action-value function, $Q_\pi(x_{k+1}, a_{k+1})$.

In order to reach a good description of all state-action combinations, one technique often used during learning is ϵ-greedy policy, where ϵ describes the probability of taking a random action instead of the maximizing action. This is the exploration-exploitation trade-off, where exploration often is needed in order to ensure visits to all states and exploitation is needed for convergence to the optimal value function. The ϵ-greedy policy is therefore

$$\pi(x_k, a_k) = \begin{cases} \text{random } a_k \in \mathcal{A} & \text{with probability } \epsilon \\ \underset{a_k \in \mathcal{A}}{\arg\max}\, Q(x_k, a_k) & \text{with probability } 1-\epsilon, \end{cases} \tag{6.13}$$

where $\epsilon \in [0, 1]$ is the exploration rate. Combining this yields the Q-learning algorithm, as it always learns the Q-value through maximization, no matter the policy used for taking actions. This gives the algorithm seen in table 6.1 for Q-learning.

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\epsilon > 0$, discount rate $\gamma \in [0, 1]$
Initialize $Q(x, a)$ for all $x \in \mathcal{X}, a \in \mathcal{A}$ arbitrarily, except that $Q(terminal, \cdot) = 0$

Loop for each episode
  Initialize x_0
  Loop for each step of episode $k = 0, 1, 2, ...$
    Choose $a_k \in \mathcal{A}$ using ϵ-greedy policy derived from $Q(x_k, a_k)$
    Apply action a_k, observe r_{k+1}, x_{k+1}
    $Q(x_k, a_k) \leftarrow Q(x_k, a_k) + \alpha_k \left[r_{k+1} + \gamma_k \max_{a^* \in \mathcal{A}} Q(x_{k+1}, a^*) - Q(x_k, a_k) \right]$
  until x_k is terminal

Table 6.1. Q-learning algorithm.

It can be seen, that the algorithm is set up to run in episodes, which is applicable for the IPC, since when the pendulum has a too large angle it cannot be saved and the episode is stopped and the state set as terminal. Another terminating condition for the IPC setup is if the end stops of the rail are hit. In this way, Q-learning is a value iteration algorithm, where the maximum Q-value is approximated in each step, and the policy is directly derived greedily from the Q-function. The update parameters α, ϵ and γ are also called hyper parameters and are the learning rate, exploration probability and discount rate respectively.

Another approach updates the action-value function based on the actual action to be taken. This method is called SARSA, because first one observes a state, x_k, sets an action, a_k, observes a reward, r_{k+1}, and state, x_{k+1}, and then sets a new action, a_{k+1}. The acronym thereby comes from State-Action-Reward-State-Action. This means that it is a policy iteration approach, as it evaluates the policy directly in order to approximate the action-value function. The exploration-exploitation is also a key factor in this approach, however this means that the Q-value at a probability of ϵ gets updated through a random action. The algorithm for this is seen in table 6.2.

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\epsilon > 0$, discount rate $\gamma \in [0, 1]$
Initialize $Q(x, a)$ for all $x \in \mathcal{X}, a \in \mathcal{A}$ arbitrarily, except that $Q(terminal, \cdot) = 0$

Loop for each episode
    Initialize x_0
    Choose $a_0 \in \mathcal{A}$ using ϵ-greedy policy derived from $Q(x_0, a_0)$
    Loop for each step of episode $k = 0, 1, 2, ...$
        Apply action a_k, observe r_{k+1}, x_{k+1}
        Choose $a_{k+1} \in \mathcal{A}$ using ϵ-greedy policy derived from $Q(x_{k+1}, a_{k+1})$
        $Q(x_k, a_k) \leftarrow Q(x_k, a_k) + \alpha_k \left[r_{k+1} + \gamma_k Q(x_{k+1}, a_{k+1}) - Q(x_k, a_k)\right]$
    until x_k is terminal

Table 6.2. SARSA algorithm.

The main difference between the two approaches is, that the Q-learning is off-policy and SARA is on-policy. On-policy is, that the update of the approximation is done based on the knowledge that the current policy is being followed in next step. For the off-policy case, the new updated action-value function is used to find an ϵ-greedy optimal action. Thereby the difference is that SARSA applies the action, which is used for the update and Q-learning does not, as it always updates greedily.

This also means that SARSA is more conservative since it does not always update optimally, which depending on the environment can lead to the choice of algorithm. These two algorithms are both tried in the following section through simulation, as well as their obtained policies greedily on the physical IPC setup.

6.4 Implementation and simulation trials

The nature of RL is to be learning directly on the real system, as it is a model-free approach, and thereby a simulation environment for initial training is not expected present. However, as this thesis focuses on getting experience with RL, it is chosen to start learning in simulation by use of the previously developed plant model. The two algorithms shown in tables 6.1 and 6.2 are both implemented in simulation with the ODE environment setup previously described for the system in MATLAB. In this way, initial training can be made in MATLAB and then the training on the system only means refinement to the real system. If the number of training episodes in simulation seems fair for tryout directly on the real system from scratch, this can be done afterward.

Both the state and action space are discretized as required by the two methods. Thereby the number of buckets and how they are divided for each state and action, are tuning parameters. In general, smaller state-action spaces gives a faster convergence and ensures each entry is updated more frequent. However, the cost of this is less refinement in the states used in the policy, and the actions to choose between, causing it to act less like a normal continuous space controller. The design and tuning parameters are therefore as follows:

- State and action bucket amounts

- State and action bucket sizes
- Reward function - both for transitions during the episodes and the terminating states
- Hyper parameters α, γ, ϵ
- Hyper parameters' decay rate

The algorithms are implemented as a function within MATLAB, such that they easily can be converted to C++ code with MATLAB Coder, for testing on the real system. Three phases, training, evaluation and testing on the real IPC, are used in the development, as seen in figure 6.5. First, the training script is set up, where an ODE simulation runs with a random initial state. This simulation runs until a terminating state is reached. If the terminating state is not reached within 20 seconds the simulation is stopped and the specific episode is deemed successful. The simulation is terminated and deemed failed if $|\theta| > \pi/4 \;||\; x < 0 \;||\; 0,77 < x$. The two cart positions represent the two end stops. The simulation runs in a loop, such that a new episode starts when the first terminates. The new episode is initialized in randomly chosen physical states, but the Q-table and policy are kept for further training. Thereby each simulation in the loop constitutes an episode for the learning algorithms.

A second MATLAB script is made as evaluation and validation of the trained policy from the training script. This script uses the greedy policy based on the obtained Q-table. In this environment, the encoder quantization, as well as the EKF is implemented, in order to make it as similar as possible to the real the IPC setup. The Q-table and the information about the discretization, are in the training script saved in a .mat file after ended training. This allows the test environment to automatically adopt the discretization between different tests, directly from the file containing the Q-table. The test environment performs several sequential simulations with random initial states, where the goal of each is to balance 50 seconds, and the average balancing time is desired to be high. If this simulation is successful, the function executing the greedy policy based on the Q-table, as well as the discretization definitions from the .mat file, is converted to C++ by use of MATLAB Coder directly and can be tested on the IPC system. This allows for fast evaluation of the trained policy, from training script to evaluation simulation and to the final IPC system test, as only the discretization is defined in the first training script and automatically is transferred through the two next phases.

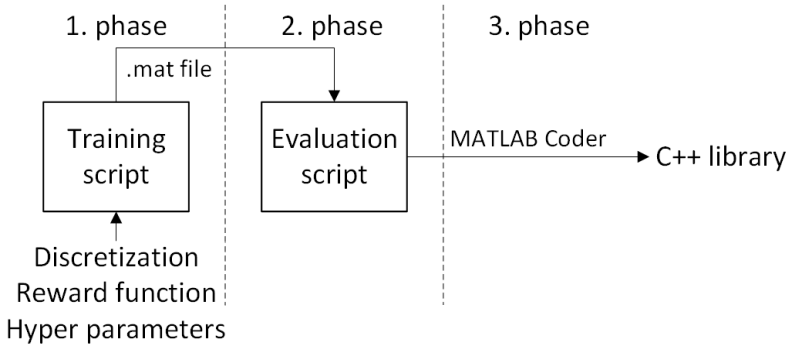


Figure 6.5. The concept of the three phases used for training, evaluation of the learned policy and test of the policy on the real IPC.

As mentioned in the introduction of this chapter, it has not been possible to find a viable policy to be applied on the IPC setup, therefore this section contains a description of what has been tried in order to try to make it work. Section 6.5 shows the most promising simulation results, as well as a non-successful try on the real IPC. The following description of the tried variations of parameters holds for both the Q-learning and SARSA approaches, since both gave similar results and showed the same flaws when applied to the setup. Extensive tries have been put into both Q-learning and SARSA without obtaining good results. There have been difficulties with reaching convergence in simulation and the few which has converged nicely, has only been able to do it on the specific computer with a specific seed for the random generator. That the two different computers used in the thesis did not yield the same training results due to different random sequences from the random generator, indicates how sensitive the converging parameter choices are to the random data it uses for the training, which is problematic.

Several different tries with both the amount of state and action buckets and their sizes have been done, where it is found that in general fewer states bucket are preferred as more buckets easily makes the convergence hard or impossible. It is found that two buckets for the cart position seem to be a good choice in order to achieve convergence, as seen in the illustration in figure 6.6. These are then placed on each side of the middle of the rail as

$$\text{Cart position index} = \begin{cases} 1 & \text{if } x < x_{middle} \\ 2 & \text{otherwise} \end{cases}. \tag{6.14}$$

Alternatively, it could be the cart reference instead of the middle, in order to include this in the quantization. Another possibility is to have three buckets, with the hope that the cart will stay in the middle bucket most of the time. This has been proven difficult, where a large bucket seems to be acceptable despite no clear convergence, however it then drifts a lot around within this bucket, which means tracking is not an option. If the middle bucket is smaller it is harder to achieve convergence. An idea of training on a wide middle bucket and making it smaller after the training has ended has been considered, however never tried as no good convergence on a wide middle bucket is obtained.

The pendulum angle needs to be divided into more buckets, as this is the most important variable

for balancing. One choice for these, which seems to work in simulation, is to have nine buckets divided evenly between $\pm 12°$, where the center bucket covers $\pm 1,5°$ as illustrated in figure 6.6. However, a wide variety of bucket amounts have been tried, as well as not evenly divided sizes.

The two velocities have been kept to a minimum amount such that they are similar to the cart position, just around zero instead, as more just made it harder to converge.

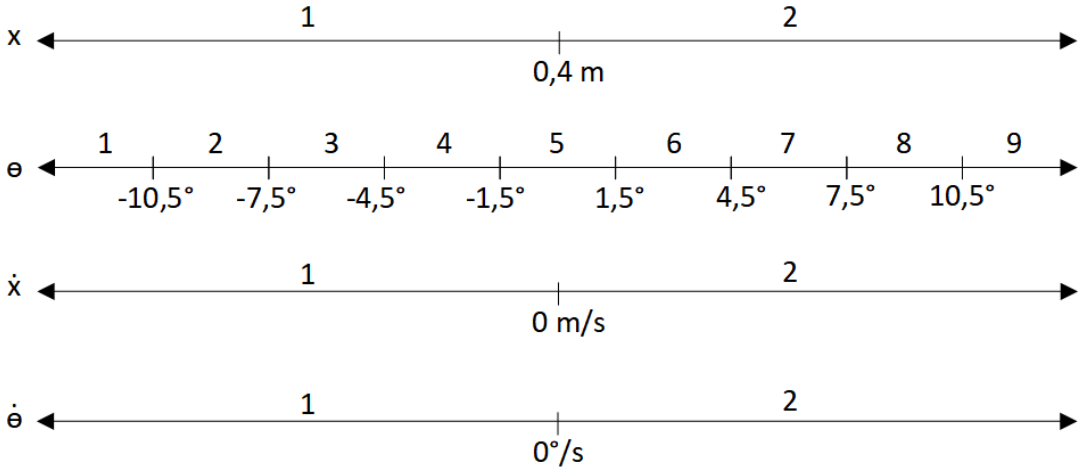


Figure 6.6. The concept of the discretization to indices in the Q-table matrix.

For the action space, several different things seem to work in simulation, where one approach similar to most literature is to simply have two actions - one for each direction (e.g. ± 5 A). Another approach that seems to work decently is a lot of actions spread evenly (e.g. 21 actions within ± 5 A).

It has also been observed, that a lower control frequency of 50 Hz gives much better convergence, which is expected to be due to the increased chance of transitioning between states after an action. Based on this, the MLC runs at 50 Hz, meaning only every third Kalman estimation is fed to the MLC.

Several different reward functions have been tried with varying success rate in simulation. A classic approach within RL is to use the reward function

$$r_k(x_k) = \begin{cases} k_1 & \text{if } x_k \text{ is terminating} \\ k_2 & \text{otherwise} \end{cases} \tag{6.15}$$

where k_2 can be a positive number (often $= 1$) in order to reward longer runs, and a larger negative number k_1, when a terminating state is reached. The idea is, that the system over time learns to balance in order to avoid the terminating states which are punished. Several reward functions similar to cost functions in a control context such as

$$r_k(x_k) = k_0 - k_1 x^2 - k_2 \theta^2 - k_3 \dot{\theta}^2, \tag{6.16}$$

have also been tried, where all gains are positive and k_0 ensures that the best state-action combinations stay the best.

For the hyper parameters several things have been tried as well. Most sources use a large discount parameter, γ, (e.g. $\geq 0,95$) in order to focus on the long-term rewards, a small exploration probability, ϵ, (e.g. $<$ 0,01) and a small to moderate learning rate, α (e.g. $\leq$ 0,3). Especially α determines the speed of the convergence, however too large learning rates might make it hard to obtain convergence. Decaying parameters has also been tried, where especially ϵ needs to decrease in order not to explore too often after some iterations. This has also been applied to the learning rate, α, just with a slower decay rate, e.g.

$$\epsilon_l = 0,999 \cdot \epsilon_{l-1}, \tag{6.17}$$

$$\alpha_l = 0,99999 \cdot \alpha_{l-1}, \tag{6.18}$$

where l is the episode number. Another approach seen in literature is to make the step size uphold the criteria

$$\sum_{l=0}^{\infty} \alpha_l = \infty, \tag{6.19}$$

$$\sum_{l=0}^{\infty} \alpha_l^2 < \infty, \tag{6.20}$$

which guarantees convergence to the optimum, when having a non-changing, discrete and finite system where each state-action pair is visited sufficiently many times. For SARSA an extra criterion is, that $\epsilon \to 0$ for $l \to \infty$, such that the policy ends up being greedy [48]. This has however not given any better results either. These approaches for the hyper parameters have also been tried with minimum bounds, such that parameters do not get too small to have any influence.

6.5 Obtained results

This section shows the obtained results from the simulation as well as a failed attempt on the real IPC. In the end, a discussion about the convergence problem is presented.

An example with good convergence is seen in figure 6.7. The figure shows the balancing time versus the episode number for a training run. Clear convergence is obtained in episode 488. It should be noted, that these parameters are the only ones found, which converges this nicely. The normal obtained type of less clear convergence is presented afterward.

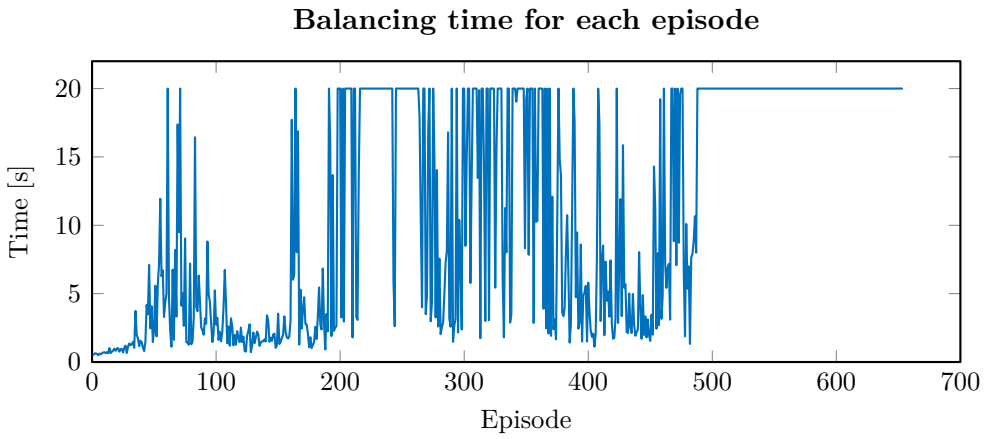


Figure 6.7. Good convergence of training with SARSA with the following parameters. State boundaries: cart position $= \begin{bmatrix} 0,1 & 0,4 & 0,7 \end{bmatrix}$, pendulum angle $= \begin{bmatrix} -0,182 & -0,130 & -0,078 & -0,026 & 0,0 & 0,026 & 0,078 & 0,130 & 0,182 \end{bmatrix}$, cart velocity $= \begin{bmatrix} -0,012 & 0 & 0,012 \end{bmatrix}$, angular velocity $= \begin{bmatrix} -0,15 & 0 & 0,15 \end{bmatrix}$, action space $\mathcal{A} = \begin{bmatrix} -3 & 3 \end{bmatrix}$ and hyper parameters: $\gamma = 0,99$, $\alpha_0 = 0,5$, $\alpha_k = \max(0,995 \cdot \alpha_{k-1}, 0,2)$, $\epsilon_0 = 0,05$, $\epsilon_k = 0,99 \cdot \epsilon_{k-1}$.

The parameters for the training are described in the figure text. The boundaries describe where the buckets separates. E.g. in the case of a boundary defined as $\begin{bmatrix} -2 & 2 \end{bmatrix}$ would have a buckets in the interval $B_1 = [-\infty, -2)$, $B_2 = [-2, 2)$ and $B_3 = [2, \infty]$.

It is however also seen that it almost converges a couple of times earlier, but fails again. This is expected to be due to suddenly entering an insufficiently visited state or because the cart drifts into the rail and the probably decent policy is forgotten due to the punishment from the terminating state. This training is thereby moved to phase two in figure 6.5, in order to validate the learned policy. The result here is, that all 20 runs are successful at the full 50 seconds, which indicates a successful policy is obtained, in terms of keeping the system from failing. One of the evaluation runs from phase two can be seen in figure 6.8.

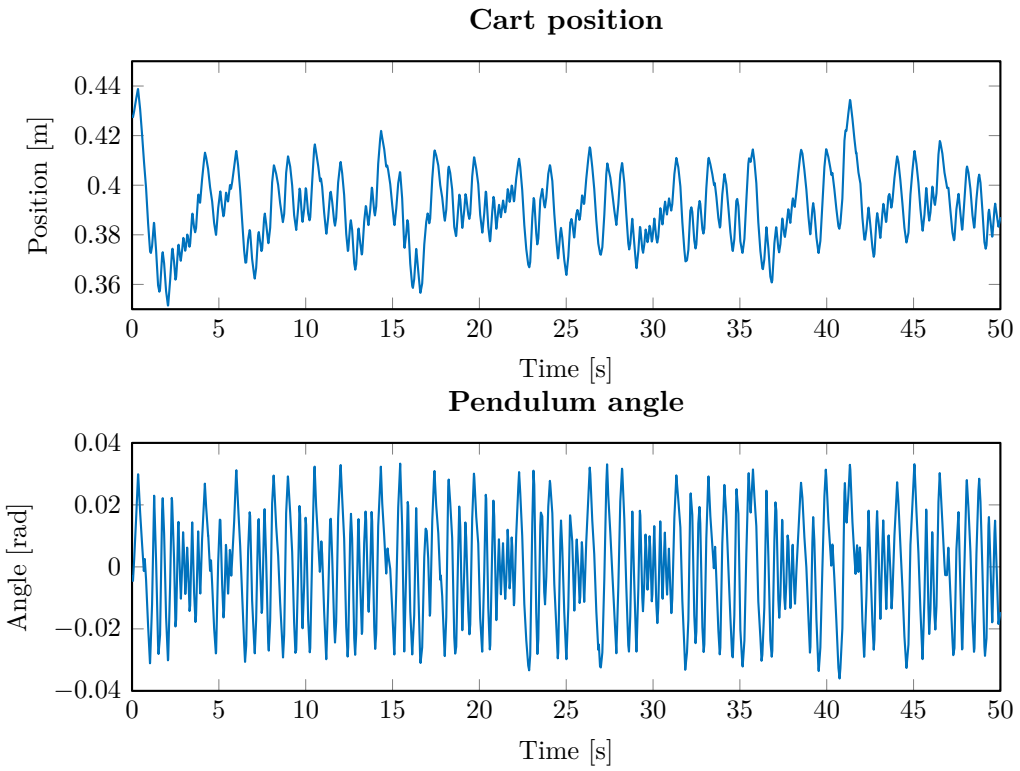


Figure 6.8. A single trial of the simulation test from phase two, of the policy trained in figure 6.7.

It can be seen, that the cart mainly oscillates back and forth between the two middle buckets. This is deemed a good simulation result with fine tracking of the boundary between the two middle buckets on 0,4 m, which as previously mentioned could be changed for the cart reference.

However, when this is applied to the real IPC system, it fails to balance for more than two seconds, despite starting in an initial angle of $\theta \approx 0$ and close to the center of the rail. Just before it fails, the pendulum has a small angle and instead of balancing the pendulum, it loses the pendulum angle to a point, where it cannot catch it again and simply pushes the cart to a side until it hits the rail. An example of this is seen in figure 6.9.

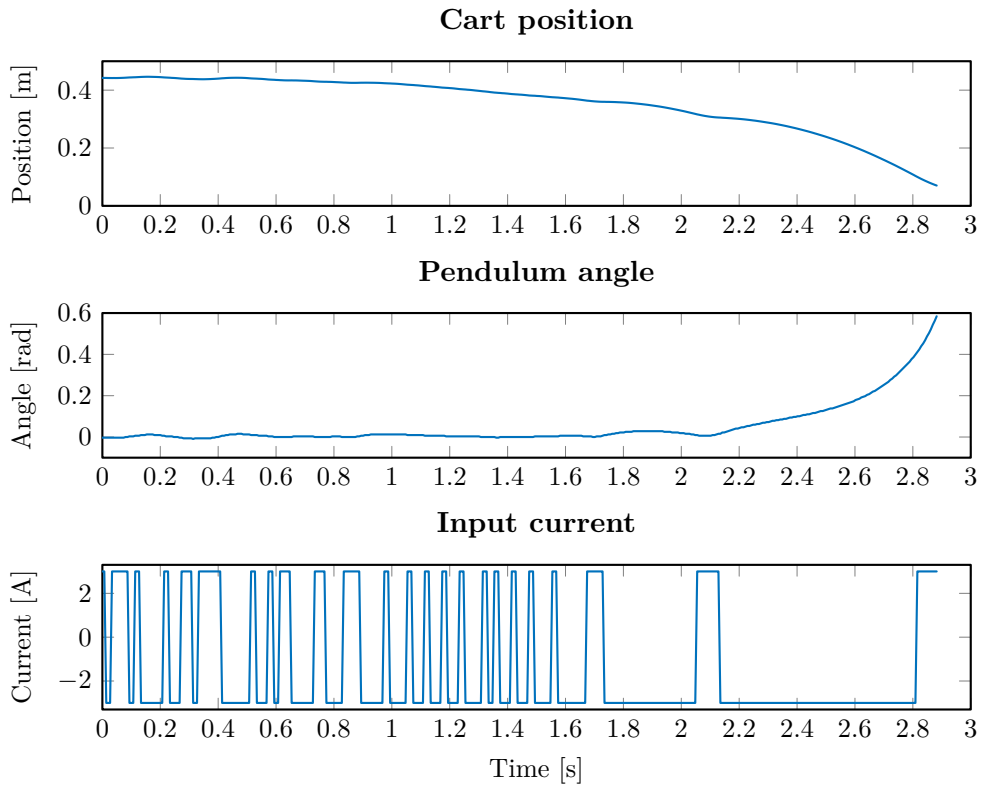


Figure 6.9. Test on the real IPC setup of the policy trained in figure 6.7.

As it can be seen, the control current looks as expected and maximum negative current is applied in order to try to save the pendulum angle, however this is not successful. The observed behavior could indicate that the cart frictions in the real setup are higher than in the simulation, as the cart does not get properly under the pendulum to keep it stabilized. Due to this, it has been tried to increase the cart friction in the test simulation, where the learned policy is evaluated. In this case, the simulation shows similar results as in figure 6.9 and loses the pendulum. Due to this, it has been tried to train on a model, where the friction has been increased in the cart. The evaluation still looks good in simulation, however the same behavior is seen on the real IPC. It has also been tried to multiply the control current on the real IPC with a factor, in order to increase it as a compensation for maybe higher friction on the real system. It has also been tried to increase the current for the training. This has all showed the same type of behavior, though without the clear convergence seen in figure 6.7. Other of the attempts have not converged at all.

Many of the tried parameter combinations have resulted in no convergence at all, and only the previously presented have converged correctly. Multiple tries have though been obtained, where some convergence has been obtained, but not as clear as in the previously shown case. Such an example can be seen in figure 6.10. It can be seen, that some convergence is reached after 1246 iterations. However, a lot of spikes where only short balancing time is reached can be seen afterward.

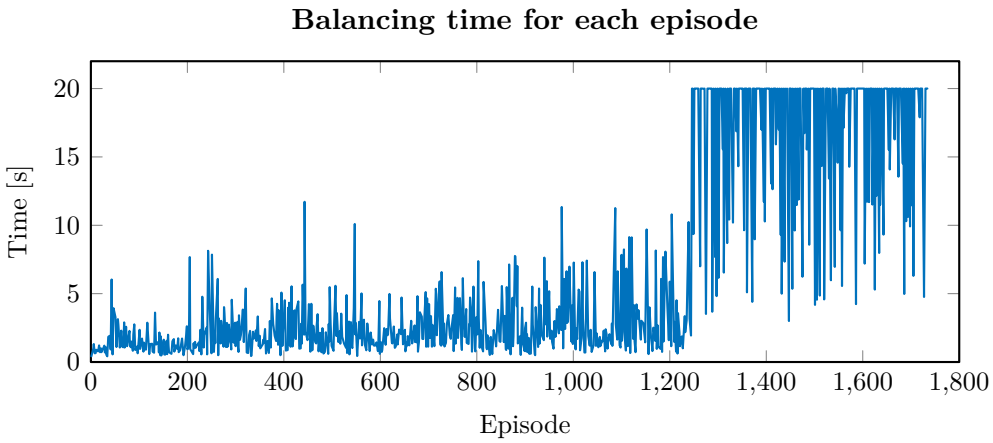


Figure 6.10. Decent convergence of training with SARSA with the following parameters. State boundaries: cart position = $\begin{bmatrix}0, 4\end{bmatrix}$, pendulum angle = $\begin{bmatrix}-0,182 & -0,130 & -0,078 & -0,026 & 0,026 & 0,078 & 0,130 & 0,182\end{bmatrix}$, cart velocity = $\begin{bmatrix}0\end{bmatrix}$, angular velocity = $\begin{bmatrix}0\end{bmatrix}$, action space $\mathcal{A} = \begin{bmatrix}-5 & 0 & 5\end{bmatrix}$ and hyper parameters: $\gamma = 1$, $\alpha = 0,3$, $\epsilon_0 = 0,001$, $\epsilon_k = 0,99 \cdot \epsilon_{k-1}$.

By checking the state trajectories from the training episodes for these specific parameters after the convergence, it is seen, that with these parameters the cart always stays in the right x-bucket of the two used in this simulation. As only two buckets are used, it cannot see when it is close to the end of the rail and thereby drift around inside this bucket and accidentally hits the right end of the rail.

If a training with this type of convergence is stopped and the policy then evaluated, it normally shows decent results in the evaluation script. In general, these do not behave as well as the one shown in figure 6.8 and drifts quite a bit more and with larger angles during the balancing. In most of the cases, it is however capable of balancing the 50 seconds in the simulation in phase two. Nevertheless, when applied to the real IPC system, it behaved similarly to the shown test in figure 6.9 and fails after maximum a couple of seconds.

The same behavior is seen for all trials on the IPC setup, despite many different training parameters, where they look acceptable in simulation evaluation. It is expected that it might be because of states which were visited insufficiently during training are reached, or due to the simulation model not representing the system fully, especially the varying friction along the rail. The fact that the model does not fully fit is a drawback, and training further directly on the IPC could be an obvious method. However, due to the long convergence times and unclear convergence in most of the cases within simulations, this has not been tried.

During the testing of different parameters, the MATLAB code has been checked extensively, as well as rewritten in order to minimize the chance of coding errors. Furthermore, a MATLAB implementation of SARSA on an inverted pendulum on a cart has been found and used for comparison [33]. This is a MATLAB implementation of the OpenAI Gym environment [41], which builds on the model and parameters set in [6], which is widely used for comparing and showing results within RL, where the inverted pendulum on a cart is used.

The found MATLAB code builds on a frictionless setup with the rail length increased from the normal 4,8 m from the OpenAI Gym environment, to 8 m. This learning setup converges very fast using the SARSA algorithm, often in a couple of hundred episodes. Based on this, the learning algorithm in this script is concluded working. The simulation model is thereby interchanged with the simulation setup of this thesis, with a rail of $\approx 0,8$ m. In this case, the convergence is much slower if converging at all. If it converges, it is with the spikes as seen in figure 6.10, and no clear convergence is found. Both in our own and in the found MATLAB implementation, our simulation model has been tried both with and without frictions, with no change in the results. Due to this, it can be concluded that the dimensions and parameters of the IPC setup for this thesis must be much harder to do RL on, than the ones normally used in examples online. This is expected to be due to the fact, that when the rail is longer, the cart is allowed to drift more and not hitting the rail before the episode is stopped due to time.

It has been tried to increase the rail length for the IPC model of this thesis, which eases the convergence, but not to clear convergence. This shows that these methods are not well suited for IPC setups with the dimensions and parameters of this thesis, e.g. short rail length. Furthermore, these methods are not deemed good for tracking either since the cart position needs to quantized that hard. The standard simulation used in RL therefore seems to be much easier to control and be a little more forgiving, not only due to the rail length, but also the other model parameters, which is the expected reason for no good results in this thesis, even though a lot of examples are available online for the inverted pendulum on a cart in RL.

It was expected that this quantified approach would deliver a poor tracking performance and thereby function approximation has simultaneously been investigated in order to handle continuous state spaces. This also should limit the number of adjustable parameters which are to be chosen by hand, and make tracking of the cart position better. This is described in the following section.

6.6 Function approximation of the action-value function

RL methods which rely on discrete state and action spaces such as the previously described Q-learning or SARSA clearly suffers from "the curse of dimensionality". As the state and action space grows, the number of entries in the action-value table grows exponentially, as the number of entities is given by the product of the numbers of discretizations in each state and the action space. This causes problems for large spaces, as the states are not updated often enough and will have limited experience. Due to this, the policy converges very slowly, if converging at all [48]. Given that the nature of the system is discrete, the table representation is the precise representation of the action-value function. When manual discretization is applied to the system, as in the previous section for the IPC, the action-value function is approximated by the table which can cause problems as previously described. Due to this, other approximation methods are used, which can deal with continuous states and thereby could give an advantage in the precision of the approximation. Furthermore, using function approximation, FA, for the action-value function, has the advantage of generalization. This means, that given a new state is encountered, the action is chosen based on previous encounters and experiences close to the new state. In this way, the algorithm is able to handle previously unknown states. However, generalization can also have the disadvantage, that every change in the parameters affects other states as well, which potentially

in some situations can decrease the performance of the other states.

The problem of function approximation for reinforcement learning is to approximate the real action-value function, $Q_\pi(x, a)$,

$$\hat{Q}(x, a, w) \approx Q_\pi(s, a), \tag{6.21}$$

by use of the approximation function, $\hat{Q}(x, a, w)$, parameterized by a set of parameters or weights, $w \in \mathbb{R}^m$. The task is then to update the weights, w, as new information is obtained in an incremental way, such that the real action-value function is approximated.

Many function approximators exist, however some of them are particularly often used in RL, which are linear combinations of features or neural networks [45]. It is chosen to try the linear combination of features in this thesis, as it has often been mentioned in the literature as well as successfully applied in simulation to a double inverted pendulum on a cart in [21]. A feature vector is a vector of linearly independent functions, $\phi_i(x, a) : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$, also sometimes referred to as basis-functions, where $\mathcal{X}$ and $\mathcal{A}$ are the state space and action space respectively. The feature vector

$$\phi = \begin{bmatrix} \phi_1(x, a) & \phi_2(x, a) & \cdots & \phi_m(x, a) \end{bmatrix}^T \tag{6.22}$$

extracts the relevant data for the approximation from the original data and is a general technique in ML. The approximation of the action-value function, $\hat{Q}(x, a, w)$, is then expressed in terms of a linear parameterization of the elements from the feature vector

$$\hat{Q}(x, a, w) = \sum_{i=1}^{m} \phi_i(x, a) w_i = \phi^T(x, a) w. \tag{6.23}$$

Identification of the features are problem specific and are in general not trivial to design [5], however the number of parameters only grows linearly with the number of features, which is usually much smaller than the required number of discrete states. It is clear, that an approximation error will be present if the real action-value function is not possible to describe linearly by the basis functions, from where the design of these is critical.

As a relation to the described discrete states from the previous section, it can be seen that it is a special case of function approximation utilizing equation 6.23. In the discrete case, the basis functions are used as indicator functions for the state and action space separated into m disjoint subsets X_i for $i = 1...m$, such that

$$\phi_{i+(j-1)/m}(x, a) = \begin{cases} 1 & \text{if } x \in X_i \text{ and } a = a_j \\ 0 & \text{Otherwise} \end{cases}, \tag{6.24}$$

where a_j is the j^{th} discrete action [9] (section 3.3). In this case, only one basis function is active at the time, corresponding to an indication of the current discrete state the system is in. The weight

for each basis function, are thereby the Q-value for the respective state. Thereby the discrete case is a special case of function approximation, indicating that equation 6.23 seems intuitively valid. This type of feature vector is of cause not desired, as nothing will be gained compared to the previously described Q-table. The tried feature vectors for the thesis is described later in section 6.6.1.

In order to update the weights of the linear combination, stochastic gradient descent, SGD, is utilized, which requires the feature vector to be differentiable. Recall from section 6.3.1 that the update of the Q-function for Q-learning is given as

$$Q(x_k, a_k) \leftarrow Q(x_k, a_k) + \alpha \left(r_{k+1} + \gamma \max_{a^*} Q(x_{k+1}, a^*) - Q(x_k, a_k) \right). \tag{6.25}$$

In order to derive the update of the weights in the FA case, it is assumed that the correct Q-value for the current state-action pair, $Q^*(x_k, a_k)$, is known. SGD is then utilized to minimize the squared error between the optimal value and the current Q-value for the given state and action. The mean squared error, MSE, to be minimized in each step is thereby

$$J(w) = E_\pi \left[\left(Q^*(x_k, a_k) - \hat{Q}(x_k, a_k, w) \right)^2 \right], \tag{6.26}$$

where $E_\pi[\cdot]$ denotes the expected states and actions given policy π is followed [45]. Gradient descent states, that given a differentiable function, $J(w)$, parameterized by weights, w, a local minimum can be found by adjusting the weights in the direction of the negative gradient

$$\Delta w = -\frac{1}{2}\alpha \Delta_w J(w) = -\frac{1}{2}\alpha \begin{bmatrix} \frac{\partial J(w)}{\partial w_1} & \cdots & \frac{\partial J(w)}{\partial w_m} \end{bmatrix}^T, \tag{6.27}$$

where α is the step size and $\Delta_w J(w)$ is the gradient with respect to the weights, of the function to minimize. Utilizing this to minimize equation 6.26 yields

$$\Delta w = -\frac{1}{2}\alpha \Delta_w J(w) = \alpha E_\pi \left[\left(Q^*(x_k, a_k) - \hat{Q}(x_k, a_k, w) \right) \Delta_w \hat{Q}(x_k, a_k, w) \right]. \tag{6.28}$$

As TD is utilized, the gradient is sampled at each timestep, from where it can be shown through stochastic approximation theory, that updating with gradient descent at each timestep, known as SGD, will equal calculating the expected value, from where the expectation can be left out [45]. The weight adjustment at each timestep is thereby given as

$$\Delta w = \alpha \left(Q^*(x_k, a_k) - \hat{Q}(x_k, a_k, w) \right) \Delta_w \hat{Q}(x_k, a_k, w). \tag{6.29}$$

Utilizing this for optimizing a function which is linear in the parameters, LIP, as equation 6.23, the gradient is given as

$$\Delta_w \hat{Q}(x_k, a_k, w) = \phi(x_k, a_k) \tag{6.30}$$

and thereby the update of the weights from equation 6.29 will be

$$\Delta_w = \alpha \left(Q^*(x_k, a_k) - \hat{Q}(x_k, a_k, w) \right) \phi(x_k, a_k). \tag{6.31}$$

However, in reality, the optimal Q-value, $Q^*(x_k, a_k)$, is not known, as the concept of RL is that nothing is known in advance. It is from section 6.3.1 known, that TD can be used to approximate the optimal Q-value. Because of this, $Q^*(x_k, a_k)$ is substituted by the TD target in order to give the function approximator a target function to approximate [45]. The update of the weights are thereby

$$\begin{aligned} w &\leftarrow w + \Delta_w && (6.32)\\ &= w + \alpha \left(r_{k+1} + \gamma \max_{a^*} \hat{Q}(x_{k+1}, a^*, w) - \hat{Q}(x_k, a_k, w) \right) \phi(x_k, a_k) && (6.33)\\ &= w + \alpha \left(r_{k+1} + \gamma \max_{a^*} \phi^T(x_{k+1}, a^*) w - \phi^T(x_k, a_k) w \right) \phi(x_k, a_k), && (6.34) \end{aligned}$$

as equation 6.23 is utilized in the Q-learning algorithm. Similarly, the same principle can be used for SARSA, where both still requires exploration e.g. in terms of ϵ-greedy. The on-policy algorithm SARSA converges to a point, where it chatters around the optimal solution. However, the off-policy algorithm Q-learning is not guaranteed to converge, but will in many cases work fine [45]. For non-linear approximators such as NNs, neither of the methods have guaranties for convergence in this simple form, but also often performs well [45].

6.6.1 Design of reinforcement learning based on temporal difference with function approximation

In [21] a double inverted pendulum on a cart is successfully stabilized by use of Q-learning utilizing linear function approximation by use of a feature vector in simulation in OpenAI Gym. With the inspiration of the therein utilized features, the following feature vector is tried in the simulation environment for this thesis. The feature vector is defined as

$$\phi(x, a) = \begin{bmatrix} x_1 & x_1^2 & x_2 & x_2^2 & x_3 & x_3^2 & x_4 & x_4^2 & ax_1 & ax_2 & ax_3 & ax_4 \end{bmatrix}^T. \tag{6.35}$$

It is observed, that the weight diverges to ∞ or $-\infty$ in just a couple of tens of episodes. This has been observed for all tried feature vectors and hyper parameters. Some literature such as [34] states, that in order to improve the convergence

$$\sum_i |\phi_i(x, a)| \leq 1 \tag{6.36}$$

should hold for all state-action pairs. If normalization of the feature vector is implemented in the simulation, the weights do not diverge, however still no useful learning is obtained. This is the case both for Q-learning and SARSA. A vast variety of both feature vectors and reward functions,

as well as tuning of the hyper parameters, have been tried. It has also been tried to remove the rail length constraint, such that the rail is infinitely long, still without good results, as well as removing the frictions and increasing and decreasing the action space, all without success. Instead of incorporating the action into some of the features, another approach where state-depended feature vectors, each consisting of N basis functions $\bar{\phi}_1(x)...\bar{\phi}_N(x) : \mathcal{X} \to \mathbb{R}$ independent of the action, is stacked, one for each action [5, 9, 45]. Using this method, all the basis functions not corresponding to the current action are zero, as seen here for the j^{th} action

$$\phi(x, a_j) = [\underbrace{0 \quad \cdots \quad 0}_{a_1} \quad \cdots \quad \underbrace{\bar{\phi}_1(x) \quad \cdots \quad \bar{\phi}_N(x)}_{a_j} \quad \cdots \quad \underbrace{0 \quad \cdots \quad 0}_{a_M}]^T. \tag{6.37}$$

By using this approach, it is not necessary to determine how to incorporate the actions into the feature vector itself. Furthermore, this method describes one whole function for each action, meaning one slice through the Q-value function per action and can thereby be more precise and remove a design parameter [9]. However, this has not given any good results either.

It has in this thesis only been tried with polynomials of the states as basis functions, as the previous example. In general, polynomials also includes adding a 1 as the first feature in order to be able to introduce a bias, as well as features which are products of the original states, in order to take interaction between states into account. No combination has been found to give good results. However instead of defining the features by hand, other LIP approximations could be used, such as coarse coding, radial basis functions or similar [48] (section 9.5), which have not been investigated due to time constraints.

Another example of Q-learning using feature-based FA successfully applied to an IPC simulation is seen in [39], which is not based on the normally used model in RL, but still has a rail length of 4,8 m. Based on the found successful papers as well as the vast majority of literature describing this as a normally used approach to deal with big or continuous state spaces, it was initially expected to work. As described in section 6.5 the problems could be due to the specific system and its dimensions and parameters, but could also be due to an implementation error, even though extensive checking of the code against multiple sources have been made throughout the thesis.

Function approximation in terms of a linear combination of features or even a non-linear NN approximation were expected to be able to improve the clear flaw, from a control perspective, of the discretization from the methods from previous section. However, due to the problems, time constraints never allowed for tries with NNs as expected in this thesis, which together with a properly designed reward function were expected to give acceptable results from a control perspective.

Research in RL today focuses on improving the convergence speed and guaranties of the algorithms and especially deep NNs are a big research area, as it has shown great improvements and the capability of solving advanced RL problems [46]. Deep NNs for Q-learning is referred to as deep Q-network, DQN, and is considered the newest breakthrough in RL and has boosted the activity and interest of RL. The deep NN allows the NN to find its own most optimal features instead of designing them by hand and combined with batch updates, experience replay of state-transitions with corresponding actions and rewards as well as a lot of other methods, this has shown great results. This has however not been investigated in this thesis due to time constraints.

6.7 Conclusion

Throughout this chapter, an overview of MLC approaches has been established and reinforcement learning has been further investigated and tried in the simulation environment for the IPC for this thesis. Even though multiple successful simulation results are found online with stabilization of an inverted pendulum on a cart by use of Q-learning and SARSA algorithms, it has not been possible to obtain successful results for this thesis. Stabilization in simulation has been obtained, however mostly not with clear convergence. The trained controller is however not capable of stabilizing the IPC in the real setup. No solution to this is found and can be both due to a code error, as well as a problematic dimension/parameter combination of the IPC setup used in this thesis. However, even if a parameter combination is found, which constitutes a success, this method is not deemed appropriate for control of the IPC of this thesis. This is due to the extensive amount of tuning required as well as the poor tracking performance due to the discretization. When function approximation is utilized, no successful results have been obtained, even though successful simulation examples have been found online on other IPC systems. Due to extensive debugging and tries in order to get it working, more complex methods have not been investigated due to time constraints.

7 Conclusion

This thesis has been made as an investigation of two control techniques, non-linear classic control and machine learning control, MLC, applied to an inverted pendulum on a cart, IPC. In order to design the classic control and have a simulation environment for easier testing, a model for the IPC setup has been derived and the parameters of the system have been estimated. The simulation environment mimics the dynamics of the system quite well, as well as the sensor quantization of the real system. The largest deviations between the simulation environment and the real IPC, is that the real IPC setup has varying frictions along the rail.

As two control methods have been investigated, a control structure has been set up such that easy testing and comparison is available on the IPC setup. A friction compensator has also been developed in order to slightly ease the design of the non-linear control as well as the state observer. This is able to approximately remove the Coulomb friction affecting the cart. Along with this, an extended Kalman filter, EKF, has been designed, as only sensors for the cart position, x, and pendulum angle, θ, are available. The EKF is able to estimate all the states of the derived model very well, by only using the two sensors. The EKF was intended to be used for only the non-linear control, however due to time constraints it was also used for the MLC.

In the non-linear control, it was chosen to use a sliding mode controller, SMC, where two controllers were developed, which both are able to balance the pendulum in the unstable equilibrium. The difference between the two designs is that one of them has varying aggression depending on how far from the reference point it is, where the other controller is equally aggressive all over. The designs did not include reference tracking, however by changing the reference for the cart slowly, both the controllers are able to track a reference. A steady state error is present in the cart position, which depends on the position on the rail due to the varying friction. Overall however the two designed SMC controllers are deemed quite good with the ability to stabilize the pendulum satisfactorily, as well as be able to handle disturbances and parameter variations.

For the MLC an overview of some methods has been given and the choice of method fell on reinforcement learning, RL, due to personal interests. In the thesis the two methods SARSA and Q-learning have been tried. There have been several problems with these methods in the thesis, as it has not been possible to find a solution able to balance the real pendulum. It has however been possible to find several combinations of design parameters, which are able to balance the pendulum in simulation, but not in the real system. Learning directly on the IPC setup has not been done, since convergence in simulation often takes up to several thousands of episodes, which would be very tedious and time-consuming. Furthermore, convergence-like behavior is only obtained in some cases, which means it might not even be possible to obtain given the first set of chosen parameters on the real system. Several comparisons with other implementations and simulations have been done, in order to search for mistakes in the learning setup, but to no avail. It has by incorporating the simulation model of this thesis into another found learning demonstration in MATLAB been concluded, that convergence of the setup of this thesis, is much harder to obtain than for the normal IPC model used in the RL community. This could be the reason for, that no good results have been obtained. Simultaneously, function approximation of the action-value function has also been investigated in order to handle continuous states. This has however not

shown any good results either, as convergence has not even been reached in simulation. Overall it has been concluded that simple RL methods like Q-learning and SARSA are not well suited for the IPC setup available in the thesis.

8 Future work

In this thesis, four main concepts have been investigated namely modelling of the system, Kalman filtration, non-linear control and machine learning. In the following, the future work for the four concepts is described.

The obtained model for the inverted pendulum on a cart, IPC, is deemed to fit the behavior of the real system well enough for control purposes. The main problem is the varying friction along the rail on the real system, which is not present in the model. However, obtaining the parameters for modelling the varying friction is deemed very hard and thereby this would not be investigated further. Instead the designed controllers should be capable of handling this when going from simulation to the real system.

The designed observer based on an extended Kalman filter works very well and the state estimates follow the states closely in the non-linear simulation, as well as performs well on the real system. No future work is deemed necessary for this part of the thesis.

The designed non-linear controller in terms of a sliding mode controller has a steady state error in the cart position, due to the friction in the cart. This steady state error is furthermore varying depending on the reference position, as the friction varies along the rail. In order to account for the steady state error, an integrator should be incorporated into the sliding mode controller.

The tried machine learning controller in terms of reinforcement learning has not yielded any good results. The problem is expected to be due to the dimensions and parameters of the IPC setup used in this thesis, which is not deemed suited for the tried concepts of reinforcement learning. In order to validate this, it should be tried to use the simulation model used in the reinforcement community, in the learning script for this thesis, to check whether this converges easier. The tried methods of RL can be expanded into using deep neural networks as function approximators, which expands the tried methods for more advanced problems. It is however unclear, whether this can solve the problem or not. Instead, other types of reinforcement learning should be investigated, e.g. an actor-critic setup, which might be better suited for continuous state spaces and in general is a more advanced method. Other methods such as PILCO have also shown great results, also on a physical IPC, showing swing-up and stabilization after only 17,5 seconds of interaction with the setup [29]. This method is a model learning reinforcement learning method, meaning it learns the model of the system and uses the reinforcement to determine the best controller. Other machine learning methods such as NN methods could also be investigated for the IPC setup of this thesis. Furthermore, it is, in the end, desired to investigate, how some machine learning controller can be expanded to run without the model based Kalman filter.

Bibliography

[1] Murad Abu-Khalaf and Frank L. Lewis. Nearly optimal state feedback control of constrained nonlinear systems using a neural networks hjb approach. *Annual Reviews in Control*, 28(2):239 – 251, 2004.

[2] Sören Andersson, Anders Söderberg, and Stefan Björklund. Friction models for sliding dry, boundary and mixed lubricated contacts. *Tribology international*, 40(4):580–587, 2007.

[3] T.M. Apostol. *Mathematical Analysis 2nd ed.* Addison-Wesley series in mathematics. Addison-Wesley, 1974.

[4] Arduino. Arduino due specification. `https://store.arduino.cc/arduino-due`, 2018.

[5] André da Motta Salles Barreto and Charles W Anderson. Restricted gradient-descent algorithm for value-function approximation in reinforcement learning. *Artificial Intelligence*, 172(4-5):454–482, 2008.

[6] Andrew G Barto, Richard S Sutton, and Charles W Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5):834–846, 1983.

[7] Alexander Bogdanov. Optimal control of a double inverted pendulum on a cart. Technical report, CSEE, OGI School of Science and Engineering, OHSU, 2004.

[8] Basilio Bona and Marina Indri. Friction compensation in robotics: an overview. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pages 4360–4367. IEEE, 2005.

[9] Lucian Busoniu, Robert Babuska, Bart De Schutter, and Damien Ernst. *Reinforcement learning and dynamic programming using function approximators*, volume 39. CRC press, 2010.

[10] Harun Chowdhury, Hazim Moria, Abdulkadir Ali, Iftekhar Khan, Firoz Alam, and Simon Watkins. A study on aerodynamic drag of a semi-trailer truck. *Procedia Engineering*, 56:201 – 205, 2013. 5th BSME International Conference on Thermal Engineering.

[11] Jesper Hede Christensen and Rasmus Christiansen. Swing up and stabilisation of an inverted pendulum on a cart using nonlinear methods. Master thesis, Aalborg University, 2017.

[12] Geoffrey Clark, Venkatavaradhan Lakshminarayanan, and Shubham Sonawani. Final project - q learning to balance inverted pendulum. 2016.

[13] Shuang Cong, Guodong Li, and Beichen Ji. A novel pid-like neural network controller. *IFAC Proceedings Volumes*, 38(1):121 – 126, 2005. 16th IFAC World Congress.

[14] André da Motta Salles Barreto and Charles W. Anderson. Restricted gradient-descent algorithm for value-function approximation in reinforcement learning. *Artificial Intelligence*, 172(4):454 – 482, 2008.

[15] Elmer P. Dadios, Patrick S. Fernandez, and David J. Williams. Genetic algorithm on line controller for the flexible inverted pendulum problem. *JACIII*, 10:155–160, 2006.

[16] Olurotimi Dahunsi, Jimoh Pedro, and OTC Nyandoro. System identification and neural network based pid control of servo- hydraulic vehicle suspension system. 101, 09 2010.

[17] Technical editor for controldesign.com Dave Perkon. Is machine learning smart enough to help industry? `https://www.controldesign.com/articles/2016/is-machine-learning-smart-enough-to-help-industry/`, April 2016.

[18] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

[19] Thomas Duriez, Steven L Brunton, and Bernd R Noack. *Machine Learning Control-Taming Nonlinear Dynamics and Turbulence*. Springer, 2017.

[20] Camron Godbout. The bright future of reinforcement learning. `https://medium.com/apteo/the-bright-future-of-reinforcement-learning-a66173694f88`, March 2018.

[21] Fredrik Gustafsson. Control of inverted double pendulum using reinforcement learning. 2016.

[22] Dr. Nhut Tan Ho. Course in modeling and simulation of dynamic systems.

http://www.ecs.csun.edu/~nhuttho/me584/Chapter%203%20Mechanical%20Systems_part2.pdf, September 2010.

[23] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[24] Jesper H. Jørgensen, Jonas Ørndrup, Jacob Kjersgaard, and Andrea Løvemærke. Automated container crane. Bachelor thesis, Aalborg University, 2016.

[25] Boonsri Kaewkham-ai and Kasemsak Uthaichana. Comparative study on friction compensation using coulomb and dahl models with extended and unscented kalman filters. In *Industrial Electronics and Applications (ICIEA), 2012 7th IEEE Conference on*, pages 191–195. IEEE, 2012.

[26] H.K. Khalil. *Nonlinear Systems*. Pearson Education. Prentice Hall, 2002.

[27] Torben Knudsen. Slides from course nonlinear control systems at aalborg university. As digital attachment or online (requires access) at https://www.moodle.aau.dk/pluginfile.php/1045899/mod_resource/content/3/KF.pdf, 2017.

[28] Torben Knudsen. Slides from course nonlinear control systems at aalborg university. As digital attachment or online (requires access) at https://www.moodle.aau.dk/pluginfile.php/1045902/mod_resource/content/5/EKFAndUKFHandOut.pdf, 2017.

[29] PILCO learner. Cart-pole swing-up and stabilization with pilco. https://www.youtube.com/watch?v=XiigTGKZfks.

[30] FL Lewis and Shuzhi Sam Ge. Neural networks in feedback control systems. *Mechanical Engineer's Handbook*, 2005.

[31] FW Lewis, Suresh Jagannathan, and A Yesildirak. *Neural network control of robot manipulators and non-linear systems*. CRC Press, 1998.

[32] Broadcom Limited. Datasheet for broadcom limited hctl-2021-plc quadrature decoder/counter. https://docs.broadcom.com/docs/AV01-0041EN, 2006.

[33] Jose Antonio Martin. Reinforcement learning cart pole problem with sarsa. https://github.com/david78k/pendulum/tree/master/matlab/SARSACartPole.

[34] Francisco Melo and Isabel Ribeiro. Q-learning with linear function approximation. 2007.

[35] Valeri Mladenov, Georgi Tsenov, L Ekonomou, Nicholas Harkiolakis, and Panagiotis Karampelas. Neural network control of an inverted pendulum on a cart. 8, 01 2009.

[36] Pirooz Mohazzabi and Siva P. Shankar. Damping of a simple pendulum due to drag on its string. *Journal of Applied Mathematics and Physics*, 5(1):122–130, 2017.

[37] Maxon Motor. Datasheet for maxon ads 50/10 motorcontroller. https://www.maxonmotor.com/medias/sys_master/root/8796918349854/ADS-50-10-201583-11-EN-281-283.pdf, 2011.

[38] Maxon Motor. Datasheet for maxon 370365 dc motor. https://www.maxonmotor.com/medias/sys_master/root/8825409470494/17-EN-133.pdf, 2017.

[39] Savinay Nagendra, Nikhil Podila, Rashmi Ugarakhod, and Koshy George. Comparison of reinforcement learning algorithms applied to the cart-pole problem. In *Advances in Computing, Communications and Informatics (ICACCI), 2017 International Conference on*, pages 26–32. IEEE, 2017.

[40] R Nave. Harmonic oscillator. http://hyperphysics.phy-astr.gsu.edu/hbase/oscda.html.

[41] Gym OpenAI. Cartpole-v0. https://gym.openai.com/envs/CartPole-v0/.

[42] Tom S. Pedersen. Mechanical modelling of a dc motor. Lecture note at Aalborg University, 2015.

[43] Roberto Petrella, Marco Tursini, Luca Peretti, and Mauro Zigliotto. Speed measurement algorithms for low-resolution incremental encoder equipped drives: a comparative analysis. In *Electrical Machines and Power Electronics, 2007. ACEMP'07. International Aegean Conference on*, pages 780–787. IEEE, 2007.

[44] Lasse Scherffig. Reinforcement learning in motor control. 2002.

[45] David Silver. Course in reinforcement learning. http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html, 2015.

[46] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[47] Sparkfun. Teensy 3,6 specification. https://www.sparkfun.com/products/14057, 2018.

[48] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction. Second edition.* MIT press Cambridge, 2018.

[49] Avago Technologies. Datasheet for heds 5540 optical quadrature encoder. https://www.infineon.com/dgdl/Infineon-Encoder_HEDS-5540-A14-AP-v01_00-EN.pdf?fileId=5546d46147a9c2e40147d3d593970357, 2012.

[50] Teensy. Download and installation of teensyduino. https://www.pjrc.com/teensy/td_download.html, February 2018.

[51] Anna Vasičkaninová, Monika Bakošová, Alojz Mészáros, and Jiří Jaromír Klemeš. Neural network predictive control of a heat exchanger. *Applied Thermal Engineering*, 31(13):2094–2100, 2011.

[52] Rafael Wisniewski. Mechanical systems ii - lagrange mechanics. Lecture note at Aalborg University, September 2016.

A Comments for the upgrade from Arduino Due to Teensy 3,6

As mentioned in section 2.2, the original Arduino Due has been upgraded to a Teensy 3,6, in order to have a faster MCU in the setup, which also includes a floating-point unit. This appendix describes this upgrade, how it is made and what it means for the future usage of the setup.

The Teensy 3,6 is an Arduino IDE compatible MCU, which means its code can be compiled and the MCU can be programmed through the Arduino IDE, as any other Arduino. In order to do this, an expansion for the Arduino IDE, Teensyduino, needs to be installed, as it is not an original Arduino board. This extension is cross-platform and can be downloaded from the Teensy website [50]. Almost all the normal Arduino functions work for the Teensy as well, and no problems have been found during this thesis.

The Teensy has been soldered to an Arduino Mega/Due prototype shield, which means it has the form factor of the original Arduino Due, such that a back configuration can be made within minutes if desired.

In table A.1 the pin connections of the Arduino Due can be seen, as well as the corresponding pins on the Teensy, as it has not been possible to obtain the same pinout. This also means that the pins need to be changed in the libraries written in previous projects, in order for them to work after the Teensy upgrade. However, this change has been carried out for the earlier projects demos, which were deemed working by the supervisor, at the start of this thesis. As it can be seen in the table, the setup is made ready for another pendulum, which is not installed yet, however, these pins have also been routed on the Teensy 3,6 breakout board.

Pin functionality	Arduino Due pin	Teensy 3,6 pin
Motor controller cart enable	48	12
Motor controller pendulum enable	50	11
Motor controller cart current reference	DAC0	DAC0/A21
Motor controller pendulum current reference	DAC1	DAC1/A22
Motor controller extra pendulum current reference	PWM7	7
Decoder parallel data bus	40-47	25-32
Decoder high/low byte data select	35	39
Decoder sled output enable	30	35
Decoder sled reset	28	33
Decoder pendulum output enable	29	34
Decoder pendulum reset	31	36
Decoder extra pendulum output enable	33	38
Decoder extra pendulum reset	32	37
Current measurement cart	A8	A8 / 23
Current measurement pendulum	A9	A9 / 24

Table A.1. Pin-mapping from Arduino Due to Teensy 3,6.

Both the Arduino Due and the Teensy is 3,3 V units, however, the Arduino has a 3,3 V and 5 V regulated power output to supply e.g. shields. This is not the case for the Teensy, which only has a 3,3 V regulated power output and the unregulated voltage from the USB from the programming computer. Both the 3,3 V and 5 V power outputs are utilized in the IPC test setup, and it has thereby been necessary to incorporate an external 5 V power supply into the setup, as the voltage from the USB port were not precise and stable enough. This means, that two main sockets now needs to be plugged in, in order for the setup to work, instead of only one, this is however no problem and does not change the functionality or usability of the test setup.

B Parameter estimation

This appendix describes the parameter estimation of the test setup in the laboratory. The parameter estimation will be carried out in several parts in order to limit the number of parameters to estimate at a time. This is done by estimating the parameters for the pendulum and the cart separately and afterward validating and adjusting the parameters for obtaining a combined model.

Firstly, the cart-related parameters will be estimated, which is the mass, m_c, the Coulomb friction coefficient, $\mathcal{F}_{c,c}$, and the viscous friction coefficient $B_{v,c}$.

Secondly, the parameters related to the pendulum will be estimated, which is the two friction coefficients Coulomb friction, $\mathcal{F}_{c,p}$, and viscous friction, $B_{v,p}$ as the pendulum length, l, and the bob weight, m_b, already are known.

During the estimations for the cart, the pendulum stick is fixed to avoid effects on the cart and the bob itself is detached. When the pendulum is under investigation the cart is fixed, in order to simplify the physics involved and having fewer parameters affecting the measurements.

B.1 Cart parameters

For the cart parameters the estimation is split into two parts. Firstly, the two friction coefficients are estimated, by having the cart run at a constant velocity and use that the applied force here equals the total friction in the system. This is possible because at a constant velocity the inertia will not contribute to the dynamics, as there is no acceleration. This yields the following equation based on the x-component of equation 3.22:

$$\underbrace{(m_c + m_b)\ddot{x}}_{=0} - \underbrace{m_b l \cos(\theta)\ddot{\theta}}_{=0} + \underbrace{m_b l \sin(\theta)\dot{\theta}^2}_{=0} = F_a - F_{B,c} \tag{B.1}$$

$$0 = F_a - B_{v,c}\dot{x} - \mathcal{F}_{c,c}\tanh(k_{tanh}\dot{x}) \tag{B.2}$$

Thereby at a constant velocity, the friction force will be given directly by the applied force. As described in section 3.5, the viscous friction coefficient will be given by the gradient of the friction force and the Coulomb will be given by the crossing with $\dot{x} = 0$. The static friction does not affect this test, as the velocity is not zero or close to.

There has been made 11 tests in each direction in the range of ± 0.06 to ± 1 m/s. It is done in both directions in order to determine whether there is a difference between the friction affecting the positive and negative movement. When the cart reaches a steady state, the mean of the velocity is taken as well as the mean control current to calculate the applied force, as a true constant velocity is impossible to get.

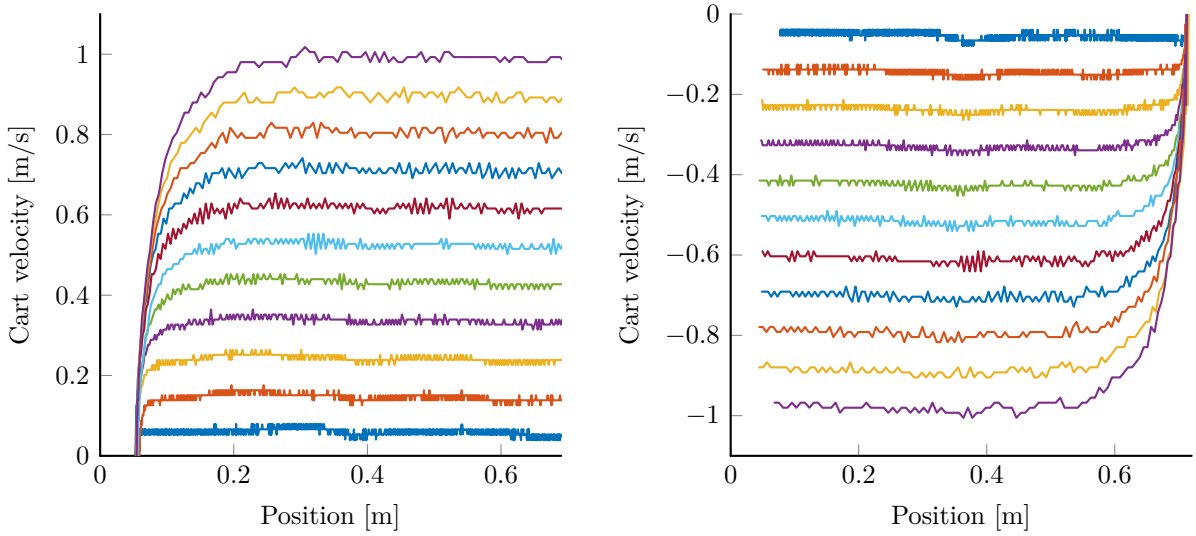
The steady state velocity of the cart is achieved by designing a proportional controller, which fairly quickly reaches a steady state velocity. This controller is hand-tuned to get a decent response. A proportional controller is chosen due to its simplicity and as the steady state error does not matter for the test. The controller is implemented as the following, where $\dot{x}$ is approximated over

two samples, as described in section 4.3.

$$i_a = (v_{ref} - \dot{x})K_p \tag{B.3}$$

The measurements from the tests can be seen in figure B.1, where B.1a represents the positive movement and B.1b the negative movement. In the plot the velocity is compared to the position, because it might indicate some varying friction along the rail.

As seen in figure B.1 the velocities reaches a sufficient steady state velocity. An average friction is in this test found, which is deemed ok, as the controller is expected to be able to deal with uncertainties.



(a) Velocity steps in positive direction.

(b) Velocity steps in negative direction.

Figure B.1. Illustrations of the friction forces acting on the mechanical system.

Two of the related control signals can be seen in figure B.2. These signals are very jumpy due to the quantization of the velocity estimate as described in section 4.3 and the high gain in the controller.

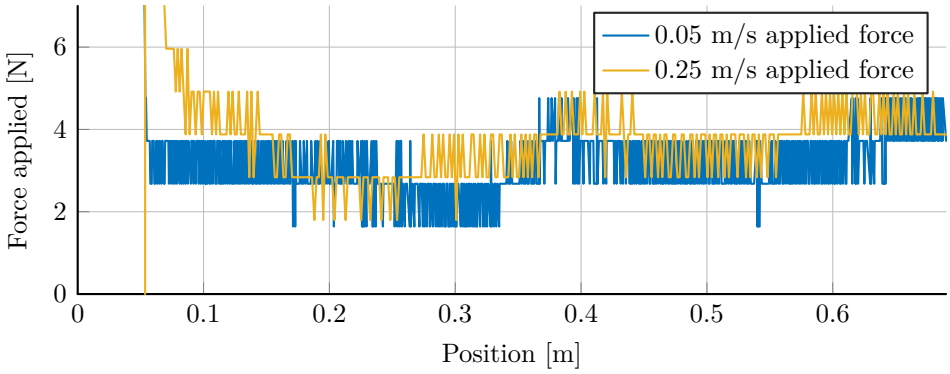


Figure B.2. Two of the control signals for the positive direction.

Due to its slightly varying velocity, it is chosen to use the mean of the velocities and control signals in order to approximate a constant velocity at a certain applied force. The velocities are expected to vary due to different frictions along the rail, which can be felt when manually moving the cart along the rail. The mean is taken from when the velocity has settled reasonably. This yields the following velocities and forces, which corresponds to the frictions.

Velocity [m/s]	-0.980	-0.888	-0.794	-0.702	-0.610	-0.516	-0.423	-0.330
Friction [N]	-5.613	-5.407	-5.252	-5.015	-4.762	-4.620	-4.335	-4.250
Velocity [m/s]	-0.237	-0.144	-0.053	0.059	0.146	0.241	0.336	
Friction [N]	-4.071	-3.821	-3.709	3.201	3.654	3.658	3.634	
Velocity [m/s]	0.431	0.525	0.618	0.712	0.804	0.896	0.989	
Friction [N]	3.713	3.762	4.055	4.182	4.371	4.766	4.919	

Table B.1. Velocities and corresponding friction forces.

This data is used to regress a model for the positive and negative directions respectively. The result of that can be seen in figure B.3, where both the mean of the velocities and the regressed model can be seen.

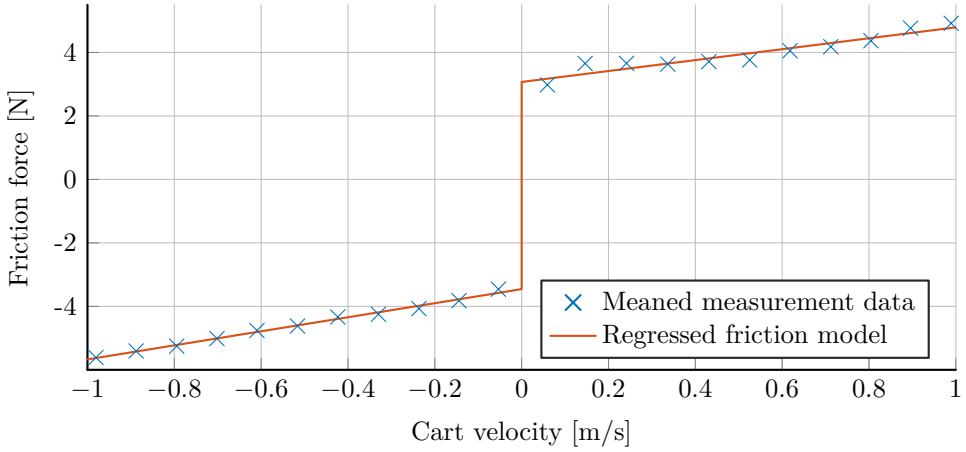


Figure B.3. Friction forces present at constant velocity.

In the figure it can from the regression be seen, that the Coulomb friction is different from whether the cart moves in the positive or negative direction. It can also be observed that the viscous friction varies depending on the direction. The direction dependent frictions are thereby

$$B_{v,c}(\dot{x}) = \begin{cases} 1.614 \ \frac{\text{N}}{\text{m/s}} & \text{if } \dot{x} > 0 \\ 2.091 \ \frac{\text{N}}{\text{m/s}} & \text{if } \dot{x} < 0 \end{cases}, \tag{B.4}$$

$$\mathcal{F}_{c,c}(\dot{x}) = \begin{cases} 3.147 \text{ N} & \text{if } \dot{x} > 0 \\ 3.544 \text{ N} & \text{if } \dot{x} < 0 \end{cases}. \tag{B.5}$$

It can be seen that the friction force increases affinely according to the velocity as expected. It has however been observed that the system has varying frictions along the rail, which was also found by the previous group working on the test setup. Despite this, the estimated parameters presented are deemed a fitting average and the controller should be robust enough to handle the rest of the uncertainties.

In order to determine the static friction of the cart, a force equaling the Coulomb friction is applied and it is tested throughout the rail length, whether the cart is just about to or moves very slowly. This is not the case, as sometimes the cart accelerates and other places it is stuck. However, this is deemed because of the changing frictions as previously mentioned, instead of due to static friction. Based on this, the static friction is deemed so small, that it can be neglected for the cart in this thesis.

With the parameters for the friction in the cart determined, the mass of the cart, m_c, is to be estimated. This is done by applying a force, which causes the cart to accelerate. This test is performed multiple times at different applied forces and each of these step responses are used to estimate the mass.

The dynamics of the system will be given by the x-component of equation 3.22, where the pendulum does not contribute, such that the following equation is obtained, where F_{leftover} is the leftover force after the Coulomb friction has be subtracted from the input force.

$$m_c\ddot{x} = F_a - B_{v,c}\dot{x} - \mathcal{F}_{c,c} \tag{B.6}$$

$$\ddot{x} = \frac{1}{m_c}F_{\text{leftover}} - \frac{B_{v,c}}{m_c}\dot{x} \tag{B.7}$$

The mass is thereby estimated using the dynamic equation B.7, using the MATLAB function *greyest*, which lets the user estimate specific parameters of a state space system and specify the rest. The system is therefore specified such that only the mass, m_c, is estimated.

Six tests have been carried out, where the position is plotted against the time in figure B.4. These tests yielded the estimations seen in table B.2.

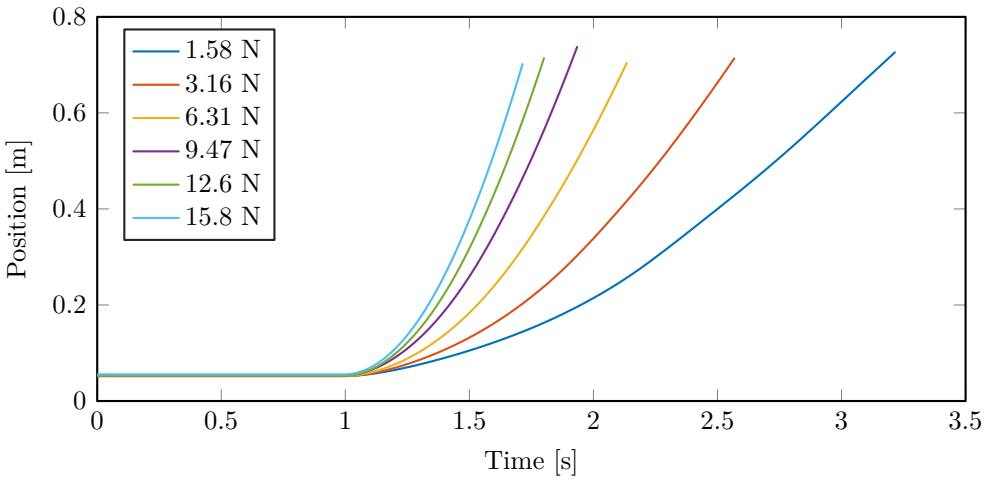


Figure B.4. Position of the cart given different "leftover" forces.

$F_{\mathbf{leftover}}$	1.58 N	3.16 N	6.31 N	9.47 N	12.6 N	15.8 N	**Mean**
$\mathbf{m_c}$	4.567	5.069	5.583	5.607	5.654	5.805	5.381

Table B.2. Results from the estimations of the mass of the cart for different "leftover" forces, as well as the mean.

A tendency can be seen, as the mass seems to be estimated higher, the higher the force used. As equation B.7 is as expected for a second order system for a dampened mass, the tendency must come from the parameters used in the equation, in this case, the estimated frictions and the motor constant, K_t, which was taken from the datasheet and not measured. A similar test has been conducted where the motor is not used, but objects with known weight and the gravitational acceleration are used to accelerate the cart. These tests have shown similar tendencies, despite the motor and motor controller not as an error source. It is thereby expected to be due to the frictions, which could be estimated wrong, as the applied force in these tests came from the motor, whose motor constant could deviate from the value in the datasheet. E.g. a too big viscous friction could imply the seen tendency in the mass, as too much friction is modelled in the system, which scales linearly with the velocity, yielding a in higher cart mass for higher forces in the estimation. However, due to time constraints, no further investigation in this matter has been done. The mean of the estimated masses is thereby used, and the controller is expected to be able to handle the uncertainties.

Thereby, the parameters of the cart have been estimated and they will be combined for a sub-validation, to check whether the parameters fit when combined, without influence from the pendulum.

The validation is carried out by applying a step in the positive direction followed by a step in the negative direction. This should show the dynamics of the cart. The results can be seen in figure B.5, where it can be seen that the simulated response does not fit very well. Multiple identical tests have been carried out in order to check whether the system is consistent.

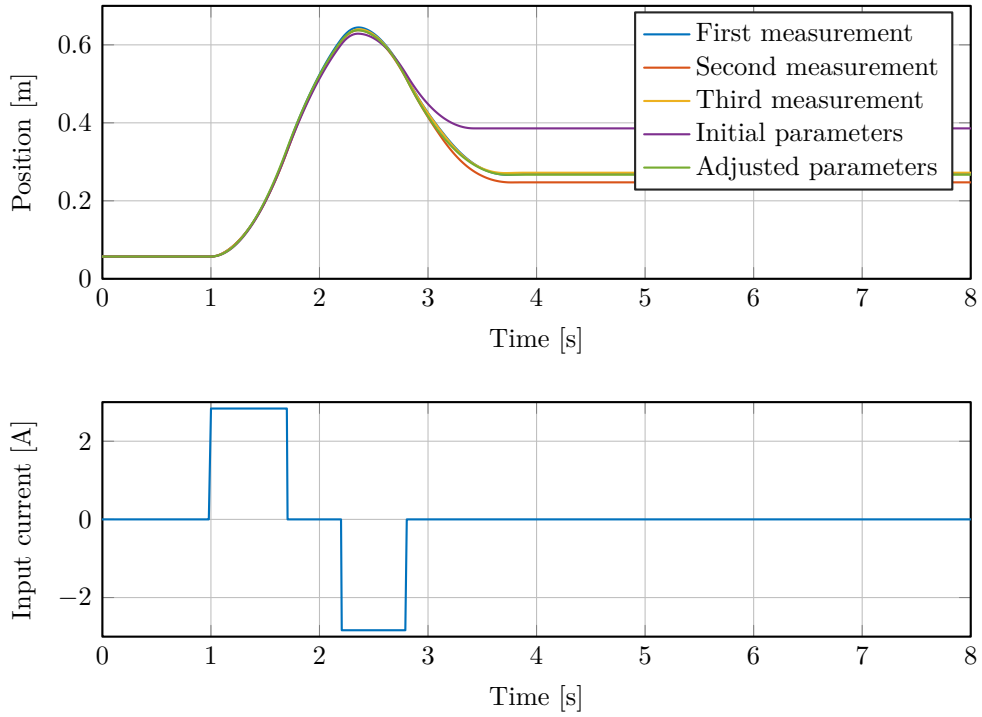


Figure B.5. Measurements vs simulated results with initial and adjusted parameters.

It can be seen that the tests were fairly consistent and the parameters have therefore been adjusted to fit in between those. This gives the following adjusted parameters for the cart:

Parameter	m_c	$B_{v,c}$	$\mathcal{F}_{c,c}$
Value	5.273	1.937 if $\dot{x} > 0$, 1.422 if $\dot{x} < 0$	3.021 if $\dot{x} > 0$, 2.746 if $\dot{x} < 0$

Table B.3. Adjusted cart parameters.

These parameters are again simulated against new tests, in order to ensure the parameters are not changed to fit the above specific test but in general yields worse results. The further sub-validations all looks satisfactory and the corrected parameters are thereby deemed correct for the sub-validation. One of the test-sequences is shown in figure B.6, which shows that the measurement from the test setup varies slightly. This has been observed to be a consistent issue when running at lower velocities, and is expected to be because of the varying frictions along the rail and slightly varying start positions of the cart might then mean different frictions along the test. Despite this, the simulation is deemed to fit quite well.

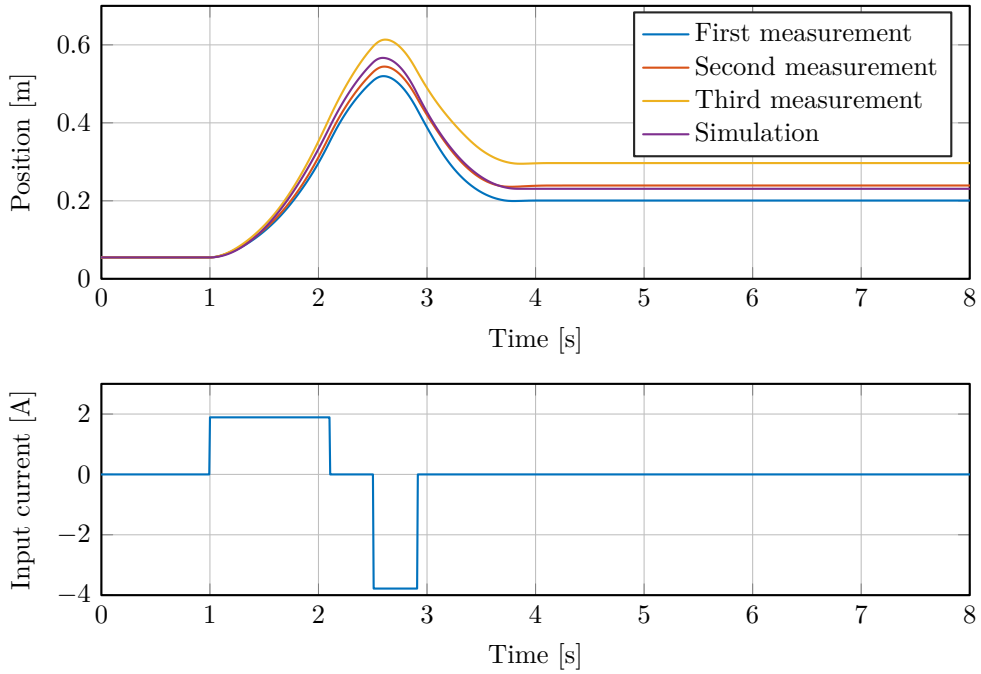


Figure B.6. Validation of cart parameters.

The frictions and mass of the cart have been determined and therefore only the friction parameters related to the pendulum need to be estimated, which will be presented in the following section.

B.2 Pendulum parameters

For the pendulum two parameters need to be estimated, the Coulomb friction coefficient, $\mathcal{F}_{c,p}$, and the viscous friction coefficient, $B_{v,p}$. This is done in two steps where the Coulomb friction first is determined and afterward the viscous friction.

The Coulomb friction coefficient for the pendulum is determined by attaching a small bob weight to the pendulum (26 g and 50 g), and then lifting the pendulum to the largest angle, at which the pendulum does not move. In such a case the following formula can be used to calculate the Coulomb torque, τ_c,

$$\tau_c = |g m_b l \sin(\theta)|. \tag{B.8}$$

The absolute value is used as only the coefficient should be estimated. It should be noted, that the found friction is the sum of Coulomb and static friction. However, as it is observed for the cart that barely any static friction is present despite having the pulley and belt attached, it is thereby deemed that no static friction is present in the pendulum, as the same motor type is used as its rotational joint.

This test has been done several times with both weights, where the following measurements have been made, where $m_b = 26$ g.

θ_{26g} [deg]	-2.34	-2.7	2.34	2.34	-2.34	-2.16	-2.34	2.7
$\tau_{c,26g}$ [mNm]	3.0	3.5	3.0	3.0	3.0	2.8	3.0	3.5

Table B.4. Measured angles for estimating the Coulomb friction with $m_b = 26$ g and $l = 0.2875$ m.

In order to use this, the mean is taken such that an estimate of the Coulomb friction with a mass of $m_b = 26$ g yielding

$$\mathcal{F}_{c,p,26g} = \bar{\tau}_{c,26g} = 3.1 \cdot 10^{-3}. \tag{B.9}$$

The same has been done for the case with $m_b = 50$ g, where the measurements is seen in the following table.

θ_{50g} [deg]	1.44	-1.62	1.44	1.44	-1.44
$\tau_{c,50g}$ [mNm]	3.6	4.1	3.6	3.6	3.6

Table B.5. Measured angles for estimating the Coulomb friction with $m_b = 50$ g and $l = 0.2925$ m.

The mean is again applied to find the friction coefficient at the given mass,

$$\mathcal{F}_{c,p,50g} = \bar{\tau}_{c,50g} = 3.7 \cdot 10^{-3}. \tag{B.10}$$

The mean of the two friction coefficients is taken to find a final Coulomb friction

$$\mathcal{F}_{c,p} = \frac{1}{2}(\mathcal{F}_{c,p,26g} + \mathcal{F}_{c,p,50g}) = 3.4 \cdot 10^{-3} \text{ Nm}. \tag{B.11}$$

Thereby the Coulomb friction coefficient has been determined, and the only parameter left to determine is the viscous friction coefficient for the pendulum. The viscous friction is determined by releasing the pendulum in an initial angle and letting it swing down and settle. The amplitude should then follow the exponential decay as an underdamped harmonic oscillator [40]. An underdamped harmonic oscillator has a peak amplitude defined from the exponential decaying expression

$$\theta_{peak} = \theta_0 \mathrm{e}^{-\frac{b}{2m}t}. \tag{B.12}$$

Where:

θ_{peak}	is the peak position of the mass	[rad]
θ_0	is the initial position of the mass	[rad]
b	is the viscous friction coefficient	$[\frac{\text{N}}{\text{rad/s}}]$
m	is the weight of the mass	[kg]

The friction is thereby estimated, by estimating an exponential decay of the peaks by the oscillations of the pendulum. Three similar test has been conducted, where one of them can be seen in figure B.7. The first swing of the pendulum has been removed, to avoid the noise from the release of the pendulum. The nominal weight of $m_p = 201$ g has been used for this test.

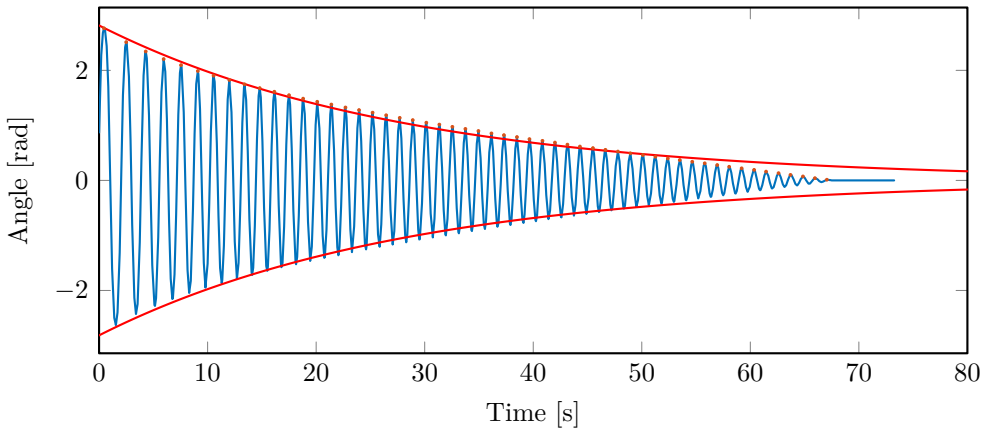


Figure B.7. Exponential decay of the pendulum, where the angle is offset to zero, despite the pendulum is hanging down, as this eases the estimation.

For each test the peaks are found, using the MATLAB function *findpeaks*, which finds all local maxima, and the exponential function is then estimated using the function *fit*, with the argument *'exp1'*. This then estimates the exponential function

$$f(x) = \alpha e^{\beta x}. \tag{B.13}$$

As the MATLAB function fits without offset, the measurement is offset to swing around zero instead of π as the test is conducted. It should be noted, that the estimation does not fit well in the end at small angles, which is expected to be because the underdamped harmonic oscillator does not include Coulomb friction. It is therefore expected that it is necessary to tune the model parameters such that the Coulomb and viscous friction fit together afterward.

It has been seen that all three tests gave similar results, with viscous friction coefficients as seen in table B.6.

Test	1	2	3	**Mean**
$\mathbf{B_{v,p}}$	0.0142	0.0142	0.0144	0.0143

Table B.6. Estimation of viscous friction coefficients for the pendulum.

The mean of the estimations in table B.6 are used as the viscous coefficient for the pendulum. Thereby the parameters for the pendulum has been estimated.

A sub-validation is therefore performed for the pendulum, in order to check how well the model fits and to tune the parameters, as this is expected necessary, due to the method for estimating

the viscous friction, which did not consider the contribution from the Coulomb friction. The test is similar to the one used for estimating the viscous friction and can be seen in figure B.8.

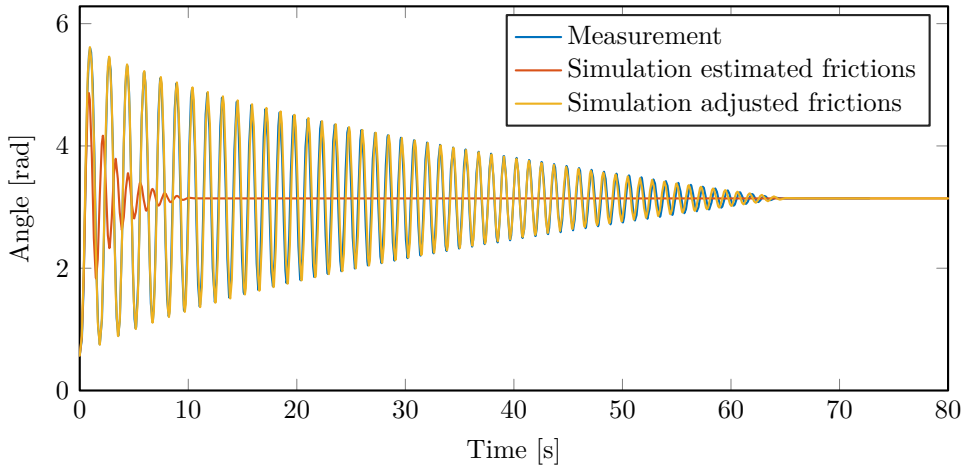


Figure B.8. Verification and adjustment of model for the pendulum parameters.

It can easily be seen, that the estimated friction does not fit very well, when combined, thereby the parameters have been adjusted by hand, in order to gain the fitment seen in figure B.8.

The fitting has been done by looking at the plot and if the amplitude is too high in the end, then more Coulomb is needed and vice versa. The adjusted parameters can be seen in table B.7, where all the parameters found for the entire IPC system can be seen as well.

Parameter	$B_{v,c}$	$\mathcal{F}_{c,c}$	m_c	$B_{v,p}$	$\mathcal{F}_{c,p}$
Value	$1.937\ \frac{\text{N}}{\text{m/s}}$ if $\dot{x} > 0$, $1.422\ \frac{\text{N}}{\text{m/s}}$ if $\dot{x} < 0$	3.021 N if $\dot{x} > 0$, 2.746 N if $\dot{x} < 0$	5.273 kg	$0.0004\ \frac{\text{Nm}}{\text{rad/s}}$	0.004 Nm

Table B.7. Final parameters for the model of the IPC system.

For comparison the previous project group, [11] (page 26), found the following coefficients:

Parameter	$B_{v,c}$	$\mathcal{F}_{c,c}$	m_c	$B_{v,p}$	$\mathcal{F}_{c,p}$
Value	$0\ \frac{\text{N}}{\text{m/s}}$	3.4 N if $\dot{x} > 0$, 2.45 N if $\dot{x} < 0$	4.75 kg	$0.00067\ \frac{\text{Nm}}{\text{rad/s}}$	0.0038 Nm

Table B.8. Parameters from group 17gr1030, [11], page 26.

It can be seen that the mass, m_c, the cart Coulomb friction, $\mathcal{F}_{c,c}$, and the pendulum frictions are all relatively close, thereby verifying that the determined parameters seems plausible. The viscous friction was discarded for the previous project, however in this thesis the belt has been tightened, leading to more friction. The belt tension is a tradeoff between decreased backlash and increased

friction, and as a lot backlash was present before in order to lower the friction, the belt has for this thesis been tightened a bit. Thereby the viscous friction in the cart for this thesis is larger than previously.

C Friction compensation analysis

In this appendix, the system will be tested in order to analyze whether a friction compensation based on just the sign of the control signal, as in equation 4.1 is good enough, or the more advanced method in equation 4.4 is necessary. The advanced method will be necessary if the dynamics of the system causes a significant delay between a change in input direction and a change of system movement direction.

In this appendix, tests are carried out on the IPC system in order to investigate whether the dynamics cause such a delay. In the tests, an input is applied as a square wave, such that the applied current changes equally in both directions. Several tests have been carried out, with period times of 100 ms, 200 ms, 600 ms and 1 s, and currents of $\pm 1,5$ and $\pm 4,5$ A. In these tests the pendulum has been fixed, in order to remove the influence of the bob's inertia. One such tests is seen in figure C.1 with a square wave period of 600 ms and $\pm 4,5$ A input, where the position response can be seen. It can here be seen, as described in chapter 3, that the friction varies along the rail, as the cart does not oscillate back and forth the same place, but drifts until a place with similar friction in both directions is found. This however also means, that the compensation will not be perfect at all places and an average will be used.

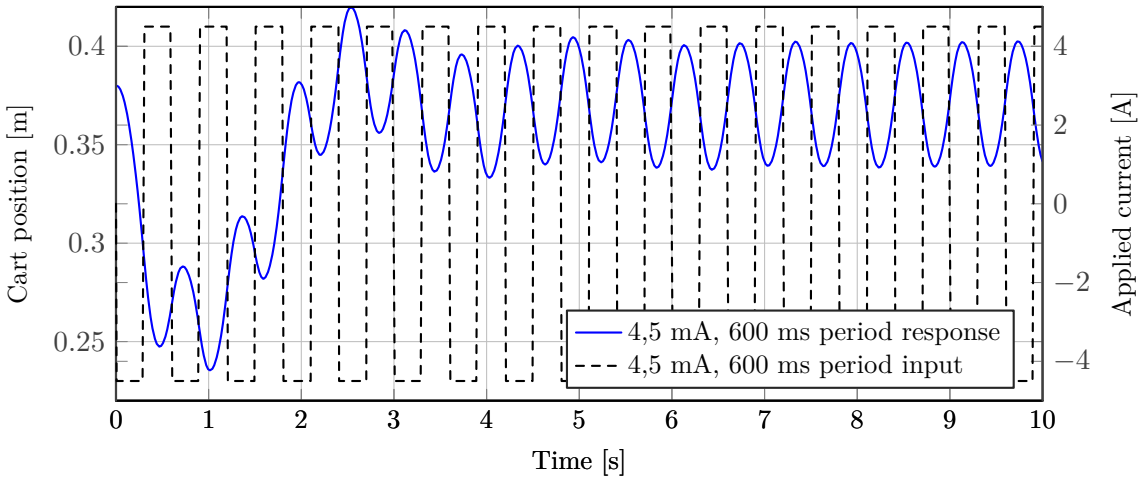


Figure C.1. System response given square wave input of ±4,5 A, with 600 ms period.

In order to investigate which compensation is needed, the velocity is estimated directly through a differentiation. This can be seen in figure C.2, where the first three seconds are plotted. Both the applied input and the velocity is shown in the same plot in order to visualize the delay from when the input changes direction, compared to the sign-change in velocity.

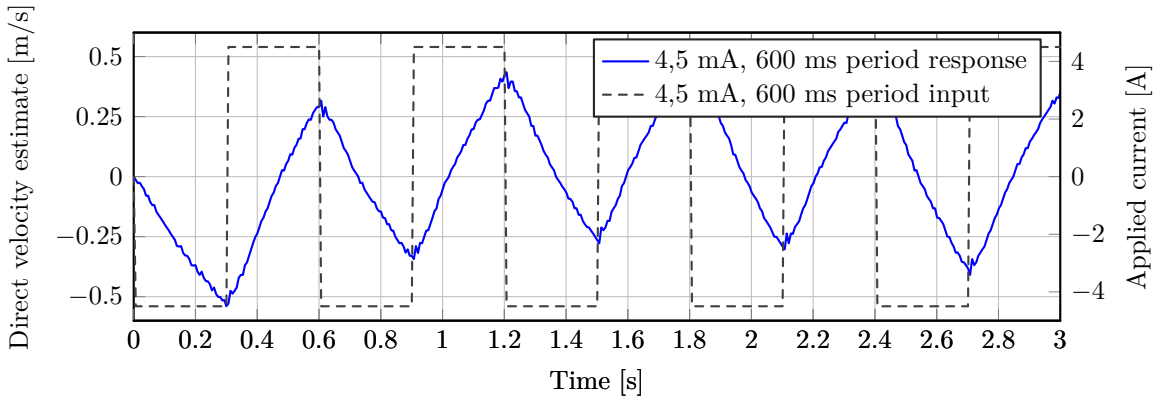


Figure C.2. System velocity given square wave input of 4,5 A, with 600 ms period.

It can be seen that the cart starts to decelerate immediately, when the input changes direction, however the cart is still moving in the same direction several samples after the change. The same is seen for the tests with periods of 100 ms, 200 ms and 1 s.

In order to visualize how well the friction compensation based solely on the input will perform, graphs for the tests has been made which shows when the compensation would perform incorrectly. This is done with the following formula, where a zero represents a true compensation and a one represents a wrong compensation.

$$\text{Wrong compensation identifier} = \begin{cases} 1 & \text{if } \text{sign}(i_a) \neq \text{sign}\left(\dot{x}\right) \\ 0 & \text{if } \text{sign}(i_a) = \text{sign}\left(\dot{x}\right) \end{cases} \tag{C.1}$$

This is done for all the tests and the results can be seen in the following, where only cuts of the data are presented for convenience and overview, however the same behavior is seen in all data. It should be noted that the velocities are based on direct derivative estimations on the position of the cart, and can thereby indicate zero velocity at low velocities, causing minor errors in the analysis. This is however not on a scale which will affect the overall result. Firstly the 100 ms period tests are presented.

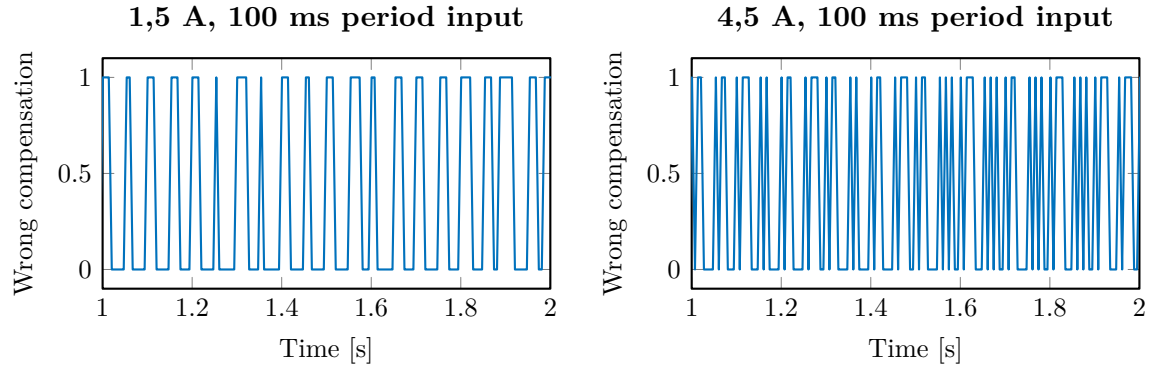


Figure C.3. Indication of time where the compensation is wrong, given compensator which is only input dependable.

For these tests, the compensation only depending on the input, gives a theoretically wrong compensation ≈ 39 % of the time, for the ±1,5 A pulses and ≈ 43 % of the time, for the ±4,5 A pulses, for the 1 second window plotted. Next the 200 ms period tests are shown.

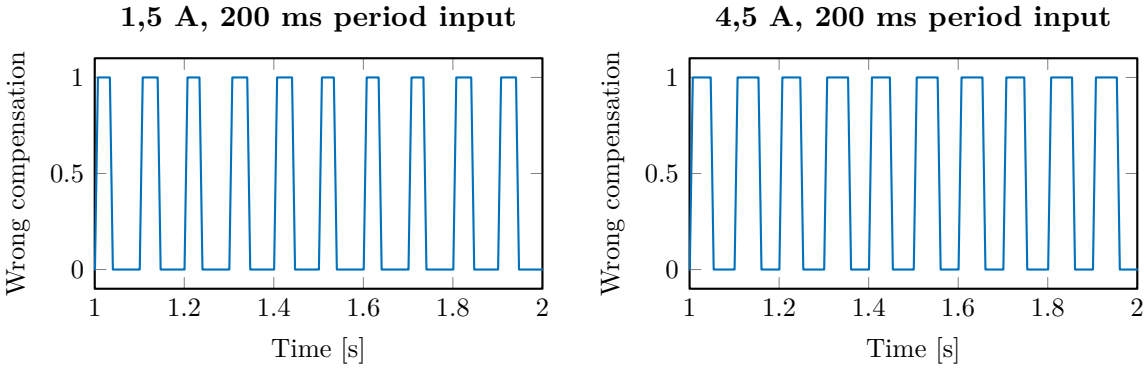


Figure C.4. Indication of time where the compensation is wrong, given compensator which is only input dependable.

Here the compensation is wrong for ≈ 37 % of the time, for the ±1,5 A pulses and ≈ 51 % of the time, for the ±4,5 A pulses, for the 1 second window plotted. Next is the 600 ms period tests.

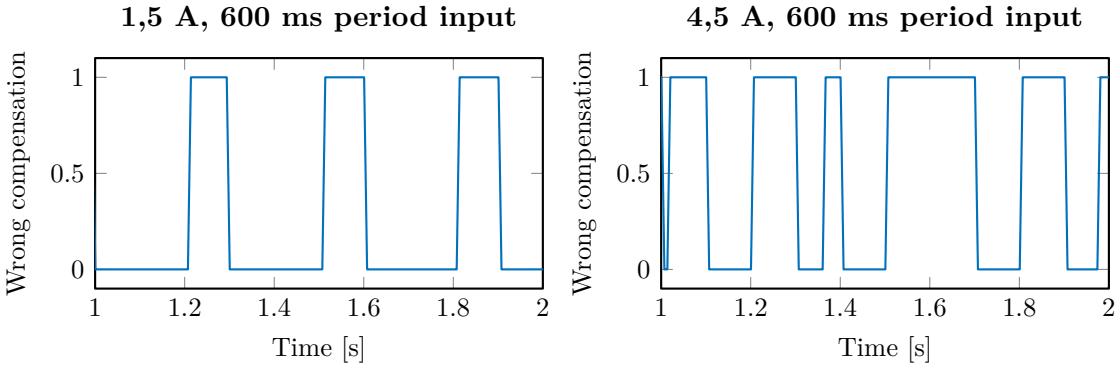


Figure C.5. Indication of time where the compensation is wrong, given compensator which is only input dependable.

Here the compensation is wrong for ≈ 28 % of the time, for the ±1,5 A pulses and ≈ 55 % of the time, for the ±4,5 A pulses, for the 1 second window plotted. Lastly is the 1000 ms period tests.

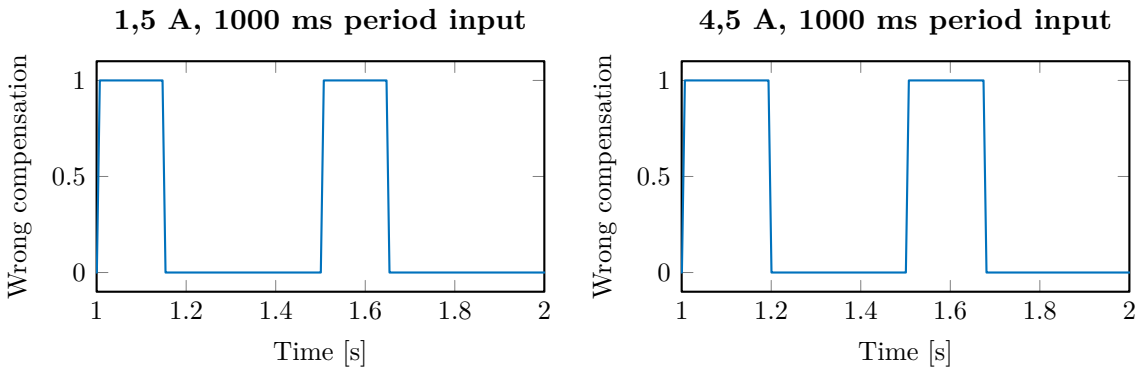


Figure C.6. Indication of time where the compensation is wrong, given compensator which is only input dependable.

Here the compensation is wrong for ≈ 29 % of the time, for the $\pm 1{,}5$ A pulses and ≈ 37 % of the time, for the $\pm 4{,}5$ A pulses, for the 1 second window plotted.

Based on these results it is concluded that the delay in dynamics are significant enough that the more advanced compensation, based on both velocity estimate and input should be used. This is also deemed even more appropriate, since the non-linear controller used in chapter 5 is a switching controller. This means that the input will change direction almost every sample, and still move the cart to one direction.

D Derivations of equations

This appendix is for longer derivations of equations, which is inappropriate for the main report.

D.1 Linearization of the system for the linear Kalman filter

In this appendix, the system model is linearized, such that it can be used for the linear Kalman filter design. The linearization is done via a Taylor expansion and is done in the operating point $\bar{\mathbf{x}} = \begin{bmatrix} 0 & \pi & 0 & 0 \end{bmatrix}^T$ and $\bar{i_a} = 0$, to find a small signal model. The Taylor expansion is given by,

$$\frac{d}{dt}x = f(x,u) = f(\bar{x},\bar{u}) + \frac{\partial f}{\partial x}\Big|_{x=\bar{x},u=\bar{u}}\hat{x} + \frac{\partial f}{\partial u}\Big|_{x=\bar{x},u=\bar{u}}\hat{u} + \textit{higher order terms.} \tag{D.1}$$

Where:

$\bar{x}$	is the operating point for the state	[1]
$\bar{u}$	is the operating point for the input	[1]
$\hat{x}$	is the deviation from the operating point for the state	[1]
$\hat{u}$	is the deviation from the operating point for the input	[1]

To get a linear model, only the first order Taylor expansion is used, and the higher order terms are discarded, thereby giving the following linearization method for the model,

$$\frac{d}{dt}x = f(x,u) \approx f(\bar{x},\bar{u}) + \frac{\partial f}{\partial x}\Big|_{x=\bar{x},u=\bar{u}}(x-\bar{x}) + \frac{\partial f}{\partial u}\Big|_{x=\bar{x},u=\bar{u}}(u-\bar{u}). \tag{D.2}$$

This corresponds to using the gradient of the system in the operating point and thereby having a linear model. The constant term corresponds to the origin offset in the coordinate transformation from big signal to small signal.

For convenience, the model given by equations 3.39 to 3.44, is presented here again, just without the Coulomb friction in the cart due to the compensation in section 4.1,

$$\begin{aligned} \dot{\mathbf{x}} &= \begin{bmatrix} \dot{x} \\ \dot{\theta} \\ \ddot{x} \\ \ddot{\theta} \end{bmatrix} = \underbrace{\begin{bmatrix} \dot{x} \\ \dot{\theta} \\ f_1(\theta)\ F_B(\dot{x},\dot{\theta}) + f_2(\theta,\dot{\theta}) \end{bmatrix}}_{f(\mathbf{x})} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ g_1(\theta) \end{bmatrix}}_{g(\mathbf{x})} i_a, \\ y &= \begin{bmatrix} x \\ \theta \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}}_{h(\mathbf{x})} \mathbf{x}, \end{aligned} \tag{D.3}$$

where

$$f_1(\theta) = -\begin{bmatrix} \left(m_c + m_b \sin^2(\theta)\right)^{-1} & \cos(\theta)\left(l\left(m_c + m_b \sin^2(\theta)\right)\right)^{-1} \\ \cos(\theta)\left(l\left(m_c + m_b \sin^2(\theta)\right)\right)^{-1} & \left(m_c + m_b\right)\left(m_b l^2\left(m_c + m_b \sin^2(\theta)\right)\right)^{-1} \end{bmatrix}, \tag{D.4}$$

$$F_B(\dot{x}, \dot{\theta}) = \begin{bmatrix} B_{v,c}\dot{x} \\ B_{v,p}\dot{\theta} + \mathcal{F}_{c,p}\ \tanh\left(k_{tanh}\dot{\theta}\right) \end{bmatrix}, \tag{D.5}$$

$$f_2(\theta, \dot{\theta}) = -\begin{bmatrix} \frac{m_b l \sin(\theta)\dot{\theta}^2 - g m_b \sin(\theta)\cos(\theta)}{m_c + m_b \sin^2(\theta)} \\ \frac{m_b l \sin(\theta)\cos(\theta)\dot{\theta}^2 - g \sin(\theta)(m_c + m_b)}{l\left(m_c + m_b \sin^2(\theta)\right)} \end{bmatrix}, \tag{D.6}$$

$$g_1(\theta) = \frac{K_t}{r}\begin{bmatrix} \left(m_c + m_b \sin^2(\theta)\right)^{-1} \\ \cos(\theta)\left(l\left(m_c + m_b \sin^2(\theta)\right)\right)^{-1} \end{bmatrix}. \tag{D.7}$$

It can be seen in the model, that it consists of two non-linear functions, $f(\mathbf{x})$ and $g(\mathbf{x})i_a$, which has to be linearized. Firstly the term $f_1(\theta)F_B(\dot{x}, \dot{\theta})$ is to be linearized, which is done in the following. Only the partial derivatives where the function is depending on the derivating variable is presented, as the rest will equal zero.

$$\begin{aligned} f_1(\theta)F_B(\dot{x}, \dot{\theta}) &\approx \underbrace{f_1(0)F_B(0,0)}_{=0} + \underbrace{\frac{\partial f_1 F_B}{\partial x_2}\Big|_{x=\bar{x},u=\bar{u}}}_{=0}\hat{x}_2 + \frac{\partial f_1 F_B}{\partial x_3}\Big|_{x=\bar{x},u=\bar{u}}\hat{x}_3 + \frac{\partial f_1 F_B}{\partial x_4}\Big|_{x=\bar{x},u=\bar{u}}\hat{x}_4 \\ &= -\begin{bmatrix} \frac{1}{m_c + m_b \sin^2(\bar{x}_2)}B_{v,c}\hat{x}_3 + \frac{\cos(\bar{x}_2)}{l(m_c + m_b \sin^2(\bar{x}_2))}\left(B_{v,p} + F_{c,p}k_{tanh}\mathrm{sech}^2(k_{tanh}\bar{x}_4)\right)\hat{x}_4 \\ \frac{\cos(\bar{x}_2)}{l(m_c + m_b \sin^2(\bar{x}_2))}B_{v,c}\hat{x}_3 + \frac{m_b + m_c}{m_b l^2(m_c + m_b \sin^2(\bar{x}_2))}\left(B_{v,p} + F_{c,p}k_{tanh}\mathrm{sech}^2(k_{tanh}\bar{x}_4)\right)\hat{x}_4 \end{bmatrix} \\ &= -\begin{bmatrix} \frac{1}{m_c}B_{v,c}\hat{x}_3 - \frac{1}{l m_c}\left(B_{v,p} + F_{c,p}k_{tanh}\right)\hat{x}_4 \\ \frac{-1}{l m_c}B_{v,c}\hat{x}_3 + \frac{m_b + m_c}{m_b l^2 m_c}\left(B_{v,p} + F_{c,p}k_{tanh}\right)\hat{x}_4 \end{bmatrix}. \end{aligned} \tag{D.8}$$

The next step is the linearization of the function $f_2(\theta, \dot{\theta})$,

$$\begin{aligned} f_2(\theta, \dot{\theta}) &\approx \underbrace{f_2(0,0)}_{=0} + \frac{\partial f_2}{\partial x_2}\Big|_{x=\bar{x},u=\bar{u}}\hat{x}_2 + \underbrace{\frac{\partial f_2}{\partial x_4}\Big|_{x=\bar{x},u=\bar{u}}}_{=0}\hat{x}_4 \\ &= -\begin{bmatrix} \frac{\left(m_b l \cos(\bar{x}_2)\bar{x}_4^2 - g m_b(\cos^2(\bar{x}_2) - \sin^2(\bar{x}_2))\right)\left(m_c + m_b \sin^2(\bar{x}_2)\right) - \left(m_b l \sin(\bar{x}_2)\bar{x}_4^2 - g m_b \sin(\bar{x}_2)\cos(\bar{x}_2)\right)m_b \sin(2\bar{x}_2)}{(m_c + m_b \sin^2(\bar{x}_2))^2}\hat{x}_2 \\ \frac{-\left(g\cos(\bar{x}_2)l(m_c + m_b \sin^2(\bar{x}_2)) - g\sin(\bar{x}_2)l m_b \sin(2\bar{x}_2)\right)}{(l(m_c + m_b \sin^2(\bar{x}_2)))^2}\hat{x}_2 \end{bmatrix} \\ &= \begin{bmatrix} \frac{g m_b}{m_c} \\ \frac{-g}{l m_c} \end{bmatrix}\hat{x}_2. \end{aligned} \tag{D.9}$$

And lastly the function $g_1(\theta)i_a$ is to be linearized

$$
\begin{aligned}
g_1(\theta)i_a &\approx \underbrace{g_1(0)\cdot 0}_{=0} + \frac{\partial g_1 i_a}{\partial x_2}\Big|_{x=\bar{x},u=\bar{u}}\hat{x}_2 + \frac{\partial g_1 i_a}{\partial i_a}\Big|_{x=\bar{x},u=\bar{u}}\hat{i}_a \\
&= \frac{K_t}{r}\begin{bmatrix} \frac{-m_b\sin(2\bar{x}_2)}{(m_c+m_b\sin^2(\bar{x}_2))^2}\hat{x}_2 + \frac{1}{m_c+m_b\sin^2(\bar{x}_2)}\hat{i}_a \\ \frac{-sin(\bar{x}_2)\big(l(m_c+m_b\sin^2(\bar{x}_2))\big)-\cos(\bar{x}_2)lm_b\sin(2\bar{x}_2)}{(l(m_c+m_b\sin^2(\bar{x}_2)))^2}\hat{x}_2 + \frac{cos(\bar{x}_2)}{l(m_c+m_b\sin^2(\bar{x}_2))}\hat{i}_a \end{bmatrix} \\
&= \frac{K_t}{r}\begin{bmatrix} \frac{1}{m_c} \\ \frac{-1}{lm_c} \end{bmatrix}\hat{i}_a.
\end{aligned}
\tag{D.10}
$$

Thereby the linearizations in equations D.8 to D.10 can be substituted into equations D.3 yielding the small signal linear state space model

$$
\begin{aligned}
\dot{\hat{\mathbf{x}}} &= \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{gm_b}{m_c} & \frac{-B_{v,c}}{m_c} & \frac{1}{lm_c}\left(B_{v,p}+F_{c,p}k_{tanh}\right) \\ 0 & \frac{-g}{m_c} & \frac{B_{v,c}}{lm_c} & -\frac{m_b+m_c}{m_b l^2 m_c}\left(B_{v,p}+F_{c,p}k_{tanh}\right) \end{bmatrix}\hat{\mathbf{x}} + \begin{bmatrix} 0 \\ 0 \\ \frac{K_t}{rm_c} \\ \frac{-K_t}{rlm_c} \end{bmatrix}\hat{i}_a, \\
y &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}\mathbf{x}.
\end{aligned}
\tag{D.11}
$$

The system has thereby been linearized in the point $\bar{\mathbf{x}} = \begin{bmatrix} 0 & \pi & 0 & 0 \end{bmatrix}^T$ and $\bar{i}_a = 0$, and can be used for the design of the linear Kalman filter.

D.2 Derivation of Jacobians for the EKF

In this appendix, the Jacobians used for the EKF are derived. The derivation is based on equations 3.39 to 3.44, and they are repeated for here for convenience, where the Coulomb friction in the cart is removed, due to the compensation in section 4.1.

$$
\begin{aligned}
\dot{\mathbf{x}} &= \begin{bmatrix} \dot{x} \\ \dot{\theta} \\ \ddot{x} \\ \ddot{\theta} \end{bmatrix} = \underbrace{\begin{bmatrix} x_3 \\ x_4 \\ f_1(x_2)\ F_B(x_3,x_4)+f_2(x_2,x_4) \end{bmatrix}}_{f(\mathbf{x})} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ g_1(x_2) \end{bmatrix}}_{g(\mathbf{x})} i_a \\
y &= \begin{bmatrix} x \\ \theta \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}}_{h(\mathbf{x})}\mathbf{x}
\end{aligned}
\tag{D.12}
$$

where

$$f_1(x_2) = -\begin{bmatrix} \left(m_c + m_b \sin^2(x_2)\right)^{-1} & \cos(x_2)\left(l\left(m_c + m_b \sin^2(x_2)\right)\right)^{-1} \\ \cos(x_2)\left(l\left(m_c + m_b \sin^2(x_2)\right)\right)^{-1} & (m_c + m_b)\left(m_b l^2\left(m_c + m_b \sin^2(x_2)\right)\right)^{-1} \end{bmatrix}, \tag{D.13}$$

$$F_B(x_3, x_4) = \begin{bmatrix} B_{v,c} x_3 \\ B_{v,p} x_4 + \mathcal{F}_{c,p} \tanh\left(k_{tanh} x_4\right) \end{bmatrix}, \tag{D.14}$$

$$f_2(x_2, x_4) = -\begin{bmatrix} \frac{m_b l \sin(x_2) x_4^2 - g m_b \sin(x_2)\cos(x_2)}{m_c + m_b \sin^2(x_2)} \\ \frac{m_b l \sin(x_2)\cos(x_2) x_4^2 - g \sin(x_2)(m_c + m_b)}{l\left(m_c + m_b \sin^2(x_2)\right)} \end{bmatrix}, \tag{D.15}$$

$$g_1(x_2) = \frac{K_t}{r}\begin{bmatrix} \left(m_c + m_b \sin^2(x_2)\right)^{-1} \\ \cos(x_2)\left(l\left(m_c + m_b \sin^2(x_2)\right)\right)^{-1} \end{bmatrix}. \tag{D.16}$$

However, since the EKF in section 4.4 is based on a discrete time model, the Jacobian is also based on the discrete model. The model is discretized based on the Forward Euler method,

$$x_{k+1} = \mathbf{x}_k + T_s\left(f\left(\mathbf{x}_k\right) + g(\mathbf{x}_k) u_k\right), \tag{D.17}$$

$$y_k = h\left(\mathbf{x}_k\right). \tag{D.18}$$

$$F_k(\mathbf{x}_k) = \frac{\delta\left(\mathbf{x}_k + T_s\left(f(\mathbf{x}_k) + g(\mathbf{x}_k) i_{a_k}\right)\right)}{\delta \mathbf{x}_k^T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + T_s \begin{bmatrix} \frac{\delta \dot{x}_k}{\delta \mathbf{x}_k^T} \\ \frac{\delta \dot{\theta}_k}{\delta \mathbf{x}_k^T} \\ \frac{\delta f_1(\theta_k)}{\delta \mathbf{x}_k^T} F_B(\dot{x}_k, \dot{\theta}_k) + \frac{\delta f_2(\theta_k, \dot{\theta}_k)}{\delta \mathbf{x}_k^T} + \frac{\delta g_1(\theta_k)}{\delta \mathbf{x}_k^T} i_{a_k} \end{bmatrix} \tag{D.19}$$

$$H_k(\mathbf{x}_k) = \frac{\delta h(\mathbf{x}_k)}{\delta \mathbf{x}_k^T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{D.20}$$

The Jacobian for equation D.19, is found, in individual parts in the following, however due to the equations long form, they will not be put into a combined matrix. Firstly the two first rows are computed, as they are simple.

$$\begin{bmatrix} \frac{\delta x_{3_k}}{\delta \mathbf{x}_k^T} \\ \frac{\delta x_{4_k}}{\delta \mathbf{x}_k^T} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{D.21}$$

Secondly the terms for $f_1 F_B$ are computed, which is split as the following, where the derivatives then are derived for each column afterwards.

$$\frac{\delta f_1(x_{2_k}) F_B(x_{3_k}, x_{4_k})}{\delta \mathbf{x}_{1_k}} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{D.22}$$

$$\frac{\delta f_1(x_{2_k})F_B(x_{3_k},x_{4_k})}{\delta \mathbf{x}_{2_k}} = -\begin{bmatrix} \frac{-m_b l\sin(2x_{2_k})F_{B_c} - \big(\sin(x_{2_k})\big(m_c+m_b\sin^2(x_{2_k})\big)+m_b\sin(2x_{2_k})\cos(x_{2_k})\big)F_{B_p}}{l(m_c+m_b\sin^2(x_{2_k}))^2} \\ \frac{-lF_{B_c}\sin(x_{2_k})\big(m_c+m_b\sin^2(x_{2_k})\big)-lF_{B_c}m_b\sin(2x_{2_k})\cos(x_{2_k})-F_{B_p}(m_c+m_b)\sin(2x_{2_k})}{l^2(m_c+m_b\sin^2(x_{2_k}))^2} \end{bmatrix} \tag{D.23}$$

$$\frac{\delta f_1(x_{2_k})F_B(x_{3_k},x_{4_k}}{\delta \mathbf{x}_{3_k}} = -\begin{bmatrix} f_{1_{1,1}}B_{v,c} \\ f_{1_{2,1}}B_{v,c} \end{bmatrix} \tag{D.24}$$

$$\frac{\delta f_1(x_{2_k})F_B(x_{3_k},x_{4_k})}{\delta \mathbf{x}_{4_k}} = -\begin{bmatrix} f_{1_{1,2}}(B_{v,p}+F_{c,p}k_{tanh}\text{sech}^2(k_{tanh}x_{4_k})) \\ f_{1_{2,2}}(B_{v,p}+F_{c,p}k_{tanh}\text{sech}^2(k_{tanh}x_{4_k})) \end{bmatrix} \tag{D.25}$$

The derivative for f_2 is then derived, where it first is split up again, as the matrices otherwise gets too large, to fit on the page.

$$\frac{\delta f_2(x_{2_k},x_{4_k})}{\delta \mathbf{x}_k^T} = \begin{bmatrix} 0 & \frac{\delta f_{2_1}}{\delta \mathbf{x}_{2_k}} & 0 & \frac{\delta f_{2_1}}{\delta \mathbf{x}_{4_k}} \\ 0 & \frac{\delta f_{2_2}}{\delta \mathbf{x}_{2_k}} & 0 & \frac{\delta f_{2_2}}{\delta \mathbf{x}_{4_k}} \end{bmatrix} \tag{D.26}$$

The derivative of each column is again found, which can be seen below.

$$\begin{aligned} &\frac{\delta f_2(x_{2_k},x_{4_k})}{\delta \mathbf{x}_{2_k}} \\ &= -\begin{bmatrix} \frac{\big(m_b l\cos(x_{2_k})x_{4_k}^2 - gm_b\big(1-\cos(2x_{2_k})\big)\big)(m_c+m_b\sin^2(x_{2_k})) - \big(m_b l\sin(x_{2_k})x_{4_k}^2 - gm_b\sin(x_{2_k})\cos(x_{2_k})\big)m_b\sin(2x_{2_k})}{(m_c+m_b\sin^2(x_{2_k}))^2} \\ \frac{\big(m_b l x_{4_k}^2(1-\cos(2x_{2_k}))-g\cos(x_{2_k})(m_c+m_b)\big)\big(l(m_c+m_b\sin^2(x_{2_k}))\big)-\big(m_b l x_{4_k}^2\sin(x_{2_k})\cos(x_{2_k})-g\sin(x_{2_k})(m_c+m_b)\big)lm_b\sin(2x_{2_k})}{l^2\big(m_c+m_b\sin^2(x_{2_k})\big)^2} \end{bmatrix} \end{aligned} \tag{D.27}$$

$$\frac{\delta f_2(x_{2_k},x_{4_k})}{\delta \mathbf{x}_{4_k}} = -\begin{bmatrix} \frac{2m_b l\sin(x_{2_k})x_{4_k}}{m_c+m_b\sin^2(x_{2_k})} \\ \frac{2m_b l\sin(x_{2_k})\cos(x_{2_k})x_{4_k}}{l(m_c+m_b\sin^2(x_{2_k}))} \end{bmatrix} \tag{D.28}$$

Lastly for the input matrix, $g_1 i_{a_k}$, the derivative is found below.

$$\frac{\delta g_1(x_{2_k})i_{a_k}}{\delta \mathbf{x}_k^T} = \begin{bmatrix} 0 & \frac{-K_t m_b\sin(2x_{2_k})}{r\big(m_c+m_b\sin^2(x_{2_k})\big)^2} & 0 & 0 \\ 0 & \frac{-K_t\big(\sin(x_{2_k})(l(m_c+m_b\sin^2(x_{2_k})))+\cos(x_{2_k})lm_b\sin(2x_{2_k})\big)}{rl^2\big(m_c+m_b\sin^2(x_{2_k})\big)^2} & 0 & 0 \end{bmatrix} i_{a_k} \tag{D.29}$$

Thereby the Jacobians of the system for the EKF are comprised of equations D.19 to D.29. The Jacobians can thereby be used in the implementation of the EKF.