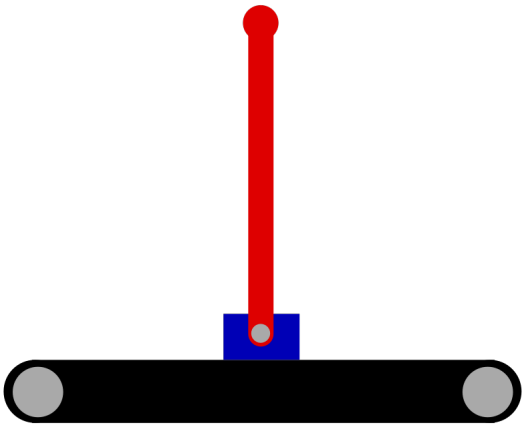
# The Inverted Pendulum

Semesterproject in Control and

Simulation of autonomous systems




**Project group 5**

Alex Ellegaard - aelle20 Anders Lind-Thomsen - andli20

Simon Christensen - simch20 Peter Frydensberg - pefry20

Thomas Therkelsen - ththe20 Victoria Jørgensen - vijoe20

**Supervisor**

Christian Schlette




BEng in Robot Systems

TEK MMMI

University of Southern Denmark


May 31st 2022

## Abstract

This report presents techniques for simulating and mathematically modeling an Inverted-Pendulum-Cart–System in order to develop control systems that are sufficiently effective at stabilizing the physical system. It begins with an introduction to the digital simulation of the system; Which includes a rigid body model as well as a visualization of the system and a control system design that lay the foundation and sets the expectations for mathematical modeling and control system design & implementation.

Furthermore, a mathematical model of the system is elucidated. In the procedure of determining a model of the system, Euler-Lagrange-modeling has been used. Then, the system is linearized and put on State-Space form, before being Laplace-transformed. After modeling the system, the control systems design process is described, which includes; Classical control with emphasis on PID control structures and modern control with emphasis on optimal control, specifically LQR control.

Moreover, the PLC implementation of the previously designed control systems is explained and tested, after which the performance is discussed. Lastly, a swing-up method is researched and tested.

# Preface

## Special Thanks

Special thanks should be given to Associate Professor Aljaz Kramberger for exemplary counseling regarding the design of control systems in this project, as well as to Associate Professor Stefan Brock for definitive guidance and assistance concerning Swing-Up.

## Contents of the ZIP File

The zip file contains the following files

- Machine Expert PLC code
- "video1.mp4" - Shows unstable cascade controller from simulated gains
- "video2.mp4" - Shows pendulum controller from simulated gains
- "video3.mp4" - Shows first stable controller
- "video4.mp4" - Shows final controller
- "video5.mp4" - Shows swingup without cascade

## Work Distribution

The work during the project has been distributed as follows

- Alex, Anders & Victoria: Implementation and tuning of PID controller on the PLC.
- Thomas: Rigidbody and Simulink modeling, and design of simulated controllers.
- Peter & Simon: Modeling of the equations of motion, and design of simulated controllers.

Alex Ellegaard

Simon Bork

Anders Lind-Thomsen

Thomas Therkelsen

Peter Frydensberg

Victoria Jørgensen

# Table of Contents

# 1 Introduction

The fourth industrial revolution, as some refer to it, is constantly evolving, as seen in the automation of many industrial and routine processes. One of the many advantages of automating a process is that the mistakes that people make during the manual process are eliminated. In light of these things, a wide range of robots are constructed for all kinds of purposes.

In this project, the robot is going to be based upon a PLC which will be controlling an inverted-pendulum-cart-system, which is inherently unstable. This is marginally resembling the classic kids' game of trying to keep a pencil upright on your palm by adjusting for the movement of the pencil with your hand. This is a case in which you can notice the robot eliminating human error, which is due to the fact that people cannot accurately measure and react to the motion of the pencil. - Whereas the robot can quantify the error precisely and adjust for it in a short amount of time.

Stabilizing an inverted pendulum on a cart is a textbook control problem. It is one that many will face in their journey through the world of engineering. This version of the problem is taken one step further, as it is not just an inverted pendulum on a cart. Rather, it is an inverted pendulum on a cart, strapped to a conveyor belt, powered by a DC motor, and controlled by a PLC.

## 1.1 Problem

***How do you stabilize the inverted pendulum?***

Moreover, these subquestions are answered

- How is the dynamical model of the system simulated and mathematically modeled such that a control system can be designed for it?
- How is a control system designed that sufficiently stabilizes the physical system?
- How is the control system implemented on the PLC?

## 1.2 Performance specifications

We wish to implement a control system on the PLC which is able to control the system by fulfilling the following requirements

- The system should be able to stabilize both the cart and pendulum around a desired reference point for at least 2 minutes with the reference point also being the initial condition.
- The system should be able to stabilize both the cart and pendulum around a desired reference point for at least 2 minutes with an offset of approximately 10 cm as the initial condition.
- The controller should be strong enough to catch and stabilize both the cart and pendulum after an upswing.

# 2 State of the Art

As previously mentioned, stabilizing an inverted pendulum on a cart is a classic engineering problem which is what will be surveyed and compared in this chapter.

## 2.1 Work [1]

The first work which is explored is the report of a Bachelor's Degree project in Aerospace Technology Engineering made by Erik Martínez Ramírez from Polytechnic University of Catalonia in January 2018.

Ramírez models the inverted pendulum system utilizing Newton-Euler modeling and linearizes the system around the upright position of the pendulum, using Simulink. He then proceeds to design and simulate full-state feedback, observer-based state feedback, PID control, and cascade PID control.

After the simulations, Ramírez tests the various control structures to find out which strategy works best on the physical system. Based on the tests, he concludes that the implemented state-feedback approach works, while the cascade PID control does not work. According to Ramírez, the reason for the cascade PID control not working is likely due to the lack of proper digitization of the control system.

## 2.2 Work [2]

This report is a Bachelor's thesis in Industrial Engineering and Management written by Rosalie van Oosterhout from the University of Groningen in 2018.

The report derives the transfer functions of the system using both Newton-Euler-based modeling and Euler-Lagrange-based modeling. This is done to show that both methods result in the same equations of motion. Moreover, state-space is also used as an alternative method to describe the input and output behavior of the system. The transfer functions and state-space matrices, which were obtained from the modeling of the system, are validated by doing simulated tests.

A PID controller is implemented both to control the angle of the pendulum and the position of the cart. When this was tested on the physical system, it was concluded that there was resonance in the aluminum pendulum. This is combatted with a hollow plastic pendulum with a weight at the top that is used to remove the resonance. After testing the hollow inverted pendulum it is shown that even though there is no resonance anymore, the stabilization of the pendulum is less optimal under the same conditions as the original pendulum.

### 2.3 Work [3]

The last report elucidated was written in 2011 by M. Hamza, Z. Rehman, Q. Zahid, F. Tahir, and Z. Khalid, from the Department of Electrical Engineering in Islamabad, Pakistan. Though this report is exclusively based on simulation, it is worth mentioning because of the linearization method used in various publications found online.

First, the equations of motion are derived using the Euler-Lagrange method. Hereafter, the equations are linearized by making use of small angle approximations:

$$\sin\theta \approx \theta, \cos\theta \approx 1, \theta^2 \approx 0 \tag{1}$$

These approximations are done to skip the lengthy linearization process, by approximating the non-linear elements of the equations of motion. A PID controller is then implemented in a parallel structure to control the system. Besides the PID controller, different control methods are also implemented such as Linear Quadratic Regulation and Fuzzy-Logic control. It is lastly concluded the designed Fuzzy-Logic controller delivers the best control and disturbance rejection performance and consumes relatively little control energy.

# 3 Simulated Modeling and Control Systems Design

To get an idea of how the system behaves under force control and to decide which control structure to implement, a model of the system and a few control systems are designed in Simulink, using a combination of Simscape and Simulink components. This segment was inspired by the CTMS website [4].

## 3.1 Rigid Body Model

In the rigid body model, the motor and belt are both omitted for simplification's sake. Using the parameters seen below, a simplified model of the real-life system is created using a few different Simscape components. This simultaneously generates a visualization as seen in figure 1.

The parameters of the system are given by

| Name | Symbol | Value | Unit |
|---|---|---|---|
| Length of cart | $l_c$ | 0.203 | m |
| Length of pendulum | $l_p$ | 0.350 | m |
| Width of cart | $w_c$ | 0.0038 | m |
| Width of pendulum | $w_p$ | 0.006 | m |
| Mass of cart | $m_c$ | 0.5 | kg |
| Mass of pendulum | $m_p$ | 0.0084 | kg |
| Cart damping | $b_c$ | 5 | $\mathrm{N}/(\mathrm{m}/\mathrm{s})$ |
| Pendulum damping | $b_p$ | 0.0012 | $\mathrm{N}/(\mathrm{rad}/\mathrm{s})$ |

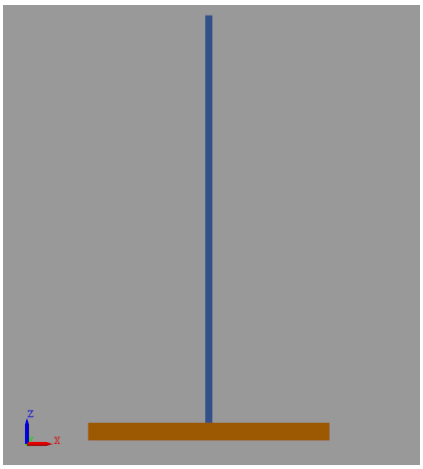


Figure 1: Rigid Body Model

## 3.2 System Model

Then, the model is hooked up to some converters, to allow for control systems design to commence.

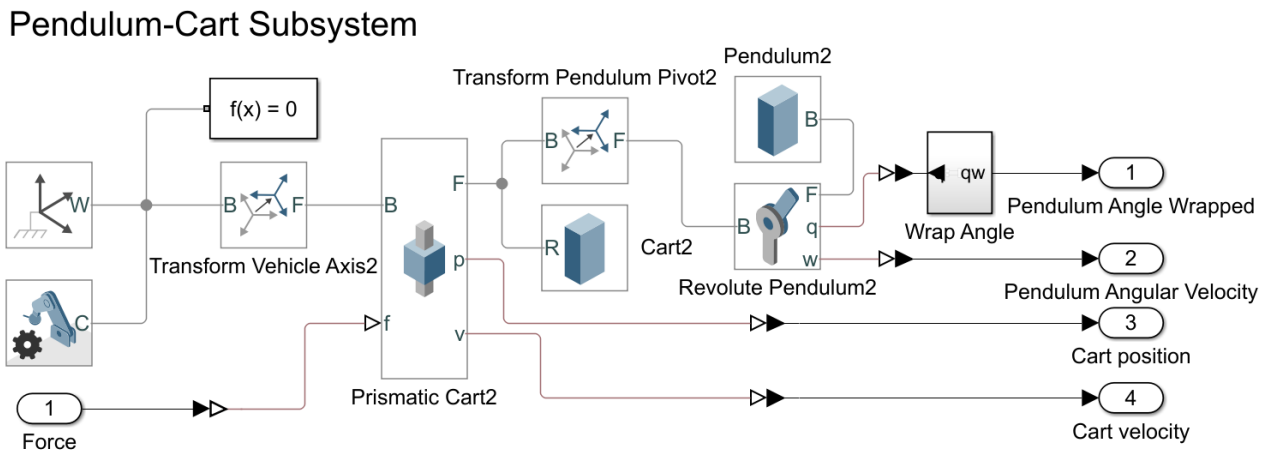



Figure 2: Pendulum-Cart Simscape Subsystem

Before initiating control systems design, it should be deducted whether the model is accurate. This is tested by applying neither disturbance nor control to the system, and reading the four states $\theta, \dot{\theta}, x, \dot{x}$.

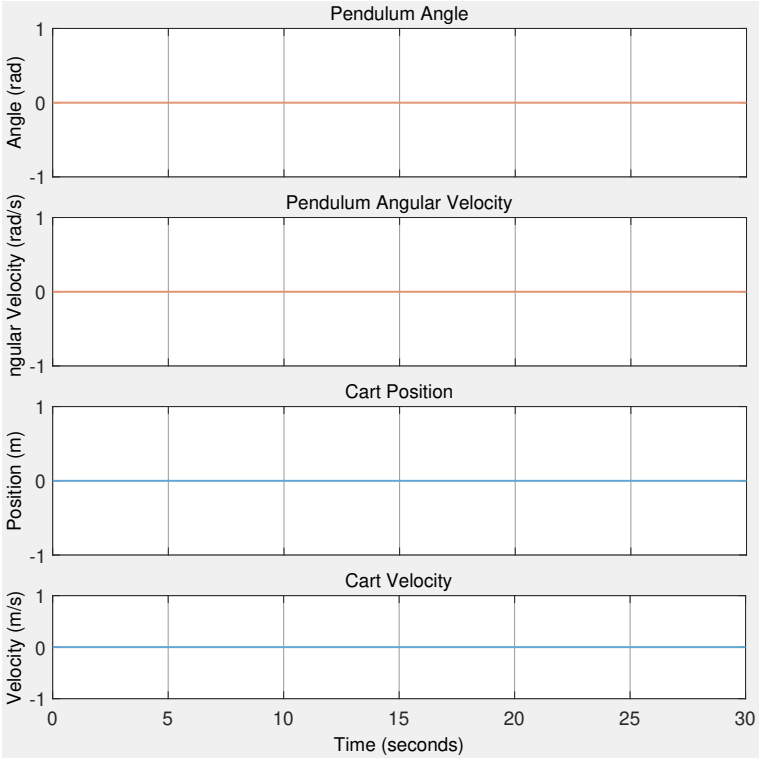



Figure 3: Disturbanceless Rigid Body System

As seen in figure 3, the pendulum starts at $0$ [rad] (pointing upright) and stays that way. This is due to the fact that it's a simulated reality with no added disturbances, and shows that the model is accurate. The different control structures will now be tested and compared, this will set the expectations for the mathematical derivation of the system

### 3.3 Parallel PID Control

A parallel PID controller system is made using the previously defined Pendulum-Cart Subsystem as the plant, and a few Simulink components, the main one being the built-in PID block.

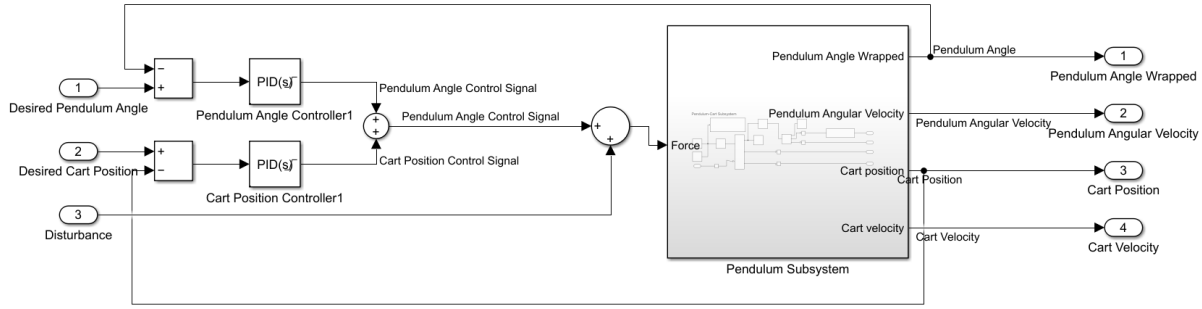
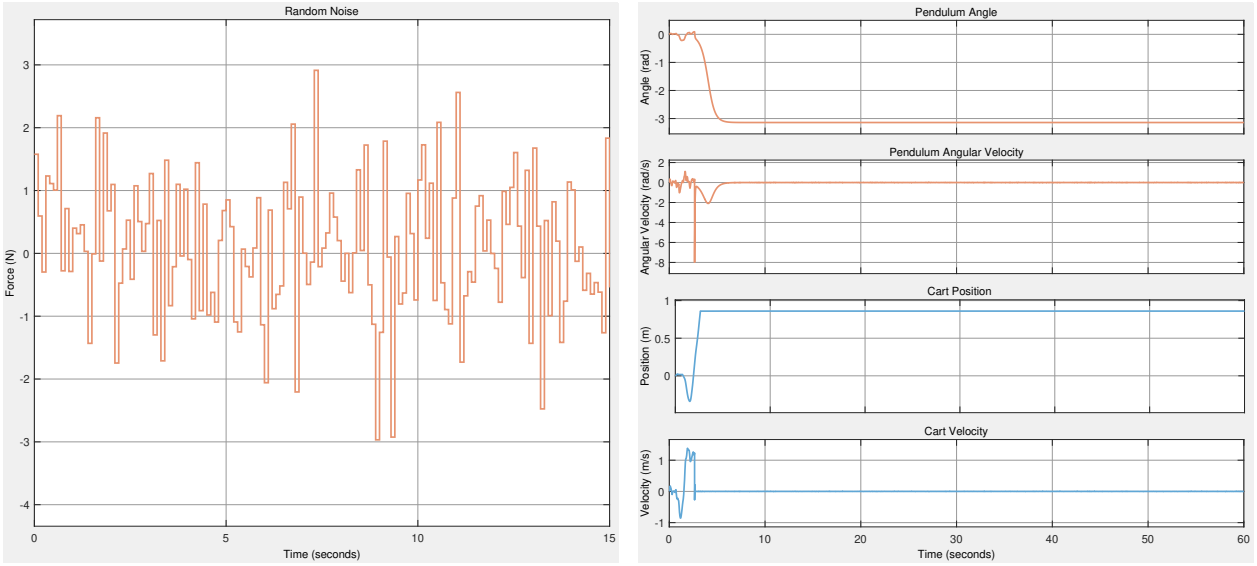



Figure 4: Simscape/Simulink Parallel Control System

To test the system's disturbance rejection and stability, a white noise generator is added to the force input of the system. The noise is seen in figure 5a below. The PID controllers used in the parallel control structure are tuned independently of each other, done by disconnecting one controller while the other is being tuned, and vice versa. It is worth noting, that the pendulum angle is wrapped to be between $-\pi$ and $\pi$ [rad]. Then the system performance is tested by giving the pendulum a reference of 0 [rad] and the cart a step input as the reference. The simulated control system's performance is seen in figure 5b.




(a) Disturbance Graphs (b) Parallel Control System Response

Figure 5: Disturbance and Parallel System Response

As seen in figure 5b, the pendulum becomes unstable as the cart attempts to track the reference at the same time as the pendulum. This means that the parallel control structure is not sufficient for stabilizing the system, and no further performance analysis will be made on this system as it is insufficient.

### 3.4 Cascade PID Control

Since parallel control did not stabilize the system, a cascade PID controller system is tested. It is made using the previously defined Pendulum-Cart Subsystem as the Plant, and a few Simulink components.

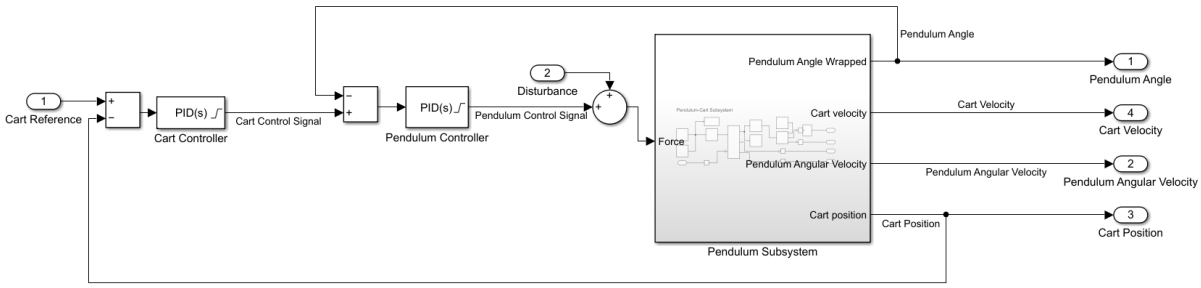



Figure 6: Simscape/Simulink Cascade Control System

The same disturbance is applied to the cascade system, as seen in figure 5a. Unlike in a parallel structure, the inner controller is tuned first, followed by the outer controller without disconnecting either. There is an unofficial rule for cascade control, which states that the inner loop should be anywhere from $4 - 10$ times faster than the outer loop [5, 6], depending on application and context. The fact of the matter is though, that the exact number doesn't matter how much faster the inner controller is, as long as it ensures sufficient performance. This system is tested in the same way as the parallel structure. The performance of the cascade system is seen in figure 7.

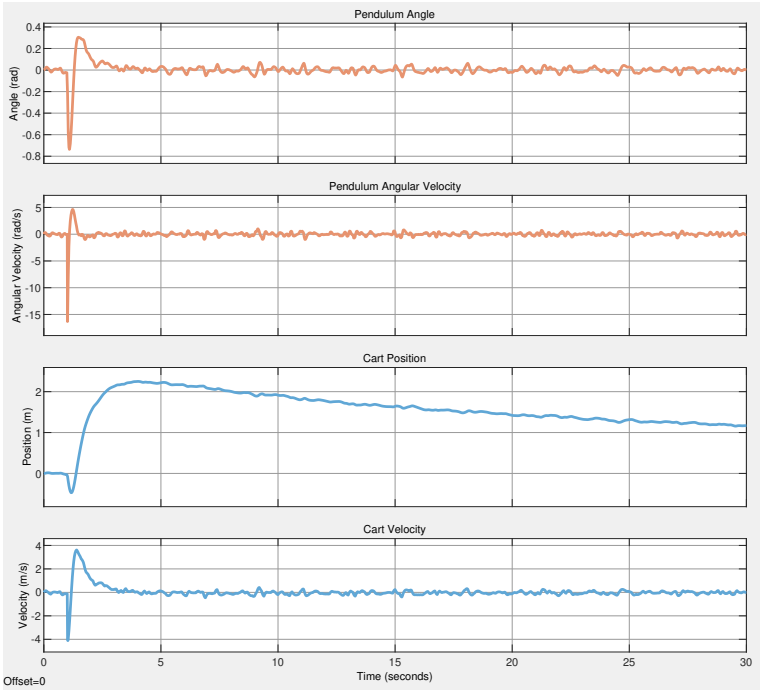



Figure 7: Cascade Control System Response

Unlike the parallel control system, this system can stabilize both the pendulum and the cart. Since no time-domain specifications have been set yet, no further performance rating will be made. In conclusion, it is expected that a cascade structure will perform better than a parallel structure, on the physical system.

# 4 Mathematical Modeling

This project revolves around controlling a SIMO - system (single input, multiple outputs) - a cart with a pendulum attached to it. The goal of the control system is to control the pendulum to an upright position, while simultaneously rejecting outside disturbances, The pendulum has 2 DOF (Degree(s) of freedom), and the cart has 1 DOF. The only input to the system is from the cart, which comes from a DC motor. An illustration of the system and a list of parameters are seen below.

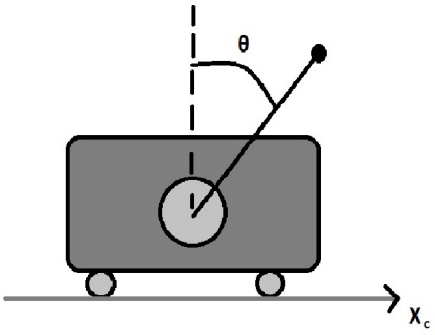



Table 1: Illustration of system

The parameters describing the system are

| Name | Symbol | Value | Unit |
|---|---|---|---|
| Mass of cart | $m_c$ | 0.5 | kg |
| Mass of pendulum | $m_p$ | 0.166 | kg |
| Cart damping | $b_c$ | 5 | $^{N}/_{(rad/s)}$ |
| Pendulum damping | $b_p$ | 0.0012 | $^{N}/_{(rad/s)}$ |
| Gravitational pull | g | 9.82 | $^{m}/_{s^2}$ |
| Length of pendulum | $l_p$ | 0.35 | m |
| Cart position | $x_c$ | - | m |
| Cart velocity | $\dot{x}_c$ | - | $^{m}/_{s}$ |
| Cart acceleration | $\ddot{x}_c$ | - | $^{m}/_{s^2}$ |
| Pendulum angle | $\theta$ | - | rad |
| Pendulum angular velocity | $\dot{\theta}$ | - | $^{rad}/_{s}$ |
| Pendulum angular acceleration | $\ddot{\theta}$ | - | $^{rad}/_{s^2}$ |
| Cart input force | u | - | N |

In order to control the system, a model of the system needs to be determined. This can be done by Euler-Lagrange modeling or Newton-Euler modeling. Here, Euler-Lagrange modeling is utilized.

## 4.1 Euler-Lagrange Modeling

The equations of motion are derived from the following formulas

$$\mathcal{L} = E_{kin} - E_{pot} \tag{2}$$

Where $\mathcal{L}$ is the Lagrangian, and $E_{kin}, E_{pot}$ are the kinetic and potential energy of the system respectively.

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial \mathcal{L}}{\partial \dot{q}} - \frac{\partial \mathcal{L}}{\partial q} = Q \tag{3}$$

Where $q$ is a vector of the generalized coordinates $q = [x_c, \theta]^T$, and $Q$ is a vector of generalized forces, in this case friction and input force $Q = [u - b_c \cdot \dot{x}_c, -b_p \cdot \dot{\theta}]^T$.

The kinetic energy of the system is given by

$$E_{kin} = \frac{1}{2} \cdot (m_c + m_p) \cdot \dot{x}^2 + m_p \cdot \dot{x} \cdot l_p \cdot \dot{\theta} \cdot \cos(\theta) + \frac{1}{2} \cdot m_p \cdot l_p^2 \cdot \dot{\theta}^2 \tag{4}$$

This consists of the kinetic energy from the translational velocity of the pendulum and the cart along the x-axis $\left(\frac{1}{2} \cdot (m_c + m_p) \cdot \dot{x}^2\right)$, the kinetic energy from the velocity of the pendulum along the y-axis $\left(m_p \cdot \dot{x} \cdot l_p \cdot \dot{\theta} \cdot \cos(\theta)\right)$ and lastly the kinetic energy coming from the angular velocity of the pendulum $\left(\frac{1}{2} \cdot m_p \cdot l_p^2 \cdot \dot{\theta}^2\right)$.

The potential energy of the system is given by the gravitational force on the pendulum in the y-direction

$$E_{pot} = m_p \cdot g \cdot l_p \cdot \cos\theta \tag{5}$$

By using equation 2, 3, 4 and 5, the equations of motion are calculated in the following procedure.

The first step is to calculate the Lagrangian.

$$\mathcal{L} = E_{kin} - E_{pot} = \frac{1}{2} \cdot (m_c + m_p) \cdot \dot{x}^2 + m_p \cdot \dot{x} \cdot l_p \cdot \dot{\theta} \cdot \cos(\theta) + \frac{1}{2} \cdot m_p \cdot l_p^2 \cdot \dot{\theta}^2 - m_p \cdot g \cdot l_p \cdot \cos\theta \tag{6}$$

The second step is to calculate the differentials with respect to the Lagrangian

$$\begin{bmatrix} \frac{\partial \mathcal{L}}{\partial x} \\ \frac{\partial \mathcal{L}}{\partial \theta} \\ \frac{\partial \mathcal{L}}{\partial \dot{x}} \\ \frac{\partial \mathcal{L}}{\partial \dot{\theta}} \end{bmatrix} = \begin{bmatrix} 0 \\ m_p \cdot l_p \cdot \sin(\theta) \cdot (g - \dot{\theta} \cdot \dot{x}) \\ (m_c + m_p) \cdot \dot{x} + m_p \cdot l_p \cdot \dot{\theta} \cdot \cos(\theta) \\ m_p \cdot l_p \cdot \dot{x} \cdot \cos(\theta) + m_p \cdot l_p^2 \cdot \dot{\theta} \end{bmatrix} \tag{7}$$

The third step is to calculate the time-derivatives of the derivatives of the Lagrangian. The chain rule of derivation is used here

$$\begin{bmatrix} \frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial \mathcal{L}}{\partial \dot{x}} \\ \frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial \mathcal{L}}{\partial \dot{\theta}} \end{bmatrix} = \begin{bmatrix} (m_c + m_p) \cdot \ddot{x}_C + m_p \cdot l_p \cdot \ddot{\theta} \cdot \cos(\theta) - m_p \cdot l_p \cdot \sin(\theta) \cdot \dot{\theta}^2 \\ m_p \cdot l_p^2 \cdot \ddot{\theta} + m_p \cdot l_p \cdot \ddot{x}_C \cdot \cos(\theta) - m_p \cdot l_p \cdot \dot{x} \cdot \sin(\theta) \cdot \dot{\theta} \end{bmatrix} \tag{8}$$

The last step is to combine the previous calculations, in order to get the equation of motion

$$\begin{bmatrix} (m_c + m_p) \cdot \ddot{x}_C + m_p \cdot l_p \cdot \ddot{\theta} \cdot \cos(\theta) - m_p \cdot l_p \cdot \sin(\theta) \cdot \dot{\theta}^2 \\ m_p \cdot l_p^2 \cdot \ddot{\theta} + m_p \cdot l_p \cdot \ddot{x}_C \cdot \cos(\theta) - m_p \cdot l_p \cdot \dot{x} \cdot \sin(\theta) \cdot \dot{\theta} - m_p \cdot l_p \cdot \sin(\theta) \cdot (g - \dot{\theta} \cdot \dot{x}) \end{bmatrix} = \begin{bmatrix} u - b_c \cdot \dot{x}_C \\ -b_p \cdot \dot{\theta} \end{bmatrix} \tag{9}$$

These equations of motion are non-linear because of the trigonometric functions and the '$\dot{\theta}^2$'. Therefore, they have to be linearized, before modern and classical control strategies can be used on the system.

### 4.2 Linearization

In order to linearize, the Jacobian matrix is utilized with linearization around an equilibrium point. The equilibrium point is a system state, around which, the system should be moving. In our case, this point corresponds to the upright pendulum-position, and an arbitrary cart position. This method also brings the system directly to the state-space form which is used for modern control strategies. The Jacobian utilizes a first order Taylor-expansion, but the mathematics behind the Jacobian will not be further explained. Before linearizing the system, the general state-space notation is described, since it plays a vital role in the linearization.

The state-space form is a way of describing a linear time-invariant system (LTI)

$$\dot{x} = A \cdot x + B \cdot u \qquad\qquad y = C \cdot x + D \cdot u \tag{10}$$

Here, x is the state-space vector, B is the input matrix, y describes the output of the measurements, C is the output matrix and D is the direct feedthrough matrix, which in many cases is set to a matrix of zeros. In our case, the state-vector x is equal to $x = [x_c, \dot{x}_c, \theta, \dot{\theta}]^T$.

In order to linearize with the Jacobian matrix, an equilibrium point needs to established. This point is given by: $\left(\bar{u}, \bar{x}_c, \dot{\bar{x}}_c, \bar{\theta}, \dot{\bar{\theta}}\right) = (0, 0, 0, \pi, 0)$, since the pendulum should be in upright position $\left(\bar{\theta}\right)$ with ideally no angular velocity $\left(\dot{\bar{\theta}}\right)$. In this pendulum-position, the cart should be standing still $(\dot{\bar{x}}_c, \bar{u})$ in an arbitrary position, here set to 0 for simplicty. In order to use the Jacobian, the two non-linear second order equations of motion need to be reformulated into four first order equations. Here MATLAB is used to isolate the different state-variables from equation 10

$$\begin{bmatrix} v \\ \dot{v} \\ \omega \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ -\frac{l_p \cdot m_p \cdot \sin(\theta)^2 + b_p \cdot \cos(\theta) - u + b_c \cdot v - g \cdot l_p \cdot m_p \cdot \cos(\theta) \cdot \sin(\theta)}{-l_p \cdot m_p \cdot \cos(\theta)^2 m_p + m_c} \\ \dot{\theta} \\ -\frac{b_p \cdot \omega \cdot m_p + b_p \cdot \omega \cdot m_p - u \cdot l_p \cdot m_p \cdot \cos(\theta) - g \cdot l_p \cdot m_p^2 \cdot \sin(\theta) + \omega^2 \cdot l_p^2 \cdot m_p^2 \cdot \cos(\theta) \cdot \sin(\theta) + b_c \cdot v \cdot l_p \cdot m_p \cdot \cos(\theta) - g \cdot l_p \cdot m_p \cdot m_c \cdot \sin(\theta)}{-l_p^2 \cdot m_p^2 \cdot \cos(\theta)^2 + m_c \cdot l_p \cdot m_p + m_p^2 \cdot l_p} \end{bmatrix} \tag{11}$$

These equations are now first-order equations and are ready to be used in the Jacobian. In order to use the Jacobian, some more notation is presented. The first element of the right-hand vector of equation 13 is called 'eq1', the second element of the right-hand vector of equation 13 is called 'eq2', the third element is called 'eq3', and the same procedure with the fourth element which is called 'eq4'.

The formula for the $A$ and $B$ state-space matrices are as follows:

$$A = \begin{bmatrix} \frac{\partial eq1}{\partial x} & \frac{\partial eq1}{\partial \dot{x}} & \frac{\partial eq1}{\partial \theta} & \frac{\partial eq1}{\partial \dot{\theta}} \\ \frac{\partial eq2}{\partial x} & \frac{\partial eq2}{\partial \dot{x}} & \frac{\partial eq2}{\partial \theta} & \frac{\partial eq2}{\partial \dot{\theta}} \\ \frac{\partial eq3}{\partial x} & \frac{\partial eq3}{\partial \dot{x}} & \frac{\partial eq3}{\partial \theta} & \frac{\partial eq3}{\partial \dot{\theta}} \\ \frac{\partial eq4}{\partial x} & \frac{\partial eq4}{\partial \dot{x}} & \frac{\partial eq4}{\partial \theta} & \frac{\partial eq4}{\partial \dot{\theta}} \end{bmatrix}\Bigg|_{\left(u,x,\dot{x},\theta,\dot{\theta}\right)=\left(\bar{u},\bar{x},\dot{\bar{x}},\bar{\theta},\dot{\bar{\theta}}\right)} \qquad B = \begin{bmatrix} \frac{\partial eq1}{\partial u} \\ \frac{\partial eq2}{\partial u} \\ \frac{\partial eq3}{\partial u} \\ \frac{\partial eq4}{\partial u} \end{bmatrix}\Bigg|_{\left(u,x,\dot{x},\theta,\dot{\theta}\right)=\left(\bar{u},\bar{x},\dot{\bar{x}},\bar{\theta},\dot{\bar{\theta}}\right)} \tag{12}$$

These equations output the following state-space-matrices which can then be used to describe the system, and modern control can therefore be utilized. The C-matrix is determined from what is being measured which is the angle $\theta$ and the position $x_c$. The final structure becomes

$$\begin{bmatrix} \dot{x} \\ \ddot{x}_C \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -10.0 & -1.6498 & 0.0134 \\ 0 & 0 & 0 & 1 \\ 0 & 55.8659 & 64.0769 & -0.5208 \end{bmatrix} \cdot \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ 2 \\ 0 \\ -11.1732 \end{bmatrix} \cdot \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} \tag{13}$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} \tag{14}$$

In order to do classical control, the transfer functions of the system need to be determined. This can be done using the MATLAB function **ss2tf(**A,B,C,D**)** [7]. The transfer functions become

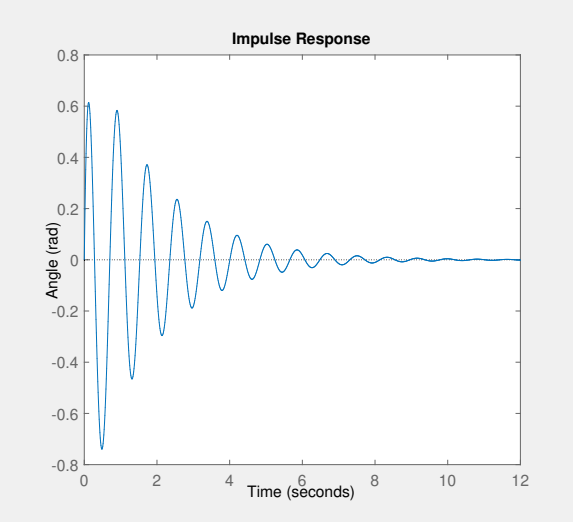
$$G_P(s) = \frac{\theta(s)}{u(s)} = \frac{-11.17\ s}{s^3 + 10.52\ s^2 - 59.62\ s - 548.6} \tag{15}$$

$$G_C(s) = \frac{x(s)}{u(s)} = \frac{2\ s^2 + 0.8917\ s - 109.7}{s^4 + 10.52\ s^3 - 59.62\ s^2 - 548.6\ s} \tag{16}$$

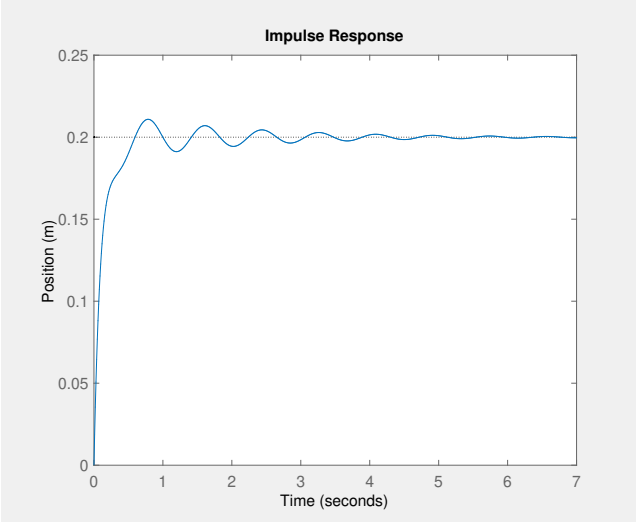
### 4.3 Model Verification

In order to test the modeling and linearization, the impulse responses of both systems are presented and analyzed. This is done both for the equilibrium point with the pendulum up, and the equilibrium point with the pendulum down. If the graphs then match the expected behavior it is concluded that the model and linearization are adequate.

For the equilibrium point with the pendulum down, the pendulum angle is expected to oscillate and converge to $\theta = 0$ due to friction and gravity. The cart is expected first to go to an x-position, then to begin oscillating because of the pendulum, and lastly converge to a standstill due to friction.



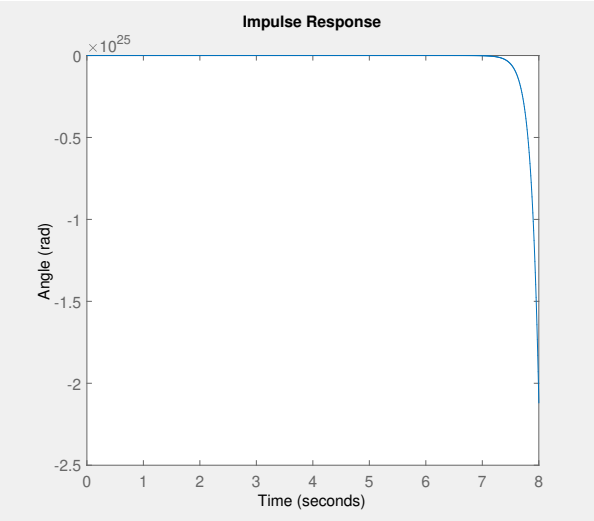

(a) Pendulum Impulse-response



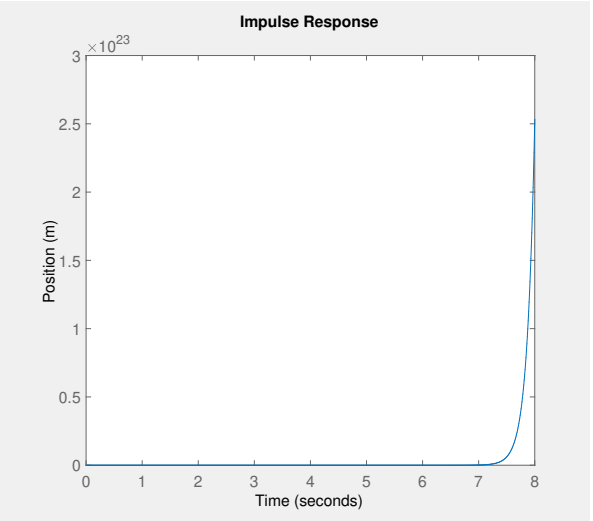

(b) Cart Impulse-response

Figure 8: Impulse-responses for the equilibrium point with the pendulum angle downwards

For the equilibrium point having the pendulum pointing upwards, the cart and the pendulum are expected to be unstable and diverge from the reference.




(a) Pendulum Impulse-response




(b) Cart Impulse Response

Figure 9: Impulse responses for the equilibrium point with the pendulum angle upwards

As seen on the graphs, the expectations match the graphs, so this confirms that the modeling is true. It is the equilibrium point corresponding to the upwards pendulum angle which is chosen to be controlled.

The system has now been modeled using Euler-Lagrange-modeling and linearized with the Jacobian matrix. The state-space matrices and the transfer functions have also been determined, so it is now possible to do classical and modern control strategies.

# 5 Classical Controller Design

## 5.1 System Analysis

Before doing any control systems design, the system should be analyzed to determine which approach should be taken. In figure 10 below, the Pole-Zero map and open-loop step-responses of the system are seen.

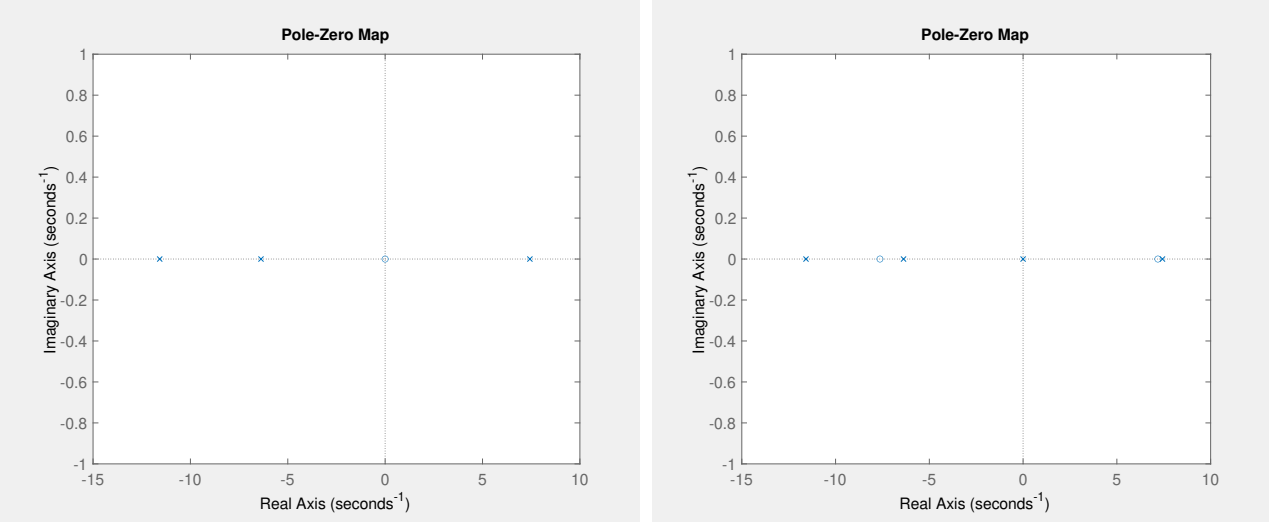



Figure 10: System Pole-Zero Maps

The Pole-Zero map shows that both systems have a pole in the right half-plane, making both of them unstable. To decide which kind of controller to use, root locus plots are generated to find out which kind of controller can cause the poles to be moved to the left half-plane, and hence stabilize the system.

## 5.2 SISO PID Control

This section shows classical control system strategies for SISO-systems (single input, single output) and is therefore not implemented on the physical system as it is a SIMO system. The design procedure is based only on controlling the pendulum.

### 5.2.1 Design-procedure

The first step of designing a controller with classical control strategies is to set up the performance requirements in the time domain.

- $M_p$ is overshoot. This is should be less than $25\%$.
- $t_r$ is rise time. This is should be $0.5$ seconds.
- $t_s$ is settling time. This is should be a $15\%$ settling time of $2$ seconds.
- There should be no steady-state error with a step input.

In figure 11 below, the different symbolic values of the specifications are seen.

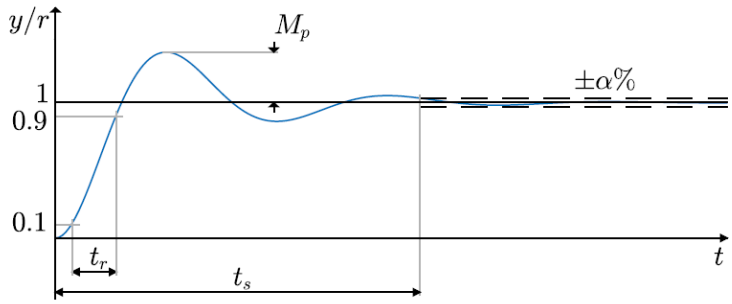



Figure 11: Time-Domain Requirements

These time-domain specifications have to be transformed into frequency-domain specifications. The frequency-domain specifications show the desired pole locations in the $s$-domain and can be found using the root locus method. This is the method that will be presented here. Firstly, the poles of the closed loop system have to be moved to the left of the blue lines in the $s$-domain shown in figure 13.

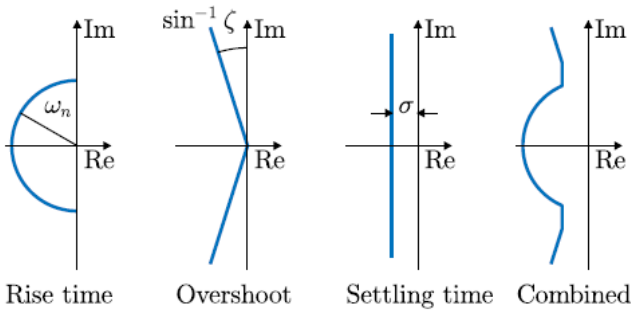



Figure 12: Frequency-Domain Requirements

The different requirements can be calculated using the following formulas

$$\omega_n \geq \frac{1.8}{t_r} = 3.6 \qquad \zeta \geq \sqrt{\frac{(\frac{\ln M_p}{-\pi})^2}{1+(\frac{\ln M_p}{-\pi})^2}} = 0.4037 \qquad \sigma \geq \frac{-\ln(\frac{\alpha}{100})}{t_s} = 0.4335 \tag{17}$$

Furthermore, the system has to be type one, in order to eliminate steady-state error with a step input. This means that there needs to be at least one pole in the origin of the loop gain.

These requirements can now be used in the root locus method. First, the system structure and closed loop transfer function need to be set up. The closed loop transfer function for a PID control system is given by

$$H_p = \frac{G(s) \cdot K(s)}{1 + G(s) \cdot K(s)} \tag{18}$$

Here $K(s)$ is the controller, and $G(s)$ is the pendulum transfer function determined in the modeling segment. The loop-gain is defined as $L(s) = G(s) \cdot K(s)$, and this can be used to determine the system type, and is also directly used in the root locus. The next step is to try different controllers, to see if they can be stabilized and comply with the frequency-domain specifications.

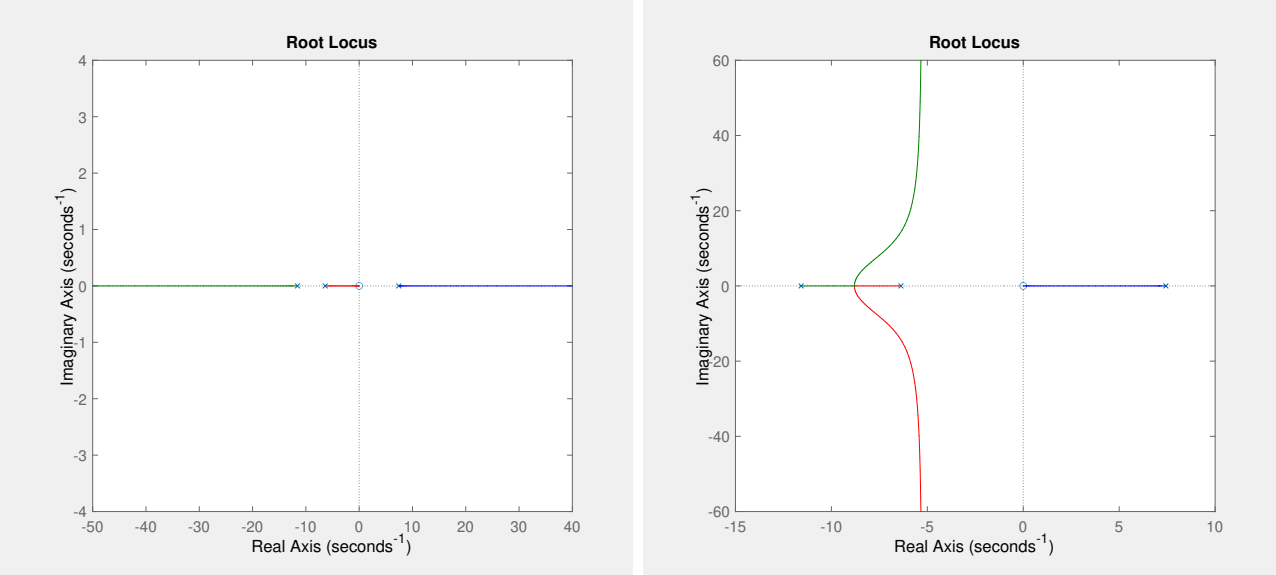
#### 5.2.1.1 P controller

P controller

$$K(s) = K_p \tag{19}$$

Where $K_p$ is the proportional gain. A P controller multiplies the calculated error $e(s) = r(s) - y(s)$ by the gain $K_p$, and generates a control-signal from that.

The root locus is plotted to see if the poles can be moved to the desired locations.




(a) Root Locus with positive gain (b) Root Locus with negative gain

Figure 13: P Controller Root Locus

As seen on the root locus plot, the system can't be stabilized using only a P controller, since the poles cannot be moved into the left half-plane of the $s$-domain. Therefore, it's attempted to add a zero to the controller with a D-term.

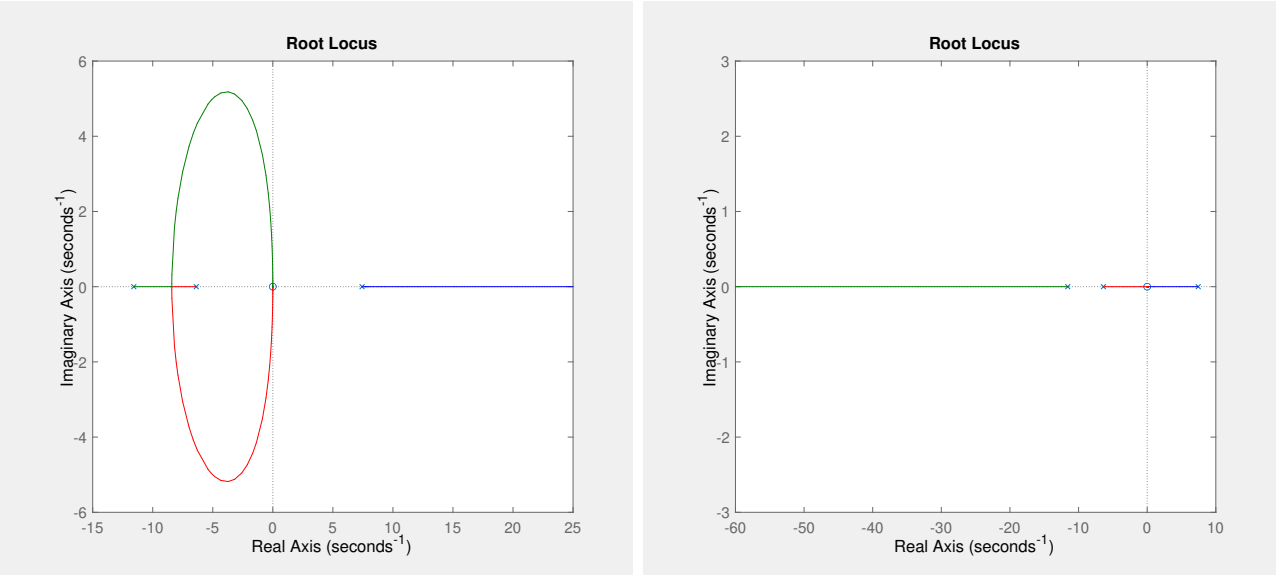
#### 5.2.1.2 PD controller

A PD controller on the parallel form is given by

$$K(s) = K_p + K_d \cdot s \tag{20}$$

Where $K_d$ is the differential gain. A D-term multiplies the derivative of the error $\frac{\mathrm{d}e(t)}{\mathrm{d}t}$ by the gain $K_d$ and generates a control signal from that.

The root locus is plotted to see if the poles can be moved to the desired locations.




(a) Root Locus with positive gain (b) Root Locus with negative gain

Figure 14: PD controller Root Locus

As seen on the root locus plot, the system also can't be stabilized using a PD-controller, since the poles cannot be moved into the left half-plane of the $s$-domain. Therefore, a PI controller is tested.

#### 5.2.1.3 PI controller

A PI controller on the parallel form is given by:

$$K(s) = K_p + K_i \cdot \frac{1}{s} \tag{21}$$

Where $K_i$ is the integral term, which sums up the error over time $\int_{t_0}^{t} e(\tau)d\tau$, and multiplies it with the constant $K_i$.

The poles still cannot be moved into the stable part of the $s$-domain, so a PID controller is tested.

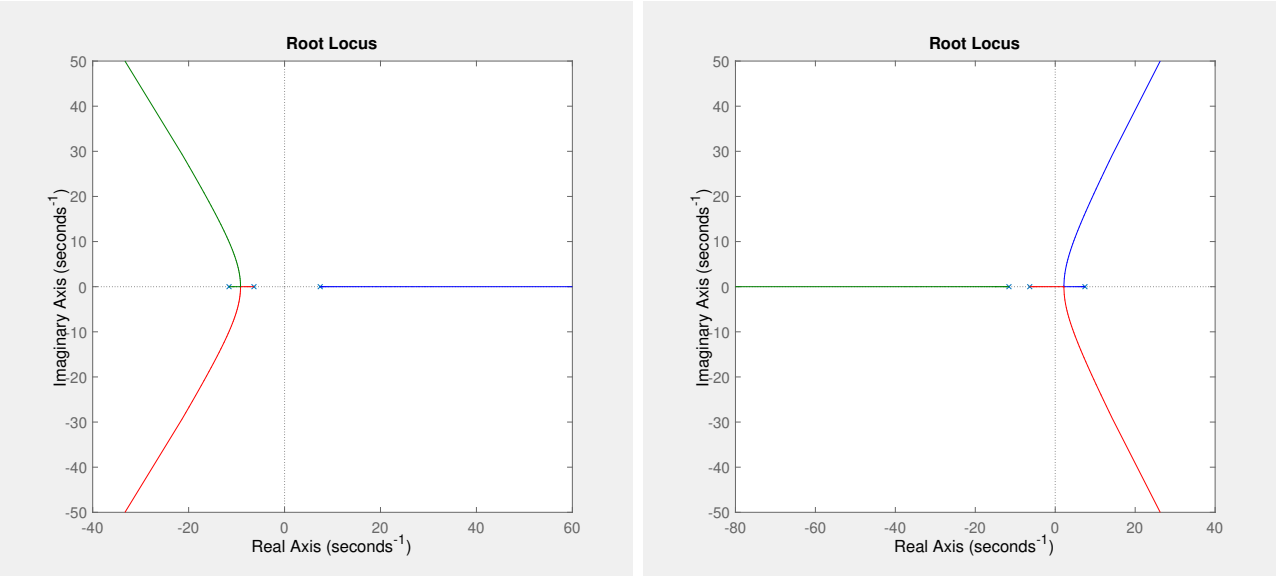
#### 5.2.1.4 PID controller

A PID controller on the parallel form is given by:

$$K(s) = K_p + K_i \cdot \frac{1}{s} + K_d \cdot s \tag{22}$$

This controller is a combination of all terms.

This expression has three unknowns $K_p, K_i$ and $K_d$, and is tuned using the built-in MATLAB function **pidtune(**system, 'type'**)** [8]. The root locus for this controller is presented here.




(a) Root Locus with positive gain (b) Root Locus with negative gain

Figure 15: PI Controller Root Locus

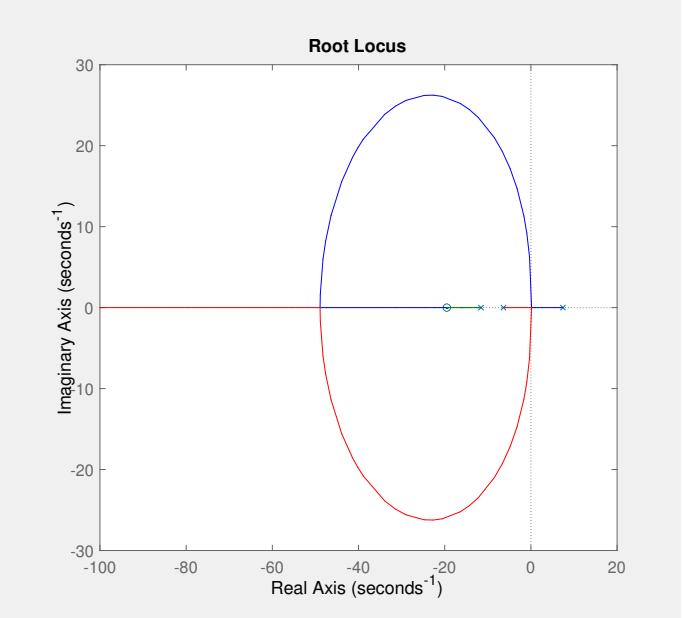



Figure 16: PID Controller Root Locus

As seen, the poles can be moved into the stable region of the $s$-domain by using PID controller gains given by

$$K(s) = -354 - 3460 \cdot \frac{1}{s} - 9.07 \cdot s \tag{23}$$

This controller also moves our system poles to locations that comply with our performance specifications. These poles are given by

$$P = \begin{bmatrix} -47.8 + 7.7 \cdot i & -47.8 - 7.7 \cdot i & -16.2 \end{bmatrix} \tag{24}$$

Therefore, they all comply with our performance specifications and the resulting step-response on the closed loop system becomes

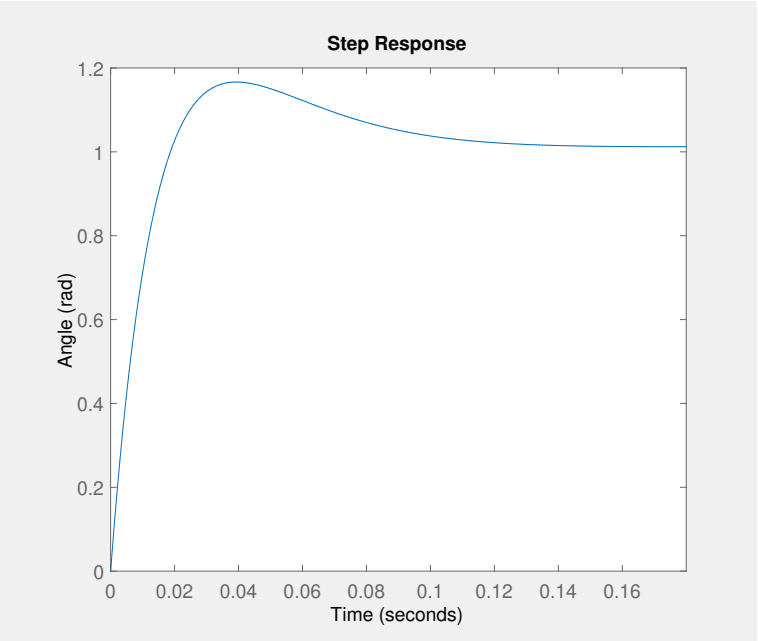



Figure 17: Closed-Loop Step-Response

The performance of this system is

$$M_p = 14.99\ \% \qquad t_r = 0.0143\ [s] \qquad t_s = 0.0824\ [s] \tag{25}$$

This is the controller, which should then be digitalized and implemented on the PLC. This will however not be done, since the model deviates too much from the actual system. This is probably due to the parameters being wrong. Therefore, the controllers being implemented will be tuned in Simulink instead.

## 5.3 SIMO PID Control

In the previous section, it was concluded that the root locus method is only usable for SISO-systems. In this case, the system is SIMO, so another method is needed. Here, two ways of controlling the system are mentioned, namely parallel control and cascade control. The models are made in Simulink and the controllers are tuned for these systems using Simulink's PID tuner.

### 5.3.1 Parallel Control

Based on the previously defined PID control system, which only adjusts the pendulum angle, a parallel PID control system can be designed. A parallel structure is chosen because it is desired to not only control the pendulum angle, but also the cart position at the same time using two different references. One of the downsides of this method is that the two systems, the cart, and the pendulum, will be prioritized equally. This is a downside, in our case, since the pendulum angle should be prioritized and not the position.

#### 5.3.1.1 Structure

The parallel structure can be seen in figure 20.

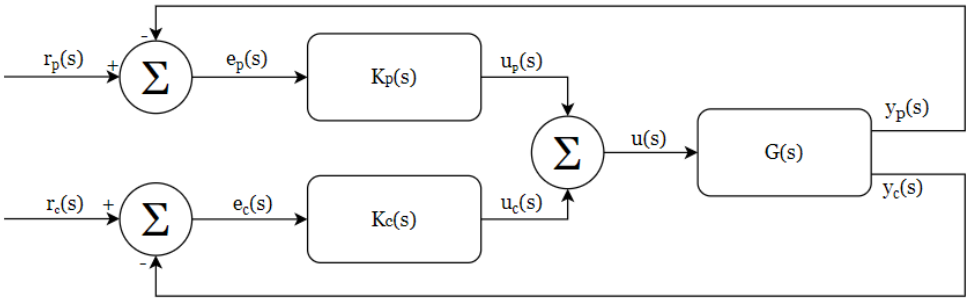



Figure 18: Parallel PID Control Structure

Here, $r_p(s)$ and $r_c(s)$ are the references for the cart and the pendulum, $e_p(s)$ and $e_c(s)$ are the errors, $K_P(s)$ and $K_C(s)$ are the controllers, $G(s)$ is the plant consisting of the two transfer functions, which output $y_p(s)$ and $y_c(s)$ respectively. The input to the system is $u(s)$ and is equal to the sum of the controllers' outputs $u_c(s)$ and $u_p(s)$.

#### 5.3.1.2 Tuning

A valid way of tuning the controllers of a parallel structure is to tune each controller independently of each other. In our case, the pendulum controller has already been designed, so only the cart controller is missing. To speed up and simplify the process, the system is designed and tuned in Simulink as it has built-in features for that. The State-Space model looks as follows

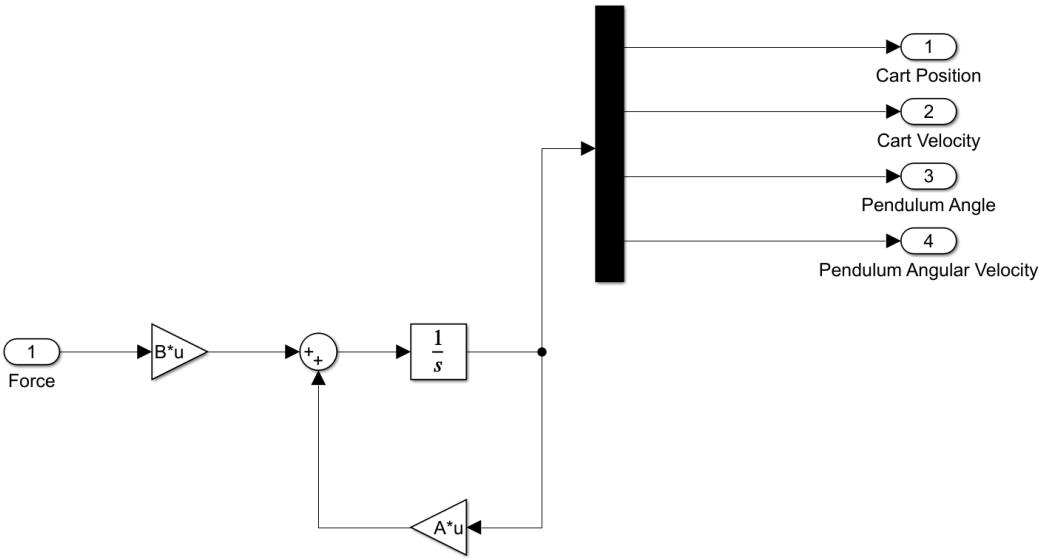



Figure 19: State-Space Model

Then, the parallel structure is made as previously shown in figure 18, and tuned using the built-in tuner for the Simulink PID block, ending up with the following controller parameters

$K_P(s)$ gains

$$K_p = 665.3913 \qquad K_i = 3794.7715 \qquad K_d = 27.1564 \tag{26}$$

$K_C(s)$ gains

$$K_p = 24.0370 \qquad K_i = 19.0188 \qquad K_d = 5.1235 \tag{27}$$

This results in the following performance of the system given a step-input as the cart reference, and $0$ [rad] as the pendulum reference.

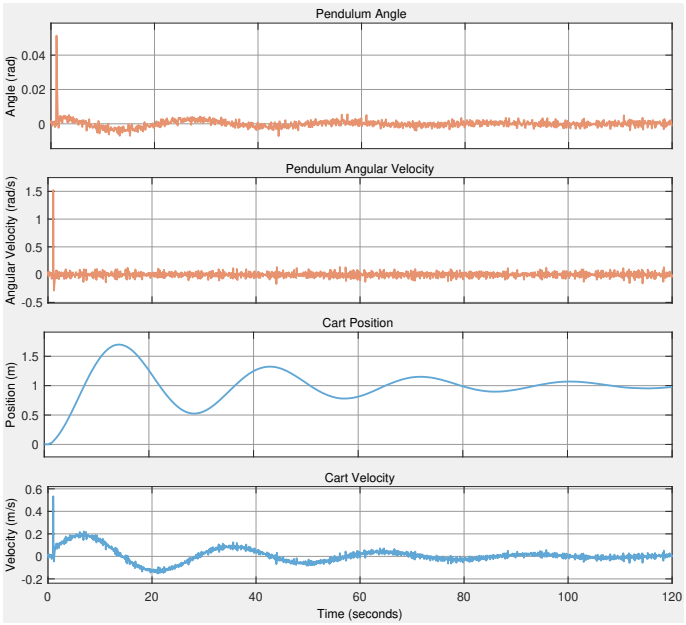



Figure 20: Parallel PID Input-response

#### 5.3.1.3 Performance

Analyzing the cart position seen in figure 20, the system performance is computed.

The performance of this system is

$$M_p = 70.200\ \% \qquad t_r = 8.327\ [\text{s}] \qquad t_s = 72.499\ [\text{s}] \tag{28}$$

Comparing this with the performance specifications made in section 5.2.1, shows that this system does not meet the set requirements. This means another approach must be taken, to stabilize the system accordingly.

### 5.3.2 Cascade Control

Another way of controlling the system, which performed better on the rigid body system, is a cascade structure. This structure similarly utilizes multiple controllers, where the outer controller gets a custom reference, and the inner controller moves towards the linearization point.

#### 5.3.2.1 Structure

In the figure below, the cascade control system is shown.

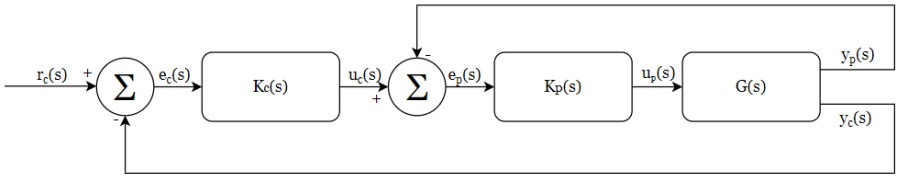



Figure 21: Cascade PID Control structure

Here, $r_c(s)$ is the reference for the cart, $e_p(s)$ and $e_c(s)$ are the errors, $K_P(s)$ and $K_C(s)$ are the controllers, $G(s)$ is the plant consisting of the two transfer functions, which output $y_p(s)$ and $y_c(s)$ respectively. The input to the system is $u_p(s)$, which is calculated by the two controllers.

Here the outer controller $K_c$ controls the outer loop and gives its output as input to the inner controller $K_p$. This inner controller computes a control signal rejecting disturbances before passing it on to the plant $G(s)$. Here only one reference is passed $r_c(s)$, which is given to the outer controller. The inner controller will control the inner system to move towards its equilibrium point. As previously mentioned it is essential for the inner loop to respond much faster than the outer loop.

#### 5.3.2.2 Tuning

While the previous systems were tuned using root locus and some manual tweaking, this system is tuned in a slightly different manner, namely using the built-in PID Tuner from the PID Block in Simulink. The final gains for the two controllers are seen below.

$K_P(s)$ gains

$$K_p = 142.91 \qquad K_i = 623.09 \qquad K_d = 8.1459 \tag{29}$$

$K_C(s)$ gains

$$K_p = 0.0105 \qquad K_i = 0,0001 \qquad K_d = 0,1101 \tag{30}$$

#### 5.3.2.3 Performance

Using the previously defined gains in the cascade control system, the following performance is obtained.

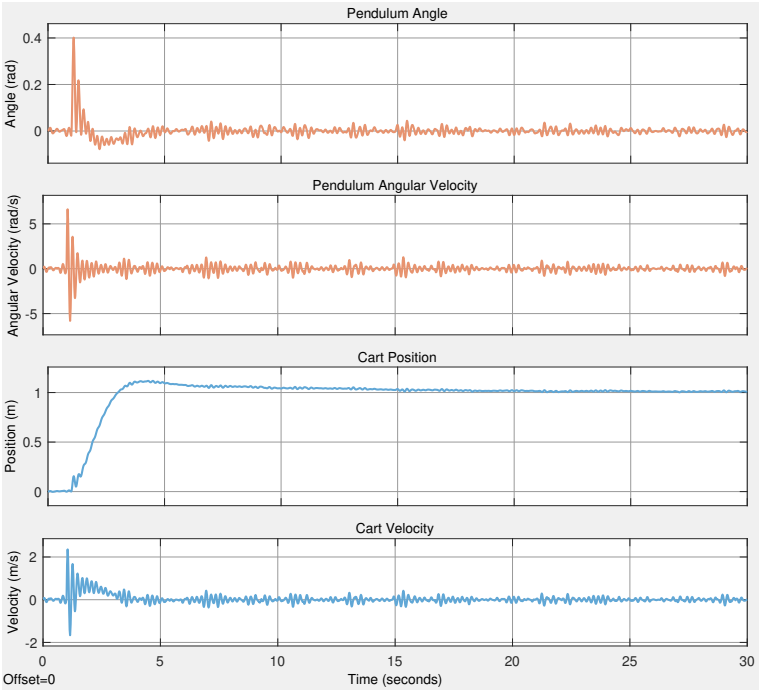



Figure 22: Cascade PID Input-response

Analyzing the cart position seen in figure 20, the system performance is computed.

The performance of this system is:

$$M_p = 12.2\ \% \qquad t_r = 1.771\ [s] \qquad t_s = 2.565\ [s] \tag{31}$$

Which absolutely complies with the performance specifications, meaning that this system is ready to be realized for the real system.

# 6 Realization

When going from a simulated system to a real-world system, several considerations must be made. First of all, the controllers need to be digitized, since computers do not work in continuous time. Second of all, the controllers do not take actuator saturation into account, which is when the motors cannot deliver the requested control signal. Third, the parameters of the simulated systems do not fully match the real setup.

## 6.1 Digitization

Digitization is the process of converting controllers from the continuous $s$-domain to the discrete $z$-domain, and then turning these into difference equations. This is necessary in order to implement classical control on computers and PLCs. When digitizing a transfer function, the response will have a delay in comparison to the simulations in continuous time. This will in our case be neglected because of the high sampling rate of the PLC. The conversion can be done by utilizing one of several different methods, which all map points from $s$-domain to $z$-domain differently from each other. The mapping methods can be seen in the following table.

| Method | Approximation |
|---|---|
| Forward Rule | $s = \frac{z-1}{T_s}$ |
| Backward Rule | $s = \frac{z-1}{T_s \cdot z}$ |
| Trapezoid Rule | $s = \frac{2}{T_s}\frac{z-1}{z+1}$ |

Here, $T_s$[s] is the sample time and has to match the PLC's sample time. The safest method to use is the trapezoid rule since all points from the left half plane of the $s$-domain are mapped to the inside of the unit circle in the $z$-domain. This method assures that the system with the controller will not suddenly become unstable. This can for example happen with the forward rule, where points from the left half plane of the $s$-domain can be mapped to anywhere in the $z$-domain. After using one of these mapping methods, the transfer functions can be turned into difference equations which can then can be implemented on the PLC. The controller from equation 23 will be used as an example of the discretization process.

First, the trapezoidal rule is used with a sample time $T_s = 0.001$[s]

$$K(z) = \frac{u(z)}{e(z)} = K(s)|_{s=\frac{2}{T_s}\frac{z-1}{z+1}} \tag{32}$$

This operation outputs the following discrete-time transfer function

$$K(z) = \frac{u(z)}{e(z)} = \frac{-18540 \cdot z^2 + 36280 \cdot z - 17790}{z^2 - 1} \tag{33}$$

The discrete transfer function is turned into a difference equation by first dividing each term with the highest order $z$

$$u(z) \cdot (1 - z^{-2}) = e(z) \cdot (-18540 + 36280 \cdot z^{-1} - 17790 \cdot z^{-2}) \tag{34}$$

And then replacing the $n$'th order exponentials on the $z$'s with a $n$'th sample delay, which results in the difference equation

$$u_k = -18540 \cdot e_k + 36280 \cdot e_{k-1} - 17790 \cdot e_{k-2} + u_{k-2} \tag{35}$$

This is the final difference equation, which can then be implemented on the PLC, and the procedure is repeated if there is more than one controller. This approach will however not be used in our case, since a built-in PID function block is utilized rather than being implemented manually.

### 6.2 Parameter Estimation

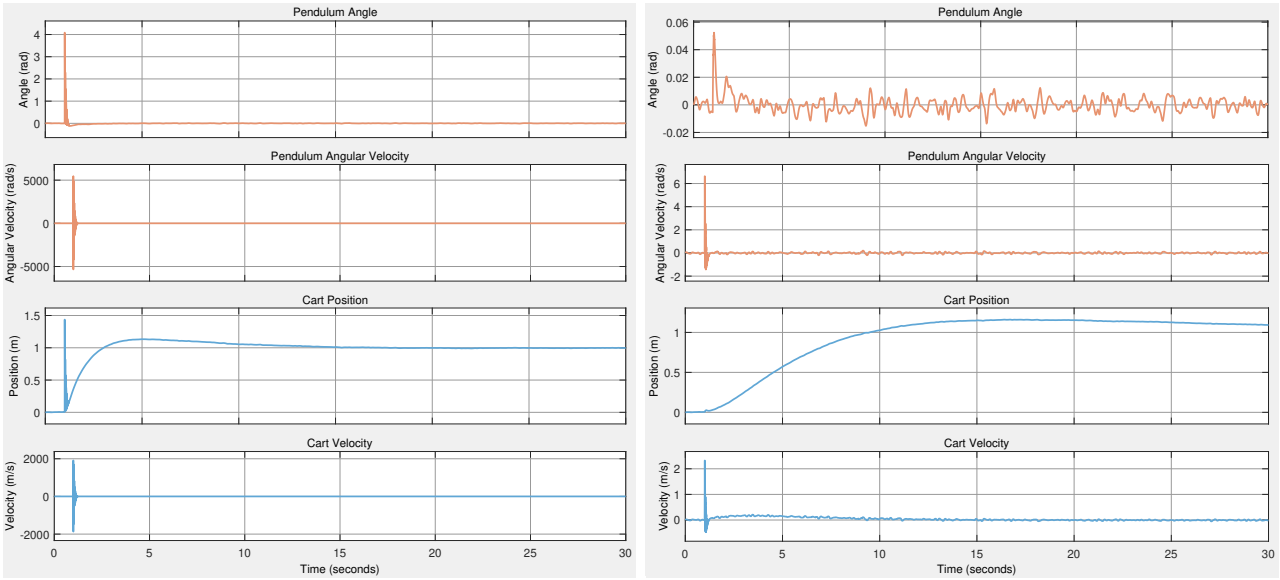
Another aspect of moving from simulation to a real setup is the number of errors on parameters. Ideally, before making a model of the system, a parameter estimation should be made, which would then make the model more precise and less prone to error. Even then, there would still be errors due to measuring insecurities. In our case, this is left as future work due to deadlines rapidly approaching. This is explained in detail in the discussion section.

### 6.3 Saturation & Anti-Windup

Another aspect of using a real setup is the reality of a limited force on the actuators. In a simulated environment, motors can be given an input in the regions of $10^7$ [N] and above. This however does not represent the real world. Therefore, an actuator model must be included. The most typical way of doing this is to calculate the maximum force that the actuators can handle, and then clamp the output of the controllers to this. For example, if an actuator can handle $10$ [N] at most, and the controller requests $11$ [N], the actuator model activates and sets the to $10$ [N].

This however presents a new issue, namely windup. The I-term of the PID controller will continue to add up when the system goes into saturation, which is when the actuator model activates, and this results in unwanted behavior. Therefore an anti-windup model must be included. There are generally two ways of doing this. One is called conditional integration and the other method is called back-calculation.

Conditional integration is about turning off the I-term and setting its contribution to the controller to 0 when the system is in saturation. This is the simplest method and gets the job done. Back-calculation is about recalculating the I-term when the system is in saturation. Here a tracking time constant $T_t$ is used, and it can be altered to change the behavior of the system to ones needs. The effect of including an actuator model and anti-windup can be seen in the following figure




(a) Without an actuator model and anti-windup (b) With an actuator model and anti-windup

Figure 23: The effect of an actuator model and anti-windup

# 7 Modern Control

This section describes the theory behind modern control and afterward how to control our system using Linear Quadratic Regulation. However, this control of the system will not be implemented.

## 7.1 Observability and Controllability

The main difference between classical control and modern control is that classical control utilizes transfer functions and modern control utilizes state-space matrices. The design procedure is based on controlling the entire system.

The first step is to check whether the system is observable and controllable. A system is controllable if there exists an input so that all states can be reached at some arbitrary time $T$. If a certain state can never be reached it isn't controllable. This can mathematically be checked using the state-space matrices

$$\mathcal{C} = \begin{bmatrix} B & B \cdot A & B \cdot A^2 & B \cdot A^3 \end{bmatrix} \tag{36}$$

If the rank of this controllability matrix $\mathcal{C}$ is equal to the number of states (4 in our case), the system will be controllable. The rank of the matrix is checked in MATLAB, and it has rank 4, meaning it's controllable.

The second step is to check whether the system is observable or not. A system is observable if $y(t) = 0$ it can be concluded that $x(t) = 0$. In practice, an observer is used to estimate the states that are not measured. In our case both $\theta$ and $x_c$ are measured, but not $\dot{\theta}$ and $\dot{x}_c$. Therefore it is necessary to include an observer, and an observer can only be included if the system is observable.

Mathematically, this can also be checked in the state-space matrices. If the rank of this observability matrix $\mathcal{O}$ seen in Equation 37, is equal to the number of states, the system will be observable. The rank of the matrix is checked in MATLAB, and it has rank 4 and the system can therefore be observed.

$$\mathcal{O} = \begin{bmatrix} C \\ C \cdot A \\ C \cdot A^2 \\ C \cdot A^3 \end{bmatrix} \tag{37}$$

Now that the system has been checked for observability and controllability, optimal control can be utilized.

### 7.2 Optimal Control (LQR)

An LTI-control system on the state-space form is considered as defined in Equation 10. Control for this type of system is optimal if it minimizes the cost function

$$\mathcal{J} \ = \ \int_0^\infty x^T Q x + u^T R u \ dt \tag{38}$$

where $Q = Q^T$ is a positive semi-definite matrix and $R = R^T$ is a positive definite matrix. The optimal state feedback law is then given by

$$u \ = \ Fx \tag{39}$$

Where $F = -R^{-1}B^T P$, and P is a stabilizing solution to the Algebraic Riccati Equation in Equation 40, and the eigenvalues of $A - BR^{-1}B^T P$ are in the open left half plane.

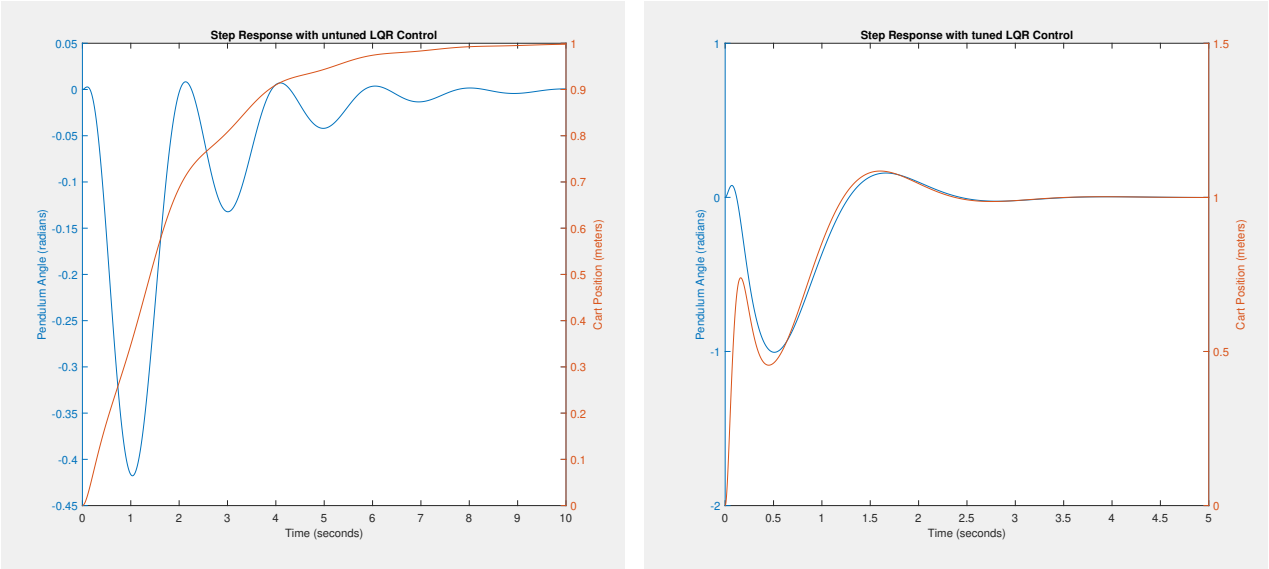
$$A^T P + PA - PBR^{-1}B^T P + Q \ = \ 0 \tag{40}$$

where $A \, \epsilon \, \mathbb{R}^{n \times n}$, $B \, \epsilon \, \mathbb{R}^{n \times m}$ are matrices, $n$ is the number of states and $m$ is the number of inputs.

LQR will be used to determine the state-feedback control matrix $F$. MATLAB offers a command **lqr(**A, B, Q, R**)** [9] where two parameters $Q$ and $R$ can be adjusted to balance the importance of the control effort and error. The simplest case is when $R = 1$ and $Q = C' \cdot C$. By tuning the nonzero elements in the $Q$ matrix, a desirable response can be achieved.

$$Q \ = \ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{41}$$

The element (1,1) represents the weight of the position of the cart and the element (3,3) represents the weight of the pendulum's angle. Knowing how to interpret the $Q$ matrix makes it possible to tune the weight parameters in the matrix, resulting in a $F$ matrix that outputs an optimal controller.




(a) $Q(1,1) = 1,\ Q(3,1) = 1$ (b) $Q(1,1) = 1500,\ Q(3,1) = 5000$

Figure 24: LQR Control Step-Responses

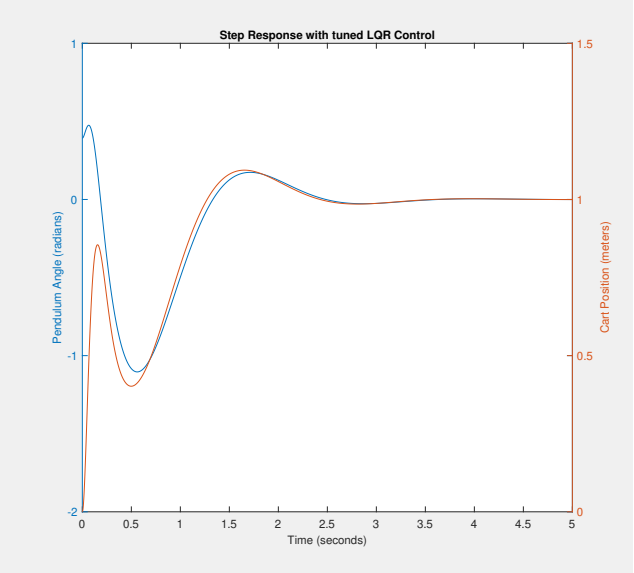
In figure 24a, the plot of the response without tuning the $Q$ matrix is shown. The system does not satisfy the specified requirements. The angle of the pendulum and the position of the cart slowly converges towards zero but the settling times need to be greatly reduced together with the pendulum's rise time. By increasing the weights (1,1) and (3,3) the settling time and rise time will both decrease. In figure 24b, it's seen that see the response of the system has vastly increased to the point of satisfaction, after changing the non-zero elements of the $Q$ matrix to weigh the two systems properly.

# 8 Simulated Tests

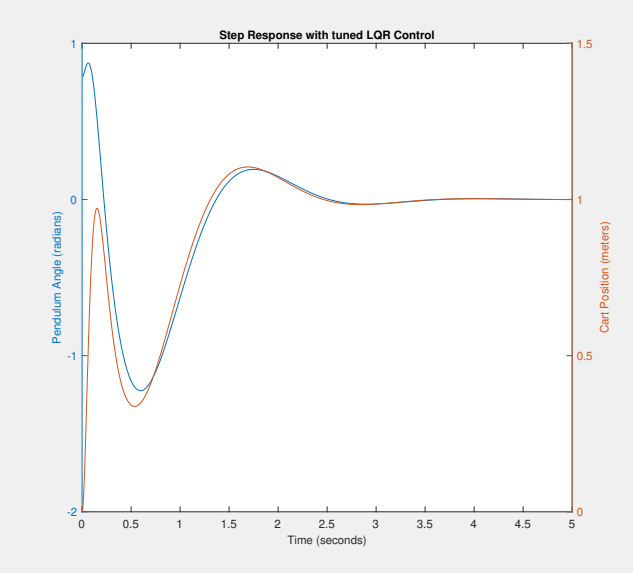
Now that a few different control structures have been made, some systematic tests are made to compare the different systems given different initial conditions. Two tests are made on two control structures, namely the continuous-time tuned LQR control and the discrete-time cascade PID control.

The tests are done by giving the pendulum a reference of $0$ [rad] and the cart a step input of $1$ [m]. The initial condition of the pendulum angle is in the first test $\pi/8$ and in the second test $\pi/4$.

## 8.1 Continuous-Time LQR Control




(a) Input-response for $\theta = \pi/8$




(b) Input-response for $\theta = \pi/4$

Figure 25: Continuous-Time Weighed LQR Control Tests

To rate these test results, the cart response is analyzed as it gets a step input. The performance is as follows

$\theta = \pi/8$

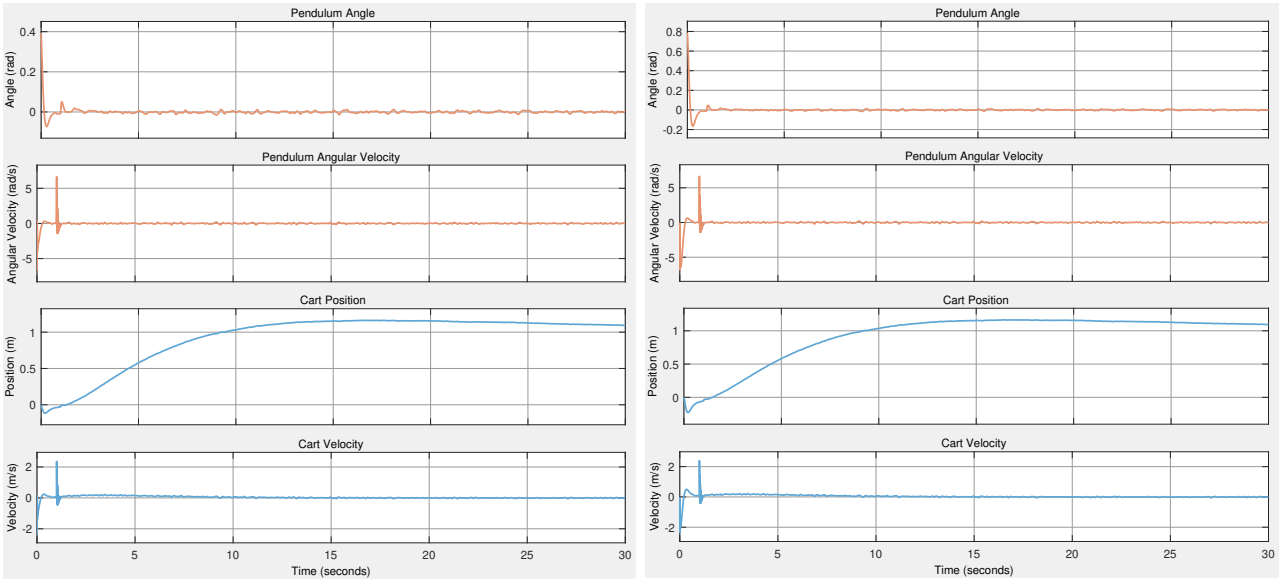
$$M_p = 9\ \% \qquad t_r = 1.2151\ [\text{s}] \qquad t_s = 1.06\ [\text{s}] \tag{42}$$

$\theta = \pi/4$

$$M_p = 10\ \% \qquad t_r = 1.2771\ [\text{s}] \qquad t_s = 1.12\ [\text{s}] \tag{43}$$

These tests show that even with a high initial condition for the pendulum such as $\theta = \pi/4$, the system performs well within the defined specifications.

## 8.2 Discrete-Time Cascade PID Control




(a) Input-response for $\theta = \pi/8$ (b) Input-response for $\theta = \pi/4$

Figure 26: Discrete-Time Cascade PID Control Tests

Where the performances of the two tests are as follows

$\theta = \pi/8$

$$M_p = 16.3\,\% \qquad t_r = 7.84\ [\text{s}] \qquad t_s = 19.737\ [\text{s}] \tag{44}$$

$\theta = \pi/4$

$$M_p = 16.5\,\% \qquad t_r = 7.89\ [\text{s}] \qquad t_s = 21.192\ [\text{s}] \tag{45}$$

These tests of the discrete-time cascade control system, show that all values increase as the initial pendulum angle increases. Another thing seen by the analysis of the performance of the system is that it does not comply with the requirements defined in section 5. Despite the system not complying with the specifications, this is the control system that will be implemented on the real-life system using the PLC.

# 9 Implementation on PLC

This section will contain an overview of the physical system as well as a presentation of the software in which the control system logic is designed. Some considerations regarding the motor control and a relation between the analog motor input and the physical force on the system is developed. The main focus of this section is to test a classical control cascade structure on the physical setup and to find the optimal gains for the controllers. Moreover, a swing up will be implemented and tested.

## 9.1 Hardware

The physical system consists of a motor connected through gearing to a conveyor belt on which a cart is located. Two limit switches are located close to the edges of the conveyor belt. The full setup can be seen in figure 27. The system is controlled by a Schneider M241 PLC. The PLC is programmed in Structured Text (IEC 61131-3) through Schneider's own software *EcoStruxure Machine Expert Logic Builder* (MELB).

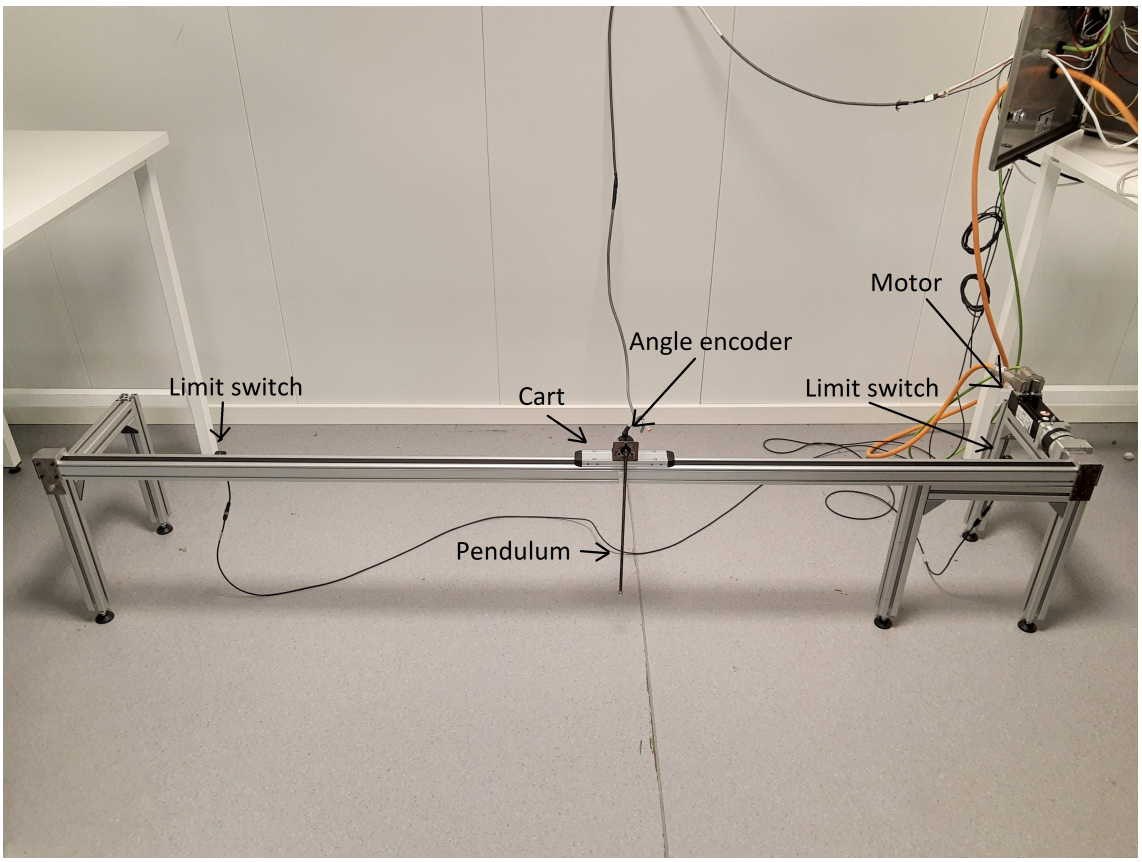



Figure 27: The physical setup

## 9.2 Motor Control

In order to control the motor, a signal $f$ between $-10,000$ and $10,000$ is sent from the PLC to the motor. The relation between this value and the cart velocity is found experimentally by using different input values and measuring the time it takes the cart to cover a specific distance. Specifically, the cart is moved from one limit switch to the other with different $f$ values. When using this method, we already take into account the frictional force between the cart and conveyor as well as the internal friction of the motor.

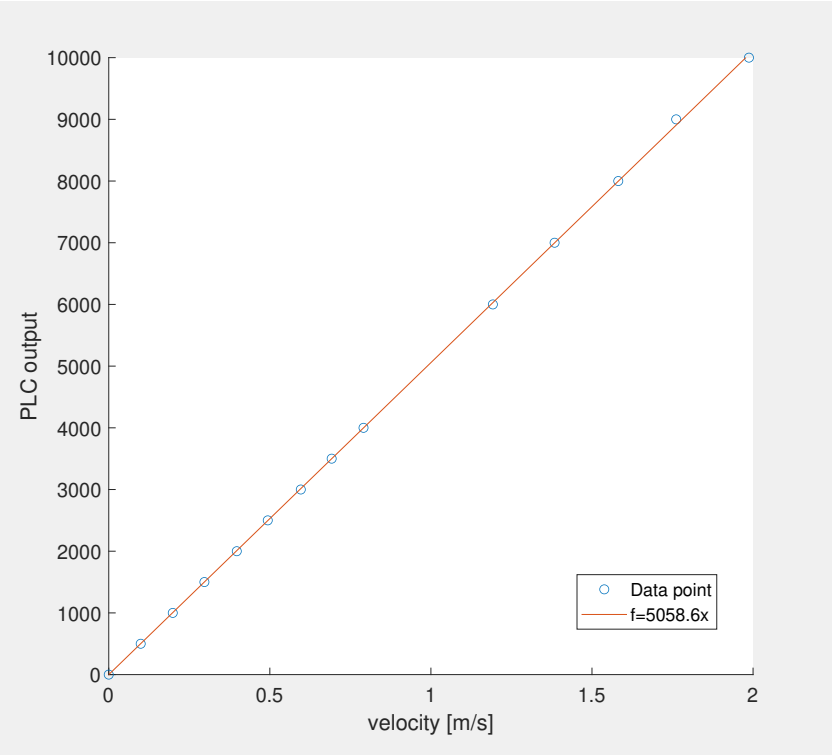



Figure 28: Plot of the input $f$ as a function of the cart velocity $\dot{x}_c$

Using linear regression, a function $f(\dot{x}_c)$ is found, which calculates the input $f$ to reach a desired velocity $\dot{x}_c$.

$$f(\dot{x}_c) = 5058.6 \cdot \dot{x}_c \tag{46}$$

In the modeling of the system, the output of a PID controller is a force. As the motor is using speed-control, this force has to be converted to a velocity $\dot{x}_c$. By simplifying the problem the acceleration can be calculated as follows

$$\ddot{x}_c = \frac{F}{m_p + m_c} \tag{47}$$

This is disregarding the varying contribution the pendulum has on the system depending on its angle. Therefore this is not the optimal solution to the problem, but given the relatively low mass of the pendulum compared to the cart the relation is regarded as good enough. With acceleration know the change in velocity $\Delta\dot{x}_c$ can be calculated using the PLC cycle time $T_{cycle}$.

$$\Delta\dot{x}_c = \ddot{x}_c \cdot T_{cycle} \tag{48}$$

Where $m_p = 0.084$ [kg], $m_c = 0.5$ [kg] and $T_{cycle} = 0.001$ [s].

## 9.3 Program overview

MELB has a built-in PID-function block, which takes a proportional gain, an integral gain, a derivative gain, a reference value, and the current value of the process variable as inputs, and outputs a control signal. This block is used in a cascade structure, with the pendulum controller being the inner controller, and the cart

controller being the outer controller as seen in figure 6. The control signal from this cascade implementation is then converted to a change in velocity using the relation shown in equation 47.

### 9.3.1 Program Structure

The program implemented on the PLC has a startup phase, where the motor driver is configured and the middle of the conveyor belt is found and defined using a homing motion. Then, the program starts either the control sequence or the swing up sequence, which is introduced in section 9.6.2.

The control sequence starts the controller once the pendulum reaches the upright position. The output of the controller is converted to a change in velocity and this velocity is converted to a motor input as explained in section 9.2. If the velocity exceeds the motor limit, it is set to the maximum value of the motor. If the numerical value of the angle gets too big, the controller is stopped. As a safety measure, if any of the limit switches are activated, the motor is stopped. See appendix figure 1 and 2 for diagrams of the program structure.

## 9.4 Testing of Simulated Controller Gains

The gains tested in this section, are the best theoretical cascade gains found in the classical design section 5.3.2.2. The gains are tested by logging the position of the cart and angle of the pendulum every $0.01[s]$. The data from this test is plotted in figure 29 below. The test can also be seen on "video1.mp4" The graph shows, that the system is very unstable, as the angle quickly diverges from 0. The test is automatically stopped after 0.55 seconds as the angle gets too steep.

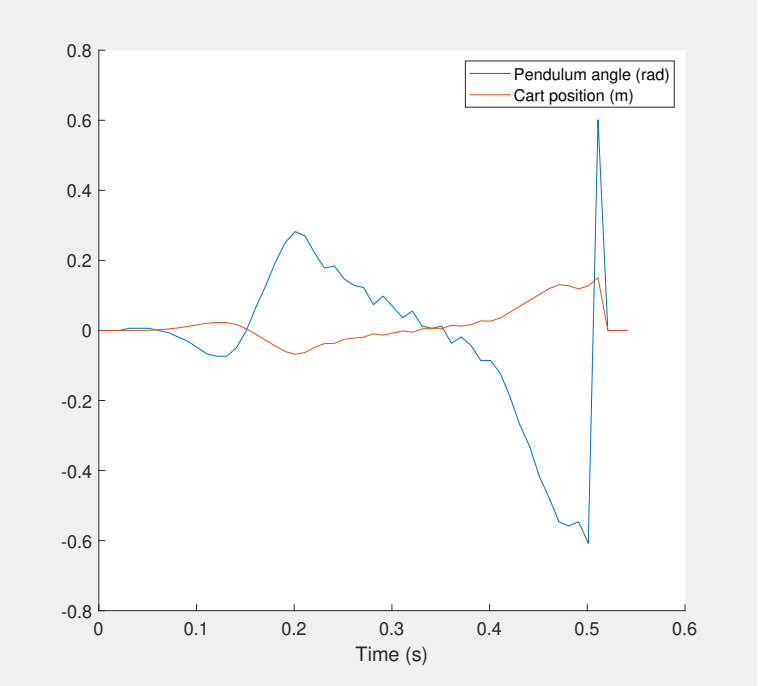



Figure 29: Input-response of the system using the simulated gains of the cascade structure

From the test it is concluded that the simulated cascade gains are not capable of stabilizing the system. It's suspected that this is due to the lack of parameter estimation, as this would result in a big difference between

the simulation and the real system. In an attempt to make the system stable, the inner PID is tested separately to find out if the pendulum controller is stable on its own. If this is true, then only the outer controller needs to be manually tuned. The test from the inner PID is seen on figure 30, aswell as "video2.mp4".

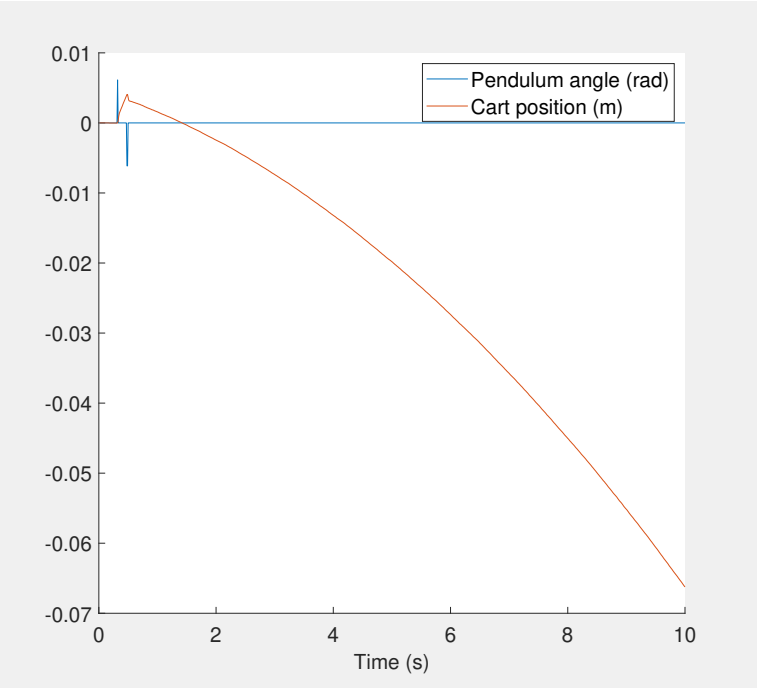



Figure 30: Input-response of the system using the simulated gains of inner PID controller

This test shows that the inner PID gains are sufficient for stabilizing the pendulum. The cart drifts until it hits one of the end-stops since the outer cart controller is disabled. As the simulated inner controller cause the pendulum to become stable, the objective is now to manually tune the cart controller.

## 9.5 Cart Controller Tuning

This section will focus on systematically tuning the cart controller until the entire system is stable. This results in a 3-dimensional problem. To reduce the complexity of finding the optimal gains for the outer controller, the design procedure presented in figure 31 is followed. Firstly a P controller is tuned according to the design procedure. When a sufficient P controller has been found, the proportional gain is used in a PI controller, where the objective now is to tune the integral gain following the same design procedure. Lastly, when a PI controller has been tuned, these gains are used in a PID controller with the derivative gain being the final unknown.

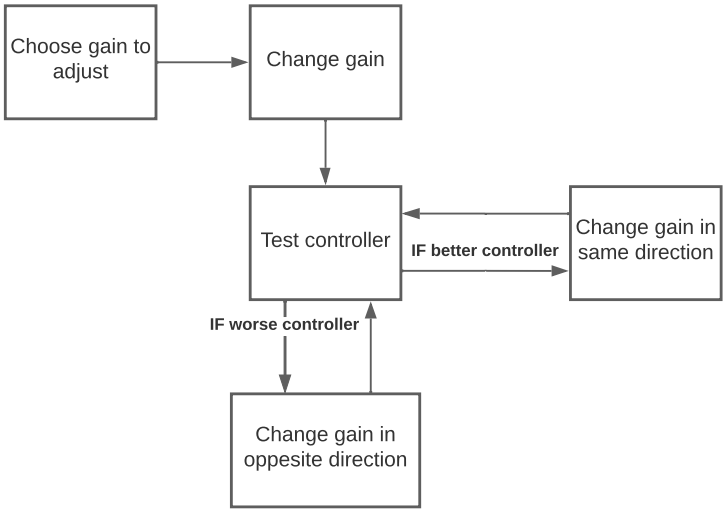



Figure 31: The general process for tuning PID gains

This systematic tuning resulted in a stable cascade controller which was able to hold the pendulum in an upright position while also stabilizing the cart towards the middle of the system - even with -0.1 [m] offset as the initial condition. All the tests and evaluations leading up to this first stable controller can be seen in Appendix B.

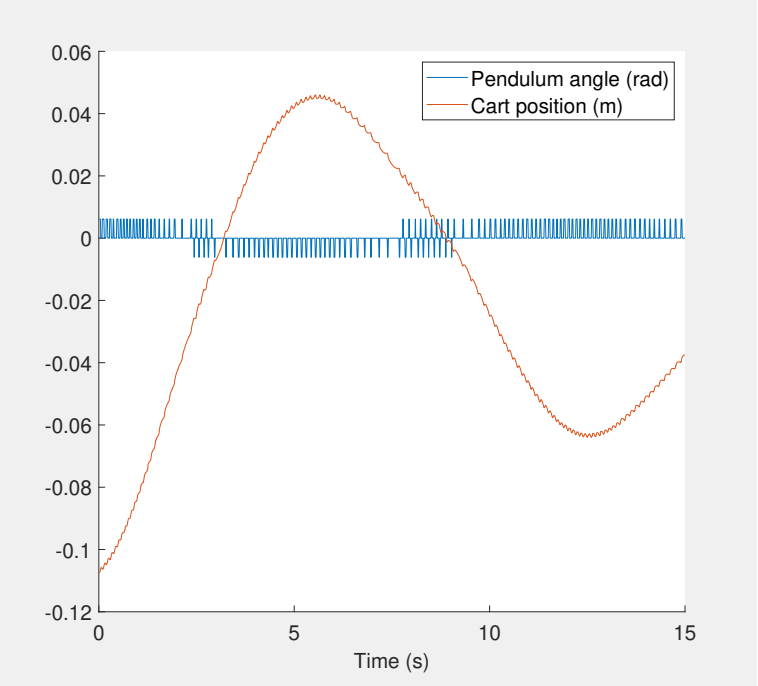



Figure 32: First stable controller

The graph in figure 32 shows our first stable controller, tested with an initial cart position of approximately -0.1 [m]. It is clear that the oscillations of the cart slowly become smaller, and eventually, the position of the cart will converge to 0 [m]. This can be seen on "video3.mp4". The pendulum controller is not challenged by the inadequate cart controller, thus the pendulum is able to stay almost constantly at the reference angle.

The conclusion on the tested PID controller in figure 32 is that a more responsive cart controller is desired. The following tuning process of the cart controller is continuously carried out as isolated tests and analysis of the individual gains. Contemplating a set of rules, the individual gains are tuned until the desired behavior of the system is achieved. A more responsive and aggressive movement of the cart is achieved by increasing the proportional gain. A reason to decrease the proportional gain would be that the required velocity to stabilize the system could exceed the physical limitations of the motor. This disadvantage counts as well for an over-increased integral gain. This problem could be solved if an actuator model and anti-windup system would be implemented. However, this would not be trivial and the reason why will be explained in detail in the discussion section.

The integral gain causes the cart to move more aggressively towards the reference. Therefore, picking an optimal value would result in a slight decrease in overshoot, and a decrease in settling time. The final tuning option is the derivative gain. This gain provides an anticipatory reaction to the system based on the tendency of the error. This means that theoretically when the derivative gain is increased the overshoot is decreased. However, increasing the derivative gain too much can result in the system over-correcting the anticipated error. The aforementioned also makes the controller vulnerable to noise and thus a derivative gain that is too big can result in the system becoming unstable. The derivative term is especially vulnerable to high-frequency noise, thus implementing a low-pass filter could mitigate this problem.

This knowledge has been used to further tune the controller to improve the response. This resulted in a new set of tests starting from test #15 in the *PID test* in appendix. The final cart gains, which perform the best, are as follows.

$K_C$ gains

$$K_p = 0.4 \qquad K_i = 8 \qquad K_d = 0.036 \tag{49}$$

The original pendulum gains are kept the same

$K_P$ gains

$$K_p = 142.91 \qquad K_i = 623.09 \qquad K_d = 8.1459 \tag{50}$$

The final gains for the controller are tested, and the data is shown in figure 33. Here, the logged data of pendulum angle and cart position is based on a test where the system is left in its most stable state, near the cart controller reference, and without any external forces added to the system.

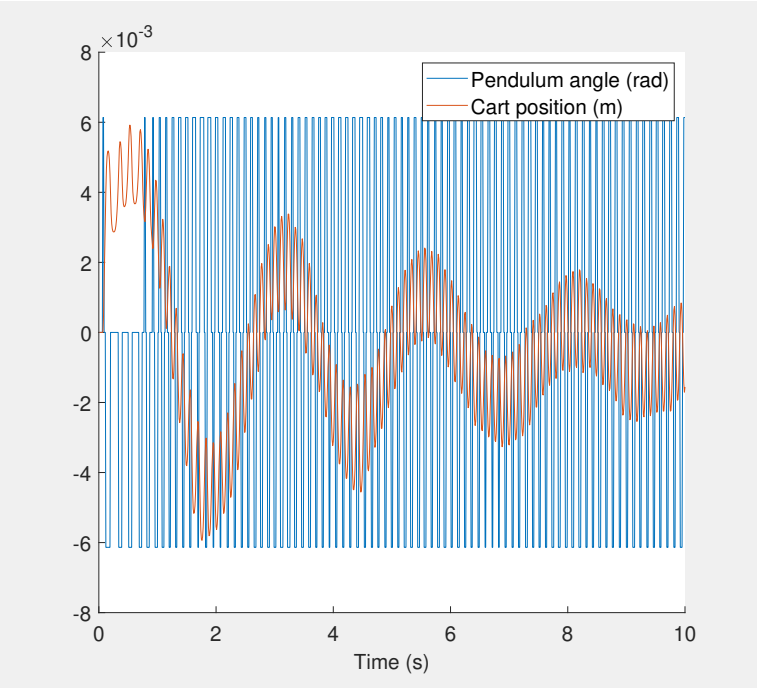



Figure 33: Final controller - No disturbance

The graph shows that the pendulum stays close to the angle reference, namely at zero radians. The deviations have a value of either $-6 \cdot 10^{-3}$ or $6 \cdot 10^{-3}$ [rad]. This is due to the resolution of the encoder.

To further test the stability of the final controller the cart is given an initial condition of $-0.11$[m], along with the reference of 0 [m]. This test is shown on figure 34.

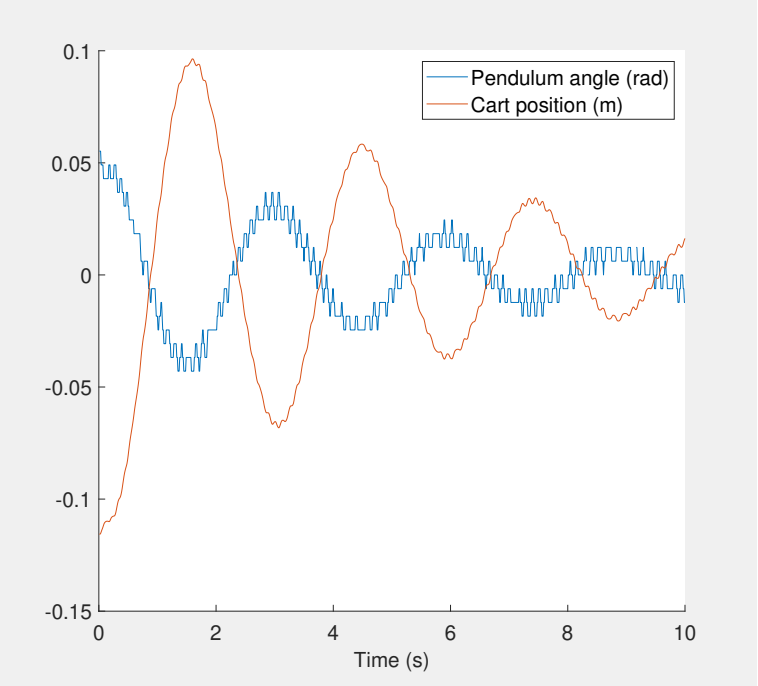



Figure 34: Final controller - With disturbance

In figure 34 it is shown that both the cart position and pendulum angle converges towards their reference. This

can also be seen on "video4.mp4" Comparing the final controller on figure 34 to the previous stable controller on figure 32 it's seen that a faster settling time can be obtained on the final controller, at the cost of a larger overshoot.

## 9.6 Swing-Up

A swing-up method is added as an extra feature to the existing system. In the following section, the combination of the control system and the swing-up method is reviewed.

### 9.6.1 Theory

The swing-up control theory is based on the article "Swing-Up Methods for Inverted Pendulum" written by Professor Brock [10].

Brock explains a classical swing-up strategy based on energy control. The purpose of the model is to pump energy into the pendulum until the pendulum reaches the upright position. Brock presents the equation of motion as well as an equation for the total energy of the pendulum and lastly derives equation 51. It is stated that the total energy of the pendulum is growing if

$$\ddot{x}_c = -\ddot{x}_{c\_max} \cdot \text{sign}(\dot{\theta}) \cdot \text{sign}\left(\cos\left(\theta\right)\right) \tag{51}$$

Where $\ddot{x}_{c\_max}$ is the maximum allowed acceleration of the cart. In practice this would result in a constant acceleration of the cart in either the negative or positive direction, depending on the angular velocity and the angle of the pendulum.

### 9.6.2 Practical Implementation

Before starting the swing up method, the cart is moved to the edge of the conveyor belt and any movement of the pendulum is stopped. This is done to give as much length for movement as possible and to ensure the initial condition for the pendulum is the downright position. From this position, the method is activated, and when the pendulum is close to the upright position, the PID controller is given control of the system. If the angle then gets too big, the pendulum is considered lost and the swing up method is activated again.

### 9.6.3 Testing

First, the swing up is tested with the cascade control found in Equation 49 and Equation 50. The swing up method worked and the pendulum achieved the upright position, but the angular velocity was too big for the controller to catch the pendulum.

As the cascade controller is slightly worse at keeping the pendulum upright than the pendulum PID controller, the same test was performed on the pendulum controller. This is seen on figure 35. From this figure, it can be seen that the pendulum controller was able to catch and stabilize the pendulum after approximately 2.5 [s] for a total duration of 0.7 [s]. The test can also be seen on "video5.mp4", where the period when the controller stabilizes the pendulum is very obvious.

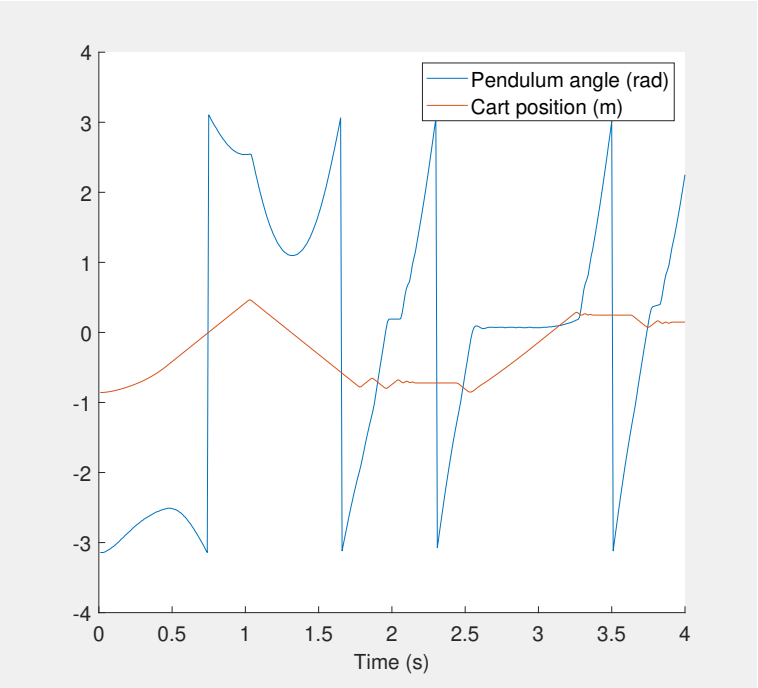



Figure 35: Swing up with pendulum control. The angle is wrapped between $-\pi$ and $\pi$

## 10 Discussion

When modeling the system, several decisions have been made with the purpose of simplifying the system, which has the downside that they could lead to some errors. First off, in order to use methods that assume linear time invariance, the system has been linearized. This means that the model of the system is only accurate around the equilibrium point, and gets less and less accurate when deviating more from it. This is potentially a source of errors. Foreover, the parameters given by the project supervisors do not accurately match the actual parameters of the system. In order to minimize that problem, a parameter estimation should be made, but even then, there would remain some measurement uncertainties. Furthermore, the system is assumed to have no structural irregularities. For instance, the conveyor belt is assumed to be perfectly parallel to the ground, the pendulum rod is assumed to be a straight line and the pendulum is assumed to be perpendicular to the cart. Additionally, the conveyor belt bends a bit in the middle, which causes the cart to always slide to the middle. Finally, there are also some sources of error when developing the classical and modern controllers. First off, the transfer function and state-space model are derived from the system modeling, so the errors from the modeling segment carry over. Secondly, the frequency-domain performance specifications are only guiding for higher order systems.

When moving from the simulated environment to the real setup several errors can occur. First off, anti-windup and an actuator model have in our case been neglected, since the controllers are tuned so that the motor limitations are never exceeded when doing regular tests. However, this will could be a problem if more extreme tests were to be made. Second off, the cycle time of the PLC, and therefore also the sampling time, has a vital role in the accuracy of the system. In our case, a value of 0.001 [s] has been chosen, and as it can be seen on both video and graphs, this is sufficient. For instance, no information is lost due to a big cycle time.

Before implementing anything on the PLC, we had to decide whether we should use cascade- or parallel control. The cascade control system completely failed to stabilize after implementing the classical swing-up method. The most obvious reason is that the controller needs a higher level of responsiveness. This could be achieved by increasing the P-gain, and the I-gain correspondingly, but as mentioned in 9.5, the increase in control gains may cause the control system output to exceed physical limitations on the motor. The output is saturated to fit the maximal input to the motor, however, this will cause a wind-up error to grow over time. This problem is usually handled by implementing anti-windup in the PI controller. However, since the control system is modeled to output a force, and the physical system takes a velocity variable as input, the error between those can not be calculated and used to implement anti-windup.

Since the responsiveness of the cart control was too weak, we tested the swing up method with the pendulum controller by itself. However, the energy of the pendulum was still too big, and the controller was only able to catch the pendulum for a very short timespan. A solution to this problem would be to implement another swing-up method where the pendulum comes to a near standstill when at the upright position. The slower pendulum could be achieved by having a slower cart movement, and with the current implementation, the slowest cart movement is still too high. By using an implementation that accounts for the length of the conveyor belt, it should be possible to decrease the speed enough so the controller can catch the pendulum.

# 11 Future Work

The next logical step on the theoretical side of the project is to research and test Fuzzy-Logic Control (FLC), which is a control style that can be applied to anything from industrial machine control to biomedical instruments and even has some security applications. FLC usually performs much better than other control techniques, when utilized in complex systems which are only vaguely defined. This can be attributed to the fact that FLC systems are made to emulate human deductive thinking, to the point where it can be defined as AI. [11, 12]

To improve on the swing-up, the implementation proposed by Stefan Brock [10], utilizing a specified conveyor length, could be implemented. This method accounts for both the distance the cart is allowed to move and its maximum velocity. With this, it should be possible to make sure the pendulum gets a relatively low speed when it reaches the upright position, and thus it should be easier for the controller to catch it.

# 12 Conclusion

In this project, we have stabilized the inverted pendulum. Developing a rigid body simulation was the first step in settling on a cascade controller structure. Thereafter, a dynamical model of the system was derived using Euler-Lagrange equations. This model was then linearized and put on the state-space form, and transfer functions were obtained. The model has then been verified by comparing impulse responses for different linearization points. Based on the linearized equations, both modern and classical control methods have been utilized, and the respective controllers simulated. Moreover, the responses of these controllers have been compared to the performance specifications of the system.

Two PID controllers have been implemented in a cascade structure on the physical system. The inner controller regulates the angle of the pendulum and the outer controller handles the position of the cart. The system is considered stable as it does not diverge within the 2-minute timeframe, as the disturbance rejection and reference tracking of the system complies with the specifications, such as being able to recover from a 10 centimeter initial condition on the cart, within 2 minutes. The swing-up implementation is capable of getting the pendulum into the upright position, but with a too high angular velocity so the controller is incapable of catching it.

# 13 References


(1) Ramírez, E. M. Study and design of the control system of an inverted pendulum, Thesis, 2018, https://bit.ly/3wJBvg2.

(2) Van Oosterhout, R. Inverted Pendulum Demonstrator, Thesis, 2018, https://bit.ly/3kW1RFX.

(3) Hamza, M.; u. Rehman, Z.; Zahid, Q.; Tahir, F.; Khalid, Z. Real-Time Control of an Inverted Pendulum: A Comparative Study, Article, 2011, https://bit.ly/3MO7HWc.

(4) For MATLAB, C. T.; Simulink Inverted Pendulum: Simscape Modeling, Online, https://bit.ly/3r3Glm7.

(5) Control Station What Is Cascade Control? How Is Cascade Control Configured?, Online, 2014, https://bit.ly/3GwaIYY.

(6) McMillan Cascade Control Perspective Tips, Online, 2014, https://bit.ly/38V3xNC.

(7) Matlab, MathWorks Inc. ss2tf, Online, https://bit.ly/3z5Bvd7.

(8) Matlab, MathWorks Inc. pidtune, Online, https://bit.ly/3sUt1RR.

(9) Matlab, MathWorks Inc. lqr, Online, https://bit.ly/3LMCktJ.

(10) Brock, S. Swing-up methods for inverted pendulum. In: Int. Conf. Electrical Drives and Power Electronics, 2003, DOI:10.6084/m9.figshare.697505.

(11) ScienceDirect Fuzzy-Logic Control, Online, 2022, https://bit.ly/3MdDU8x.

(12) GeeksforGeeks Fuzzy Logic Control System, Online, 2020, https://bit.ly/38Vcbvz.

# Appendix

## Appendix A

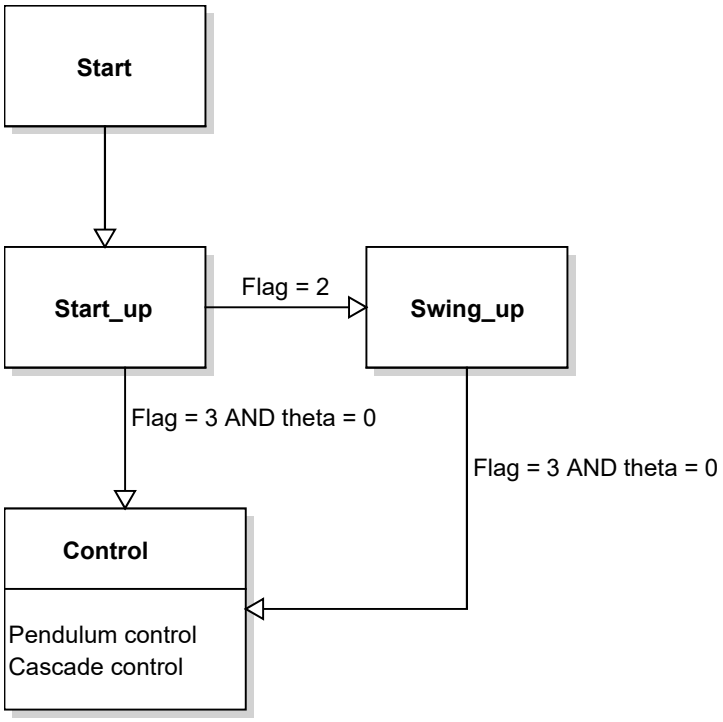



Figure 1: Architecture of the implemented program (Flag is specified by the user before the program is run)

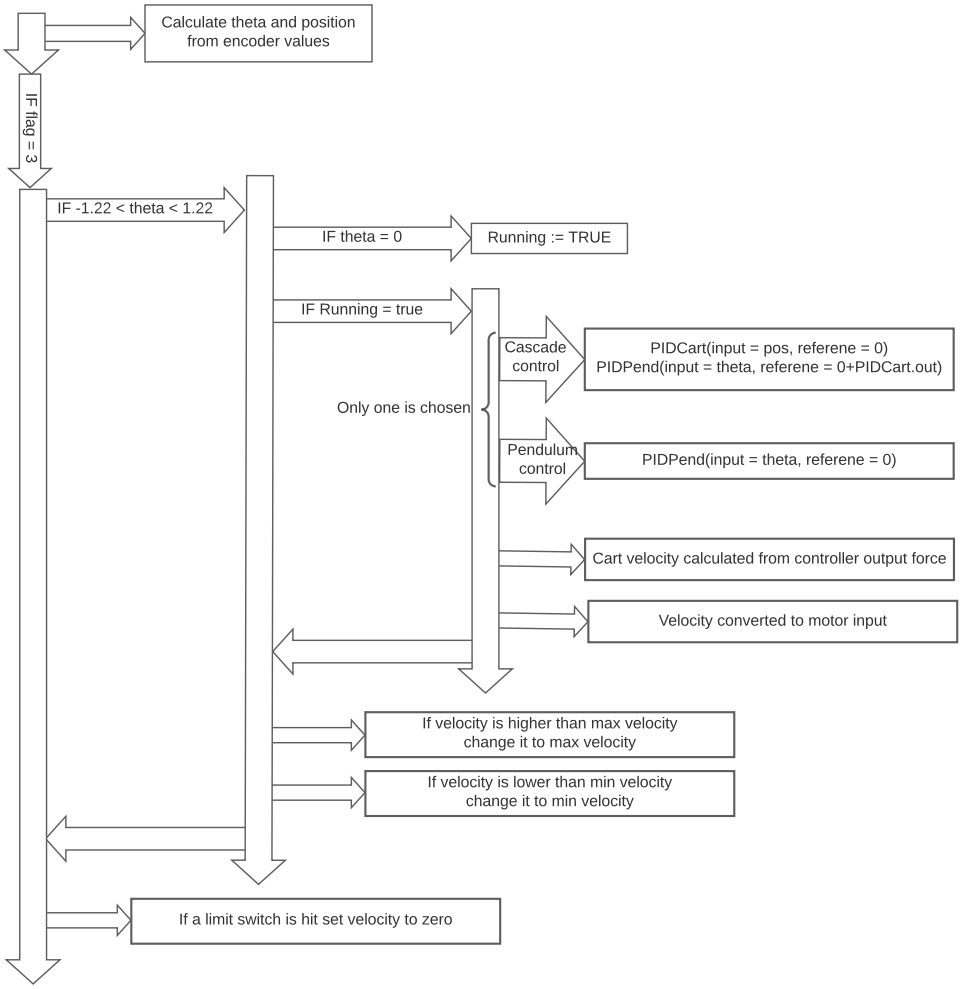



Figure 2: The control sequence of the program for every cycle

## Appendix B

Note that the integral and derivative gains is on a different notation than the one used elsewehere in the report. This is due to the way the Schneider PID block works. The relation is as follows:

$$K_i = T_i \cdot K_p \tag{52}$$

$$K_d = T_d \cdot K_p \tag{53}$$

| Test nr. | K_p | T_i | T_d | Timing | Observationer |
|---|---|---|---|---|---|
| **#1** | 0,0247 | 0,0523 | 4,6839 | | Chaotic system. We make an isolated test of the pendulum controller and we conclude that something is wrong with the cart controller gains. We then make an isolated test of K_P. |
| **#2** | 0,0247 | 0 | 0 | 2m 11s | The pendulum stays upright and the cart changes direction when passing the reference. However, the overshoot is too large, and the limits are hit. |
| **#3** | 0,015 | 0 | 0 | 1m 9s | Larger overshoot. Vi hit the limits quicker, and conclude that K_P should be a higher value. |
| **#4** | 0,035 | 0 | 0 | 1m 59 | A bit smaller overshoot, but not better than the test made with the original K_P value. We try a value somwhere between these two. |
| **#5** | 0,03 | 0 | 0 | 2m 20s | Longest time without hitting limits. We choose to go with this K_P, and move on to test T_I values. |
| **#6** | 0,03 | 0,0523 | 0 | 5.5s | We start with T_I from the theoretically best gains (simulated). The pendulum quickly falls over, but the cart has a larger respons and regulates more towards the reference. We try with a smaller T_I value. |
| **#7** | 0,03 | 0,005 | 0 | 3s | The pendulum falls over faster, and the cart control is worsened. We test a significantly larger T_I value. |
| **#8** | 0,03 | 5 | 0 | 8,5s | The pendulum stays upright for longer. Better regulation of the cart. We test yet a larger value for T_I. |
| **#9** | 0,03 | 50 | 0 | -\|\|- | The pendulum and the cart is stable. |
| | | | | | **WE CONTINUE TO TEST THE SYSTEM BY ADDING AN OFFSET TO THE CART OF 8.5 CM (HALF A CART LENGTH FROM THE REFERENCE)** |
| **#10** | 0,03 | 50 | 0 | 2m 11s | Relatively large overshoot when adding the offset. The oscillations become larger until we eventually hit the limits. |
| **#11** | 0,03 | 200 | 0 | 9m 9s | The overshoot grows slower, but still eventually hits the limits. |
| **#12** | 0,03 | 1000 | 0 | -\|\|- | The cart oscillates for very long, and the overshoot grows very little for every time that the cart passes the reference. |
| **#13** | 0,03 | 10000 | 0 | -\|\|- | More or less same behavior as the previous test. Maybe marginally better, but not visible by sight. We choose to move on to testing T_D with T_I = 1000. |
| **#14** | 0,03 | 1000 | 4,6839 | 0s | We test T_D from the simulated system. Not good - instant kaos. We try a lower value for T_D. |

| Test nr. | K_p | T_i | T_d | Timing | Observationer |
|---|---|---|---|---|---|
| | | | | | **THE FOLLOWING VALUES FOR TIMIMG INDICATES WHEN THE SYSTEM STABILIZES AT THE CART REFERENCE, WHEN STARTING THE SYSTEM WITH AN OFFSET OF APPROX. 82 CM. (HALF THE LENGHT OF THE CONVEYOR)** |
| **#15** | 0,03 | 1000 | 0,5 | 45s | Stabillity is obtained after 45 seconds. We evaluate that we need a more powerful controller. Thus we make K_P bigger and adapt T_I and T_D to the new K_P |
| **#16** | 0,06 | 500 | 0,2 | 35s | Stabillity obtained after 35s. We test with an even bigger K_P-value |
| **#17** | 0,3 | 100 | 0,05 | -\|\|- | Faster respons and generally a more aggressive controller. This also means a larger overshoot, and we instantly hit the limits. We test with a larger value for T_D. |
| **#18** | 0,3 | 100 | 0,1 | | Better results but the overshoot is still too big. |
| **#19** | 0,3 | 100 | 0,2 | | The system is technically stable, but the pendulum oscillates due to T_D which is too large. We must conclude that 0.1 is the best value for T_D, and we decrease T_I, to reduce the overshoot. |
| **#20** | 0,3 | 50 | 0,1 | | We no longer hit the limits, but we still desire to have a faster response, and try to increase the value of K_P. |
| **#21** | 0,6 | 25 | 0,05 | | The output of the control system becomes too big for the physical motor. K_P needs to be decreased. |
| **#22** | 0,45 | 33,3 | 0,0667 | | K_P is now good, meaning that we have a fast response of the system. This is the maximum value for K_P that the system can take, without the output force of the system exceeding the motors capacity. However, the cart hits the limit switches. We test with a smaller T_I value to decrease overshoot. |
| **#23** | 0,45 | 20 | 0,0667 | | Still hitting the limits, we test again with a larger T_D. |
| **#24** | 0,45 | 20 | 0,08 | | We no longer hit the limits, but are still very close. We try to decrease T_I further. |
| **#25** | 0,45 | 10 | 0,08 | | Feels more stable when moving the pendulum around, but the system exceeds the motors capacity and the cart hist the limits. We try to decrease K_P. |
| **#26** | 0,4 | 11,25 | 0,09 | | We no longe exceed the motors capacity, but still hit the limit switches. We increase T_I to make the cart reference stronger. |
| **#27** | 0,4 | 20 | 0,09 | | Smaller overshoot and stronger response. We conclude that these make the optimal controller gains. |