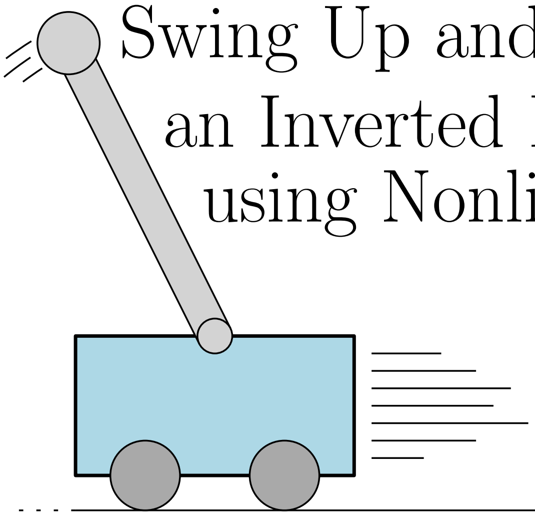


Swing Up and Stabilisation of an Inverted Pendulum on a Cart using Nonlinear Methods

Jesper Hede Christensen

Rasmus Christiansen

Master's Thesis
Control and Automation
Department of Electronic Systems
June 2017

AALBORG UNIVERSITY
STUDENT REPORT

Master's Thesis – 4th Semester
Faculty of Engineering and Science
School of Information and
Communication Technologies
Control and Automation
Fredrik Bajers Vej 7C
9220 Aalborg East
webinfo@es.aau.dk

Title:

Swing Up and Stabilisation of an Inverted Pendulum on a Cart using Nonlinear Methods

Theme:

Nonlinear control

Project period:

Spring 2017
2017-02-01 – 2017-06-08

Group number:

1030

Group member:

Jesper Hede Christensen
Rasmus Christiansen

Supervisors:

John-Josef Leth
Palle Andersen

Pages: 112

Appendices: 8

Concluded: 2017-06-08

Abstract:

This thesis describes the process of designing a nonlinear control scheme for swinging up and stabilising an inverted pendulum on a cart.
A model of the cart and pendulum dynamics has been derived using Lagrangian mechanics. The actuator dynamics are modelled by a first order transfer function. The combined model has been implemented in Simulink and verified. An Extended Kalman Filter has been developed to estimate the full state vector.
A two-step control strategy has been developed, which consists of *swing-up* controllers and *stabilising* controllers. A *catch trigger* algorithm manages the transition between the controllers. Swinging up the pendulum concerns controlling the energy of the system such that the pendulum reaches its heteroclinic orbits.
The stabilising controllers keep the cart close to a position reference while keeping the pendulum upright at its unstable equilibrium. Six controllers; three swing-up and three stabilising, have been designed, implemented and compared.
Results indicate minimal difference between the stabilising controllers when considering their ability to keep the pendulum upright and the cart at a desired position.

Preface

This Master's thesis is produced by group CA10-1030 and concludes the fourth and final semester in the Master's programme Control and Automation at the Department of Electronic Systems, Aalborg University.

Citation before a full stop regards sentence and after full stop at the end of a paragraph regards the entire paragraph. Chapters or sections starting with "The following is based on [Reference]" means the entire chapter or section are based entirely on that source.

The .zip-file handed in along with this document contains the source code developed during the course of the project:

- MATLAB/Simulink model
- C/C++ implemented code

Thanks to John-Josef Leth and Palle Andersen from Aalborg University for great support and supervision during the project.

Jesper Hede Christensen

Rasmus Christiansen

Figure 1: *The members of group 17GR1030.*

Resumé

Dette projekt omhandler ulineært reguleringsdesign af et omvendt pendul på en vogn, og forsøger at besvare problemstillingen:

Hvordan kan et ulineært reguleringsdesign udvikles således det kan svinge det omvendte pendul på vognen op og stabilisere det, og hvordan klarer den ulineære, stabiliserende regulering sig med lineære reguleringsmetoder?

For at besvare dette er et fysisk inverteret pendul på en vogn, som står tilrådighed i reguleringslaboratoriet på Aalborg Universitet, blevet analyseret og modelleret. Den mekaniske del af modellen er udledt på baggrund af Lagrange-mekanik, og den elektriske del er approksimeret ved en første ordens overføringsfunktion. Den mekaniske model er opstillet på tilstandsform, og et udvidet Kalman-filter er designet således alle tilstande kendes.

Den overordnede reguleringsstrategi er baseret på to faser, en opsvingende og en stabiliserende fase. Dertil er der designet en algoritme, som varetager overgangen mellem faserne, styret af pendulets vinkel og den mekaniske energi i pendulet.

Designet af reguleringen til opsvinget er baseret på mekanisk energi, og forsøger at tvinge pendulets faseportræt ud mod dets heterokliniske kredsløb. Pendulets friktion er udeladt fra designprocessen, hvilket gør at heuristiske metoder har vist sig nødvendige til at svinge pendulet op til dets ustabile ligevægtspunkt. Der er designet tre forskellige metoder til opsving.

Idet pendulet nærmer sig sit ustabile ligevægtspunkt, og kriterierne for overgangsalgoritmen er opfyldt, vil den stabiliserende regulering overtage styringen. Til dette er der designet et kaskadekoblet PID-reguleringssystem, en LQG-regulator og en sliding-mode regulator, hvis formål er at stabilisere pendulet i dets ustabile ligevægtspunkt, samt at holde vognen nær en ønsket position.

I alt er der designet, implementeret, testet og sammenlignet seks forskellige reguleringsmetoder på det fysiske system. Til opsving viser metoden baseret på pendulets energi alene mest lovende resultater. Derudover viser resultaterne minimal forskel mellem de stabiliserende reguleringsdesigns.

Definitions

Notation

Throughout this thesis, general mathematical notation is used. However, in order to avoid ambiguity, the following conventions are used:

- Vectors and matrices are denoted in bold, lowercase and bold uppercase text, respectively, e.g. $\mathbf{x}$ and $\mathbf{A}$.
- Trigonometric functions $\sin(\cdot)$ and $\cos(\cdot)$ taken with respect to one variable will have the parentheses omitted, e.g. $\sin x = \sin(x)$, but $\sin xk = \sin(x) \cdot k$.
- Variables sampled at time step $k \cdot T_s$ will be subscripted x_k.
  Variables sampled at time step $k \cdot T_s$ and based on data from the previous sample will be subscripted $x_{k|k-1}$, or, if based on data from the current sample, $x_{k|k}$.
- Matrix indices will be referred to by $\mathbf{M}_{(i,j)}$, meaning the index at row i and column j of matrix $\mathbf{M}$.
- A matrix size is indicated by $\mathbf{M}_{a \times b}$, where a and b denotes the row and column length respectively.
- Variable references are denoted with a superscripted asterisk, e.g. x^*.
- Variables estimates are denoted with a hat, e.g. $\hat{x}$.

Acronyms

ARM Advanced RISC Machine.

back-EMF back-electromotive force.

DAC Digital to Analog Converters.

EKF Extended Kalman Filter.

LQG Linear-Quadratic-Gaussian.

LQR Linear-Quadratic Regulator.

MCU Microcontroller Unit.

MSE Mean Square Error.

PCB printed circuit board.

PID Proportional-Integral-Derivative.

RMSE Root Mean Square Error.

SMC Sliding-Mode Controller.

SRAM Static Random-Access Memory.

Nomenclature

Table 1 lists the most commenly used variables and constants.

Symbol	**Description**	**Unit**
x	Cart position	m
θ	Pendulum angle	rad
$\dot{x}$	Cart velocity	$\frac{\mathrm{m}}{\mathrm{s}}$
$\dot{\theta}$	Pendulum velocity	$\frac{\mathrm{rad}}{\mathrm{s}}$
u	System input	F
$\mathbf{x}$	State vector	–
$\mathbf{r}$	Position vector	–
$\mathbf{q}$	Generalised coordinate vector with x and θ	–
$\mathbf{M}$	Inertial properties matrix	–
$\mathbf{C}$	Centrifugal force vector	–
$\mathbf{B}$	Friction vector	–
$\mathbf{G}$	Gravitational force vector	–
F	Force	N
E, E_p	System energy and pendulum energy	J
c	Coulomb friction	F
v	Viscous friction	F
g	Gravitational acceleration	$\frac{\mathrm{m}}{\mathrm{s}^2}$
m_c, m_p	Cart and pendulum mass	kg
l	Pendulum length	m

Table 1: *List of most commen variables and constants used throughout the thesis.*

Contents

Introduction 1

In April 2014, aerospace manufacturer *SpaceX* launches their *Falcon 9* rocket on a resupply mission to the International Space Station. The first stage, the *booster stage*, manages to perform a controlled splashdown after separation from the rest of the rocket. Later, in January 2015, SpaceX attempts a full, vertical landing of the booster stage onto a drone ship. While this ends with a crash, it has been attempted several times since, and SpaceX succeeded in landing the booster stage on land in December 2015.[1]

Figure 1.1: *Falcon 9 booster stage landing on the "Just Read the Instructions" drone ship.[2]*

Landing a rocket vertically is difficult, as it involves decelerating, manoeuvring and balancing the rocket at the same time. Deceleration is achieved by providing thrust in the opposite direction of the velocity vector of the rocket, but manoeuvring requires angling the thrust vector, and this produces both linear and rotational acceleration, depending on the rocket design.[3]

This behaviour resembles balancing a pole on the fingertip – an inverted pendulum.

An inverted pendulum is a textbook example of a highly nonlinear, open-loop unstable system. It is unstable in the sense that in order to keep it upright, it needs swift, active stabilisation. Without such stabilisation, it will fall over.[4]

With a nonlinear system, the task of designing an appropriate control scheme becomes challenging. Linear control methods can be applied, but require a linearised model to design. Alternatively, nonlinear control design deals with stabilisation and control of nonlinear systems directly.[5]

This thesis describes the analysis, modelling and control of an inverted pendulum system, focusing on nonlinear control design in comparison with linear ones.

System Description 2

This chapter describes the test setup considered in this project. First, the general construction and connections are described. Then, in **section 2.1** the functionality of the used Microcontroller Unit (MCU) is presented. The description continues with the motor controller and sensors in **section 2.2** and **section 2.3** respectively. In **section 2.4**, some initial system observations are presented.

The setup is located in the control laboratory at Aalborg University and is depicted in **figure 2.1**.

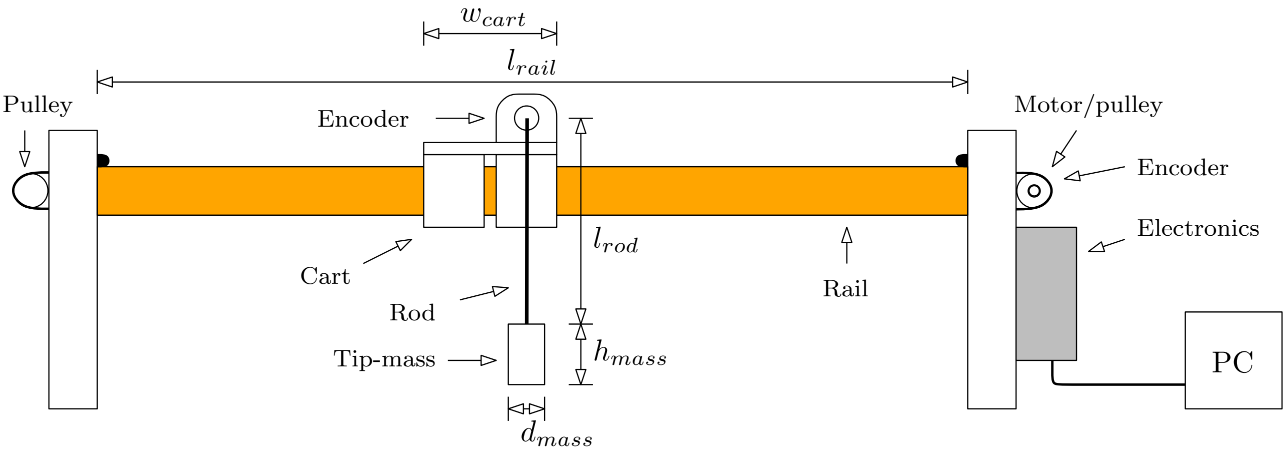


Figure 2.1: *Test setup with essential physical dimensions.*

It consists of a pendulum attached to a cart. The cart is attached to the rail using ball bearings, and is able to move horizontally. A toothed belt connects the cart to the pulleys at each end of the rail. The pulley at the right side is driven by a maxon DC motor, using current control, which effectively drives the cart. Attached to the motor is an incremental optical encoder which is used to measure the cart position.

The pendulum consists of a light metal rod attached to another maxon DC motor, though this is only used for the encoder attached to it in order to measure the angular displacement of the pendulum.

A cylindrical, modular mass is attached at the end of the pendulum rod. It is modular in the sense that it consist of multiple, threaded masses of 100, 75, 50 and 26 grams, which can be screwed together in any combination. By default, all masses will be used. A list of dimensions and the tip-mass is given in **table 2.1**.

The grey box on **figure 2.1** symbolises the electronics of the setup, which includes a maxon servo amplifier[6], an Arduino Due[7], a power supply unit and finally a printed circuit board (PCB), which contains a voltage shield and quadrature decoders.

The Arduino Due operates at 3.3 V, which is insufficient for controlling the DC motor through the power amplifier, as this requires a voltage range of ± 10 V at its input pin. Thus, the voltage shield is used to map the Arduino output of 0 – 3.3 V to the amplifier range. The Arduino is used to compute e.g. controller output and also serves as the interfacing device between sensors and the motor. A C++

Name	Value	Unit	Description
m_p	251	g	Tip-mass of the pendulum
l_{rod}	28.28	cm	Length of the rod between tip-mass and joint
d_{rod}	1	cm	Diameter of the rod
h_{mass}	10.4	cm	Height of tip-mass
d_{mass}	1.95	cm	Diameter of tip-mass
l_{rail}	88.9	cm	Length of the cart rail
w_{cart}	33.45	cm	Width of the cart
l	33.45	cm	Pendulum length, $l_{rod} + 0.5h_{mass}$
r	3	cm	Radius of the pulleys

Table 2.1: *System parameters measured by scale or measurement tape.*

library, `Joint.cpp`, has been supplied for the low-level interaction between these.

The Arduino-motor-encoder signal path is depicted in **figure 2.2**.

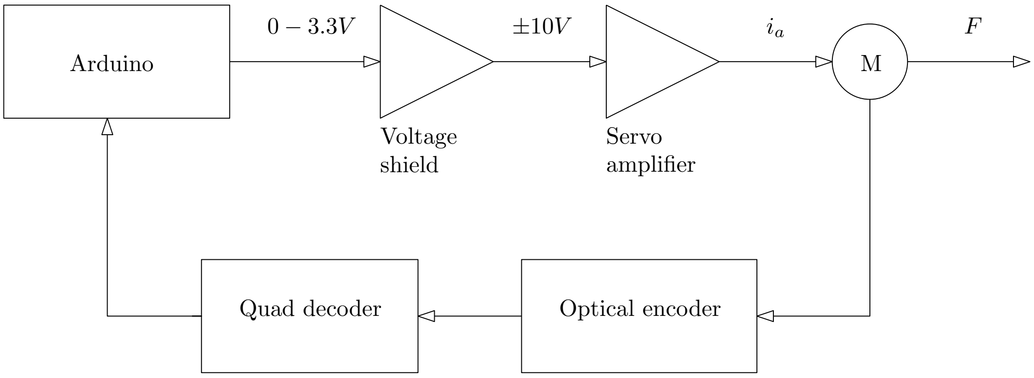


Figure 2.2: *Representation of the signal path between the Arduino to the motor, and back through the encoder and decoder.*

2.1 Arduino Due

The Arduino Due is an MCU with a 32-bit Atmel SAM3X8E Advanced RISC Machine (ARM) Cortex-M3 processor with a clock speed of 84 MHz. It has 512 kilobytes of flash memory and 100 kilobytes of Static Random-Access Memory (SRAM), and is thus well suited for the purpose of this project. It also features two 12-bit Digital to Analog Converters (DAC).

The connections between the Arduino and the remaining hardware is depicted in **figure 2.3**.

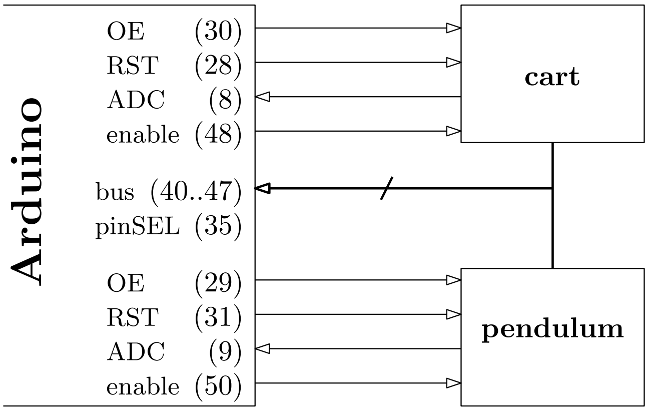


Figure 2.3: *Arduino pin connection between the cart and pendulum hardware.*

The connections between the Arduino and the cart and pendulum are similar. Each connection is displayed with the pin/peripheral name and pin number. The diagram is simplified such that the blocks *cart* and *pendulum* each contain the servo amplifier, motor and optical encoder, respectively. Each connection is described as follows:

DAC Digital to analog converter: Converts a digital number to a voltage output from the Arduino. Used to set a motor current reference for the motor amplifier.

OE Output enable: Enables reading from the encoder when set to binary low.

bus 8 pin data bus.

pinSEL Pin selector: Pin used to select whether to read low byte (binary high) or high byte (binary low) reading on the data bus. The low and high bytes are then combined to create a 16-bit signed integer value.

enable Enable pin: Used for initialization of servo amplifier output.

RST Reset pin: Resets the encoders.

The data bus is used to read decoded sensor values, and works as follows

1. Pin OE of the device to read from is pulled low to enable data output.
2. Low byte is requested and received.
3. High byte is requested and received.
4. Pin OE is pulled high again, ending the transaction.
5. The low and high bytes are combined to form the 16-bit integer value.

The Arduino samples the sensors with a period of 0.05 s and sends this, along with other data, via the USB serial protocol at 115200 baud to a computer.

2.2 Servo amplifier

The servo amplifier is used to provide current control of the motor. The servo amplifier reads a current reference from a ±10 V pin, and applies this to the motor in an internal current control loop.

The current reference is set as a 12-bit integer, which is converted to an analogue value using the DAC on the Arduino board, outputting a corresponding voltage in the $0 - 3.3$ V range. This is amplified by the voltage shield and applied to the servo amplifier pin in the ± 10 V range.

Appendix A describes a test which has been conducted to determine the relation between the 12-bit integer value and the current supplied to the motor. The results are summarized in **figure 2.4**.

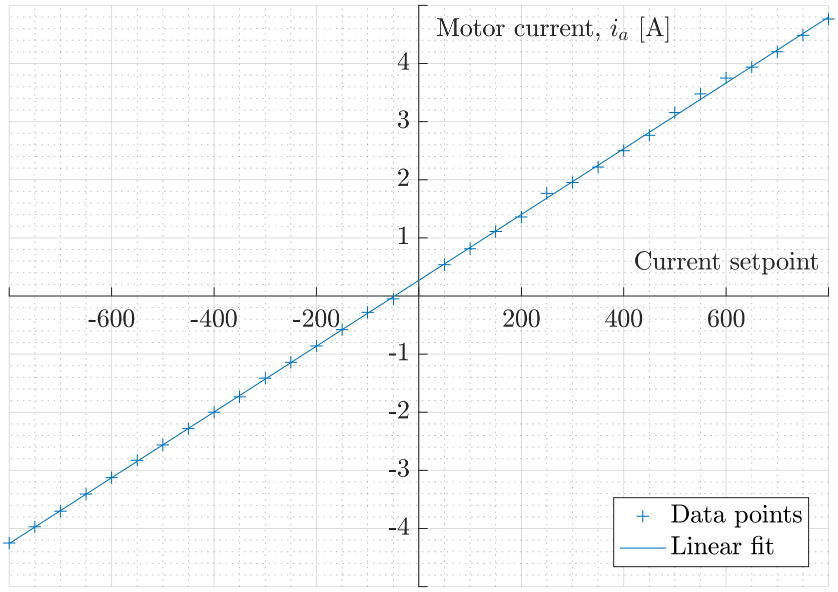


Figure 2.4: *Integer to current test in order to determine direct relationship between the integer value of the current setpoint and the current.*

When setting a current reference value, called the *setpoint*, the corresponding 12-bit integer is shifted by 2000 in the `Joint.cpp` library in order to be able to represent negative values and drive the motor in both directions.[(1)]

A relation between the setpoint and the force applied to the cart is found using the motor torque constant, k_t [8], radius of the pulley attached to the motor, r, and the slope of the data in **figure 2.4**, α. This relation is found to be

$$F = i_a \frac{k_t}{r} = setpoint \cdot \alpha \frac{k_t}{r} \tag{2.1}$$

$$\Leftrightarrow \quad setpoint = \frac{r}{\alpha k_t} F, \qquad \alpha = 0.0057 \tag{2.2}$$

The relation is used in a modification of the `Joint.cpp` library, such that a force reference can be set instead of an integer value.

2.3 Optical encoder and quadrature decoder

Each motor has an incremental optical encoder attached to its shaft, which is used along with a quadrature decoder to measure the cart and pendulum positions by counting small increments of rotation on the motor shafts.

[(1)] In **figure 2.4**, it is seen that setting the setpoint to zero does not result in a current control reference of zero ampere. This is corrected by adjusting the offset from 2000 to 1953.

The optical encoder is an Avago HEDS-5540 [9] with two channels. It contains an LED from which the light is redirected by a lens onto an encoder disc, which has a pattern of holes that lets light through in a similarly patterned fashion when rotated. The light pattern hits a group of photodiodes, where each outputs a current depending on the amount of light that hits it.

A detector circuit is used to detect the lighting pattern from the photo diodes, and outputs a quadrature signal in two channels, A and B, where channel A is 90 degrees out of phase with channel B, indicating the direction of the motor rotation. The working principle of the optical encoder is shown in **figure 2.5**.

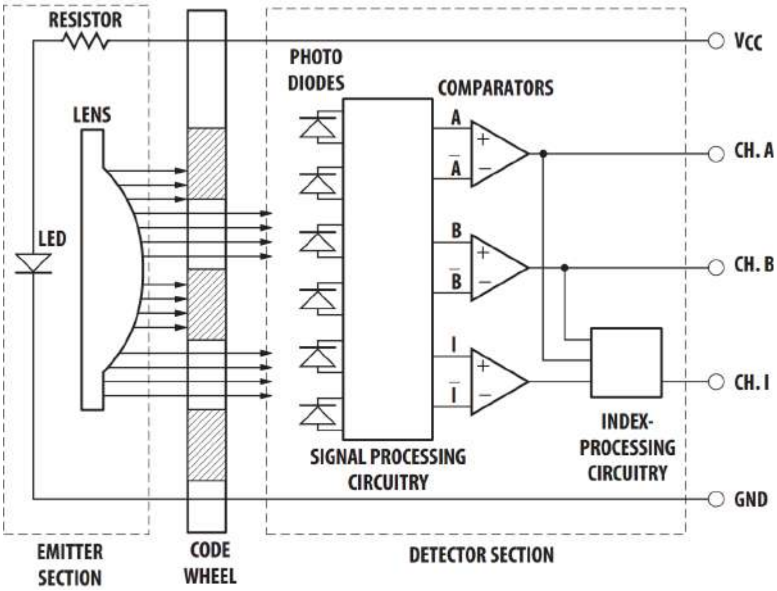


Figure 2.5: *Block diagram of the optical encoder HEDS-5540[9]*

The output of the optical encoder is decoded using a Avago HCTL-2021-PLC [10] quadrature decoder IC chip, which is located on the PCB with the voltage shield. This decoder features a 16-bit up/down counter and an 8-bit bus interface, which is connected to the Arduino.

The result is a *tick* count, depending on how far the motor shafts are rotated from their initial position. The decoder outputs a count of 2000 ticks per revolution, and the pendulum angle have 2π radians per revolution providing a resolution of

$$\min \Delta\theta = \frac{2\pi}{2000} = \pi \cdot 10^{-3} \frac{rad}{ticks} \tag{2.3}$$

The maximum travel length of the cart is $l_{rail} - w_{cart} = 0.769$ m, and the total number of ticks when the cart moves on the rail is 8737, providing a resolution of

$$\min \Delta x = \frac{0.769}{8737} = 88.016 \cdot 10^{-6} \frac{m}{ticks} \tag{2.4}$$

The resolution of the position measurements is thus sufficient for this project.

2.4 Rail and cart observations

The entire system has been made available *as is* in the beginning of the project. This section serves to document some initial observation about the test setup.

Initially, it has been observed that the cart does not run smoothly on the rails, and the friction depends on its position and direction.

Therefore, before proceeding to estimate any parameters, some alterations to the system have been considered:

- Leveling of the system has been adjusted.
- The rail has been polished and greased.

The cart moves on the rail using ball bearings, and is driven by a toothed belt and a motor. By detaching the belt from the cart and letting it move freely on the rail, it has been observed to move with less friction.

The ball bearings introduce some friction as well, especially at specific points were the Coulomb friction increases. That is, at low velocities the cart tends to get stuck and needs extra force to move. It is assumed that the ball bearing friction is amplified through the power transmission, i.e. the belt and pulleys.

Furthermore, the belt introduces friction between its teeth and the pulley. In an attempt to minimize this, another type of belt with a Teflon treated surface has been tested as described in **appendix D**. However, the new belt introduces more friction since the current consumption at the same velocity increased. This is assumed to be due to the new belt being stiffer, such that more mechanical work is wasted on bending the belt.

Thus, the original belt is used for the project.

Problem Statement 3

Given the topic described in **chapter 1** and the system description in **chapter 2**, the following problem statement is developed:

How can a nonlinear control scheme be developed in order to swing up and stabilise an inverted pendulum on a cart, and how does the stabilising control compare to linear approaches?

3.1 Requirements

The requirements for considering the control scheme successful is divided into two categories; functional and additional requirements.

3.1.1 Functional requirements

The functional requirements are the necessary capabilities that must be fulfilled in order to consider the scheme operational.

1) Able to autonomously swing up and stabilise the pendulum in its upright position.
2) Keep the cart from colliding with the physical barriers of the system.

3.1.2 Additional requirements

The additional requirements are features that are desirable, but not necessary.

1) Swing up the pendulum to within ± 10 degrees (~ 0.1745 radians) of its upright position.
2) Swing up the pendulum in less than 10 seconds.
3) Stabilise the pendulum with oscillations less than ± 2 degrees (~ 0.0349 radians).
4) Keep the cart within ten centimetres of the reference position while stabilising the pendulum.
5) Offer robustness for up to 10 percent variation in the tip-mass.

3.2 Outline

In order to fulfil the requirements stated in **section 3.1**, a system model and control laws will be derived and implemented. The derivation of the system model is presented in **chapter 4**.

A two-step control strategy is considered and described in **chapter 6**, as the control scheme required to swing the pendulum up differs from the one required to stabilise it. The proposed swing-up controllers are presented in **chapter 7** and the stabilising controllers in **chapter 8**. Several of these controllers require information of the cart and pendulum velocities, and thus an Extended Kalman Filter (EKF) is designed in **chapter 5**.

All the control schemes are implemented on the Arduino Due at the physical test setup, as described in **chapter 9**, and the results are shown in **chapter 10**.

Modelling 4

This chapter describes the modelling process of the mechanical and electrical part of the system. First, a model of the electrical part, motor and amplifier is derived in **section 4.1**, followed by a model derivation for the system dynamics using Lagrangian mechanics in **sections 4.2** and **4.3**. In **section 4.4**, the model is reformulated into state-space form for later reference and the implementation in MATLAB is described in **section 4.5**. Finally, the model is verified in **section 4.6**.

4.1 Motor electromechanics

The DC motor model is based on the circuit diagram in **figure 4.1**.

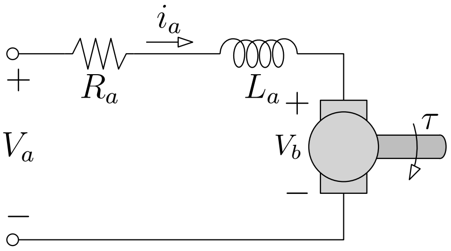


Figure 4.1: *DC motor circuit diagram.*

Here, i_a is the current through the armature, L_a and R_a are the motor inductance and resistance, respectively, V_b is the back-electromotive force (back-EMF), a voltage generated by the shaft motion, and V_a is the voltage applied to the motor terminals.

From Kirchhoffs voltage law, the motor dynamics are described by

$$V_a(t) = L_a \frac{di(t)}{dt} + R_a + V_b(t) \tag{4.1}$$

and the torque produced by the motor is

$$\tau(t) = k_t i(t) \tag{4.2}$$

where k_t is the motor torque constant.

A block diagram illustrating the interaction between the motor dynamics and the system mechanics is shown in **figure 4.2**.

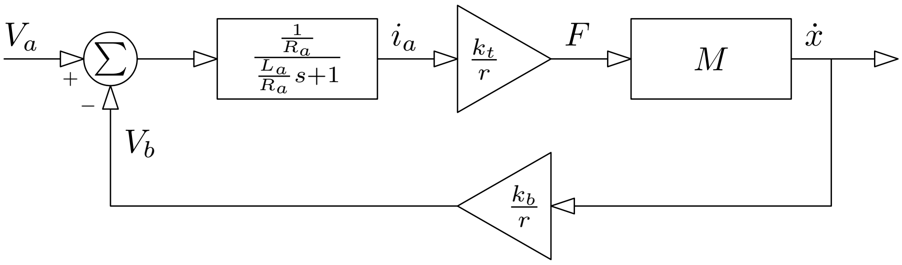


Figure 4.2: *Motor and system interaction.*

In this, M is the mechanical part of the system, which will be considered in **section 4.2**. The parameters k_t and k_b are the motor torque and back-EMF constants, respectively, which are equal, when assumed that the DC motor is ideal, thus $k_b = k_t$.

The servo amplifier features an internal current control for the motor, which is considered as depicted in **figure 4.3**, where the back-EMF voltage is viewed as a disturbance, and the mechanics block M has been omitted.

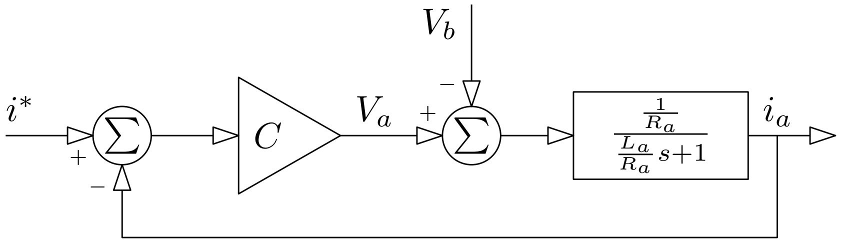


Figure 4.3: *Motor model including current controller*

Here, C is the current controller, and i^* is the current reference. From **figure 4.3** it is possible to establish a transfer function of the motor in the s-domain as

$$H(s) = \frac{i_a}{V_a - V_b} = \frac{\frac{1}{R_a}}{\frac{L_a}{R_a}s + 1} = \frac{K_m}{\tau_m s + 1} \tag{4.3}$$

where K_m and τ_m are constants to be determined experimentally.

In order to do this, a pulse train test was conducted where a signal pulsing between ±4 N (approximately ±1.3 A) with a duration of 500 ms was applied to the motor. Measuring the current directly from the servo amplifier using an oscilloscope yields the results in **figure 4.4**.

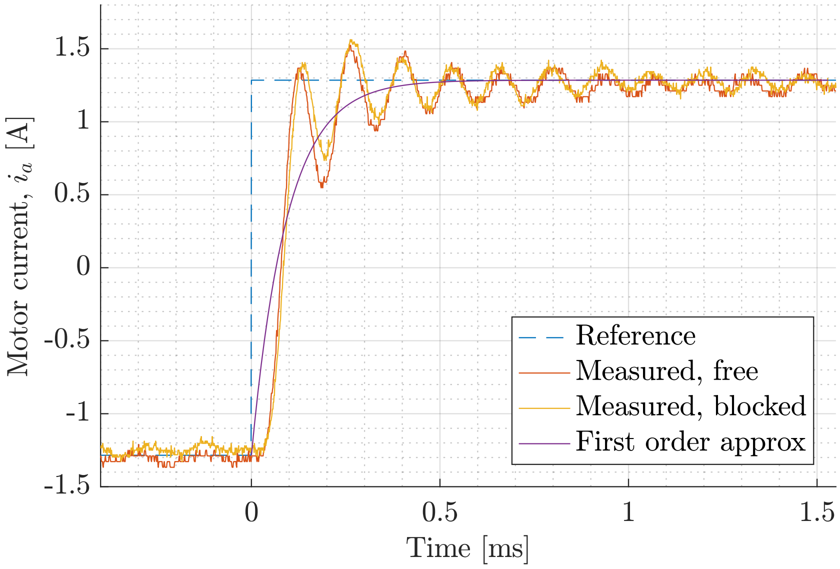


Figure 4.4: *Motor step responses from two tests, where the motor have been blocked and able to run freely. The noise on the step responses is assumed to be commutator noise, which is protortional to the velocity by some constant.*

The signals *Measured, free* and *Measured, blocked* are two different tests using the same input, and refers to whether the motor has been blocked or been able to move freely. This has been done to investigate the disturbance of the back-EMF voltage. As it can be seen, the signals are almost identical, concluding that the servo amplifier is able to suppress this disturbance. The settling time of the current control less than 1 ms, which is much faster than the sampling period of the system of 0.05 ms.

A first order approximation is assumed sufficient to represent the dynamics of the current control, and is found, using MATLAB function `tfest`, to be

$$H(s) = \frac{1.065 \cdot 10^4}{s + 1.058 \cdot 10^4} \tag{4.4}$$

4.2 Cart and pendulum mechanics

The following is based on [11].

In order to model the forces acting on the cart and pendulum, Lagrangian mechanics are used. **Figure 4.5** depicts the mechanical system, where the position vector of the cart is denoted $\mathbf{r}_c$ and the position vector of the pendulum, at its assumed centre of mass, is denoted $\mathbf{r}_p$.

The mass of the pendulum, assumed evenly distributed, and the distance from the pendulum joint at $\mathbf{r}_c$ to its centre of mass at $\mathbf{r}_p$, are denoted by m_p and l, respectively, for which the measured values are listed in **table 2.1**. The pendulum rod is assumed massless. Finally, the angular displacement of the pendulum, from the upright position, is denoted by θ.

It should be noted that the mass of the cart, denoted m_c, is actually a unified parameter for the masses and inertias present in the entire power transmission, i.e. from the motor to the cart itself. Thus, the parameter m_c includes:

- Rotor inertia of the motor
- Pulley inertias
- Belt mass
- Cart mass

Furthermore, the belt is assumed to be non-elastic, thus no phase shift in the power transmission should occur.

It is seen from **figure 4.5** that

$$\mathbf{r}_c(t) = \begin{bmatrix} x(t) \\ 0 \end{bmatrix}, \qquad \mathbf{r}_p(t) = \begin{bmatrix} x(t) - l\sin\theta(t) \\ l\cos\theta(t) \end{bmatrix} \tag{4.5}$$

$$\dot{\mathbf{r}}_c(t) = \begin{bmatrix} \dot{x}(t) \\ 0 \end{bmatrix}, \qquad \dot{\mathbf{r}}_p(t) = \begin{bmatrix} \dot{x}(t) - l\cos\theta(t)\dot{\theta}(t) \\ -l\sin\theta(t)\dot{\theta}(t) \end{bmatrix} \tag{4.6}$$

In order to simplify the notation throughout the rest of this section, explicit notation of time dependency will be omitted, i.e.

$$x = x(t), \qquad \theta = \theta(t) \tag{4.7}$$

$$\mathbf{r}_c = \mathbf{r}_c(t), \qquad \mathbf{r}_p = \mathbf{r}_p(t) \tag{4.8}$$

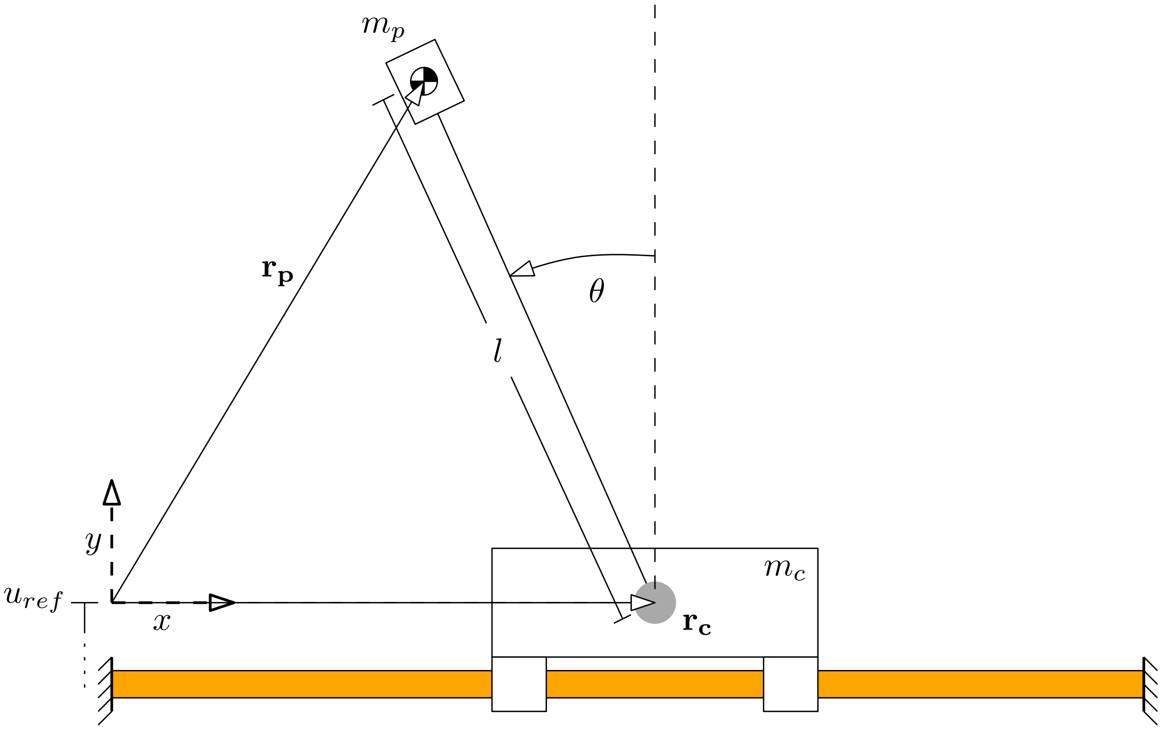


Figure 4.5: *Diagram of cart, pendulum, and rail. Essential vectors, parameters and variables are shown.*

and equivalently for their derivatives.

The differential equations describing the system mechanics are derived based on the kinetic energy T and the potential energy U of the system, where the *Lagrangian* is defined by

$$L = T - U = (T_c + T_p) - (U_c + U_p) \tag{4.9}$$

The potential energies of the system are defined by

$$U_c = u_{ref} \tag{4.10}$$

$$U_p = m_p g l \cos\theta + u_{ref} \tag{4.11}$$

where u_{ref} is the reference potential, i.e. the height that potential energies are relative to, and g is the gravitational constant.

The kinetic energy of the cart is defined by

$$T_c = \frac{1}{2} m_c \dot{\mathbf{r}}_c^T \dot{\mathbf{r}}_c = \frac{1}{2} m_c \begin{bmatrix} \dot{x} & 0 \end{bmatrix} \begin{bmatrix} \dot{x} \\ 0 \end{bmatrix} = \frac{1}{2} m_c \dot{x}^2 \tag{4.12}$$

as the cart is only capable of moving on the horizontal axis.

The kinetic energy of the pendulum is expressed by

$$T_p = \frac{1}{2} m_p \dot{\mathbf{r}}_p^T \dot{\mathbf{r}}_p = \frac{1}{2} m_p \left(\left(\dot{x} - l\cos\theta\dot{\theta} \right)^2 + \left(-l\sin\theta\dot{\theta} \right)^2 \right) \tag{4.13}$$

The Lagrangian thus becomes:

$$\begin{aligned} L(x,\dot{x},\theta,\dot{\theta}) &= \frac{1}{2}m_c\dot{x}^2 + \frac{1}{2}m_p\left(\left(\dot{x} - l\cos\theta\dot{\theta}\right)^2 + \left(-l\sin\theta\dot{\theta}\right)^2\right) - m_p gl\cos\theta - 2u_{ref} \\ &= \frac{1}{2}(m_c + m_p)\dot{x}^2 - m_p l\cos\theta\dot{\theta}\dot{x} + \frac{1}{2}m_p l^2\dot{\theta}^2 - m_p gl\cos\theta - 2u_{ref} \end{aligned} \tag{4.14}$$

From this, the dynamics of the system are found by the *Euler-Lagrange* equation with generalised coordinates $\mathbf{q} = [x \quad \theta]^T$ and external forces $\mathbf{Q}$ as

$$\begin{aligned} \mathbf{Q} &= \frac{d}{dt}\frac{\partial L}{\partial \dot{\mathbf{q}}} - \frac{\partial L}{\partial \mathbf{q}} \\ &= \frac{d}{dt}\begin{bmatrix} \frac{\partial L}{\partial \dot{x}} \\ \frac{\partial L}{\partial \dot{\theta}} \end{bmatrix} - \begin{bmatrix} \frac{\partial L}{\partial x} \\ \frac{\partial L}{\partial \theta} \end{bmatrix} \\ &= \frac{d}{dt}\begin{bmatrix} (m_c + m_p)\dot{x} - m_p l\cos\theta\dot{\theta} \\ m_p l^2\dot{\theta} - m_p l\dot{x}\cos\theta \end{bmatrix} - \begin{bmatrix} 0 \\ m_p l\dot{x}\sin\theta\dot{\theta} + m_p gl\sin\theta \end{bmatrix} \\ &= \begin{bmatrix} (m_c + m_p)\ddot{x} - m_p l\left(\cos\theta\ddot{\theta} - \sin\theta\dot{\theta}^2\right) \\ m_p l^2\ddot{\theta} - m_p l\left(-\dot{x}\sin\theta\dot{\theta} + \ddot{x}\cos\theta\right) \end{bmatrix} - \begin{bmatrix} 0 \\ m_p l\dot{x}\sin\theta\dot{\theta} + m_p gl\sin\theta \end{bmatrix} \\ &= \begin{bmatrix} (m_c + m_p)\ddot{x} - m_p l\cos\theta\ddot{\theta} + m_p l\sin\theta\dot{\theta}^2 \\ m_p l^2\ddot{\theta} - m_p l\cos\theta\ddot{x} - m_p gl\sin\theta \end{bmatrix} \end{aligned} \tag{4.15}$$

Including externally applied actuation on the cart as well as friction in both the cart and the pendulum, $\mathbf{Q}$ can be written as

$$\mathbf{Q} = \begin{bmatrix} F \\ 0 \end{bmatrix} - \mathbf{B}(\dot{x},\dot{\theta}) \tag{4.16}$$

where $\mathbf{B}(\dot{x},\dot{\theta})$ denote cart and pendulum frictions and F denotes the input forces transferred from the motor, through the belt and to the cart.

Letting **equation (4.16)** equal **equation (4.15)** and isolating the accelerations, $\ddot{x}$ and $\ddot{\theta}$, yields the vector function form:

$$\begin{bmatrix} F \\ 0 \end{bmatrix} - \mathbf{B}(\dot{x},\dot{\theta}) = \underbrace{\begin{bmatrix} m_c + m_p & -m_p l\cos\theta \\ -m_p l\cos\theta & m_p l^2 \end{bmatrix}}_{\mathbf{M}(\theta)}\begin{bmatrix} \ddot{x} \\ \ddot{\theta} \end{bmatrix} + \underbrace{\begin{bmatrix} m_p l\sin\theta\dot{\theta}^2 \\ 0 \end{bmatrix}}_{\mathbf{C}(\theta,\dot{\theta})} + \underbrace{\begin{bmatrix} 0 \\ -m_p gl\sin\theta \end{bmatrix}}_{\mathbf{G}(\theta)} \tag{4.17}$$

where $\mathbf{B}(\dot{x},\dot{\theta})$ indicates friction, $\mathbf{M}(\theta)$ includes all inertial properties, $\mathbf{C}(\theta,\dot{\theta})$ expresses centrifugal force and $\mathbf{G}(\theta)$ expresses gravitational force.

For convenience, this can be rearranged into:

$$\begin{bmatrix} \ddot{x} \\ \ddot{\theta} \end{bmatrix} = \mathbf{M}(\theta)^{-1}\left(\begin{bmatrix} F \\ 0 \end{bmatrix} - \mathbf{B}(\dot{x},\dot{\theta}) - \mathbf{C}(\theta,\dot{\theta}) - \mathbf{G}(\theta)\right) \tag{4.18}$$

This concludes the general structure of the mechanical model. The following section will elaborate on the friction terms.

4.3 Cart and pendulum frictions

The following is based on [12].

In **section 4.2**, the dynamics related to the cart and pendulum mechanics have been derived. This includes an external force caused by friction, which will be further elaborated in this section.

Two frictional components are present in the cart/pendulum system, namely the friction force affecting the movement of the cart and the friction torque affecting the pendulum. While the latter is only expected to depend on the angular velocity of the pendulum, the cart friction expresses frictional forces and torques in the following:

- Contact between the ball bearings and the rail.
- Contact between the belt and the gears in the power transmission.
- Internal friction in the motor.

In general, friction can be modelled solely as a function of the velocity, and for this project will be based on the friction components called Coulomb and viscous frictions, assuming the Stribeck-term is negligible, which is supported by the data in **appendix B.2**. Furthermore, effects from aerodynamic drag are considered negligible, with a brief analysis of the topic conducted in **appendix F**.

In this case, the Coulomb friction represents the necessary applied force in order to achieve acceleration, and is also referred to as *static friction*. Viscous friction, on the other hand, is the force produced due to surface contact at a specific velocity.

The friction is modelled as

$$\mathbf{B}(\dot{x},\dot{\theta}) = \begin{bmatrix} B_c(\dot{x}) \\ B_p(\dot{\theta}) \end{bmatrix} = \begin{bmatrix} c_c \operatorname{sgn}(\dot{x}) + v_c\dot{x} \\ c_p \operatorname{sgn}\left(\dot{\theta}\right) + v_p\dot{\theta} \end{bmatrix} \tag{4.19}$$

for translational friction force in the power transmission and cart, B_c, and rotational friction torque in the pendulum joint, B_p, respectively. **Figure 4.6** illustrates these components separately and combined. Similar to the mass parameter m_c, the expression for $F_{c,v}$ will represent frictions from the power transmission in addition to the cart/rail friction.

An issue persists when using the sign function in simulation, as it tends to be numerically unstable at low velocities. This is due to the numerical value of the velocity never becoming exactly zero, causing the sign function to discontinuously jump between ± 1.

In order to solve this, a *sigmoid* can be used to approximate the sign function. For this, the hyperbolic tangent, $\tanh(\cdot)$, is used, and the frictions thus become

$$B_c(\dot{x}) = c_c \tanh(k\dot{x}) + v_c\dot{x} \tag{4.20}$$

$$B_p(\dot{\theta}) = c_p \tanh(k\dot{\theta}) + v_p\dot{\theta} \tag{4.21}$$

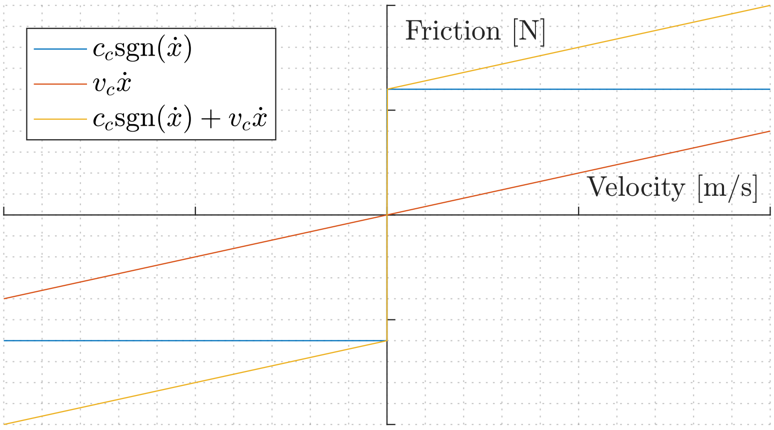


Figure 4.6: *General friction model with Coulomb friction $c_c \operatorname{sgn}(\dot{x})$, viscous friction $v_c\dot{x}$ and combined friction $c_c \operatorname{sgn}(\dot{x}) + v_c\dot{x}$. Similarly for the pendulum friction torque.*

with k being a sufficiently large parameter that determines the steepness of $\tanh(\cdot)$ through zero, and equivalently for the rotational friction. **Figure 4.6** shows a comparison of the original and the approximate expressions of the combined friction.

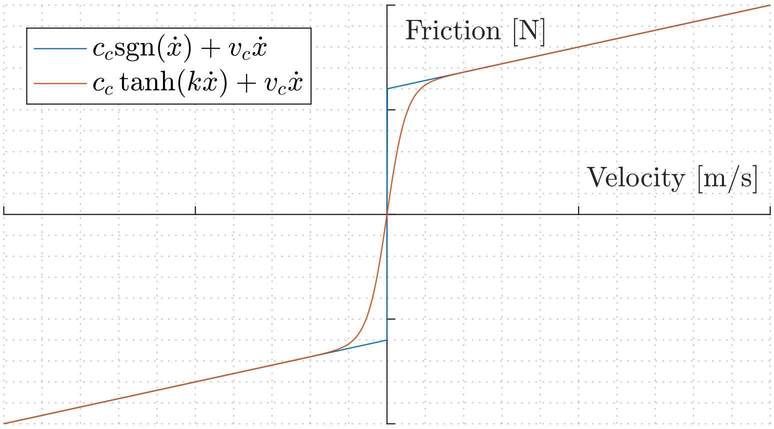


Figure 4.7: *Friction model $c_c \operatorname{sgn}(\dot{x}) + v_c\dot{x}$ and a continuous approximation, $c_c \tanh(k\dot{x}) + v_c\dot{x}$.*

4.3.1 Translational friction in the cart

In order to estimate the coefficients for the translational friction, a series of tests are presented in **appendix B**. In this, the data shows that the viscous coefficient for the cart is approximately zero. However, the data regarding the translational Coulomb friction are inconsistent and are therefore disregarded.

As an alternative, the cart has instead been subjected to a pulse input, starting from either end of the rail and accelerated towards the middle before stopping due to friction. Using this data and the dynamic model, an optimisation routine using MATLAB function `fmincon` has been set up.

In this, the model in **equation (4.18)** is subjected to the same input and then compared with the data in order to determine proper parameters, namely cart mass m_c, Coulomb friction when moving right, $c_{c,R}$, and when moving left, $c_{c,L}$. The directional dependency is chosen as per observation of the physical test setup, as described in **section 2.4**.

The mass is included as it cannot be directly measured due to physical restrictions in the test setup.

The cost function used for the optimisation is the Root Mean Square Error (RMSE) of the cart position, summed for each direction:

$$C_{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{c_{i,R}} - \overline{x}_{c_{i,R}})^2} + \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{c_{i,L}} - \overline{x}_{c_{i,L}})^2} \tag{4.22}$$

with $\overline{x}_c$ indicating the simulation while R and L indicates direction.

With this cost function, the optimisation problem is formulated as:

$$\begin{aligned} &\min_{m_c,\ c_{c,R},\ c_{c,L}} && C_{RMSE} \\ &\text{subject to} && m_c,\ c_{c,R},\ c_{c,L} \in [1,5] \end{aligned} \tag{4.23}$$

with initial values being three for all the variables.

The solution to the optimisation problem is found to be:

$$m_c = 3.8495 \tag{4.24}$$

$$c_{c,R} = 3.6192 \tag{4.25}$$

$$c_{c,L} = 3.1041 \tag{4.26}$$

with the data and simulation corresponding to the solution is shown in **figure 4.8**.

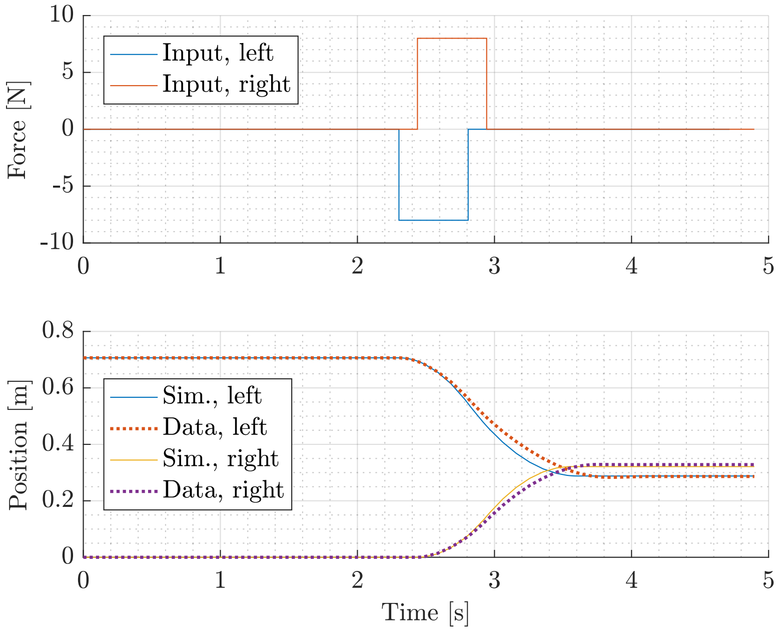


Figure 4.8: *Data and simulation of the optimisation solution. Top figure shows the input signals in each direction, and the bottom figure shows the real data and simulation with the found parameters.*

Conclusively, some differences between the test data and the simulation persist, but the found coefficients are assumed to suffice. The final form of the friction model thus becomes:

$$B_c(\dot{x}) = c_c \tanh(k\dot{x}), \qquad c_c = \begin{cases} c_{c,R} & \text{if } \dot{x} \geq 0 \\ c_{c,L} & \text{if } \dot{x} < 0 \end{cases} \tag{4.27}$$

4.3.2 Rotational friction in the pendulum joint

For the pendulum friction model, Coulomb and viscous friction coefficients are needed. These coefficients, denoted c_p and v_p, are found by the methods described in **appendix E**.

Figure 4.9 shows the method for finding c_p, as the pendulum is manually adjusted to find the maximum angle ϕ at which it remains stationary, using mass $m_p = 26$ g and by computing the torque, τ_c, produced by gravity.

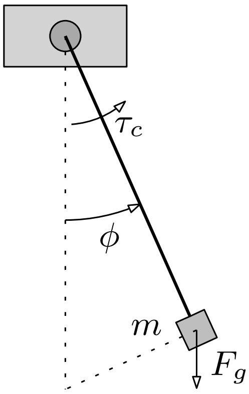


Figure 4.9: *Pendulum system with variables and parameters used for Coulomb friction estimation.*

In order to find v_p, the pendulum has been raised manually, then released and let oscillate until coming to a full stop. From this underdamped, harmonic motion, the coefficient is determined by fitting the peaks to an exponentially decaying function, as described in **appendix E.2** and shown in **figure 4.10**.

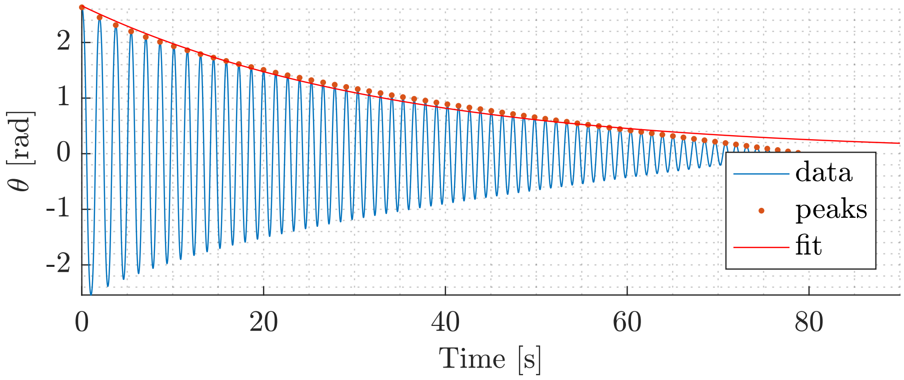


Figure 4.10: *Data and exponential fit used for determining the viscous friction coefficient.*

The two coefficients have been found to be:

$$c_p = 0.0030 \qquad v_p = 0.01479 \tag{4.28}$$

Since the response in **figure 4.10** are also subject to the Coulomb torque, the individually found values are adjusted to fit this.

Using the values

$$c_p = 0.0035 \qquad v_p = 0.00067 \tag{4.29}$$

yields the result shown in **figure 4.11**.

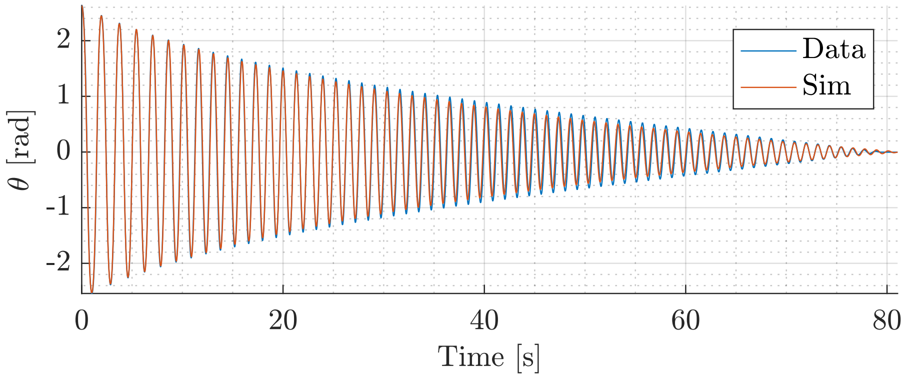


Figure 4.11: *Response of the pendulum swinging freely using adjusted friction coefficients.*

It is seen that the viscous coefficient is lowered significantly as the Coulomb friction torque is added to the model.

Conclusively, the friction model for the entire system is

$$\mathbf{B}(\dot{x},\dot{\theta}) = \begin{bmatrix} B_c(\dot{x}) \\ B_p(\dot{\theta}) \end{bmatrix} = \begin{bmatrix} c_c(\dot{x})\ \mathrm{sgn}\,(\dot{x}) \\ c_p\ \mathrm{sgn}\left(\dot{\theta}\right) + v_p\dot{\theta} \end{bmatrix} \tag{4.30}$$

with

$$c_p = 0.0035 \tag{4.31}$$

$$c_v = 0.00067 \tag{4.32}$$

$$c_c(\dot{x}) = \begin{cases} c_{c,R} = 3.6192 \text{ if } \dot{x} \geq 0 \\ c_{c,L} = 3.1041 \text{ if } \dot{x} < 0 \end{cases} \tag{4.33}$$

4.4 State space formulation

For convenience, the mechanical model, presented at **equation (4.18)** can be formulated as a state space system.

Considering the system state vector $\mathbf{x} = [x_1 \quad x_2 \quad x_3 \quad x_4]^T = [x \quad \theta \quad \dot{x} \quad \dot{\theta}]^T$, the nonlinear state space system representation becomes:

$$\begin{aligned} \dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{\theta} \\ \ddot{x} \\ \ddot{\theta} \end{bmatrix} &= \begin{bmatrix} \dot{x} \\ \dot{\theta} \\ \mathbf{M}(\theta)^{-1}\left(\begin{bmatrix} F \\ 0 \end{bmatrix} - \mathbf{B}(\dot{x},\dot{\theta}) - \mathbf{C}(\theta,\dot{\theta}) - \mathbf{G}(\theta)\right) \end{bmatrix} \\ &= \begin{bmatrix} \dot{x} \\ \dot{\theta} \\ -\alpha(\mathbf{x}) - \beta(\mathbf{x}) \\ -\gamma(\mathbf{x}) - \delta(\mathbf{x}) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \left(m_c + m_p \sin^2\theta\right)^{-1} \\ \cos\theta \left(l(m_c + m_p \sin^2\theta)\right)^{-1} \end{bmatrix} F \end{aligned} \tag{4.34}$$

where system input $u = F$ and

$$\alpha(\mathbf{x}) = \frac{c_c \tanh(k\dot{x}) + m_p l \sin\theta\dot{\theta}^2}{m_c + m_p \sin^2\theta} \tag{4.35}$$

$$\beta(\mathbf{x}) = \cos\theta \frac{c_p \tanh(k\dot{\theta}) + v_p\dot{\theta} - m_p g l \sin\theta}{l(m_c + m_p \sin^2\theta)} \tag{4.36}$$

$$\gamma(\mathbf{x}) = \cos\theta \frac{c_c \tanh(k\dot{x}) + m_p l \sin\theta\dot{\theta}^2}{l(m_c + m_p \sin^2\theta)} \tag{4.37}$$

$$\delta(\mathbf{x}) = (m_c + m_p) \frac{c_p \tanh(k\dot{\theta}) + v_p\dot{\theta} - m_p g l \sin\theta}{m_p l^2(m_c + m_p \sin^2\theta)} \tag{4.38}$$

and the system output is

$$\mathbf{y} = h(\mathbf{x}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \mathbf{x} \tag{4.39}$$

For simplicity and later reference, this can be compactly written as

$$\begin{aligned} \dot{\mathbf{x}} &= \begin{bmatrix} \dot{x} \\ \dot{\theta} \\ f_c(\mathbf{x}) \\ f_p(\mathbf{x}) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g_c(\mathbf{x}) \\ g_p(\mathbf{x}) \end{bmatrix} u \\ &= f(\mathbf{x}) + g(\mathbf{x})u \end{aligned} \tag{4.40}$$

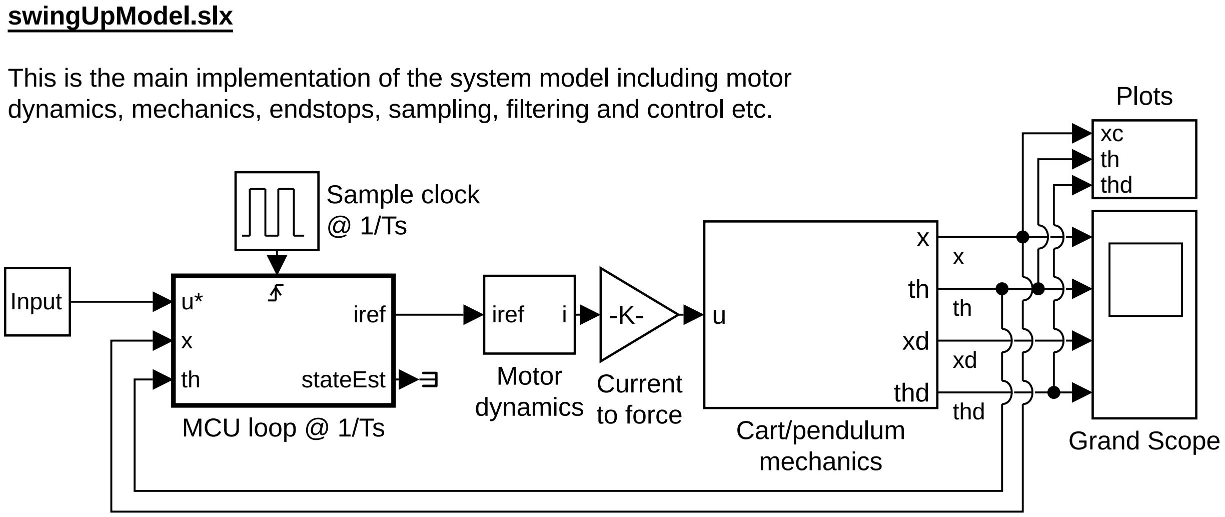
4.5 Simulink implementation

This section concerns the MATLAB/Simulink-implementation of the system. The model is based on several files:

- `swingUpModel.slx`
  The main implementation of the system model where the simulation is carried out using an ordinary differential equation solver.
- `params.m`
  Initialisation script that holds the values of all parameters needed to run the simulation along with default values for some simulation settings.
- `swingUp.m`
  Script used to call `params.m` and change simulation settings from their defaults.

The remainder of this section will focus on the content of `swingUpModel.slx`. The top-level of the file is shown in **figure 4.12**.

Starting from the left, the block *Input* contains several Simulink source blocks (e.g. steps, pulse trains) which is used to simulate the system response of open-loop inputs.



swingUpModel.slx

This is the main implementation of the system model including motor dynamics, mechanics, endstops, sampling, filtering and control etc.

Figure 4.12: *Simulink model file swingUpModel.slx, top-level.*

The next block, *MCU loop*, is a triggered subsystem which is run at each rising edge of the output from the block *Sample clock*, a pulse train with a period of $T_s = 0.005\ \mathrm{ms}$. Its main purpose of *MCU loop* is to contain the data filtering and control algorithms.

The triggering is used to emulate sampling effects on the data fed into the block as well as on the control output. Additionally, though not shown, this block emulates quantisation of the inputs *x* and *th* (θ). The block in its entirety will be further elaborated in **section 9.1**.

The block *Motor dynamics* contains the first-order approximated transfer function described in **section 4.1** and saturation based on the analysis in **appendix A**. The gain in *Current to force* is the product of the torque coefficient of the motor, k_t, and the inverse of the pulley radius, r^{-1}.

The subsystem *Cart/pendulum mechanics* contains the core of the simulation, namely the dynamics described in **section 4.2**. The content of this subsystem is shown in **figure 4.13**.

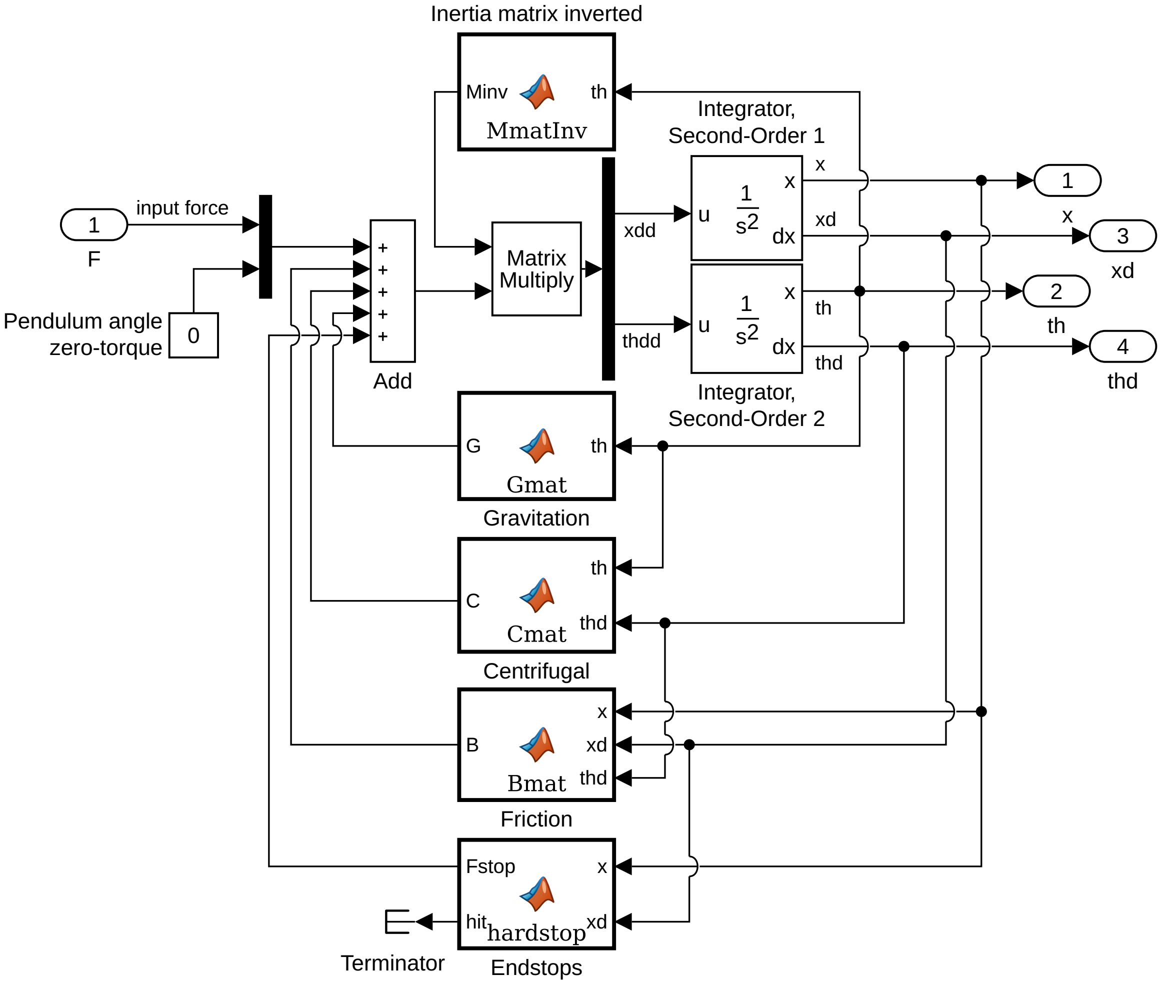


Figure 4.13: *Simulink model file swingUpModel.slx, cart/pendulum mechanics subsystem.*

As such, this is **equation (4.18)**, split up into its subcomponents, followed by double integration. In addition to this, the block *Endstops* is an implementation of the barriers at each end of the rail, modelled as two very stiff spring-damper systems with coefficient found by trial-and-error.

4.5.1 Data visualisation

To the far right of **figure 4.12**, the two blocks named *Grand Scope* and *Plots* serve the purpose of data visualisation.

Grand Scope is the visualisation of the states over time. *Plots* is a graphical animation of the cart and pendulum as well as an animated phase-plot of $(\theta, \dot{\theta})$, both of which are updated along with the progress of the simulation. This is made as a way to intuitively verify the behaviour of the system while running a simulation.

Snapshots of the two animations are shown in **figures 4.14** and **4.15**.

For real-time updates on these plots, an item from the MATLAB Community File Exchange called *Real-time Pacer* is used, as it slows down the simulation speed to approximately real-time. [13]

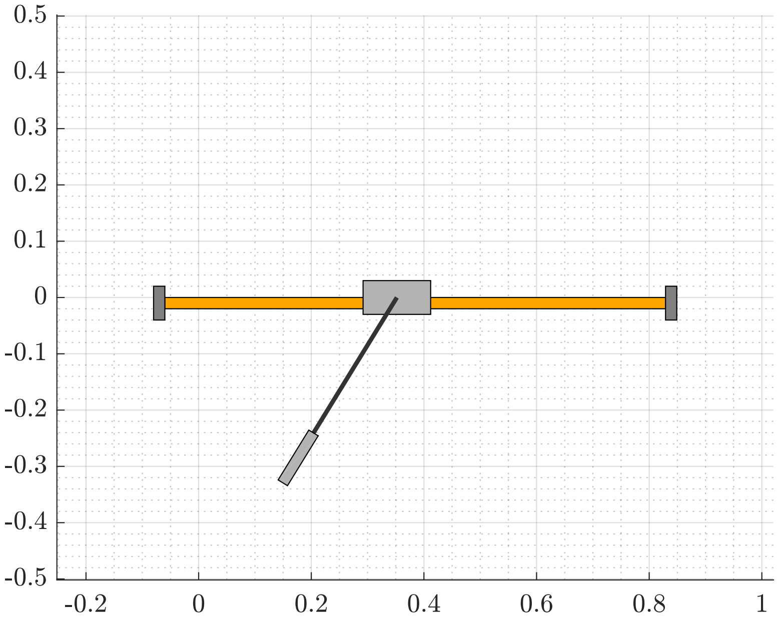


Figure 4.14: *Snapshot of the system animation, mid-simulation.*

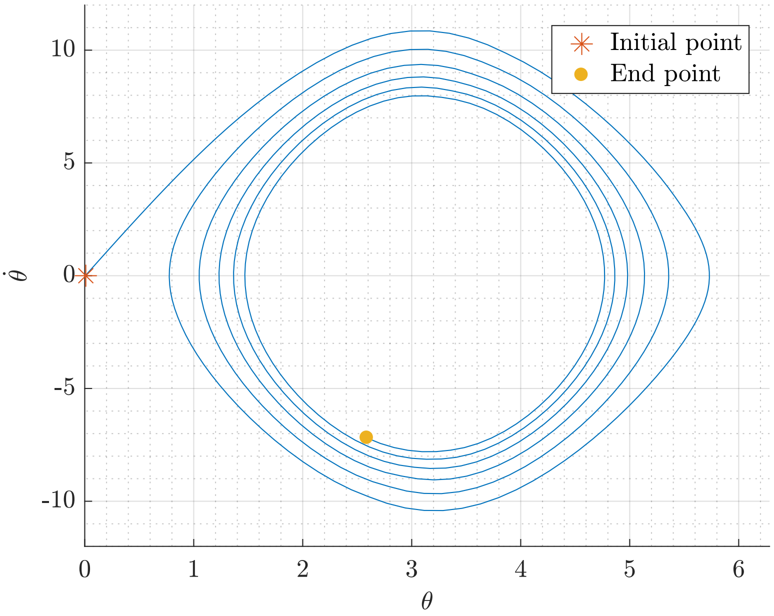


Figure 4.15: *Snapshot of the pendulum phase animation, mid-simulation.*

4.6 Model verification

In this section, the model with the found parameters will be tested in order to evaluate to which degree it represents the physical system. This will be done in two different scenarios:

Input Cart acceleration/deceleration with pendulum started in its stable equilibrium. Two successive input force pulses, in opposite directions, are applied to the open-loop system.

Oscillation Free pendulum motion from the vicinity of the unstable equilibrium. Without any actuation input, both the pendulum and the cart are allowed to move freely.

In both cases, the system position responses are logged along with the applied input (if any). Then, the simulation model is initialised with the same initial conditions and is subjected to the same input, after which the responses are compared.

The first scenario is considered in order to verify the effect of actuation input to the entire system, while the second is in order to check the coupled effects between the pendulum and the cart under a highly dynamic pendulum motion.

Figure 4.16 shows the results of both scenarios with the parameters initially determined, as described previously in this chapter. For the input test, it is seen that the model shows almost the same response as the data, except for a slight deviation during movement in the negative x-direction. The coupled response from the pendulum also responds according to the data.

However, for the oscillation test, a slight deviation of about five centimetres is seen in the cart position, as the modelled cart tends to drift a bit to the left. Additionally, the physical pendulum motion is initially more damped than the model.

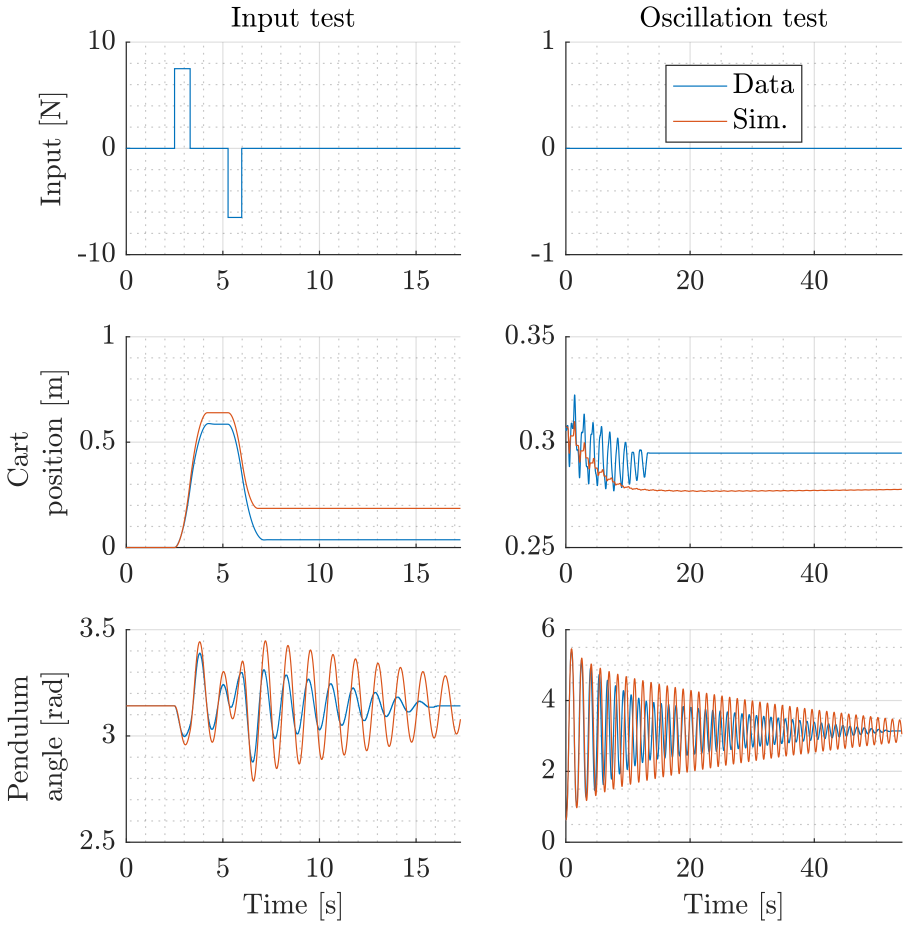


Figure 4.16: *Model verification test results using the initially found parameters. The three plots on the left-hand side are from the input test, and the other three are from the oscillation test.*

In order to further investigate the pendulum response data, **figure 4.17** shows two cases of the oscillation test; with and without the transmission belt attached. Initially, the pendulum tends to be damped quicker with the belt attached, but after about 22 seconds, the oscillations are synchronous.

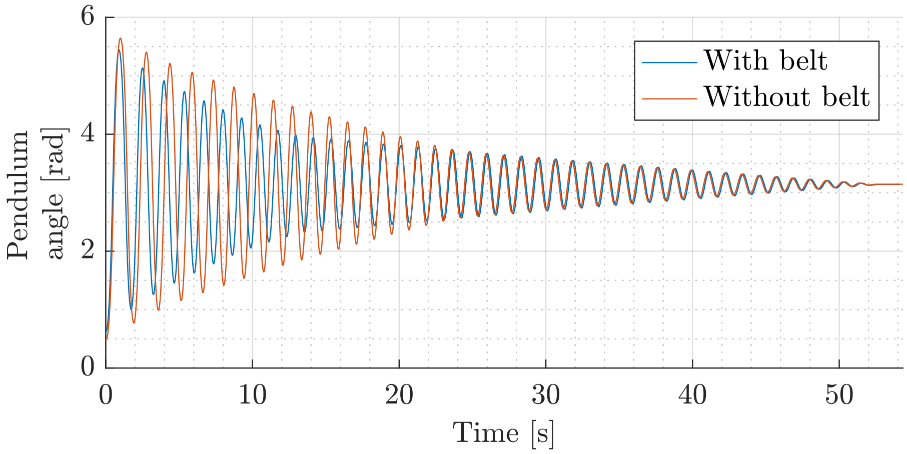


Figure 4.17: *Comparison of two oscillation tests; with and without the belt attached to the cart.*

Observations of the cart motion during the two tests show that the change in damping occurs when the centrifugal force decreases beyond the point where it is unable to move the cart, i.e. at about 22 seconds without the belt and 13 seconds with the belt. This is also seen in **figure 4.16** on the middle-right subfigure. The test without the belt does not have corresponding cart position data, as this is measured by the encoder on the motor.

Given the results in **figure 4.16**, some adjustments to the system parameters are made, which are presented in **table 4.1**

	Old	New	Unit
m_c	3.8495	4.75	[kg]
$c_{c,R}$	3.6192	3.4	[N]
$c_{c,L}$	3.1041	2.45	[N]
c_p	0.0038	0.0038	[Nm]
v_p	0.00067	0.00067	[Nms/rad]

Table 4.1: *Parameter adjustments.*

These new parameter values produce the results shown in **figure 4.18**, where it is seen that the simulated motion of both the cart and the pendulum is rather accurate with respect to the data for the input test. However, for the oscillation test, the cart drifts about 6 cm to the left as opposed to just 1.5 cm in **figure 4.16**, and the issue with the pendulum damping persists. The new parameters will replace the ones previously found. It is notable, however, that the mass is significantly higher and the friction are lower. This may be related to that fact that m_c does not only account for the cart mass, but also the inertias in the pulleys and the motor.

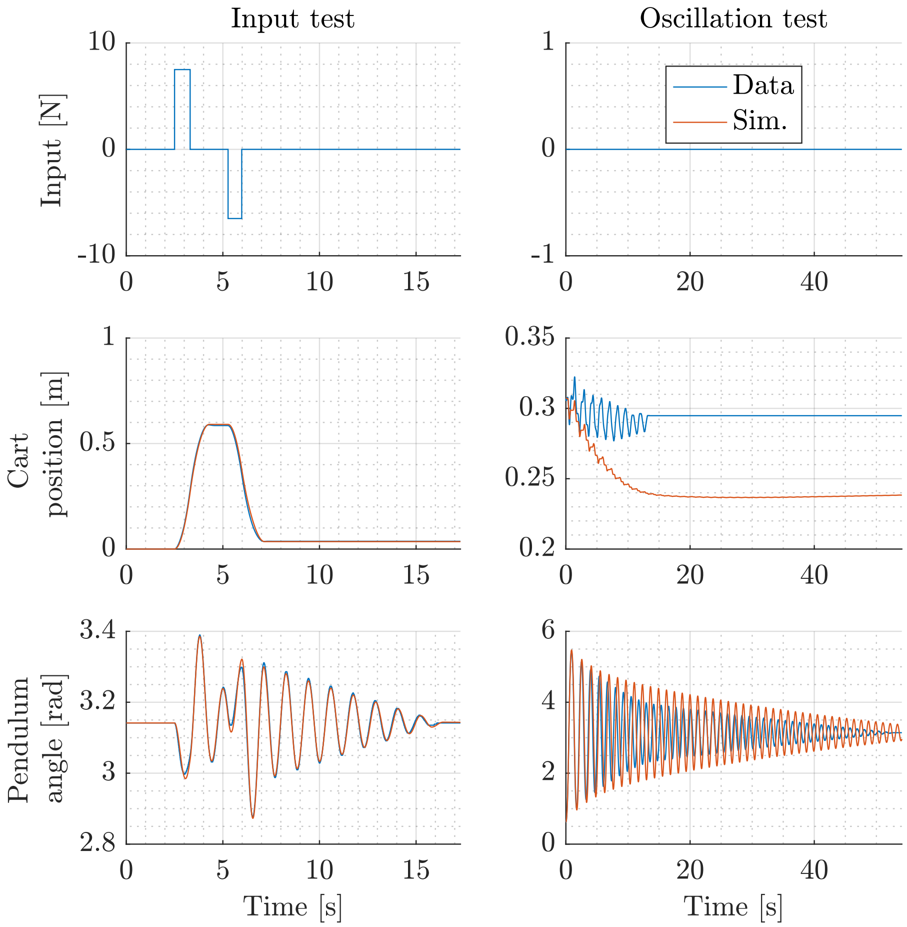


Figure 4.18: *Model verification test using the adjusted parameters.*

State Estimation 5

The physical test setup is equipped with relative encoders attached to the motors, and are therefore only capable of measuring displacement relative to the initial position.

Since the full state-vector is needed for the controllers designed in **chapters 7** and **8** the topic of state estimation is relevant and will be covered in this chapter, specifically to estimate the velocities, $\dot{x}$ and $\dot{\theta}$.

5.1 Numerical derivative

Given the fact that the measured states are positions x and θ, the simplest approach to estimating their first time derivatives are by doing so numerically.

For a function of time t, $f(t)$, the derivative is (backwards) defined by:

$$\frac{df(t)}{dt} = \lim_{\Delta t \to 0} \frac{f(t) - f(t - \Delta t)}{\Delta t} \tag{5.1}$$

Given a sampling time T_s, the states $\dot{x}$ and $\dot{\theta}$ can thus be approximated by:

$$\left.\begin{aligned} \dot{x}(t) &\approx \frac{x(t) - x(t - T_s)}{T_s} = \frac{x_k - x_{k-1}}{T_s} = \dot{x}_k \\ \dot{\theta}(t) &\approx \frac{\theta(t) - \theta(t - T_s)}{T_s} = \frac{\theta_k - \theta_{k-1}}{T_s} = \dot{\theta}_k \end{aligned}\right\} t \in T_s \cdot \mathbb{Z}^+ \tag{5.2}$$

However, one issue arises when computing the derivative this way. Due to the resolution of the encoders, quantization of the data occurs. This is shown in **figure 5.1**, which illustrates the final few oscillations of the pendulum when let to swing freely from $\theta \approx 0$ with the resolutions described in **section 2.3**.

Quantization is an issue regarding the numerical derivative as it will limit the resolution of the velocity estimate based on the quantisation step size and the sampling time. This is shown in **section 5.3**, where the smallest non-zero value of the numerical derivative is

$$\min \Delta\dot{\theta} = \frac{\min \Delta\theta}{T_s} \approx 0.6283 \, \frac{\text{rad}}{\text{s}} \tag{5.3}$$

given $T_s = 0.005$ seconds.

The equivalent value for $\dot{x}$, although not shown, is

$$\min \Delta\dot{x} = \frac{\min \Delta x}{T_s} \approx 0.0176 \, \frac{\text{m}}{\text{s}} \tag{5.4}$$

In order to address this, the data can subsequently be filtered by e.g. a moving average (MA) filter. MA-filters consider a sample window and output the average of the samples:

$$\dot{\theta}_{k,MA} = \frac{1}{n} \sum_{i=0}^{n-1} \dot{\theta}_{k-i} \tag{5.5}$$

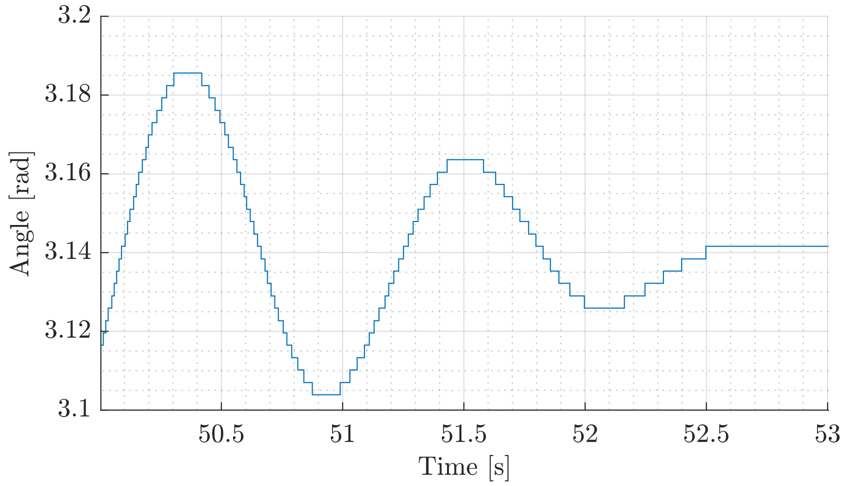


Figure 5.1: *Raw pendulum angle data which shows the resolution of the measured signal.*

where n is the window, i.e. the number of samples to be used in the averaging. State $\dot{x}_k$ is found equivalently. The effect of the moving average is also seen in **section 5.3** for window sizes of both 10 and 50. It is seen that the 10-sample MA increases the resolution while introducing some phase delay. The 50-sample MA increases the resolution even further, but also introduces a much larger phase delay.

5.2 Extended Kalman Filter

The following is based on [14].

Another option in terms of state estimation is the Kalman filter, which is a linear prediction-correction filter that uses a system model to predict the state vector based on the previous time step and the system input, and then corrects the current estimate given a measurement.

For highly nonlinear systems, such as the one considered in this project, the linear Kalman filter becomes insufficient. Instead, an EKF is used, as it uses a nonlinear model to propagate the system states and output, and a linearisation in the most recent state vector estimate for the matrix computations.

The discrete-time system dynamics with state noise w_k and measurement noise v_k are expressed by

$$\mathbf{x}_{k+1} = f_k(\mathbf{x}_k, u_k) + w_k, \qquad w_k \in \ \mathrm{NID}(\mathbf{0}, \mathbf{Q}_k) \tag{5.6}$$

$$\mathbf{y}_{k+1} = h_k(\mathbf{x}_k) + v_k, \qquad v_k \in \ \mathrm{NID}(\mathbf{0}, \mathbf{R}_k) \tag{5.7}$$

where $f_k(\mathbf{x}_k, \mathbf{u}_k)$ is a discretisation of the continuous system dynamics in **section 4.4** by the forward Euler method, i.e.

$$\begin{aligned} f_k(\mathbf{x}_k, u_k) &= \mathbf{x}_k + T_s\dot{\mathbf{x}}(\mathbf{x}_k) \\ &= \mathbf{x}_k + T_s\left(f(\mathbf{x}_k) + g(\mathbf{x}_k)u_k\right) \end{aligned} \tag{5.8}$$

$$h_k(\mathbf{x}_k) = h(\mathbf{x}_k) \tag{5.9}$$

with $f(\mathbf{x}_k)$, $g(\mathbf{x}_k)$ and $h(\mathbf{x}_k)$ defined in **equation (4.40)**, page **21**.

The matrices $\mathbf{Q}_k$ and $\mathbf{R}_k$ are the covariance matrices for the model accuracy and measurement noise, respectively.

Using this model, the Kalman filter is applied in two steps, namely a measurement update (correction) and a time update (prediction).

5.2.1 Correction step

The correction step uses information from new samples of $\mathbf{y}_k$ and u_k in addition to the state vector prediction made during the previous sample, $\hat{\mathbf{x}}_{k|k-1}$:

$$\hat{\mathbf{y}}_{k|k-1} = h_k(\hat{\mathbf{x}}_{k|k-1}) \tag{5.10}$$

This is used to compute the system output residual

$$\tilde{\mathbf{y}}_{k|k-1} = \mathbf{y}_k - \hat{\mathbf{y}}_{k|k-1} \tag{5.11}$$

The Kalman gain, defined by

$$\mathbf{K}_k \triangleq \mathbf{P}_{k|k-1}\mathbf{H}_k^T \left(\mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^T + \mathbf{R}_k\right)^{-1} \tag{5.12}$$

where $\mathbf{P}_{k|k-1}$ is the covariance matrix and $\mathbf{H}_k$ is the linearised output function. The Kalman gain is used to correct the state estimate by:

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\tilde{\mathbf{y}}_{k|k-1} \tag{5.13}$$

Finally, the covariance matrix is computed by

$$\mathbf{P}_{k|k} = \left(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k\right)\mathbf{P}_{k|k-1}\left(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k\right)^T + \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^T \tag{5.14}$$

5.2.2 Prediction step

Following the correction step, a prediction of the state vector is made for the next sample, based on the model, current state estimate and current input, as

$$\hat{\mathbf{x}}_{k+1|k} = f_k(\hat{\mathbf{x}}_{k|k}, u_k) \tag{5.15}$$

followed by a prediction of the covariance matrix:

$$\mathbf{P}_{k+1|k} = \mathbf{F}_k\mathbf{P}_{k|k}\mathbf{F}_k^T + \mathbf{Q}_k \tag{5.16}$$

where $\mathbf{F}_k$ is the linearised system dynamics function.

The predicted values are then saved for the next iteration of the filter, namely when the system is sampled again T_s time units later:

$$k+1|k \quad \rightarrow \quad k|k-1 \qquad \begin{cases} \hat{\mathbf{x}}_{k+1|k} & \rightarrow \quad \hat{\mathbf{x}}_{k|k-1} \\ \mathbf{P}_{k+1|k} & \rightarrow \quad \mathbf{P}_{k|k-1} \end{cases} \tag{5.17}$$

which is then used in the correction step at the next sample.

5.2.3 Jacobian computation

The linearised system equations, namely matrices $\mathbf{H}_k$ and $\mathbf{F}_k$, must be derived analytically or otherwise found in order to be used in the filter.

These matrices are given by their Jacobians, defined by

$$\mathbf{H}_k \triangleq \frac{\partial h_k(\mathbf{x})}{\partial \mathbf{x}^T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{5.18}$$

$$\begin{aligned} \mathbf{F}_k \triangleq \frac{\partial f_k(\mathbf{x}_k)}{\partial \mathbf{x}_k^T} &= \frac{\partial}{\partial \mathbf{x}_k^T} \left(\mathbf{x}_k + T_s \left(f(\mathbf{x}_k) + g(\mathbf{x}_k) u_k \right) \right) \\ &= \mathbf{I} + \frac{\partial}{\partial \mathbf{x}_k^T} \left(T_s \left(f(\mathbf{x}_k) + g(\mathbf{x}_k) u_k \right) \right) \end{aligned} \tag{5.19}$$

As evaluating **equation (5.19)** results in a very large matrix expression, which must be re-evaluated and re-implemented after every change to the system equations, an alternative approach is taken, namely numerical computation of the Jacobian entries:

$$\mathbf{F}_k = \begin{bmatrix} \dfrac{\partial f_k(\mathbf{x}_k)}{\partial x_{k,1}} & \cdots & \dfrac{\partial f_k(\mathbf{x}_k)}{\partial x_{k,n}} \end{bmatrix}, \qquad n = \dim(\mathbf{x}_k) \tag{5.20}$$

with

$$\frac{\partial f_k(\mathbf{x}_k)}{\partial x_{k,i}} = \frac{f_k(\mathbf{x}_k^{pert}) - f_k(\mathbf{x}_k)}{\epsilon}, \qquad \mathbf{x}_k^{pert} = \begin{cases} \mathbf{x}_{k,j} + \epsilon \text{ if } j = i \\ \mathbf{x}_{k,j} \text{ otherwise} \end{cases}, \quad j = 1, \ldots, n \tag{5.21}$$

This effectively perturbs the system equations by ϵ, one entry of the state vector at a time.

5.3 Comparison

Figure 5.2 shows a simulation comparison of the estimation methods described in this chapter.

The uppermost subfigure shows several oscillations from a simulation of the pendulum swinging freely. The two smaller subfigures are close-ups of the left and right boxes marked on the uppermost one, to see the estimates of both large and small signals.

The figure shows that the numerical estimate without averaging does not suffice in estimating the velocities. Applying a moving average filter improves the resolution as the windows size increases, but at the cost of added phase delay, which is undesirable for feedback control.

Finally, the extended Kalman filter shows the best accuracy. However, one thing must be taken into account, namely that the EKF uses a "perfect" model in this case, as it is identical to the simulation model. In reality, this model will have inaccuracies compared to the real system.

In order to address this, the $\mathbf{Q}_k$ matrix must be tuned to account for the model inaccuracies. The $\mathbf{R}_k$ matrix must likewise be tuned in order to account for measurement noise.

Initially, $\mathbf{Q}_k$ will be identity and tuned as found necessary. $\mathbf{R}_k$, on the other hand can be set according to the variance of the measurements. **Appendix G** shows that the quantisation is much more significant than any measurement noise, implying the variance of the signal is close to zero.

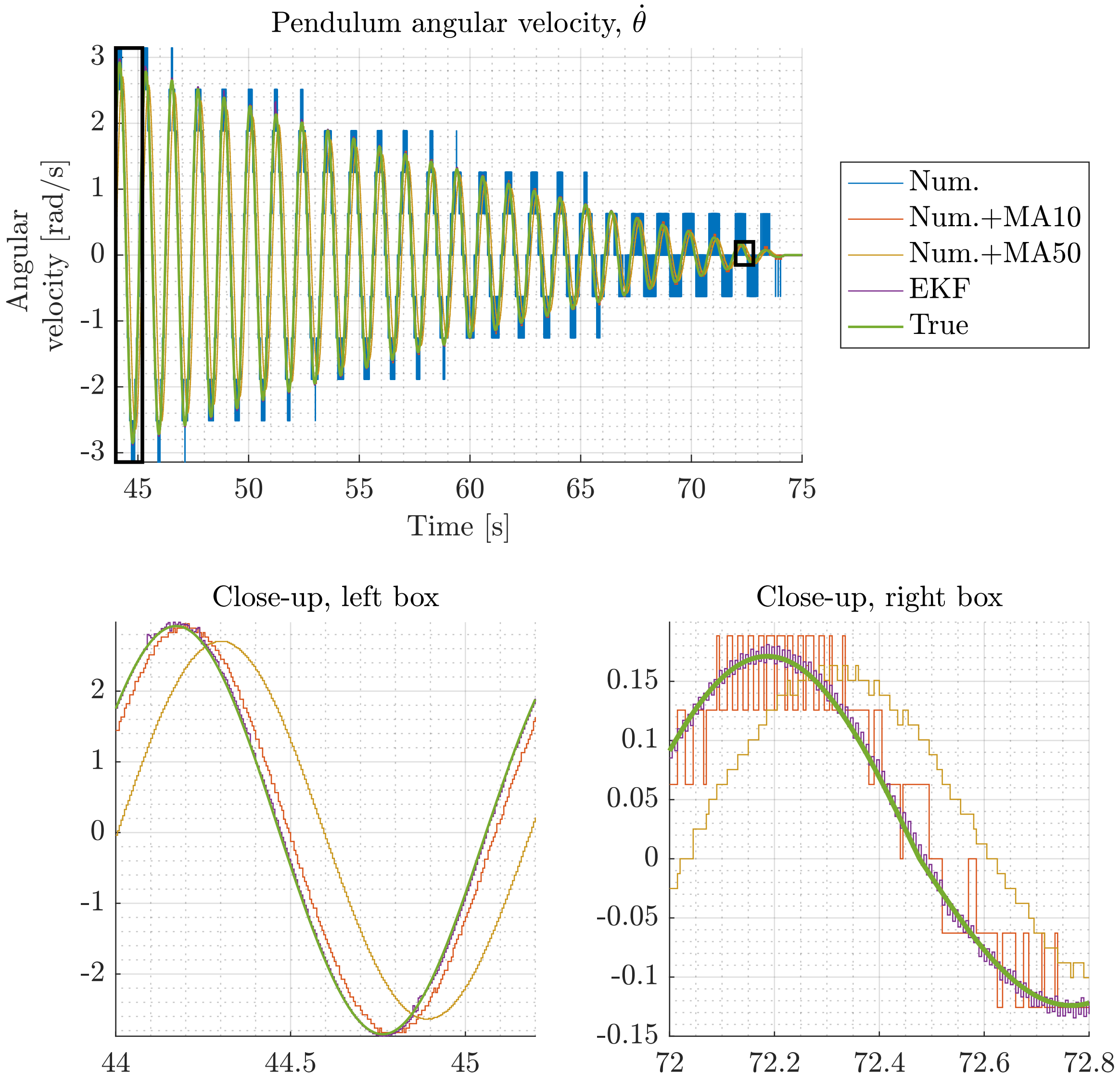


Figure 5.2: *Comparison of the different estimation methods. The top figure shows the last part of a simulation, and the bottom two are close-ups of the beginning and end. The different estimation methods are; the Numerical derivative (Num.), numerical derivative with 10-sample moving average filter (Num.+MA10), numerical derivative with 50-sample moving average filter (Num.+MA50), EKF and finally the model (True). Note that the raw numerical derivative has been omitted on the close-ups for readability.*

The EKF will be tested and tuned in **section 9.2**.

Control Strategy 6

This chapter will describe general control strategy for this project. The chosen approach is based on a two-step control strategy, such that one controller will swing up the pendulum, and another will stabilise it. The objective of the control is thus:

1) Set the pendulum in oscillation from initial state $\mathbf{x}_0 = [x_0 \quad \theta_0 \quad \dot{x}_0 \quad \dot{\theta}_0]^T = [0 \quad \pi \quad 0 \quad 0]^T$ and increase its mechanical energy to reach within $\theta = \pm 10°$ while keeping $|x^* - x| < 10$ cm, as stated in **section 3.1.1**.
2) Bring the pendulum from this oscillation to the control references, $\mathbf{x}^* = [x^* \quad 0 \quad 0 \quad 0]^T$.

For each step, multiple controller types will be tested and compared. The controller for the first step will be referred to as the *swing-up controller* while the second one will be called the *stabilising controller*. The necessity of two controllers is based on the idea that the swing-up controller must carefully increase the mechanical energy while keeping the cart from hitting the physical end-stops. On the other hand, the stabilising controller must keep the pendulum in an upright position and the cart at its reference, and is expected to be more aggressive.

An overview of the control strategy is shown in **figure 6.1**.

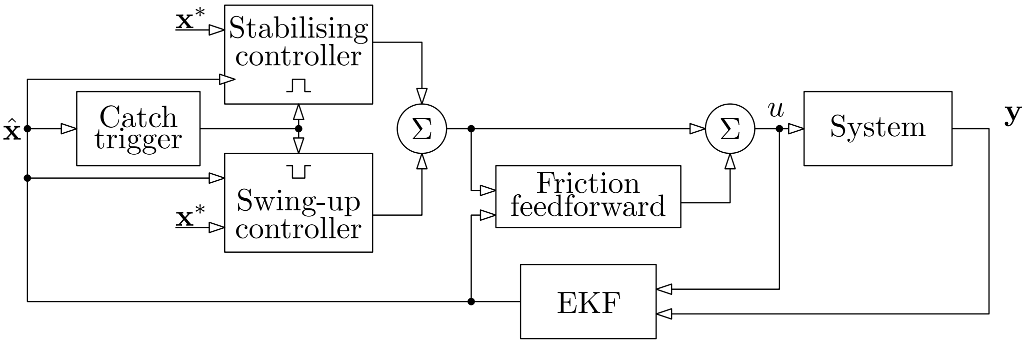


Figure 6.1: *Control system overview. The icons in the controller blocks indicate when each is active based on the output of Catch trigger.*

Before proceeding to the design, a few definitions and lemmas regarding system stability will be presented for later reference in **section 6.1**. Then, an analysis of the mechanical energy of the pendulum and the concept of heteroclinic orbits will be carried out in **section 6.2**.

As seen in **figure 6.1**, the cart friction $B_c(\dot{x})$ is feedforwarded due to its size and discontinuous nature. This will be further described in **section 6.3**.

The block in **figure 6.1** called *Catch trigger* is an algorithm that decides which one of the controllers should be active, and its functionality will be described in **section 6.4**.

The designs of the swing-up and stabilising controllers are described in **chapter 7** and **chapter 8**, respectively.

6.1 Stability

In this section, a few notions of stability and system convergence will be listed for later reference, as they will be used in the design process described in **chapter 7** (Control Design: Swing-up) and **chapter 8** (Control Design: Stabilisation).

Considering a general system of the form

$$\dot{x} = f(x) \tag{6.1}$$

the notion of stability is defined in the following along with Lyapunov's and LaSalle's theorems that will be used to prove stability.

Definition 6.1.1 (Stability). [4]
The equilibrium point $x = 0$ of ***equation (6.1)*** *is*

- *stable if, for each $\epsilon > 0$, there is $\delta = \delta(\epsilon) > 0$ such that*

$$\|x(0)\| < \delta \Rightarrow \|x(t)\| < \epsilon, \quad \forall t \geq 0 \tag{6.2}$$

- *unstable if it is not stable.*
- *asymptotically stable if it is stable and δ can be chosen such that*

$$\|x(0)\| < \delta \Rightarrow \lim_{t\to\infty} x(t) = 0 \tag{6.3}$$

Theorem 6.1.1 (Lyapunov's stability theorem). [4]
Let $x = 0$ be an equilibrium point for ***equation (6.1)*** *and $D \subset \mathbb{R}^n$ be a domain containing $x = 0$. Let $V : D \to \mathbb{R}$ be a continuously differentiable function such that*

$$V(0) = 0 \text{ and } V(x) > 0 \text{ in } D - \{0\} \tag{6.4}$$

$$\dot{V}(x) \leq 0 \text{ in } D \tag{6.5}$$

Then, $x = 0$ is stable. Moreover, if

$$\dot{V}(x) < 0 \text{ in } D - \{0\} \tag{6.6}$$

then $x = 0$ is asymptotically stable.

Theorem 6.1.2 (LaSalle's theorem). [4]
Let $\Omega \subset D$ be a compact set that is positively invariant with respect to ***equation (6.1)****. Let $V : D \to \mathbb{R}$ be a continuously differentiable function such that $\dot{V}(x) \leq 0$ in Ω. Let E be the set of all points in Ω where $\dot{V}(x) = 0$. Let M be the largest invariant set in E. Then every solution starting in Ω approaches M as $t \to \infty$.*

6.2 Mechanical energy and heteroclinic orbits

The swing-up control strategy is based on the mechanical energy of the system, and thus the reference shown in **figure 4.5** on page **14**, defining the system and its parameters, is revised. The potential energy reference, u_{ref}, is placed where the pendulum is at rest in its stable equilibrium point, as depicted in **figure 6.2**, namely

$$u_{ref} = m_p g l \tag{6.7}$$

in order to avoid negative potential energy.

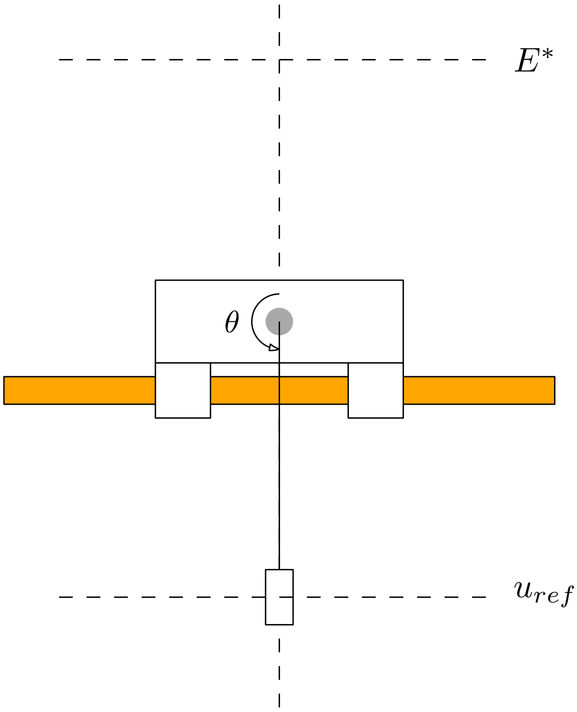


Figure 6.2: *Placement of the potential energy reference, at the center of mass of the pendulum when it is hanging straight down.*

This also indicates that the potential energy, when the pendulum is upright, corresponds to the energy reference

$$E^* = 2m_pgl \tag{6.8}$$

Swinging up the pendulum will specifically concern controlling the energy such that the pendulum reaches its heteroclinic orbits. These orbits are found by investigating the case where the energy of the pendulum is equal to E^*, i.e.

$$\begin{aligned} 0 &= E^* - E_p \\ &= 2m_pgl - \frac{1}{2}m_pl^2\dot{\theta}^2 - m_pgl(\cos\theta + 1) \\ &= (1 - \cos\theta)g - \frac{1}{2}l\dot{\theta}^2 \end{aligned} \tag{6.9}$$

Solving for $\dot{\theta}$ yields the expression for the heteroclinic orbits

$$\dot{\theta} = \pm\sqrt{\frac{2g}{l}(1 - \cos\theta)} \tag{6.10}$$

This is a function of θ, and becomes zero when $\theta = 2\pi k$ for $k \in \mathbb{Z}$. The trajectories of **equation (6.10)** is visualised for $\{\theta \in \mathbb{R} \mid 0 \leq \theta \leq 2\pi\}$ in **figure 6.3** in bright red.

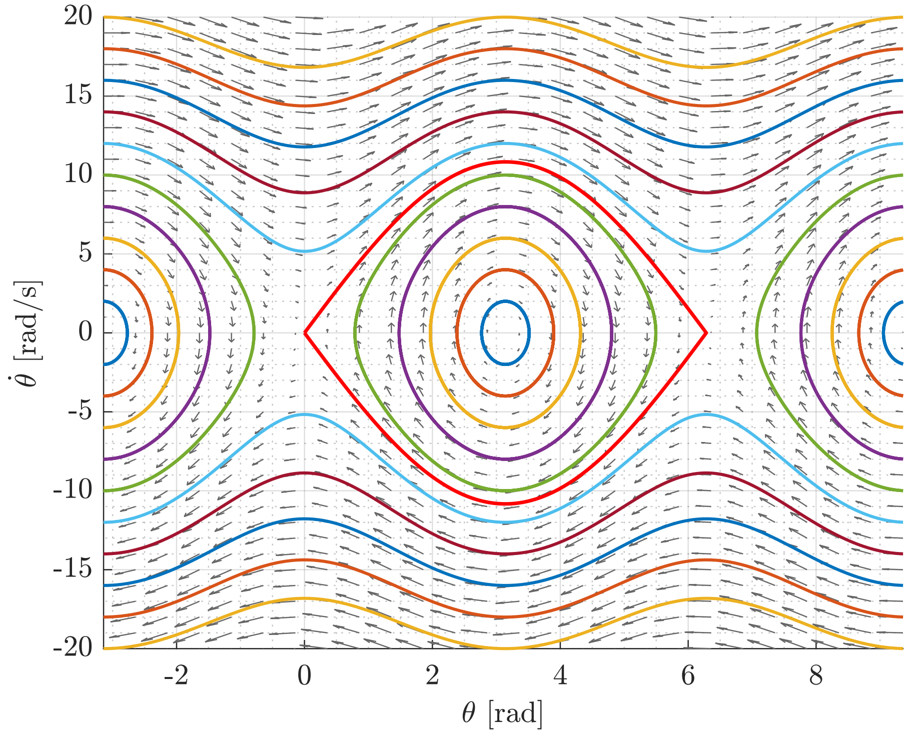


Figure 6.3: *Phase plot of the system* ***without*** *friction.*

At either side of a heteroclinic orbit exists an equilibrium point, specifically at $(\theta, \dot{\theta}) = (0, 0)$ and $(\theta, \dot{\theta}) = (2\pi, 0)$. Thus, a heteroclinic orbit is defined as a trajectory which connects two different equilibrium points. Physically, however, these two points corresponds to the same vertical angle of the pendulum. The trajectories outside the heteroclinic orbits are trajectories where the energy in the system exceeds the energy reference, and they will remain on the indicated paths as long as there is no friction. The trajectories inside are thus trajectories where the energy is lower than the reference.

Including friction in the system equations will yield trajectories like depicted in **figure 6.4**.

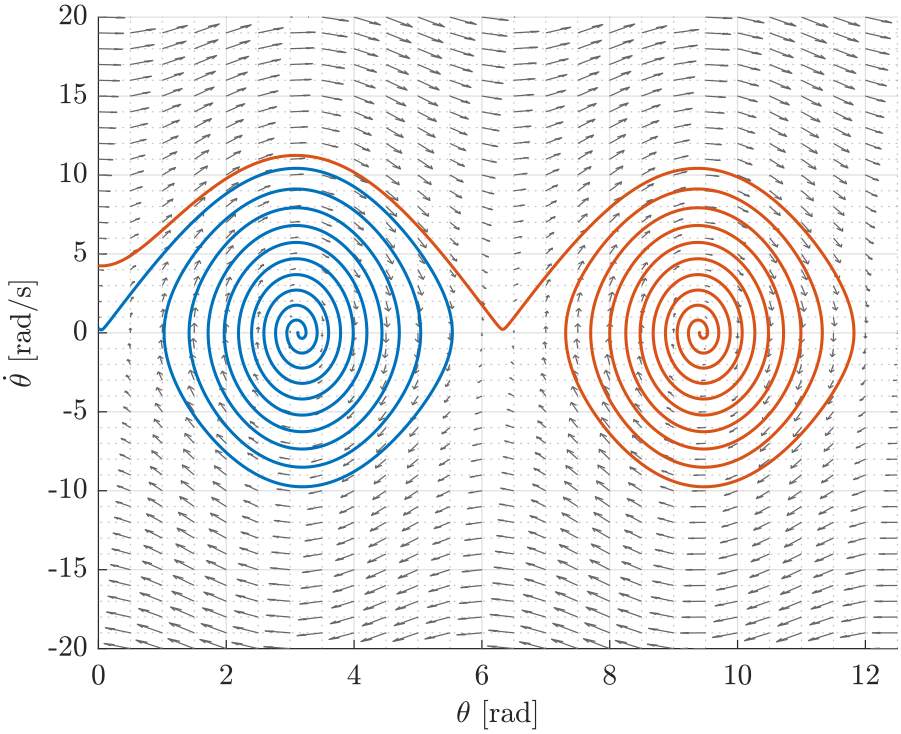


Figure 6.4: *Phase plot of the pendulum system* ***with*** *friction. The friction coefficients have been chosen larger than the actual values for illustrative purposes.*

The friction removes energy from the system and creates stable equilibria at the points $\theta = (2k+1)\pi$ for $k \in \mathbb{Z}$. That is, when within the heteroclinic orbits of two equilibrium points, the system will converge towards the stable equilibrium point between them, if no control input is present.

From the above analysis, the purpose of the swing-up controller can be described as to bring the system from its initial state to the heteroclinic orbits.

6.3 Cart friction feedforward

From **section 4.3.1** and **appendices B** and **C**, the friction exerted on the cart has proved to be troublesome. While viscous friction is insignificant, the Coulomb term is direction-dependent and shows tendencies of depending on the position as well.

To simplify the design process in **chapters 7** and **8**, the Coulomb friction of the cart is omitted from the model equations, and added to the controller output.

By removing the Coulomb friction, the friction vector from **section 4.4** is redefined as

$$\mathbf{B}(\dot{\theta}) = \begin{bmatrix} 0 \\ c_p \tanh(k\dot{\theta}) + v_p\dot{\theta} \end{bmatrix} \tag{6.11}$$

for use in **chapters 7** and **8**, and the cart friction is added by feedforward as depicted in **figure 6.1**, with

$$B_{ff}(\dot{x}, u) = c_c(\dot{x}) \tanh(k \cdot u), \qquad c_c(\dot{x}) = \begin{cases} c_{c,R} & \text{if } \dot{x} \geq 0 \\ c_{c,L} & \text{if } \dot{x} < 0 \end{cases} \tag{6.12}$$

where u is the output from the active controller and k is a tuning parameter that determines how steep $\tanh(\cdot)$ is through zero. The $\tanh(\cdot)$ in the feedforward is dependent on the sign of u rather than $\dot{x}$ in order to avoid e.g. adding positive friction compensation to negative u, effectively counteracting the control action of the active controller.

6.4 Catch trigger

While the control scheme is initialised with the swing-up controller, an algorithm is needed to determine when to trigger the switch to the stabilising controller.

The conditions under which the stabilising controller will be activated are based on two criteria:

- Pendulum energy: $|E^* - E_p| \leq 0.05E^*$
- Angular span: $|\theta| \leq 30^\circ = 0.5236\ \mathrm{rad}$

where $E_p = \frac{1}{2}m_p l^2 \dot{\theta}^2 + m_p g l(cos\theta + 1)$ is the mechanical energy of the pendulum, i.e. without considering the cart.

If none, or only one of these two criterias are met, the output of the swing-up controller will be applied to the system. When both are satisfied, the output from the stabilising controller is applied to the system.

Note that when one controller is active, the other remains passive. In terms of implementation, it is beneficial to completely skip computing the output of the passive controller.