Model Predictive Control of an Inverted Pendulum

Anuar Nurlybayev, Pawan Bhatla, Rahul Panicker

Project Group
ie3
July 2023

1. Examinor: Prof. Dr.-Ing. Timm Faulwasser
2. Examinor: Jens Püttschneider, M.Sc.

Abstract

The stabilization of unstable systems is an important area of research in the field of control systems engineering. Effective control strategies have their origins in human behaviour. The use of predictions of expected behaviour in determining a control strategy is intuitively apparent.

This project explores the development of such a predictive algorithm, namely a model predictive control algorithm for the swing-up and stabilization of an Inverted Pendulum at the upper, unstable equilibrium point. The inverted pendulum is a popular example for advanced control techniques due to the nature of the system, that is, a highly non-linear system with stabilization around an unstable equilibrium point.

The pendulum is mounted on a cart which is connected with a chain to the first encoder, that measures the movement of the cart. The second encoder measures the rotational dynamics of the pendulum. The developed predictive control algorithm utilizes optimal control to recalculate the input signal to the pendulum and decide the current control output. This recalculation allows the pendulum to reach the desired set point in an optimal fashion. The optimal control problem is built to be solved on a host pc, with Python the programming language of choice. The microcontroller unit on the pendulum, receives the encoder readings and sends it to the host pc.

Besides the control of the pendulum's angle, it is also necessary to control the angular velocity, linear position and linear velocity of the cart. The algorithm was able to stabilize the pendulum at the central position and in the upright position.

Contents

List of Figures

1 Introduction

1.1 Motivation

In the realm of control systems, the inverted pendulum stands as a classic benchmark that fascinates both educators and students. In this project, we delve into the reasons behind the inverted pendulum's popularity as a control benchmark and test an advanced control scheme, namely model predictive control, on said benchmark.

The motivation behind the popular choice of the inverted pendulum as an example for teaching control theory stems from its inherent complexity and non-linearity. Model predictive control, on the other hand, is a relatively young algorithm that combines the areas of optimal control and model-based feedback control to create a controller greater than the sum of its parts.

Designing and implementing control algorithms for the inverted pendulum gives practical insights into concepts such as feedback control, stability analysis, and system identification. In various domains, such as robotics, aerospace, and automation, systems exhibiting similar dynamics and control challenges can be found.

In summary, the inverted pendulum's excellence as a popular control benchmark lies in its ability to demonstrate core principles of control theory in action, and establish connections to real-world systems. By mastering the control of the inverted pendulum, we acquire valuable skills and insights that are transferable to real-world applications, paving the way for innovation and progress in the field of control systems engineering

1.2 Existing Controllers

The control of inverted pendulum systems has been a topic of extensive research in the field of control systems engineering. Over the years, various controllers have been developed to tackle this task. The controllers implemented in the current iteration of the inverted pendulum involve two major controllers working in tandem, for the two major tasks involved in the control of the inverted pendulum: swing up and stabilization.

Energy-Based swing-up controller addresses a crucial aspect of the inverted pendulum system, the swing-up maneuver. This involves imparting sufficient energy to the pendulum to raise it from the downward hanging position to the upright position. This controller focuses on achieving this energy transfer while simultaneously controlling the system's dynamics. By exploiting the system's potential energy, this controller allows for a smooth and efficient swing-up motion, paving the way for subsequent stabilization using other control techniques.

Linear Quadratic Regulator (LQR) is effectively the simplest optimal controller. A classical control technique widely used for designing optimal control strategies for linear systems. It aims

to find a control law that minimizes a quadratic cost function, considering both the system's states and control inputs. The LQR controller for the inverted pendulum system utilizes the system's linearised dynamics and employs state feedback to stabilize the pendulum in an upright position. By formulating the problem as an optimization task, the LQR controller provides an optimal control solution, effectively maintaining stability.

The combination of the LQR controller and the energy-based swing-up controller forms the swing up and stabilization controller for the inverted pendulum system. The energy-based swing-up controller enables the system to transition from an unstable state to a controllable region, while the LQR controller maintains stability and optimal performance once the pendulum is upright. This combined approach allows for a comprehensive control strategy that addresses both the swing-up maneuver and the subsequent stabilization, ensuring the successful control of the inverted pendulum system.

1.3 Advanced Controllers

On further reading one can understand that there are many advanced control schemes that have been devised for the swing up and stabilization of the inverted pendulum system, as alluded to in [1, 2] and one of the frontrunners of these advanced control techniques is the concept of model predictive control. The reason for the popularity of MPC for this particular application are manifold, the primary reason is the ability of MPC to handle non-linear systems along with state and input constraints. This means that only one controller is necessary for both parts of the pendulum motion as swing up and stabilization can both be done by the same controller.

2 Basics

The aim of this project is to study and devise an advanced control method, in this case model predictive control (MPC), for a highly non-linear system with an unstable equilibrium, in this case an inverted pendulum. As the name suggests, model predictive control involves the mathematical model of the system. By solving an optimization problem using states that are fed back to the this model, one can predict the optimal state trajectory for the next N time steps. This N is called the prediction horizon of the controller and is an important parameter that can be tuned for better performance.

In this chapter we will look at the mathematical modelling of this system as well some basic MPC theory that enables one to design a stable and optimal controller.

2.1 Inverted pendulum

The inverted pendulum or the "cart-pole" system, as mentioned before is a standard test case for new control strategies. For MPC it is first important to model this system mathematically, the following section gives an brief overview of the real system followed by the mathematical modelling of this system.

2.1.1 Physical System

The outer construction of the pendulum, as shown in Figure 2.2, consists of extruded aluminium and custom-designed elements made of PET-G material. The internal components, shown in Figure 2.1 are also custom-designed and made of either PET-G or PLA+ [3].

The internal components consist of the following:

- Pendulum connector : Connects the pendulum arm to the cart.
- Rotary encoder :To measure the angle of rotation of the pendulum arm.
- Cart rails : For cart movement.
- Motor : A stepper motor which moves the cart.
- Motor driver : Used to control the stepper motor.
- MCU : The ESP32 microcontroller unit responsible for generating the PWM signal for the motor.
- Limit switches : 2 limit switches at either side of the cart mark the limits of the cart.

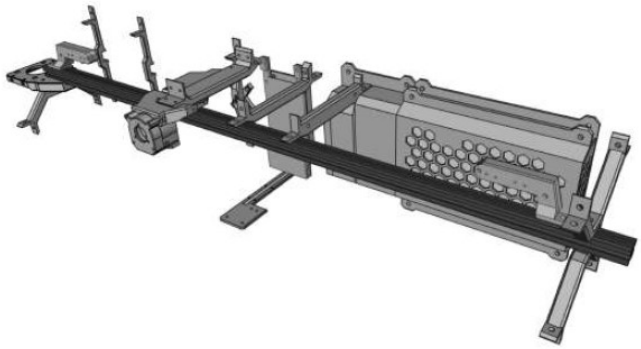


Figure 2.1: internal components [3].

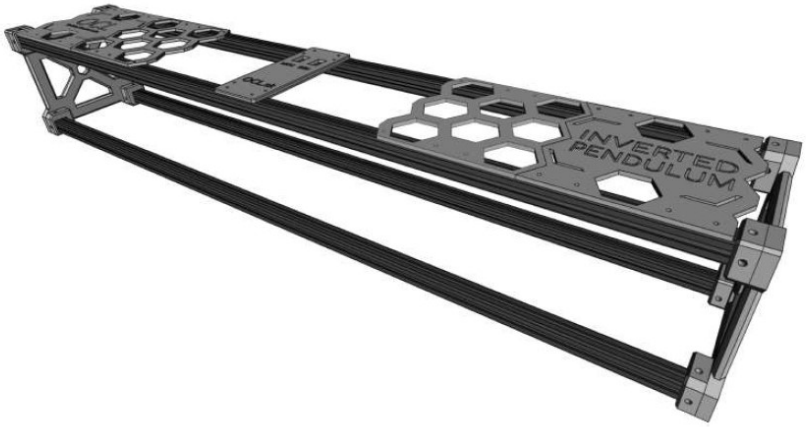


Figure 2.2: outer covering [3].

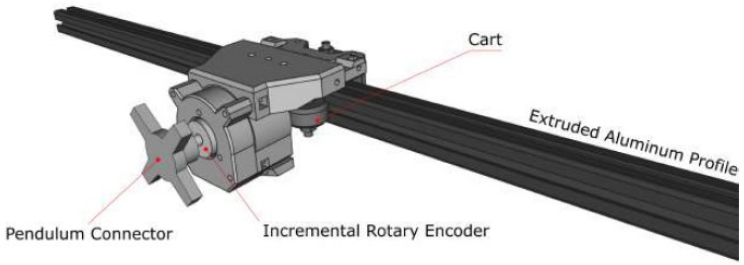
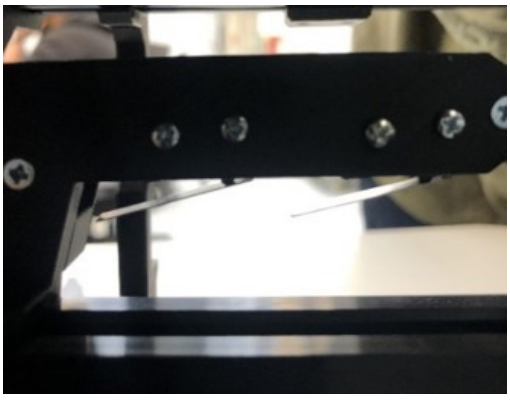


Figure 2.3: pendulum cart and rotary encoder [3].

Figure 2.4: inverted pendulum : front view

(a) motor and encoder.

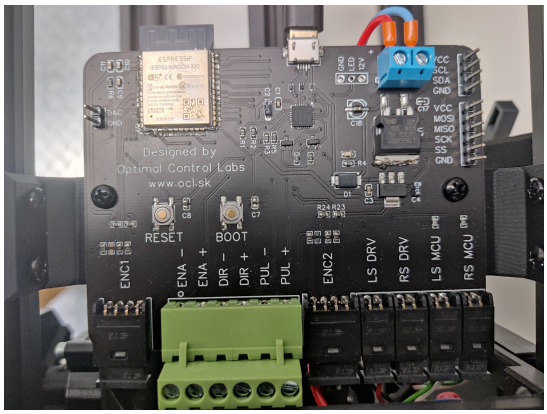


(b) limit switch.

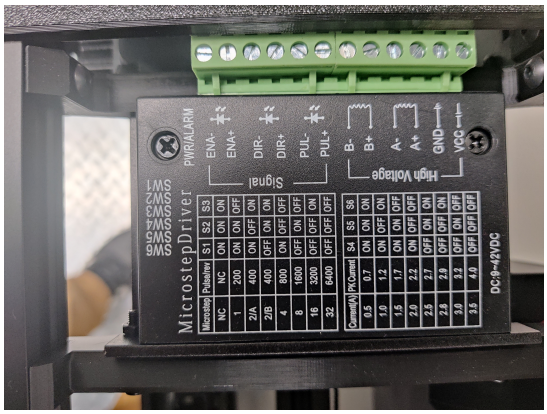
Figure 2.5: pendulum components: motor and limit Switch.

The motor is controlled using a motor driver and an ESP32 microcontroller, shown in Figure 2.6, that produces Pulse Width Modulated (PWM) signals to control the step angle to produce desired velocities. Since the input chosen was the cart acceleration, we have to first convert this to the cart velocity at current time. This can be done using the relation,

$$v_{cart} = u \cdot dt + v_0$$



(a) ESP32 microcontroller



(b) motor driver

Figure 2.6: pendulum components: microcontroller and driver

where u is the cart acceleration, dt is the sampling time of the pendulum and v_0 is the velocity at the previous time step. This is then converted to the frequency of the PWM signal using the relation

$$f_{pwm} = v_{cart} \cdot 20,000$$

This is explained in more detail in later chapters.

2.1.2 Mathematical Model

We first start by defining a few variables and what they represent in our inverted pendulum system :

- Cart position: $x(t)$
- Pendulum centre of gravity x-Position: $x_s(t)$
- Pendulum centre of gravity y-Position: $y_s(t)$
- Pendulum angle measured from the top equilibrium: $\theta(t)$
- Pendulum length: ℓ
- Pendulum mass: m
- Pendulum moment of inertia: $J = m\ell^2/12$
- Angular friction coefficient: b
- Contact forces between pendulum and cart: $F_x(t), F_y(t)$
- Gravity constant: g

We then define the coordinates,linear velocity and acceleration of the centre of gravity of the pendulum arm as (x_s, y_s), $(\dot{x}_s, \dot{y}_s)$ and $(\ddot{x}_s, \ddot{y}_s)$ respectively. These can then be calculated as follows:

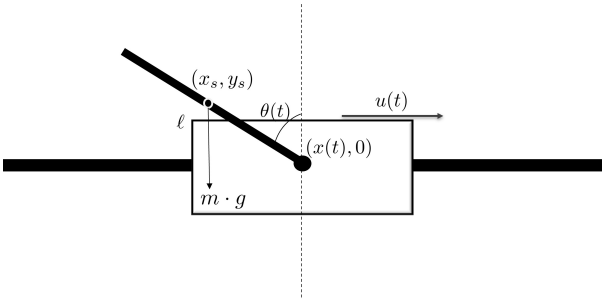


Figure 2.7: pendulum schematic

$$
\begin{aligned}
x_s &= x - \frac{\ell}{2}\sin(\theta) \\
\dot{x}_s &= \dot{x} - \frac{\ell}{2}\dot{\theta}\cos(\theta) \\
\ddot{x}_s &= \ddot{x} - \frac{\ell}{2}\ddot{\theta}\cos(\theta) + \frac{\ell}{2}\dot{\theta}^2\sin(\theta) \\
y_s &= \frac{\ell}{2}\cos(\theta) \\
\dot{y}_s &= -\frac{\ell}{2}\dot{\theta}\sin(\theta) \\
\ddot{y}_s &= -\frac{\ell}{2}\ddot{\theta}\sin(\theta) - \frac{\ell}{2}\dot{\theta}^2\cos\theta
\end{aligned}
$$

In first principle modelling one uses basic physical conservation laws to model the system, in chemical reactors for example, one can use molar balance equations. In this case we use force balance equations.

So applying force balance in x-direction of the pendulum we get:

$$m\ddot{x}_s = F_x \tag{2.1}$$

Similarly, applying force balance in y-direction of the pendulum, we get:

$$m\ddot{y}_s = F_y - mg \tag{2.2}$$

Another possible quantity that can be balanced is torque, so applying torque balance to the pendulum's centre of gravity (CoG), we obtain:

$$J\ddot{\theta} = F_x\frac{\ell}{2}\cos(\theta) + F_y\frac{\ell}{2}\sin(\theta) - b\dot{\theta} \tag{2.3}$$

Plugging (2.1) and (2.2) into (2.3) to obtain

$$\left(J + m\frac{\ell^2}{4}\right)\ddot{\theta} = m\ddot{x}\frac{\ell}{2}\cos(\theta) + mg\frac{\ell}{2}\sin(\theta) - b\dot{\theta} \tag{2.4}$$

Since $J = m\ell^2/12$ we get $(J + m\ell^2/4) = m\ell^2/3$. This results in a reformulated equation which reads:

$$\ddot{\theta} = \frac{3\cos(\theta)}{2\ell}\ddot{x} + \frac{3\sin(\theta)}{2\ell}g - \frac{3b}{m\ell^2}\dot{\theta} \tag{2.5}$$

$$\begin{aligned} \ddot{\theta} &= \frac{3g}{2\ell}\sin(\theta) - \frac{3b}{m\ell^2}\dot{\theta} + \frac{3}{2\ell}\cos(\theta)u \\ \ddot{x} &= u \end{aligned} \tag{2.6}$$

We can also redefine the friction coefficient $b' = \frac{3b}{m\ell^2}$, with this we get

$$\begin{aligned} \ddot{\theta} &= \frac{3g}{2\ell}\sin(\theta) - b'\dot{\theta} + \frac{3}{2\ell}\cos(\theta)u \\ \ddot{x} &= u. \end{aligned} \tag{2.7}$$

from the above derivation of the system model, one can decide on the states, inputs and outputs of the inverted pendulum model as follows :

- States:
  - $x_1(t) \doteq \theta(t)$ - angular displacement of pendulum
  - $x_2(t) \doteq \dot{\theta}(t)$ - angular velocity of the pendulum
  - $x_3(t) \doteq x(t)$ - linear displacement of the cart
  - $x_4(t) \doteq \dot{x}(t)$ - linear velocity of the cart
- Input:
  - $u(t) \doteq \ddot{x}(t)$ i.e. input = cart acceleration
- Measured quantities:
  - $x_1(t)$ from pendulum encoder
  - $x_3(t)$ from cart encoder

And so the full dynamic equation of the inverted pendulum can be written as

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} x_2 \\ \frac{3g}{2\ell}\sin(x_1) - \frac{3b}{m\ell^2}x_2 + \frac{3}{2\ell}\cos(x_1)u \\ x_3 \\ u \end{bmatrix} \tag{2.8}$$

2.2 Model Predictive Control

As mentioned before, model predictive control is a very popular strategy among other advanced control methods. There are many reasons for its popularity, some being that it is able to consider the entire non-linear system without any linearising approximations and it is also very adept at handling constraints.

Model predictive control is intrinsically linked with the concept of optimal control as MPC involves the solution of an open loop optimal control problem (OCP) at every time step. In the simple MPC scheme shown in Figure 2.8, the optimal control inputs for the next p steps are calculated and the very first input is applied to the system. The states are then measured after this application and the process is repeated at every time step. The parameter p here is known as the prediction horizon and is an important parameter that can be tuned for better results.

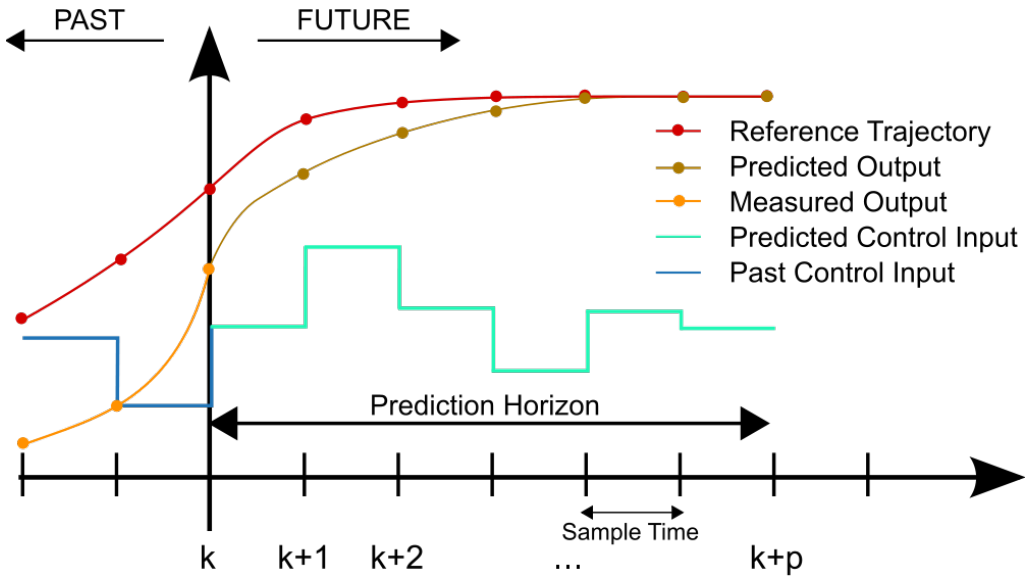


Figure 2.8: A simple MPC Scheme [4]

2.2.1 Optimal Control Problems(OCP)

An optimal control problem involves the mathematical formulation and analysis of determining an optimal trajectory for a dynamic system, subject to constraints and other performance criteria (minimizing the cost function). The OCP for a non-linear system with a quadratic cost function and constraints is as follows:

$$\min_{\substack{x_0,x_1\ldots x_N \\ u_0,u_1\ldots u_{N-1}}} \sum_{k=0}^{N-1} \left(x_k^T Q x_k + u_k^T R u_k\right) + x_N^T Q_f x_N^T \tag{2.9}$$

$$\begin{aligned} \text{subject to} \quad & x_{k+1} = f(x_k, u_k) \\ & x_0 = x(0) \\ & u_k \in \mathbb{U}, x_k \in \mathbb{X} \\ & \forall k \in [0, N-1] \\ & x_N \in \mathbb{X}_f \end{aligned} \tag{2.10}$$

Where x_k and u_k are the states and inputs respectively of the k-th time step and x_N denotes the terminal state.

Q and R matrices are the state and input weight matrices respectively and Q_f is the weight of the terminal state.

The constraints include the regular state and input constraints along with the dynamic model of the system as an additional continuity constraint.

The cost function can be extended to include a different goal state by replacing the x_k term with $(x_k - x_{goal})$.

2.2.2 Stability

While no concrete stability guarantees for Non-Linear MPC (NMPC) can be made, the theory behind Linear MPC has been well established for some time.This can be in practice adapted to NMPC.

The stability analysis of Linear MPC can be divided mainly into two parts :

a. Recursive Feasibility: Since the MPC algorithm involves the solution of an Optimization problem at every time step, one has to ensure that the problem remains feasible. This is done by proper choice of terminal sets. If the terminal set is chosen to be control invariant, one can ensure recursive feasibility. Written down mathematically, if $\forall x \in \mathbb{X}_f \; \exists \; \kappa(x) \in \mathbb{U}$ such that $f(x, \kappa(x)) \in \mathbb{X}_f$, then one can achieve recursive feasibility [5, 6].

a. Stability of the Origin: Stability of the MPC algorithm in inherently tied to the choice of the terminal ingredients of the OCP. Recursive feasibility was achieved by selecting a suitable terminal set and likewise the stability at the origin can be confirmed by the right choice of terminal weights. A possible choice of terminal set is as follows:

For positive definite Q and R matrices, solve the infinite horizon LQR problem. Since the infinite horizon LQR results in a constant quadratic optimal cost P, this can be assigned as the terminal weight to ensure stability about the origin [5–7].

2.2.3 Optimality

Good choices of terminal ingredients can lead to stability. But sub-optimalities can arise due to the choice of prediction horizon and for any given initial condition there exists a prediction horizon N_{min} that is optimal for the constrained closed loop system.

Any solution that satisfies the constraints is termed to be a feasible solution, whereas the feasible solution with the least cost is called the Optimal solution. For any prediction horizon, $N > N_{min}$ the performance of the closed loop does not visibly improve, and if $N < N_{min}$ and the optimization problem is feasible, the closed loop can still be close to optimality because of the receding horizon (feedback) implementation.

3 Development

Since the implementation of the MPC algorithm is on a real system, and optimization is very computationally heavy, performing the computation on the ESP32 microcontroller was not the best way forward. Thus, communication was established between a host PC where all the heavy computation would take place and the microcontroller would be responsible for applying the calculated input and measuring the states of the pendulum, followed by relaying this information to the host PC. A schematic of this setup can be found in Figure 3.1.

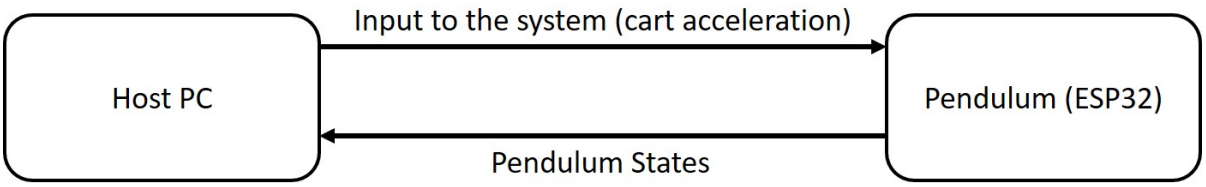


Figure 3.1: Concept Schematic

In this chapter we will take a look at both halves of this setup with a look into the MPC algorithm on the host PC side, the Arduino interface on the pendulum side and the established communication channel connecting the two halves.

3.1 MPC

As mentioned before, the first half of the system is the host PC where optimization is performed, with Python as the programming language of choice. For the purpose of easy access and understanding we encapsulated the parts into separate classes on python. 2 major tasks are carried out on the PC, namely : optimization and interfacing. In this section we will look at the optimization part in detail, with the interfacing being dealt with in further sections.

3.1.1 Optimization on Python

As suggested in Chapter 2, the optimization problem uses a simple quadratic cost setup. The python script that performs the task of optimization is done with the help of the ACADOS package. ACADOS is the successor of the ACADO software package developed at KU Leuven and University of Freiburg by the team of Prof. Moritz Diehl [8].

ACADOS offers extremely fast optimization time and makes it possible to run near the real time optimization requirement of the inverted pendulum.

The state is measured and passed on to the *step*() function that calculates the next controller step. The algorithm for which is as follows:

Algorithm 1 MPC Control step

Data: x_k: Data at time step k
Solver : a prebuilt ACADOS OCP solver object to be passed as an input to the *solve*() function
Result: u: Control input for time step $k+1$
if *first time = "True"* **then**
| initial guess for solver object ← first initial guess
else
| initial guess for solver object ← warm started initial guess
end
x_{opt}, u_{opt} ← Solve(solver,initial guess)
$u \leftarrow u_{opt}[0]$ ▷ First Optimal Input

Return : u

You are free to choose the first initial guess to the solver. Here, we load the solution of the first optimal control problem from a file as the first initial guess.

As explained before, MPC involves using the first part of the optimal input trajectory to apply on the system. Although the rest of the sequence of inputs is not utilized, it can be reused in the next sampling instant to generate a rough estimate of optimal solution of the next QP problem. This technique is called warm-start and its primary reason is to reduce computational complexity in solving the next QP [**Otta2015**].

The optimization strategy used is sequential quadratic programming (SQP). This involves breaking down the problem into smaller quadratic program (QP)subproblems and optimizing each individual quadratic program [9].

These subproblems are solved using the partial condensing HPIPM(High Performance Interior Point Method). This involves the construction of a barrier function that is used to manage the inequality constraint [9].

Different solver options were tested and while the regular SQP method was far too slow in calculating the solutions, SQP with real time iterations (SQP_RTI) [**Diehl2005**, 8] was able to compute the solutions extremely fast due to a smaller local horizon . And in the interest of speed, the hessian, which is necessary for the calculation of optimal trajectories, is approximated using the Gauss-Newton approximation. This makes the problem more efficient.

The optimization problem solved at every iteration is structured like the one described in previous chapters, which is as follows:

$$\min_{\substack{x_0,x_1\ldots x_N \\ u_0,u_1\ldots u_{N-1}}} \sum_{k=0}^{N-1} \left(x_k^T Q x_k + u_k^T R u_k\right) + x_N^T Q_f x_N^T \tag{3.1}$$

$$\begin{aligned} \text{subject to} \quad & x_{k+1} = f(x_k, u_k) \\ & x_0 = x(0) \\ & u_k \in \mathbb{U}, x_k \in \mathbb{X} \\ & \forall k \in [0, N-1] \end{aligned} \tag{3.2}$$

The values of Q_k, R and Q_f were tuned for optimal performance. The final values of these parameters are:

$$Q_k = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 10^{-4} & 0 & 0 \\ 0 & 0 & 10^{4} & 0 \\ 0 & 0 & 0 & 10^{-2} \end{bmatrix} \qquad R = \begin{bmatrix} 10 \end{bmatrix}$$

Q_f is decided by solving the infinite horizon LQR problem with the above mentioned Q_k and R values. This is common practice to achieve stability with MPC.

3.2 Pendulum

The pendulum is operated by the ESP32 microcontroller unit (MCU), a powerful and generic MCU with integrated Wi-Fi and Bluetooth connectivity for a wide range of applications [10]

Since the ESP32 is an arduino peripheral one can use the arduino IDE to code instructions into the MCU. The ESP main *loop*() function involves receiving, storing new data using the $recvWithEndMarker()$ and $storeNewData()$ functions. The algorithm for the main loop is as follows:

Algorithm 2 Pendulum main loop

Data: x_k: linear position data at time step k
θ_k: angular position data at time step k
dx_k: linear velocity data at time step k
$d\theta_k$: angular velocity data at time step k
u : received floating number of the input acceleration

Result: u_k: The frequency needed to generate a desired linear velocity corresponding to the input signal.

$u \leftarrow read()$ ▷ read() function reads a byte of serial data and stores it in an array

$\theta_k \leftarrow$ (PendulumEncoderPulses + 800) * $2\pi/1600$
$x_k \leftarrow$ MotorEncoderPulses/20,000 [3]

if *current time - start time* $>$ *5ms* **then**
| $dx_k, d\theta_k \leftarrow$ velocities calculated from measured valued of displacement
else
| current time $\leftarrow$ esp_timer_get_time() ▷ This line updates the current time variable
end
sendStates($x_k, dx_k, \theta_k, d\theta_k$) ▷ Sends the states to the PC every 5ms
$u_k \leftarrow$ real dynamics() ▷
start time $\leftarrow$ current time ▷ This reassigns the start time as the current time
Return : u_k

The *real_dynamics*() function calculates the value of the frequency of the PWM signal needed to generate the required velocity. The values of x_k and θ_k values from the encoders of the pendulum.

$$\theta_k = \frac{PendulumEncoderPulses + 800) \cdot 2\pi}{1600} \tag{3.3}$$

$$x_k = \frac{MotorEncoderPulses}{20,000} \tag{3.4}$$

The encoder generates 400 pulses per revolution per channel. It has two channels, therefore, if the rising and falling edges are encountered, the resulting resolution is 1600 pulses per revolution. If we want to convert the pulses to angle in radians the simple math can be incorporated. And since we want the initial stable equilibrium position to correspond to an angle of π radians, we add an additional 800 to the value of the rotary encoder pulses. Similarly, the position can also be calculated by dividing the motor encoder pulses by a value of 20,000 [3].

3.3 Interfacing

Now that we have established the 2 sides of communication channel, we can elaborate on the channel itself. In this section we will explain in detail the communication procedure on both the PC side and the microcontroller side.

3.3.1 PC interface

As mentioned before, python is the language used for optimization purposes and therefore is used to develop the PC interface of the communication channel as well. pySerial is a popular package for USB communication on Python and that is the package used in this particular application as well.

Reading states from microcontroller:

The Arduino has been programmed to continuously send the pendulum states every 5ms. To trigger the measurement of states on the PC side we use the *measure*() function.

The read_all() method in the serial module will read all the data that is available on the serial port and store it in a variable, this forms our buffer. We then extract the last 42 bytes from this buffer. With 4 states, one input and one delimiting character ("?") the total message size is 21 bytes. We extract the most recent 42 bytes in case of incomplete messages.

The extracted message is then saved 4 bytes at a time to extract the 4 states.

Applying input:

After calculating the input, u, as directed in section 3.1.1, this input has to be applied to the pendulum cart. This can be done using the *apply*() function on the PC side. This simply uses the *struct.pack*() method to convert the input to its binary representation and *pySerial.write*() function to send it to the pendulum via the connected serial port.

3.3.2 Pendulum interface

As mentioned before, the pySerial package is used to establish a communication channel between the PC and the ESP32 microcontroller which controls the motor which in turns controls the pendulum. The ESP32 generates pulse width modulated (PWM) signals that cause motion in the stepper motor, generated by the LEDC peripheral on the EPS32, and thereby moving the pendulum cart.

Receiving from PC: the read() function

The read function involves sequentially reading a byte of serial data and storing it in an array until the end of the message. This message is the input to the pendulum. The delimiter for this message is the $'\backslash n'$ symbol.

Sending to PC: real_dynamics() and sendStates() functions

The controller's output data is a sequence of four states, each of 4 bytes with the symbol,'?', denoting the start of this sequence. To decrease the length of the message and increase the communication speed, the recorded floating point data was written as an address using IEEE754 standards instead of sending ASCII data of each number in the message. This is written into the serial buffer by the **sendStates()** function.

The **real_dynamics()** function is responsible for the calculation of the frequency of the PWM signal from the received value of cart acceleration. This can be done using the following relation:

$$f_{pwm} = v_{cart} \cdot 20,000 \tag{3.5}$$

This relation comes from the formula for

While calculating the frequency of the PWM signal, one has to account for the application of both high and low frequency signals. This can be done by adjusting the duty cycle resolution of the ESP32 clock. Since frequency and duty resolution are inversely related one has to choose a resolution that can handle higher as well as lower frequencies. This can be done using the formula provided in [10].

$$Duty = \log_2 \left(\frac{f_{clock}}{f_{pwm} * N_{clock}} \right) \tag{3.6}$$

$$N_{clock} = \mathrm{A} + \frac{\mathrm{B}}{256} \tag{3.7}$$

Here N_{clock} stands for the effective division factor of the arduino clock and the A, B here stand for the integer and fractional part of the division factor of the divider within the timer.

From this one can calculate that for high velocities we require a duty resolution of 1-11 bits and for lower velocities we require a duty resolution of 8-20 bits. From this a resolution of 8 bits was chosen. In theory resolutions of 9,10 and 11 bits are also valid choices.

4 Results

The results of this experiment can be broadly classified into 2 categories, namely simulation results and those of the real system. The following chapter looks into these results in detail.

4.1 Simulation Results

To test the accuracy of the model and the speed of the optimization algorithm, simulations were performed and studied.

For the implementation of the MPC algorithm, the simulation was conducted on the PC, with a simulated pendulum class with internal functions that simulate the behavior of a real pendulum.

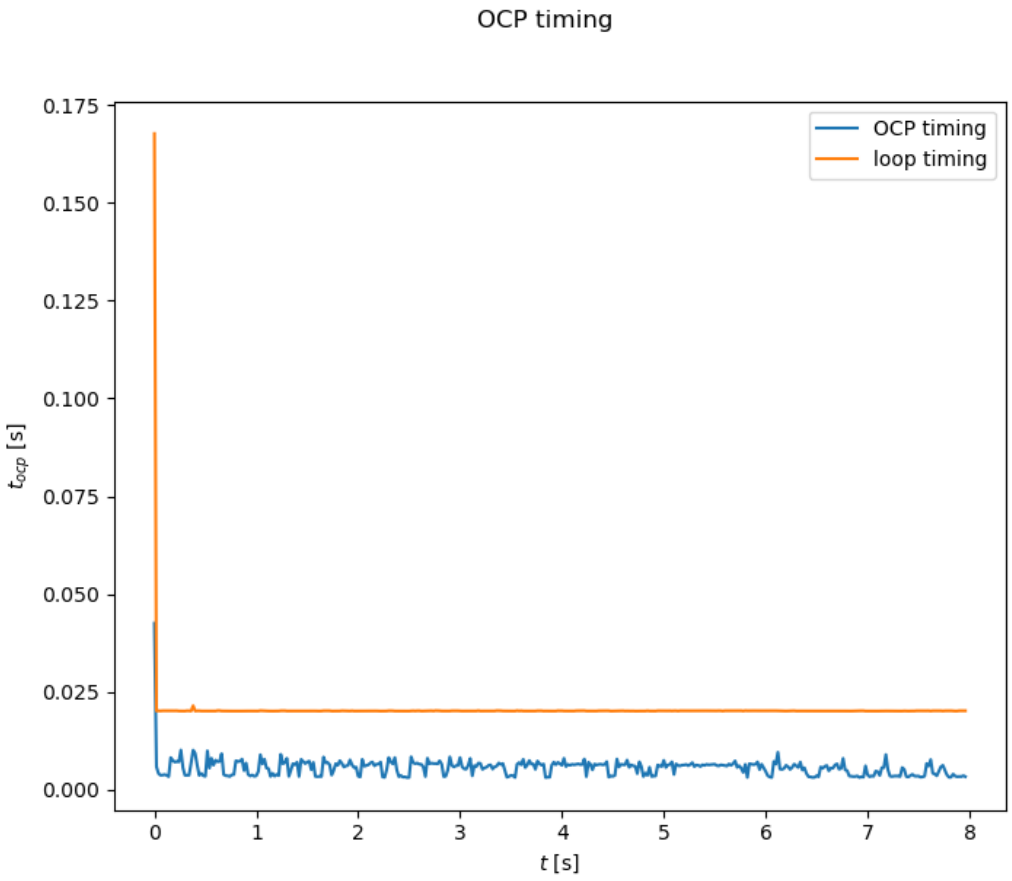


Figure 4.1: Timing graph: Simulation

A necessary condition for proper functioning of the real system is that the measurement of states and the calculation, application of the input must happen within the sampling time of the controller, which is **20** ms in our case.

Figure 4.1 depicts the time taken to solve the OCP at every iteration of the main loop and it is solved in less than 10 milliseconds at almost every iteration, which leaves enough time to communicate with the pendulum.

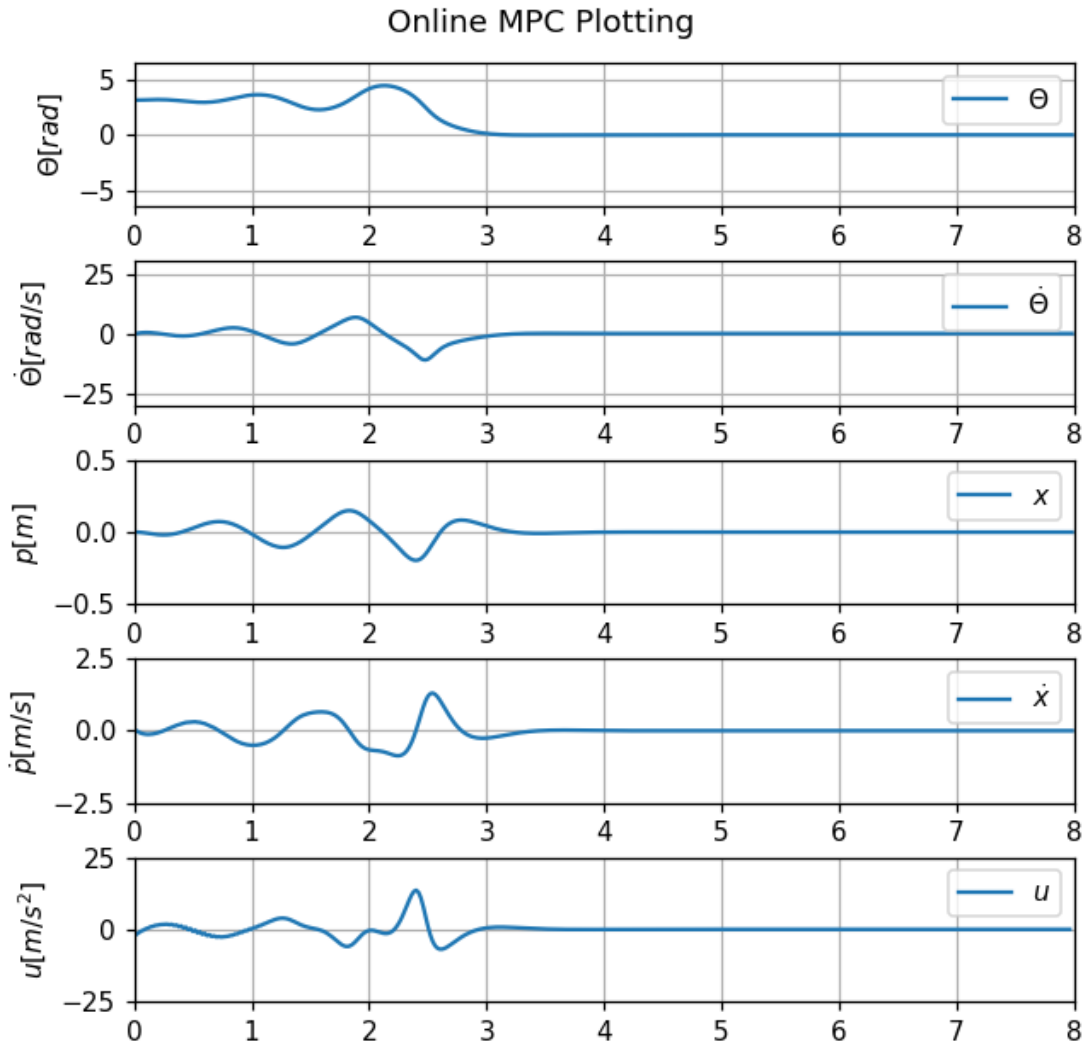


Figure 4.2: Simulation

Figure 4.2 depicts the states during the MPC algorithm which converges to the origin as is expected.

4.2 Real system results

Upon running the algorithm on the real system, the following results were obtained:

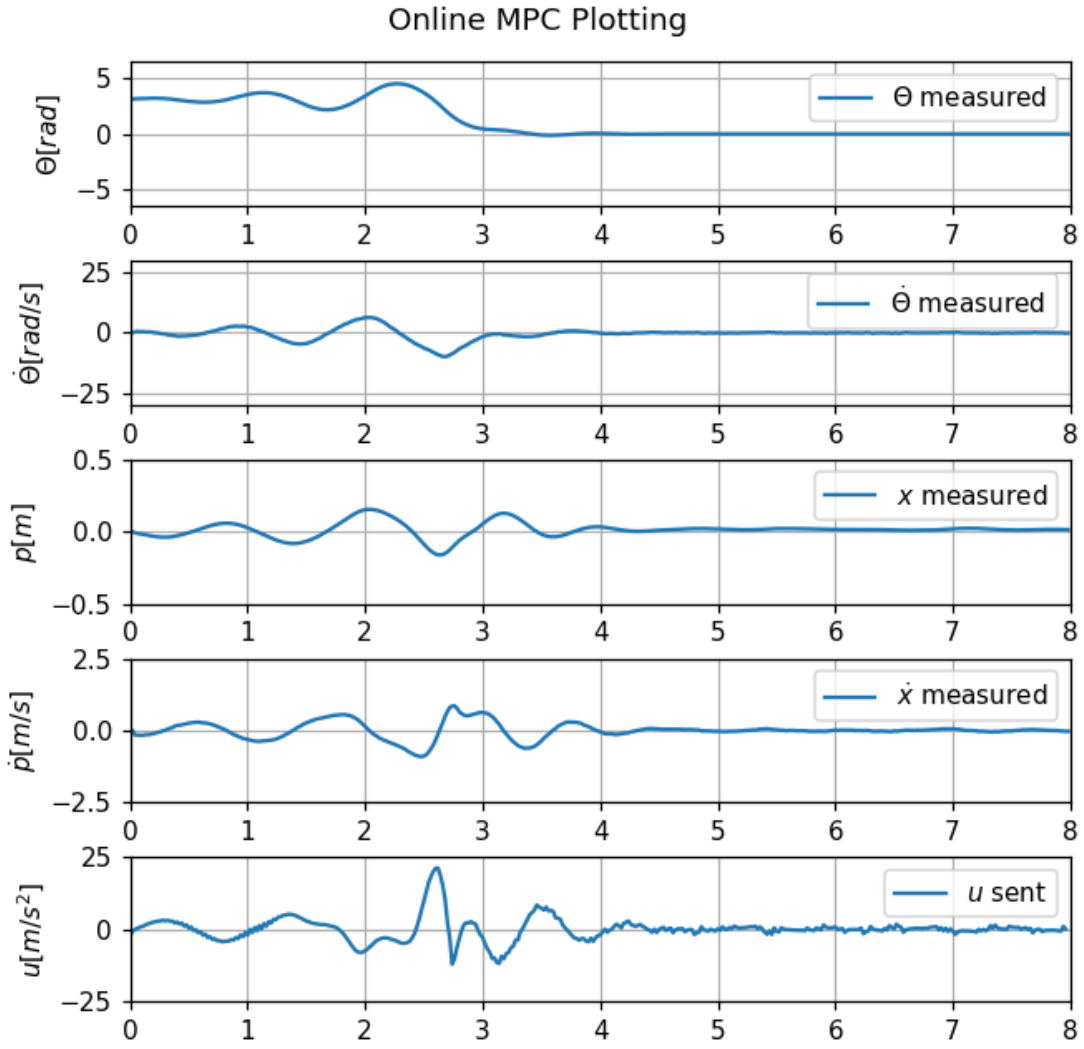


Figure 4.3: Real Pendulum

As seen here, the states closely converge to the origin and therefore the pendulum stabilizes in an upright position within 3.5-4 seconds. There are also slight disturbances on the input which is expected due to real world factors such as different friction coefficients and drag.

5 Conclusion

In this chapter we will go over the conclusions drawn from this project and discuss the scope of future projects on this system.

The complete control system consists of a host PC which is used for computation purposes and an inverted pendulum, produced by Optimal Control Labs, s.r.o. The main aim of this project is to stabilize the pendulum in the upright position at the centre of the rail. This pendulum uses it's in built sensors to measure and/or calculate the states, and is able to provide near perfect state feedback.

Considering the initial requirements, the MPC algorithm does indeed stabilize the pendulum in an upright position. The system responds well to slight disturbances of the pendulum at the unstable equilibrium. Therefore, the requirements of the project have been met successfully.

5.1 Outlook

The findings suggest that MPC is a promising approach for controlling inverted pendulum systems and other complex and unstable systems. We believe that further work should focus on the following aspects to further enhance the application of MPC in this domain:

- **Observer based MPC**: In this project, we were reliant on the system for state feedback, which though mostly accurate, could be improved. State observers like the Kalman filter which reduces the impact of measurement and process noise on the state estimates could be implemented for smoother and more accurate control.

- **Advanced MPC Algorithms**: Investigating and developing more advanced MPC algorithms can help in improving control performance. Strategies like multi-stage or tube based MPC are promising avenues to look into in this regard.

- **Embedded MPC**: As mentioned in Chapter 3 the optimization was performed on the host PC due to the computational complexity of the MPC algorithm. But there is a way to circumvent this. Embedded MPC involves the condensing of the QP and reducing the memory requirement, enough so to perform it on embedded hardware.

- **Machine Learning Integration**: Like mentioned before, exploring more complex control algorithms is an encouraging path in moving forward and the next logical step is the integration of machine learning techniques into these control strategies.These methods, like explicit MPC, can lead to less computation heavy control algorithms, which may decrease control complexity and greatly improve overall performance.

In conclusion, MPC offers a powerful and robust control strategy for stabilizing an inverted pendulum system. By further refining the control algorithms, optimizing implementation, and exploring new applications, MPC can continue to play a vital role in addressing challenging control problems in various fields.

Bibliography

[1] E. Susanto, B. Rahmat, and M. Ishitobi, "Stabilization of rotary inverted pendulum using proportional derivative and fuzzy controls," in *2022 9th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*, 2022.

[2] P. Pinyopawasutthi, D. Banjerdpongchai, and Y. Oishi, "Design of output feedback nonlinear model predictive control for inverted pendulum on cart," in *2018 18th International Conference on Control, Automation and Systems (ICCAS)*, 2018.

[3] O. C. Labs, *User manual for linear inverted pendulum*, Optimal Control Labs, 2022.

[4] *Model predictive control 101*, *Medium*, `https://pallavrawat.medium.com/model-predictive-control-101-28de17208c39`, 2023.

[5] X. Fang and W.-H. Chen, "Model predictive control with preview: Recursive feasibility and stability," *IEEE Control Systems Letters*, vol. 6, pp. 2647–2652, 2022.

[6] F. Allgower, R. Findeisen, and Z. K. e. a. Nagy, "Nonlinear model predictive control: From theory to application," *Journal-Chinese Institute Of Chemical Engineers*, vol. 35, no. 3, pp. 299–316, 2004.

[7] J. Rawlings, D. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation and Design*. Nob Hill Publishing, 2017.

[8] U. of Freiburg SYSCOP, *Acados documentation*, `https://docs.acados.org/about_acados/index.html`.

[9] J. Nocedal and S. J. Wright, *Numerical Optimization*. Springer, 2006.

[10] E. Systems, *Esp32 technical reference manual*, Expressif Systems, 2022.