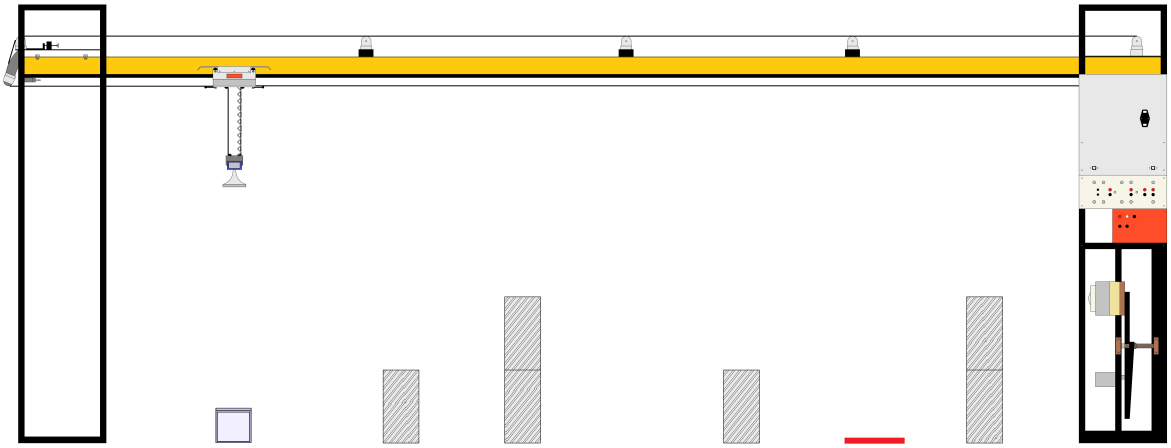
Gantry Crane Path Planning and Control



Project Thesis

Aalborg University
Electronics and IT

AALBORG UNIVERSITY

STUDENT REPORT

Title:
Gantry Crane Path Planning and Control

Theme:
Robotics 10

Project Period:
Fall 22-Spring 23

Project Group:
Group 950

Participant(s):
Malthe Lang Mignon
Alberto Martin Monroy Bjørn
Jonathan Midtgaard Jensen

Supervisor(s):
Jan Dimon Bendtsen

Page Numbers: 122

Date of Completion:
September 29, 2023

Abstract:

The report studies the design and implementation of a control system for a gantry crane paired with a path planner for the automation of gantry crane related tasks.
A research of different state-of-the-art approaches to gantry crane automation, as well as gantry crane control and various other solutions related to the issue at hand has been included in this project.
This paper also contains an in-depth explanation of the design and components of a 2D gantry crane, as well as the mathematical model of said crane. After the structure of the available crane was presented, a final solution, designed for path planing and control of a 2D overhead crane, is proposed, implementing a Model Predictive Controller that utilises a Potential Field-based path planner.
The solution was then tested based on a number of requirements which defined the capabilities that the solution should include, and a conclusion along with a number of considerations for further improvements have been made.

Contents

Preface

Reading manual
The report is written in a one-page format, meaning it is intended to be read digitally. Citations placed on the left of punctuation marks, address the specific sentence located to the left for the citation. If a citation is placed on the right side of a punctuation mark, the citation addresses the whole paragraph.

Documentation

Malthe Lang Mignon
<mmigno17@student.aau.dk>

Alberto Martin Monroy Bjørn
<amonro18@student.aau.dk>

Jonathan Midtgaard Jensen
<jonjen16@student.aau.dk>

Chapter 1

Introduction

Gantry cranes are considered to be an essential part of various industries, including manufacturing, construction and shipping. The latter being the focus of this paper.

The freight industry has its largest network by sea, which conclusively means that most of the wares bought and sold are transported by ships, approximately 90% according to journalist Rose George [1]. Sea transportation is therefore the biggest contributor to the world's trade of goods. This is where gantry cranes come into the picture. According to the United Nations Conference on Trade and Development (UNCTAD), the container port throughput was measured to be of approximately 800 million TEU[1] in 2020 [2]. To be able to maintain the shipment of the 800 million containers a year, ports need to be equipped with various gantry cranes, capable of loading and unloading them.

In general, these cranes require of a human operator to carry out the different tasks it has to perform. During the last decade, a number of automations have been implemented on gantry cranes, for example in the planification of the loading and unloading of containers as proposed by Martínez et al. [3]. Yang et al. [4] mentions the existence of the Xiamen Ocean Gate automated container terminal, in China. The port contains three main sections, the unloading of the ships section, the container transportation section and the container placing section, which is where gantry cranes are utilised. The cranes pick up the containers from the Automatic Guided Vehicles[2] and place them in the correct position within the container yard, an area of land within the port's ground where containers are stored for further transportation.
This example confirms the existence of automated gantry cranes, for example in the mentioned Xiamen Port, where some ZPMC gantry cranes are in use. These companies tend

[1]Twenty-foot Equivalent Unit: *represents the volume of a standard 20 feet long intermodal container used for loading, unloading, repositioning and transshipment*[2].

[2]These vehicles transport the containers unloaded from the ships to the container yard, where containers are organised and placed for further use

to avoid releasing the algorithms utilised for performing path planning and control of their cranes, which has reduced the ability of the group to obtain relevant information on the subject. Nevertheless, the research carried out in this paper has shown that overhead cranes have been used in various fields and some conclusions could be derived.

Some of the conclusions reached from the research show that reducing the cost and increasing the efficiency of the loading and unloading tasks have been of interest to various companies related to the shipping industry. One of these, Fuji Electric Co., Ltd. proposed the automation of gantry crane as a research topic. Within this topic, three main concerns were found, namely, sway angle under unfavorable conditions, path planning optimisation and the final landing problem. These concerns have in common the need for a control system that can manage them all.

In this paper, following the proposal presented by Fuji Electric Co., a controller is presented and the performance of two path planning algorithms have been compared. The paper is organised as follows: **Chapter 2** contains work related to the path planning problem and different algorithms in use nowadays, as well as some control solutions which could, potentially, be utilised on gantry cranes. **Chapter 3** describes the equations of motions of an existing gantry crane (of small dimensions), from which a 3D model is derived. Furthermore, the intricacies of setting up the physical system is also explained in this chapter. **Chapter 4** states the requirements of specifications for the solution presented in this paper. It also describes how the different path planning algorithms are compared, as well as the conditions that the controller must follow. Next, **Chapter 5** defines the implementation of the algorithms researched, on the physical crane. In **Chapter 6**, the tests performed on the solutions and the results obtained from them, are listed. The following two chapters, **Chapter 7** and **Chapter 8**, conclude the paper.

Chapter 2

Technical Analysis

In this chapter the various matters which can be related to the problem at hand will be analysed. Firstly, some related work will be introduced. Then, some more specific areas of study will be analysed.

2.1 Overhead Cranes

Gantry cranes, also referred to as overhead cranes, in the shipping industries are utilised extensively in the in-port transportation of containers and the placing of these on the correct positions. Two types of cranes can be seen in ports, quay cranes and yard cranes. Quay cranes are those that have the task of loading and unloading containers from the ships, and are placed on the quay[1]. The second crane mentioned, yard cranes, refer to the gantry cranes situated on the container yards[2]. The focus of this project is on these yard cranes, as the proposal received and the available crane concurs with the physical aspects of these -cranes and their utilisation.

In **Figure 2.1**, the unloading of a container, as well as its placing on the yard, can be seen. The grid shown under the crane represents the container yard, where each individual container placement is a single rectangle. The figure does not show the quay crane which would be responsible for performing tasks A to B.

[1]Platform close to the water that ease the accessibility to the ships
[2]Area within a port where the containers are stored before their transportation

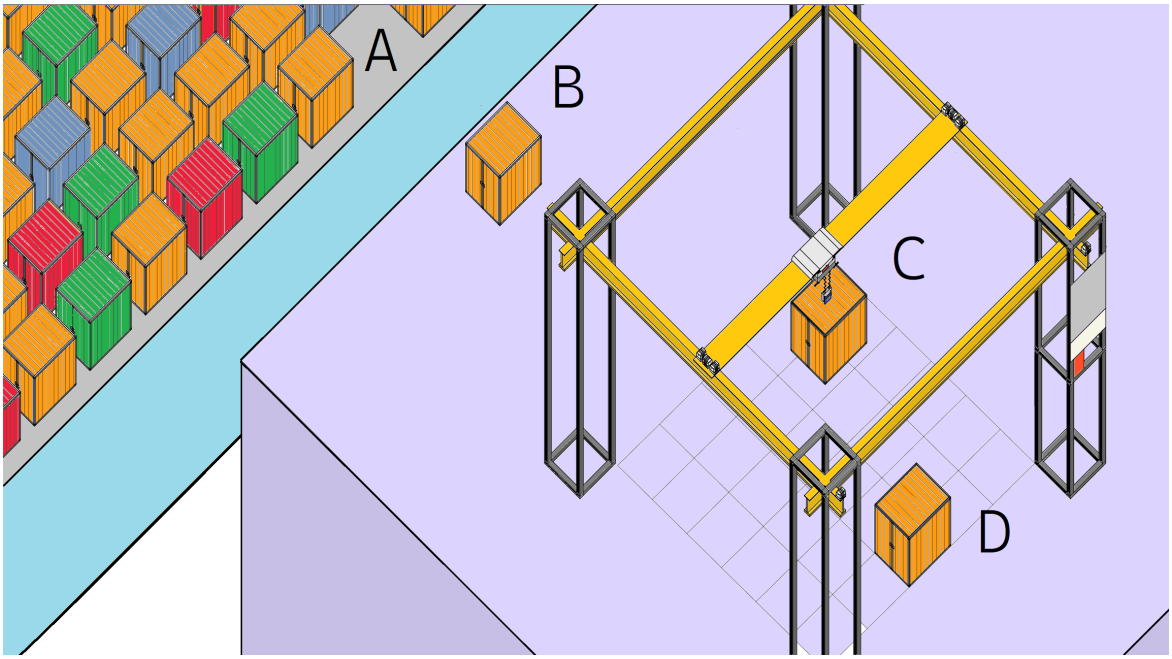


Figure 2.1: Figure depicting an example of the tasks performed when unloading a shipping container from a ship. Task A: the container is lifted from the ship by the quay crane and moved towards land. Task B: the container is placed on the ground, close to yard crane. Task C: the gantry crane picks the container and moves it towards the container's specified position. Task D: the crane places the container on the floor or, on top of another container.

Another element that sometimes appears in these tasks is the use of a motorised platform that serves as a connection between the two types of cranes mentioned.

The implementation of this project was done on a down-scaled gantry crane located at Aalborg University's Control Lab at Fredrik Bajers Vej 07C 2-104. The crane is referred to as a 2D crane, as it only allows the movement of a payload in two dimensions, namely width and height. It has to be mentioned that, although the payload can swing in a three dimensional space due to disturbances on the wires, the crane is still considered to only act on two dimensions.

In the following image, **Figure 2.2**, a schematic drawing of the crane can be seen. The parts that form said crane are also listed and will be further expanded in following paragraphs.

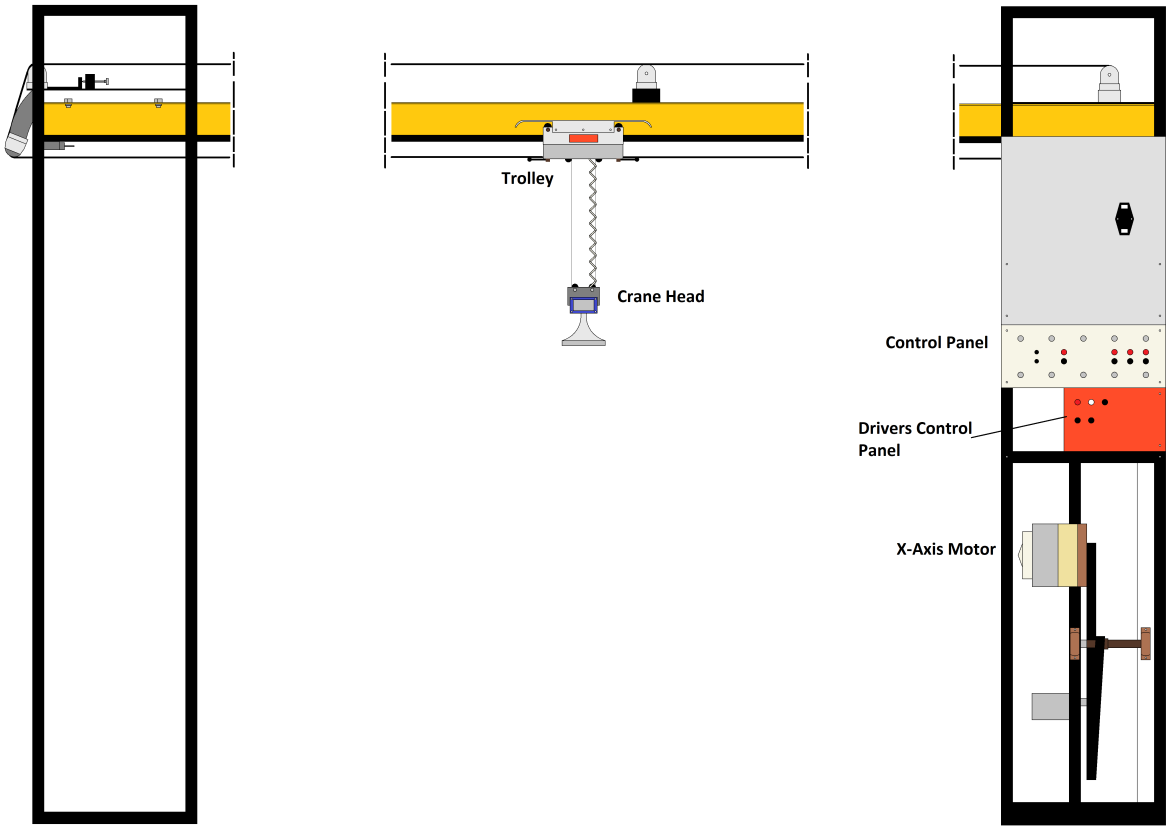


Figure 2.2: Depiction of the crane used for the implementation and testing of the solution presented in this paper. Due to the size of the original image some parts, considered unimportant, have been removed.

2.1.1 Crane Overview

In this section, an overview of the crane is given based on the drawing shown in **Figure 2.2**.

The head of the crane, see **Figure 2.3**, is one of the only mobile elements of the crane. It moves lengthwise and crosswise to pick and place objects within its workplace.

The head consists of an Arduino Nano MPU-6050, a gyroscope, an accelerometer, and an electromagnet. The Arduino Nano is implemented with a script that allows the user to send and receive information from the sensors. It also enables the use of the magnet, for attaching and dropping the objects. The connection is done using the cable seen on the top-right corner of the image.

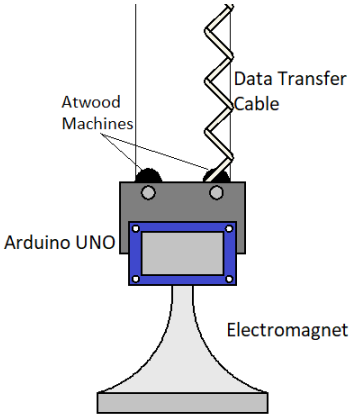


Figure 2.3: Head of the crane.

The trolley, see **Figure 2.4**, is pulled by the motors through some steel cables and has the head hanging from itself. The trolley does not contain any electronic device and its only purpose is to hold the head, moving in the x-axis in both positive and negative direction.

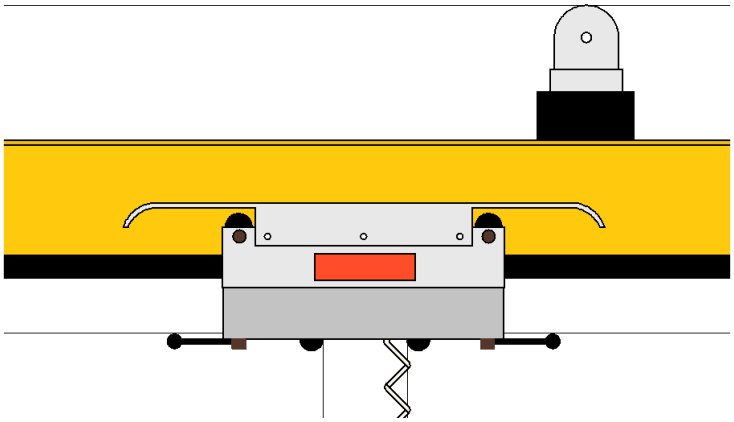


Figure 2.4: The trolley, which connects the head to the structure of the crane.

In the figure to the right, **Figure 2.5**, the motor and gears that actuate the mentioned wires can be seen. Behind the visible parts, shown in the image, is the other motor which manages the up- and downward movement of the head. The motors are Axem Disc DC Servo Motor type F9 M2.

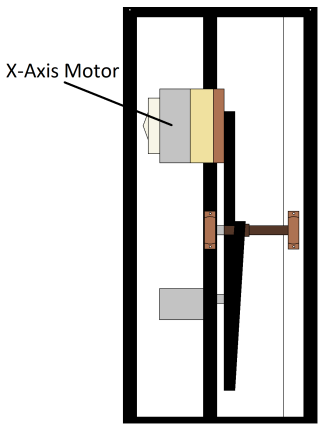


Figure 2.5: Section of the crane where both motors and their respective gears are located.

These motors are managed by the Motor Drivers shown in **Figure 2.6**. There is one driver per motor, both placed behind the metal plate. These drivers had to be reprogrammed to fit the project. This is briefly explained in **Section 3.2.1**.

In the same figure, two more panels can be seen. The red panel contains buttons and plugs that allows the user to control the drivers physically, by allowing current to pass through or preventing the drivers from receiving any, these are the two black switches at the bottom of the panel. On top pf these two, three connectors are visible. The red allows for an external positive supply, the white one is for connecting an additional sensor and lastly, the black connector is for ground.

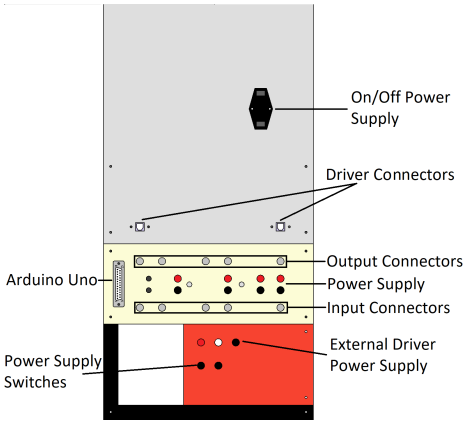


Figure 2.6: Section of the crane where the motor drivers and various connectors are located.

The other panel, between the red and the metal panels, contains multiple connectors and switches to control and measure signals that from/to the crane.

The following table contains the accuracy specifications of the sensors used on the crane.

Sensor	Accuracy
Potentiometer	0.02%
Tachometer	1.5 %
IMU	0.05 °/s RMS

Table 2.1: Accuracy of sensors.

2.2 Related Work

The research on gantry cranes spans over a wide variety of areas of use, though, in general, the research tends to focus on individual subsystems of what can be defined as the overall Gantry Crane System. One example of a general solution for an automated crane is presented by Sawodny et al. [5], where they introduce the idea of a gantry crane as a large robotic workspace. Apart from the concept of the crane being a robot, the control system is approached as a decentralised linear control, i.e., the control is divided between some subsystems, each of which will generally have a means of communicating with the others.
As mentioned, researchers emphasise their work on less generalised solutions than the one by Sawodny et al. Thus, the obtained papers have been divided into those focused on Control Systems and those focused on Path/Trajectory Planning.

2.2.1 Path/Trajectory Planning

Path Planning is a crucial system needed to achieve crane automation, as the main task of said cranes is to transport a payload from one point to another. Thus, a path planner that is capable of delivering a fast, safe path is necessary. Different approaches to path planning on overhead cranes have been researched.

Non-deterministic Algorithms

A number of papers prefer the use of more flexible approaches to path planning and utilise non-deterministic algorithms. These can derive multiple valid solutions for a problem, even when the initial conditions are the same. One of the advantages of these are, as mentioned, the flexibility of obtaining some random path to the goal. This can be considered positive as the problem itself allows for multiple solutions.

One recurring non-deterministic approach is the Rapidly-exploring Random Tree (RTT). This algorithm efficiently searches the specified area by randomly generating points in space and connecting the latest node to each non-obstacle point. One example of this being utilised on a crane is the paper from Zhimei et al. [6], which combines the RRT algorithm with another non-deterministic method, namely Particle Swarm Optimisation (PSO). A "tree" is made using the RRT and then, the PSO algorithm smooths the path allowing the crane to perform it.

Another implementation of the RRT algorithm is in Du et al. [7], where the improved RRT-Connect is utilised. RRT-Connect constructs two RRTs, one at the start node and one at the goal, that grow towards each other through some greedy heuristic.

One additional paper that implements a non-deterministic solution to crane path planning is Waikoonvet et al. [8]. The algorithm in use is the Ant Colony Optimisation (ACO) for planning a collision free path for the crane, modified by using the cubic spline formula to generate a trajectory.

Deterministic Algorithms

Contrary to their counterpart, deterministic algorithms will always output the same solution for the same initial conditions, meaning no randomness is involved in deriving a solution.

Vu et al. [9] present a motion planning solution for a laboratory gantry crane. The planning is divided into offline pre-computation of trajectories and an online trajectory planning. The offline part of the solution computes the possible trajectories of the crane between two points in the workspace, where obstacles are present. Both points are considered to be within known sub-spaces, described as grids. Then, an optimal trajectory is planned offline using the direct transcription method. The online planning is performed if the start and end states are changing while the gantry crane is performing a task. A seven milliseconds calculation-time for a single trajectory is obtained using the database built with the offline trajectories computed. When the start and/or end state of the trajectory changes, the online planner selects the trajectory closest to the solution, from the mentioned database, and reduces the deviation from the closest trajectory to the new better-planned trajectory.

Other researchers attempt to reduce the swing produced by the crane's trolley on the payload. This issue affects not only the physical component of the crane, i.e., the payload knocking on the surrounding elements of the workspace, but it can also have an effect on energy consumption. Zhou et al. [10] presented a solution for an optimal path planner for a gantry crane, utilising five trajectories from some other models for comparison. All the five trajectories are considered by Zhou et al. to be un-optimised. To solve this, they present three optimal control strategies, a swing-optimal model, an energy-optimal model and lastly, a time-optimal model. The first model tries to optimise the crane for the swing of its payload while moving horizontally. The energy-optimal model minimises the energy consumption on each trajectory. The last model searches for the minimal transportation time of moving a payload from an initial to a final state. Another proposal which focuses on minimising time is Zhang et al. [11], which implements the quasi-convex optimisation method to find the solution which takes the least time while considering the constraints of the angle of the payload, the restricted velocity and acceleration, and the jerk the trolley suffers while moving.

The idea of a trajectory where the coupling behaviour between trolley and payload in a crane is taken into consideration has been proposed by Ning et al. [12]. The approach presented is based on the theory that two interacting objects are considered as coupled.

This helps at planning a trajectory that minimises payload swing. The specific model followed for the generation of the path is the S-curve motion profile. This motion is a smoothed-out variant of the trapezoidal profile, which contains sharp turns. By using this type of motion profile together with the coupling theory, the unwanted sway angle of the payload gets dampened down. The S-curve profile is also used by Fang et al. [13], with the main difference that empirical data is used to bound the trajectory within desired constraints. The constraints determined in this paper are the limitation of the velocity at which the trolley moves, for which a maximum velocity was set. The acceleration is also constrained similarly to the velocity by defining a maximum a acceleration that ensures the payload reaches its goal without any mishaps. Lastly, the trolley must move towards the goal coordinate, both for time efficiency and for energy efficiency.

Path Planning for mobile cranes

Algorithms for searching paths have also been applied to self-driven gantry cranes, which as their name implies, are capable of moving around without human intervention. Some examples for this are the research performed by Kaneshige et al. [14] and [15].
Since focus of this paper is on the craning task of the gantry crane, research on mobile cranes will not be necessary.

Summary

Different solutions regarding path planning for gantry cranes have been researched. Three groups have been found, the deterministic and non-deterministic algorithms, and path planning for mobile cranes. This last group has been considered as unimportant for this project and thus, will not be explored further. Non-deterministic algorithms appear to allow for more flexibility towards path planning than the deterministic ones, this is due to the first one allowing for random correct solutions for a problem. Some of the mentioned non-deterministic algorithms are RTT, PSO or ACO. In the case of deterministic algorithms, these focus on deriving solutions capable of responding to certain inputs to derive an specific output, where there is no randomness involved.
As seen, various approaches can be implemented in a gantry crane to solve the path planning problem. Nevertheless, due to the research not highlighting any specific solution, the research on path planning algorithms is expanded further in **Section 2.3**.

2.2.2 Control Systems

Robotic solutions always include a control system that organises and commands the solution. Gantry cranes also need this characteristic, especially so for automated gantry cranes, as no human will have direct control of the crane. Some crane control methods have been researched in this section.

The paper from Fang et al. [13], mentioned previously, does also implement a control method that utilises S-curve profile for the motion planner. A control method is designed based on the energy analysis of the overhead crane to derive an adaptive tracking controller. This controller keeps the trolley within the set limits of velocity and acceleration while reducing the swing of the payload. According to the results shown, the control method ensures an asymptotic tracking result, i.e. the disturbances become insignificant compared to the tracking of either velocity or acceleration.

One common strategy in the control of gantry cranes is to minimise the sway angle of the payload, this can be seen in [16] and [17]. The first article utilises a combination of two PI controllers to control the trolley with the payload attached. For the output of this controller to be considered optimal, the controller parameters need to be adjusted. This was done by utilising PSO to obtain the optimum four parameters. In article [17], an improved PID controller that minimises payload swinging is proposed. A PID controller is designed using an approximate model of the system and Fuzzy logic is utilised for tuning the parameters of said PID. According to the authors, this is needed due to PID controllers having difficulties performing with varying parameters. Sorensen et al. [18] also utilises a PID-based controller for solving payload sway. Their approach differs in that the solution is derived based on three sub-modules, an oscillation cancelling sub-module, a sub-module for precisely measuring payload position and a disturbance rejection sub-module.

One more article that combines PSO, as a parameter tuner, and a PID controller is [19]. The controller consists of a PID and a PD controller, managing the position of the trolley and minimising the sway of the payload. As with the approach presented by Suvorov et al.[16], PSO is utilised to optimise the parameters of both controllers. The PSO used is improved by introducing a priority condition where steady-state is considered the highest priority, then overshoot, followed by settling time. This means that the optimal solution will be decided by the particles whose fitness function fit best the desired outcomes in the respective order.

Summary

Some gantry crane controllers have been listed. Different approaches have been discovered, some utilise PID controllers while on some other cases, the authors design their own control system, for example Fang's et al. solution, which bases its controller on energy analysis.
One focus which several authors seem to emphasise is the sway angle. This has to be taken into account when designing a control system for a gantry crane. Some controllers have been listed but it has been considered that they may not cover all the possibilities that could be implemented into the final solution of this paper. Therefore, a more in-depth analysis of control systems can be seen in **Section 2.4**.

2.3 Path Planning Algorithms: A Review

In **Section 2.2**, path planning approaches for gantry cranes have been discussed. Different algorithms have been used with distinct solutions for each of them. The research did not show any clear preference for any specific algorithm, but it depicted two clear approaches. Those where deterministic algorithms were utilised and those where the preference was on non-deterministic path planning algorithms. At the same time, a third group was discovered, that is, Evolutionary Algorithms. The algorithms forming this group may be assigned to one of the other two types of algorithms, but due to the nature of evolutionary algorithms, it has been decided to create a separate subsection to describe them. Following this discovery, a review of various path planning algorithms has been made. These do not necessarily relate directly to gantry cranes, but have been chosen as there exists a possibility that they can be implemented on cranes. A comparison on some algorithms from the different categories could give an insight into which type of path planning algorithms are more optimal than the others for a gantry crane.

2.3.1 Non-deterministic algorithms:

Non-deterministic algorithms refer to some process containing a certain degree of randomness that outputs variable solutions. Different algorithms have been found and categorised, those are PSO, Simulated Annealing and, ACO.

Particle Swarm Optimization:

Proposed by Kennedy et al.[20], is the optimisation of a problem by improving a solution over many iterations, based on some measure of quality. A number of possible solutions, "particles", populate a search-space. Using a formula to stir them around, the particles will position themselves in space, according to some criteria that defines both, each particle's local best known positions and the space's best known positions. For each iteration, the positions get updated and the particles will get attracted to these knew best known positions.

In [21], different variants and applications of PSOs are listed. One type of PSO that he refers to are those that combine it with one or more algorithms, such as the hybrid algorithm proposed by Girija et al.[22]. Here, they combine PSO with Artificial Potential Field (APF). APF is used to generate a potential field, this field repels the particles of the PSO from whatever obstacles may be in the environment. This increases the speed at which the PSO can find the optimal solution for the path planning. Another improved version is the PSO-Gray Wolf Optimisation (GWO) proposed by Cheng et al.[23]. The GWO emulates the behaviour of wolfs in their hunting environment, where a leadership hierarchy is followed by the wolfs. GWO increases the speed at which the PSO can derive a solution. It does that by applying a hierarchy of the fittest solution to the particles, before PSO is applied. GWO orders the particles in terms of the optimal fitness value, inciting the other

particles to move to the best known positions.

PSO algorithms have one main advantage, and that is its low number of tuning parameters. On the contrary, some of the most important disadvantages are premature convergence, poor global search capability, and slow convergence speed. Premature convergence happens due to the decrease of particle velocity, which forces the particles to local minima, not converging on the desired goal [24]. This disadvantage derives into the one mentioned next, poor global search capabilities. Since the particles get lost in local minima, the search for the optimal solution is less effective or, may never reached. It could also be mentioned that, when combined with other algorithms, such as APF of GWO, the PSO algorithm improves in its performance.

Simulated Annealing:

Annealing refers to the process of heating a metal or alloy close to its melting point and then, allowing it to cool down. During the slow cooling, the atomic movements of the excited atoms get reduced until a lowest-energy state is reached [25]. This process is utilised to reduce the hardness of the material, to make it more malleable by heating it up. When it cools, the material recovers its stiffness. Simulated Annealing works by following this principle.
The algorithm works by pre-defining a reasonable solution and an initial temperature for the problem at hand, as well as an end condition that stops the annealing. A temperature reduction function is defined based on how the temperature should decrease. At each time step, the temperature will drop and a new solution will be obtained. The cost between the previous and the new solution is calculated. If the cost is greater than zero, then the new solution is better, so it gets accepted. If not, the cost is lower than zero and the previous solution is considered superior. Before accepting this solution, a random number between zero and one is selected and if this number is under some value, it gets accepted, else, the new solution gets selected.

Miao et al.[26] present an enhanced simulated annealing algorithm for path planning. The enhancement made to the algorithm is the initial determination of a path, which reduces the computation resources needed to run the algorithm. The dynamic environment considers static and moving obstacles which the robot should avoid. The initial path is derived offline, and only considers static obstacles. When the robot initiates its motion, it follows the predefined path. If a moving obstacle gets detected by its sensors, the robot calculates its direction and determines whether or not it will collide with it, calculating a new path if needed. The online calculation of the new path is evaluated using Simulated Annealing.
Multiple uses of Simulated Annealing come together with some other algorithm which have the constrain of getting stuck in local minima. This is because Simulated Annealing is capable of making the system leave this minima. One example of this is [27], where

Zhu et al. utilise Simulated Annealing together with APF, briefly explained in the PSO description. A path planning approach is proposed in [27], using APF to traverse the environment and reach the end goal. As mentioned, potential fields have the drawback of falling into local minimum, not being capable of leaving them. By combining it with PSO, this minima do not pose a threat and the robot should be capable of reaching the desired solution.

Some advantages of simulated annealing are, the easy coding and use, its robustness against noisy data as well as the ability to treat nonlinear models. It also benefits from its ability to escape local minima and reach the global optimum solution. The cooling aspect of this algorithm makes it slow at deriving a solution. This trade-off can be obviated if the time aspect is unimportant for the application. Even when good against various local minima, if the environment is smooth and the number of local minima can be neglected, it could be considered excessive the implementation of simulated annealing, considering the slowness of the algorithm.

Ant Colony Optimization:

Method proposed by Dorigo et al.[28], inspired on the pheromone-based communication that ants use. The algorithm works by sending a number ants into a search-space from an initial position. The ants search this space leaving a trail of pheromones looking for the goal. When an ant reaches the goal, it follows its own trail back "home", updating the pheromones left on the path with information. This information varies depending on the application of ACO, in terms of path planning, said information could describe the length of the path or the total time needed to cover the path. This process of sending ants and retrieving path information is iterated several time. The data for the paths found will be used as a baseline for the new ants, which, due to a random component, will sometimes stray from from the path, generating a new one. If the newest path is considered as better, depending on the requirements defined, it will overwrite the previous path.

In 2018, Akka et al.[29] proposed an improved ACO for path planning pf robotic systems. Akka introduces an stimulant to the algorithm. This stimulus supports the ants in selecting the next step on the grid, directing them to the desired solution. By doing this, the convergence increases, enhancing the speed at which the algorithm finds a solution. Yee et al.[30] utilise ACO to obtain the fastest path between the start and the goal nodes. The solution presented in the paper is a direct implementation of the ACO algorithm. They tested the algorithm against different maps with varying difficulties, which were obtained from Sedighi et al.[31] where a genetic algorithm was used for path planning. The results were compared, and according to Yee et al., ACO derives faster paths than the genetic algorithm developed by Sedighi et al.

Ant Colony Optimisation has its main disadvantage in the convergence speed and accu-

racy with large datasets. Nevertheless, it is advantageous in that it can adapt to dynamic environments, i.e. it can adjust itself to changes in real time.

2.3.2 Evolutionary Algorithms

Evolutionary Algorithms are search strategies based on the Darwin's theory of evolution, encompassing natural selection and genetics. These try to derive a solution which may not be the most optimal, but are still considered of good quality. According to Petrwoski et al.[32], these are generally implemented on problems whose solutions are considered unrealistic by "exact methods". An Evolutionary Algorithm uses mechanisms inspired by biological evolution, such as reproduction, mutation, recombination, or selection. Depending on the mechanisms, evolutionary algorithms can be divided into different groups. The types of Evolutionary Algorithms are [33]:

- **Genetic Algorithms:** derive a solution based on recombination and mutation[3] (optimization problems). These algorithms have a group of possible outcomes, depending on the problem at hand. These problem are defined using an objective function on which the different outcomes are fitted to. The confidence of a solution being good is contingent on how well the outcome fits the mentioned objective function. This process evaluates each individual of the possible outcomes group, from which the better valued solutions are passed onto a new generation of the group. And the process is repeated, deriving only the "best" individuals/solutions to a problem. The solution is outputted as a binary string of fixed length.
  An example of its implementation as a path planner can be seen in the paper mentioned previously, from Sedighi et al. [31]. Utilising the Genetic Algorithm instruments mentioned, and following the requirements set by the problem, in this case the path length and the number of turns. The generations will continuously mutate to fit the objective function, until reaching the optimal path.

- **Genetic Programming:** is similar to Genetic Algorithms but introduces genetic operators. One main difference between the two is that Genetic Programming can be expressed as a tree of functions, whereas Genetic Algorithms work in a linear matter, resulting in a binary string. The use of a tree-like structure allows for exchange of information between the branches. Genetic Algorithms are generally used to generate optimised programs of variable length, as its tree-like structure can be seen as multiple functions correlated to each other and that have been optimised. The main evolutionary operators used are mutation and crossover[4].
  Kala et al. [34] utilise Genetic Programming to optimise the path planing of multiple robots that have to collaborate with each other. The author divides the system into

[3]Values of particular genes/solutions are changed randomly.
[4]Parts of an outcome are exchanged between two selected outcomes.

two, master and slave, whereas the slave system is formed by the paths of all individual robots respectively. And the master, is the optimal general path for the swarm. The tree-like structure allows the paths of the individual robots to communicate with each other, permitting the master to compute the optimal path.

- **Evolutionary Programming:** genetic programming outputs optimised programs of variable length, on the contrary, the Evolutionary Programming's structure of the program is fixed, and it is the parameters that are allowed evolve. It mainly uses the mutation operator, compared to genetic programming which also utilises crossover. Combined with an Artificial Bee Colony (ABC) algorithm, Contreras-Cruz et al.[35] utilise Evolutionary Programming to optimise the smoothness and length of the paths. The ABC algorithm generates numerous paths, all of which get evaluated by the evolutionary programming. It generates a population of feasible paths obtained by the ABC, exposes the population to the environment, checks if the members fit the objective function, mutates some members and selects some of them for a new population. Repeating this process until some condition set by the author is met.

- **Evolution Strategy:** uses vectors to represent the outcomes and utilises self-adaptive mutation rates. Evolution Strategies originate from the idea of natural selection, in which individuals with features valuable for survival are more prone to pass these features down to the following population than negative characteristic that hinder the individual's survival.
  Evolution Strategy is used by Zhao et al. [36] to optimise non-linear problems, they specifically use the Covariant Matrix Adaptive Evolution Strategy (CMA-ES). CMA-ES works by adjusting the mutation direction of the members in a population, by continuously updating the covariance matrix until the optimal solution is reached. The algorithm goes as follows, it initialises the parameters and settings; mutates the population; evaluates the population individuals and the preferred individuals are chosen as the parents for the following generation; next, the parameters: the mean, followed by the covariance matrix and the step size, get updated. If the conditions are met, the algorithm will output a solution, if not, the process is repeated for the new population.

- **Differential Evolution:** is mainly used in optimisation problems, specifically in a continuous space. One main difference compared to Genetic Algorithms is the use of directional information of the individuals in a population by assigning a goal and unit vectors. The algorithm optimises the problem with a number of potential solutions which are mutated by a specified parameter and compared to a crossover rate parameter, which describes the rate at which two individuals (solutions) interchange elements with each other. Some benefits of these algorithms, stated in [32], are the low computational complexity and effort, the ease of use, and the efficiency of memory usage.
  Xiaobing et al. [37] developed a constrained differential evolution algorithm for

UAVs. The algorithm randomly generates a population. The individuals are evaluated based on the fitness function and the constraint violations, and are then sorted as feasible and unfeasible. For the next step, some of the individuals are selected based on their rankings, which defines the possibility of each individual to be selected. After the selection, a vector is generated using the mutation and crossover. This vector is utilised to generate "off-springs" that will shape the following evolution. Repeating this process until reaching the optimal solution.

2.3.3 Deterministic Algorithms

Deterministic algorithms refer to those methods which, given some inputs, will always output the same solution and in the case of robotic solutions, the device will always perform the same actions. The research highlighted two types of deterministic algorithms: Fuzzy Logic-based, and Potential Fields, these can be seen in the following subsections.

Fuzzy Logic

Introduced by mathematician L. A. Zadeh [38] and later expanded by the same author in [39], Fuzzy Logic is defined as a logic where the truth can be any variable within the range 0 to 1. This is similar to binary logic where the solution instead of being one of two options, 0 or 1, the solution in fuzzy logic can be any number between the mentioned interval. This increases the possibility of finding solutions depending on the degree of truth or falseness, referred to as "partial truth". Fuzzy logic is utilised in various fields in robotics, of which the focus will be on path planning.

Pandey et al.[40] developed a fuzzy logic-based path planner with static obstacle avoidance. The designed robot had the task of moving in an environment with no previous knowledge about it. The algorithm works by utilising multiple input variables and outputting some variables parallel to the inputs received. Using the relationship between inputs and outputs, a path where objects are avoided is generated. The input variables mentioned are `Obstacle In Front`, `Obstacle Right`, and `Obstacle Left`; whereas the outputs are `Right Wheel Speed` and `Left Wheel Speed`. These variables are part of their Fuzzy Inference System, the three inputs mentioned are evaluated by 27 rules of their own according and the two output variables are obtained. In [41], Wang et al. propose the use of fuzzy logic for solving the local minima problem that affect numerous path planning algorithms in unknown environment. The robot in this paper finds a path with the implemented Path-Searching behaviour. This technique follows the idea of memory grid, which stores obstacles and grid points in a memory. The path is selected within the stored paths, based on the number of obstacles and grid points of each of them. The more grid points there are in a path, the longer it takes for the robot to traverse, thus the less preferred it is. Fuzzy logic is then utilised for the obstacle avoidance, following a similar approach as the one proposed by Pandey et al.[40]. Where the distance to each object is calculated and

divided into one of the three directions the robot can go (left, right front). Then, a set of rules determine the behaviour the robot has, i.e. the speed of the robot, depending on the distance and position of the obstacles.

One of the advantages of Fuzzy Logic is the similarities it can have to human reasoning, which could be considered as the best way of solving a problem. This comes with a trad-off, and that is that recreating human reasoning can be difficult and time consuming, as these reasonings have to be translated from a human point of view to a robotic language. According to Hentout et al. [42], Fuzzy Logic is beneficial when the problem at hand is difficult to analyse with other approaches.

Potential Fields

One approach utilised for path planning and obstacle avoidance is Potential Fields. These are domains in which Laplace's equations are true, for example electrical fields. In robotics, Potential Fields have been utilised in numerous occasions to perform path planning and obstacle avoidance for both ground robots as well as aerial robots. Due to the variety of existing implementations, applying a Potential Field path planner on gantry crane could be feasible.
Generally, Potential Fields consist of two fields, i.e. Attractive Fields and Repulsive Fields. The combination of these two is what makes this algorithm suitable to traverse environments with obstacles. Attractive fields are used to define the goal towards which the robot should got. Whereas Repulsive fields try to keep the robot away from both the boundaries of the workspace and from the obstacles. On each point of the environment, a potential is computed, by summing the mentioned fields together. This will derive point with high potential and point with low potential. Wherein the low potentials are generally considered to define the correct direction of the robot towards the goal, and high potential points drive the robot away from itself, which again would drive the robot towards the lowest potential, the goal.

An example of Potential Fields being utilised for path planning is in the paper by Hwang et al. [43]. The planner is designed to be a two step process, a global planner and a local planner. First, an initial path is generated using Minimum Potential Valleys (MPV) which are the set of saddle points and local minima. It selects the shortest path between start and goal with the less amount of obstacles that could cause a crash. The local planner will then change the reference path so that the robot does not collide with any obstacle as well as impose a cost function that optimises the path in terms of length and time. Wang et al. [44] derived a potential field algorithm based on steady-state heat transfer. It follows the concepts of heat transfer, which they define as a "sink pulling heat in". The main aspect of this approach compared to the previously explained Potential field is, that it utilises a heat conductivity variable, K, to assign heat values instead of potentials per se. One

advantage that steady-state heat transfer has is its harmonic property[5], that eliminates the local minima problem that the potential field algorithm tends to suffer from.

Potential Fields are considered to be computationally efficient in terms of time, as mentioned by [45], which also remarks the ability of these algorithms to allow online parametrisation and configuration. In terms of disadvantages, the main one is its inability to escape local minima on its own. As mentioned in the above paragraph, there exists approaches that can solve this problem, for example the mentioned steady state heat transfer approach [44]. Lin et al. [45] also mentions the difficulties potential fields suffer with the oscillations of the system when passing between obstacles.

Summary

Three groups of path planning algorithms have been researched, namely Deterministic, Evolutionary, and other Non-deterministic Algorithms. Reviewing non-deterministic algorithms has proven that the three main types are PSO, ACO and Simulated Annealing. It also revealed that Ant Colony Optimisation could be a potential solution for the path planning of the gantry crane. That is because it has relatively low memory consumption, and can adjust to a dynamic environment. Simulated Annealing has been discussed to be too slow for this solution and PSO is considered to be implementable only if it is combined with some other algorithm, which could prove to be time and resource consuming.
Within the deterministic algorithms only two have been discussed, both of which have been considered as potential solutions for this project. The advantages of both these algorithms, Fuzzy Logic and Potential Fields, are deemed as positive to the solution.
The last grouping of algorithms mentioned are the Evolutionary Algorithms. These follow the principles of Darwin's theory of evolution, such as mutations or crossovers. The main advantage of these algorithms is the ability to derive an approximate solution to a problem. This is, in many cases, preferable than spending an excessive amount of processing time calculating the optimal solution.

The analysis of the various algorithms described in this chapter has shown various approaches that may be suitable for a gantry crane. Within the ones listed in this section, its has been decided to test and implement two algorithms, Ant Colony Optimisation and Potential Fields. The first one was selected as it appears to be the most suitable from the non-deterministic algorithms researched. A potential field solution seems to fit the project at hand, as its generated paths are similar to how a crane operator handles the pick and place of containers. The evolutionary algorithms have been discarded from possible solution for the path planning in this project as they are considered to not always derive the optimal solution, which is desired for in this project.

[5]Function where its value at any point is equal to the mean of its values along any circle around that point

2.4 Control Systems: A Review

Control systems are solutions implemented into various kinds of systems to manage and regulate how these behave. A brief description of these will be given in the following list, and some of their advantages and disadvantages will be described.

- **PID Control.** This controller combines three fundamental controllers, namely Proportional control (P-control), Integral control (I-control) and Derivative control (D-control). A PID controller is a closed-loop control system that relies on feedback to compensate for the error signal by applying the three control approaches mentioned. P-control amplifies the error signal, as it makes the control signal directly proportional to the error signal. The I-control generates an output which is equal to the integral of the error signal. Similarly, the D-controller outputs a signal that is the derivative of the error signal. These three can be combined and a control signal, which takes into account the three signal characteristics mentioned, can be obtained.

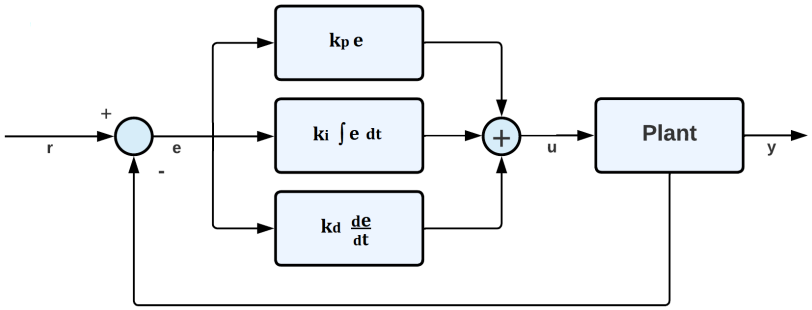


Figure 2.7: Illustration of a PID Controller. u refers to the control signal, r is the reference/input signal and y, the output signal. e is the error signal. From top to bottom, the first block refers to the action performed by the proportional control. The block under is the integral control. And lastly, the derivative control. Combining these three, the control signal u is obtained.

Multiple solutions utilise PID controllers to solve the control problem. This is due to the benefits obtained from each individual controllers, P, I and D. Furthermore, when combining these, some of the disadvantages that these controller have on their own get cancelled by the effect the others have on it. For example, the integral controller can counteract the steady-state error [6] generated by the proportional controller.

- **Fuzzy Control.** This type of control system tries to utilise a 'human-like' way of interpreting data to solve a problem. It resembles Boolean logic, where the solution can only be 1 or 0, in that 0 and 1 are both possible outcomes. The difference with Fuzzy Logic is that outcomes within 0 and 1 are also considered as valid solutions for a problem. One way of describing the functioning of fuzzy logic is using the programming expression 'if ... else ', where `if` an event happens, then something

[6] A steady-state error is the difference between the setpoint and the output when time goes to infinity, meaning the system's output has deviated from its initial output.

happens, consequentially, if another event were to happen (`else`) then something different may be outputted.

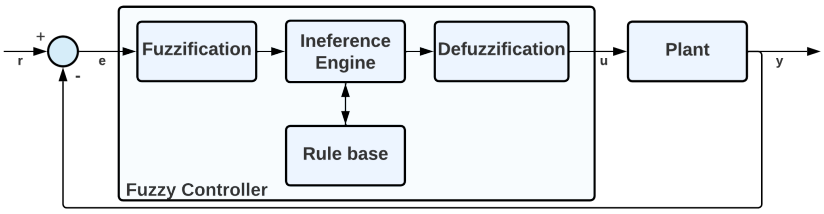


Figure 2.8: Illustration of a Fuzzy Logic Controller. u refers to the control signal, r is the reference/input signal and y, the output signal. e is the error signal.

Fuzzy controllers consist of *if-else* statements, which reduces the complexity of designing such a controller. It is also mentioned that these controllers may have a better performance in non-linear systems. Even when utilising the *if-else* statements to design a controller, designing it can prove to be time consuming as multitude of variations can exist for a single problem, and all have to be defined using the mentioned statements.

- **Model Predictive Control.** A control solution that utilises a process model to derive future actions of the system. By estimating what the system will do in a future instance, the Model Predictive controller (MPC) can derive the control output to change/keep the desired outcome, using what has been defined as a Control Horizon. This horizon describes how a variable will behave in the future based on the control signal applied to the system to achieve the desired solution. This prediction is repeated every cycle and the control signal is updated, trying to make the system follow the desired trajectory.

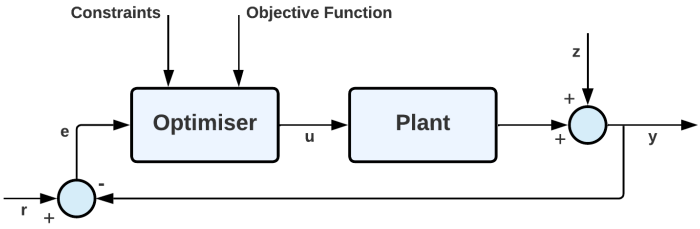


Figure 2.9: Illustration of an MPC. u refers to the control signal, r is the reference/input signal and y, the output signal. e is the error signal and z the disturbances affecting the system. This illustration has been based on [46].

MPCs are some of the most utilised controllers, this is due to the multiple advantages that come with its implementation. Some of these advantages are: it can control multiple variables simultaneously and is robust against disturbances. Furthermore, it can predict disturbances that may happen in some following time step, allowing the

controller to prepare a response beforehand.[47] In terms of disadvantages, Airikka et al.[48] mentions the difficulties this method has at handling non-linear processes.

- **Neural Network-based Control.** Neural Networks (NNs) are a set of approaches part of Machine Learning that imitate the brain of animals to solve numerous problems. One of the applications of NNs is in control systems. NN-based controllers utilise the ability of this networks to learn, making them optimal for systems that have to adapt to changes. As mentioned in previous paragraphs, some controllers have the ability to adjust controller parameters to force the system to act in a desired fashion. This can be achieved using NNs. When some disturbances are detected, these can be back-propagated through the network, which tunes the Neural Network, increasing its accuracy to detect and react to these. One of the disadvantages of this type of controllers is that a training phase, with its needed training data, has to be acquired.

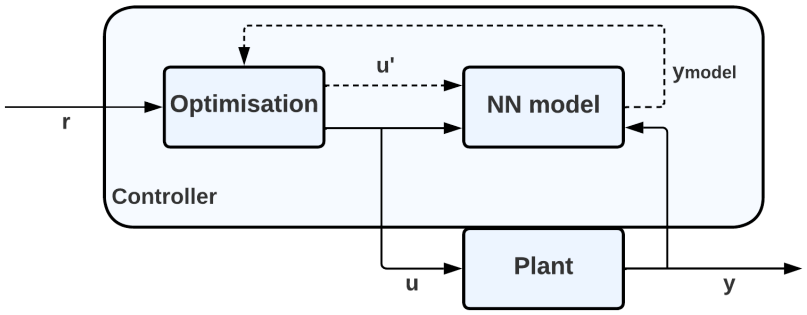


Figure 2.10: Illustration of a Neural Network-based Controller. The diagram depicts how the NN solution could be trained. u refers to the control signal, and y, the output signal. u' is the provisional control signal and y_{model} is the neural network model response. This illustration has been based on [49]

These solutions have the benefit of increasing the computational speed, a sought after quality in the control industry. By reducing the time needed for these calculations, faster reactions can be achieved by the controller. Another advantage of NN-based Control is the ability to 'learn', i.e. the controller can adjust the parameters on its own, and through an extended period of time. One of the few disadvantages of this type of approaches is the training phase. Training such a system and obtaining the necessary data for it has been proven to be cumbersome in some cases, as the data has to have sufficient quality and quantity.

Summary

In this section, four control approaches have been reviewed; PID control, Fuzzy control, MPC and NN-based control. From these, Model Predictive Control has been selected to be the controller used for this project. This has been decided by comparing the benefits and

drawbacks that the different control approaches offer. Neural Network-based controllers have been discarded due to the training phase, which has been considered as time and resource consuming compared to the other options listed. Another consideration made in account for this is that NN-based controllers are deemed as unnecessarily complex for a task such as the one presented in this project.
Fuzzy control was also rejected by similar reasons as NN-based controllers. The parametrisation of the controller as well as the design of the rules that make the fuzzy logic has been deemed as impractical for this project.
The last dismissed controller is the PID. The advantages that PID control offers this project have been considered to be positive and could potentially be a suitable solution for gantry crane control. Nevertheless, MPC offered slightly better benefits than PIDs, such as its ability to generate anticipatory control signals to predict possible outcomes and compensate for whatever may need compensation.

2.5 Summary

In this chapter, various solutions regarding gantry cranes have been researched and compared. From the initial investigation, it was decided to divide the rest of the analysis into control and path planning sections. From these various algorithms, some were listed and selected for further implementation on this project. In terms of path planning algorithms, Ant Colony Optimisation and Potential Field have been selected as they were deemed to better suit the project, as mentioned earlier. Since only one path planner can be at use for the final solution, the two path planning algorithms will be tested and compared. Resulting in a planner that can be proven to perform better than the others in the context of path planning of a gantry crane. The other distinctive group researched was Control Systems. Four controllers were researched and only one ended up being selected as the most viable option for this project, namely MPC. It was selected as two of the other options were considered to be time consuming compared to the other two. MPC was chosen over PID due to its ability to anticipate for any challenge that may happen.

In the following chapter, **Chapter 3**, the hardware setup for sensing and controlling the physical crane will be explained, and the mathematical model of the crane will be presented. In **Chapter 4** the requirements for the solution will be determined. The implementation of the solution will then be presented in **Chapter 5**. And the solution will be tested to determine whether it fulfills the requirements in **Chapter 6**. The results will then be evaluated in **Chapter 7** with a discussion of the results obtained from testing the solution. Finalising this paper in a conclusion, **Chapter 8**.

Chapter 3

Practical Implementation

In this chapter some of the steps required to make the physical and modelled cranes are mentioned and clarified. The first section delves into the construction and tuning of the control box that contains the microcontroller in charge of commanding the crane. Following the mentioned section, a description of the motor drivers and their utilisation are described, together with the various electronic components of the crane. Lastly, the mathematical model is described and the steps performed to derive it are shown.

3.1 Control Box

The crane is property of Aalborg University and has been lent to various groups throughout the years. This means that various changes have been made to the crane, improving it over time. The control box previously used to control the crane is missing from the laboratory. And the project group has therefore decided to make a new one, drawing inspiration from a previous version described in [50].

In order to implement a solution on this crane, a new control box was built. To make the control box, a micro-controller, to send and receive data, and some type of cabling between the crane and the micro-controller were needed. It was decided to use an Arduino Mega 2560 and a DB-25 connector, as the crane was made with such a connector in mind. The construction of the box will be detailed in the following **Subsections 3.1.1** and **3.1.2**.

3.1.1 DB-25 Connector

The Arduino does not have built-in connectivity for a DB-25 cable. Instead, each connecting wire from the cable was soldered onto a shield and connected to the micro-controller pins. In **Figure 3.1**, an image of the DB-25 connector can be seen.

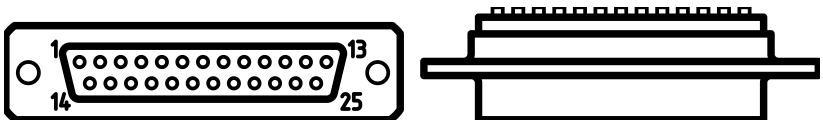


Figure 3.1: Schematic drawing of a male DB-25 connector. The numbering of the pins is as shown, it begins with pin 1 at the top left corner and goes through to pin 25, changing rows after pin 13. The numbering is important as different approaches may utilise different numbering.

A cable containing both a male and female DB-25 connector was used. The female part was cut off and the inside wires were exposed. The wires were each matched to the corresponding pin on the male part of the connector using a multimeter. And the function of each wire was identified based on the pin mapping from a previous paper on the crane [50]. The following table contains the relevant information regarding the pins and the corresponding wires **Table 3.1**.

Pin	Colour	Connect.	Description	Interface
1	White/ Brown	RX-Head	Serial Data transmitted to head	V-range $\rightarrow$ 0V - 5V Baud-rate $\rightarrow$ 9600
2	Black	X-Tacho	Voltage from X-tachometer (filtered)	V-range $\rightarrow \pm$26V
3	White	Y-Tacho	Voltage from Y-tachometer (filtered)	V-range $\rightarrow \pm$26V
5	Purple	Y-Driver A02	Analogue voltage from Y-driver	V-range $\rightarrow$ 0V - 4V
7	Teal	X-Enable	Enables X-Driver	V-range $\rightarrow$ 0V - 36V Logic 0 $\rightarrow$ < 1V Logic 1 $\rightarrow$ > 2.4V
8	Green	Y-Pos	Voltage from Y-potentiometer (unfiltered)	V-range $\rightarrow$ 0V - 5V
9	Yellow	X-Driver A02	Analogue voltage from X-driver	V-range $\rightarrow$ 0V - 4V
10	Pink	Y-Driver A01	Analogue voltage from Y-driver	V-range $\rightarrow$ 0V - 4V
11	Orange	Vin	Positive power supply to controller	STD-V $\rightarrow$ 15V
12	Red	Y-Enable	Enables Y-Driver	V-range $\rightarrow$ 0V - 36V Logic 0 $\rightarrow$ < 1V Logic 1 $\rightarrow$ > 2.4V
13	Brown	Pos-Supply	Positive voltage supply from controller	STD-V $\rightarrow$ 5V
14	Black/ Light Gray	X-Tacho	Voltage from X-tachometer (unfiltered)	V-range $\rightarrow \pm$26V

Table 3.1 – continued from previous page

Pin	Colour	Connect.	Description	Interface
15	Black/ Dark Grey	X-Driver A01	Analogue voltage from X-driver	V-range → 0V - 4V
16	Purple/ White	Y-Tacho	Voltage from Y-tachometer (unfiltered)	V-range → ±26V
17	Blue/ White	X-PWM	PWM for X-motor Driver	V-range: 0V - 36V Freq.: 10Hz - 5E3Hz Duty-cycle:10% - 90%
18	Green/ Black	TX-Head	Serial Data received from head (unfiltered)	V-range → 0V - 5V Baud-rate → 9600
19	Green/ White	Y-PWM	PWM for Y-motor Driver	V-range: 0V - 36V Freq.: 10Hz - 5E3Hz Duty-cycle: 10% - 90%
20	Yellow/ Black	X-Pos	Voltage from X-potentiometer (unfiltered)	V-range → 0V - 5V
21	Black/ White	Y-Pos	Voltage from Y-potentiometer (filtered)	V-range → 0V - 5V
22	Orange/ Black	X-Pos	Voltage from X-potentiometer (filtered)	V-range → 0V - 5V
23	Orange/ White	GND	Ground connection	STD-V → 0V
24	Red/ Black	Pos-Supply	Positive voltage supply	STD-V → 5V
25	Red/ White	TX-Head	Serial Data received from head (filtered)	V-range → 0V - 5V Baud-rate → 9600

Table 3.1: Table containing the pins from the DB-25 cable that connects the Arduino to the crane. The colours listed refer to the colouring of each of the wires in the DB-25 connector. The pins which are not used in transmitting and receiving information to and from the crane are not listed.

3.1.2 Arduino: Electronics

As mentioned, an Arduino Mega 2560 was used to control the crane. The controller had to be arranged such that the data flow going through the DB-25 connector, from the crane, could be interpreted and used.

The DB-25 could now be connected to the Arduino, by wiring the 25 cables to the board. Due to the quantity of cables that had to be connected, it was decided to utilise a Shield on which the wires could be soldered. A drawn representation of both the Arduino and the Shield can be seen in **Figures 3.2** and **3.3**.

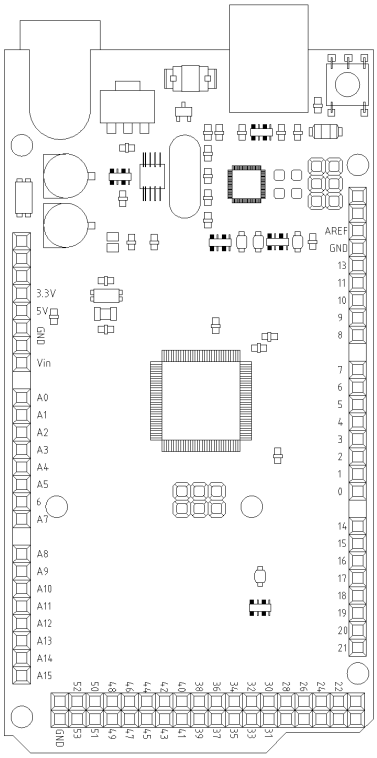


Figure 3.2: Arduino Mega 2560.

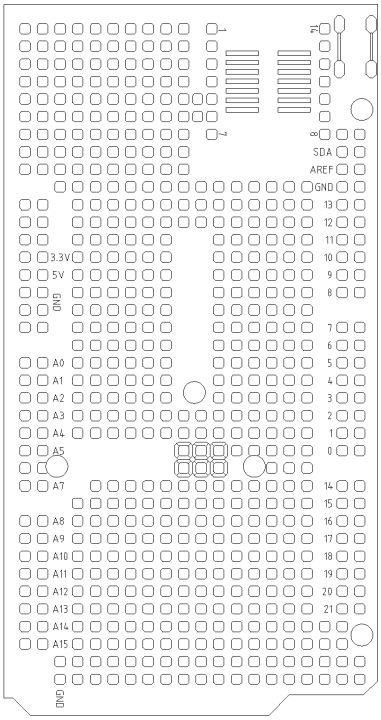


Figure 3.3: Arduino Shield.

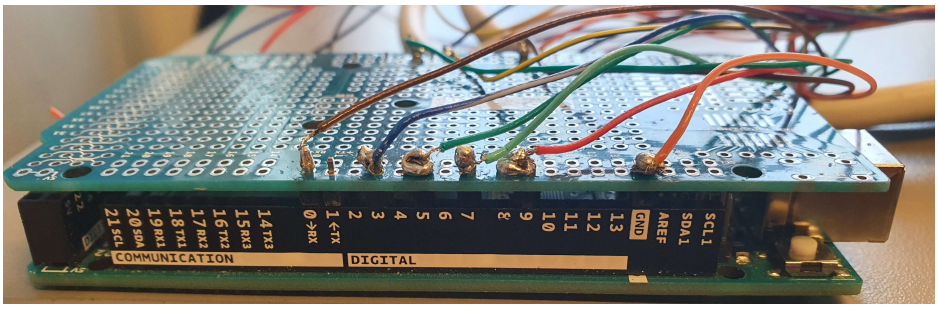


Figure 3.4: Side view of an Arduino Mega 2560 with an Arduino Shield placed on top.

With this, the end of the cable (male DB-25 connector) could be plugged to the crane. The cables used and their respective Arduino Mega pin-outs are depicted in the following figure. The image contains the naming of the pins shown by the specifications.

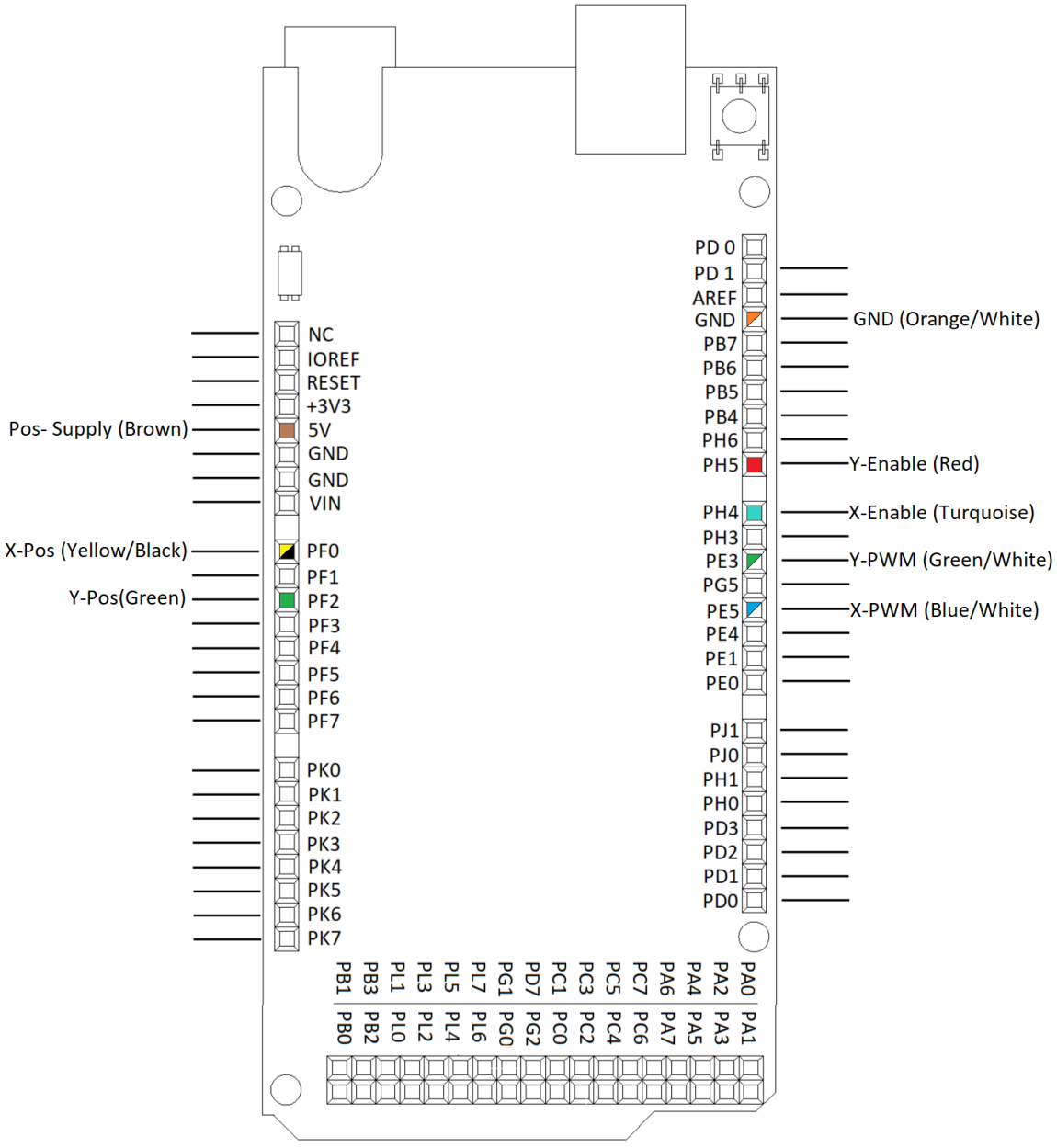


Figure 3.5: Schematic of the Arduino MEGA pin-outs and their respective use within this project. Within the Arduino, the microcontroller pins can be seen, outside of it, the crane elements connected to each pin.

3.2 Crane Electronics

This section delves into the different electronic components of the overhead crane.

3.2.1 ESCON Motor Driver

The movement of the crane is carried out by two motors, one for each direction. Each motor is controlled by an ESCON motor driver which can be modified in the program "ESCON studio". The specifications of the connected motor, as well as control settings need to be defined in the ESCON setup. Once all the system parameters have been chosen, the controller monitor in ESCON studio can be used to check whether or not the driver is receiving control signals and/or outputting measurement data.

3.2.2 PWM Signals

Communication with the motor drivers happens by using PWM signals sent from the Arduino Mega through the PWM wires in the DB-25. The motor drivers then send a corresponding voltage to the motors. The angular accelerations of the motors are proportional with the PWM signals in the duty cycle span $[10, 90]\%$. Where a duty cycle of 50% corresponds to no acceleration, 90% corresponds to maximum positive acceleration, and 10% corresponds to maximum negative acceleration. The outermost duty cycles of $[0, 10)\%$ and $(90, 100]\%$ also correspond to no acceleration.

3.2.3 Enable Signals

The motor drivers need to be enabled by an enable signal before they can be used to control the crane. This signal is sent from the Arduino mega when the main program is executed, through the X-Enable and Y-Enable wires, **Figure 3.5**, in the DB-25 connector.

3.2.4 Tachometers

There are two tachometers on the crane, which are used to measure the velocity of the trolley and the load, respectively. The sensors output a signal with a voltage that is proportional to the velocity, each through a wire of the DB-25 connector. The voltage range of these signals are $[-26, 26]$V, where a positive versus negative voltage represents the direction of the crane's velocity. The signals need to be received by the Arduino micro controller to be used in controlling the crane. However, the voltage range of the tachometer signals is outside of the tolerances of the micro controller, which is only $[-0.5, 5.5]$V. Therefore, an op amp (Operational Amplifier) circuit is used to convert the voltage range of the signal to $[0, 5]$V, which is the recommended voltage range of the micro controller pins [51].

Op Amp Circuit

It has been decided that a non-inverting summing amplifier will be used for each tachometer signal, as it can add an offset, V_{off}, to the given tachometer signal, V_{tach}, resulting in a non-negative signal. Such a circuit can be described by **Equation 3.1**:

$$V_{out} = \left(1 + \frac{R_f}{R_i}\right) \cdot \frac{V_{tach} \cdot R_2 + V_{off} \cdot R_1}{R_2 + R_1} \tag{3.1}$$

When $R_1 = R_2$, the equation can be simplified to:

$$V_{out} = \left(1 + \frac{R_f}{R_i}\right) \cdot \frac{V_{tach} + V_{off}}{2} \tag{3.2}$$

Where R_f is the feedback resistor and R_i is the resistor connected to the inverting input of the op amp.

The resistance part of the equation is essentially a scalar, and leaving it out for now and choosing an offset of 26V results in a voltage range of $[0, 26]$, which has the desired minimum voltage of 0V.

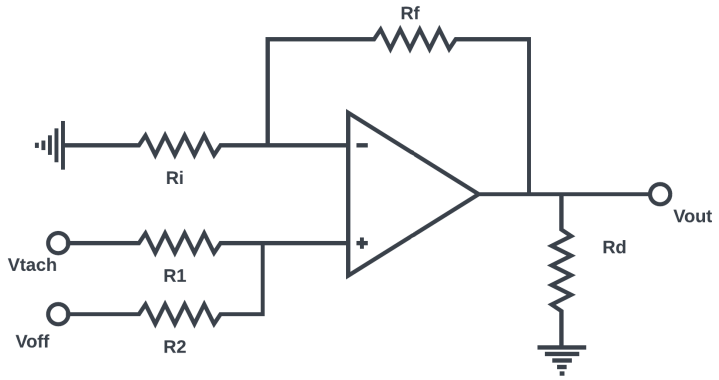


Figure 3.6: A diagram of the op amp circuit. Where V_{tach} is the signal from the tachometer, V_{off} is the offset voltage, and V_{out} is the output voltage. R_1 is the resistor on the tachometer input, R_2 is on the offset input, R_i is on the inverting input of the amplifier, R_f is on the feedback loop, and R_d is the voltage dividing resistor. R_d and the GND connection from the output would not typically be present in a summing amplifier.

The resistors in the circuit can then be chosen such that the maximum voltage is scaled to the desired 5V. However, mathematically, it would require a resistor with a negative resistance to do so, because otherwise the left side of **Equation 3.2** would always be ≥ 1.

To solve this problem, it was decided to connect a resistor between the output and GND as can bee seen in **Figure 3.6**, it was theorised it would function like a voltage divider (see

Appendix A). A variable resistor has been used to tune the voltage dividing resistor, R_d, and R_i such that the output voltage is within the desired voltage range. The other resistors, R_f, R_1, and R_2 were chosen to be 100kΩ, as high resistance results in low current, which is desired in this case.

This results in an output signal with a voltage range of $V_{out} = [0, 5]$V, given a tachometer signal of $V_{in} = [-26, 26]$V. Through testing several inputs, it has been concluded that the input-output relation is linear. [52]

3.2.5 IMU

The orientation data from the IMU is received by an Arduino Nano micro controller, which is attached to the head of the crane. The swing angle of the head of the crane is calculated on said micro controller. This angle data is then sent to the Arduino mega through serial communication wires in the DB-25 connector.

3.2.6 Electromagnet

The Arduino Nano is also responsible for activating and deactivating the electromagnet on the head of the crane, which is used to pick and place the containers. Controlling the magnet is done through the same serial communication wires as the angle data from the IMU. When a command is sent from the PC to the Arduino mega, the mega sends a signal to the Arduino Nano, which then enables or disables the magnet. The commands used are M1 to enable, and M0 to disable, followed by a new line character, \n.

3.2.7 Potentiometers

Two potentiometers, which are attached to the crane, are used for measuring the position of the trolley and the head of the crane. The potentiometers output a voltage which is proportional to the position, which is sent through the DB-25 connector and received by the Arduino mega micro controller. Additionally, the potentiometers are powered by the Arduino mega, through the Pos-Supply wire, **Figure 3.5** in the DB-25.

3.3 Model of overhead crane

In this section, a model of the overhead crane system will be created for the purpose of isolating the equations of motion. This crane has been lent by Aalborg University and has been described in **Section 2.1**. An extended version of the steps necessary to derive the model can be seen in **Appendix B**.

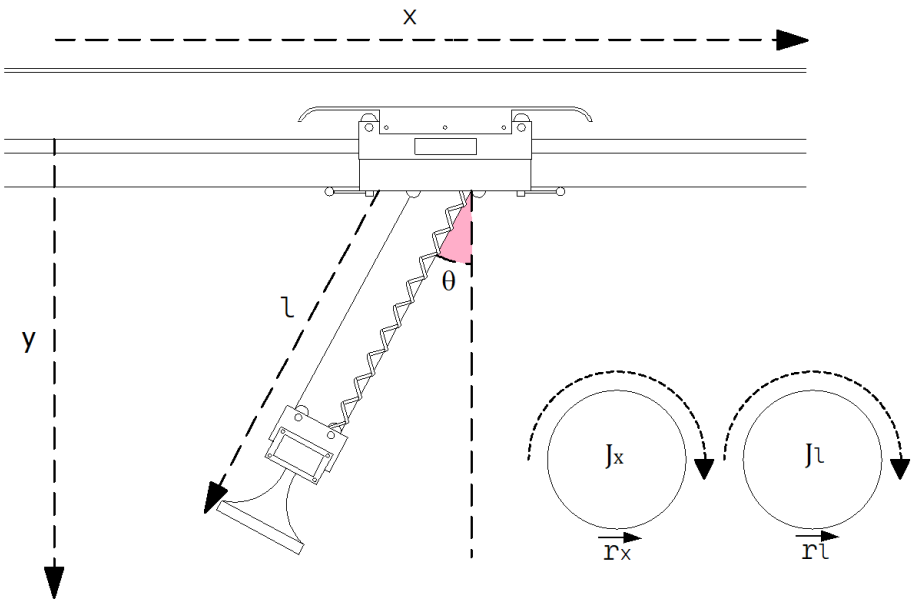


Figure 3.7: Head and Trolley with angles and positions

where	x	:	Position on the horizontal axis	(m)
	y	:	Position on the vertical axis	(m)
	l	:	Length of the wire	(m)
	θ	:	Angle between the load and its equilibrium point	$(°)$
	J_x	:	Moment of Inertia for the x direction motor	$(\cdot)$
	J_l	:	Moment of Inertia for the wire length motor	$(\cdot)$
	r_x	:	Radius of the X direction drum	(m)
	r_l	:	Radius of the wire-length drum	(m)

The model will be derived using Lagrange equations, which start by describing the systems kinetic and potential energies, which in turn require the positional equation shown below in **Equation 3.3**.

$$\vec{p}_t = x_t \cdot \vec{i} + y_t \cdot \vec{j}\,|_{y_t=0} \tag{3.3}$$

where	$\vec{p}_t$	:	Vector position of the trolley	(m)
	x_t	:	Trolley x position	(m)
	$\vec{i}$	:	Unit vector for x	$(\cdot)$
	y_t	:	Trolley y position	(m)
	$\vec{j}$	:	Unit vector for y	$(\cdot)$

Only a 2-D crane is considered for now, resulting in the y component for the trolley being zero.

The position of the load is given in **Equation 3.4**.

$$\begin{aligned}\vec{p}_l = x_l \cdot \vec{i} + y_l \cdot \vec{j} &= (x_t + l \cdot sin(\theta)) + (y_t + l \cdot cos(\theta)) \mid_{y_t=0} \\ &= (x_t + l \cdot sin(\theta)) + l \cdot cos(\theta)\end{aligned} \tag{3.4}$$

where
$\vec{p}_l$	:	Vector position of the trolley	(m)
x_l	:	Trolley x position	(m)
y_l	:	Trolley y position	(m)
l	:	Length of the wire	(m)
θ	:	Angle between trolley and load	$(°)$

With the positional descriptions in place, the Lagrangian can be derived. The kinetic energy of the system comes from four sources, namely the trolley and the load, as well as their respective servo motor drums.
The kinetic energy of the trolley:

$$T_t = \frac{1}{2} \cdot m_t \cdot \dot{x}_t^2 \tag{3.5}$$

where
T_t	:	Kinetic energy of the trolley	(J)
m_t	:	Mass of trolley	(kg)
$\dot{x}$	:	x velocity of trolley	(m/s)

Kinetic energy of the load:

$$T_l = \frac{1}{2} \cdot m_l \cdot \left(\dot{x}_l^2 + \dot{y}_l^2\right) \tag{3.6}$$

where
T_l	:	Kinetic energy of the load	(J)
m_l	:	Mass of load	(kg)
$\dot{x}$	:	x velocity of load	(m/s)
$\dot{y}$	:	y velocity of load	(m/s)

The kinetic energy of the rotating wire drums are calculated as a rotational kinetic energy instead, which substitutes the mass term for the moment of inertia of the body and the velocity for angular velocity of the drum. This angular velocity is not a measured quantity in this project, and is thus related to the linear velocity of the wire and the drums radius.
The kinetic energy of the drum for x is presented in **Equation 3.7**.

$$T_{px} = \frac{1}{2} \cdot J_x \cdot \left(\frac{\dot{x}}{r_x}\right)^2 \tag{3.7}$$

where
T_{px}	:	Kinetic energy of the drum for x	(J)
J_x	:	Moment of Inertia of the drum for x	$(kg \cdot m^2)$
r_x	:	Radius of the drum for x	(m)

The kinetic energy of the drum for hoisting is presented in **Equation 3.8**.

$$T_{pl} = \frac{1}{2} \cdot J_l \cdot \left(\frac{2\dot{l}}{r_l}\right)^2 \tag{3.8}$$

where T_{pl} : Kinetic energy of the drum for l (J)
J_l : Moment of Inertia of the drum for l $(kg \cdot m^2)$
r_l : Radius of the drum for l (m)

For T_{pl}, the velocity is multiplied by two due to the gearing in the drum for the hoisting wire being different to its counterpart in x, resulting in a different approximation.
The kinetic energy of the system is the sum of its parts and is thus given in **Equation 3.9**;

$$T = \frac{1}{2} \cdot m_t \cdot \dot{x}_t^2 + \frac{1}{2} \cdot m_l \cdot \left(\dot{x}_l^2 + \dot{y}_l^2\right) + \frac{1}{2} \cdot J_x \cdot \left(\frac{\dot{x}}{r_x}\right)^2 + \frac{1}{2} \cdot J_l \cdot \left(\frac{2\dot{l}}{r_l}\right)^2 \tag{3.9}$$

The second term of this equation describes the kinetic energies of the system relating to the velocity of the load, which is not a measured parameter in this project. It can however be further related to the velocity of the trolley using **Equation 3.4**, also shown below in **Equation 3.10**.

$$\vec{p}_l = (x_t + l \cdot sin(\theta)) + l \cdot cos(\theta) \tag{3.10}$$

where $x_l = (x_t + l \cdot sin(\theta))$ and $y_l = l \cdot cos(\theta)$.
Isolating the terms representing x_l and y_l, these can be differentiated and squared to obtain $(\dot{x}_l^2 + \dot{y}_l^2)$ in **Equation 3.9**.

$$\dot{x_l}^2 + \dot{y}_l^2 = \dot{x_t}^2 + \dot{l}^2 + l^2\dot{\theta}^2 + 2\dot{x}_t\dot{l}sin(\theta) + 2\dot{x}_t l\dot{\theta}cos(\theta) \tag{3.11}$$

Introducing $(\dot{x}_l^2 + \dot{y}_l^2)$ into **Equation 3.9** gives,

$$\begin{aligned} T = {} & \frac{1}{2} \cdot m_t \cdot \dot{x}_t^2 + \frac{1}{2} \cdot m_l \cdot \left(\dot{x_t}^2 + \dot{l}^2 + l^2\dot{\theta}^2 + 2\dot{x}_t\dot{l}sin(\theta) + 2\dot{x}_t l\dot{\theta}cos(\theta)\right) + \frac{1}{2} \cdot J_x \cdot \left(\frac{\dot{x}}{r_x}\right)^2 \\ & + \frac{1}{2} \cdot J_l \cdot \left(\frac{2\dot{l}}{r_l}\right)^2 \end{aligned} \tag{3.12}$$

The complete equation for the kinetic energies has now been acquired. The next step is to derive the potential energy.

Assuming the origin for y is set at the trolleys level, it effectively does not have any potential energy, leaving only the load, which is also defined as having zero potential energy when it is at $y = 0$. The potential energy equation is thus given as:

$$V = -m_l \cdot g \cdot y_l = -m_l \cdot g \cdot (y_t + l \cdot cos(\theta)) \mid_{y_t=0} = -m_l \cdot g \cdot l \cdot cos(\theta) \tag{3.13}$$

where V : Potential energy of system (J)
g : Gravity constant = 9.82 $(m \cdot s^{-2})$

The Lagrangian L for the system can now be presented:

$$\begin{aligned} L = T - V = \frac{1}{2} m_t \dot{x}_t^2 + \frac{1}{2} m_l \left({\dot{x}_t}^2 + \dot{l}^2 + l^2 \dot{\theta}^2 + 2\dot{x}_t \dot{l} sin(\theta) + 2\dot{x}_t l \dot{\theta} cos(\theta) \right) \\ + \frac{1}{2} J_x \left(\frac{\dot{x}}{r_x} \right)^2 + \frac{1}{2} J_l \left(\frac{2\dot{l}}{r_l} \right)^2 + m_l \cdot g \cdot l \cdot cos(\theta) \end{aligned} \tag{3.14}$$

With this, the equations of motion can be found. Three variables are present, so an equation for each is needed. The Lagrangian equation when no external force is present and the Lagrangian when an external force is present, can be defined as:

$$0 = \frac{d}{dt} \frac{\partial}{\partial \dot{p}} L - \frac{\partial}{\partial p} L \tag{3.15}$$

$$F_z = \frac{d}{dt} \frac{\partial}{\partial \dot{p}} L - \frac{\partial}{\partial p} L \tag{3.16}$$

Equation of motion for $\ddot{x}$

The servo motor, pulling the trolley's wire, is considered an external force, meaning **Equation 3.16** is used. The force can represented as shown below in **Equation 3.17** together with a friction term,

$$F_x = \frac{K_{ex} \cdot I_x}{r_x} - B_x \cdot \dot{x}_t \tag{3.17}$$

where F_x : Force acting along x (N)
K_{ex} : Motor constants for trolley motor $(\cdot)$
I_x : Current delivered to trolley motor (A)
r_x : Radius of trolley drum (m)
B_x : Friction coefficient for moving along x (N)
$\dot{x}$: Velocity of the hoisting system (m/s)

The Lagrangian equation for x can now be calculated.

$$\frac{K_{ex} \cdot I_x}{r_x} - B_x \cdot \dot{x}_t = \frac{d}{dt} \frac{\partial}{\partial \dot{x}_t} L - \frac{\partial}{\partial x_t} L \tag{3.18}$$

Which, after some computations, derives in the following equation of motion for $\ddot{x}$:

$$\begin{aligned} \ddot{x}_t = \frac{1}{m_t + m_l + \frac{J_x}{r_x^2}} \Bigg(\frac{K_{ex} I_x}{r_x} - B_x \dot{x}_t - m_l \ddot{l} sin(\theta) - 2 m_l \dot{l} \dot{\theta} cos(\theta) \\ - m_l l \ddot{\theta} cos(\theta) + m_l l \dot{\theta}^2 sin(\theta) \Bigg) \end{aligned} \tag{3.19}$$

Equation of motion for $\ddot{l}$

For calculating the forces acting along the hoisting wire, the Lagrange's equation will be solved with respect to l. There is a motor applying force on this axis as well, thus the same form of Lagrange is used. The force term is presented in **Equation 3.20**;

$$F_l = \frac{2K_{el} \cdot I_l}{r_l} - B_l \cdot \dot{l} \tag{3.20}$$

where
F_l	:	Force applied by hoisting motor	(N)
K_{el}	:	Motor constants for hoisting motor	$(\cdot)$
I_l	:	Current delivered to the hoisting motor	(A)
r_l	:	Radius of hoisting motor drum	(m)
B_l	:	Friction coefficient for moving along y	$(\cdot)$
$\dot{l}$	:	Velocity of the hoisting system	(m/s)

Thus the Lagrange equation to solve is shown in **Equation 3.21**.

$$\frac{K_{el} \cdot I_l}{r_l} - B_l \cdot \dot{l} = \frac{d}{dt}\frac{\partial}{\partial \dot{l}}L - \frac{\partial}{\partial l}L \tag{3.21}$$

The solution of calculating the Lagrangian derives in **Equation 3.22**

$$\ddot{l} = \frac{1}{m_l + \frac{4J_l}{r_l^2}} \left(\frac{K_{el} I_l}{r_l} - B_l \dot{l} + m_l l \dot{\theta}^2 + m_l g cos(\theta) - m_l \ddot{x}_t sin(\theta) \right) \tag{3.22}$$

Equation of motion for $\ddot{\theta}$

Since wind is not considered in this project, there is no active force acting on the angle θ. Friction and air resistance for dampening the sway are also not considered, thus the resulting Lagrange equation is equal to zero, as seen in **Equation 3.23**.

$$0 = \frac{d}{dt}\frac{\partial}{\partial \dot{\theta}}L - \frac{\partial}{\partial \theta}L \tag{3.23}$$

Besides that change, the calculations are the same as for x and l, with each term done separately as before. Obtaining the equation of motion for $\ddot{\theta}$:

$$\ddot{\theta} = \frac{1}{l}\left(-2\dot{\theta}\dot{l} - \ddot{x}_t cos(\theta) - g sin(\theta)\right) \tag{3.24}$$

Isolating the accelerations

The model of the crane is simulated using Simulink, a Matlab programming environment used for simulation and modelling of various systems. To simulate this model, Simulink requires a certain order to simulate the three equations of motion. This could be solved

by adding delays and stating the preferred order. Nevertheless, it was decided that, to simplify the modelling and simulation, the equations will be isolated from each other, i.e., $\ddot{l}$ and $\ddot{\theta}$ will be modified as to not directly contain $\ddot{x}$.

First, the right-hand side of the equations for $\ddot{l}$ and $\ddot{\theta}$ are introduced into the equation of motion for $\ddot{x}$, so that **Equation 3.19** only contains $\ddot{x}$.
Thereafter, the newly obtained equation of motion for $\ddot{x}$ can be introduced into **Equations 3.22** and **3.24**, equations of motion for $\ddot{l}$ and $\ddot{\theta}$, respectively.

$$\ddot{x}_t = \frac{1}{\left(m_t + m_l + \frac{J_x}{r_x^2} - m_l cos^2(\theta) - \frac{m_l^2 sin^2(\theta)}{m_l + \frac{4J_l}{r_l^2}}\right)} \left[\frac{K_{ex} I_x}{r_x} - B_x \dot{x}_t - \frac{m_l K_{el} I_l sin(\theta)}{r_l m_l + \frac{4J_l}{r_l}} + \frac{m_l B_l \dot{l} sin(\theta)}{m_l + \frac{4J_l}{r_l^2}} + m_l l \dot{\theta}^2 sin(\theta) \left(1 - \frac{m_l}{m_l + \frac{4J_l}{r_l^2}}\right) + m_l g sin(\theta) cos(\theta) \left(1 - \frac{m_l}{m_l + \frac{4J_l}{r_l^2}}\right)\right] \tag{3.25}$$

$$\ddot{l} = \frac{1}{m_l + \frac{4J_l}{r_l^2}} \left[\frac{K_{el} I_l}{r_l} - B_l \dot{l} + m_l l \dot{\theta}^2 + m_l g cos(\theta) - \frac{m_l sin(\theta)}{\left(m_t + m_l + \frac{J_x}{r_x^2} - m_l cos^2(\theta) - \frac{m_l^2 sin^2(\theta)}{m_l + \frac{4J_l}{r_l^2}}\right)} \left(\frac{K_{ex} I_x}{r_x} - B_x \dot{x}_t - \frac{m_l K_{el} I_l sin(\theta)}{r_l m_l + \frac{4J_l}{r_l}} + \frac{m_l B_l \dot{l} sin(\theta)}{m_l + \frac{4J_l}{r_l^2}} + m_l l \dot{\theta}^2 sin(\theta) \left(1 - \frac{m_l}{m_l + \frac{4J_l}{r_l^2}}\right) + m_l g sin(\theta) cos(\theta) \left(1 - \frac{m_l}{m_l + \frac{4J_l}{r_l^2}}\right)\right)\right] \tag{3.26}$$

$$\ddot{\theta} = \frac{1}{l} \left[-2\dot{\theta}\dot{l} - \frac{cos(\theta)}{\left(m_t + m_l + \frac{J_x}{r_x^2} - m_l cos^2(\theta) - \frac{m_l^2 sin^2(\theta)}{m_l + \frac{4J_l}{r_l^2}}\right)} \left(\frac{K_{ex} I_x}{r_x} - B_x \dot{x}_t - \frac{m_l K_{el} I_l sin(\theta)}{r_l m_l + \frac{4J_l}{r_l}} + \frac{m_l B_l \dot{l} sin(\theta)}{m_l + \frac{4J_l}{r_l^2}} + m_l l \dot{\theta}^2 sin(\theta) \left(1 - \frac{m_l}{m_l + \frac{4J_l}{r_l^2}}\right) + m_l g sin(\theta) cos(\theta) \left(1 - \frac{m_l}{m_l + \frac{4J_l}{r_l^2}}\right)\right) - g sin(\theta)\right] \tag{3.27}$$

With this, the process of deriving the mathematical model for the crane can be considered finished. As mentioned at the beginning of this section, the expanded version of the computations described here can be seen in **Appendix B**. In **Appendix C** a 3D version of this section can be seen, derived for future implementation.

3.4 Summary

Chapter 2 analysed different approaches for designing a path planning and control of an overhead crane. From the analysis, it was concluded to implement and test two algorithms for path planning, ACO and Potential Fields. As the controller, MPC proved to be a reasonable approach and was, therefore, selected as the controller for this solution. One requirement of MPC is the need for a mathematical version of the system as well as knowledge of what the inputs and outputs said controller can receive from the crane. To obtain this information, the dynamic model of the system has been derived.
This chapter started by describing how the crane communicates with an external device, by means of a control box. As mentioned, said box was not available, so a new one had to be built. After the communication was solved, the mathematical model of the overhead crane was derived and can be seen in **Section 3.3**.

Some requirements will be set in the following chapter, so that the solution can be tested and accepted within the desired boundaries. Afterwards, the implementation chapter will describe the solution designed and how it was implemented.

Chapter 4

Requirements Specification

This chapter will state and and justify the requirements that are needed for the solution to be considered acceptable, as well as which tests are necessary to verify that these requirements are met.

4.1 Requirements

There are three main subsystems in the solution: MPC, hardware and trajectory planning. All of these will be required to follow a number of conditions which will be stated in this section. The requirements will be divided into these three groups. The layout of this section will start by defining the requirements, then, explaining the reasoning and benefits of each. Lastly, an acceptance test for each requirement will be design.

4.1.1 MPC Requirements

1.1 *For a simulated model of the system, any x and l position must be reachable for the MPC.* The MPC must be able to move to and reach a steady state for any given reference for x and/or l.

1.2 *For a simulated model of the system, the MPC must be able to follow any trajectory operating within the limits of the physical crane.* The system relies on the MPC being capable of following any path that it receives, within the boundaries set by the physical limitations of the crane.

4.1.2 Hardware Requirements

2.1 *The system must be able to measure the position states using potentiometers.* The position states, x, and l, from the inbuilt crane potentiometers, must be available at all times and represent the physical system.

2.2 *The system must be able to measure the velocity states using tachometers.* The velocity states, $\dot{x}$, and $\dot{l}$, from the inbuilt crane tachometers, must be available at all times and represent the physical system.

2.3 *The system must be able to measure the sway angle state using an IMU.* The sway angle state, θ, from the IMU on the crane head, must be available at all times and represent the physical system.

2.4 *The system must be capable of picking up and placing containers.* The system must be able to enable and disable the electromagnet on command, and thereby allow for the crane to lift, move, and release the shipping container models.

2.5 *The system must be able to move the trolley and crane head in all available directions.* The system must be able control the trolley acceleration in both directions along the x-axis, by sending PWM control signals to the x-motor controller. The system must be able to control the wire length, l, by sending PWM control signals to the y-motor controller. In addition, the system must be able to send a valid enable signal to each motor controller, and thereby enable them.

4.1.3 Trajectory Planning Requirements

3.1 *The PF planner must not fall into the Local Minima Problem.* Potential Field planning tends to have difficulties at generating a path due to getting stuck in local minima. Test if the solution can solve this.

3.2 *Potential Field Path Planning has to compute the shortest path.* The goal of the path planning is to derive the shortest path from start to end, therefore, the planner is only allowed to be 10cm longer than the shortest path possible.

3.3 *ACO Path Planning has to compute the shortest path.* The goal of the path planning is to derive the shortest path from start to end, therefore, the planner is only allowed to be 10cm longer than the shortest path possible.

3.4 *Comparison between path planning algorithms.* Compare the results of both algorithms to discover their benefits and disadvantages against each other.

4.1.4 System Requirements

4.1 *The MPC must be capable of giving commands to the hardware components of the system.* The control signals derived from the MPC must be followed by the physical crane.

4.2 *The solution must complete a task by the continuous interaction of the three subsystems mentioned, Path Planner, MPC, and Hardware.* The overall system consists of three subsystems that must collaborate with each other for a container crane to perform the task of moving containers.

4.2 Acceptance Tests

After stating and defining the requirements, these will have to be accepted using some procedure that ensures its correct implementation. The following table will contain the acceptance test for each of the requirements stated in this chapter.

Req:	Test procedure:	Acceptance:
	MPC Requirements	
1.1	A reference x and l will be fed to the MPC. Record the outcome position, velocity and acceleration of the system variables.	The MPC must be capable of controlling the system such that its controlled variables reach steady state. The result of this test can be considered accepted if, for the reference x and l, the simulated output reaches a steady state.
1.2	The MPC will follow a desired path. The simulated outcome of the system will be recorded.	The solution has been designed to follow a desired trajectory, therefore, the MPC must be capable deriving the necessary control signals so that the system follows said path. The test is considered passed if the MPC is capable of following the desired path within the physical limitations that the crane actually has.
	Hardware Requirements	
2.1	Record data of the potentiometers measuring the cranes position for both x and l.	The requirement can be considered to be fulfilled if the potentiometer measurements can be obtained and recorded for further use.
2.2	Record data of the tachometers measuring the cranes position for both $\dot{x}$ and $\dot{l}$.	The requirement can be considered to be fulfilled if the tachometer measurements can be obtained and recorded for further use.

Continued on next page

Table 4.1 – continued from previous page

Req:	Test procedure:	Acceptance:
2.3	Record data of the IMU where angle of the crane's head θ using the IMU.	The requirement can be considered to be fulfilled if the IMU measurements for the crane's head angle can be obtained and recorded for future use.
2.4	The crane will be commanded to pick and place a container using the electromagnet.	The test can be deemed as passed if it can be proved that the electromagnet can be utilised on command.
2.5	The system will be asked to move the crane's head in both the horizontal and vertical directions.	The requirement can be deemed as met if the PWM signals and enable signals used to control the motor controllers deliver the desired motions.
Trajectory Planning Requirements		
3.1	Place 1-3 obstacles in different locations of the workspace. Run the program for 10 distinct iterations, and store the results.	Potential Field has the disadvantage of getting stuck in local minima. The solution should have a remedy for this problem. This test compares the results of the planner with the added complement and without it. The test can be considered as passed if the results show that the add-on allows the planner to solve the local minima problem.

Continued on next page

Table 4.1 – continued from previous page

Req:	Test procedure:	Acceptance:
3.2	Place 1-3 obstacles in different locations of the workspace. Run the program for 10 distinct iterations, and store the results.	This test examines the ability of the Potential Field Path Planner to generate a path from start to goal. This will be compared to a manually generated path, considered to be the benchmark path for the specific workspace layout. This requirement can be considered to be met if the path generated for each iteration using the potential field approach is within 10cm.
3.3	Place 1-3 obstacles in different locations of the workspace. Run the program for 10 distinct iterations, and store the results.	This test examines the ability of the ACO Path Planner to generate a path from start to goal. This will be compared to a manually generated path, considered to be the benchmark path for the specific workspace layout. This requirement can be considered to be met if the path generated for each iteration using the ACO approach is within 10cm.
3.4	Compare the data obtained when testing **Requirements 3.2** and **3.3**.	One of the goals of this project is to discover which algorithm is more suitable for the solution. This test will result in a decision as to which algorithm will be part of the final solution. The algorithm that derives the shortest paths, compared with their respective manually generated paths, will be considered to be the optimal path planner for this project.
System Requirements		

Continued on next page

Table 4.1 – continued from previous page

Req:	Test procedure:	Acceptance:
4.1	Make MPC move the crane to a point in the workspace and check that it can stabilise in said position, for both x and l.	
4.2	The task of moving a container from one location to another must be completed. Said task should begin by computing a path from start to finish, which the controller must then pass on to the crane as control signals. The crane must pick up the object from the initial location and place it at the desired end position.	For the requirement to be met, the subsystems must be capable of interacting with each other to perform the desired task.

Table 4.1: Table specifying the tests and evaluation conditions for the proposed solution.

Chapter 5

Implementation

This chapter will delve into the process of implementing a solution for an overhead crane. Following the research in **Chapter 2**, a path planning algorithm will be designed, namely, Potential Field. To control the system, a Model Predictive Controller will be implemented.

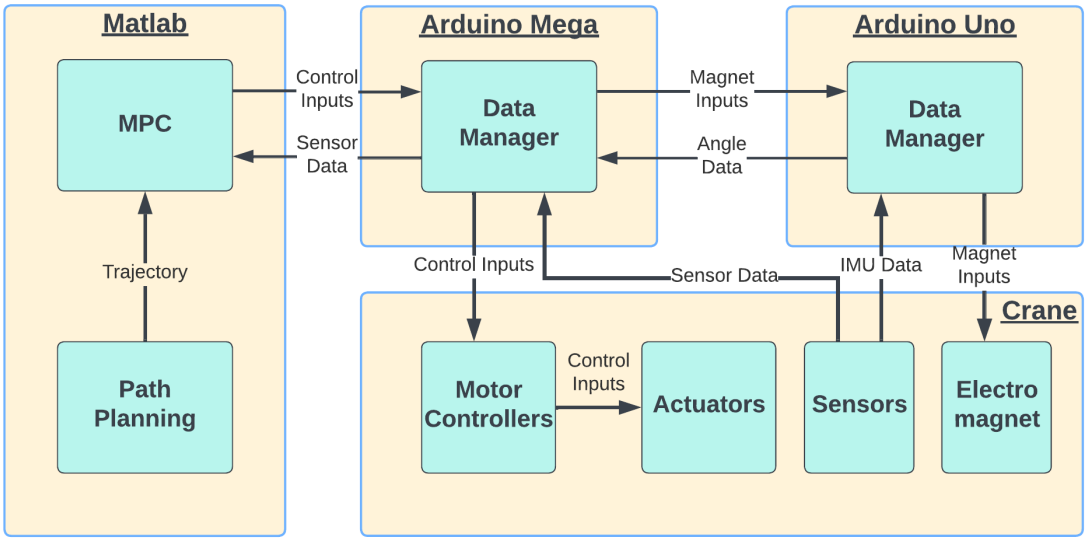


Figure 5.1: System Architecture of an overhead crane managed by a Model Predictive Controller. The crane's trajectory is computed utilising a Potential Field-based path planner, which is sent to the controller. Here, the path is processed and motor inputs are derived. Next, the Arduino program sends the motor inputs to the crane's actuators. Sensors are implemented on the crane, which detected various states of the crane. These, are fed back to Arduino and then back to the MPC.

Figure 5.1 shows the three main groups of the system architecture, Matlab, Arduino and Crane. The first one, Matlab, refers to the programs that are implemented in Matlab and

cannot be executed directly on the crane. The initial phase is to compute a path for the crane to perform. Then, the controller receives the coordinates and derives two signals, which control both motors. The second subsystem is, as it name implies, the program executed in an Arduino Mega. This microcontroller receives the input signals from the MPC and further feeds it to the crane. Between the first and second subsystems there exists data flow in both directions, as the MPC not only sends the input signals but but also needs measurements of the crane's state. This leads to the third group, Crane. It consists of three hardware components, the motor drivers, the actuators and the sensors. The drivers take the input signals, in the form of PWM signals, and transform it into a current, which makes the actuators move the crane. The sensors are tasked with detecting the changes in position of the head of the crane, see **Figure 2.3**.

The following sections will describe the different subsystems mentioned and how these are implemented.

5.1 Path Planning Algorithms

Two algorithms have been implemented, Potential Field and ACO. The theory behind each of them and how the implementation for each of them is will be described in the following subsections.

Potential Field: Path Planning

Potential Fields in Robot Navigation are spaces within which, at each point, the potential energy of obstacles, robots or any object present in the surroundings, are represented. This representation of the potential energy is derived by the attractive and repulsive forces. These forces are the essential elements that makes Potential Fields work in robot navigation and obstacle avoidance. In this section, the equations used to derive both attractive and repulsive forces and how these are utilised to derive a path planning and obstacle avoidance algorithm that can be applied to the gantry crane.

The generation of a potential field starts by assigning a potential to each point on the space. The robot will navigate through this space, starting at the start point, with the highest potential and moving towards the lowest potential at the target point. As mentioned in the above paragraph, two forces are entrusted with defining the potential field, assigning a direction to each point.
The potential energy U at each point is the sum of attractive potentials and repulsive potentials **Equation 5.1**.

$$U(\vec{r}) = U_{attractive}(\vec{r}) + U_{repulsive}(\vec{r}) \tag{5.1}$$

In terms of robot navigation, only the target location will have an attractive potential. Whereas multiple obstacles that may appear in the workspace, generate a repulsive poten-

tial. the robot should maintain distance from the obstacles and move towards the goal.

$$U(\vec{r}) = U_{attractive}(\vec{r}) + \sum_{i=1}^{n} U_{repulsive}(\vec{r}) \tag{5.2}$$

Both the attractive and repulsive aspects of potential fields affect the movement of the robot. These effects can be describe as forces, which have the following relationship with the potential energy:

$$\vec{F}(\vec{r}) = -\nabla U(\vec{r}) \tag{5.3}$$

As shown, the force is equal to the negative gradient of the potential at each point. $F(\vec{r})$ is, same as with $U(\vec{r})$ in **Equation 5.1**, the sum of the attractive forces and repulsive forces. Following the relation ship between potential energy and force, **Equation 5.3**, the following equation can be obtained:

$$\vec{F}(\vec{r}) = -\nabla U_{attractive}(\vec{r}) + \sum_{i=1}^{n} \Big(-\nabla U_{repulsive}(\vec{r}) \Big) \tag{5.4}$$

The gradient $\nabla U(\vec{r})$ can be described in terms of its Cartesian components. Relating this to $\vec{F}(\vec{r})$, the following equation is obtained:

$$\vec{F}(\vec{r}) = -\vec{\nabla} U(\vec{r}) = -\begin{bmatrix} \frac{\partial U}{\partial x} & \frac{\partial U}{\partial y} \end{bmatrix}^T = \begin{bmatrix} F_x & F_y \end{bmatrix} \tag{5.5}$$

By computing the partial derivative of the potential energy $U(\vec{r})$ in terms of x, y, the force constituents on each direction can be derived, namely F_x and F_y.
The computation of the repulsive and attractive forces differ, thus they have to be calculated individually.

Attractive Force: Target

The attractive potential energy of the target can be described linearly as

$$U_{attractive}(r) = k_t \sqrt{(x - x_t)^2 + (y - y_t)^2} \tag{5.6}$$

where k_t is a constant gain dependent on the potential of the target, and r is the distance between the present position of the robot and the target position.
From **Equation 5.5**, the forces on x and y can be isolated into the following:

$$\begin{aligned} F_{tx} &= -\frac{\partial U_{attractive}}{\partial x} = \frac{-k_t(x - x_t)}{\sqrt{(x - x_t)^2 + (y - y_t)^2}} \\ F_{ty} &= -\frac{\partial U_{attractive}}{\partial y} = \frac{-k_t(y - y_t)}{\sqrt{(x - x_t)^2 + (y - y_t)^2}} \end{aligned} \tag{5.7}$$

Figure 5.2 shows what the potential field for the crane would look like when only the attractive force is affecting it. The solution shown is for the specified target, but would be similar to any other target within the workspace.

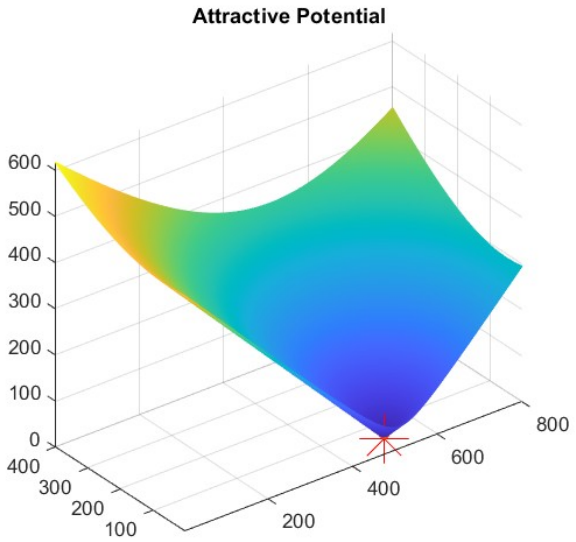


Figure 5.2: Attractive Force. The graph depicts the workspace of the crane in terms of its potential. On this instance, the crane starts at $p(1,1)$ and will slide straight to the target point at (510, 50).

Repulsive Force: Obstacles

The repulsive force, inherent to obstacles in this case, can be computed in a similar manner to the attractive force. The repulsive potential energy can be described as:

$$U_{repulsive}(r) = \frac{k_o}{r} = \frac{k_o}{\sqrt{(x-x_o)^2 + (y-y_o)^2}} \tag{5.8}$$

where k_o is a constant gain that depends on the potential energy of the obstacle(s). Utilising the above equation, the forces on each direction can be computed.

$$\begin{aligned} F_{ox} &= -\frac{\partial U_{repulsive}}{\partial x} = \frac{-k_o(x-x_o)}{[(x-x_o)^2 + (y-y_o)^2]\sqrt{(x-x_t)^2 + (y-y_t)^2}} \\ F_{oy} &= -\frac{\partial U_{repulsive}}{\partial y} = \frac{-k_o(y-y_o)}{[(x-x_o)^2 + (y-y_o)^2]\sqrt{(x-x_o)^2 + (y-y_o)^2}} \end{aligned} \tag{5.9}$$

As multiple objects may appear in the workspace of the crane, the following equation is necessary.

$$F_{ox} = \sum_i F_{ox_i} \qquad F_{oy} = \sum_i F_{oy_i} \tag{5.10}$$

where i refers to the obstacles. An example of plotting the repulsive field in Matlab is shown in **Figure 5.3**.

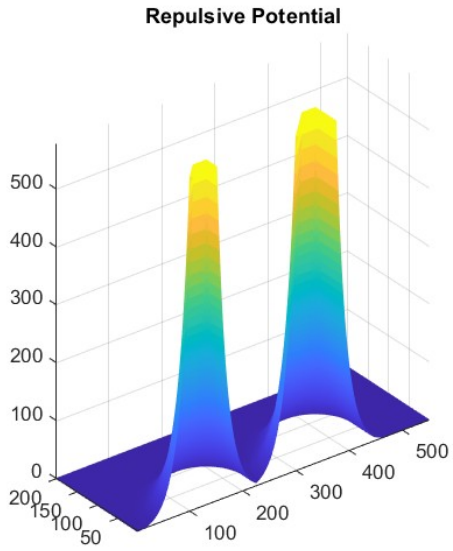


Figure 5.3: Repulsive Force. The graph depicts the repulsive force generated by the two obstacles.

Lastly, the derived forces of both the target and the obstacles can be combined to obtain the complete solution:

$$F_x = F_{tx} + \sum_i F_{ox_i} \qquad F_y = F_{ty} + \sum_i F_{oy_i} \tag{5.11}$$

The last equation needed is the following, which is used to derive the "heading" angle of the robot. The arc-cosine function allows the computation of the mentioned heading-angle, utilising the three forces obtained in the previous step.

$$\alpha = atan2\left(F_y, F_x\right) \tag{5.12}$$

The implementation on Matlab was done with some changes compared to what has been shown until now. **Equation 5.8** has been changed to the following:

$$U_{repulsive}(r) = k_o \cdot \left(\frac{1}{\frac{d}{K+1}} - \frac{1}{d_0}\right)^2 \tag{5.13}$$

where, k_o is a constant gain that depends on the potential energy of the obstacle(s). d is the distance between the obstacle and the actual position of the robot, K is a scaling factor which determines the strength of the repulsion, its effect can be seen in **Figure 5.6**. Lastly, d_0 is a condition added to the equation that determines the limits at which the repulsive force has an effect. Increasing and decreasing this value also strengthens or weakens the repulsive force.

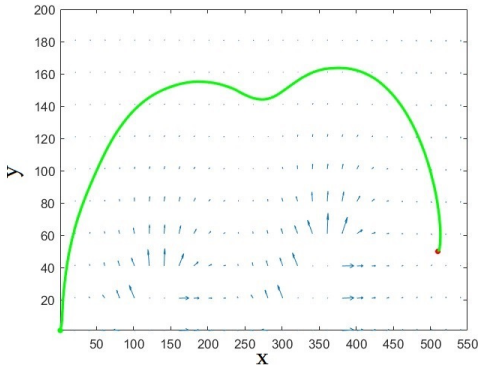


Figure 5.4: $K = 100$

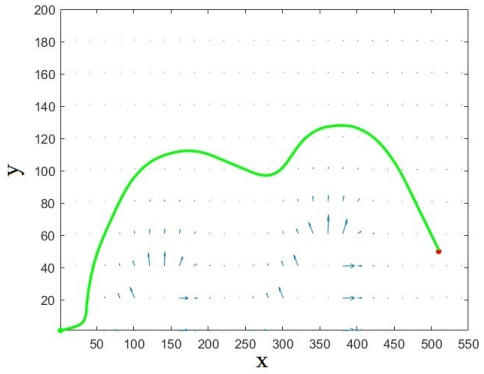


Figure 5.5: $K = 50$

Figure 5.6: As seen in the graphs, an increase in K affects the swells the area affected by the repulsive force. On the other hand, decreasing said value diminishes its effect.

The combination of the attractive and repulsive fields is what determines the path the system will follow. This can be seen in **Figure 5.7**.

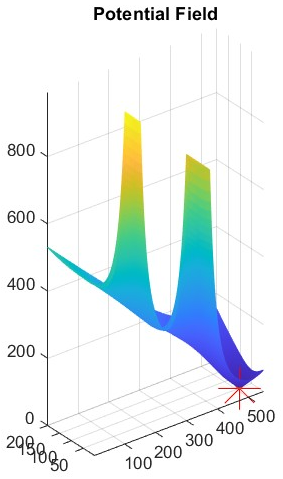


Figure 5.7: Graph depicting the combination of **Figures 5.2** and **5.3**.

The program written in Matlab is described in the following paragraphs. This program has been designed to compute the path and obtain the data necessary for the controller to manage the movement of the crane.

The approach takes as foundation the notion of defining the workspace as a binary image. Each pixel on this image will be assigned either a 1 or 0, being ones those pixel that compose the obstacles. In the following figure, **Figure 5.9**, this idea is represented in both

matrix form and as a binary image.

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	0	0
0	0	0	0	0	0	1	1	0	0
0	0	0	1	1	0	1	1	0	0
0	0	0	1	1	0	1	1	0	0

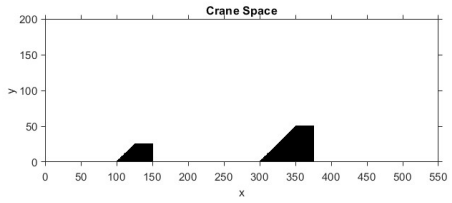


Figure 5.8: Example of a workspace where there are two obstacles.

Figure 5.9: On the left, a simplified matrix version of a workspace where there are two obstacles. On the right, the same workspace is represented as a binary image.

Visually, the objects shown in the images are not the same. On the matrix, the objects are represented as rectangles, 2:2 and 4:2, whereas on the binary image, the obstacles are shown as a combination of a rectangle together with a triangle. This is due to a design choice taken during the development of the program, where the researched disadvantages of potential fields from **Section 2.3**, were taken into consideration. The main problem found for the Potential Field algorithm was its inability to escape local minima. To solve this, the obstacles have been given a semi-trapezoid structure, where the slope on its left is of 45°.

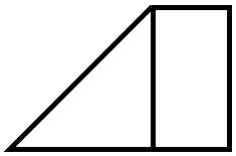


Figure 5.10: Obstacle design. The center rectangle is considered to be the actual object and the adjacent triangle is added to aid the program at not getting stuck in local minima.

Utilising the frame, shown in **Figure 5.10**, for encasing the obstacle increases the area of the repulsive field for each obstacle. The slope added to the object generates a field that naturally forces the system to slide up the side the object.

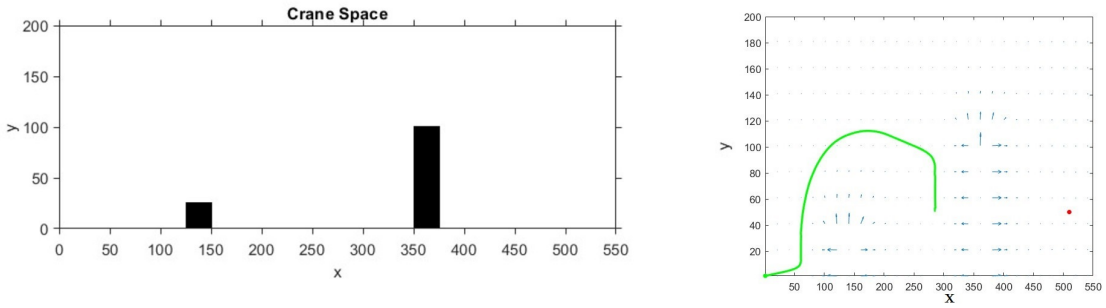


Figure 5.11: The left graph shows the obstacles without any additional area added. As seen on the right side image, this causes the path to get stuck on a local minima and thus, never reaching the target.

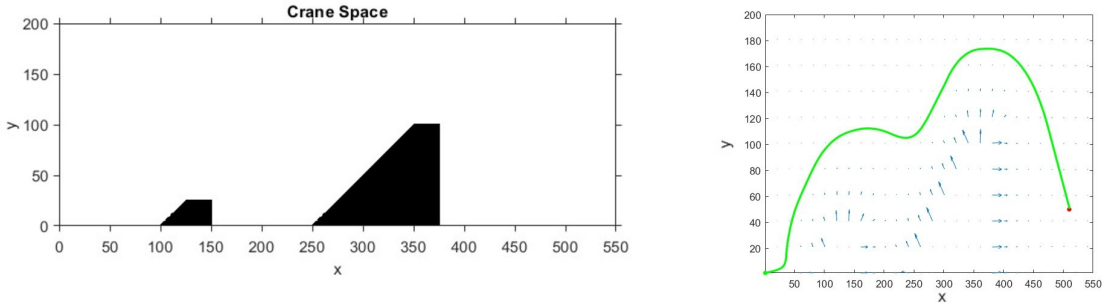


Figure 5.12: The use of a slope on the left side of the obstacle allows for the system to avoid getting stuck on the local minima and reach the target position.

This approach tries to minimise the probability of the path planner getting trapped in local minima. To ensure this technique does perform as desired, testing should be done to it.

After defining the workspace with its obstacles, start and goal positions, the repulsive forces are computed. Firstly, the Euclidean distance transform of the image is computed. The transform contains the distance between all the pixels in the image and their respective nearest obstacle pixel. **Equation 5.13** is then used, and a matrix, `Fo`, of the size of the grid is obtained. This matrix contains the repulsive force at each point on the workspace. In a same matter, the attractive field is derived and a matrix `Ft` is obtained, assigning a value to each pixel on the grid. After that, both `Ft` and `Fo` are added together (`F`) and the path can be derived.

The function `planner.m` takes the force matrix `F`, the start and target coordinates, and outputs the path for the crane. The first step is to derive the negative gradient of `F` using `gradient(-F)` which outputs the `x` and `y` components of said gradient, `gx` and `gy` respectively. These are then used in the main loop, which will loop through the desired number of iterations. `gx` and `gy` used to compute δ, which is the vector that contains the value and direction of the gradient at the current coordinate and iteration. In this case, only the

direction in δ is used. The new coordinate is calculated by adding the direction times a velocity to the current coordinate.
This process is iterated until the distance between actual position and target position is under the desired tolerance value or, until the loop reaches the end of allowed iterations.

Ant Colony Optimisation: Path Planning

Ant Colony Optimisation is a useful tool in Robot Navigation for pathfinding, as the research in **Chapter 2** showed. This approach is based on the natural behaviour of ants when foraging their surroundings. These ants will move around on the ground while leaving some pheromones. If the ant finds a meal, it will backtrack its own steps, updating the pheromone trail it had previously made. The update in pheromones aids other ants find their way towards the target. One main benefit of this approach is the existence of randomness, which make the ants move spontaneously, creating a different path which, if better[1], will encourage other ants to follow it.

Taking the ants behaviour as a basis, a program that finds the shortest path can be designed. In the following paragraphs, the implementation of an ACO path planner will be shown/demonstrated.

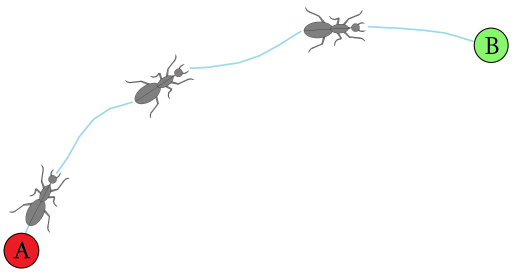


Figure 5.13: First ant starts at A and finds B, leaving a pheromone behind.

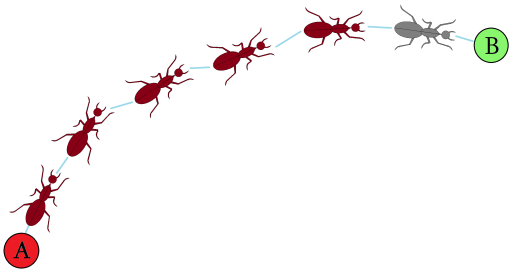


Figure 5.14: Other ants follow the trail left by the first ant, allowing them to go from A to B.

Figure 5.15: Illustration of how ant pheromone works for finding a path.

Figure 5.15 gives an example of the behaviour of six ants. On the left, one ant discovers a target and leaves a trail of pheromones. Then, five other ants find said trail an follow it, to get to the destination.

Now, the implementation will be explained. The first step is to create an environment where the ants will travel. For this project, a node-based grid is utilised. The grid is made of nodes that cover all reachable space. The ants will go from node to node until reaching the target node given by the user. In order to add the presence of obstacles

[1]Shorter, safer, easier to traverse.

to the workspace, some of these node are made unreachable for the ants. The function `createWorkspace` contains the data used to generate the workspace, as well as some additional information that is used for path finding.

Listing 5.1: `workspace` is made of five elements, namely `x`, `y`, `n`, `D`, and `isObstalce`. `x` and `y` state the size of the workspace, and `n` is the number of nodes in the workspace, based on the size of it. The matrix `D` contains the length of the distances between nodes, and lastly, `isObstacle` is used to define where the obstacles are located in the workspace.

```
function workspace = createWorkspace(nrows, ncols, obstacleNodes)
    x = 1:nrows;
    y = 1:ncols;
    [y, x] = meshgrid(y, x);
    x = x(:);
    y = y(:);

    n = numel(x); % Number of nodes.

    D = zeros(n, n); % Matrix of distances between nodes.

    isObstacle = false(n, 1); % Define obstacle nodes.

    isObstacle(obstacleNodes) = true; % Set obstacleNodes to true.

    for i = 1:n - 1

        for j = i + 1:n

            D(i, j) = sqrt((x(i) - x(j))^2 + (y(i) - y(j))^2);

            D(j, i) = D(i, j);

        end

    end

    workspace.n = n;

    workspace.x = x;

    workspace.y = y;

    workspace.isObstacle = isObstacle;

    workspace.D = D;

end
```

The ants are then generated. This is done by creating empty arrays for each ant's path and the cost assigned to each of these. The cost is based on their distance from start to goal,

the lower the distance, the lower cost is assigned to the paths. Ultimately searching for the shortest path. The following code shows the ant generation. the matrix `ant` contains all the ants, with their respective path and cost.

Listing 5.2: Ant generation. the `ant` matrix contains all the ants used and ther respective paths and cost of path.

```
1      emptyAnt.Path = []; % Empty ant path array.
2
3      emptyAnt.Cost = []; % Empty ant cost array.
4
5      ant = repmat(emptyAnt, nAnts, 1); % Matrix containing all ants.
6
7      bestAnt.Cost = inf; % Best ant.
```

Next, the main loop. The ants are "sent" to the workspace in order to find the target. As mentioned, the search works by going from node to node until the target is found. The selection of each new node is based on the probability of the ant going to each of the nodes. This probability is calculated by utilising the following equation.

$$P = \tau^{\alpha} \cdot \eta^{\beta} \tag{5.14}$$

where; P: Probability matrix, τ: Pheromone matrix, α: Pheromone exponential weight, η: Desirability between nodes, and β: Heuristic exponential value. Changing these values modifies the conditions for the selection of nodes. P is the normalised so that the probabilities for each node fall between 0 and 1. Then, a function named `randNode` takes the probability matrix P and outputs, randomly, a node `j`.

Listing 5.3: Firstly, a number between 0 and 1 is obtained. Then, the cumulative sum of the probability matrix is derived. Lastly, the first indice of nonzero value that complies with the condition $r <= C$, is selected.Elements with higher probabilities will have a bigger cumulative sums, i.e., it will have higher probabilities of being selected.

```
1  function j = randNode(prob)
2      r = rand;
3
4      C = cumsum(prob);
5
6      j = find(r ≤ C, 1, 'first');
7
8  end
```

If the value obtained in `j` is not an obstacle, it will be added to the path. The `isObstacle` variable is utilised during this step by comparing `j` to the list of nodes in `isObstacle`, which represent obstacles.

The cost of the updated path is calculated and compared with the general[2] best cost path. The pheromone matrix τ is then updated, by reinforcing the edges that are part of better paths and evaporating the pheromones, such that the system "forgets", over time, the suboptimal edges. The cost mentioned in this section has been defined as the path length.

Listing 5.4: A distance `d` is calculated by adding the distances in the ith and jth elements `D.` into it.

```
function d = PathLength(path, model)
    n = numel(path); % Returns the number of elements in path.

    path = [path path(1)]; % Array of the nodes in a path, starts and ...
        ends at the same node.

    d = 0;

    for i = 1:n

        d = d + model.D(path(i), path(i+1));

    end

end
```

This process is done for all ants and iterations, set before running the program. When the number of iterations is met, the solution with the shortest path is outputted, see **Figure 5.16**.

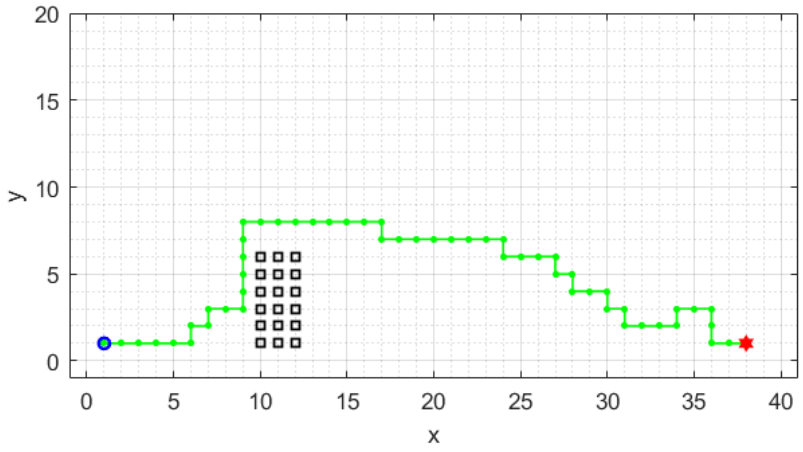


Figure 5.16: Example of a path derived by the ACO planner. The start is at the blue circle and goes to the red hexagram. The black squares simulate an obstacle of size six by three.

[2]Refers to the least costly path between the ants.

5.2 Measurement Unit Normalization

This section will explain some of the normalisation applied to the measurements received from the potentiometers and tachometers of the crane. The initial units that these measurements had did not align with the rest of the solution, therefore, they need to be changed.

5.2.1 Potentiometers

The measurements obtained with the potentiometers are given as a value between 0 and 1024. This makes modelling of the system difficult, and therefore, in this section a conversion from potentiometer reading to m is shown.

To convert from potentiometer unit to meters, the crane was moved every $10cm$ along the x-axis and $3cm$ along the l-axis, separately, and potentiometer readings and manual measurements were recorded. Potentiometer readings were matched to manual measurements in meters.
The matched potentiometer and manual measurements were fitted onto a polynomial utilising a Matlab polynomial fit function, which takes every pair of points and generates a polynomial. This could be used to calculated the position in meters of the crane every time a potentiometer reading is obtained. Instead, since the maximum number of possible potentiometer readings of the crane is known, all the positions in meters can be calculated and a map created. This way, the position in meters can always be obtained. Which is a reduction of computational time compared to calculating the conversion between potentiometer readings and m every time a potentiometer measurement is obtained. In the following figure, **Figure 5.19**, the fitted function for both x and l can be seen.

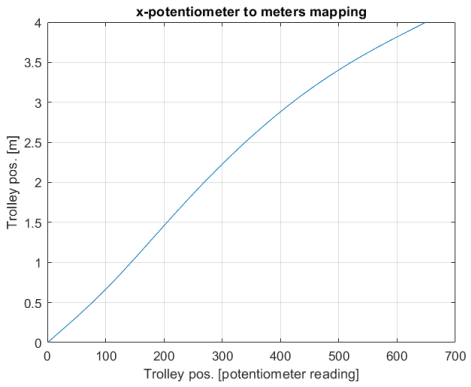


Figure 5.17: Horizontal axis potentiometer readings mapped to meters.

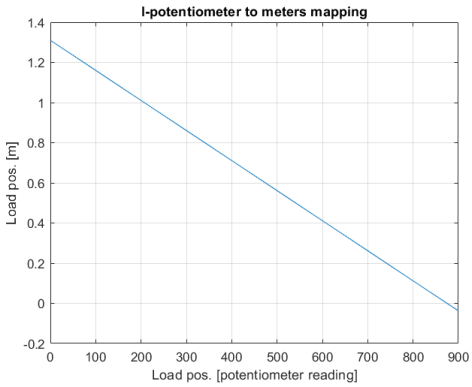


Figure 5.18: Vertical axis potentiometer readings mapped to meters.

Figure 5.19: Mapping of both vertical and horizontal axes potentiometer readings to meters.

As seen in **Figure 5.18**, the mapping is one to one, simplifying the conversion between

the load's position and the potentiometer readings. On the other hand, for the x-axis, **Figure 5.17**, the mapping is more intricate. It is now available for the implementation of the controller on the crane.

5.2.2 Tachometers

Similarly, the digital tachometer signals are also in the range of [0,1024]. In order to convert this into m/s, an open loop test was run for x, and l respectively. The maximum velocity was reached, and the velocity in m/s was calculated as the rate of change of the potentiometer readings. This was done using the average of a span of potentiometer readings at maximum velocity. By comparing the result of this calculation to the tachometer measurement, it was possible to make a conversion from tachometer reading into m/s. The Matlab script was then edited such that all received tachometer measurements from the Arduino are normalized to m/s using the same scaling factor.
The following figure shows the difference between the original tachometer readings and the normalized tachometer readings, which are scaled to m/s. Where the original tachometer readings are represented as a number between [-1,1].

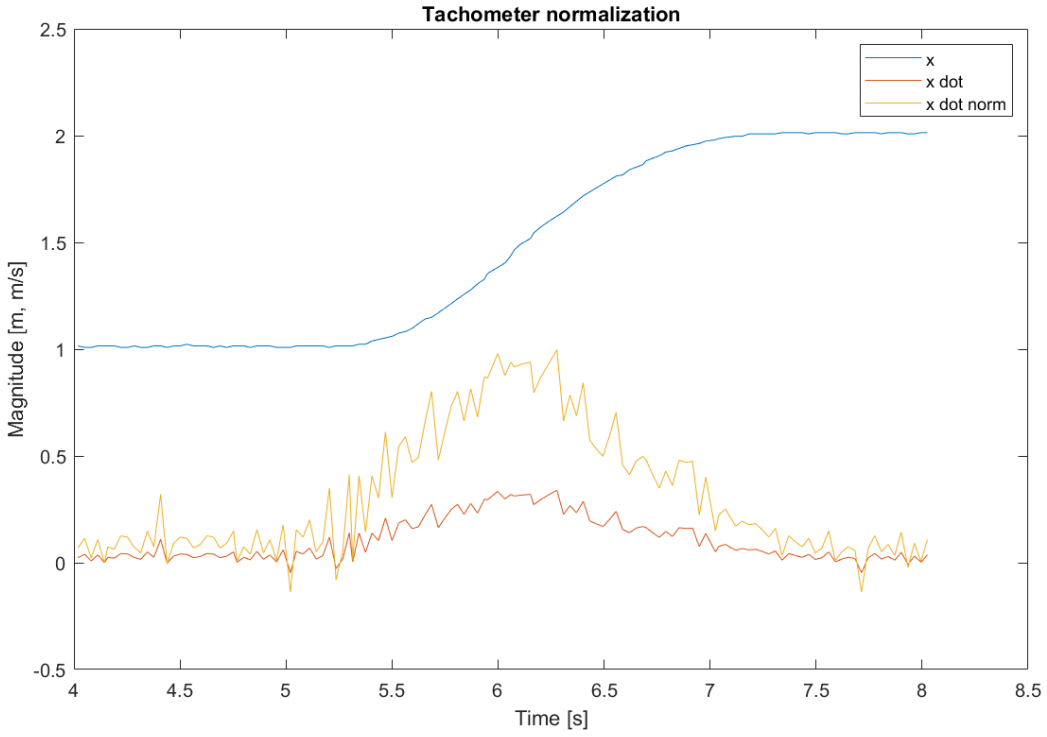


Figure 5.20: This graph shows an open loop test of the physical system, with the trolley position, x, the original tachometer signal, x dot, represented in the range [-1,1] to fit in the graph, and the normalized tachometer readings in m/s, x dot norm

5.3 Model Predictive Control

MPC is a technique utilised in control theory to manage a dynamical system by deriving the optimal solution to the problem at hand. The implementation of such a controller will be explained in this section.

The first step of the implementation has already been shown, namely, modelling the system, **Section 5.4**. The model is a nonlinear system which would require a nonlinear MPC. Such an MPC does exist but requires extensive computational power, as described by Blet et al. [53]. Nevertheless, having a nonlinear system can be beneficial for testing the ability of the MPC at controlling the real system. Linearisation, for this reason, has been deemed as a necessary step for implementing a linear-MPC on the crane.

The steps for designing a Model Predictive Controller will be enumerated in the following list.

1. Discretise the Input signal
2. Define the Objectives and Constraints
3. Define the Cost function
4. State the Prediction Horizon
5. Solve the Optimisation Problem
6. Generate a Control signal
7. Update the MPC based on the output measurements
8. Repeat process

1. Discretise the Input signal: The model derived with help of Computed Torque is continuous-time, but MPC requires it to be discrete.

2. Define the Objectives and Constraints: The crane comes with certain constraints, both due to its physical and digital properties. These constraints have to be added to the MPC so it generates a control signal that takes these limitations into consideration. Some of these constraints are: the allowed voltage per component and the size of the crane. The main Objective for this implementation is to follow a reference trajectory, generated by the path planning program. The path received has to be adjusted so that the crane can follow it. The output signal has to be limited between 10% and 90% duty cycle, as the motors work with a signal between this interval.

3. Define the Cost function: A cost function has to be designed for the MPC. This function will embody the mentioned constraints so that the objective can be met. The goal of the system is to follow a reference path, therefore, the cost function can be designed as a Standard Cost function, for example, the one the MPC Toolbox in Matlab may utilise [54]:

$$J(z_k) = J_y(z_k) + J_u(z_k) + J_{\Delta u}(z_k) + J_\varepsilon(z_k)$$

Output Reference Tracking:

$$J_y(z_k) = \sum_{j=1}^{n_y} \sum_{i=1}^{p} \left(\frac{w_{i,j}^y}{s_j^y} \Big(r_j(k+i|k) - y_j(k+i|k) \Big) \right)^2$$

where k is current control interval, p the prediction horizon, n_y the number output variables, z_k the quadratic programming decision. $y_j(k+i|k)$ is the predicted value of the output at the ith horizon step and $r_j(k+i|k)$ is the reference value for the output at the same ith step. $\frac{w_{i,j}^y}{s_j^y}$ is the ratio of the tuning weight over the scaling element for the output of the plant. The equations for the other three terms that appear in the standard cost function equation, Manipulated Variable Tracking $J_u(z_k)$, Manipulated Variable Move Suppression $J_{\Delta u}(z_k)$ and, Constraint Violation $J_\varepsilon(z_k)$, have a similar variation to the equation for Output Reference Tracking, so it will not be shown.

4. State the Prediction Horizon: A limit for how far in the control process the MPC computes the control signal, has to be set. Longer horizons means more computational power is needed for the MPC to calculate the control action.

5. Solve the Optimisation Problem: The goal of the MPC is to find the optimal control signal that reduces the cost of performing said action while keeping the constrains and objective in mind.

6. Generate a Control signal: At this point, a control signal should have been derived. The signal that will be outputted will depend on the receiving system. In this case, this signal will signify a PWM value for both motors of the crane.

7. Update the MPC based on the output measurements: Utilising the sensors available on the crane, the MPC can be updated using the measurements of the crane's head position.

This process is repeated until the objective has been achieved, in this case, delivering the object to be transported to the target location.

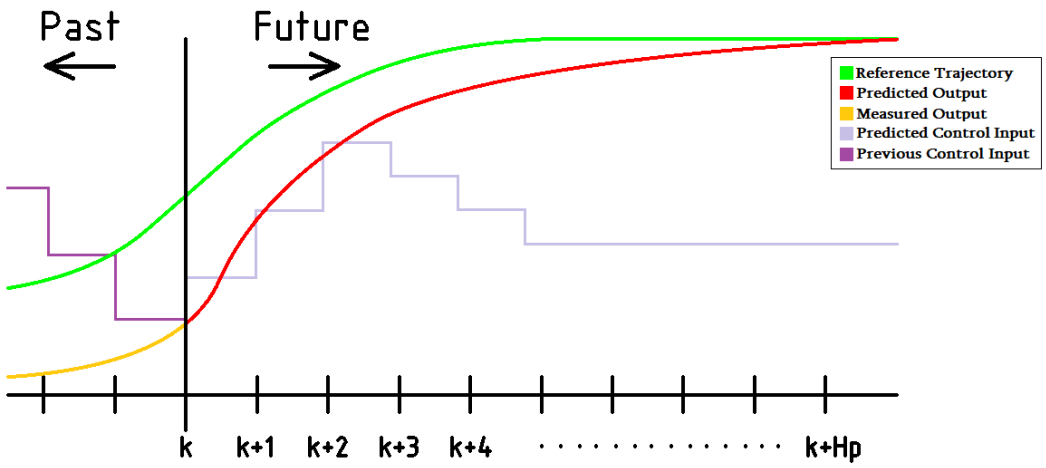


Figure 5.21: Graph depicting the functioning of MPC. The controller takes the measured states of the system and predicts a control signal for the motors that will "try" to follow the reference trajectory. H_p refers to the prediction horizon.

To solve the optimisation problem, a couple of of actions are needed beforehand. Considering the system has been discretised, the following model is utilised, [55]

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) \\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{x}(k) \end{aligned} \tag{5.15}$$

Using the backward difference operator to the system,

$$\left.\begin{aligned} \Delta\tilde{x}(k+1) &= \tilde{x}(k+1) - \tilde{x}(k) \\ \Delta\tilde{u}(k+1) &= \tilde{u}(k+1) - \tilde{u}(k) \end{aligned}\right\} \Longrightarrow \Delta\tilde{x}(k+1) = \boldsymbol{A}\boldsymbol{\Delta}\tilde{x}(k) + \boldsymbol{B}\Delta\tilde{u}(k) \tag{5.16}$$

The terms that have the tilde symbol on top, refer to the future/predicted value of said term.

The same notion can be applied to $\tilde{y}(k)$, where,

$$\Delta\tilde{y}(k+1) = \tilde{y}(k+1) - \tilde{y}(k) = \boldsymbol{C}\tilde{x}(k+1) - \boldsymbol{C}\tilde{x}(k) = \boldsymbol{C}\Delta\tilde{x}(k+1)$$

Introducing $\Delta\tilde{x}(k+1)$ from **Equation 5.16** into $\tilde{y}(k+1)$, allows it to be organised as:

$$\begin{aligned} \tilde{y}(k+1) - \tilde{y}(k) &= \boldsymbol{C}\left(\boldsymbol{A}\Delta\tilde{x}(k) + \boldsymbol{B}\Delta\tilde{u}(k)\right) \\ \tilde{y}(k+1) &= \tilde{y}(k) + \boldsymbol{C}\boldsymbol{A}\Delta\tilde{x}(k) + \boldsymbol{C}\boldsymbol{B}\Delta\tilde{u}(k) \end{aligned} \tag{5.17}$$

Setting the system back together yields

$$\begin{bmatrix} \Delta\tilde{x}(k+1) \\ \tilde{y}(k+1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{0} \\ \boldsymbol{C}\boldsymbol{A} & \boldsymbol{I}_{H_p} \end{bmatrix} \begin{bmatrix} \Delta\tilde{x}(k) \\ \tilde{y}(k) \end{bmatrix} + \begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{C}\boldsymbol{B} \end{bmatrix} \Delta\tilde{u}(k) \tag{5.18}$$

The equation will be simplified as follows,

$$\begin{aligned} &\tilde{y}(k) = \begin{bmatrix} \mathbf{0} & \boldsymbol{I}_{H_p} \end{bmatrix} \begin{bmatrix} \Delta\tilde{x}(k) \\ \tilde{y}(k) \end{bmatrix},\ \tilde{x}_a(k) = \begin{bmatrix} \Delta\tilde{x}(k) \\ \tilde{y}(k) \end{bmatrix},\ \boldsymbol{A}_a = \begin{bmatrix} \boldsymbol{A} & \mathbf{0} \\ \boldsymbol{CA} & \boldsymbol{I}_{H_p} \end{bmatrix}, \\ &\boldsymbol{B}_a = \begin{bmatrix} \boldsymbol{B} \\ \boldsymbol{CB} \end{bmatrix},\ \textit{and}\ \boldsymbol{C}_a \begin{bmatrix} \mathbf{0} & \boldsymbol{I}_{H_p} \end{bmatrix} \end{aligned} \tag{5.19}$$

This is done so that the system can be represented as:

$$\begin{aligned} \tilde{x}_a(k+1) &= \boldsymbol{A}_a\tilde{x}_a(k) + \boldsymbol{B}_a\Delta\tilde{u}_a(k) \\ \tilde{y}(k) &= \boldsymbol{C}_a\tilde{x}_a(k) \end{aligned} \tag{5.20}$$

With the system in this layout, the objective function can be solved. This function tries to derive a control signal for each time step k so that a predicted state can be obtained, which can then predict the system's outputs $\tilde{y}$. These predicted measurements can then be introduced in **Equation 5.20** and the process of computing the control signal can be repeated until reaching H_p. This can be described in matrix form as,

$$\begin{bmatrix} \tilde{x}_a(k+1|k) \\ \tilde{x}_a(k+2|k) \\ \vdots \\ \tilde{x}_a(k+H_p|k) \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_a \\ \boldsymbol{A}_a^2 \\ \vdots \\ \boldsymbol{A}_a^{H_p} \end{bmatrix} \tilde{x}_a(k) + \begin{bmatrix} \boldsymbol{B}_a & & \\ \boldsymbol{A}_a\boldsymbol{B}_a & \boldsymbol{B}_a & \\ \vdots & & \ddots \\ \boldsymbol{A}_a^{H_p-1}\boldsymbol{B} & \dots & \boldsymbol{B}_a \end{bmatrix} \begin{bmatrix} \Delta\tilde{u}(k) \\ \Delta\tilde{u}(k+1) \\ \vdots \\ \Delta\tilde{u}(k+H_p-1) \end{bmatrix} \tag{5.21}$$

After deriving $\tilde{x}_a$, it can be added to:

$$\begin{aligned} \begin{bmatrix} \tilde{y}(k+1|k) \\ \tilde{y}(k+2|k) \\ \vdots \\ \tilde{y}(k+H_p|k) \end{bmatrix} &= \begin{bmatrix} \boldsymbol{C}_a\tilde{x}_a(k+1|k) \\ \boldsymbol{C}_a\tilde{x}_a(k+2|k) \\ \vdots \\ \boldsymbol{C}_a\tilde{x}_a(k+H_p|k) \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{C}_a\boldsymbol{A}_a \\ \boldsymbol{C}_a\boldsymbol{A}_a^2 \\ \vdots \\ \boldsymbol{C}_a\boldsymbol{A}_a^{H_p} \end{bmatrix} \tilde{x}_a(k) + \begin{bmatrix} \boldsymbol{C}_a\boldsymbol{B}_a & & \\ \boldsymbol{C}_a\boldsymbol{A}_a\boldsymbol{B}_a & \boldsymbol{C}_a\boldsymbol{B}_a & \\ \vdots & & \ddots \\ \boldsymbol{C}_a\boldsymbol{A}_a^{H_p-1}\boldsymbol{B} & \dots & \boldsymbol{C}_a\boldsymbol{B}_a \end{bmatrix} \begin{bmatrix} \Delta\tilde{u}(k) \\ \Delta\tilde{u}(k+1) \\ \vdots \\ \Delta\tilde{u}(k+H_p-1) \end{bmatrix} \end{aligned} \tag{5.22}$$

Simplifying the above as $\boldsymbol{Y} = \boldsymbol{W}\tilde{x}_a(k) + \boldsymbol{Z}\Delta\boldsymbol{U}$, it can be utilised to solve the cost function such that the cost of following a reference path gets reduced.

$$\boldsymbol{V}(\Delta\boldsymbol{U}) = \frac{1}{2}(\boldsymbol{r}_{H_p} - \boldsymbol{Y})^T\boldsymbol{Q}(\boldsymbol{r}_{H_p} - \boldsymbol{Y}) + \frac{1}{2}\Delta\boldsymbol{U}^T\boldsymbol{R}\Delta\boldsymbol{U} \tag{5.23}$$

$\boldsymbol{Q}$ and $\boldsymbol{R}$ are weight matrices that have to be fine-tuned. These weights are selected based on the desired objective. Depending on which states/measurements are more valuable for

a certain solution, the weights will be chosen accordingly. $\boldsymbol{r}_{H_p}$ is the reference vector that contains the path that the system must follow.
The next steps will describe how $\Delta\boldsymbol{U}$ is isolated, i.e., derive the control signal from k to $k + H_p - 1$. The first-order necessary condition test is used to check the cost function,

$$\frac{\partial V}{\partial \Delta \boldsymbol{U}} = -(\boldsymbol{r}_{H_p} - \boldsymbol{W}\tilde{x}_a(k) - \boldsymbol{Z}\Delta\boldsymbol{U})^T\boldsymbol{Q}\boldsymbol{Z} + \Delta\boldsymbol{U}^T\boldsymbol{R} \tag{5.24}$$

The equation is then set to zero.

$$\begin{aligned} -\boldsymbol{r}_{H_p}^T\boldsymbol{Q}\boldsymbol{Z} + \tilde{x}_a(k)^T\boldsymbol{W}^T\boldsymbol{Q}\boldsymbol{Z} + \Delta\boldsymbol{U}^T\boldsymbol{Z}^T\boldsymbol{Q}\boldsymbol{Z} + \Delta\boldsymbol{U}^T\boldsymbol{R} = \boldsymbol{0} \\ (\boldsymbol{R} + \boldsymbol{Z}^T\boldsymbol{Q}\boldsymbol{Z})\Delta\boldsymbol{U} = \boldsymbol{Z}^T\boldsymbol{Q}(\boldsymbol{r}_{H_p} - \boldsymbol{W}\tilde{x}_a(k)) \end{aligned} \tag{5.25}$$

$\Delta\boldsymbol{U}$ can be isolated, and is now represented as $\Delta\boldsymbol{U}^*$, thus:

$$\Delta\boldsymbol{U}^* = (\boldsymbol{R} + \boldsymbol{Z}^T\boldsymbol{Q}\boldsymbol{Z})^{-1}\boldsymbol{Z}^T\boldsymbol{Q}(\boldsymbol{r}_{H_p} - \boldsymbol{W}\tilde{x}_a(k)) \tag{5.26}$$

The second derivative test can then be applied, resolving whether $\Delta\boldsymbol{U}^*$ is a strict minimiser of the cost function V or not.

$$\frac{\partial^2 V}{\partial \Delta \boldsymbol{U}^2} = \boldsymbol{R} + \boldsymbol{Z}^T\boldsymbol{Q}\boldsymbol{Z} > 0 \tag{5.27}$$

A point z^* is a strict minimiser of an optimisation problem $min_{z\in\Omega} f(z)$ and, $\exists\epsilon > 0$ such that $f(z^*) < f(z)$
for $z \in \Omega$ $\{z^*\}$ satisfying $||z - z^*|| \leq f(z)$ [56]

Lastly, utilising the equation for $\Delta\boldsymbol{U}^*$, $\Delta\boldsymbol{u}(k)$ can be obtained.

$$\begin{aligned} \Delta\boldsymbol{u}(k) &= \begin{bmatrix}\boldsymbol{I}_m & \boldsymbol{O} & \cdots & \boldsymbol{O}\end{bmatrix}\left(\boldsymbol{R} + \boldsymbol{Z}^T\boldsymbol{Q}\boldsymbol{Z}\right)^{-1}\boldsymbol{Z}^T\boldsymbol{Q}\left(\boldsymbol{r}_{H_p} - \boldsymbol{W}\tilde{x}_a(k)\right) \\ &= \begin{bmatrix}\boldsymbol{I}_m & \boldsymbol{O} & \cdots & \boldsymbol{O}\end{bmatrix}\left(\boldsymbol{R} + \boldsymbol{Z}^T\boldsymbol{Q}\boldsymbol{Z}\right)^{-1}\boldsymbol{Z}^T\boldsymbol{Q}\cdot\boldsymbol{r}_{H_p} \\ &- \begin{bmatrix}\boldsymbol{I}_m & \boldsymbol{O} & \cdots & \boldsymbol{O}\end{bmatrix}\left(\boldsymbol{R} + \boldsymbol{Z}^T\boldsymbol{Q}\boldsymbol{Z}\right)^{-1}\boldsymbol{Z}^T\boldsymbol{Q}\boldsymbol{W}\cdot\begin{bmatrix}\boldsymbol{I}_m \\ \boldsymbol{O}\end{bmatrix}\Delta\tilde{x}(k) \\ &- \begin{bmatrix}\boldsymbol{I}_m & \boldsymbol{O} & \cdots & \boldsymbol{O}\end{bmatrix}\left(\boldsymbol{R} + \boldsymbol{Z}^T\boldsymbol{Q}\boldsymbol{Z}\right)^{-1}\boldsymbol{Z}^T\boldsymbol{Q}\boldsymbol{W}\cdot\begin{bmatrix}\boldsymbol{O} \\ \boldsymbol{I}_{H_p}\end{bmatrix}\Delta\tilde{y}(k) \end{aligned} \tag{5.28}$$

I_m, I_n and I_{H_p} are identity matrices of sizes m,n,H_p (where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$ and $C \in \mathbb{R}^{H_p\times n}$).
Equation 5.28 can be turned into a block diagram that describes the MPC. This can be seen in the following figure, **Figure 5.22**.

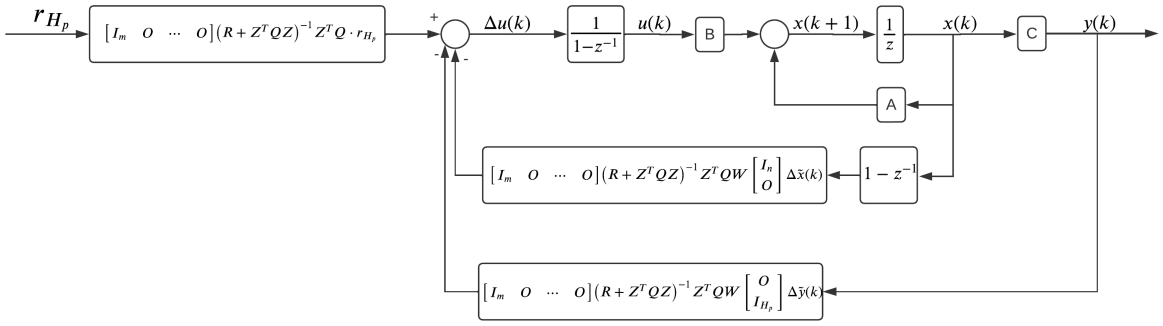


Figure 5.22: Block diagram of an MPC. The terms seen are directly obtained from **Equation 5.28**.

The implementation of the MPC in this project was done utilising the Simulink MPC Designer App, part of Model Predictive Control Toolbox, and then exported as an MPC object to be utilised in the Matlab workspace. This tool simplifies the creation of an MPC, as it almost only requires to introduce different parameters to it for a controller to be generated. First, an MPC block is added to Simulink and connected to the crane's plant, see **Figure 5.23**, where two `MPC` blocks can be seen. Then, the inputs and outputs are defined, together with the desired sample time. For this project, the number of inputs is two and only a single output. The first input, `mo`, contain the measured outputs of the plant. The other input, `ref`, is the reference path that the crane has to follow. The sample time was set to 0.05s. With this, the controller is created. The next steps are to tune the constraints and requirements.

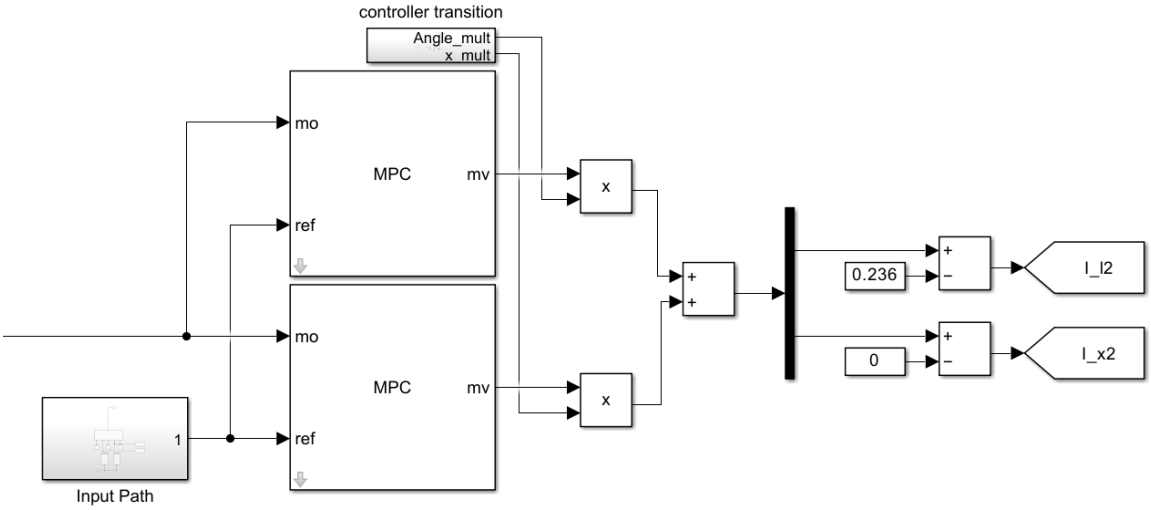


Figure 5.23: Simulink implementation of the MPC.

5.4 Computed Torque

The path planning algorithms will deliver a course which the crane will follow. For this to happen, the solution needs to have a controller that can manage the different actuators and sensors for the task to be performed within certain requirements. The controller will receive a path and translate the information into inputs which the actuators can understand. To be capable of deriving the correct conversion between path data (coordinates) and actuator data (current), the model of the system is needed, as described in **Section 3.3**. One technique which can be utilised is linearisation, which, as described in the previous section, could reduce the computational power required by the MPC to control the crane. One approach to linearise a system, is Computed Torque, which will be utilised in this project.

Dynamic systems can be described as,

$$M(z) \cdot \ddot{z} + C(\dot{z}, z)\dot{z} + V(z)\dot{z} + G(z) = \tau \tag{5.29}$$

where, $M(z)$ is the mass matrix of the system, C is the Coriolis or Centripetal Matrix, V is the matrix that contains the friction coefficients, G is the gravity vector and lastly, τ is the input matrix. These matrices describe the system and are obtained from the equations of motions in **Equations 3.25**, **3.26**, and **3.27**.

The final form of the mentioned matrices can be seen in **Appendix D**.

Computed-torque control is used to derive a linear closed loop system based on the states of said system. The linearisation starts with deriving a control input that cancels the nonlinear terms that appear in the model, **Equation 5.29**, so that a direct relationship between input signal and output can be obtained. This can be done by rewriting the equation into:

$$\ddot{z} = M(z)^{-1} \cdot [\tau - C(\dot{z}, z)\dot{z} - V(z)\dot{z} - G(z)] \tag{5.30}$$

The following step is to derive an input τ that can compensate for the nonlinear terms, namely $M(z)$, $C(\dot{z}, z)\dot{z}$, $V(z)\dot{z}$, and $G(z)$.

$$\tau = \tilde{M}(z) \cdot \Lambda + \tilde{C}(\dot{z}, z)\dot{z} + \tilde{V}(z)\dot{z} + \tilde{G}(z) \mid \Lambda = \ddot{z}_d + k_{MPC} \cdot e \tag{5.31}$$

where $e(t) = z_d(t) - z(t)$ is the tracking error. This error is the difference in position between what has been inputted into the crane and the measured position. And, the matrices containing a tilde refer to the compensation matrices which are used to compensate the system of the nonlinear terms.

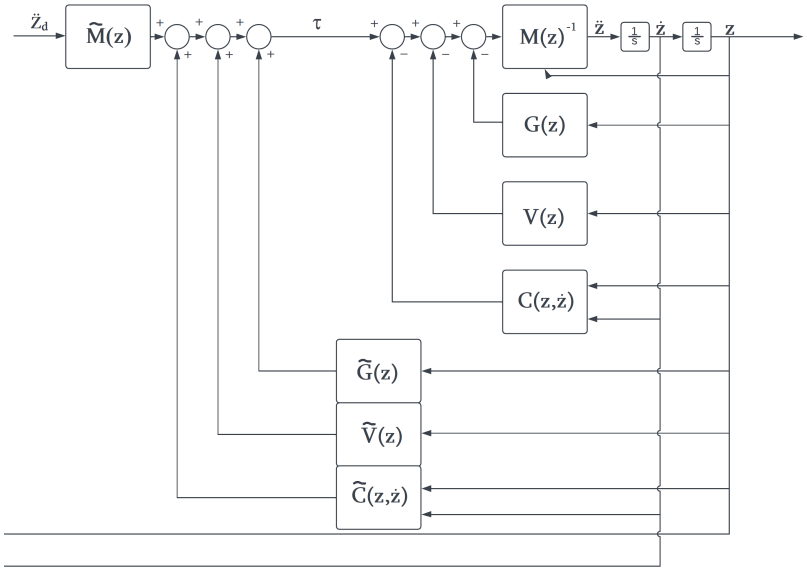


Figure 5.24: Block diagram of the system with a computed torque control that linearises it. The input $\ddot{Z}$ is obtained from a controller which will be described later on the report. The two lines at the bottom of the image, which depict the measured position and velocity of the crane, are connected to the controller which, for the same reason, will not be showed until a later section.

A block diagram of the system with computed-torque applied to it can be seen in **Figure 5.24**, depicting the operational principle of computed-torque. As it can be seen, the matrices $\tilde{M}(z)$, $\tilde{C}(z, \dot{z})$, $\tilde{V}(z)$ and $\tilde{G}(z)$ are inputted into the plant where they alleviate the effects of the non-linearities of the model, thus obtaining an almost linear system.

The implementation of this method has been done in Simulink and utilising the services it provides. One main different from the theory presented in this chapter is the use of linear equations to perform computed torque control, compared to the use of matrices. One example of this is shown in the following figure.

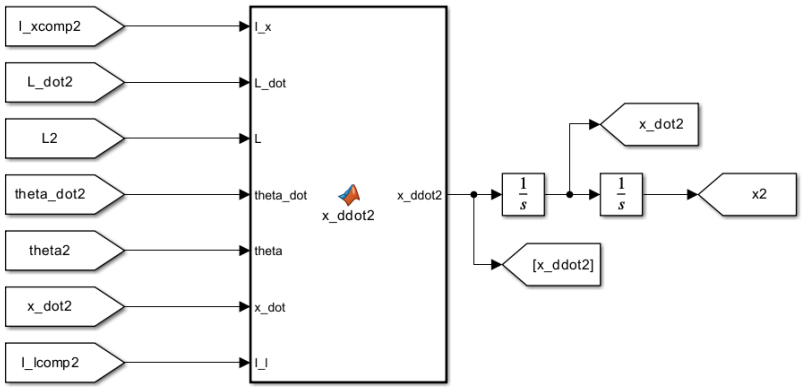


Figure 5.25: Snippet of the computed-torque Simulink diagram where $\ddot{x}$ is being treated. The `function` block `x_ddot2` includes the equation of motion for $\ddot{x}$. To it, the input signal `I_xcomp2` is the computed torque signal, depicted as τ in **Figure 5.24**. The mentioned input signal contains the necessary computation to linearise the effect of the mass, Coriolis, Friction and gravity matrices.

In **Figure 5.25**, the Simulink diagram of the computed-torque implementation on the crane can be seen for $\ddot{x}$. For the other variables, a similar implementation is applied. The output for x of the system after computed-torque can be seen in **Figure 5.26**.

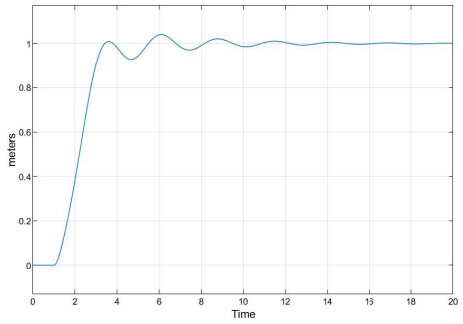


Figure 5.26: Output of the position of the crane. The system has was set to reach an x position of one meter. As seen, it reaches the destination after 3.4 seconds, but has difficulties stabilising. The stabilisation problem could be solved with the implementation of a controller.

5.5 Taylor series Linearisation

The system designed in this project is inherently non-linear. This aspect, as mentioned in previous sections, increases the complexity of designing a control system capable of managing the crane. In **Section 5.4**, a solution that compensates for the non-linearities of the system was designed. The outcome from this implementation exposed a drawback on this project. This is because of the combination of Computed Torque linearisation with

the model predictive controller. It was made clear that computed torque derived a system where the input was 1:1 with the output, the controller always deems the system capable of following its instructions. This oversimplified the solution and is not true for the real crane, thus this approach was deemed as insufficient.

The linear system calculated with computed torque was tested together with a simple MPC created with the built-in MPC generator from Matlab. The results were not completely unusable, but could not be introduces into the crane itself and only performed as desired in a simulated environment. For this reason, another approach for linearising the system was researched, namely Taylor Series Linearisation.

With knowledge regarding the system, the following two block diagrams, **Figures 5.27** and **5.28**, can be defined. The system is divided into two diagrams as x and l are independent from each other.

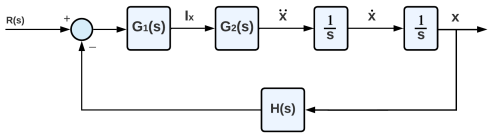


Figure 5.27: Block diagram of the system for x-position where θ has no effect. $R(s)$ is the reference or desired x-position. $G_1(s)$ is the control transfer function and $G_2(s)$, the plant's transfer function. Lastly, $H_(s)$ is the feedback block.

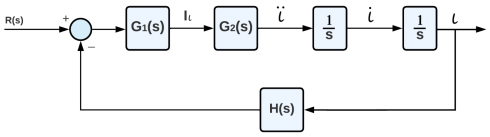


Figure 5.28: Block diagram of the system for l-position.$R(s)$ is the reference or desired l-position. $G_1(s)$ is the control transfer function and $G_2(s)$, the plant's transfer function. Lastly, $H_(s)$ is the feedback block.

Figure 5.29: Block diagrams depicting the control loops for both x- and l-positions.

As seen in the block diagrams for x and l, θ does not appear. In reality, **Figure 5.27** should contain θ as, the equation of motion for both variables show, θ is dependent on x. This, together with the inability to directly control θ in this system, requires the above diagram for x to include an inner loop that adds a θ control. This can be seen in **Figure 5.30**. The block diagram for l remains the same as shown.

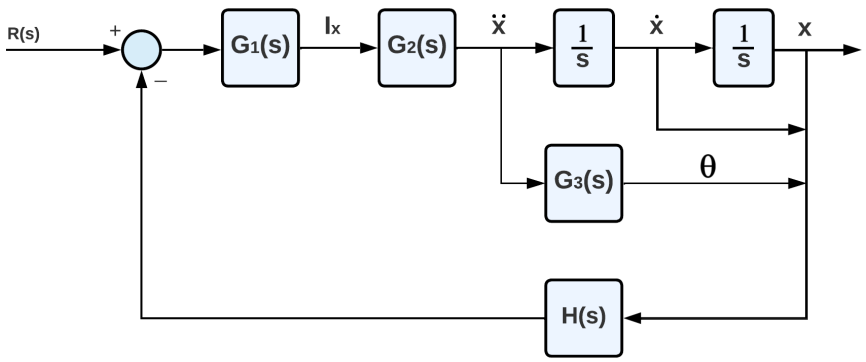


Figure 5.30: Block diagram of the system for x-position influenced by θ. This block diagram is the same as **Figure 5.27** but contains an inner loop that includes θ. The inner loops works in a similar matter to the on showed in the previous figure. $R(s)$ is the reference or desired variables. $G_1(s)$ is the controller , in this case, the MPC. $G_2(s)$ is the system's plant. $G_3(s)$ is the pendulum's transfer function, which shows the output θ. $H(s)$ is the feedback transfer function.

The diagrams depicted in the previous figures will be utilised in a later step of the linearisation process with Taylor series, for defining the transfer functions necessary to complete the system. For example, the transfer function relating I_x to θ.

For the purpose of implementing a linear MPC, the current non-linear system will be linearised. This will be achieved using first order Taylor series approximation, given by $f(z_p + \hat{z})$, around an equilibrium point, described by z_p, where l_p is the current wire length.

$$f(z_p + \hat{z}) \approx f(z_p) + \nabla f(z_p)^T \hat{z} \tag{5.32}$$

$$\hat{z} = \begin{bmatrix} \ddot{\theta} & \dot{\theta} & \theta & \ddot{l} & \dot{l} & l & \ddot{x} & \dot{x} & I_x & I_l \end{bmatrix}^T \tag{5.33}$$

$$z_p = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & l_p & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{5.34}$$

Equation 5.32 has to be calculated for the three variables of the system, x, l and θ. The process of obtaining each of the linearised equations of motion will be shown step by step in the following subsections.

Linearisation of $\ddot{x}$

The equation for the first order Taylor series approximation of $\ddot{x}$ is shown below,

$$\ddot{x}(z_p + \hat{z}) = \ddot{x}(z_p) + \nabla \ddot{x}(z_p)^T \hat{z} \tag{5.35}$$

As a reminder of the current nonlinear equation for $\ddot{x}$, its equation can be seen next,

$$\ddot{x}_t = \frac{1}{m_t + m_l + \frac{J_x}{r_x^2}} \left(\frac{K_{ex} I_x}{r_x} - B_x \dot{x}_t - m_l \ddot{l} sin(\theta) - 2 m_l \dot{l} \dot{\theta} cos(\theta) - m_l l \ddot{\theta} cos(\theta) + m_l l \dot{\theta}^2 sin(\theta) \right) \tag{5.36}$$

Using first order Taylor series approximation, the gradient of the **Equation 5.36** can be found,

$$\nabla \ddot{x}(\hat{z}) = \begin{bmatrix} \frac{\partial}{\partial \ddot{\theta}} \ddot{x} \\ \frac{\partial}{\partial \dot{\theta}} \ddot{x} \\ \frac{\partial}{\partial \theta} \ddot{x} \\ \frac{\partial}{\partial \ddot{l}} \ddot{x} \\ \frac{\partial}{\partial \dot{l}} \ddot{x} \\ \frac{\partial}{\partial l} \ddot{x} \\ \frac{\partial}{\partial \ddot{x}} \ddot{x} \\ \frac{\partial}{\partial \dot{x}} \ddot{x} \\ \frac{\partial}{\partial I_x} \ddot{x} \\ \frac{\partial}{\partial I_l} \ddot{x} \end{bmatrix} = \begin{bmatrix} -m_l l cos(\theta) \\ -2 m_l \dot{l} cos(\theta) + 2 m_l l \dot{\theta} sin(\theta) \\ m_l \big(cos(\theta)(l\dot{\theta}^2 - \ddot{l}) + sin(\theta)(2\dot{l}\dot{\theta} + l\ddot{\theta}) \big) \\ -m_l sin(\theta) \\ -2 m_l \dot{\theta} cos(\theta) \\ -m_l \ddot{\theta} cos(\theta) + m_l \dot{\theta}^2 sin(\theta) \\ 0 \\ -B_x \\ \frac{K_{ex}}{r_x} \\ 0 \end{bmatrix} \frac{1}{m_t + m_l + \frac{J_x}{r_x^2}} \tag{5.37}$$

Both **Equations 5.36** and **5.37** are evaluated at the equilibrium point,

$$\nabla \ddot{x}(z_p) = \begin{bmatrix} -m_l l_p & 0 & 0 & 0 & 0 & 0 & 0 & -B_x & \frac{K_{ex}}{r_x} & 0 \end{bmatrix}^T \tag{5.38}$$

$$\ddot{x}(z_p) = 0 \tag{5.39}$$

These are now substituted into **Equation 5.40**,

$$\ddot{x}(z_p + \hat{z}) = 0 + \frac{-m_l l_p \ddot{\theta} - B_x \dot{x} + \frac{I_x K_{ex}}{r_x}}{m_t + m_l + \frac{J_x}{r_x^2}} \tag{5.40}$$

Linearisation of $\ddot{l}$

For $\ddot{l}$, the following equations needs to be solved,

$$\ddot{l}(z_p + \hat{z}) = \ddot{l}(z_p) + \nabla \ddot{l}(z_p)^T \hat{z} \tag{5.41}$$

The current nonlinear equation for $\ddot{l}$ is shown below.

$$\ddot{l} = \frac{1}{m_l + \frac{4 J_l}{r_l^2}} \left(\frac{K_{el} I_l}{r_l} - B_l \dot{l} + m_l l \dot{\theta}^2 + m_l g cos(\theta) - m_l \ddot{x}_t sin(\theta) \right) \tag{5.42}$$

Using first order Taylor series approximation, the gradient of the **Equation 5.42** can be found,

$$\nabla \ddot{l}(\hat{z}) = \begin{bmatrix} \frac{\partial}{\partial \theta}\ddot{l} \\ \frac{\partial}{\partial \dot{\theta}}\ddot{l} \\ \frac{\partial}{\partial \theta}\ddot{l} \\ \frac{\partial}{\partial l}\ddot{l} \\ \frac{\partial}{\partial \dot{l}}\ddot{l} \\ \frac{\partial}{\partial l}\ddot{l} \\ \frac{\partial}{\partial \ddot{x}}\ddot{l} \\ \frac{\partial}{\partial \dot{x}}\ddot{l} \\ \frac{\partial}{\partial I_x}\ddot{l} \\ \frac{\partial}{\partial I_l}\ddot{l} \end{bmatrix} = \begin{bmatrix} 0 \\ 2m_l l\dot{\theta} \\ -m_l \ddot{x} cos(\theta) - m_l g sin(\theta) \\ 0 \\ -B_l \\ m_l \theta^2 \\ -m_l sin(\theta) \\ 0 \\ 0 \\ \frac{K_{el}}{r_l} \end{bmatrix} \frac{1}{m_l + \frac{4J_l}{r_l^2}} \tag{5.43}$$

Equations 5.42 and **5.43** are now evaluated at the equilibrium point,

$$\nabla \ddot{l}(z_p) = \frac{1}{m_l + \frac{4J_x}{r_x^2}} \begin{bmatrix} 0 & 0 & 0 & 0 & -B_l & 0 & 0 & 0 & 0 & \frac{2k_{el}}{r_l} \end{bmatrix}^T \tag{5.44}$$

$$\ddot{l}(z_p) = \frac{m_l g}{m_l + \frac{4J_l}{r_l^2}} \tag{5.45}$$

The two previous equations, **Equations 5.44** and **5.45**, are now substituted into **Equation 5.46**,

$$\ddot{l}(z_p + \hat{z}) \approx \frac{1}{m_l + \frac{4J_l}{r_l^2}} \left(m_l g - B_l \dot{l} + \frac{2k_{el} I_l}{r_l} \right) \tag{5.46}$$

Linearisation of $\ddot{\theta}$

Lastly, the Taylor series equation for $\ddot{\theta}$ is,

$$\ddot{\theta}(z_p + \hat{z}) = \ddot{\theta}(z_p) + \nabla \ddot{\theta}(z_p)^T \hat{z} \tag{5.47}$$

Solved using current nonlinear equation for $\ddot{\theta}$ shown below.

$$\ddot{\theta} = \frac{1}{l} \left(-2\dot{l}\dot{\theta} - \ddot{x} cos(\theta) - g sin(\theta) \right) \tag{5.48}$$

Using first order Taylor series approximation, the gradient of the **Equation 5.48** can be found,

$$\nabla\ddot{\theta}(\hat{z}) = \begin{bmatrix} \frac{\partial}{\partial\theta}\ddot{\theta} \\ \frac{\partial}{\partial\dot{\theta}}\ddot{\theta} \\ \frac{\partial}{\partial\theta}\ddot{\theta} \\ \frac{\partial}{\partial l}\ddot{\theta} \\ \frac{\partial}{\partial\dot{l}}\ddot{\theta} \\ \frac{\partial}{\partial\ddot{l}}\ddot{\theta} \\ \frac{\partial}{\partial\ddot{x}}\ddot{\theta} \\ \frac{\partial}{\partial\dot{x}}\ddot{\theta} \\ \frac{\partial}{\partial I_x}\ddot{\theta} \\ \frac{\partial}{\partial I_l}\ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 \\ -2\dot{l} \\ \ddot{x}sin(\theta) - gcos(\theta) \\ 0 \\ -2\dot{\theta} \\ \frac{-1}{l}\left(-2\dot{l}\dot{\theta} - \ddot{x}cos(\theta) - gsin(\theta)\right) \\ -cos(\theta) \\ 0 \\ 0 \\ 0 \end{bmatrix} \frac{1}{l} \tag{5.49}$$

Equations 5.48 and **5.49** are evaluated at the equilibrium point,

$$\nabla\ddot{\theta}(z_p) = \frac{1}{l_p}\begin{bmatrix} 0 & 0 & -g & 0 & 0 & 0 & -1 & 0 & 0 & 0 \end{bmatrix}^T \tag{5.50}$$

$$\ddot{\theta}(z_p) = 0 \tag{5.51}$$

The two previous equations, **Equations 5.50** and **5.51**, are now substituted into **Equation 5.52**,

$$\ddot{\theta}(z_p + \hat{z}) \approx \frac{1}{l_p}\left(-g\theta - \ddot{x}\right) \tag{5.52}$$

Laplace Transform

The linearised equations of motion have now been computed. As mentioned before, it can now be applied the Laplace transform on them to obtain the transfer functions that complete the system. Converting the linearised system into transfer functions will be necessary to construct the linear MPC that will be tested and introduced into the solution.

To find the transfer function, first take the Laplace Transform of the differential equations, **Equation 5.53** (with zero initial conditions).

$$\begin{aligned} \ddot{x}(z_p + \hat{z}) &\approx \frac{-m_l l_p\ddot{\theta} - B_x\dot{x} + \frac{I_x K_{ex}}{r_x}}{m_t + m_l + \frac{J_x}{r_x^2}} \\ \ddot{l}(z_p + \hat{z}) &\approx \frac{1}{m_l + \frac{4J_l}{r_l^2}}\left(m_l g - B_l\dot{l} + \frac{2k_{el}I_l}{r_l}\right) \\ \ddot{\theta}(z_p + \hat{z}) &\approx \frac{1}{l_p}\left(-g\theta - \ddot{x}\right) \end{aligned} \tag{5.53}$$

The transfer function is the ratio of output to input, for example, for the $\ddot{l}$ equation, the transfer function is $\mathcal{L}\left(\frac{\ddot{l}}{I_l}\right)$.

The following equation contains a constant term $m_l g$, making it an affine equation. This term will be removed during the Laplace transform calculations as it simplifies the mathematical operations needed to perform Laplace transform. It can be later added when simulating the system or when controlling the real crane.

Separate $\ddot{l}$ equation into input and output components.

$$\begin{aligned} \ddot{l} &= \frac{-B_l\dot{l} + \frac{2k_{el}I_l}{r_l}}{m_l + \frac{4J_l}{r_l^2}} \\ \left(m_l + \frac{4J_l}{r_l^2}\right)\ddot{l} + B_l\dot{l} &= \frac{2k_{el}}{r_l}I_l \end{aligned} \tag{5.54}$$

Now, calculate Laplace transform:

$$\begin{aligned} \mathcal{L}\left(\left(m_l + \frac{4J_l}{r_l^2}\right)s^2 + B_l s\right) &= \mathcal{L}\left(\frac{2k_{el}}{r_l}\right) \\ G_{l1} = \mathcal{L}\left(\frac{\ddot{l}}{I_l}\right) &= \frac{2k_{el}}{r_l}\frac{1}{s\left(\left(m_l + \frac{4J_l}{r_l^2}\right)s + B_l\right)} \end{aligned} \tag{5.55}$$

For the $\ddot{\theta}$ equation,

$$\begin{aligned} \ddot{\theta} &= -\frac{g\theta}{l_p} - \frac{\ddot{x}}{l_p} \\ \ddot{\theta} + gl_p^{-1}\cdot\theta &= -l_p^{-1}\cdot\ddot{x} \end{aligned} \tag{5.56}$$

Then, the following Laplace transformation $\mathcal{L}\left(\frac{\ddot{\theta}}{\ddot{x}}\right)$, is computed.

$$\begin{aligned} \mathcal{L}\left(\ddot{\theta} + gl_p^{-1}\cdot\theta\right) &= \mathcal{L}\left(-l_p^{-1}\cdot\ddot{x}\right) \\ \mathcal{L}\left(\frac{\ddot{\theta}}{\ddot{x}}\right) &= \frac{-s^2 l_p^{-1}}{s^2 + l_p^{-1}g} \end{aligned} \tag{5.57}$$

The same is done for the $\ddot{x}$ equation, and $\mathcal{L}\left(\frac{\ddot{x}}{I_x}\right)$ is obtained. The resulting Laplace transform is:

$$G_{x1} = \mathcal{L}\left(\frac{\ddot{x}}{I_x}\right) = \frac{k_{ex}}{r_x}\frac{s^3 l_p + sg}{s^3 l_p\left(m_t + \frac{I_x}{r_x^2}\right) + s^2 B_x l_p + s\left(m_t + m_l + \frac{I_x}{r_x^2}\right)g + B_x g} \tag{5.58}$$

G_{x1} is, in itself, a transfer function that appears in the block diagram shown in **Figure 5.30** as $G_2(s)$. It has instead been named G_{x1}, i.e., two more transfer functions that derive from $\ddot{x}$ appear in said figure, namely G_{x2} for the integration from $\ddot{x}$ to $\dot{x}$ and G_{x3}, which shows the double integration between $\ddot{x}$ and x. These are shown in the following equations, which depict the relationship between the input I_x and the outputs $\dot{x}$ and x, needed to fit the linear model to the designed MPC.

$$G_{x2} = \mathcal{L}\left(\frac{\dot{x}}{I_x}\right) = \mathcal{L}\left(\frac{\ddot{x}}{I_x}\frac{1}{s}\right) = \frac{k_{ex}}{r_x}\frac{s^2 l_p + g}{s^3 l_p\left(m_t + \frac{J_x}{r_x^2}\right) + s^2 B_x l_p + s\left(m_t + m_l + \frac{J_x}{r_x^2}\right)g + B_x g} \tag{5.59}$$

$$G_{x3} = \mathcal{L}\left(\frac{x}{I_x}\right) = \mathcal{L}\left(\frac{\ddot{x}}{I_x}\frac{1}{s^2}\right) = \frac{k_{ex}}{r_x}\frac{s^2 l_p + g}{s^4 l_p\left(m_t + \frac{J_x}{r_x^2}\right) + s^3 B_x l_p + s^2\left(m_t + m_l + \frac{J_x}{r_x^2}\right)g + s B_x g} \tag{5.60}$$

To obtain G_3 (**Figure 5.30**), the equation relating $\ddot{\theta}$ and $\ddot{x}$, **Equation 5.57**, can be used. This equation relates accelerations whereas the final result is needed as positions. This can be solved by adding a second degree integrator that reduces $\ddot{\theta}$ to θ:

$$\begin{aligned} G_3 &= \mathcal{L}\left(\frac{\theta}{I_x}\right) = \mathcal{L}\left(\frac{\ddot{x}}{I_x}\frac{\ddot{\theta}}{\ddot{x}}\frac{1}{s^2}\right) \\ &= \frac{k_{ex}}{r_x}\frac{-s}{s^3 l_p\left(m_t + m_l + \frac{J_x}{r_x^2} - m_l\right) + s^2 B_x l_p + s g\left(m_t + m_l + \frac{J_x}{r_x^2}\right) + B_x g} \end{aligned} \tag{5.61}$$

The system contains an additional variable, l. The transfer function for this part of the system can be derived independently from the other two. The block diagram for l can be seen depicted in **Figure 5.28**. As with the previous block diagram, l also includes $\ddot{l}$ and $\dot{l}$. The first one has already been computed, **Equation 5.55**. $\dot{l}$ and l can be seen in the following two equations:

$$G_{l2} = \mathcal{L}\left(\frac{\dot{l}}{I_l}\right) = \mathcal{L}\left(\frac{\ddot{l}}{I_l}\frac{1}{s}\right) = \frac{2k_{el}}{r_l}\frac{1}{s\left(\left(m_l + \frac{4J_l}{r_l^2}\right)s + B_l\right)}\frac{1}{s} = \frac{2k_{el}}{r_l}\frac{1}{s^3 m_l + s^3\frac{4J_l}{r_l^2} + s^2 B_l} \tag{5.62}$$

$$G_{l3} = \mathcal{L}\left(\frac{l}{I_l}\right) = \mathcal{L}\left(\frac{\ddot{l}}{I_l}\frac{1}{s^2}\right) = \frac{2k_{el}}{r_l}\frac{1}{s\left(\left(m_l + \frac{4J_l}{r_l^2}\right)s + B_l\right)}\frac{1}{s^2} = \frac{2k_{el}}{r_l}\frac{1}{s^4 m_l + s^4\frac{4J_l}{r_l^2} + s^3 B_l} \tag{5.63}$$

At this point, the transfer functions needed for the control system have been obtained.

Matlab Implementation

Lastly, the translation from the mathematical equations, shown in this section, to the Matlab implementation will be described.

The goal of utilising the Taylor/Laplace approach is to derive the transfer functions that describe the system, which is comprised of two block diagrams, **Figures 5.30** and **5.28**. The transfer functions shown in the mentioned diagrams have been calculated and the final control loop for x and θ can be seen in **Figure 5.31**, for l in **Figure 5.32**.

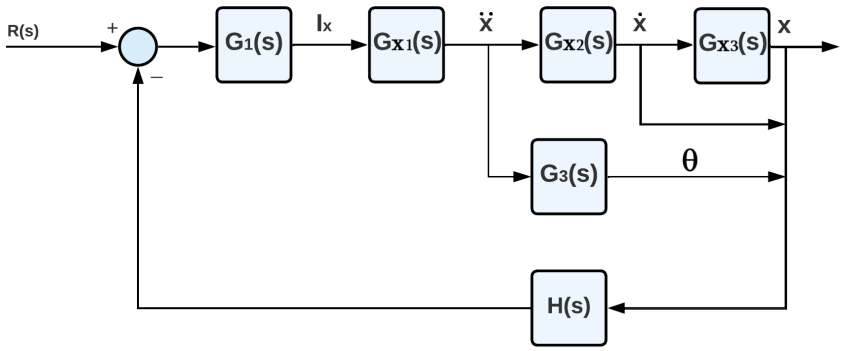


Figure 5.31: Same block diagram as depicted in **Figure 5.30** but with the names given after the mathematical computations shown in this section.

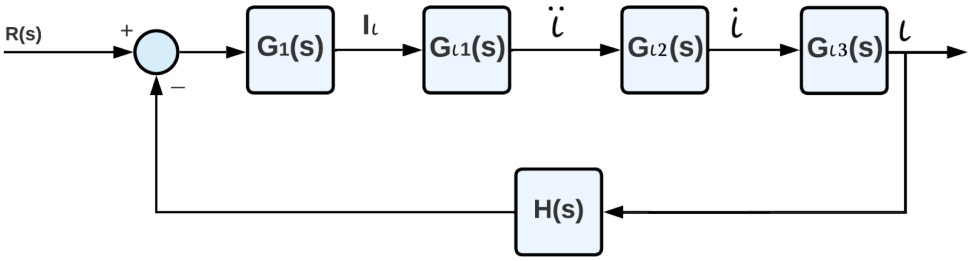


Figure 5.32: Same block diagram as depicted in **Figure 5.28** but with the names given after the mathematical computations shown in this section.

From the above diagrams, the system transfer function for each can be derived. The system transfer function for **Figure 5.31** is,

$$\begin{bmatrix} \dot{x} \\ x \\ \theta \end{bmatrix} = \begin{bmatrix} \frac{\dot{x}}{I_x} \\ \frac{x}{I_x} \\ \frac{\theta}{I_x} \end{bmatrix} \cdot I_x \tag{5.64}$$

For **Figure 5.32**,

$$\begin{bmatrix} \dot{l} \\ l \end{bmatrix} = \begin{bmatrix} \frac{\dot{l}}{I_l} \\ \frac{l}{I_l} \end{bmatrix} \cdot I_l \tag{5.65}$$

Combined to represent the whole system, the linear model's transfer function can be described as the following system of matrices,

$$\begin{bmatrix} \dot{x} & 0 \\ x & 0 \\ \theta & 0 \\ 0 & \dot{l} \\ 0 & l \end{bmatrix} = \begin{bmatrix} \frac{\dot{x}}{I_x} & 0 \\ \frac{x}{I_x} & 0 \\ \frac{\theta}{I_x} & 0 \\ 0 & \frac{\dot{l}}{I_l} \\ 0 & \frac{l}{I_l} \end{bmatrix} \cdot \begin{bmatrix} I_x \\ I_l \end{bmatrix} \tag{5.66}$$

To introduce **Equation 5.66** into a Matlab script, the function `tf(['numerator'], ['denominator'])` can be utilised. Said function only requires the user to introduce the coefficients accompanying s in the numerator and denominator, respectively.

```
TF_Ix_xdot = tf([l_P*b 0 g*b],[a B_x*l_P g*M_1 g*B_x]);
TF_Ix_x = tf([l_P*b 0 g*b],[a B_x*l_P g*M_1 g*B_x 0]);
TF_Il_ldot = tf(6*c, [(m_l+M_L2) B_l_lin]);
TF_Il_l = tf(6*c, [(m_l+M_L2) B_l_lin 0]);
TF_Ix_theta = tf([-1*b 0],[a B_x*l_P g*M_1 g*B_x]);
```

Figure 5.33: Each transfer function as they are in Matlab

The transfer function are organised into a matrix which then represents the plant of the system. It is then possible to create an MPC using the built in Matlab functions. It is preferred to have the system in discrete time, so the c2d Matlab function is used, which in turn requires the system in state space format.

The constraints, horizons and weights are chosen at this point, whereafter the MPC is simulated for a chosen amount of timesteps and a matrix of references.

```
sspace = ss(Mat_TF);
sspaced = c2d(sspace,0.05);
MV = struct('Min',{-11, -11},'Max',{11, 11}); %Apply constraints for Manipulated variable
ssmpcobj = mpc(sspaced, Ts, 120, 3, [], MV); %Tune Prediction Horizon and Control Horizon
ssmpcobj.Weights.OutputVariables = [7, 8, 0.1, 8, 25]; %Tune weights

sim_mpc = sim(ssmpcobj,201, [0, 2, 0, 1, 0]); % Simulation, amount of timesteps, reference matrix
```

Figure 5.34: The MPC is made using Matlab functions

5.6 System Validation

This section will contain tests and essential parameter determination that ensures the proper implementation of the system. Some of the tests have the purpose of validating

the model described in **Section 3**. As well as ensuring that the proper electrical system has been designed. This section will serve a preliminary assurance that the system delivers within some desired minimum.

Initially, the model generated output signals that, seen with the naked eye, did not correlate with the outputs recorded from the real system. One of the possible issues that came up from this was the friction of the system. The initial friction coefficient used were obtained from a previous group of researchers that operated on the same crane to the one used in this project. Due to to the difference seen between the behaviour of the model and the real system, this issue was further investigated.
With the correct friction coefficients, the model can be re-validated, testing that the output signals from the model equals the crane's response to the same input values.

5.6.1 Friction Coefficient

Friction is the force that prevents two objects, in contact with each other, from sliding or rolling relative to one another. This force places a constraint on deriving the input signals of the motors, therefore, it needs to be correctly defined so that the modelled system reacts the same way as the real overhead crane. The friction of the system has been described by its Coulomb, static and viscous friction, F_c, F_s and F_v respectively.

Previously, in **Section 3.3**, the only friction term used was B which was a placeholder for the now computed friction coefficients. Friction based on F_c, F_s and F_v has the following form:

$$F = F_v \cdot v + F_c \cdot sgn(v) + F_s \cdot sgn(v) \tag{5.67}$$

where v is velocity.

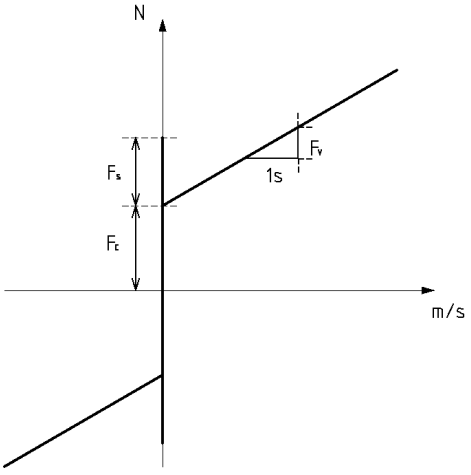


Figure 5.35: Graph of the friction based on the Coulomb, static and viscous frictions.

The focus of this section is to obtain the friction coefficients for the three types of friction described in **Figure 5.35**. This will be done by a series of tests that will allow for the computation of said coefficients.

The first test will not consider static friction and only Coulomb and viscous friction are derived. The friction model of such instance, where friction is only dependent on F_c and F_v, can be described as:

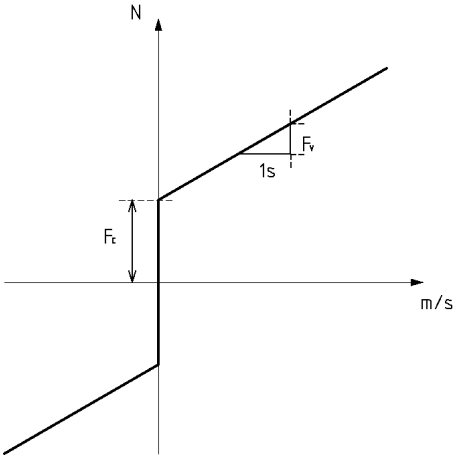


Figure 5.36: Graph of the friction based on the Coulomb and viscous frictions.

$$F = F_v \cdot v + F_c \cdot sgn(v) \tag{5.68}$$

Firstly, the friction coefficients that refer to the x-axis will be computed, thus the coefficients will be written as F_{xc} and F_{xv}. Since the goal is to obtain one value for each coefficient, two equations are needed:

$$\begin{aligned} 1)\ \ddot{x} &= \frac{k_{ex} \cdot_1 I_x}{\left(m_t + \frac{J_x}{r_x^2}\right) \cdot r_x} - \frac{F_{xv} \cdot \dot{x}_1 + F_{xc} \cdot sgn(\dot{x})}{m_t + \frac{J_x}{r_x^2}} \\ 2)\ \ddot{x} &= \frac{k_{ex} \cdot_2 I_x}{\left(m_t + \frac{J_x}{r_x^2}\right) \cdot r_x} - \frac{F_{xv} \cdot \dot{x}_2 + F_{xc} \cdot sgn(\dot{x})}{m_t + \frac{J_x}{r_x^2}} \end{aligned} \tag{5.69}$$

Assuming the acceleration is 0 when the velocity is constant, and assuming neither l nor θ have any effect for this test, the equation of motion for $\ddot{x}$ will only contain the terms shown in **Equation 5.69**.

To isolate both friction variables, F_{xc} and F_{xv}, the crane will move with two different control signals I_x. The first run was performed with a control signal ${}_1I_x$ of $6.35A$ in the negative direction. The second, with a ${}_2I_x$ of $5.08A$ in the negative direction. The velocity terms seen in the equation, $\dot{x}_1$, $\dot{x}_2$, are obtained by averaging the velocity within the span of

time where the crane's velocity seems to remain constant. This value should be different for both tries. It is necessary to compute the average of the velocity as it jerks and never maintains a constant velocity, as seen in **Figure 5.37**.

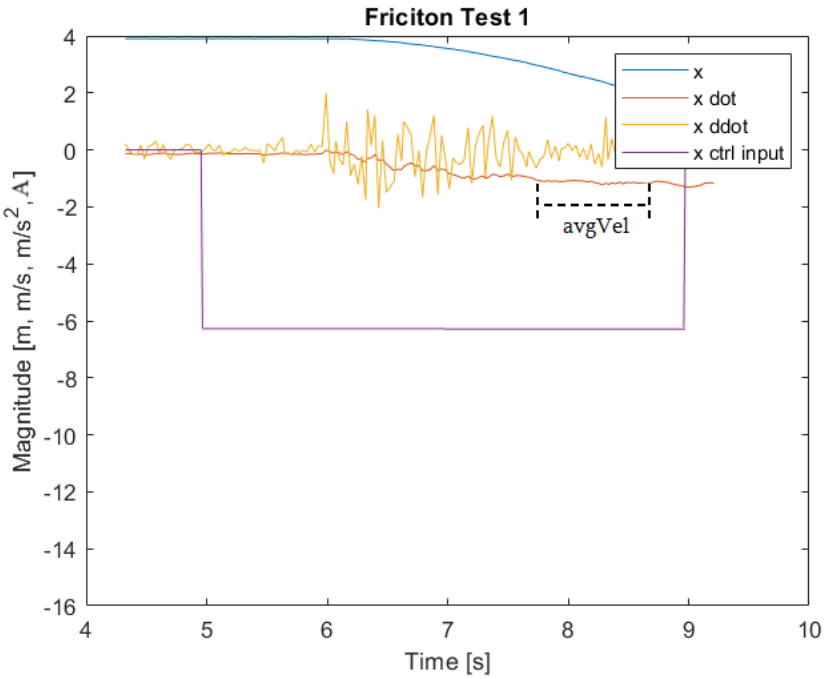


Figure 5.37: Plotted signal of the first test performed for deriving the friction coefficients. The graph depicts the position, velocity and acceleration of the crane in the x axis. The control signal ${}_1I_x$ can also be seen. The span of time considered to be the most constant can be seen named as `avgVel`.

For the first test, with an input signal of $6.35A$ in the negative direction, the measured and averaged velocity was $\dot{x}_1 = -1.16\text{m/s}$. A second test using an I_x of $8.46A$ in the negative direction was performed and it outputted the same velocity $\dot{x}$ as the $6.35A$ in the negative direction signal, therefore, it has been conclude that the maximum allowed velocity for the crane is delimited to be -1.16m/s when the input signal is between $6.35A$ in the negative direction and $11A$ in the negative direction.

In order to obtain the coefficients for the viscous and coulomb frictions, a second equation is necessary. The second test was done with an input signal equal to $5.08A$ in the negative direction. The maximum velocity recorded, while remaining constant, was -0.56m/s.

$$\begin{aligned} 1)\ 0 &= \frac{k_{ex}(-6.35)}{\left(m_t + \frac{J_x}{r_x^2}\right)\cdot r_x} - \frac{F_{xv}(-1.16) + F_{xc}\cdot sgn(\dot{x})}{m_t + \frac{J_x}{r_x^2}} \\ 2)\ 0 &= \frac{k_{ex}(-5.08)}{\left(m_t + \frac{J_x}{r_x^2}\right)\cdot r_x} - \frac{F_{xv}(-0.56) + F_{xc}\cdot sgn(\dot{x})}{m_t + \frac{J_x}{r_x^2}} \end{aligned} \tag{5.70}$$

F_{xc} was isolated using the second function **2)**

$$\begin{aligned}
\frac{-0.56F_{xv} + F_{xc} \cdot sgn(\dot{x})}{m_t + \frac{J_x}{r_x^2}} &= \frac{-5.08k_{ex}}{\left(m_t + \frac{J_x}{r_x^2}\right) \cdot r_x} \\
-0.56F_{xv} + F_{xc}sgn(\dot{x}) &= \frac{-5.08k_{ex}}{r_x} \\
F_{xc} \cdot sgn(\dot{x}) &= 0.56F_{xv} - \frac{5.08k_{ex}}{r_x}
\end{aligned} \tag{5.71}$$

$F_{xc} \cdot sgn(\dot{x})$ can then be introduced in the first equation **1)** which will depend on only F_{xv}.

$$\begin{aligned}
0 &= \frac{-6.35k_{ex}}{\left(m_t + \frac{J_x}{r_x^2}\right) \cdot r_x} - \frac{-1.16F_{xv} + \left(0.56F_{xv} - \frac{5.08k_{ex}}{r_x}\right)}{m_t + \frac{J_x}{r_x^2}} \\
-1.16F_{xv} + 0.56F_{xv} - \frac{5.08k_{ex}}{r_x} &= \frac{-6.35k_{ex}}{r_x} \\
-0.6F_{xv} &= \frac{5.08k_{ex}}{r_x} - \frac{6.35k_{ex}}{r_x} \\
-0.6F_{xv} &= \frac{-1.27k_{ex}}{r_x} \\
F_{xv} &= \frac{1.27k_{ex}}{0.6r_x} = 17.19N
\end{aligned} \tag{5.72}$$

Using the value of the coefficient, the remaining Coulomb friction coefficient can be obtained.

$$F_{xc} = 0.56 \cdot 17.19 - \frac{k_{ex}(5.08)}{r_x} = 31.62N \tag{5.73}$$

After obtaining both the Coulomb and viscous friction coefficients, the static friction coefficient can be obtained. Mathematically, it can be arduous to calculate. Therefore, instead of deriving its value using mathematical approaches, the static friction can be obtained by manually tuning the input signal so that it always includes the necessary force to overcome static friction.

To obtain said value, a simple test can be performed. The crane will be fed an increasing input signal, stating at 0. The system will be measured at all times until the crane starts moving. The value of the voltage needed for the crane to start moving will be determined and utilised in the following equation:

$$\ddot{x} = \frac{k_{ex} \cdot I_x}{\left(m_t + \frac{J_x}{r_x^2}\right) \cdot r_x} - \frac{F_{xv} \cdot \dot{x}_1 + F_{xc} \cdot sgn(\dot{x}) + F_{xs} \cdot sgn(\dot{x})}{m_t + \frac{J_x}{r_x^2}} \tag{5.74}$$

This equation is identical to **Equation 5.69**, but including the term for the static friction. Assuming F_{xv} and F_{xc} are 0 when the system is still, velocity is zero, the equation can be

simplified as,

$$0 = \frac{k_{ex} \cdot I_x}{\left(m_t + \frac{J_x}{r_x^2}\right) \cdot r_x} - \frac{F_{xs} \cdot sgn(\dot{x})}{m_t + \frac{J_x}{r_x^2}} \tag{5.75}$$

The test was performed and the current necessary to compensate for the static friction can be seen in the following figure.

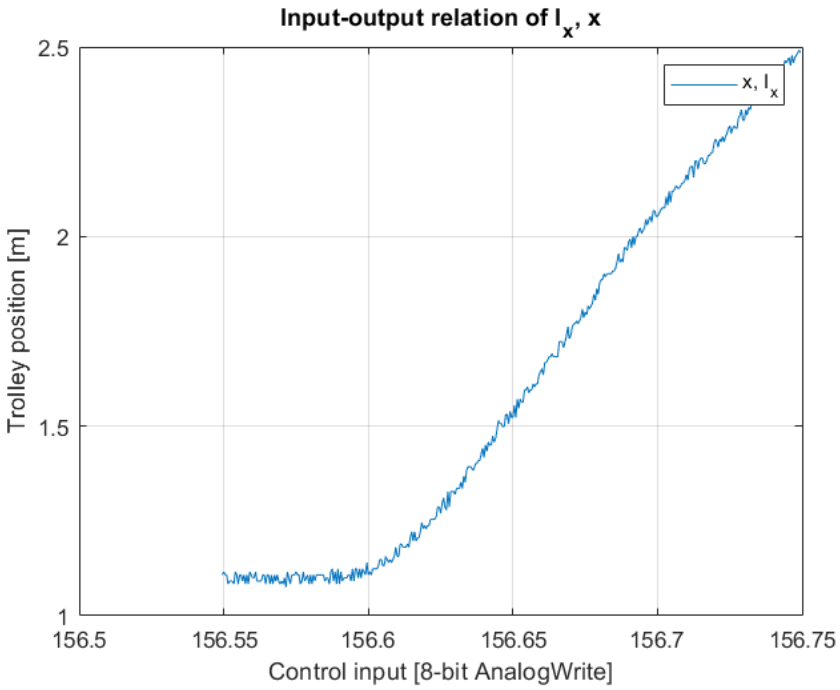


Figure 5.38: Graph depicting the minimum input signal necessary for the crane to start moving.

As seen in **Figure 5.38**, the measured velocity starts to change when the input signal reaches a value of $2.5A$. With I_x obtained, it can be introduced into **Equation 5.75**, which, after isolating F_{xs}, derives:

$$F_{xs} \cdot sgn(\dot{x}) = \frac{k_{ex} \cdot I_x}{r_x} = \frac{0.609 \cdot X}{0.075} = 20.3N \tag{5.76}$$

After obtaining the coefficients for the three frictions modelled, the model of the crane should be capable of performing the same as the real crane.

5.6.2 Model Validation

The dynamics of the crane have been represented, up to this point, in three different ways. First, is the actual response of the real crane to a control input. This response is considered to be reality and was used to tune the simulated systems, the nonlinear and linear models.

The second representation of the system is the nonlinear model, which has the second-most realistic response. And lastly, the linear model. These three system should have similar responses to the same input signals. To test that this claim is correct, the three systems will be compared.

Open loop response for maximum input signal

The static friction has not been modelled as it gave incorrect responses. A test using max control input for as long as the physical crane allows is chosen. This minimizes the effect of static friction while showing the largest amount of data for position and velocity. The response of the crane to this input is shown below in **Figure 5.39**.
Then the same test was performed for the nonlinear system in Simulink, since its response should be the most similar to the real crane.

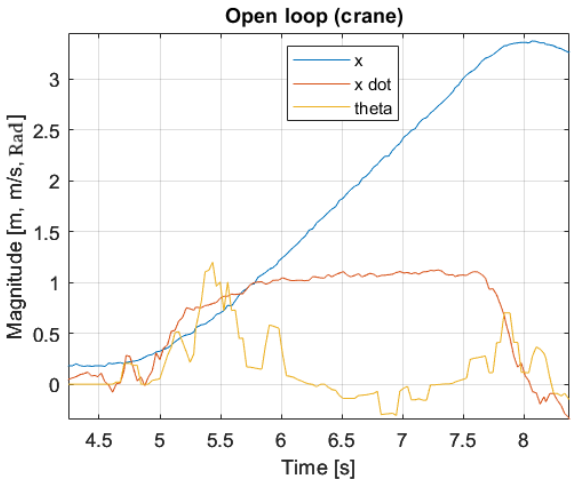


Figure 5.39: Graph showing x position, x velocity and the angle of the crane. For the test, a max input signal was given for as long as possible.

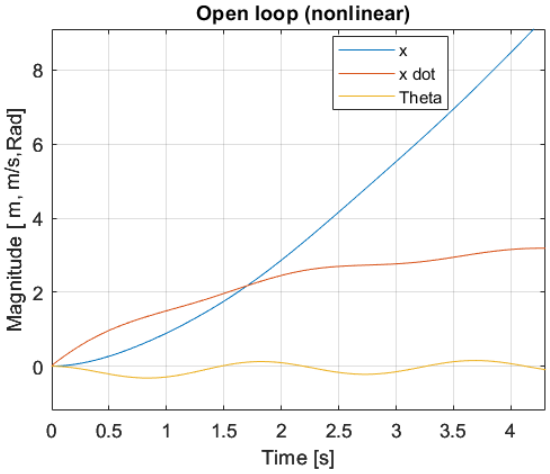


Figure 5.40: Graph showing x position, x velocity and the angle from the nonlinear model. A max input signal was given for ten seconds. The graph has been zoomed in to the same timespan as the crane.

Looking at the graphs and noticing the difference in scale of the y axis, it can be seen that the velocity of the model increases faster and for much longer. The position consequently also moves much further in the same amount of time.

As a result of this test, it was chosen to tune the friction coefficients for the models, until their responses better matched the response of the real crane. It was also chosen to further simplify the friction to only include viscous friction. This one friction coefficient was then tuned such that the steady state velocity for a given input matched the steady state of the measurements from the real crane.

This was done manually by increasing and decreasing the friction coefficient before checking the systems response. For the linear model of the system this produced a response in the correct shape but wrong magnitudes, so it was also chosen to modify the acceleration parameters. Doing so produced a model that better represents the actions of the real system as shown below in **Figure 5.41**.

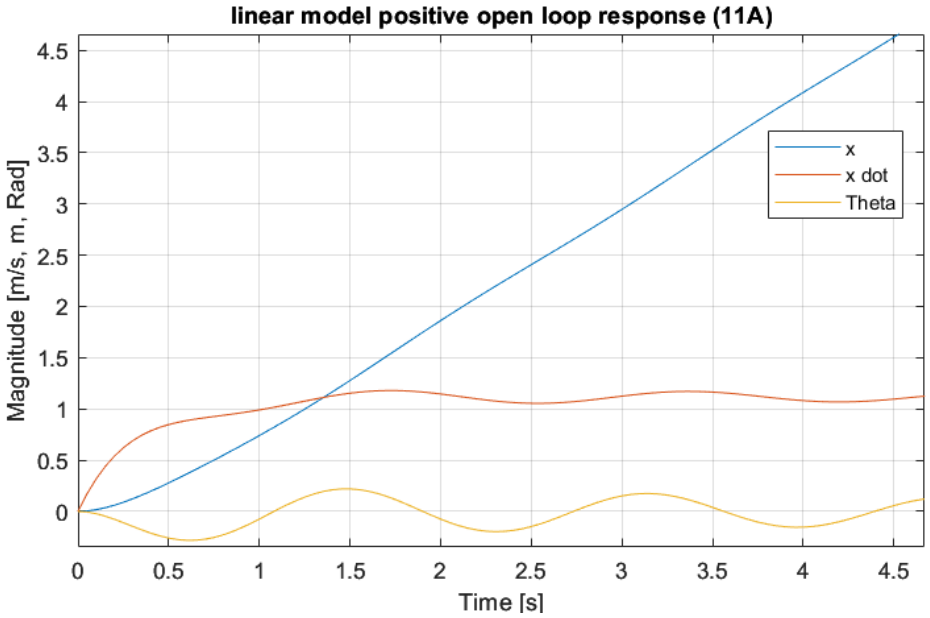


Figure 5.41: Graph showing the x position, x velocity and angle after having tuned the parameters.

Open loop response for maximum input signal, 1 second

Having tuned the models, a series of open loop tests are performed for both x and l in each direction. The systems were asked to move for one second at maximum velocity. This was done three times for both the x and y axes, in the positive and negative direction of each of them. The behaviour of both linear and non-linear simulated model, as well as the crane will be compared.

X-Axis

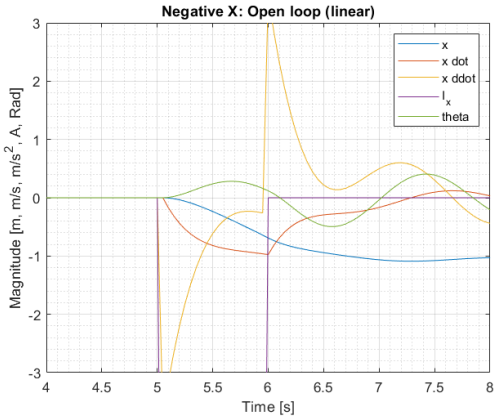


Figure 5.42: Linear system tested for the negative movement of x.

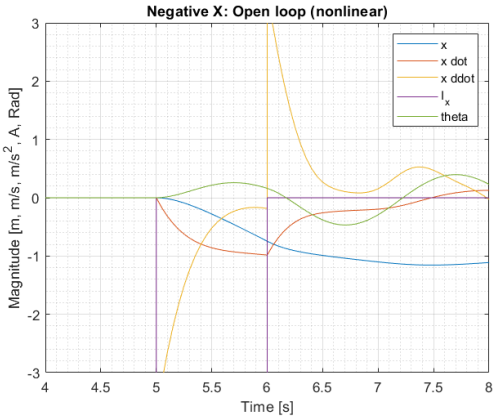


Figure 5.43: Nonlinear system tested for the negative movement of x.

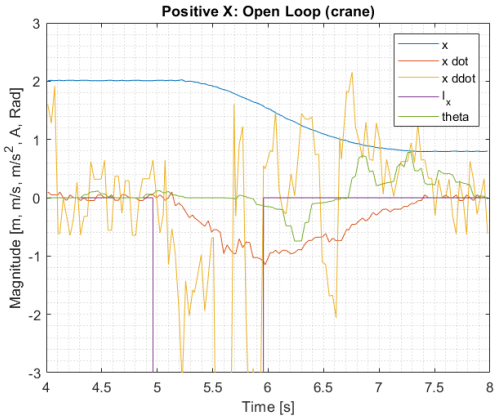


Figure 5.44: Crane tested for the negative movement of x.

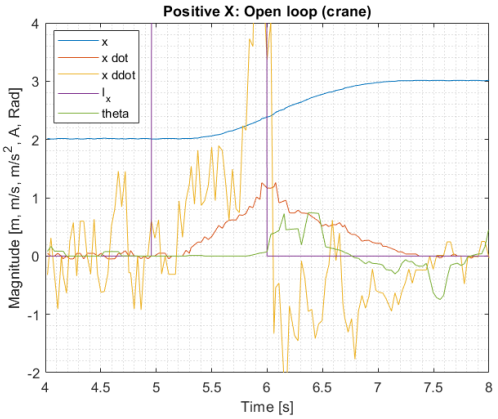


Figure 5.45: Crane tested for the positive movement of x.

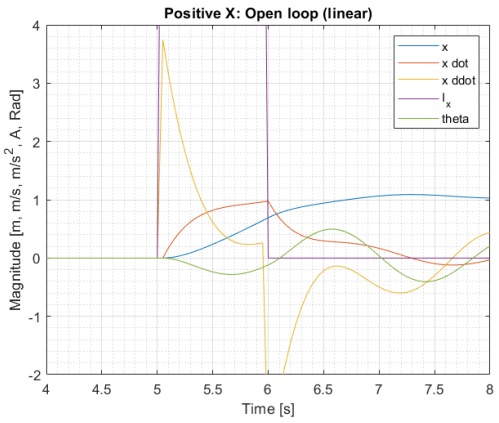


Figure 5.46: Linear system tested for the positive movement of x.

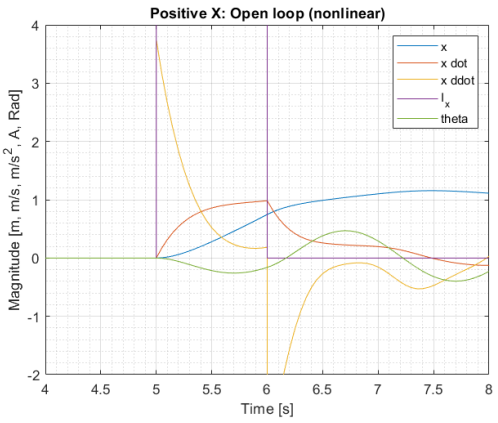


Figure 5.47: Nonlinear system tested for the positive movement of x.

The graphs depict how the the three systems respond to the same input signal, which in this case is an input signal of $11A$ in the negative and positive directions of x-axis respectively. As it can be seen, the change in position is almost the same for the graphs in both directions. Between the linear and nonlinear responses, the position, velocity and acceleration responses are close to identical, but differ from the crane's response. This is hypothesised to be due to natural noise which has not been modelled for the simulations. This is specially true for the velocity and acceleration which could have an increased noise as an aftereffect of taking the derivative of the position for the velocity and a second derivative for the acceleration.

Y-Axis

For the y-axis, the same command as for x was given for l. The crane has to move at maximum velocity for one second for both up and down directions, recording its measured y position.
The test was performed in the positive and negative direction of the y-axis.

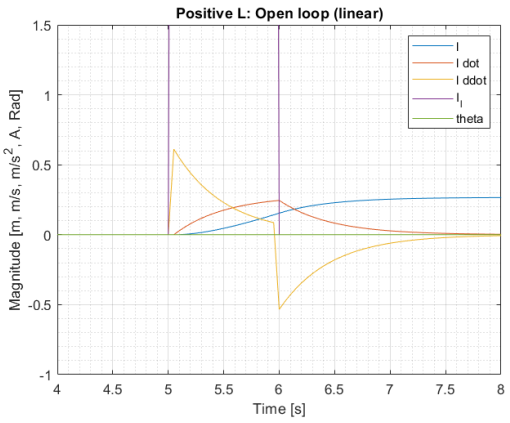


Figure 5.48: Linear system tested for the positive movement of *l*.

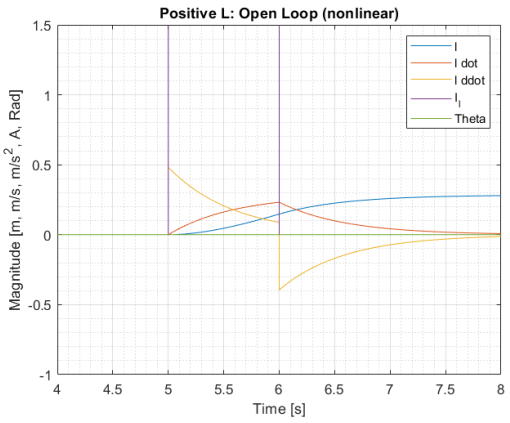


Figure 5.49: Nonlinear system tested for the positive movement of *l*.

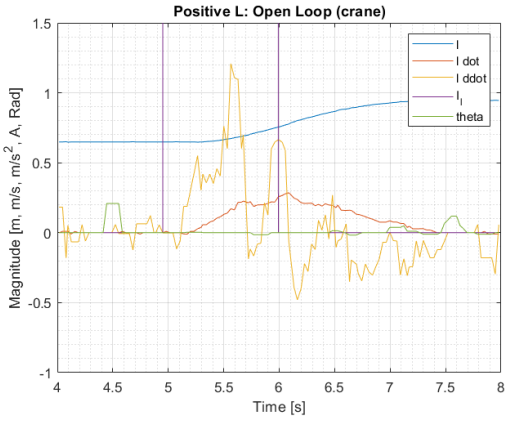


Figure 5.50: Crane tested for the positive movement of *l*.

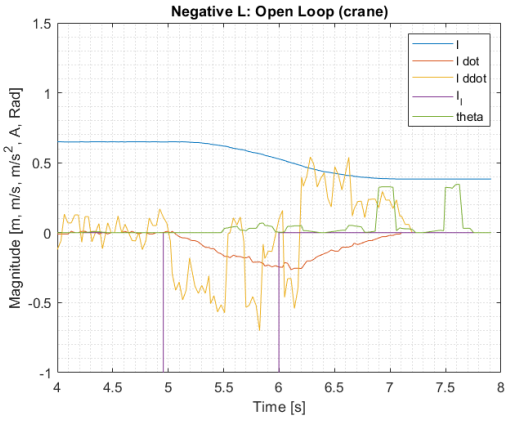


Figure 5.51: Crane tested for the negative movement of *l*.

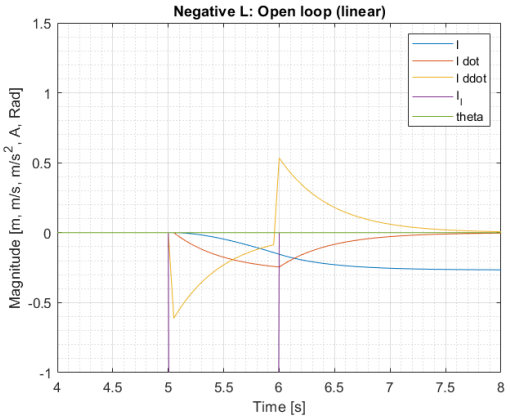


Figure 5.52: Linear system tested for the negative movement of *l*.

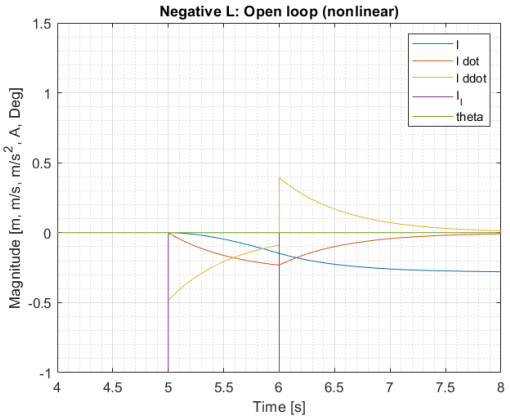


Figure 5.53: Nonlinear system tested for the negative movement of *l*.

The graphs depict how the the three systems respond to the same input signal, which in this case is an input signal of $11A$ in the negative and positive directions of y-axis respectively. As it can be seen, the change in position is almost the same for the graphs in both directions. Between the linear and nonlinear responses, the position, velocity and acceleration responses are close to identical, but differ from the crane's response. This is hypothesised to be due to natural noise which has not been modelled for the simulations. It is especially true for the velocity and acceleration which could have an increased noise as an aftereffect of taking the derivative of the position for the velocity and a second derivative for the acceleration.

5.7 Summary

In this chapter the algorithms and approaches selected in **Chapter 2**, and **Chapter 3**, have been further expanded and its implementation on this solution explained. The first subsystem described was the path planning algorithms, Ant Colony Optimisation and Potential Field planners. The next section, **Section 5.4**, delved into computed torque. An approach utilised to linearise the dynamic system, allowing for linear control algorithms to be utilised. With a linear system, a linear MPC could be implemented. **Section 5.3** describes the process of creating a Model Predictive Controller and the implementation of it in Matlab. During this process, it was noticed that due to the system being linearised using Computed Torque, the benefits of using the MPC were reduced. The dynamics of the system were simplified to the extent that it is believed that the linearised system would have practically the same performance with a PD controller as with an MPC. For this reason, a second approach for linearising was implemented, namely Taylor Series Linearisation. It was deemed that this method delivered a system that suited the designed MPC, as the complexity of the dynamic model obtained with it was greater than that of the Computed Torque approach. Lastly, the system was validated for the linear and nonlinear models. One of the additions of the model included in this section is the study of friction coefficients. These were obtained in order for the model to better imitate the real crane.

In the following chapter, the solution will be tested to verify it follows the previously set requirements. Some of the issues mentioned in this summary will be further expanded in the discussion and conclusion chapters.

Chapter 6

Testing

This chapter will display the tests performed for proving the requirements set in **Chapter 4** have been met. The test will be explained, thereafter, the results will be shown.

6.1 Test Scenario

The solution presented in this project will be designed to operate in a certain environment. In this section, the test scenario is described.

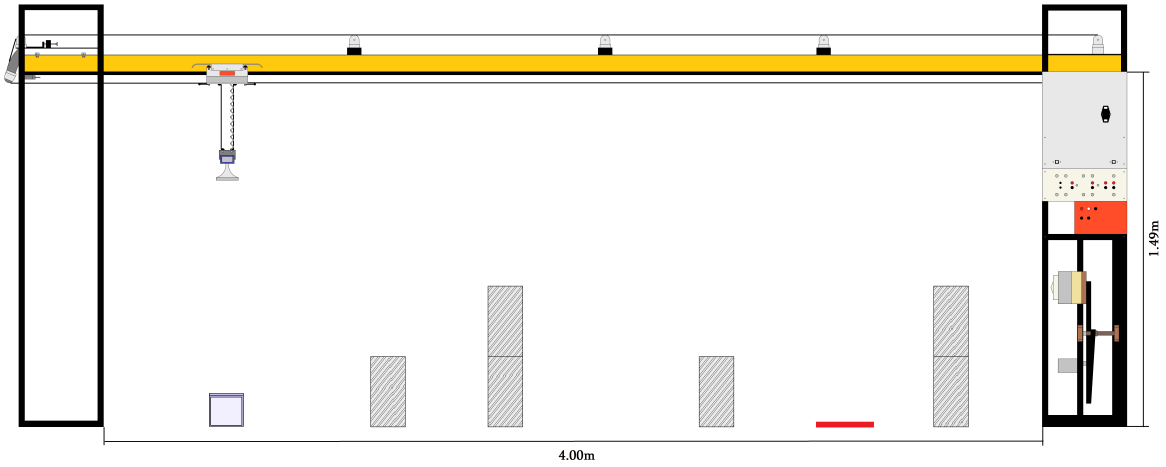


Figure 6.1: Test Scenario. Example of a scenario for the testing phase. The bottom left block is the object that will be transported from that starting location to the target location, depicted as a red mat. The other objects seen in the frame imitate containers that have been stacked beforehand, and act as obstacles for the crane to avoid.

In **Figure 6.1** an example scenario for testing the solution is displayed. Using the implemented trajectory planning, MPC, and hardware, the crane must move the container object to the target position, while avoiding the obstacles. Different obstacle configura-

tions will be used for different tests. The obstacles in this project will be wooden pillars of $116mm \times 390mm$. The measurements can be seen in **Figure 6.2.**

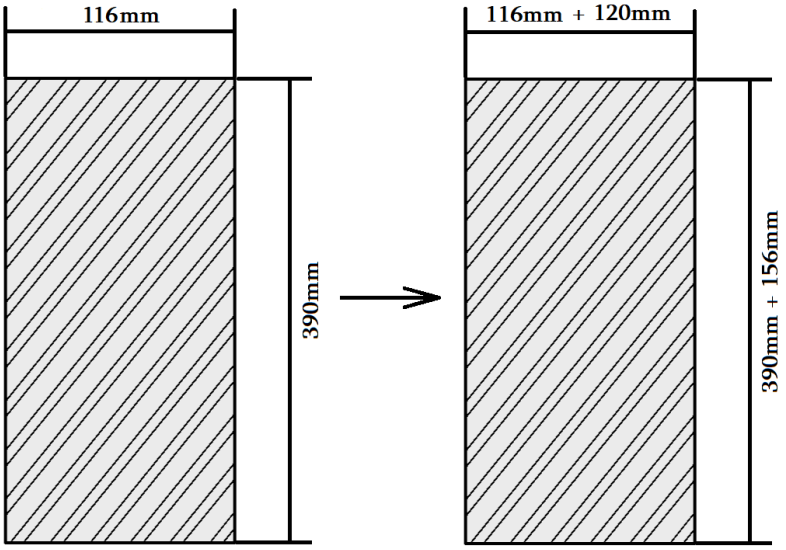


Figure 6.2: Dimensions of the obstacles. The obstacle on the left shows the dimensions that the obstacles have in the tangible workspace. The one on the right refers to the dimensions used in path planning, ensuring the size of the transported object is taken into consideration.

Figure 6.2 displays two different dimensions for the same obstacle, one for the real world obstacles and one for the path planning. This change in dimension of the obstacles has been done to ensure the transported object is taken into account when computing a path. The transported obstacle has the dimension shown in **Figure 6.3**. This was done due to the planners only measuring the center position of the crane's head, located at the bottom of the electromagnet (see **Figure 2.3** in **Section 2.1**). In the image to the right, **Figure 6.3**, a red dot has been drawn on top of the object, marking the point at which the head of the crane and the object meet. This point is the one measured by the system and outputted by the path planners, therefore, the additional lengths that the object adds to the path needs to be counteracted.

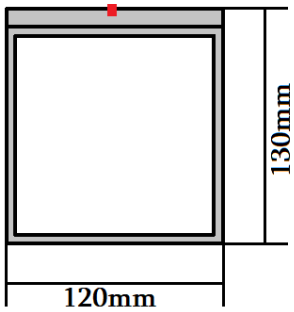


Figure 6.3: Dimensions of the transported object.

To solve this issue, an offset can be added to the virtual obstacles, as seen in **Figure 6.2**, where the offsets have already been added to the obstacle on the right. On the width of the obstacles, $120mm$ are added, for the height, $15.6mm$. These distances come from the dimensions of the transportation object which, when in motion, will add its volume to the cranes trajectory. To ensure the object does not impact with the obstacle, an additional

distance was added. The width of 15.6mm was obtained by adding a 20% increment to the height of the transportation object (130mm). For the width, a different approach was taken. The natural behaviour of the crane allows a greater sway on the horizontal motion than vertically. Therefore, instead of a 20% increment, the transportation object width was doubled, allowing an additional 60mm deviation on both sides of the obstacles. The total dimension of the obstacles for the simulation is $236mm \times 546mm$.

6.2 MPC Testing

The tests designed for the MPC will be described in this section, together with the results for each of them.

Test 1.1

Requirement 1.1: The MPC must be able to move to and reach a steady state for any given reference for x and/or l.

The test designed was:

1.1	A reference x and l will be fed to the MPC. Record the outcome position, velocity and acceleration of the system variables	The MPC must be capable of controlling the system such that its controlled variables reach steady state. The result of this test can be considered accepted if, for the reference x and l, the simulated output reaches a steady state.

The test was performed as stated in the acceptance test in the above table. A reference $x = 1m$ and $l = 1m$ were given to the MPC which then outputted the resulting action of the simulated system. The following figure, **Figure 6.4** shows the simulated system trying to reach the desired x and l position.

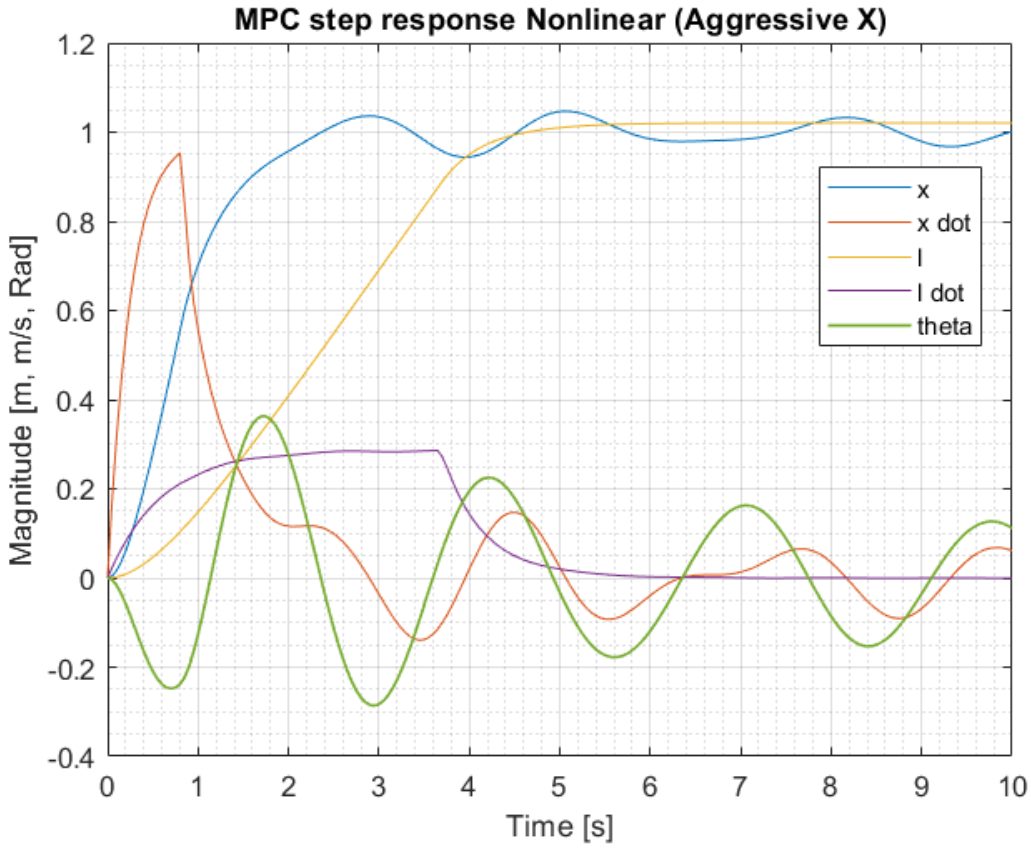


Figure 6.4: Graph depicting the steady state response of the simulated nonlinear model using a linear MPC.

As seen in the above graph, the MPC is capable of reaching steady state for the reference position. l is specifically good at reaching steady state, whereas x requires more time to do so. In this case, θ is not being controlled as much as the other two. This is because the MPC in this case has been tuned to focus more on x than θ, as both are interconnected to each other by the dynamics of the system. Both x and θ are controlled using a single input signal, this is because no direct control of θ is possible due to the cranes design. In the following figure, **Figure 6.5**, the contrary has been done. Having more focus on controlling θ than x, and maintaining the same control for l.

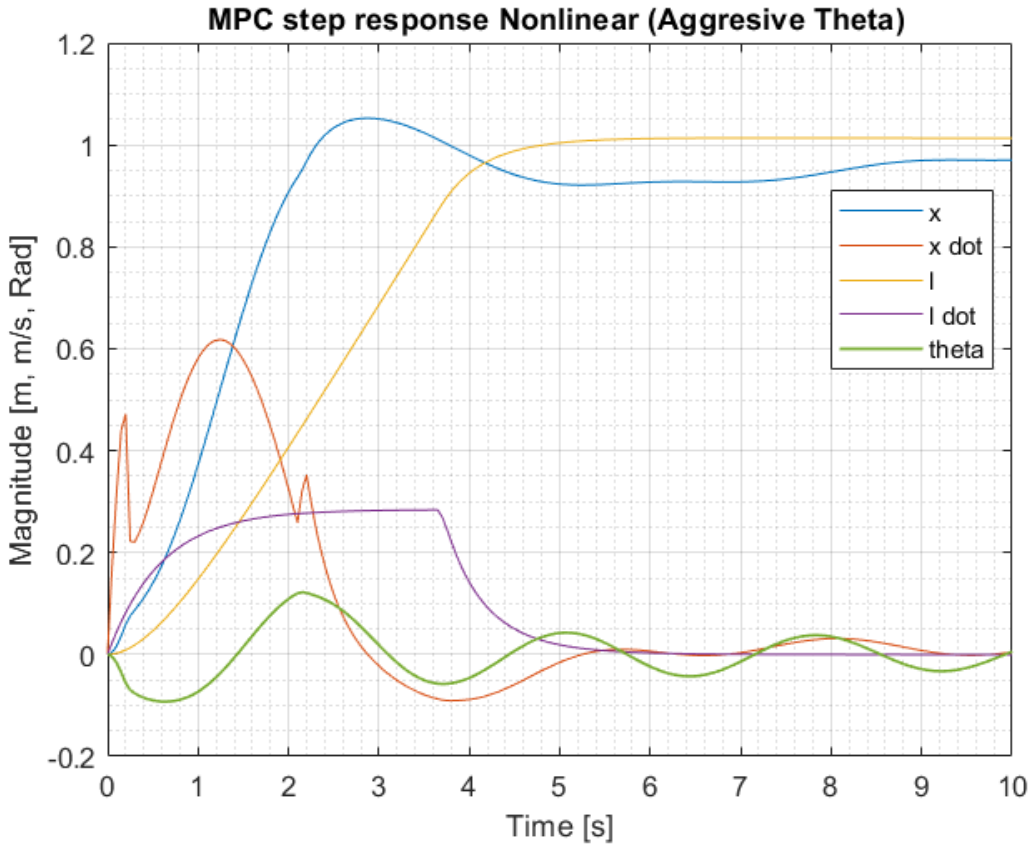


Figure 6.5: Graph depicting the same step response as the previous one but with a more aggressive control on θ than on x.

In this case, θ reacts much less than previously and its oscillation are less aggressive as in the previous test. Nevertheless, x is not capable of reaching steady state and seems to lack behind.

The MPC is capable of controlling l regardless of the other two variables. Controlling x is also possible, if the focus of the tuning falls on x and θ is neglected. The MPC has been shown to be capable of reaching steady state for both x and l, therefore, it has been deemed as passed.

Test 1.2

Requirement 1.2: The system relies on the MPC being capable of following any path that it receives, within the boundaries set by the physical limitations of the crane.

The test designed was:

1.2	The MPC will follow a desired path. The simulated outcome of the system will be recorded.	The solution has been designed to follow a desired trajectory, therefore, the MPC must be capable deriving the necessary control signals so that the system follows said path. The test is considered passed if the MPC is capable of following the desired path within the physical limitations that the crane actually has.

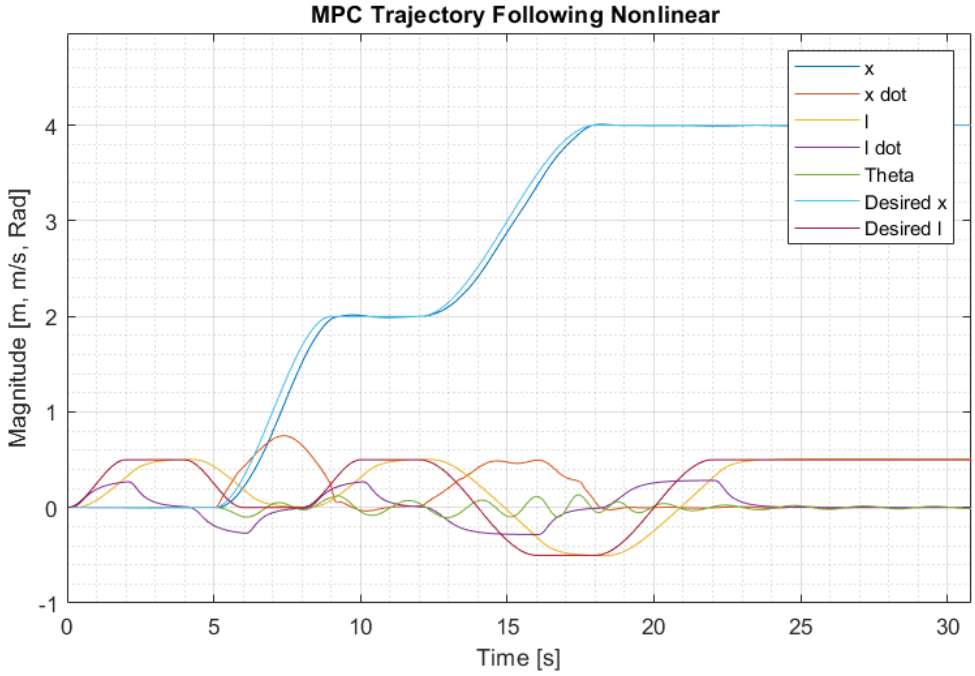


Figure 6.6: The graph shows the system following a desired path. Both x and l fall a bit behind the desired position, but it is considered normal.

Utilising one of the paths obtained from the path planning algorithm, the MPC was tasked with controlling the linear model of the system so that it followed the given path. As seen in the above graph, the MPC is capable of following the desired path. The difference between the reference path and the actual path is approximately one second between the two for x, which is also similar for l. A longer path is shown in the following graph, where the time to complete the path is of 100 seconds, which is believed to aid the MPC in following the desired path.

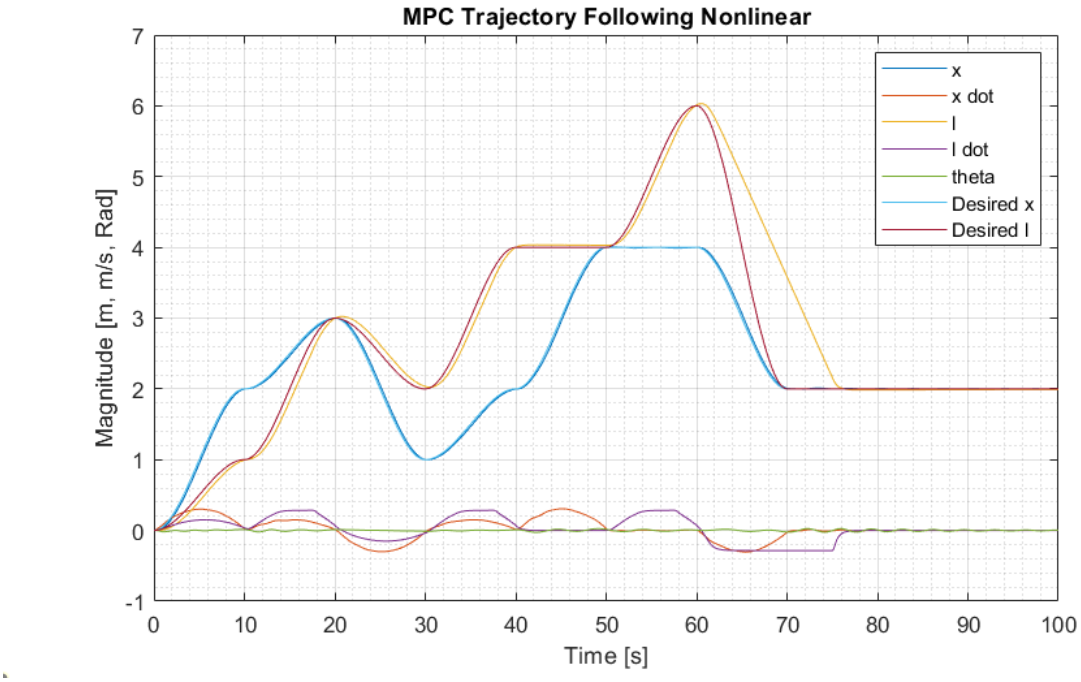


Figure 6.7: The graph shows the system following a desired path. In this case, more time has been given to the system to perform the task. Therefore, the difference between desired and actual paths appears to be smaller than in the previous graph.

Figure 6.7 shows the system following a desired path for about 100 seconds. The MPC falls behind the desired path as it is not capable to calculate fast enough the control signal necessary to follow the same path as desired. In this iteration of the test, the most clear difference between desired and actual path can be seen for l at around 60 seconds. This is due to the change in position that the followed path shows at that time cannot be followed by the system, as the velocity would go over the allowed by the crane.

The last path that will be tested is one generated offline by the potential field path planning algorithm. This path should depict the ability of the MPC to follow a path that the crane may have to performed in a real scenario.

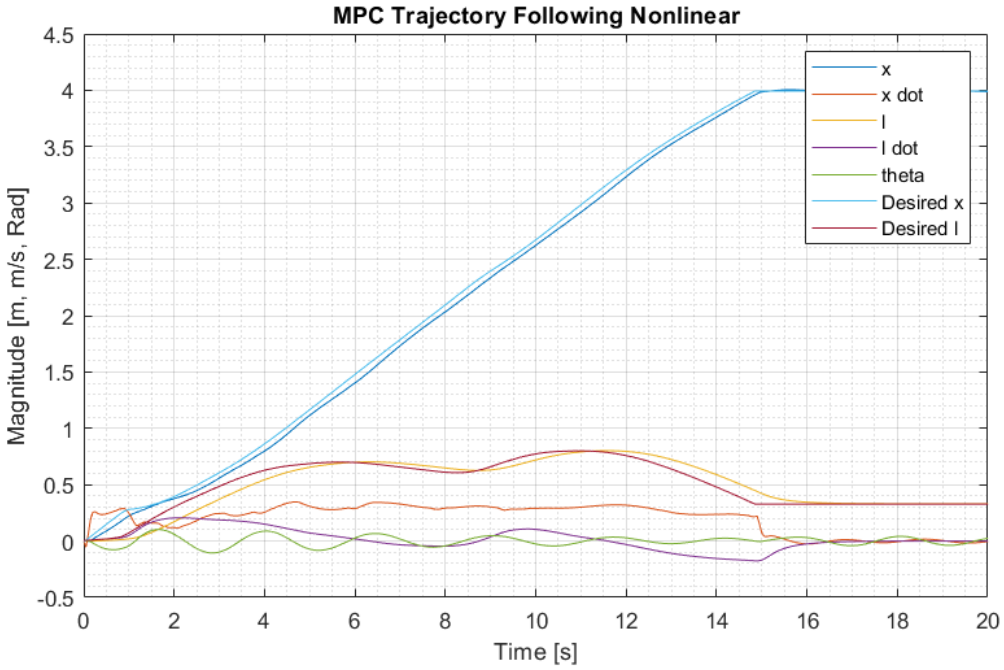


Figure 6.8: The graph shows the system following a desired path. In this case, the path to be followed has been obtained from the potential field path planning algorithm.

As shown for the three different paths, the MPC is capable of following the desired path with some deviation between the two. As an example of this, the following graph shows the deviation between desired and actual paths of both x and l for the path in **Figures 6.6** and **6.7**.

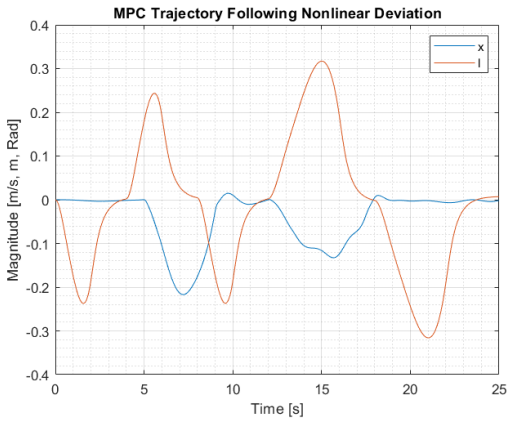


Figure 6.9: Deviation between desired and actual path for both x and l. Calculated from the path shown in **Figure 6.6**

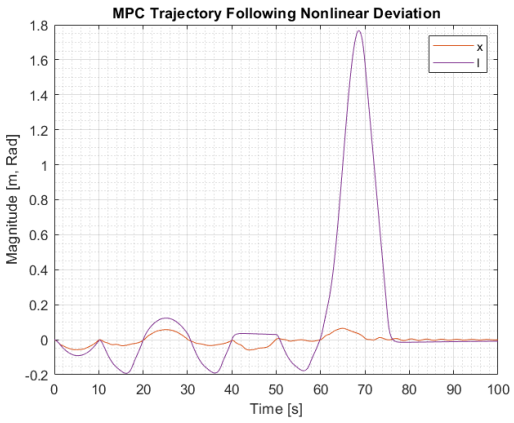


Figure 6.10: Deviation between desired and actual path for both x and l. Calculated from the path shown in **Figure 6.7**

As it can be seen, the graph on the left has a maximum deviation of about 0.31m whereas the graph on the right has a much more prominent maximum deviation of 1.75m. The big maximum deviation of the second graph is due to the unrealistic desired path, as it tries to change its l position in a timespan that the real crane would not allow.

6.3 Hardware Testing

The aim in this section is to validate the hardware requirements from **Subsection 4.1.2**. Each of said requirements are shown again below for ease of reading, and then validated.

Test 2.1

Requirement 2.1: The position states, x and l, from the inbuilt crane potentiometers, must be available at all times and represent the physical system.

The test designed was:

2.1	Record data of the potentiometers measuring the cranes position for both x and l.	The requirement can be considered to be fulfilled if the potentiometer measurements can be obtained and recorded for further use.

This requirement was achieved by first connecting the potentiometer output wires from the DB-25 cable to the Arduino Mega. The measurements are transmitted as an analog voltage, which is then converted to a digital signal by the ADC (Analog to Digital Converter) on the Arduino Mega. The Arduino Mega communicates with Matlab running on a PC, through UART serial communication. The potentiometer measurements, along with other data, are transferred from the Arduino Mega (see **Figure 6.1**), and updated in the Matlab script at a rate of about 20 Hz.

The received measurements are represented as a number in the range [0,1024]. And in order for them to represent the real position of the crane, they are converted using the mapping explained in **Section 5.2**, to a value in meters. The measurements, in units that represent the real system, are then available for use by a controller, and the requirement is considered fulfilled.

Listing 6.1: This is a code snippet from the script running on the Arduino Mega when using the crane. In short, it reads sensor data from the I/O pins and stores it in an array along with other data to be transmitted. Each array element is transmitted, separated by a comma, and a start and end marker are added to the data string so it can be distinguished from previous and future data in Matlab.

```
% Read sensor data and store it in the array
for (int i = 0; i < 4; i++) {
  sensorData[i] = analogRead(A0 + 2*i); % Array for holding sensor data
}
imuDataToFloat(); % Read angle data from nano and store as imuAngle
sensorData[4] = imuAngle;
sensorData[5] = modifiedValue1; % x control input
sensorData[6] = modifiedValue2; % l control input
sensorData[7] = j; % "time" variable

% Send sensor data to MATLAB
Serial.print("DATA"); % Start marker
for (int i = 0; i < sensorCount; i++) { % sensorCount = 8
  Serial.print(sensorData[i]); % Array holding data to be sent
  if (i < sensorCount - 1) {
    Serial.print(","); % Separator between values
  }
}
Serial.println(); % End marker
```

Additionally, see **Figure 6.11** for an example of the trolley position, x, measured by the potentiometer.

Test 2.2

Requirement 2.2: The velocity states, $\dot{x}$, and $\dot{l}$, from the inbuilt crane tachometers, must be available at all times and represent the physical system.

The test designed was:

2.2	Record data of the tachometers measuring the cranes position for both $\dot{x}$ and $\dot{l}$.	The requirement can be considered to be fulfilled if the tachometer measurements can be obtained and recorded for further use.

The crane tachometers output a voltage in the range [-26, 26]V, which is proportional to the crane's velocity. However, since the Arduino cannot handle such voltages, an op amp solution was used, as explained in **Subsection 3.2.4**, before connecting the tachometer to the Arduino. The tachometer measurements are transferred to Matlab using the same serial communication as the potentiometer measurements (see **Figure 6.1**). The digital representation of the analog signal, converted by the ADCs, are a value in the range [0,1024]. Therefore, a conversion to m/s was made, as shown in **Subsection 5.2.2**. An example

of the tachometers in use can be seen in **Figure 6.11**, where the behaviour of the physical crane does match the measurements. And the requirement is thus considered to be fulfilled.

Test 2.3

Requirement 2.3: The sway angle state, θ, from the IMU on the crane head, must be available at all times and represent the physical system.

The test designed was:

2.3	Record data of the IMU where angle of the crane's head θ using the IMU.	The requirement can be considered to be fulfilled if the IMU measurements for the crane's head angle can be obtained and recorded for future use.

The direct communication with the IMU is handled by the Arduino Nano. The sway angle, θ, is calculated and transmitted to the Arduino Mega via serial communication, where it is stored. It is then transmitted via the other serial communication channel to a PC with Matlab, where it is converted from degrees to radians and stored for use by the controller. An example of the IMU being used, to measure the sway angle can be seen in **Figure 6.11**, along with measured trolley position and trolley velocity.

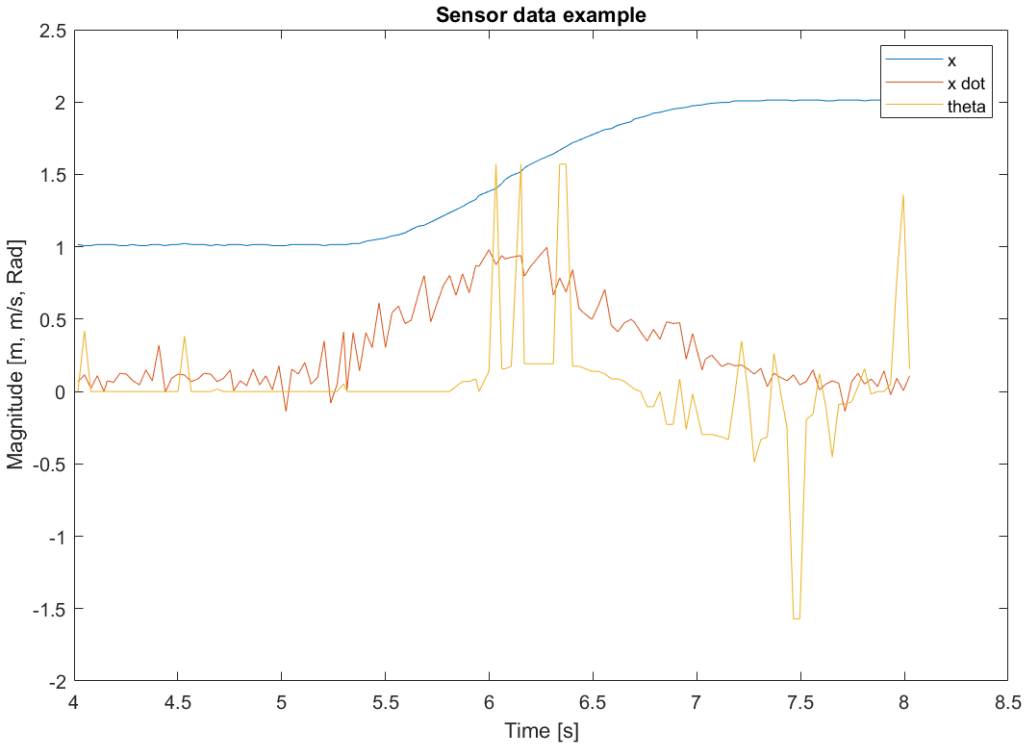


Figure 6.11: This graph shows an example of a crane movement while measuring the trolley position, x, trolley velocity, $\dot{x}$, and crane head angle, theta. These are measured by the x potentiometer, x tachometer, and IMU, respectively.

Test 2.4

Requirement 2.4: The system must be able to enable and disable the electromagnet on command, and thereby allow for the crane to lift, move, and release the shipping container models.

The test designed was:

2.4	The crane will be commanded to pick and place a container using the electromagnet.	The test can be deemed as passed if it can be proved that the electromagnet can be utilised on command.

The electromagnet is directly controlled by the Arduino Nano, which is mounted on the crane head near the electromagnet. UART Serial communication has been set up between the Arduino Mega and Arduino Nano. In addition, it was desired to control the electromagnet from the Matlab script. Therefore, the magnet commands are transmitted first from Matlab to the Arduino Mega through the previously mentioned serial connection.

Then the Arduino Mega sends are corresponding magnet command to the Arduino Nano. And finally, the Arduino Nano sends a corresponding control signal to the electromagnet. As an example of these magnet commands, a code snippet of a function from the Arduino Mega main script has been included below.

Listing 6.2: This code snippet shows a function used by the Arduino Mega. It checks the command received from the Matlab serial communication, and sends a corresponding signal to the Arduino Nano, which is mounted on the crane head. From the serial communication, the Mega receives either 0, which does nothing, 1, which switches the magnet on, or 2 which switches the magnet off

```
1  void magnetControl(){
2    if (modifiedValue3 == 1){     % magnet command received from Matlab
3        Serial3.write("M1 \n");   % turn on
4    }
5
6    else if (modifiedValue3 == 2){
7        Serial3.write("M0 \n");   % turn off
8
9    }
10 }
```

An example of the electromagnet in use can be seen in **Figure 6.12**. And since the pipeline of magnet control works, and the system is able to turn the magnet on and off as needed, the requirement is considered fulfilled.

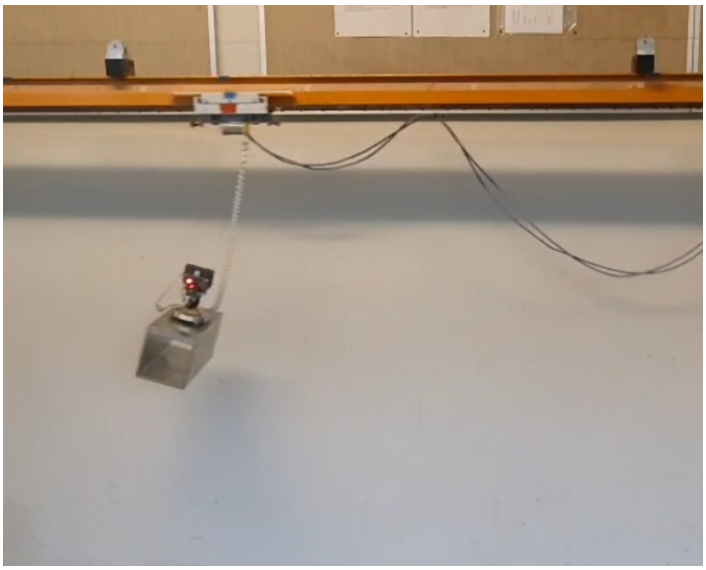


Figure 6.12: This figure shows a still frame of the crane moving, while the miniature shipping container is being held up by the electromagnet.

Test 2.5

Requirement 2.5: The system must be able control the trolley acceleration in both directions along the x-axis, by sending PWM control signals to the x-motor controller.The system must be able to control the wire length, l, by sending PWM control signals to the y-motor controller. In addition, the system must be able to send a valid enable signal to each motor controller, and thereby enable them.

The test designed was:

2.5	The system will be asked to move the crane's head in both the horizontal and vertical directions.	The requirement can be deemed as met if the PWM signals and enable signals used to control the motor controllers deliver the desired motions.

The PWM signal wires are connected between the Arduino Mega and the motor controllers through the DB-25 cable. The Matlab and Arduino Mega scripts contain functionality to send control inputs from Matlab to the Arduino, which sends a corresponding PWM signal to the motor controllers. And the motor controllers then send a control signal directly to the motors. This is all done through the previously explained serial communication channels between Matlab and the Arduino Mega, and between Arduino Mega and the motor controllers. The PWM signals are explained in further detail in **Subsection 3.2.2**.

However, before the motor controller can be controlled this way, an enable signal has to be sent to activate the motors. The enable signal wires are similarly connected to the motor controllers through the DB-25 cable. And in order to enable them, a 100% duty cycle PWM signal is sent from the Arduino Mega. This is done once when the main script is run.

Since the motor controllers are able to turn on, and the crane responds as intended to the control inputs, this requirement is deemed to be fulfilled. An example of PWM-signal being used to control the crane can be seen in **Figure 6.13**, where the input causes the state changes shown.

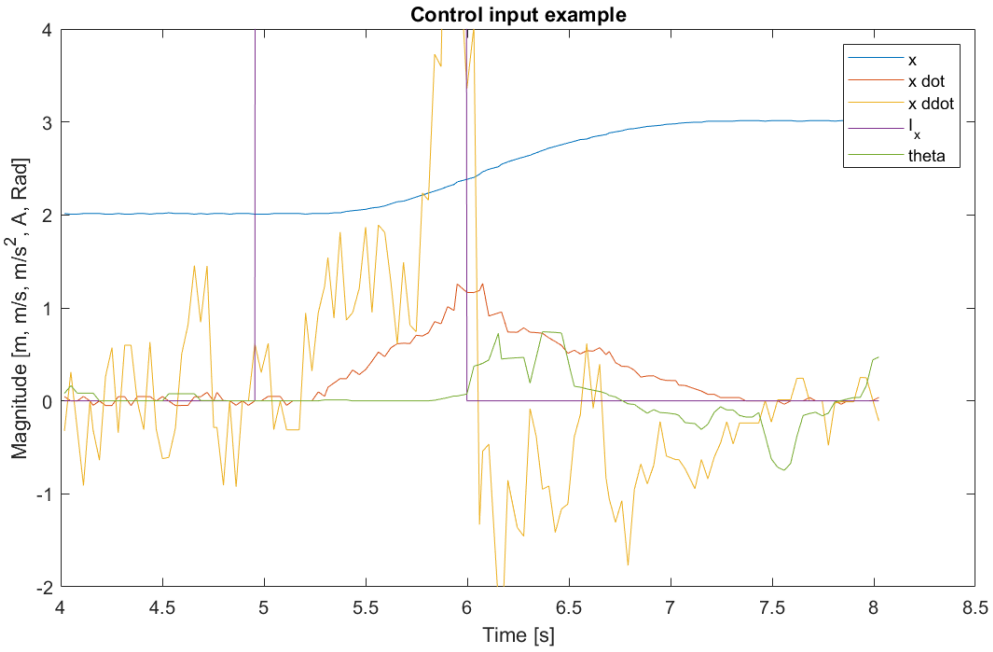


Figure 6.13: This figure shows the trolley moving along the x-axis as a result of the applied PWM-signal, I_x.

6.4 Path Planning Testing

The tests designed for the path planning algorithms will be described in this section.

Test 3.1

Requirement 3.1: Potential Field planning tends to have difficulties at generating a path due to getting stuck in local minima. Implement a solution that solves this issue.

The test designed was:

3.1	Place 1-3 obstacles in different locations of the workspace. Run the program for 10 distinct iterations, and store the results.	Potential Field has the disadvantage of getting stuck in local minima. The solution should have a remedy for this problem. This test compares the results of the planner with the added complement and without it. The test can be considered as passed if the results show that the add-on allows the planner to solve the local minima problem.

The local minima issue was solved by designing an obstacle area (no-local-minima area) that included a 45° slope on the left side of the obstacle, see **Figure 5.10** in **Section 5.1**.

The test was to generate a path from a starting position to a target position without falling into local minima. This was repeated ten times, where the position of the obstacles were changed. In the following graphs, the paths generated for the different configurations can be seen.

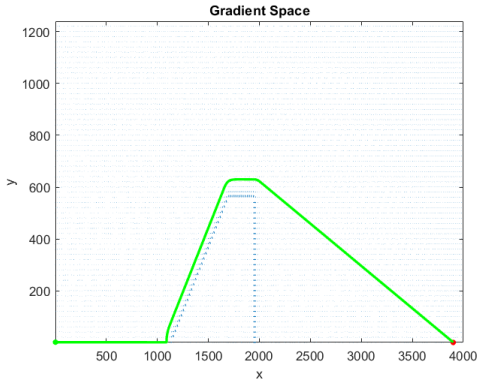


Figure 6.14: First Iteration.

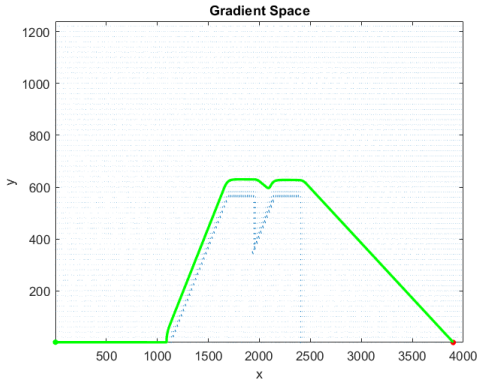


Figure 6.15: Second Iteration.

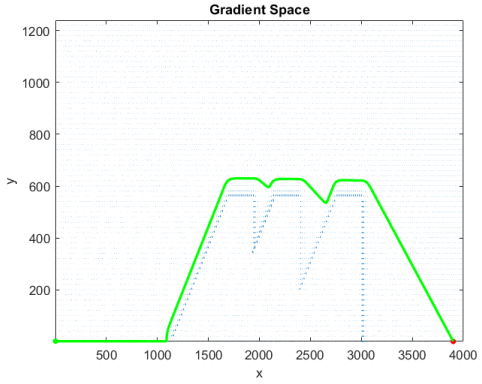


Figure 6.16: Third Iteration.

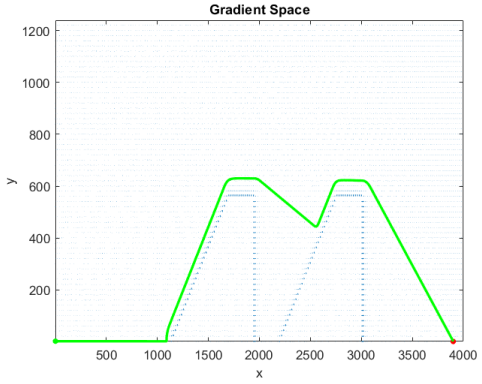


Figure 6.17: Fourth Iteration.

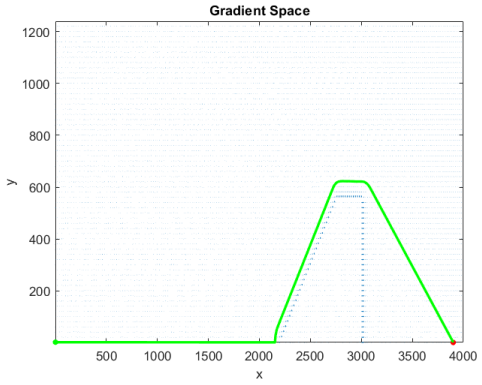


Figure 6.18: Fifth Iteration.

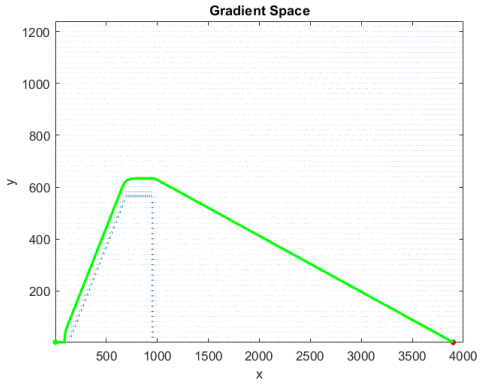


Figure 6.19: Sixth Iteration.

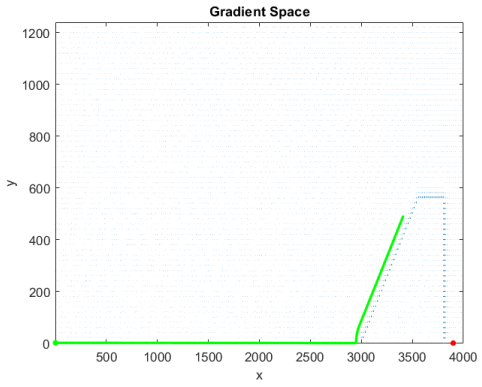


Figure 6.20: Seventh Iteration.

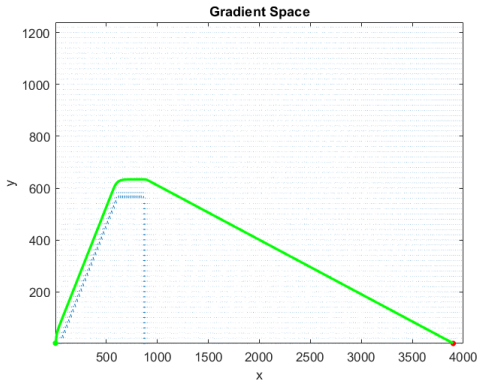


Figure 6.21: Eighth Iteration.

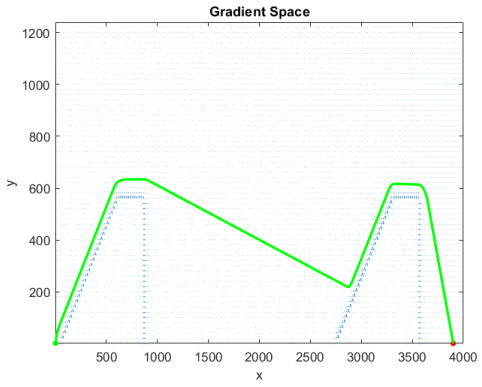


Figure 6.22: Ninth Iteration.

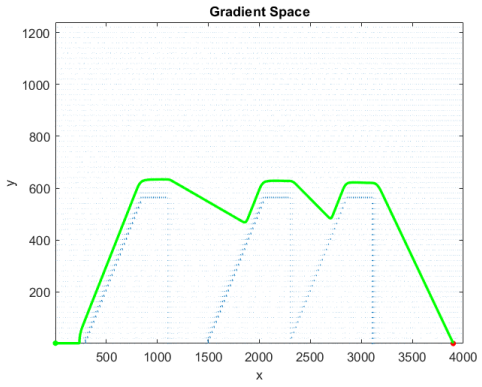


Figure 6.23: Tenth Iteration.

For nine out of ten iterations of the test, the algorithm was capable of generating a path from the start position to the goal. **Figure 6.20** shows an instance where the solution reached a standstill. The reason for this will be discussed in **Chapter 7**. Without the added no-local-minima area, the planner would always get stuck in a local minima, as shown in **Figure 6.24**.

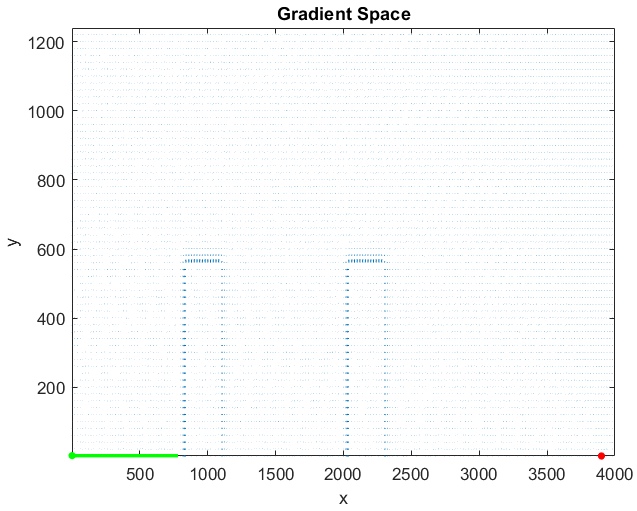


Figure 6.24: Example of the outcome when the no-local-minima area is not implemented.

The approach has shown its ability to prevent the planner from getting stuck on a local minima nine out of ten times. Therefore, the requirement has been met.

Test 3.2

Requirement 3.3: The goal of the path planning is to derive the shortest path from start to end, therefore, the planner is only allowed to be 10cm longer than the shortest path.

The test designed was:

3.2	Place 1-3 obstacles in different locations of the workspace. Run the program for 10 distinct iterations, and store the results.	This test examines the ability of the Potential Field Path Planner to generate a path from start to goal. This will be compared to a manually generated path, considered to be the benchmark path for the specific workspace layout. This requirement can be considered to be met if the path generated for each iteration using the potential field approach is within 10cm.

In order to test the ability of the potential field algorithm to generate the shortest path for different workspace configuration, a reference path is necessary. For each iteration of the test, obstacles where placed on different positions and a path, considered to be the shortest, generated manually. These are shown in the following graphs as a black line.

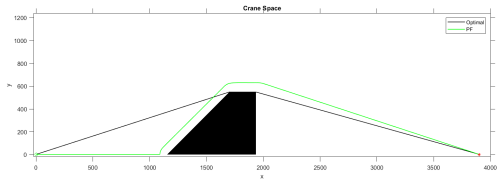


Figure 6.25: PF Path: First configuration.

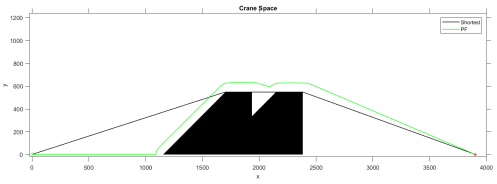


Figure 6.26: PF Path: Second configuration.

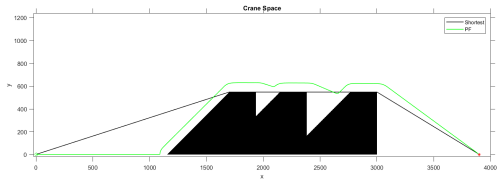


Figure 6.27: PF Path: Third configuration.

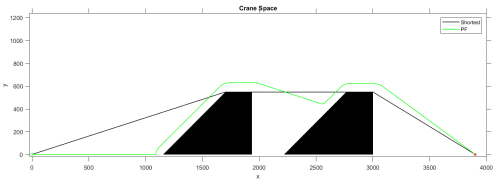


Figure 6.28: PF Path: Fourth configuration.

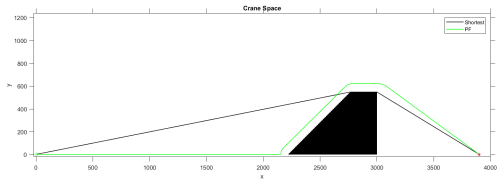


Figure 6.29: PF Path: Fifth configuration.

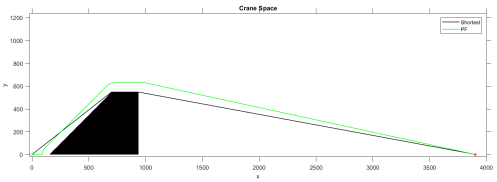


Figure 6.30: PF Path: Sixth configuration.

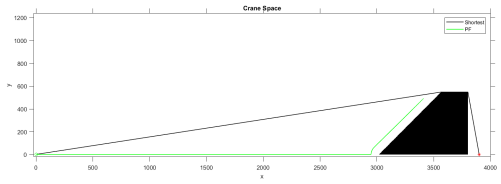


Figure 6.31: PF Path: Seventh configuration.

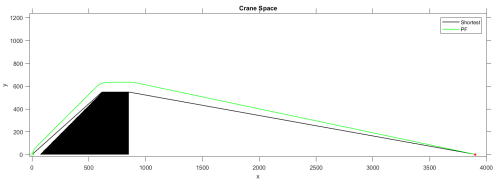


Figure 6.32: PF Path: Eighth configuration.

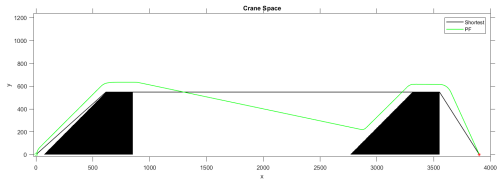


Figure 6.33: PF Path: Ninth configuration.

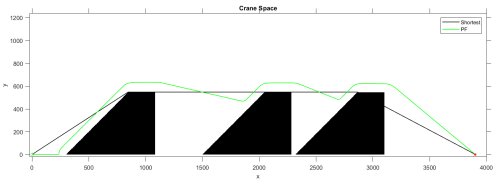


Figure 6.34: PF Path: Tenth configuration.

	Shortest Path	PF Path	Difference
1	4058.2mm	4267.8mm	209.6mm
2	4078.9mm	4306.7mm	227.8mm
3	4136.5mm	4427.3mm	290.8mm
4	4136.5mm	4465.0mm	328.5mm
5	4104.3mm	4365.9mm	261.6mm
6	4136.3mm	4236.0mm	99.7mm
7	4395.5mm	4546.2mm	150.7mm
8	4154.4mm	4234.0mm	79.6mm
9	4404.9mm	4752.0mm	347.1mm
10	4194.6mm	4543.6mm	349mm

Table 6.1: The measurement on the second column, `Shortest Path`, contains the manually generated shortest path per iteration. On its right, the measured distance of the path generated by Potential Field.

Table 6.1 compares the length of each path obtained with the algorithm against the length of the paths made manually. As it can be seen, the smallest difference between the two is 99.7mm, on the sixth iteration. It is also the only one that is within the maximum limit of 10cm. Thus, the test is considered as having failed.

Test 3.3

Requirement 3.3: The goal of the path planning is to derive the shortest path from start to end, therefore, the planner is only allowed to be 10cm longer than the shortest path.

The test designed for this requirement can be seen in the table below.

3.3	Place 1-3 obstacles in different locations of the workspace. Run the program for 10 distinct iterations, and store the results.	This test examines the ability of the ACO Path Planner to generate a path from start to goal. This will be compared to a manually generated path, considered to be the benchmark path for the specific workspace layout. This requirement can be considered to be met if the path generated for each iteration using the ACO approach is within 10cm.

The test was performed the same way as the one made for **Requirement 3.2**. Ten paths for ten different workspaces are generated by the algorithm and compared with their respective shortest path generated manually. The results can be seen in the following figures.

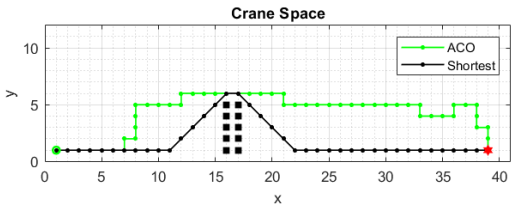


Figure 6.35: ACO Path: First configuration.

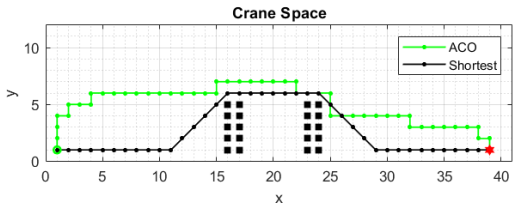


Figure 6.36: ACO Path: Second configuration.

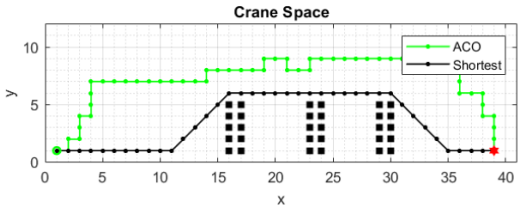


Figure 6.37: ACO Path: Third configuration.

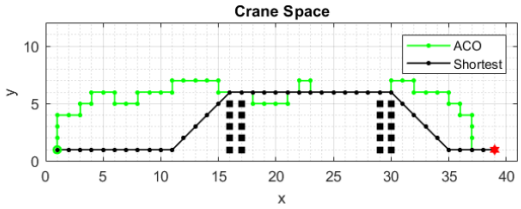


Figure 6.38: ACO Path: Fourth configuration.

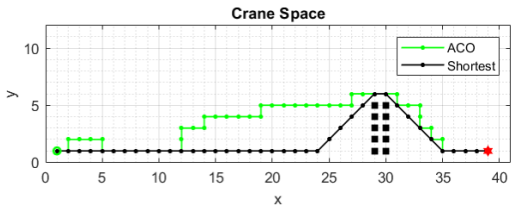


Figure 6.39: ACO Path: Fifth configuration.

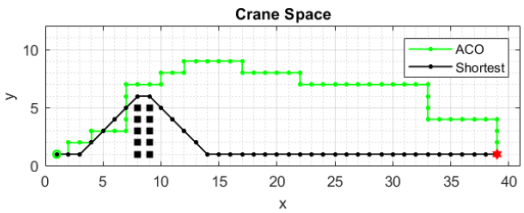


Figure 6.40: ACO Path: Sixth configuration.

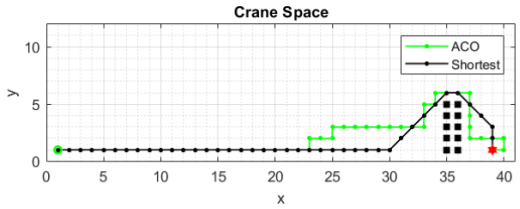


Figure 6.41: ACO Path: Seventh configuration.

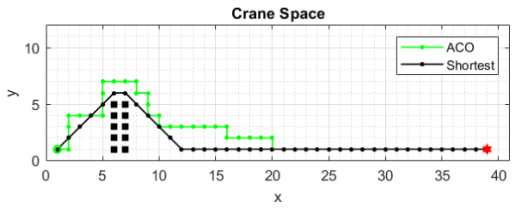


Figure 6.42: ACO Path: Eighth configuration.

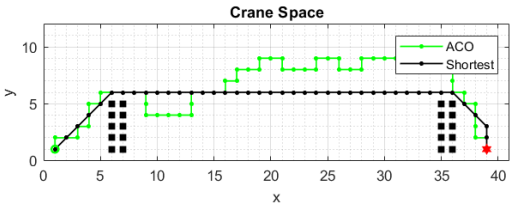


Figure 6.43: ACO Path: Ninth configuration.

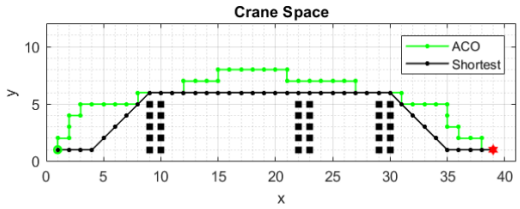


Figure 6.44: ACO Path: Tenth configuration.

The black paths represent the shortest path and the green paths are those generated by the ACO planner. The obstacle configurations depicted are similar to the ones utilised for the previous requirement. In **Table 6.2**, the lengths of both generated paths, manually and by ACO, are listed.

	Shortest Path	**ACO Path**	**Difference**
1	4214.21mm	5000mm	785.79mm
2	4214.21mm	5000mm	785.79mm
3	4214.21mm	5600mm	1385.79mm
4	4214.21mm	5800mm	1585.79mm
5	4214.21mm	5000mm	785.79mm
6	4214.21mm	5400mm	1185.79mm
7	4331.37mm	5000mm	668.63mm
8	4214.21mm	5000mm	785.79mm
9	4331.37mm	6200mm	1868.63mm
10	4214.21mm	5200mm	985.79mm

Table 6.2: The measurement on the second column, `Shortest Path`, contains the manually generated shortest path per iteration. On its right, the measured distance of the path generated by ACO.

The results seen in the above figure show the difference between the manually generated paths and the ACO-generated paths. Since the difference between the two is always above the maximum allowed deviation of 10c, the test has been deemed as not passed.

Test 3.4

Requirement 3.4: Compare the results of both algorithms to discover their benefits and disadvantages against each other.

The test designed for this requirement can be seen in the table below.

3.4	Compare the data obtained when testing **Requirements 3.2** and **3.3**.	One of the goals of this project is to discover which algorithm is more suitable for the solution. This test will result in a decision as to which algorithm will be part of the final solution. The algorithm that derives the shortest paths, compared with their respective manually generated paths, will be considered to be the optimal path planner for this project.

The results of **Tables 6.1** and **6.2** have been gathered on the following table.

	Difference PF Path	**Difference ACO Path**
1	209.6mm	785.79mm
2	227.8mm	785.79mm
3	290.8mm	1385.79mm
4	328.5mm	1585.79mm
5	261.6mm	785.79mm
6	99.7mm	1185.79mm
7	150.7mm	668.63mm
8	79.6mm	785.79mm
9	347.1mm	1868.63mm
10	349mm	985.79mm

Table 6.3: The differences shown are those obtained in the previous requirements, from **Tables 6.1** and **6.2**.

As depicted by **Table 6.3**, the potential field planner is closer to its manually generated paths in all ten iterations of the test. Therefore, as stated in **Requirement 3.4** the selected algorithm for the final solution is the Potential Field Path Planner.

6.5 System Requirements

The tests designed for the system as a whole will be described in this section, together with the results for each of them.

Test 4.1

Requirement 4.1: The control signals derived from the MPC must be followed by the physical crane.

The test designed was:

4.1	Make MPC move the crane to a point in the workspace and check that it can stabilise in said position, for both x and l.	The MPC controller generated from the linear model and tuned to move the nonlinear model, is transferred to the script communicating with the crane. A control signal is generated using the Matlab function *mpcmove*, with the relevant parameters as well as a reference of 1.

This test was performed for the simple scenario of moving x from an initial position of around 0.2 to a position of 1. The results can be seen below in **Figure 6.45**

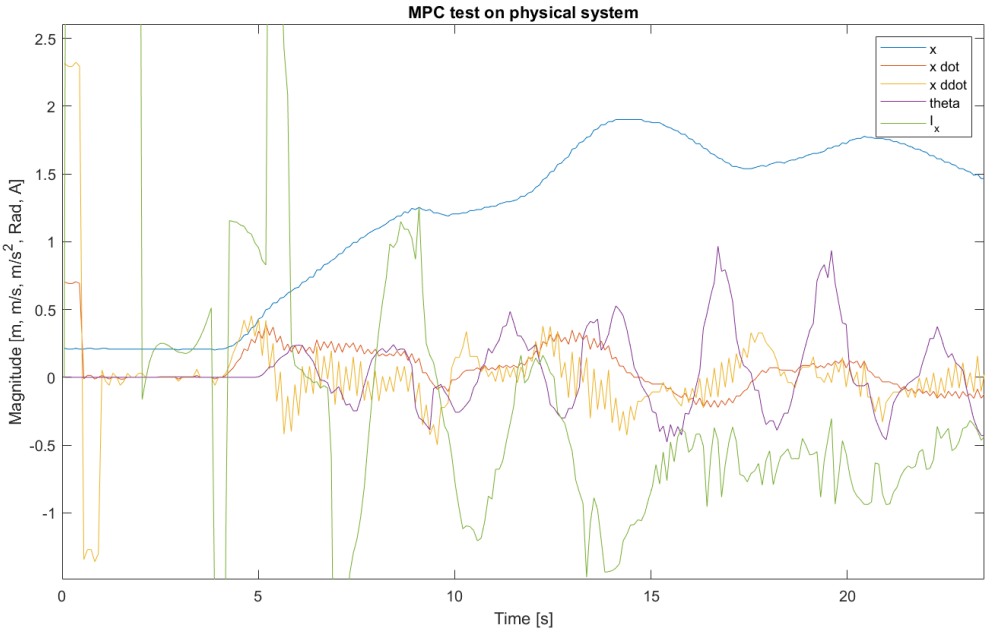


Figure 6.45: This graph shows the MPC trying to move the trolley to an x position of 1 while stabilizing theta.

As can be seen, the MPC moves the trolley past the reference and spends more than 10 seconds moving slowly back and forth, while θ grows increasingly unstable. Different MPCs

and different references have been tried with similar results. Thus it can be concluded that this requirement is not passed.

Test 4.2

Requirement 4.2: The overall system consists of three subsystems that must collaborate with each other for a container crane to perform the task of moving containers.

The test designed was:

4.2	The task of moving a container from one location to another must be completed. Said task should begin by computing a path from start to finish, which the controller must then pass on to the crane as control signals. The crane must pick up the object from the initial location and place it at the desired end position.	For the requirement to be met, the subsystems must be capable of interacting with each other to perform the desired task.

The test could not be performed as the MPC cannot reasonably control the crane. It remains an option to transport containers along a path using a simpler control system like a PD controller, but this avenue was not explored further as the main focus was on the MPC control.

Chapter 7

Discussion

The testing phase of this project yielded the results shown in the previous chapter, **Chapter 6**. In this chapter, the results considered to have meaningful outcomes as well as some other topics that appeared during the implementation of this solution will be discussed.

7.1 MPC

The requirements set for the MPC were tested on the nonlinear model, as it is considered to simulate more closely the real crane's response to input signals.

Requirement 1.1

This first requirement stated that the MPC should be capable of controlling the nonlinear model for reaching steady state in x and l. To test this, a reference position for the mentioned states was given and the output response was recorded. The MPC showed that it could control l without much of a problem, reaching steady state in both graphs shown. On the other hand, for x, it performed worse.

The control of the system was separated early in the project, to: control on x and control on l. For the second one, the solution has not shown to have any difficulty at deriving a control signal which adequately performs the tasks given to the system. The other control, I_x, has a more complex description. This is due to it controlling two states at the same time, x and θ. The graphs shown for **Requirement 1.1** depicted two version of the same task, one where I_x was tuned to better control x and one where the focus of the tuning was on θ. When controlling θ more aggressively, x had difficulties reaching the desired position but θ had a lower oscillation rate compared to the other graph. When controlling x more aggressively, θ oscillated much more and x could reach steady state much faster.

It can be discussed that focusing the control on x yielded results that can be considered

acceptable for the real crane as, even though θ oscillated much more, the maximum angle the crane's head reached was less than 23 degrees. The real effect of this oscillation should be tested in order to better evaluate the need for an extended research on anti-sway angle control for the gantry crane.

Requirement 1.2

As a continuation of the previous requirement, **Requirement 1.2** delved into the ability of the MPC to follow a desired path. Three paths were chosen, one was directly obtained from the potential field path planner and the other two were manually generated. The MPC showed that it could follow the path planner's trajectory with a maximum deviation of approximately 0.12m which is lower than the other two paths. **Figure 6.10** had a large spike at around 60 seconds and it would have been within the same maximum deviation as the 0.12m on the path planner's trajectory. This large deviation can be the result of an unrealistic path design. The manually designed path did not take into account the limits on the velocity that the actual crane imposed on the model. Therefore, since the model could never perform the change in position as fast as the desired path required, such an event must be considered when computing a path for the solution.

7.2 Hardware

The tests made for this subsystem focused on sensor measurement and communication between the different system components. All of the requirements in this section have been deemed as fulfilled.

One aspect of discussion that arose from these tests has to do with the additional states that are used for data processing. The sensors that allowed to obtain all directly measurable states are the potentiometers, the tachometers and the IMU. However, multiple other states are desired, and have, therefore, been derived from these measurements. An example of the estimated states, which are obtained as time derivatives of the measured states, can be seen in **Figure 7.1**.

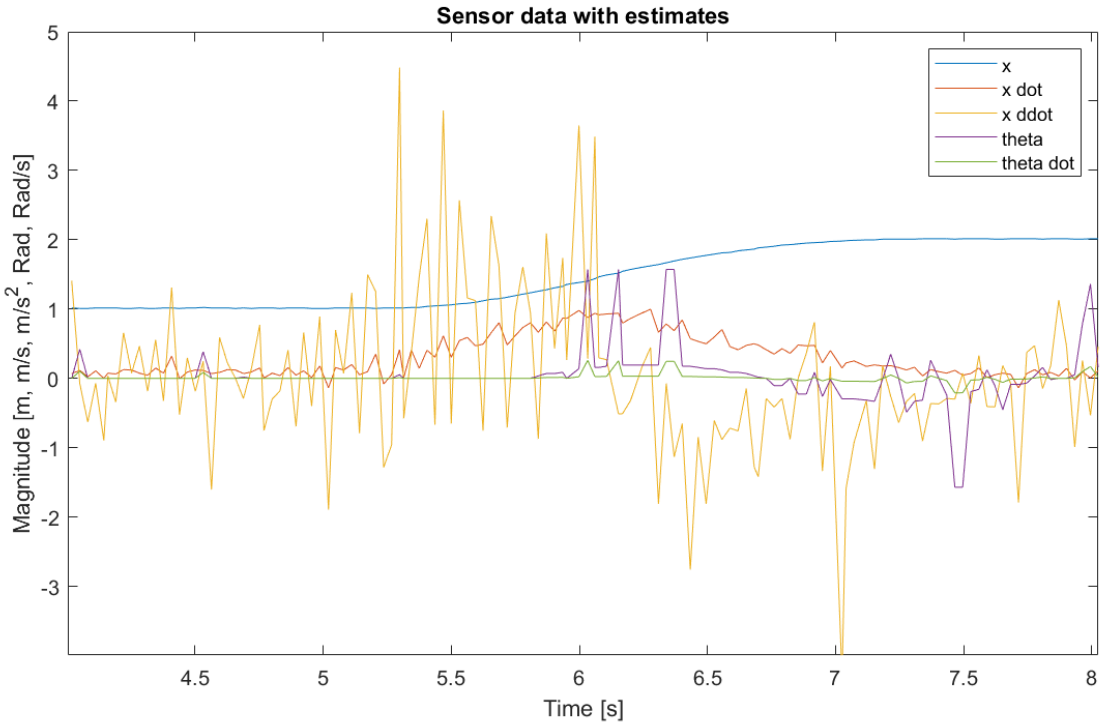


Figure 7.1: This figure shows a simple crane movement with the measured states shown, with the addition of estimated trolley acceleration, $\ddot{x}$, and estimated angular velocity, $\dot{\theta}$.

However, as it is apparent on the graph, both the measured and estimated states are subject to noise. And as it is a part of each sensor based hardware requirement, that the states must represent the real system, efforts have been made to reduce said noise. **Figure 7.2** shows an example of the implemented noise reduction.

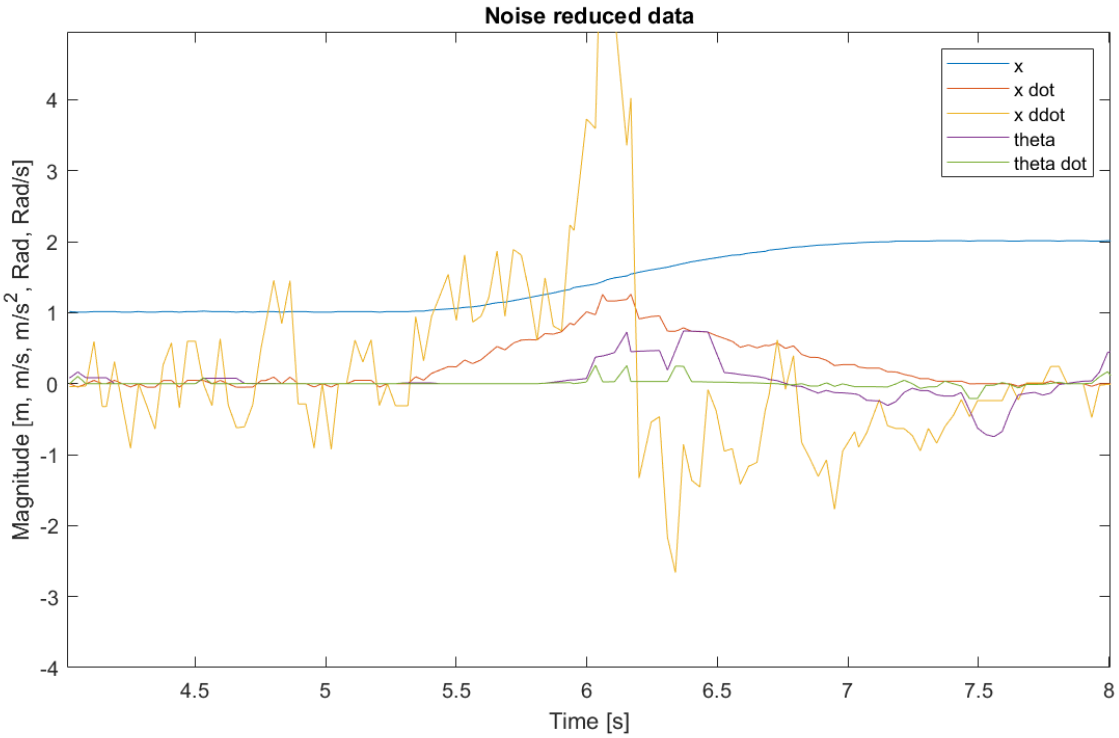


Figure 7.2: This figure shows the same test as in **Figure 7.1**, but with noise reduced versions of all the states, and with potentiometer derived trolley velocity and acceleration.

By comparing **Figures 7.1** and **7.2**, it can be seen that the signals on **Figure 7.2** are less noisy. This was achieved by implementing simple high/low pass filters, as well as a moving average filter on most of the signals. In addition, $\dot{x}$ and $\ddot{x}$ come from first and second order time derivatives of the potentiometer measurements, instead of the tachometer, since the tachometer measurements were deemed to be too noisy.

The same methods were applied to l, and these noise reduced state measurements and estimates are what is used in the other tests in this project, rather than the raw measurements.

7.3 Path Planning

Both path planning algorithms, ACO and PF, were tested in the previous chapter. The results are discussed in this section.

Requirement 3.1

For this requirement, the focus was on testing the ability of the designed solution to avoid falling into local minima and not being capable of leaving it. The results showed that placing the obstacles at certain positions generated a local minima. The following graphs show four paths where the obstacle is placed close to the target.

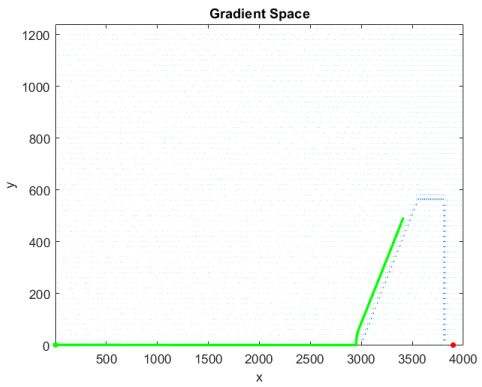


Figure 7.3: Position: 3800

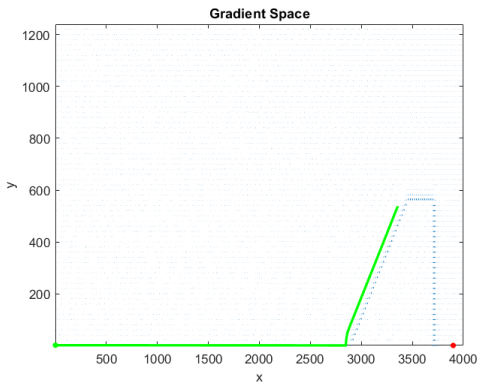


Figure 7.4: Position: 3700

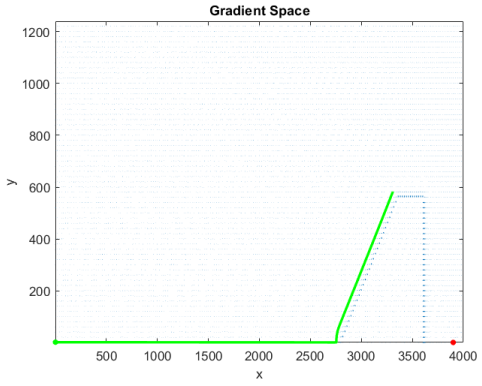


Figure 7.5: Position: 3600

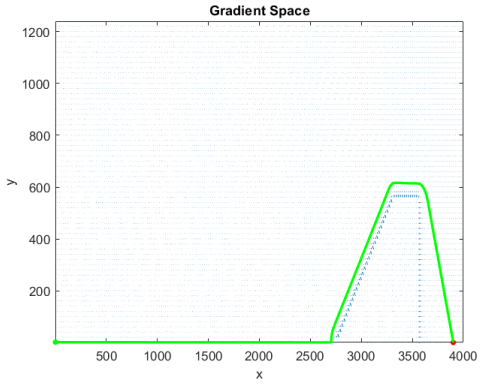


Figure 7.6: Position: 3550

Each graph shows an obstacle that has been moved away from the target coordinate. **Figure 7.3** has its rightmost face at 3800mm on the workspace. **Figures 7.4** and **7.5** are spaced 100mm from each other, at 3700mm and 3600mm. On all three, the algorithm gets stuck. The last graph, **Figure 7.6**, is the first obstacle placement where the algorithm can escape local minima and reach the goal coordinate.

This issue showed that the placement of obstacles in the workspace, and with the designed no-local-minima area, is important. For the approach designed in this project, the closest obstacle must be at least 350mm away from the target. In **Figure 7.7** an example of

the target between obstacles is shown.

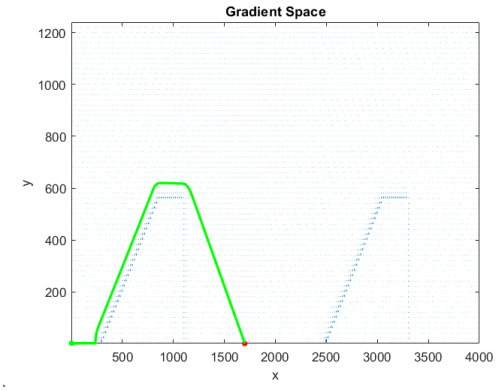


Figure 7.7: Target placed between two obstacles.

One issue which has not been mentioned is the placement of an obstacle on top of the target. This would not be possible in the physical world but, when simulating, the no-local-minima area an fall on top of a desired goal coordinate. This could be solved by removing said area to all obstacles on the right side hand of the target.

Requirements 3.2 and 3.3

Both requirements have been deemed as failed as none of the implemented path planners are capable of following the manually made shortest path with less than 10cm of deviation. Failing these tests does not signify the ability of the planners to design a path between two points while avoiding obstacles but it shows that further research on the subject is needed if the crane must follow the shortest path.

Requirement 3.4

The test was to compare the paths obtained from both path planners in the previous tests. The comparison showed that the average difference between the best path and the ones computed by the algorithms was, 234.44mm average deviation for the PF planner and 1082.358mm average deviation for the ACO planner. This meant that the PF planner was selected as a more suitable option for this project than the ACO approach.

One way which could have improved the outcome for both approaches could be to introduce a series of conditions to the path planning, specific for each algorithm. For example, for the ACO approach, introducing a weighting system to the reachable nodes in the workspace could speed up the time it takes the ants to reach the goal, by assigning increasingly higher weights the higher up in the y-axis the nodes are.

7.4 System

The MPC was able to move the crane, but it was not able to stabilise it at the reference point. The exact reason for this problem was not discovered. However, it is suspected that it may have to do with the use of the `mpcmove` command in Matlab. The Matlab command has many inherent features and assumptions, which could have cause the issue. And it is not very transparent, so it was difficult to figure out the root of the problem.

One possible solution could have been to have manually implemented the control law and manually solve the cost function. This way a number of parameters would have been directly adjustable, which might have helped understand and correct the issue.

Since the system was not able to move to a reference point using the MPC, it was not able to follow a trajectory with it either. And the test was therefore an automatic failure. However, the trajectory was generated, control inputs were able to be generated, the crane was able to respond, albeit incorrectly, and measurements were recorded. As mentioned, the issue was not found in time, and efforts were instead focused on making the simulated MPCs and models function as desired, in the final parts of the project period.

Chapter 8

Conclusion

Increasingly complex problems are being automated and the potential benefits for the shipping industry are significant. Fuji Electric Co. proposed researching this automation issue.
The task was sectioned into the more specific problems of generating a path, making an appropriate control system and implementing both of these on a physical model at the AAU Control Lab.

Several path planning methods were researched and two were chosen and implemented. The potential fields method produced reliable, expected paths which could be successfully followed. Ant colony was tried for its theoretical optimal shortest path, but was not fully realised.
A few control systems were considered and an MPC was chosen for its high potential and its ability to handle complex dynamical systems. It was successfully implemented in multiple variants of both linear and nonlinear simulated gantry cranes. It was however never able to correctly control the physical crane. One of the necessary topics that had to be researched before designing the MPC was the linearisation of the dynamic model. Two approaches for linearising the model, Computed Torque and Taylor/Laplace, were implemented but it was the Taylor/Laplace method was ultimately selected.

The physical system architecture was successfully implemented. The communication between each of the components was functional. Each of the measurable states were made available to the controller and continuously updated. And each of the controllable states were controlled, as shown with a simple PD controller, despite the unsuccessful implementation of the desired MPC on the physical system.

Future work

Full implementation of the MPC on the physical system is an obvious task for the future, along with its variants and different optimization algorithms. Nonlinear dynamics for a 3D gantry crane was made during this project but were never implemented as the scope was reduced. A truly optimal path generation is also desired, derived from distance, time or energy etc.

Appendix A

Voltage Dividers

The following section describes a set of voltage dividers which are no longer used in the hardware setup. However, the explanations about voltage dividers in general are still relevant, as voltage division is used in the current op amp circuit for the tachometers.

Voltage Dividers

The output signals from the tachometers are high enough voltage and current to potentially damage the Arduino micro controller. Therefore, voltage dividers have been made and implemented to account for this. The voltage dividers each consist of 2 resistors, R_1 and R_2, and a GND connection. They receive the tachometer signals as input, and output a downscaled output signal, V_1, to the micro controller. The absolute maximum rating for operating voltage of the micro controller I/O pins is 5.5V. For safety, the target output voltage when choosing the resistors for the voltage dividers has been rounded down to $V_1 = 5$V. The absolute maximum rating for electrical current per I/O pin on the micro controller is 40mA. However, the recommended maximum is 20mA, which will be the target current. [51]

The pins of the micro controller each utilize an internal pull-up resistor, $R_3 = 20\text{M}\Omega$, which can be treated as a parallel resistor to R_2. The parallel resistors have an equivalent resistance, R_{eq}, which will also be calculated (**Equation A.3**).

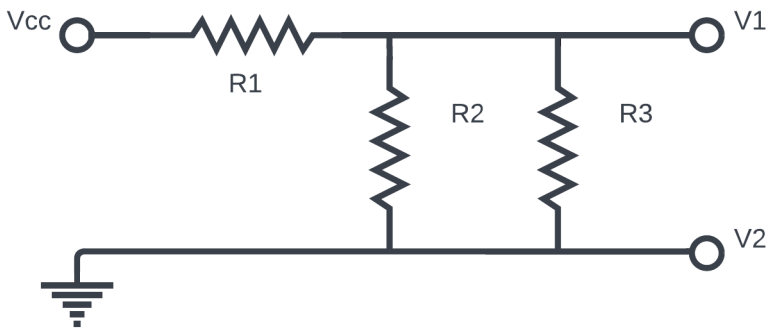


Figure A.1: This figure shows the voltage divider used for each tachometer signal in the system. V_{CC} is the voltage of the input tachometer signal, and V_1 and V_2 are the output voltages after R_1 and R_{eq} respectively. Where R_{eq} is the equivalent resistance of parallel resistors, R_2 and R_3. And R_1 is the first resistor.

The input voltage, which is the maximum voltage of the tachometer output signal, is $V_{CC} = 26\text{V}$. And V_2 is known to be equal to 0V, since the sum of voltage drops across all components in the system is equal to the input voltage, according to Kirchoff's second law. The desired resistance of the resistors with the previously mentioned desired output voltage, $V_1 = 5\text{V}$ and current $I = 20\text{mA}$, can now be calculated. From Ohm's law, we get:

$$R = \frac{V}{I} \tag{A.1}$$

And by considering each part of the system separately, we can extrapolate:

$$R_1 = \frac{V_{CC} - V_1}{I} = 1050\Omega \tag{A.2}$$

$$R_{eq} = \frac{V_1 - V_2}{I} = 250\Omega \tag{A.3}$$

R_2 can then be derived from the equivalent resistance in **Equation A.4**, based on Kirchhoff's current law;

$$\begin{aligned} R_{eq} &= \left(\frac{1}{R_2} + \frac{1}{R_3}\right)^{-1} \\ \frac{1}{R_2} &= \frac{1}{R_{eq}} - \frac{1}{R_3} \\ R_{eq} \cdot R_3 &= R_{eq} \cdot R_3 - R_2 \cdot R_3 \\ R_2 &= \frac{R_{eq} \cdot R_3}{R_3 - R_{eq}} = 250.003\Omega \end{aligned} \tag{A.4}$$

However, this results in R_2 being nearly equal to R_{eq}, since $R_3 = 20\text{M}\Omega$ has such a high resistance in comparison. The 0.003Ω difference is considered insignificant, and a 250Ω resistor will be used for R_2.

The results can be verified using ohm's law:

$$\begin{aligned} I &= \frac{V_{CC}}{R_1 + R_{eq}} = 0,02\text{A} \\ V_{CC} &= I \cdot (R_1 + R_{eq}) = 26\text{V} \\ V_1 &= V_{CC} - I \cdot R_1 = 5\text{V} \\ V_2 &= V_1 - I \cdot R_{eq} = 0\text{V} \end{aligned} \tag{A.5}$$

And the results from **Equation A.5** match the desired values.

The following figure shows the voltage drops across the resistors, R_1 and R_{eq}, which are equal to $I \cdot R_1 = 21\text{V}$ and $I \cdot R_{eq} = 5\text{V}$ respectively. [57] [51]

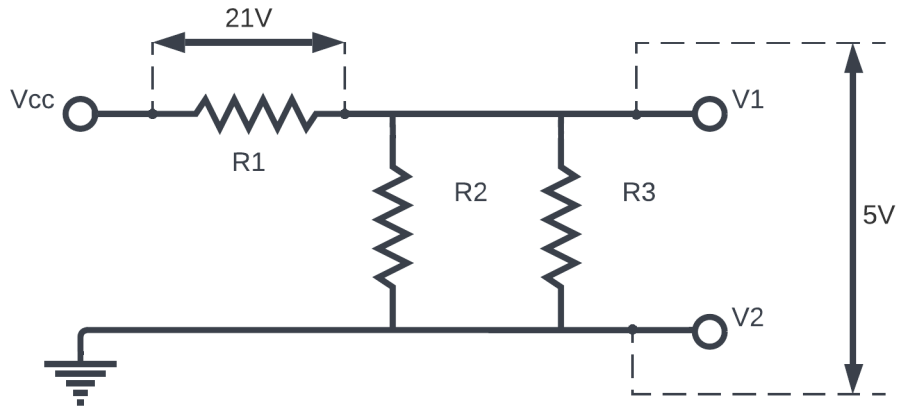


Figure A.2: The voltage drops across R_1 and R_{eq} in the voltage dividers used for the tachometer signals. Where R_{eq} is the equivalent resistance of parallel resistors, R_2 and R_3.

Appendix B

Model of overhead crane

This appendix includes an extended version for the computation of overhead crane's model in **Chapter 3**, **Section 3.3**.

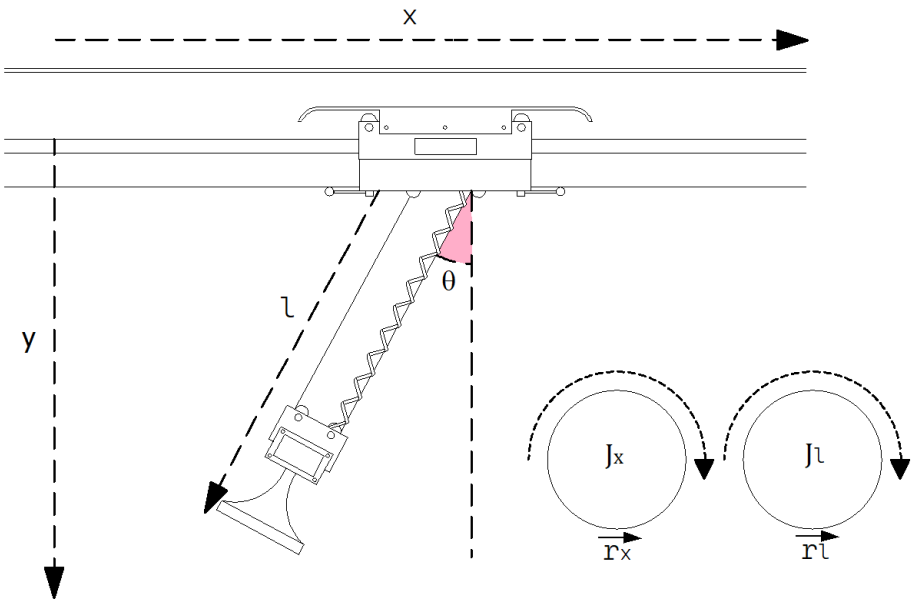


Figure B.1: Head and Trolley with angles and positions

where	x	:	Position on the horizontal axis	(m)
	y	:	Position on the vertical axis	(m)
	l	:	Length of the wire	(m)
	θ	:	Angle between the load and its equilibrium point	$(°)$
	J_x	:	Moment of Inertia for the x direction motor	$(\cdot)$
	J_l	:	Moment of Inertia for the wire length motor	$(\cdot)$
	r_x	:	Radius of the X direction drum	(m)
	r_l	:	Radius of the wire-length drum	(m)

The model will be derived using Lagrange equations, which start by describing the systems kinetic and potential energies, which in turn require the positional equation shown below in **Equation B.1**.

$$\vec{p}_t = x_t \cdot \vec{i} + y_t \cdot \vec{j} \,|_{y_t=0} \tag{B.1}$$

where	$\vec{p}_t$	:	Vector position of the trolley	(m)
	x_t	:	Trolley x position	(m)
	$\vec{i}$	:	Unit vector for x	$(\cdot)$
	y_t	:	Trolley y position	(m)
	$\vec{j}$	:	Unit vector for y	$(\cdot)$

Only a 2-D crane is considered for now, resulting in the y component for the trolley being 0.
The position of the load is given in **Equation B.2**.

$$\begin{aligned} \vec{p}_l = x_l \cdot \vec{i} + y_l \cdot \vec{j} &= (x_t + l \cdot sin(\theta)) + (y_t + l \cdot cos(\theta)) \,|_{y_t=0} \\ &= (x_t + l \cdot sin(\theta)) + l \cdot cos(\theta) \end{aligned} \tag{B.2}$$

where	$\vec{p}_l$	:	Vector position of the trolley	(m)
	x_l	:	Trolley x position	(m)
	y_l	:	Trolley y position	(m)
	l	:	Length of the wire	(m)
	θ	:	Angle between trolley and load	$(°)$

With the positional descriptions in place, the Lagrangian can be derived. The kinetic energy of the system comes from four sources, namely the trolley and the load, as well as their respective servo motor drums.
The kinetic energy of the trolley:

$$T_t = \frac{1}{2} \cdot m_t \cdot \dot{x}_t^2 \tag{B.3}$$

where	T_t	:	Kinetic energy of the trolley	(J)
	m_t	:	Mass of trolley	(kg)
	$\dot{x}$	:	X velocity of trolley	(m/s)

Kinetic energy of the load:

$$T_l = \frac{1}{2} \cdot m_l \cdot \left(\dot{x}_l^2 + \dot{y}_l^2\right) \tag{B.4}$$

where
T_l : Kinetic energy of the load (J)
m_l : Mass of load (kg)
$\dot{x}$: X velocity of load (m/s)
$\dot{y}$: Y velocity of load (m/s)

The kinetic energy of the rotating wire drums are calculated as a rotational kinetic energy instead, which substitutes the mass term for the moment of inertia of the body and the velocity for angular velocity of the drum. This angular velocity is not a measured quantity in this project, and is thus related to the linear velocity of the wire and the drums radius. The kinetic energy of the drum for x is presented in **Equation B.5**.

$$T_{px} = \frac{1}{2} \cdot J_x \cdot \left(\frac{\dot{x}}{r_x}\right)^2 \tag{B.5}$$

where
T_{px} : Kinetic energy of the drum for x (J)
J_x : Moment of Inertia of the drum for x $(kg \cdot m^2)$
r_x : radius of the drum for x (m)

The kinetic energy of the drum for hoisting is presented in **Equation B.6**.

$$T_{pl} = \frac{1}{2} \cdot J_l \cdot \left(\frac{2\dot{l}}{r_l}\right)^2 \tag{B.6}$$

where
T_{pl} : Kinetic energy of the drum for l (J)
J_l : Moment of Inertia of the drum for l $(kg \cdot m^2)$
r_l : radius of the drum for l (m)

The "2" is inserted because the gearing in the drum for the hoisting wire is slightly different, resulting in a different approximation.
The kinetic energy of the system is the sum of its parts and is thus given in **Equation B.7**;

$$T = \frac{1}{2} \cdot m_t \cdot \dot{x}_t^2 + \frac{1}{2} \cdot m_l \cdot \left(\dot{x}_l^2 + \dot{y}_l^2\right) + \frac{1}{2} \cdot J_x \cdot \left(\frac{\dot{x}}{r_x}\right)^2 + \frac{1}{2} \cdot J_l \cdot \left(\frac{2\dot{l}}{r_l}\right)^2 \tag{B.7}$$

The second term of this equation describes the kinetic energies of the system relating to the velocity of the load, which is not a measured parameter in this project. It can however be further related to the velocity of the trolley using **Equation B.2**, also shown below in **Equation B.8**.

$$\vec{p}_l = (x_t + l \cdot sin(\theta)) + l \cdot cos(\theta) \tag{B.8}$$

Isolating the term representing x_l,

$$x_l = (x_t + l \cdot sin(\theta)) \tag{B.9}$$

Differentiating with respect to time, using chain rule,

$$\frac{dx_l}{dt} = \frac{d\,(x_t + l \cdot sin(\theta))}{dt} \tag{B.10}$$

$$\frac{dx_l}{dt} = \frac{d\,(x_t + l \cdot sin(\theta))}{dx_t} \cdot \frac{dx_t}{dt} + \frac{d\,(x_t + l \cdot sin(\theta))}{dl} \cdot \frac{dl}{dt} + \frac{d\,(x_t + l \cdot sin(\theta))}{d\theta} \cdot \frac{d\theta}{dt} \tag{B.11}$$

$$\dot{x_l} = 1 \cdot \dot{x_t} + sin(\theta) \cdot \dot{l} + l \cdot \dot{\theta} \cdot cos(\theta) \tag{B.12}$$

Squaring both sides,

$$\dot{x_l}^2 = \dot{x_t}^2 + \dot{l}^2 sin^2(\theta) + l^2\dot{\theta}^2 cos^2(\theta) + 2\dot{x_t}\dot{l} sin(\theta) + 2\dot{x_t} l\dot{\theta} cos(\theta) + 2\dot{l} l\dot{\theta} sin(\theta) cos(\theta) \tag{B.13}$$

The same is done for the term representing y_l,

$$y_l = l \cdot cos(\theta) \tag{B.14}$$

$$\dot{y_l} = \dot{l} \cdot cos(\theta) - l \cdot \dot{\theta} \cdot sin(\theta) \tag{B.15}$$

Squaring both sides again,

$$\dot{y_l}^2 = \dot{l}^2 cos^2(\theta) + l^2\dot{\theta}^2 sin^2(\theta) - 2l\dot{l}\dot{\theta} cos(\theta) sin(\theta) \tag{B.16}$$

Equation B.7 contains $\dot{x_l}^2 + \dot{y_l}^2$, which is now calculated below.

$$\dot{x_l}^2 + \dot{y_l}^2 = \dot{x_t}^2 + \dot{l}^2 + l^2\dot{\theta}^2 + 2\dot{x_t}\dot{l} sin(\theta) + 2\dot{x_t} l\dot{\theta} cos(\theta) \tag{B.17}$$

The complete equation for the kinetic energies is now acquired. The next step is deriving the potential energy.
If origin for y is set at the trolleys level, it effectively does not have any potential energy, leaving only the load, which is also defined as having zero potential energy when it is at $y = 0$. It is thus given as:

$$V = -m_l \cdot g \cdot y_l = -m_l \cdot g \cdot (y_t + l \cdot cos(\theta))\,|_{y_t=0} = -m_l \cdot g \cdot l \cdot cos(\theta) \tag{B.18}$$

where V : Potential energy of system (J)
g : Gravity constant = 9.82 $(m \cdot s^{-2})$

The Lagrangian L for the system can now be presented:

$$\begin{aligned} L = T - V = {} & \frac{1}{2} m_t \dot{x_t}^2 + \frac{1}{2} m_l \left(\dot{x_t}^2 + \dot{l}^2 + l^2\dot{\theta}^2 + 2\dot{x_t}\dot{l} sin(\theta) + 2\dot{x_t} l\dot{\theta} cos(\theta)\right) \\ & + \frac{1}{2} J_x \left(\frac{\dot{x}}{r_x}\right)^2 + \frac{1}{2} J_l \left(\frac{2\dot{l}}{r_l}\right)^2 + m_l \cdot g \cdot l \cdot cos(\theta) \end{aligned} \tag{B.19}$$

With this, the equations of motion can be found. Three variables are present, so an equation for each is needed. The Lagrangian equation when no external force is present is defined as:

$$0 = \frac{d}{dt}\frac{\partial}{\partial \dot{p}}L - \frac{\partial}{\partial p}L \tag{B.20}$$

The Lagrangian equation when there is an external force present is defined as:

$$F_z = \frac{d}{dt}\frac{\partial}{\partial \dot{p}}L - \frac{\partial}{\partial p}L \tag{B.21}$$

Equation of motion for $\ddot{x}$

The servo motor pulling the trolleys wire is considered an external force, meaning **Equation B.21** is used. The force can represented as shown below in **Equation B.22** together with a friction term,

$$F_x = \frac{K_{ex} \cdot I_x}{r_x} - B_x \cdot \dot{x}_t \tag{B.22}$$

where

F_x	:	Force acting along X	(N)
K_{ex}	:	Motor constants for trolley motor	$(\cdot)$
I_x	:	Current delivered to trolley motor	(A)
r_x	:	Radius of trolley drum	(m)
B_x	:	Friction coefficient for moving along X	(N)
$\dot{x}$	:	Velocity of the hoisting system	(m/s)

The Lagrangian equation for x can now be calculated.

$$\frac{K_{ex} \cdot I_x}{r_x} - B_x \cdot \dot{x}_t = \frac{d}{dt}\frac{\partial}{\partial \dot{x}_t}L - \frac{\partial}{\partial x_t}L \tag{B.23}$$

$$\begin{aligned}\frac{\partial}{\partial \dot{x}_t}\Bigg(&\frac{1}{2}m_t\dot{x}_t^2 + \frac{1}{2}m_l\left(\dot{x_t}^2 + \dot{l}^2 + l^2\dot{\theta}^2 + 2\dot{x}_t\dot{l}sin(\theta) + 2\dot{x}_t l\dot{\theta}cos(\theta)\right)\\ &+ \frac{1}{2}J_x\left(\frac{\dot{x}_t}{r_x}\right)^2 + \frac{1}{2}J_l\left(\frac{2\dot{l}}{r_l}\right)^2 + m_l g l cos(\theta)\Bigg)\end{aligned} \tag{B.24}$$

Terms without $\dot{x}_t$ eliminated.

$$\frac{\partial}{\partial \dot{x}_t}\left(\frac{1}{2}m_t\dot{x}_t^2 + \frac{1}{2}m_l\left(\dot{x_t}^2 + 2\dot{x}_t\dot{l}sin(\theta) + 2\dot{x}_t l\dot{\theta}cos(\theta)\right) + \frac{1}{2}J_x\left(\frac{\dot{x}_t}{r_x}\right)^2\right) \tag{B.25}$$

Partial differentiation with respect to $\dot{x}_t$.

$$\begin{aligned}\frac{\partial}{\partial \dot{x}_t}L &= \frac{\partial}{\partial \dot{x}_t}\left(\frac{1}{2}m_t\dot{x}_t^2 + \frac{1}{2}m_l\left({\dot{x}_t}^2 + 2\dot{x}_t\dot{l}sin(\theta) + 2\dot{x}_t l\dot{\theta}cos(\theta)\right) + \frac{1}{2}J_x\left(\frac{\dot{x}_t}{r_x}\right)^2\right) \\ &= \frac{1}{2}m_t 2\dot{x}_t + \frac{1}{2}m_l(2\dot{x}_t + 2\dot{l}sin(\theta) + 2l\dot{\theta}cos(\theta)) + \frac{1}{2}J_x\left(\frac{2\dot{x}_t}{r_x^2}\right) \\ &= m_t\dot{x}_t + m_l(\dot{x}_t + \dot{l}sin(\theta) + l\dot{\theta}cos(\theta)) + \frac{J_x\dot{x}_t}{r_x^2}\end{aligned} \tag{B.26}$$

Now for the time derivative.

$$\begin{aligned}\frac{d}{dt}\frac{\partial}{\partial \dot{x}_t}L &= \frac{d}{dt}\left(m_t\dot{x}_t + m_l(\dot{x}_t + \dot{l}sin(\theta) + l\dot{\theta}cos(\theta)) + \frac{J_x\dot{x}_t}{r_x^2}\right) \\ &= m_t\ddot{x}_t + m_l(\ddot{x}_t + \ddot{l}sin(\theta) + 2\dot{l}\dot{\theta}cos(\theta) + l\ddot{\theta}cos(\theta) \\ &\quad - l\dot{\theta}^2 sin(\theta)) + \frac{J_x\ddot{x}_t}{r_x^2}\end{aligned} \tag{B.27}$$

Only a single term is left in order to complete the Lagrangian, but since no term in the equation contains an x without a derivative it vanishes.

$$\begin{aligned}\frac{\partial}{\partial x_t}L &= \frac{\partial}{\partial x_t}\Bigg(\frac{1}{2}m_t\dot{x}_t^2 + \frac{1}{2}m_l\left({\dot{x}_t}^2 + \dot{l}^2 + l^2\dot{\theta}^2 + 2\dot{x}_t\dot{l}sin(\theta) + 2\dot{x}_t l\dot{\theta}cos(\theta)\right) \\ &\quad + \frac{1}{2}J_x\left(\frac{\dot{x}_t}{r_x}\right)^2 + \frac{1}{2}J_l\left(\frac{2\dot{l}}{r_l}\right)^2 + m_l g l cos(\theta)\Bigg) \\ &= 0\end{aligned} \tag{B.28}$$

Thus a complete Lagrange's equation for the forces acting on x is present in **Equation B.27** and $\ddot{x}$ can be isolated.

$$\begin{aligned}&\frac{K_{ex}I_x}{r_x} - B_x\dot{x}_t = m_t\ddot{x}_t + m_l(\ddot{x}_t + \ddot{l}sin(\theta) + 2\dot{l}\dot{\theta}cos(\theta) \\ &+ l\ddot{\theta}cos(\theta) - l\dot{\theta}^2 sin(\theta)) + \frac{J_x\ddot{x}_t}{r_x^2} \\ &\frac{K_{ex}I_x}{r_x} - B_x\dot{x}_t = m_t\ddot{x}_t + m_l\ddot{x}_t + m_l\ddot{l}sin(\theta) + 2m_l\dot{l}\dot{\theta}cos(\theta) \\ &+ m_l l\ddot{\theta}cos(\theta) - m_l l\dot{\theta}^2 sin(\theta) + \ddot{x}_t\frac{J_x}{r_x^2} \\ &\frac{K_{ex}I_x}{r_x} - B_x\dot{x}_t = \ddot{x}_t\left(m_t + m_l + \frac{J_x}{r_x^2}\right) + m_l\ddot{l}sin(\theta) + 2m_l\dot{l}\dot{\theta}cos(\theta) \\ &+ m_l l\ddot{\theta}cos(\theta) - m_l l\dot{\theta}^2 sin(\theta)\end{aligned} \tag{B.29}$$

$$\ddot{x}_t = \frac{1}{m_t + m_l + \frac{I_x}{r_x^2}} \left(\frac{K_{ex} I_x}{r_x} - B_x \dot{x}_t - m_l \ddot{l} sin(\theta) - 2 m_l \dot{l} \dot{\theta} cos(\theta) \right. \\ \left. - m_l l \ddot{\theta} cos(\theta) + m_l l \dot{\theta}^2 sin(\theta) \right) \tag{B.30}$$

Equation of motion for $\ddot{l}$

For calculating the forces acting along the hoisting wire, the Lagrange's equation will be solved with respect to l. There is a motor applying force on this axis as well, thus the same form of Lagrange is used. The force term is presented in **Equation B.31**;

$$F_l = \frac{2 K_{el} \cdot I_l}{r_l} - B_l \cdot \dot{l} \tag{B.31}$$

where

F_l	:	Force applied by hoisting motor	(N)
K_{el}	:	Motor constants for hoisting motor	$(\cdot)$
I_l	:	Current delivered to the hoisting motor	(A)
r_l	:	Radius of hoisting motor drum	(m)
B_l	:	Friction coefficient for moving along Z	$(\cdot)$
$\dot{l}$	:	Velocity of the hoisting system	(m/s)

Thus the Lagrange equation to solve is shown in **Equation B.32**.

$$\frac{K_{el} \cdot I_l}{r_l} - B_l \cdot \dot{l} = \frac{d}{dt} \frac{\partial}{\partial \dot{l}} L - \frac{\partial}{\partial l} L \tag{B.32}$$

Each term on the right will be calculated individually.

$$\frac{\partial}{\partial \dot{l}} L = \frac{\partial}{\partial \dot{l}} \left(\frac{1}{2} m_t \dot{x}_t^2 + \frac{1}{2} m_l ({\dot{x}_t}^2 + \dot{l}^2 + l^2 \dot{\theta}^2 + 2 \dot{x}_t \dot{l} sin(\theta) + 2 \dot{x}_t l \dot{\theta} cos(\theta)) \right. \\ \left. + \frac{1}{2} J_x \left(\frac{\dot{x}_t}{r_x} \right)^2 + \frac{1}{2} J_l \left(\frac{2\dot{l}}{r_l} \right)^2 + m_l g l cos(\theta) \right) \tag{B.33}$$

Eliminating terms without $\dot{l}$,

$$\frac{\partial}{\partial \dot{l}} L = \frac{\partial}{\partial \dot{l}} \left(\frac{1}{2} m_l \left(\dot{l}^2 + 2 \dot{x}_t \dot{l} sin(\theta) \right) + \frac{1}{2} J_l \left(\frac{2\dot{l}}{r_l} \right)^2 \right) \tag{B.34}$$

Partial differentiation,

$$\frac{\partial}{\partial \dot{l}} L = \frac{1}{2} m_l \left(2\dot{l} + 2 \dot{x}_t sin(\theta) \right) + \frac{1}{2} J_l \left(\frac{8\dot{l}}{r_l^2} \right) \\ = m_l \dot{l} + m_l \dot{x}_t sin(\theta) + J_l \frac{4\dot{l}}{r_l^2} \tag{B.35}$$

Adding time derivative term,

$$\begin{aligned}\frac{d}{dt}\frac{\partial}{\partial \dot{l}}L &= \frac{d}{dt}\left(m_l\dot{l} + m_l\dot{x}_t sin(\theta) + J_l\frac{4\dot{l}}{r_l^2}\right) \\ &= m_l\ddot{l} + m_l\ddot{x}_t sin(\theta) + m_l\dot{x}_t\dot{\theta}cos(\theta) + J_l\frac{4\ddot{l}}{r_l^2}\end{aligned} \tag{B.36}$$

Calculating $\frac{\partial}{\partial l}L$ from **Equation B.32** separately,

$$\begin{aligned}\frac{\partial}{\partial l}L = \frac{\partial}{\partial l}\Bigg(&\frac{1}{2}m_t\dot{x}_t^2 + \frac{1}{2}m_l(\dot{x_t}^2 + \dot{l}^2 + l^2\dot{\theta}^2 + 2\dot{x}_t\dot{l}sin(\theta) + 2\dot{x}_t l\dot{\theta}cos(\theta)) \\ &+ \frac{1}{2}J_x\left(\frac{\dot{x}_t}{r_x}\right)^2 + \frac{1}{2}J_l\left(\frac{2\dot{l}}{r_l}\right)^2 + m_l g l cos(\theta)\Bigg)\end{aligned} \tag{B.37}$$

Eliminating terms without l

$$\frac{\partial}{\partial l}L = \frac{\partial}{\partial l}\left(\frac{1}{2}m_l(l^2\dot{\theta}^2 + 2\dot{x}_t l\dot{\theta}cos(\theta)) + m_l g l cos(\theta)\right) \tag{B.38}$$

Partial differentiation

$$\frac{\partial}{\partial l}L = m_l l\dot{\theta}^2 + m_l\dot{x}_t\dot{\theta}cos(\theta) + m_l g cos(\theta) \tag{B.39}$$

Putting every term back in **Equation B.32**

$$\begin{aligned}\frac{K_{el}I_l}{r_l} - B_l\dot{l} =& m_l\ddot{l} + m_l\ddot{x}_t sin(\theta) + m_l\dot{x}_t\dot{\theta}cos(\theta) + J_l\frac{4\ddot{l}}{r_l^2} \\ &- \left(m_l l\dot{\theta}^2 + m_l\dot{x}_t\dot{\theta}cos(\theta) + m_l g cos(\theta)\right)\end{aligned} \tag{B.40}$$

Isolating $\ddot{l}$

$$\begin{aligned}&\frac{K_{el}I_l}{r_l} - B_l\dot{l} + \left(m_l l\dot{\theta}^2 + m_l\dot{x}_t\dot{\theta}cos(\theta) + m_l g cos(\theta)\right) \\ &- m_l\ddot{x}_t sin(\theta) - m_l\dot{x}_t\dot{\theta}cos(\theta) = \ddot{l}\left(m_l + \frac{4J_l}{r_l^2}\right)\end{aligned} \tag{B.41}$$

$$\ddot{l} = \frac{1}{m_l + \frac{4J_l}{r_l^2}}\left(\frac{K_{el}I_l}{r_l} - B_l\dot{l} + m_l l\dot{\theta}^2 + m_l g cos(\theta) - m_l\ddot{x}_t sin(\theta)\right) \tag{B.42}$$

Equation of motion for $\ddot{\theta}$

Since wind is not considered, there is no active force acting on the angle. Friction and air resistance for dampening the sway is also not considered, thus the resulting Lagrange equation is equal to zero.

$$0 = \frac{d}{dt}\frac{\partial}{\partial\dot{\theta}}L - \frac{\partial}{\partial\theta}L \tag{B.43}$$

Besides that change, the calculations are the same, with each term done separately as before.

$$\begin{aligned}\frac{\partial}{\partial\dot{\theta}}L =& \frac{\partial}{\partial\dot{\theta}}\left(\frac{1}{2}m_t\dot{x}_t^2 + \frac{1}{2}m_l(\dot{x}_t{}^2 + \dot{l}^2 + l^2\dot{\theta}^2 + 2\dot{x}_t\dot{l}sin(\theta) + 2\dot{x}_t l\dot{\theta}cos(\theta))\right.\\ &\left. + \frac{1}{2}J_x\left(\frac{\dot{x}_t}{r_x}\right)^2 + \frac{1}{2}J_l\left(\frac{2\dot{l}}{r_l}\right)^2 + m_l g l cos(\theta)\right)\end{aligned} \tag{B.44}$$

Eliminating terms without $\dot{\theta}$

$$\frac{\partial}{\partial\dot{\theta}}L = \frac{\partial}{\partial\dot{\theta}}\left(\frac{1}{2}m_l\left(l^2\dot{\theta}^2 + 2\dot{x}_t l\dot{\theta}cos(\theta)\right)\right) \tag{B.45}$$

Partial differentiation

$$\begin{aligned}\frac{\partial}{\partial\dot{\theta}}L &= \frac{1}{2}m_l\left(2\dot{\theta}l^2 + 2\dot{x}_t l cos(\theta)\right)\\ &= m_l\dot{\theta}l^2 + m_l\dot{x}_t l cos(\theta)\end{aligned} \tag{B.46}$$

Adding time derivative

$$\begin{aligned}\frac{d}{dt}\frac{\partial}{\partial\dot{\theta}}L &= \frac{d}{dt}\left(m_l\dot{\theta}l^2 + m_l\dot{x}_t l cos(\theta)\right)\\ &= m_l\ddot{\theta}l^2 + 2m_l\dot{\theta}l\dot{l} + m_l\ddot{x}_t l cos(\theta) + m_l\dot{x}_t\dot{l}cos(\theta) - m_l\dot{x}_t l\dot{\theta}sin(\theta)\end{aligned} \tag{B.47}$$

Calculating $\frac{\partial}{\partial\theta}L$ from **Equation B.43** separately

$$\begin{aligned}\frac{\partial}{\partial\theta}L =& \frac{\partial}{\partial\theta}\left(\frac{1}{2}m_t\dot{x}_t^2 + \frac{1}{2}m_l(\dot{x}_t{}^2 + \dot{l}^2 + l^2\dot{\theta}^2 + 2\dot{x}_t\dot{l}sin(\theta) + 2\dot{x}_t l\dot{\theta}cos(\theta))\right.\\ &\left. + \frac{1}{2}J_x\left(\frac{\dot{x}_t}{r_x}\right)^2 + \frac{1}{2}J_l\left(\frac{2\dot{l}}{r_l}\right)^2 + m_l g l cos(\theta)\right)\end{aligned} \tag{B.48}$$

Eliminating terms without θ

$$\frac{\partial}{\partial\theta}L = \frac{\partial}{\partial\theta}\left(\frac{1}{2}m_l\left(2\dot{x}_t\dot{l}sin(\theta) + 2\dot{x}_t l\dot{\theta}cos(\theta)\right) + m_l g l cos(\theta)\right) \tag{B.49}$$

Partial differentiation

$$\frac{\partial}{\partial\theta}L = m_l\dot{x}_t\dot{l}cos(\theta) - m_l\dot{x}_t l\dot{\theta}sin(\theta) - m_l glsin(\theta) \tag{B.50}$$

Every term added back into **Equation B.43**

$$\begin{aligned} 0 =& m_l\ddot{\theta}l^2 + 2m_l\dot{\theta}l\dot{l} + m_l\ddot{x}_t lcos(\theta) + m_l\dot{x}_t\dot{l}cos(\theta) - m_l\dot{x}_t l\dot{\theta}sin(\theta) \\ & - \left(m_l\dot{x}_t\dot{l}cos(\theta) - m_l\dot{x}_t l\dot{\theta}sin(\theta) - m_l glsin(\theta)\right) \\ =& m_l\ddot{\theta}l^2 + 2m_l\dot{\theta}l\dot{l} + m_l\ddot{x}_t lcos(\theta) + m_l glsin(\theta) \\ =& \ddot{\theta}l + 2\dot{\theta}\dot{l} + \ddot{x}_t cos(\theta) + gsin(\theta) \end{aligned} \tag{B.51}$$

Isolating $\ddot{\theta}$

$$\ddot{\theta} = \frac{1}{l}\left(-2\dot{\theta}\dot{l} - \ddot{x}_t cos(\theta) - gsin(\theta)\right) \tag{B.52}$$

Isolating the accelerations from each other

The model of the crane is simulated using Simulink, a Matlab programming environment used for simulation and modelling of various systems. To simulate this model, Simulink requires a certain order to simulate the three equations of motion. This could be solved by adding delays and stating the preferred order. Nevertheless, it was decided that, to simplify the modelling and simulation, the equations will be isolated from each other, i.e., $\ddot{l}$ and $\ddot{\theta}$ will be modified as to not directly contain $\ddot{x}$.
To simplify the calculations, the following changes have been made:

$$J = \frac{1}{m_t + m_l + \frac{I_x}{r_x^2}},\ V = \frac{1}{m_l + \frac{4J_l}{r_l^2}},\ I_l = \frac{K_{el}I_l}{r_l},\ I_x = \frac{K_{ex}I_x}{r_x} \tag{B.53}$$

First, the right-hand side of the equations for $\ddot{l}$ and $\ddot{\theta}$ are introduced into the equation of motion for $\ddot{x}$, so that **Equation B.30** only contains $\ddot{x}$.

$$\begin{aligned} \ddot{x}_t =& J\Bigg(I_x - B_x\dot{x}_t - m_l sin(\theta)\Bigg(V\Bigg(I_l - B_l\dot{l} + m_l l\dot{\theta}^2 + m_l gcos(\theta) - m_l\ddot{x}_t sin(\theta)\Bigg)\Bigg) \\ & - 2m_l\dot{l}\dot{\theta}cos(\theta) - m_l lcos(\theta)\left(\frac{1}{l}\left(-2\dot{\theta}\dot{l} - \ddot{x}_t cos(\theta) - gsin(\theta)\right)\right) + m_l l\dot{\theta}^2 sin(\theta)\Bigg) \\ =& J\Bigg(I_x - B_x\dot{x}_t - m_l Vsin(\theta)\Bigg(I_l - B_l\dot{l} + m_l l\dot{\theta}^2 + m_l gcos(\theta) - m_l\ddot{x}_t sin(\theta)\Bigg) - 2m_l\dot{l}\dot{\theta}cos(\theta) \\ & - m_l lcos(\theta)\left(\frac{1}{l}\left(-2\dot{\theta}\dot{l} - \ddot{x}_t cos(\theta) - gsin(\theta)\right)\right) + m_l l\dot{\theta}^2 sin(\theta)\Bigg) \end{aligned} \tag{B.54}$$

$$\begin{aligned}\frac{\ddot{x}_t}{J} =& I_x - B_x\dot{x}_t - m_l V I_l sin(\theta) + m_l V B_l \dot{l} sin(\theta) - m_l^2 l V \dot{\theta}^2 sin(\theta) - m_l^2 V g sin(\theta) cos(\theta) \\ &+ m_l^2 V \ddot{x}_t sin^2(\theta) + m_l \ddot{x}_t cos^2(\theta) + m_l g sin(\theta) cos(\theta) + m_l l \dot{\theta}^2 sin(\theta)\end{aligned} \tag{B.55}$$

Isolate the terms that contain $\ddot{x}_t$ to the left-hand side of the equality.

$$\begin{aligned}&\ddot{x}_t \left(\frac{1}{J} - m_l cos^2(\theta) - m_l^2 V sin^2(\theta)\right) = I_x - B_x\dot{x}_t - m_l V I_l sin(\theta) + m_l V B_l \dot{l} sin(\theta) \\ &- m_l^2 l V \dot{\theta}^2 sin(\theta) - m_l^2 V g sin(\theta) cos(\theta) + m_l g sin(\theta) cos(\theta) + m_l l \dot{\theta}^2 sin(\theta) \\ &= I_x - B_x\dot{x}_t - m_l V I_l sin(\theta) + m_l V B_l \dot{l} sin(\theta) + m_l l \dot{\theta}^2 sin(\theta)(1 - m_l V) \\ &+ m_l g sin(\theta) cos(\theta)(1 - m_l V)\end{aligned} \tag{B.56}$$

Isolate $\ddot{x}_t$,

$$\begin{aligned}\ddot{x}_t =& \frac{1}{\left(\frac{1}{J} - m_l cos^2(\theta) - m_l^2 V sin^2(\theta)\right)} \Bigg(I_x - B_x\dot{x}_t - m_l V I_l sin(\theta) + m_l V B_l \dot{l} sin(\theta) \\ &+ m_l l \dot{\theta}^2 sin(\theta)(1 - m_l V) + m_l g sin(\theta) cos(\theta)(1 - m_l V) \Bigg)\end{aligned} \tag{B.57}$$

Thereafter, the newly obtained equation of motion for $\ddot{x}$ can be introduced into **Equations 3.22** and **3.24**, equations of motion for $\ddot{l}$ and $\ddot{\theta}$, respectively.
The simplifications made at the beginning of this subsection are switched back.

$$\begin{aligned}\ddot{x}_t =& \frac{1}{\left(m_t + m_l + \frac{J_x}{r_x^2} - m_l cos^2(\theta) - \frac{m_l^2 sin^2(\theta)}{m_l + \frac{4J_l}{r_l^2}}\right)} \Bigg[\frac{K_{ex} I_x}{r_x} - B_x\dot{x}_t - \frac{m_l K_{el} I_l sin(\theta)}{r_l m_l + \frac{4J_l}{r_l}} + \frac{m_l B_l \dot{l} sin(\theta)}{m_l + \frac{4J_l}{r_l^2}} \\ &+ m_l l \dot{\theta}^2 sin(\theta) \left(1 - \frac{m_l}{m_l + \frac{4J_l}{r_l^2}}\right) + m_l g sin(\theta) cos(\theta) \left(1 - \frac{m_l}{m_l + \frac{4J_l}{r_l^2}}\right) \Bigg]\end{aligned} \tag{B.58}$$

$$\ddot{l} = \frac{1}{m_l + \frac{4J_l}{r_l^2}} \left[\frac{K_{el} I_l}{r_l} - B_l \dot{l} + m_l l \dot{\theta}^2 + m_l g cos(\theta) - \frac{m_l sin(\theta)}{\left(m_t + m_l + \frac{J_x}{r_x^2} - m_l cos^2(\theta) - \frac{m_l^2 sin^2(\theta)}{m_l + \frac{4J_l}{r_l^2}} \right)} \right.$$
$$\left(\frac{K_{ex} I_x}{r_x} - B_x \dot{x}_t - \frac{m_l K_{el} I_l sin(\theta)}{r_l m_l + \frac{4J_l}{r_l}} + \frac{m_l B_l \dot{l} sin(\theta)}{m_l + \frac{4J_l}{r_l^2}} + m_l l \dot{\theta}^2 sin(\theta) \left(1 - \frac{m_l}{m_l + \frac{4J_l}{r_l^2}} \right) \right.$$
$$\left. \left. + m_l g sin(\theta) cos(\theta) \left(1 - \frac{m_l}{m_l + \frac{4J_l}{r_l^2}} \right) \right) \right] \tag{B.59}$$

$$\ddot{\theta} = \frac{1}{l} \left[-2\dot{\theta}\dot{l} - \frac{cos(\theta)}{\left(m_t + m_l + \frac{J_x}{r_x^2} - m_l cos^2(\theta) - \frac{m_l^2 sin^2(\theta)}{m_l + \frac{4J_l}{r_l^2}} \right)} \left(\frac{K_{ex} I_x}{r_x} - B_x \dot{x}_t - \frac{m_l K_{el} I_l sin(\theta)}{r_l m_l + \frac{4J_l}{r_l}} \right.\right.$$
$$\left. + \frac{m_l B_l \dot{l} sin(\theta)}{m_l + \frac{4J_l}{r_l^2}} + m_l l \dot{\theta}^2 sin(\theta) \left(1 - \frac{m_l}{m_l + \frac{4J_l}{r_l^2}} \right) + m_l g sin(\theta) cos(\theta) \left(1 - \frac{m_l}{m_l + \frac{4J_l}{r_l^2}} \right) \right)$$
$$\left. - g sin(\theta) \right] \tag{B.60}$$

With this, the process of deriving the mathematical model for the crane can be considered finished.

Appendix C

3D Model of Overhead Crane

This chapter contains the step-by-step derivation of the same overhead crane from **Chapter 2** but in 3D.

3-D model

This section will delve into deriving a crane model for a 3-D crane. Said crane has been designed following the previous 2-D crane and a figure depicting what that crane would look like can be seen in **Figure C.1**.

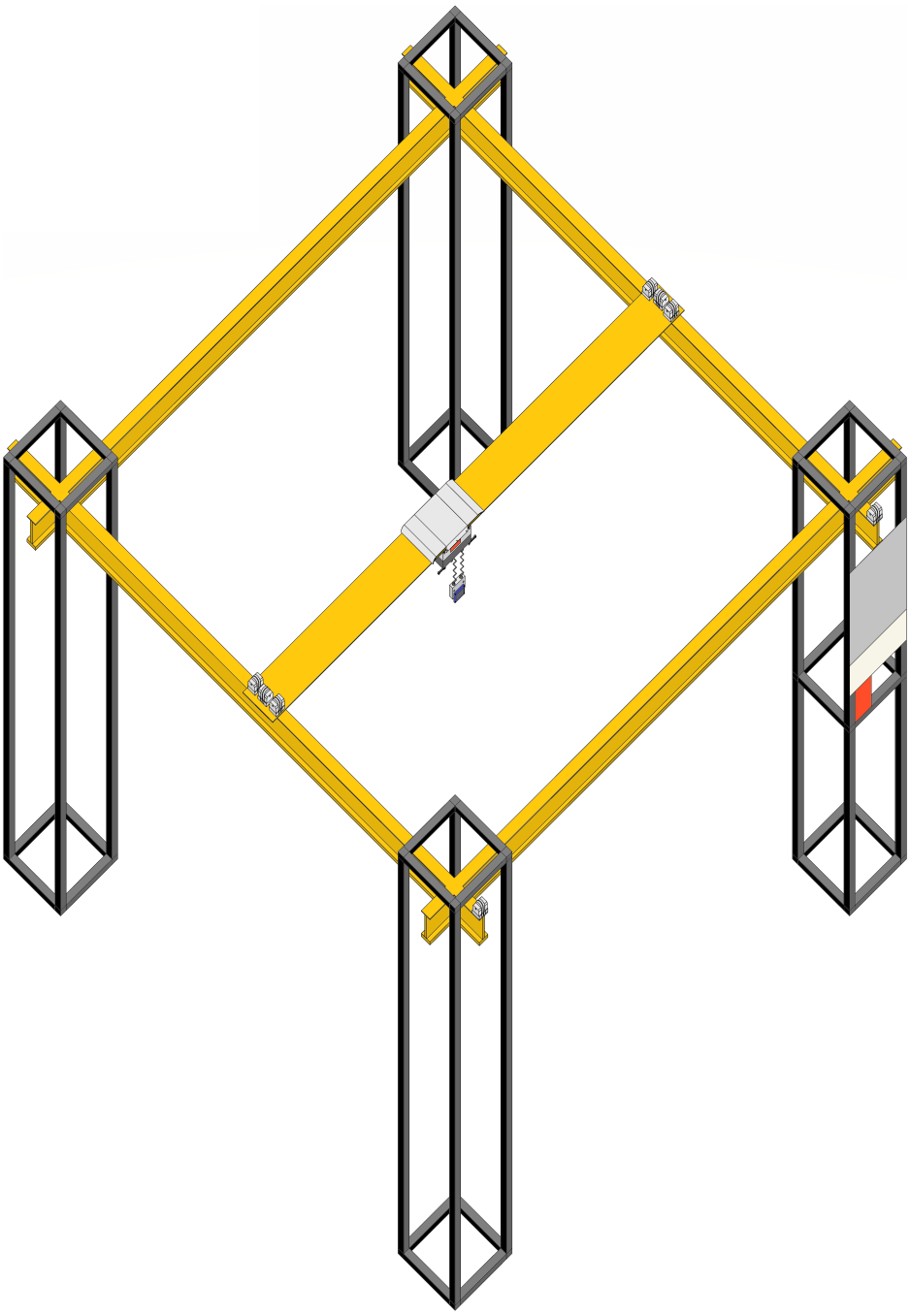


Figure C.1: Illustration of a 3-D gantry crane based on the crane from the previous section. The above crane contains depth, where the top beam (which contains the trolley and, subsequently, the head) moves in the y-direction. For this crane, it has been discussed that said beam and the trolley will act as one component, in terms of the depth motion. That is, the y movement of the trolley will be equivalent to the beams motion. On the contrary, the trolley will move on the x-axis in the same manner as it did for the 2-D crane.

This section differs from the previous in that a new dimension is added to the system. This will change the variables and the relationships between them, which will, ultimately, modify the equations derived previously in **Section 3.3**. **Figure C.2** shows said relationships and variables that will be utilised to model the 3-D crane.

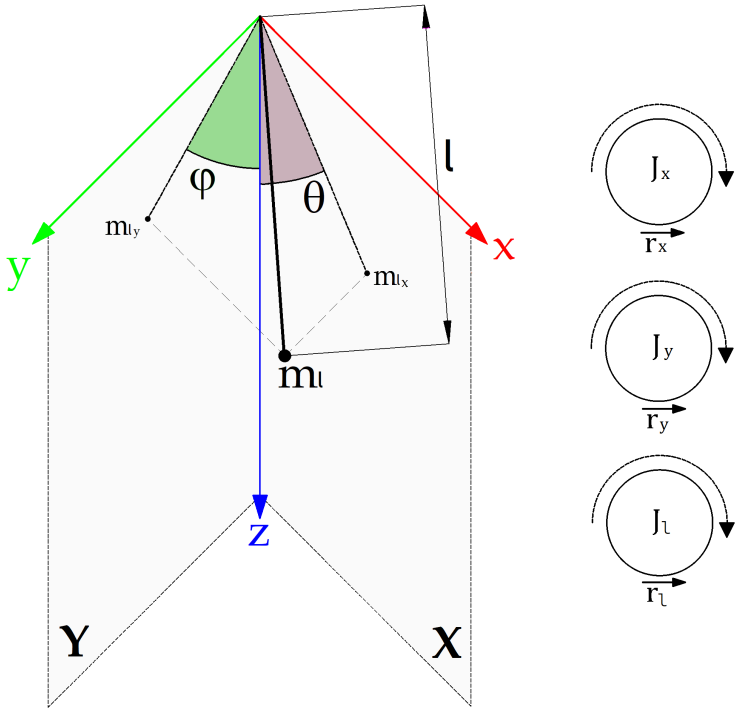


Figure C.2: Figure depicting the relationship between axes and angles. m_l refers to the center point of the load, which is depicted as a mass. m_{l_x} and m_{l_y} are the projections of m_l onto their respective planes, Y for m_{l_y} and X for m_{l_y}. These notations are not utilised in deriving the model but are for aiding in understanding the description of the position of the load. x depends on θ and y on ϕ. The combination of these two plus the length of the cable l gives the position of the load in 3D space.

where	x	:	Position on the horizontal axis	(m)
	y	:	Position on the depth axis	(m)
	z	:	Position on the vertical axis	(m)
	l	:	Length of the wire	(m)
	θ	:	Angle between the load and its equilibrium point	$(°)$
	ϕ	:	Second angle between the load and its equilibrium point	$(°)$
	J_x	:	Moment of Inertia for the x direction motor	$(\cdot)$
	J_y	:	Moment of Inertia for the y direction motor	$(\cdot)$
	J_l	:	Moment of Inertia for the wire-length motor	$(\cdot)$
	r_x	:	Radius of the X direction drum	(m)
	r_y	:	Radius of the Y direction drum	(m)
	r_l	:	Radius of the wire-length drum	(m)

The model will be derived using Lagrange equations, which start by describing the systems kinetic and potential energies. This requires the positional equation of the trolley shown below in **Equation C.1**. The trolley position only takes x and y into account, hence z_t being 0;

$$\vec{p}_t = x_t \cdot \vec{i} + y_t \cdot \vec{j} + z_t \cdot \vec{k} \mid_{z_t=0} \tag{C.1}$$

where

$\vec{p}_t$	:	Vector position of the trolley	(m)
x_t	:	Trolley X position	(m)
$\vec{i}$	:	Unit vector for X	$(\cdot)$
y_t	:	Trolley Y position	(m)
$\vec{j}$	:	Unit vector for Y	$(\cdot)$
z_t	:	Trolley Z position	(m)
$\vec{k}$	:	Unit vector for Z	$(\cdot)$

The position of the load is given in **Equation C.2**

$$\begin{aligned} \vec{p}_l &= x_l \cdot \vec{i} + y_l \cdot \vec{j} + z_l \cdot \vec{k} \\ &= (x_t + l \cdot sin(\theta)) + (y_t + l \cdot sin(\phi)) + (z_t + l \cdot cos(\theta)cos(\phi)) \mid_{z_t=0} \\ &= x_t + y_t + l\,(sin(\theta) + sin(\phi) + cos(\theta)cos(\phi)) \end{aligned} \tag{C.2}$$

where

$\vec{p}_l$	:	Vector position of the load	(m)
x_l	:	Load X position	(m)
y_l	:	Load Y position	(m)
z_l	:	Load Z position	(m)
l	:	Length of the wire	(m)
θ	:	Angle between trolley and load	$(°)$
ϕ	:	Angle between trolley and load	$(°)$

With the positional descriptions in place, the Lagrangian can be derived. The kinetic energy of the system comes from five sources, namely the trolley and load, as well as their respective servo motor drums. The trolley is controlled by the use of two servo motors, one per direction it can move to. Whereas the load's height is only affected by a single servo motor.
The kinetic energy of the trolley:

$$T_t = \frac{1}{2} \cdot m_t \left(\dot{x}_t^2 + \dot{y}_t^2\right) \tag{C.3}$$

where

T_t	:	Kinetic energy of the trolley	(J)
m_t	:	Mass of trolley	(kg)
$\dot{x}_t$	:	X velocity of trolley	(m/s)
$\dot{y}_t$	:	Y velocity of trolley	(m/s)

Kinetic energy of the load:

$$T_l = \frac{1}{2} \cdot m_l \cdot \left(\dot{x}_l^2 + \dot{y}_l^2 + \dot{z}_l^2\right) \tag{C.4}$$

where
T_l : Kinetic energy of the load (J)
m_l : Mass of load (kg)
$\dot{x}_l$: X velocity of load (m/s)
$\dot{y}_l$: Y velocity of load (m/s)
$\dot{z}_l$: Z velocity of load (m/s)

The kinetic energy of the rotating wire drums are calculated as a rotational kinetic energy instead, which substitutes the mass term for the moment of inertia of the body and the velocity for angular velocity of the drum. Angular velocity is not measured in this project, and is instead derived by relating the linear velocity of the wire and the drums radius.
The kinetic energies of the drums for the trolley and for the drum of hoisting are shown in **Equation C.5**

$$\begin{aligned} T_{px} &= \frac{1}{2} \cdot J_x \cdot \left(\frac{\dot{x}_t}{r_x}\right)^2 \\ T_{py} &= \frac{1}{2} \cdot J_y \cdot \left(\frac{\dot{y}_t}{r_y}\right)^2 \\ T_{pl} &= \frac{1}{2} \cdot J_l \cdot \left(\frac{2\dot{l}}{r_l}\right)^2 \end{aligned} \tag{C.5}$$

where
T_{px} : Kinetic energy of the drum for x (J)
T_{py} : Kinetic energy of the drum for y (J)
T_{pl} : Kinetic energy of the drum for l (J)
J_x : Moment of Inertia of the drum for x $(kg \cdot m^2)$
J_y : Moment of Inertia of the drum for y $(kg \cdot m^2)$
J_l : Moment of Inertia of the drum for l $(kg \cdot m^2)$
r_x : Radius of the drum for x (m)
r_y : Radius of the drum for y (m)
r_l : Radius of the drum for l (m)

The "2" is inserted in T_{pl} because the gearing in the drum for the hoisting wire is slightly different from the other two, resulting in a different approximation.
The kinetic energy of the system is the sum of its parts, $T_t + T_l$, and is thus given in **Equation C.6**;

$$T = \frac{1}{2} \cdot m_t \cdot \left(\dot{x}_t^2 + \dot{y}_t^2\right) + \frac{1}{2} \cdot m_l \cdot \left(\dot{x}_l^2 + \dot{y}_l^2 + \dot{z}_l^2\right) + \frac{1}{2} \cdot J_x \cdot \left(\frac{\dot{x}_t}{r_x}\right)^2 + \frac{1}{2} \cdot J_y \cdot \left(\frac{\dot{y}_t}{r_y}\right)^2 + \frac{1}{2} \cdot J_l \cdot \left(\frac{2\dot{l}}{r_l}\right)^2 \tag{C.6}$$

The second term of the Lagrangian equation describes the kinetic energies of the system in relation to the velocity of the load, which is not a measured parameter in this project. Instead, the load's velocities are related to the velocity of the trolley using **Equation C.2**, also shown below in **Equation C.7**:

$$\vec{p}_l = x_t + y_t + l \cdot (sin(\theta) + sin(\phi) + cos(\theta) \cdot cos(\phi)) \tag{C.7}$$

Isolating the term representing x_l from the above positional equation;

$$x_l = x_t + l \cdot sin(\theta) \tag{C.8}$$

Firstly, the time derivative of x_l is computed to obtained the X velocity of the load.

$$\frac{dx_l}{dt} = \frac{d\left(x_t + l \cdot sin(\theta)\right)}{dx_t} \cdot \frac{dx_t}{dt} + \frac{d\left(x_t + l \cdot sin(\theta)\right)}{dl} \cdot \frac{dl}{dt} + \frac{d\left(x_t + l \cdot sin(\theta)\right)}{d\theta} \cdot \frac{d\theta}{dt} \tag{C.9}$$

$$\dot{x_l} = \dot{x_t} + \dot{l} \cdot sin(\theta) + l \cdot \dot{\theta} \cdot cos(\theta) \tag{C.10}$$

Squaring both sides;

$$\begin{aligned} \dot{x_l}^2 =& \dot{x_t}^2 + \dot{l}^2 \cdot sin^2(\theta) + l^2 \cdot \dot{\theta}^2 \cdot cos^2(\theta) + 2 \cdot \dot{l} \cdot \dot{x_t} \cdot sin(\theta) + 2 \cdot l \cdot \dot{x_t} \cdot \dot{\theta} \cdot cos(\theta) \\ &+ 2 \cdot l \cdot \dot{l} \cdot \dot{\theta} \cdot sin(\theta) \cdot cos(\theta) \end{aligned} \tag{C.11}$$

The same is done for the term representing y_l

$$y_l = y_t + l \cdot sin(\phi) \tag{C.12}$$

Firstly, the time derivative of y_l is computed to obtained the X velocity of the load.

$$\frac{dy_l}{dt} = \frac{d\left(y_t + l \cdot sin(\phi)\right)}{dy_t} \cdot \frac{dy_t}{dt} + \frac{d\left(y_t + l \cdot sin(\phi)\right)}{dl} \cdot \frac{dl}{dt} + \frac{d\left(y_t + l \cdot sin(\phi)\right)}{d\phi} \cdot \frac{d\phi}{dt} \tag{C.13}$$

$$\dot{y_l} = \dot{y_t} + \dot{l} \cdot sin(\phi) + l \cdot \dot{\phi} \cdot cos(\phi) \tag{C.14}$$

Squaring the previous equation results in the following equation;

$$\begin{aligned} \dot{y_l}^2 =& \dot{y_t}^2 + \dot{l}^2 \cdot sin^2(\phi) + l^2 \cdot \dot{\phi}^2 \cdot cos^2(\phi) + 2 \cdot \dot{l} \cdot \dot{y_t} \cdot sin(\phi) + 2 \cdot l \cdot \dot{y_t} \cdot \dot{\phi} \cdot cos(\phi) \\ &+ 2 \cdot l \cdot \dot{l} \cdot \dot{\phi} \cdot sin(\phi) \cdot cos(\phi) \end{aligned} \tag{C.15}$$

The same is done for the term representing z_l

$$z_l = l \cdot cos(\theta) \cdot cos(\phi) \tag{C.16}$$

$$\frac{dz_l}{dt} = \frac{d\left(l \cdot cos(\theta) \cdot cos(\phi)\right)}{dl} \cdot \frac{dl}{dt} + \frac{d\left(l \cdot cos(\theta) \cdot cos(\phi)\right)}{d\theta} \cdot \frac{d\theta}{dt} + \frac{d\left(l \cdot cos(\theta) \cdot cos(\phi)\right)}{d\phi} \cdot \frac{d\phi}{dt} \tag{C.17}$$

$$\dot{z_l} = \dot{l} \cdot cos(\theta) \cdot cos(\phi) - l \cdot \dot{\theta} \cdot sin(\theta) \cdot cos(\phi) - l \cdot \dot{\phi} \cdot cos(\theta) \cdot sin(\phi) \tag{C.18}$$

And lastly, the final component needed to complete the equation for the kinetic energies is obtained by squaring $\dot{z}_l$.

$$\begin{aligned}\dot{z}_l^2 =& \dot{l}^2 \cdot cos^2(\theta) \cdot cos^2(\phi) + l^2 \cdot \dot{\theta}^2 \cdot sin^2(\theta) \cdot cos^2(\phi) + l^2 \cdot \dot{\phi}^2 \cdot cos^2(\theta) \cdot sin^2(\phi) \\ & - 2 \cdot l \cdot \dot{l} \cdot \dot{\theta} \cdot sin(\theta) \cdot cos(\theta) \cdot cos^2(\phi) - 2 \cdot l \cdot \dot{l} \cdot \dot{\phi} \cdot cos^2(\theta) \cdot sin(\phi) \cdot cos(\phi) \\ & + 2 \cdot l^2 \cdot \dot{\theta} \cdot \dot{\phi} \cdot sin(\theta) \cdot cos(\theta) \cdot sin(\phi) \cdot cos(\phi)\end{aligned} \tag{C.19}$$

The complete equation for the kinetic energies has now been acquired, see **Equation C.20**.

$$\begin{aligned}T =& \frac{1}{2} \cdot m_t \cdot \left(\dot{x}_t^2 + \dot{y}_t^2\right) + \frac{1}{2} \cdot m_l \cdot \Bigg(\dot{x_t}^2 + \dot{y}_t^2 + 2 \cdot \dot{l} \cdot \dot{x}_t \cdot sin(\theta) + 2 \cdot l \cdot \dot{x}_t \cdot \dot{\theta} \cdot cos(\theta) + \dot{l}^2 \cdot sin^2(\theta) \\ & + \dot{l}^2 \cdot sin^2(\phi) + 2 \cdot \dot{l} \cdot \dot{y}_t \cdot sin(\phi) + 2 \cdot l \cdot \dot{y}_t \cdot \dot{\phi} \cdot cos(\phi) + \dot{l}^2 \cdot cos^2(\theta) \cdot cos^2(\phi) \\ & + l^2 \cdot \dot{\theta}^2 \cdot cos^2(\theta) + l^2 \cdot \dot{\theta}^2 \cdot sin^2(\theta) \cdot cos^2(\phi) + l^2 \cdot \dot{\phi}^2 \cdot cos^2(\phi) + l^2 \cdot \dot{\phi}^2 \cdot cos^2(\theta) \cdot sin^2(\phi) \\ & + 2 \cdot l \cdot \dot{l} \cdot \dot{\theta} \cdot sin(\theta) \cdot cos(\theta) \cdot sin^2(\phi) + 2 \cdot l \cdot \dot{l} \cdot \dot{\phi} \cdot sin^2(\theta) \cdot sin(\phi) \cdot cos(\phi) \\ & + 2 \cdot l^2 \cdot \dot{\theta} \cdot \dot{\phi} \cdot sin(\theta) \cdot cos(\theta) \cdot sin(\phi) \cdot cos(\phi) \Bigg) + \frac{1}{2} J_x \left(\frac{\dot{x}_t}{r_x}\right)^2 + \frac{1}{2} J_y \left(\frac{\dot{y}_t}{r_y}\right)^2 + \frac{1}{2} J_l \left(\frac{2\dot{l}}{r_l}\right)^2\end{aligned} \tag{C.20}$$

The next step is to derive the potential energy, for this, the following presumption is stated: If the origin for the Z-axis is set at the trolleys level, this will effectively not have any potential energy, leaving only the load, which is also defined as having zero potential energy when it is at $z = 0$. It is thus given as:

$$V = -m_l \cdot g \cdot z_l = -m_l \cdot g \cdot (z_t + l \cdot cos(\theta)) \mid_{z_t=0} = -m_l \cdot g \cdot l \cdot cos(\theta) cos(\phi) \tag{C.21}$$

where V : Potential energy of system (J)
g : Gravity constant = 9.82 $(ms^{-}2)$

The Lagrangian L for the system can now be presented:

$$\begin{aligned}L =& T - V = \frac{1}{2} \cdot m_t \cdot \left(\dot{x}_t^2 + \dot{y}_t^2\right) + \frac{1}{2} \cdot m_l \cdot \Bigg(\dot{x_t}^2 + \dot{y}_t^2 + 2 \cdot \dot{l} \cdot \dot{x}_t \cdot sin(\theta) + 2 \cdot l \cdot \dot{x}_t \cdot \dot{\theta} \cdot cos(\theta) \\ & + \dot{l}^2 \cdot sin^2(\theta) + \dot{l}^2 \cdot sin^2(\phi) + 2 \cdot \dot{l} \cdot \dot{y}_t \cdot sin(\phi) + 2 \cdot l \cdot \dot{y}_t \cdot \dot{\phi} \cdot cos(\phi) + \dot{l}^2 \cdot cos^2(\theta) \cdot cos^2(\phi) \\ & + l^2 \cdot \dot{\theta}^2 \cdot cos^2(\theta) + l^2 \cdot \dot{\theta}^2 \cdot sin^2(\theta) \cdot cos^2(\phi) + l^2 \cdot \dot{\phi}^2 \cdot cos^2(\phi) + l^2 \cdot \dot{\phi}^2 \cdot cos^2(\theta) \cdot sin^2(\phi) \\ & + 2 \cdot l \cdot \dot{l} \cdot \dot{\theta} \cdot sin(\theta) \cdot cos(\theta) \cdot sin^2(\phi) + 2 \cdot l \cdot \dot{l} \cdot \dot{\phi} \cdot sin^2(\theta) \cdot sin(\phi) \cdot cos(\phi) \\ & + 2 \cdot l^2 \cdot \dot{\theta} \cdot \dot{\phi} \cdot sin(\theta) \cdot cos(\theta) \cdot sin(\phi) \cdot cos(\phi) \Bigg) + \frac{1}{2} J_x \left(\frac{\dot{x}_t}{r_x}\right)^2 + \frac{1}{2} J_y \left(\frac{\dot{y}_t}{r_y}\right)^2 + \frac{1}{2} J_l \left(\frac{2\dot{l}}{r_l}\right)^2 \\ & + m_l \cdot g \cdot l \cdot cos(\theta) cos(\phi)\end{aligned} \tag{C.22}$$

And with this, the equations of motion can be obtained. Five variables are present, so an equation for each is needed. These are $\ddot{x}$, $\ddot{y}$, $\ddot{l}$, $\ddot{\phi}$ and $\ddot{\theta}$. The Lagrangian equation when no external force is present is defined as;

$$0 = \frac{d}{dt}\frac{\partial}{\partial \dot{p}}L - \frac{\partial}{\partial p}L \tag{C.23}$$

Whereas when there is an external force present, it is defined as;

$$F_z = \frac{d}{dt}\frac{\partial}{\partial \dot{p}}L - \frac{\partial}{\partial p}L \tag{C.24}$$

Equation of motion for $\ddot{x}$

The servo motors pulling the trolleys wire are considered an external force, meaning **Equation C.24** is used. The forces can be represented as shown below in **Equation C.25**, together with a friction term

$$F_x = \frac{K_{ex} \cdot I_x}{r_x} - B_x \cdot \dot{x}_t \tag{C.25}$$

where

F_x	:	Force acting along X	(N)
K_{ex}	:	Motor constants for trolley motor	$(\cdot)$
I_x	:	Current delivered to trolley motor	(A)
r_x	:	Radius of trolley drum	(m)
B_x	:	Friction coefficient for moving along X	(N)
$\dot{x}$	:	Velocity of the hoisting system	(m/s)

The Lagrangian equation for x can now be calculated.

$$\frac{K_{ex} \cdot I_x}{r_x} - B_x \cdot \dot{x}_t = \frac{d}{dt}\frac{\partial}{\partial \dot{x}_t}L - \frac{\partial}{\partial x_t}L \tag{C.26}$$

$$\begin{aligned}
\frac{d}{dt}\frac{\partial}{\partial \dot{x}_t}L = \frac{d}{dt}\frac{\partial}{\partial \dot{x}_t}\Bigg(&\frac{1}{2}m_t\left(\dot{x}_t^2 + \dot{y}_t^2\right) + \frac{1}{2}m_l\Big(\dot{x}_t{}^2 + \dot{y}_t^2 + 2\cdot\dot{l}\cdot\dot{x}_t\cdot sin(\theta) + 2\cdot l\cdot\dot{x}_t\cdot\dot{\theta}\cdot cos(\theta) \\
&+ 2\cdot\dot{l}\cdot\dot{y}_t\cdot sin(\phi) + 2\cdot l\cdot\dot{y}_t\cdot\dot{\phi}\cdot cos(\phi) + \dot{l}^2\cdot sin^2(\theta) + \dot{l}^2\cdot sin^2(\phi) + \dot{l}^2\cdot cos^2(\theta)cos^2(\phi) \\
&+ l^2\cdot\dot{\theta}^2\cdot cos^2(\theta) + l^2\cdot\dot{\theta}^2\cdot sin^2(\theta)cos^2(\phi) + l^2\cdot\dot{\phi}^2\cdot cos^2(\phi) + l^2\cdot\dot{\phi}^2\cdot cos^2(\theta)sin^2(\phi) \\
&+ 2\cdot l\cdot\dot{l}\cdot\dot{\theta}\cdot sin(\theta)cos(\theta)sin^2(\phi) + 2\cdot l\cdot\dot{l}\cdot\dot{\phi}\cdot sin^2(\theta)sin(\phi)cos(\phi) \\
&+ 2\cdot l^2\cdot\dot{\theta}\cdot\dot{\phi}\cdot sin(\theta)cos(\theta)sin(\phi)cos(\phi)\Big) + \frac{1}{2}J_x\left(\frac{\dot{x}_t}{r_x}\right)^2 + \frac{1}{2}J_y\left(\frac{\dot{y}_t}{r_y}\right)^2 + \frac{1}{2}J_l\left(\frac{2\dot{l}}{r_l}\right)^2 \\
&+ m_l\cdot g\cdot l\cdot cos(\theta)cos(\phi)\Bigg)
\end{aligned} \tag{C.27}$$

Terms without $\dot{x}_t$ get eliminated;

$$\frac{d}{dt}\frac{\partial}{\partial \dot{x}_t}L = \frac{d}{dt}\frac{\partial}{\partial \dot{x}_t}\left(\frac{1}{2}m_t\dot{x}_t^2 + \frac{1}{2}m_l\left(\dot{x}_t{}^2 + 2\dot{l}\dot{x}_t sin(\theta) + 2l\dot{x}_t\dot{\theta}cos(\theta)\right) + \frac{1}{2}J_x\left(\frac{\dot{x}_t}{r_x}\right)^2\right) \tag{C.28}$$

First, the partial differentiation with respect to $\dot{x}_t$ is derived;

$$\frac{\partial}{\partial \dot{x}_t}L = m_t\dot{x}_t + m_l\Big(\dot{x}_t + \dot{l}sin(\theta) + l\dot{\theta}cos(\theta)\Big) + \frac{J_x\dot{x}_t}{r_x^2} \tag{C.29}$$

Then, the time derivative;

$$\frac{d}{dt}\frac{\partial}{\partial \dot{x}_t}L = m_t\ddot{x}_t + m_l\ddot{x}_t + m_l\ddot{l}sin(\theta) + 2m_l\dot{l}\dot{\theta}cos(\theta) + m_l l\ddot{\theta}cos(\theta) - m_l l\dot{\theta}^2 sin(\theta) + \frac{J_x\ddot{x}_t}{r_x^2} \tag{C.30}$$

Only a single term is left in order to complete the Lagrangian, but since no term in the equation contains an x_t without a derivative, it vanishes. Thus, a complete Lagrange's equation for $\ddot{x}$ is presented in **Equation C.32**:

$$\frac{K_{ex}I_x}{r_x} - B_x\dot{x}_t = m_t\ddot{x}_t + m_l\ddot{x}_t + m_l\ddot{l}sin(\theta) + 2m_l\dot{l}\dot{\theta}cos(\theta) + m_l l\ddot{\theta}cos(\theta) - m_l l\dot{\theta}^2 sin(\theta) + \frac{J_x\ddot{x}_t}{r_x^2} \tag{C.31}$$

$$\ddot{x}_t = \frac{1}{\frac{J_x}{r_x^2} + m_t + m_l}\left(\frac{K_{ex}I_x}{r_x} - B_x\dot{x}_t - m_l\ddot{l}sin(\theta) - 2m_l\dot{l}\dot{\theta}cos(\theta) - m_l l\ddot{\theta}cos(\theta) + m_l l\dot{\theta}^2 sin(\theta)\right) \tag{C.32}$$

Equation of motion for $\ddot{y}$

The servo motors pulling the trolleys wire are considered an external force, meaning **Equation C.24** is used. The forces can represented as shown below in **Equation C.33** together with a friction term

$$F_y = \frac{K_{ey}\cdot I_y}{r_y} - B_y\cdot\dot{y}_t \tag{C.33}$$

where

F_y	:	Force acting along y	(N)
K_{ey}	:	Motor constants for trolley motor	$(\cdot)$
I_y	:	Current delivered to trolley motor	(A)
r_y	:	Radius of trolley drum	(m)
B_y	:	Friction coefficient for moving along Y	(N)
$\dot{y}$	:	Velocity of the hoisting system	(m/s)

The Lagrangian equation for y can now be calculated.

$$\frac{K_{ey}\cdot I_y}{r_y} - B_y\cdot\dot{y}_t = \frac{d}{dt}\frac{\partial}{\partial \dot{y}_t}L - \frac{\partial}{\partial y_t}L \tag{C.34}$$

$$\frac{d}{dt}\frac{\partial}{\partial \dot{y}_t}L = \frac{d}{dt}\frac{\partial}{\partial \dot{y}_t}\Bigg(\frac{1}{2}m_t\left(\dot{x}_t^2+\dot{y}_t^2\right)+\frac{1}{2}m_l\Bigg(\dot{x}_t{}^2+\dot{y}_t^2+2\cdot\dot{l}\cdot\dot{x}_t\cdot sin(\theta)+2\cdot l\cdot\dot{x}_t\cdot\dot{\theta}\cdot cos(\theta)$$
$$+2\cdot\dot{l}\cdot\dot{y}_t\cdot sin(\phi)+2\cdot l\cdot\dot{y}_t\cdot\dot{\phi}\cdot cos(\phi)+\dot{l}^2\cdot sin^2(\theta)+\dot{l}^2\cdot sin^2(\phi)+\dot{l}^2\cdot cos^2(\theta)cos^2(\phi)$$
$$+l^2\cdot\dot{\theta}^2\cdot cos^2(\theta)+l^2\cdot\dot{\theta}^2\cdot sin^2(\theta)cos^2(\phi)+l^2\cdot\dot{\phi}^2\cdot cos^2(\phi)+l^2\cdot\dot{\phi}^2\cdot cos^2(\theta)sin^2(\phi)$$
$$+2\cdot l\cdot\dot{l}\cdot\dot{\theta}\cdot sin(\theta)cos(\theta)sin^2(\phi)+2\cdot l\cdot\dot{l}\cdot\dot{\phi}\cdot sin^2(\theta)sin(\phi)cos(\phi)$$
$$+2\cdot l^2\cdot\dot{\theta}\cdot\dot{\phi}\cdot sin(\theta)cos(\theta)sin(\phi)cos(\phi)\Bigg)+\frac{1}{2}J_x\left(\frac{\dot{x}_t}{r_x}\right)^2+\frac{1}{2}J_y\left(\frac{\dot{y}_t}{r_y}\right)^2+\frac{1}{2}J_l\left(\frac{2\dot{l}}{r_l}\right)^2$$
$$+m_l\cdot g\cdot l\cdot cos(\theta)cos(\phi)\Bigg) \tag{C.35}$$

Terms without $\dot{y}_t$ get eliminated;

$$\frac{d}{dt}\frac{\partial}{\partial \dot{y}_t}L = \frac{d}{dt}\frac{\partial}{\partial \dot{y}_t}\left(\frac{1}{2}m_t\dot{y}_t^2+\frac{1}{2}m_l\left(\dot{y}_t^2+2\dot{l}\dot{y}_t sin(\phi)+2l\dot{y}_t\dot{\phi}cos(\phi)\right)+\frac{1}{2}J_y\left(\frac{\dot{y}_t}{r_y}\right)^2\right) \tag{C.36}$$

First, the partial differentiation with respect to $\dot{y}_t$;

$$\frac{\partial}{\partial \dot{y}_t}L = m_t\dot{y}_t+m_l\dot{y}_t+m_l\dot{l}sin(\phi)+m_l l\dot{\phi}cos(\phi)+\frac{J_y\dot{y}_t}{r_y^2} \tag{C.37}$$

Then, the time derivative is computed;

$$\frac{d}{dt}\frac{\partial}{\partial \dot{y}_t}L = m_t\ddot{y}_t+m_l\ddot{y}_t+m_l\ddot{l}sin(\phi)+m_l\dot{l}\dot{\phi}cos(\phi)+m_l\dot{l}\dot{\phi}cos(\phi)+m_l l\ddot{\phi}cos(\phi)-m_l l\dot{\phi}^2sin(\phi)$$
$$+\frac{J_y\ddot{y}_t}{r_y^2} \tag{C.38}$$

Only a single term is left in order to complete the Lagrangian, but since no term in the equation contains an y_t without a derivative, it vanishes. Thus, a complete Lagrange's equation for $\ddot{y}$ can be seen **Equation C.40**.

$$\frac{K_{ey}I_y}{r_y}-B_y\dot{y}_t = m_t\ddot{y}_t+m_l\ddot{y}_t+m_l\ddot{l}sin(\phi)+2m_l\dot{l}\dot{\phi}cos(\phi)+m_l l\ddot{\phi}cos(\phi)-m_l l\dot{\phi}^2sin(\phi)+\frac{J_y\ddot{y}_t}{r_y^2} \tag{C.39}$$

$$\ddot{y}_t = \frac{1}{\frac{J_y}{r_y^2}+m_t+m_l}\left(\frac{K_{ey}I_y}{r_y}-B_y\dot{y}_t-m_l\ddot{l}sin(\phi)-2m_l\dot{l}\dot{\phi}cos(\phi)-m_l l\ddot{\phi}cos(\phi)+m_l l\dot{\phi}^2sin(\phi)\right) \tag{C.40}$$

Equation of motion for $\ddot{l}$

For calculating the forces acting along the hoisting wire, the Lagrange's equation will be solved with respect to l. There is a motor applying force on this axis as well, thus the same form of Lagrange is used. The force term is presented in **Equation C.41**;

$$F_l = \frac{2K_{el} \cdot I_l}{r_l} - B_l \cdot \dot{l} \tag{C.41}$$

where
F_l : Force applied by hoisting motor (N)
K_{el} : Motor constants for hoisting motor $(\cdot)$
I_l : Current delivered to the hoisting motor (A)
r_l : Radius of hoisting motor drum (m)
B_l : Friction coefficient for moving along Z $(\cdot)$
$\dot{l}$: Velocity of the hoisting system (m/s)

Thus the Lagrange equation to solve is shown in **Equation C.42**

$$\frac{K_{el} \cdot I_l}{r_l} - B_l \cdot \dot{l} = \frac{d}{dt}\frac{\partial}{\partial \dot{l}}L - \frac{\partial}{\partial l}L \tag{C.42}$$

Each term on the right will be calculated individually.

$$\begin{aligned}\frac{d}{dt}\frac{\partial}{\partial \dot{l}}L = \frac{d}{dt}\frac{\partial}{\partial \dot{l}}\Bigg(&\frac{1}{2}m_t\left(\dot{x}_t^2 + \dot{y}_t^2\right) + \frac{1}{2}m_l\Big(\dot{x}_t{}^2 + \dot{y}_t^2 + 2\cdot\dot{l}\cdot\dot{x}_t\cdot sin(\theta) + 2\cdot l\cdot\dot{x}_t\cdot\dot{\theta}\cdot cos(\theta)\\
&+ 2\cdot\dot{l}\cdot\dot{y}_t\cdot sin(\phi) + 2\cdot l\cdot\dot{y}_t\cdot\dot{\phi}\cdot cos(\phi) + \dot{l}^2\cdot sin^2(\theta) + \dot{l}^2\cdot sin^2(\phi) + \dot{l}^2\cdot cos^2(\theta)cos^2(\phi)\\
&+ l^2\cdot\dot{\theta}^2\cdot cos^2(\theta) + l^2\cdot\dot{\theta}^2\cdot sin^2(\theta)cos^2(\phi) + l^2\cdot\dot{\phi}^2\cdot cos^2(\phi) + l^2\cdot\dot{\phi}^2\cdot cos^2(\theta)sin^2(\phi)\\
&+ 2\cdot l\cdot\dot{l}\cdot\dot{\theta}\cdot sin(\theta)cos(\theta)sin^2(\phi) + 2\cdot l\cdot\dot{l}\cdot\dot{\phi}\cdot sin^2(\theta)sin(\phi)cos(\phi)\\
&+ 2\cdot l^2\cdot\dot{\theta}\cdot\dot{\phi}\cdot sin(\theta)cos(\theta)sin(\phi)cos(\phi)\Big) + \frac{1}{2}J_x\left(\frac{\dot{x}_t}{r_x}\right)^2 + \frac{1}{2}J_y\left(\frac{\dot{y}_t}{r_y}\right)^2 + \frac{1}{2}J_l\left(\frac{2\dot{l}}{r_l}\right)^2\\
&+ m_l\cdot g\cdot l\cdot cos(\theta)cos(\phi)\Bigg)\end{aligned} \tag{C.43}$$

Eliminating terms without $\dot{l}$;

$$\begin{aligned}\frac{d}{dt}\frac{\partial}{\partial \dot{l}}L = \frac{d}{dt}\frac{\partial}{\partial \dot{l}}\Bigg(&\frac{1}{2}m_l\Big(2\dot{l}\dot{x}_t sin(\theta) + 2\dot{l}\dot{y}_t sin(\phi) + \dot{l}^2 sin^2(\theta) + \dot{l}^2 sin^2(\phi) + \dot{l}^2 cos^2(\theta)cos^2(\phi)\\
&+ 2l\dot{l}\dot{\theta}sin(\theta)cos(\theta)sin^2(\phi) + 2l\dot{l}\dot{\phi}sin^2(\theta)sin(\phi)cos(\phi)\Big) + \frac{1}{2}J_l\left(\frac{2\dot{l}}{r_l}\right)^2\Bigg)\end{aligned} \tag{C.44}$$

Partial differentiation;

$$\begin{aligned}\frac{\partial}{\partial \dot{l}}L =& m_l\dot{x}_t sin(\theta) + m_l\dot{y}_t sin(\phi) + m_l\dot{l}sin^2(\theta) + m_l\dot{l}sin^2(\phi) + m_l\dot{l}cos^2(\theta)cos^2(\phi) \\ &+ m_l l\dot{\theta}sin(\theta)cos(\theta)sin^2(\phi) + m_l l\dot{\phi}sin^2(\theta)sin(\phi)cos(\phi) + \frac{4J_l\dot{l}}{r_l^2}\end{aligned} \tag{C.45}$$

Differentiating in terms of time:

$$\begin{aligned}\frac{d}{dt}\frac{\partial}{\partial \dot{l}}L =& m_l\ddot{x}_t sin(\theta) + m_l\dot{x}_t\dot{\theta}cos(\theta) + m_l\ddot{y}_t sin(\phi) + m_l\dot{y}_t\dot{\phi}cos(\phi) + m_l\ddot{l}sin^2(\theta) + m_l\ddot{l}sin^2(\phi) \\ &+ m_l\ddot{l}cos^2(\theta)cos^2(\phi) + m_l\dot{l}\dot{\phi}sin^2(\theta)sin(2\phi) + m_l\dot{l}\dot{\phi}sin^2(\theta)sin(\phi)cos(\phi) \\ &+ m_l\dot{l}\dot{\theta}sin(\theta)cos(\theta)sin^2(\phi) + m_l\dot{l}\dot{\theta}sin(2\theta)sin^2(\phi) + m_l l\ddot{\theta}sin(\theta)cos(\theta)sin^2(\phi) \\ &+ m_l l\ddot{\phi}sin^2(\theta)sin(\phi)cos(\phi) + m_l l\dot{\theta}^2cos(2\theta)sin^2(\phi) + m_l l\dot{\phi}^2sin^2(\theta)cos(2\phi) \\ &+ m_l l\dot{\theta}\dot{\phi}sin(\theta)cos(\theta)sin(2\phi) + m_l l\dot{\theta}\dot{\phi}sin(2\theta)sin(\phi)cos(\phi) + \frac{4J_l\ddot{l}}{r_l^2}\end{aligned} \tag{C.46}$$

Calculating $\frac{\partial}{\partial l}L$ from **Equation C.42** separately

$$\begin{aligned}\frac{\partial}{\partial l}L =& \frac{\partial}{\partial l}\Bigg(\frac{1}{2}m_l\Big(2l\dot{x}_t\dot{\theta}cos(\theta) + 2l\dot{y}_t\dot{\phi}cos(\phi) + l^2\dot{\theta}^2cos^2(\theta) + l^2\dot{\theta}^2sin^2(\theta)cos^2(\phi) + l^2\dot{\phi}^2cos^2(\phi) \\ &+ l^2\dot{\phi}^2cos^2(\theta)sin^2(\phi) + 2l\dot{l}\dot{\theta}sin(\theta)cos(\theta)sin^2(\phi) + 2l\dot{l}\dot{\phi}sin^2(\theta)sin(\phi)cos(\phi) \\ &+ 2l^2\dot{\theta}\dot{\phi}sin(\theta)cos(\theta)sin(\phi)cos(\phi)\Big) + m_l glcos(\theta)cos(\phi)\Bigg) \\ =& m_l\dot{x}_t\dot{\theta}cos(\theta) + m_l\dot{y}_t\dot{\phi}cos(\phi) + m_l l\dot{\theta}^2cos^2(\theta) + m_l l\dot{\theta}^2sin^2(\theta)cos^2(\phi) + m_l l\dot{\phi}^2cos^2(\phi) \\ &+ m_l l\dot{\phi}^2cos^2(\theta)sin^2(\phi) + m_l\dot{l}\dot{\theta}sin(\theta)cos(\theta)sin^2(\phi) + m_l\dot{l}\dot{\phi}sin^2(\theta)sin(\phi)cos(\phi) \\ &+ 2m_l l\dot{\theta}\dot{\phi}sin(\theta)cos(\theta)sin(\phi)cos(\phi)\Big) + m_l gcos(\theta)cos(\phi)\end{aligned} \tag{C.47}$$

Putting every term back in **Equation C.42**

$$\begin{aligned}\frac{K_{el}I_l}{r_l} - B_l\dot{l} = \Bigg(& m_l\ddot{x}_t sin(\theta) + m_l\dot{x}_t\dot{\theta}cos(\theta) + m_l\ddot{y}_t sin(\phi) + m_l\dot{y}_t\dot{\phi}cos(\phi) + m_l\ddot{l}sin^2(\theta) \\ & + m_l\ddot{l}sin^2(\phi) + m_l\ddot{l}cos^2(\theta)cos^2(\phi) + m_l\dot{l}\dot{\phi}sin^2(\theta)sin(2\phi) + m_l\dot{l}\dot{\phi}sin^2(\theta)sin(\phi)cos(\phi) \\ & + m_l\dot{l}\dot{\theta}sin(\theta)cos(\theta)sin^2(\phi) + m_l\dot{l}\dot{\theta}sin(2\theta)sin^2(\phi) + m_l l\ddot{\theta}sin(\theta)cos(\theta)sin^2(\phi) \\ & + m_l l\ddot{\phi}sin^2(\theta)sin(\phi)cos(\phi) + m_l l\dot{\theta}^2cos(2\theta)sin^2(\phi) + m_l l\dot{\phi}^2sin^2(\theta)cos(2\phi) \\ & + m_l l\dot{\theta}\dot{\phi}sin(\theta)cos(\theta)sin(2\phi) + m_l l\dot{\theta}\dot{\phi}sin(2\theta)sin(\phi)cos(\phi) + \frac{4J_l\ddot{l}}{r_l^2}\Bigg) - \Bigg(m_l\dot{x}_t\dot{\theta}cos(\theta) \\ & + m_l\dot{y}_t\dot{\phi}cos(\phi) + m_l l\dot{\theta}^2cos^2(\theta) + m_l l\dot{\theta}^2sin^2(\theta)cos^2(\phi) + m_l l\dot{\phi}^2cos^2(\phi) \\ & + m_l l\dot{\phi}^2cos^2(\theta)sin^2(\phi) + m_l\dot{l}\dot{\theta}sin(\theta)cos(\theta)sin^2(\phi) + m_l\dot{l}\dot{\phi}sin^2(\theta)sin(\phi)cos(\phi) \\ & + 2m_l l\dot{\theta}\dot{\phi}sin(\theta)cos(\theta)sin(\phi)cos(\phi)\Bigg) + m_l gcos(\theta)cos(\phi)\Bigg)\end{aligned} \tag{C.48}$$

$$\begin{aligned}\frac{K_{el}I_l}{r_l} - B_l\dot{l} = & m_l\ddot{l} + m_l\ddot{x}_t sin(\theta) + m_l\ddot{y}_t sin(\phi) - m_l l\dot{\theta}^2 - m_l l\dot{\phi}^2 + m_l\dot{l}\dot{\phi}sin^2(\theta)sin(2\phi) \\ & + m_l\dot{l}\dot{\theta}sin(2\theta)sin^2(\phi) + m_l l\ddot{\theta}sin(\theta)cos(\theta)sin^2(\phi) + m_l l\ddot{\phi}sin^2(\theta)sin(\phi)cos(\phi) \\ & - m_l\ddot{l}sin^2(\theta)sin^2(\phi) + m_l l\dot{\theta}^2cos^2(\theta)sin^2(\phi) + m_l l\dot{\phi}^2sin^2(\theta)cos^2(\phi) \\ & + m_l l\dot{\theta}\dot{\phi}sin(\theta)cos(\theta)sin(2\phi) + \frac{4J_l\ddot{l}}{r_l^2} - m_l gcos(\theta)cos(\phi)\end{aligned} \tag{C.49}$$

Finally, isolating $\ddot{l}$, **Equation C.51**.

$$\begin{aligned}& \frac{K_{el}I_l}{r_l} - B_l\dot{l} - m_l\ddot{x}_t sin(\theta) - m_l\ddot{y}_t sin(\phi) + m_l l\dot{\theta}^2 + m_l l\dot{\phi}^2 - m_l\dot{l}\dot{\phi}sin^2(\theta)sin(2\phi) \\ & - m_l\dot{l}\dot{\theta}sin(2\theta)sin^2(\phi) - m_l l\ddot{\theta}sin(\theta)cos(\theta)sin^2(\phi) - m_l l\ddot{\phi}sin^2(\theta)sin(\phi)cos(\phi) \\ & - m_l l\dot{\theta}^2cos^2(\theta)sin^2(\phi) - m_l l\dot{\phi}^2sin^2(\theta)cos^2(\phi) - m_l l\dot{\theta}\dot{\phi}sin(\theta)cos(\theta)sin(2\phi) \\ & + m_l gcos(\theta)cos(\phi) = \ddot{l}\left(m_l - m_l sin^2(\theta)sin^2(\phi) + \frac{4J_l}{r_l^2}\right)\end{aligned} \tag{C.50}$$

$$\begin{aligned}\ddot{l} = & \frac{1}{\left(m_l - m_l sin^2(\theta)sin^2(\phi) + \frac{4J_l}{r_l^2}\right)}\Bigg(\frac{K_{el}I_l}{r_l} - B_l\dot{l} - m_l\ddot{x}_t sin(\theta) - m_l\ddot{y}_t sin(\phi) + m_l l\dot{\theta}^2 + m_l l\dot{\phi}^2 \\ & - m_l\dot{l}\dot{\phi}sin^2(\theta)sin(2\phi) - m_l\dot{l}\dot{\theta}sin(2\theta)sin^2(\phi) - m_l l\ddot{\theta}sin(\theta)cos(\theta)sin^2(\phi) \\ & - m_l l\ddot{\phi}sin^2(\theta)sin(\phi)cos(\phi) - m_l l\dot{\theta}^2cos^2(\theta)sin^2(\phi) - m_l l\dot{\phi}^2sin^2(\theta)cos^2(\phi) \\ & - m_l l\dot{\theta}\dot{\phi}sin(\theta)cos(\theta)sin(2\phi) + m_l gcos(\theta)cos(\phi)\Bigg)\end{aligned} \tag{C.51}$$

Equation of motion for $\ddot{\theta}$

Since wind is not considered, there is no active force acting on the angle. Friction and air resistance for dampening the sway is also not considered, thus the resulting Lagrange equation is equal to zero.

$$0 = \frac{d}{dt}\frac{\partial}{\partial \dot{\theta}}L - \frac{\partial}{\partial \theta}L \tag{C.52}$$

Besides that change, the calculations are the same, with each term done separately as before.

$$\begin{aligned}\frac{d}{dt}\frac{\partial}{\partial \dot{\theta}}L = \frac{d}{dt}\frac{\partial}{\partial \dot{\theta}}\Bigg(&\frac{1}{2}m_t\left(\dot{x}_t^2 + \dot{y}_t^2\right) + \frac{1}{2}m_l\Bigg(\dot{x}_t{}^2 + \dot{y}_t^2 + 2\cdot\dot{l}\cdot\dot{x}_t\cdot sin(\theta) + 2\cdot l\cdot\dot{x}_t\cdot\dot{\theta}\cdot cos(\theta)\\
&+ 2\cdot\dot{l}\cdot\dot{y}_t\cdot sin(\phi) + 2\cdot l\cdot\dot{y}_t\cdot\dot{\phi}\cdot cos(\phi) + \dot{l}^2\cdot sin^2(\theta) + \dot{l}^2\cdot sin^2(\phi) + \dot{l}^2\cdot cos^2(\theta)cos^2(\phi)\\
&+ l^2\cdot\dot{\theta}^2\cdot cos^2(\theta) + l^2\cdot\dot{\theta}^2\cdot sin^2(\theta)cos^2(\phi) + l^2\cdot\dot{\phi}^2\cdot cos^2(\phi) + l^2\cdot\dot{\phi}^2\cdot cos^2(\theta)sin^2(\phi)\\
&+ 2\cdot l\cdot\dot{l}\cdot\dot{\theta}\cdot sin(\theta)cos(\theta)sin^2(\phi) + 2\cdot l\cdot\dot{l}\cdot\dot{\phi}\cdot sin^2(\theta)sin(\phi)cos(\phi)\\
&+ 2\cdot l^2\cdot\dot{\theta}\cdot\dot{\phi}\cdot sin(\theta)cos(\theta)sin(\phi)cos(\phi)\Bigg) + \frac{1}{2}J_x\left(\frac{\dot{x}_t}{r_x}\right)^2 + \frac{1}{2}J_y\left(\frac{\dot{y}_t}{r_y}\right)^2 + \frac{1}{2}J_l\left(\frac{2\dot{l}}{r_l}\right)^2\\
&+ m_l\cdot g\cdot l\cdot cos(\theta)cos(\phi)\Bigg)\end{aligned} \tag{C.53}$$

Eliminating terms without $\dot{\theta}$ and computing the partial derivative;

$$\begin{aligned}\frac{d}{dt}\frac{\partial}{\partial \dot{\theta}}L &= \frac{d}{dt}\frac{\partial}{\partial \dot{\theta}}\Bigg(\frac{1}{2}m_l\Bigg(2l\dot{x}_t\dot{\theta}cos(\theta) + l^2\dot{\theta}^2cos^2(\theta) + l^2\dot{\theta}^2sin^2(\theta)cos^2(\phi)\\
&\quad + 2l\dot{l}\dot{\theta}sin(\theta)cos(\theta)sin^2(\phi) + 2l^2\dot{\theta}\dot{\phi}sin(\theta)cos(\theta)sin(\phi)cos(\phi)\Bigg)\Bigg)\\
&= \frac{d}{dt}\Bigg(m_l l\dot{x}_t cos(\theta) + m_l l^2\dot{\theta}cos^2(\theta) + m_l l^2\dot{\theta}sin^2(\theta)cos^2(\phi)\\
&\quad + m_l l\dot{l}sin(\theta)cos(\theta)sin^2(\phi) + m_l l^2\dot{\phi}sin(\theta)cos(\theta)sin(\phi)cos(\phi)\Bigg)\end{aligned} \tag{C.54}$$

Computing the time derivative;

$$\begin{aligned}\frac{d}{dt}\frac{\partial}{\partial \dot{\theta}}L &= m_l\dot{l}\dot{x}_t cos(\theta) + m_l l\ddot{x}_t cos(\theta) - m_l l\dot{x}_t\dot{\theta}sin(\theta) + 2m_l l\dot{l}\dot{\theta}cos^2(\theta) + 2m_l l\dot{l}\dot{\theta}sin^2(\theta)cos^2(\phi)\\
&\quad + m_l l\dot{l}\dot{\theta}cos(2\theta)sin^2(\phi) + m_l l^2\ddot{\theta}cos^2(\theta) + m_l l^2\ddot{\theta}sin^2(\theta)cos^2(\phi) - m_l l^2\dot{\theta}^2sin(2\theta)sin^2(\phi)\\
&\quad + m_l\dot{l}^2sin(\theta)cos(\theta)sin^2(\phi) + m_l l^2\ddot{\phi}sin(\theta)cos(\theta)sin(\phi)cos(\phi) + m_l l^2\dot{\theta}\dot{\phi}cos(2\theta)sin(\phi)cos(\phi)\\
&\quad - m_l l^2\dot{\theta}\dot{\phi}sin^2(\theta)sin(2\phi) + 2m_l l\dot{l}\dot{\phi}sin(\theta)cos(\theta)sin(2\phi) + m_l l^2\dot{\phi}^2sin(\theta)cos(\theta)cos(2\phi)\end{aligned} \tag{C.55}$$

Calculating $\frac{\partial}{\partial\theta}L$ from **Equation C.52** separately:

$$\begin{aligned}\frac{\partial}{\partial\theta}L =& m_l\dot{l}\dot{x}_t cos(\theta) - m_l l\dot{x}_t\dot{\theta}sin(\theta) + m_l\dot{l}^2 sin(\theta)cos(\theta)sin^2(\phi) - m_l l^2\dot{\theta}^2 sin(\theta)cos(\theta)sin^2(\phi)\\ &- m_l l^2\dot{\phi}^2 sin(\theta)cos(\theta)sin^2(\phi) + m_l l\dot{l}\dot{\theta}cos(2\theta)sin^2(\phi) + 2m_l l\dot{l}\dot{\phi}sin(\theta)cos(\theta)sin(\phi)cos(\phi)\\ &+ m_l l^2\dot{\theta}\dot{\phi}cos(2\theta)sin(\phi)cos(\phi) - m_l glsin(\theta)cos(\phi)\end{aligned} \tag{C.56}$$

Every term is introduced back into **Equation C.52**:

$$\begin{aligned}0 =& +m_l l\ddot{x}_t cos(\theta) + 2m_l l\dot{l}\dot{\theta} - 2m_l l\dot{l}\dot{\theta}sin^2(\theta)sin^2(\phi) + m_l l^2\ddot{\theta} - m_l l^2\ddot{\theta}sin^2(\theta)sin^2(\phi)\\ &- m_l l^2\dot{\theta}^2 sin(\theta)cos(\theta)sin^2(\phi) + m_l l^2\ddot{\phi}sin(\theta)cos(\theta)sin(\phi)cos(\phi) - m_l l^2\dot{\theta}\dot{\phi}sin^2(\theta)sin(2\phi)\\ &+ 2m_l l\dot{l}\dot{\phi}sin(\theta)cos(\theta)sin(\phi)cos(\phi) + m_l l^2\dot{\phi}^2 sin(\theta)cos(\theta)cos^2(\phi) + m_l glsin(\theta)cos(\phi)\end{aligned} \tag{C.57}$$

Isolating $\ddot{\theta}$, **Equation C.58** can be obtained.

$$\begin{aligned}\ddot{\theta} =& \frac{1}{l\left(1 - sin^2(\theta)sin^2(\phi)\right)}\Bigg(-\ddot{x}_t cos(\theta) - 2\dot{l}\dot{\theta} + 2\dot{l}\dot{\theta}sin^2(\theta)sin^2(\phi) + l\dot{\theta}^2 sin(\theta)cos(\theta)sin^2(\phi)\\ &- l\ddot{\phi}sin(\theta)cos(\theta)sin(\phi)cos(\phi) + l\dot{\theta}\dot{\phi}sin^2(\theta)sin(2\phi) - 2\dot{l}\dot{\phi}sin(\theta)cos(\theta)sin(\phi)cos(\phi)\\ &- l\dot{\phi}^2 sin(\theta)cos(\theta)cos^2(\phi) - gsin(\theta)cos(\phi)\Bigg)\end{aligned} \tag{C.58}$$

Equation of motion for $\ddot{\phi}$

Since wind is not considered, there is no active force acting on the angle. Friction and air resistance for dampening the sway is also not considered, thus the resulting Lagrange equation is equal to zero.

$$0 = \frac{d}{dt}\frac{\partial}{\partial\dot{\phi}}L - \frac{\partial}{\partial\phi}L \tag{C.59}$$

The calculations are the same as the previous, with each term done separately as before. Firstly, the terms without $\dot{\phi}$ are eliminated. Then, the partial derivative in terms of $\dot{\phi}$ is

done;

$$\frac{d}{dt}\frac{\partial}{\partial\dot{\phi}}\left(\frac{1}{2}m_l\left(2l\dot{y}_t\dot{\phi}cos(\phi)+l^2\dot{\phi}^2cos^2(\phi)+l^2\dot{\phi}^2cos^2(\theta)sin^2(\phi)+2l\dot{l}\dot{\phi}sin^2(\theta)sin(\phi)cos(\phi)\right.\right.$$
$$\left.\left.+2l^2\dot{\theta}\dot{\phi}sin(\theta)cos(\theta)sin(\phi)cos(\phi)\right)\right)$$
$$=\frac{d}{dt}\left(m_l l\dot{y}_t cos(\phi)+m_l l^2\dot{\phi}cos^2(\phi)+m_l l^2\dot{\phi}cos^2(\theta)sin^2(\phi)+m_l l\dot{l}sin^2(\theta)sin(\phi)cos(\phi)\right.$$
$$\left.+m_l l^2\dot{\theta}sin(\theta)cos(\theta)sin(\phi)cos(\phi)\right) \tag{C.60}$$

Next, the above equation is derived in terms of time:

$$\frac{d}{dt}\frac{\partial}{\partial\dot{\phi}}=\frac{d}{dt}\left(m_l l\dot{y}_t cos(\phi)+m_l l^2\dot{\phi}cos^2(\phi)+m_l l^2\dot{\phi}cos^2(\theta)sin^2(\phi)+m_l l\dot{l}sin^2(\theta)sin(\phi)cos(\phi)\right.$$
$$\left.+m_l l^2\dot{\theta}sin(\theta)cos(\theta)sin(\phi)cos(\phi)\right)$$
$$=m_l l^2\ddot{\phi}+m_l l^2\ddot{\phi}sin^2(\theta)sin^2(\phi)+m_l l\ddot{l}sin^2(\theta)sin(\phi)cos(\phi)+m_l l\ddot{y}_t cos(\phi)+2m_l l\dot{l}\dot{\phi}$$
$$+m_l l^2\ddot{\theta}sin(\theta)cos(\theta)sin(\phi)cos(\phi)+m_l l\dot{y}_t cos(\phi)-m_l l\dot{y}_t\dot{\phi}sin(\phi)+m_l l\dot{l}\dot{\phi}sin^2(\theta)$$
$$-m_l l^2\dot{\phi}^2sin^2(\theta)sin(2\phi)+m_l l^2\dot{\theta}^2cos(2\theta)sin(\phi)cos(\phi)+m_l\dot{l}^2sin^2(\theta)sin(\phi)cos(\phi)$$
$$+2m_l l\dot{l}\dot{\theta}sin(2\theta)sin(\phi)cos(\phi)-m_l l^2\dot{\theta}\dot{\phi}sin(2\theta)sin^2(\phi)+m_l l^2\dot{\theta}\dot{\phi}sin(\theta)cos(\theta)cos(2\phi) \tag{C.61}$$

Calculating $\frac{\partial}{\partial\phi}L$ from **Equation C.59** separately

$$\frac{\partial}{\partial\phi}\left(\frac{1}{2}m_l\left(2\dot{l}\dot{y}_t sin(\phi)+2l\dot{y}_t\dot{\phi}cos(\phi)+\dot{l}^2sin^2(\phi)+\dot{l}^2cos^2(\theta)cos^2(\phi)+l^2\dot{\theta}^2sin^2(\theta)cos^2(\phi)\right.\right.$$
$$+l^2\dot{\phi}^2cos^2(\phi)+l^2\dot{\phi}^2cos^2(\theta)sin^2(\phi)+2l\dot{l}\dot{\theta}sin(\theta)cos(\theta)sin^2(\phi)+2l\dot{l}\dot{\phi}sin^2(\theta)sin(\phi)cos(\phi)$$
$$\left.\left.+2l^2\dot{\theta}\dot{\phi}sin(\theta)cos(\theta)sin(\phi)cos(\phi)\right)+m_l glcos(\theta)cos(\phi)\right)$$
$$=m_l\dot{l}\dot{y}_t cos(\phi)-m_l l\dot{y}_t\dot{\phi}sin(\phi)+m_l\dot{l}^2sin(\phi)cos(\phi)-m_l\dot{l}^2cos^2(\theta)sin(\phi)cos(\phi)$$
$$-m_l l^2\dot{\theta}^2sin^2(\theta)sin(\phi)cos(\phi)-m_l l^2\dot{\phi}^2sin(\phi)cos(\phi)+m_l l^2\dot{\phi}^2cos^2(\theta)sin(\phi)cos(\phi)$$
$$+2m_l l\dot{l}\dot{\theta}sin(\theta)cos(\theta)sin(\phi)cos(\phi)+m_l l\dot{l}\dot{\phi}sin^2(\theta)cos(2\phi)+m_l l^2\dot{\theta}\dot{\phi}sin(\theta)cos(\theta)cos(2\phi)$$
$$-m_l glcos(\theta)sin(\phi) \tag{C.62}$$

Re-introducing every term into **Equation C.59**:

$$\begin{aligned}0 =& m_l l^2 \ddot{\phi} + m_l l^2 \ddot{\phi} sin^2(\theta) sin^2(\phi) + m_l l \ddot{l} sin^2(\theta) sin(\phi) cos(\phi) + m_l l \ddot{y}_t cos(\phi) + 2 m_l \dot{l} \dot{\phi} \\ &+ m_l l^2 \ddot{\theta} sin(\theta) cos(\theta) sin(\phi) cos(\phi) - 2 m_l l \dot{l} \dot{\phi} sin^2(\theta) sin^2(\phi) + 2 m_l l \dot{l} \dot{\theta} sin(2\theta) sin(\phi) cos(\phi) \\ &+ m_l l^2 \dot{\phi}^2 cos^2(\theta) sin(\phi) cos(\phi) + m_l l^2 \dot{\theta}^2 cos^2(\theta) sin(\phi) cos(\phi) - m_l l^2 \dot{\theta} \dot{\phi} sin(2\theta) sin^2(\phi) \\ &+ m_l g l cos(\theta) sin(\phi)\end{aligned} \tag{C.63}$$

Isolating $\ddot{\phi}$, the last equation can be derived, **Equation C.64**.

$$\begin{aligned}\ddot{\phi} =& \frac{1}{l\,(1 + sin^2(\theta) sin^2(\phi))} \Big(- \ddot{l} sin^2(\theta) sin(\phi) cos(\phi) - \ddot{y}_t cos(\phi) - 2 \dot{l} \dot{\phi} + 2 \dot{l} \dot{\phi} sin^2(\theta) sin^2(\phi) \\ &- l \ddot{\theta} sin(\theta) cos(\theta) sin(\phi) cos(\phi) - 2 \dot{l} \dot{\theta} sin(2\theta) sin(\phi) cos(\phi) - l \dot{\phi}^2 cos^2(\theta) sin(\phi) cos(\phi) \\ &- l \dot{\theta}^2 cos^2(\theta) sin(\phi) cos(\phi) + l \dot{\theta} \dot{\phi} sin(2\theta) sin^2(\phi) - g cos(\theta) sin(\phi) \Big)\end{aligned} \tag{C.64}$$

Appendix D

Computed Torque Matrices

The final form of the matrices that form the dynamic system can be seen in the following equations:

$$M(z) = \begin{bmatrix} m_t + m_l + \frac{J_x}{r_x^2} - m_l cos^2(\theta) - \frac{m_l^2 sin^2(\theta)}{m_l + \frac{4J_l}{r_l^2}}, & 0, & 0 \\ 0, & m_l + \frac{4J_l}{r_l^2}, & 0 \\ 0, & 0, & l \end{bmatrix} \tag{D.1}$$

$$C(z,\dot{z})\dot{z} = \begin{bmatrix} 0, & 0, & m_l l\dot{\theta} sin(\theta)\left(1 - \frac{m_l}{m_l + \frac{4J_l}{r_l^2}}\right) \\ 0, & 0, & m_l l\dot{\theta} - \frac{m_l^2 l\dot{\theta} sin^2(\theta)\left(1 - \frac{m_l}{m_l + \frac{4J_l}{r_l^2}}\right)}{\left(m_t + m_l + \frac{J_x}{r_x^2} - m_l cos^2(\theta) - \frac{m_l^2 sin^2(\theta)}{m_l + \frac{4J_l}{r_l^2}}\right)} \\ 0, & -2\dot{\theta}, & -\frac{m_l l\dot{\theta} sin(\theta) cos(\theta)\left(1 - \frac{m_l}{m_l + \frac{4J_l}{r_l^2}}\right)}{\left(m_t + m_l + \frac{J_x}{r_x^2} - m_l cos^2(\theta) - \frac{m_l^2 sin^2(\theta)}{m_l + \frac{4J_l}{r_l^2}}\right)} \end{bmatrix} \tag{D.2}$$

$$V(z)\dot{z} = \begin{bmatrix} -B_x, & \frac{m_l B_l sin(\theta)}{m_l + \frac{4J_l}{r_l^2}}, & 0 \\ -\frac{B_x m_l sin(\theta)}{m_t + m_l + \frac{J_x}{r_x^2} - m_l cos^2(\theta)\frac{m_l^2 sin^2(\theta)}{m_l + \frac{4J_l}{r_l^2}}}, & -B_l - \frac{m_l^2 B_l sin^2(\theta)}{\left(m_l + \frac{4J_l}{r_l^2}\right)\left(m_t + m_l + \frac{J_x}{r_x^2} - m_l cos^2(\theta) - \frac{m_l^2 sin^2(\theta)}{m_l + \frac{4J_l}{r_l^2}}\right)}, & 0 \\ \frac{B_x cos(\theta)}{m_t + m_l + \frac{J_x}{r_x^2} - m_l cos^2(\theta) - \frac{m_l^2 sin^2(\theta)}{m_l + \frac{4J_l}{r_l^2}}}, & -\frac{m_l B_l sin(\theta) cos(theta)}{\left(m_l + \frac{4J_l}{r_l^2}\right)\left(m_t + m_l + \frac{J_x}{r_x^2} - m_l cos^2(\theta) - \frac{m_l^2 sin^2(\theta)}{m_l + \frac{4J_l}{r_l^2}}\right)}, & 0 \end{bmatrix} \tag{D.3}$$

$$G(z) = \begin{bmatrix} m_l g sin(\theta) cos(\theta) \left(1 - \frac{m_l}{m_l + \frac{4J_l}{r_l^2}}\right) \\ m_l g cos(\theta) - \frac{m_l^2 g sin^2(\theta) cos(\theta) \left(1 - \frac{m_l}{m_l + \frac{4J_l}{r_l^2}}\right)}{m_t + m_l + \frac{J_x}{r_x^2} - m_l cos^2(\theta) - \frac{m_l^2 sin^2(\theta)}{m_l + \frac{4J_l}{r_l^2}}} \\ - \frac{m_l g sin(\theta) cos^2(\theta) \left(1 - \frac{m_l}{m_l + \frac{4J_l}{r_l^2}}\right)}{m_t + m_l + \frac{J_x}{r_x^2} - m_l cos^2(\theta) - \frac{m_l^2 sin^2(\theta)}{m_l + \frac{4J_l}{r_l^2}}} - g sin(\theta) \end{bmatrix} \tag{D.4}$$

$$\tau = \begin{bmatrix} \frac{K_{ex}}{r_x}, & -\frac{m_l K_{el} sin(\theta)}{r_l m_l + \frac{4J_l}{r_l}} \\ -\frac{m_l K_{ex} sin(\theta)}{r_x \cdot \left(m_t + m_l + \frac{J_x}{r_x^2} - m_l cos^2(\theta) - \frac{m_l^2 sin^2(\theta)}{m_l + \frac{4J_l}{r_l^2}}\right)}, & \frac{m_l^2 K_{el} sin^2(\theta)}{\left(r_l m_l + \frac{4J_l}{r_l}\right) \cdot \left(m_t + m_l + \frac{J_x}{r_x^2} - m_l cos^2(\theta) - \frac{m_l^2 sin^2(\theta)}{m_l + \frac{4J_l}{r_l^2}}\right)} + \frac{K_{el}}{r_l} \\ -\frac{K_{ex} cos(\theta)}{r_x \cdot \left(m_t + m_l + \frac{J_x}{r_x^2} - m_l cos^2(\theta) - \frac{m_l^2 sin^2(\theta)}{m_l + \frac{4J_l}{r_l^2}}\right)}, & \frac{m_l K_{el} sin(\theta) cos(\theta)}{\left(r_l m_l + \frac{4J_l}{r_l}\right) \cdot \left(m_t + m_l + \frac{J_x}{r_x^2} - m_l cos^2(\theta) - \frac{m_l^2 sin^2(\theta)}{m_l + \frac{4J_l}{r_l^2}}\right)} \end{bmatrix} \tag{D.5}$$

Bibliography

[1] R. George. *Ninety Percent of Everything: Inside Shipping, the Invisible Industry That Puts Clothes on Your Back, Gas in Your Car, and Food on Your Plate*. New York: Metropolitan Books, 2013.

[2] UNCTAD. *Container port throughput, annual*. 2020. URL: https://unctadstat.unctad.org/wds/TableViewer/tableView.aspx?ReportId=13321 (visited on 09/30/2010).

[3] F. Martínez et al. "Gantry crane operations to transfer containers between trains: A simulation study of a Spanish terminal". In: *Transportation Planning and Technology - TRANSPORT PLANNING TECHNOL* 27 (Aug. 2004), pp. 261–284. DOI: 10.1080/0308106042000263069.

[4] Y. Yang et al. "Internet of things for smart ports: Technologies and challenges". In: *IEEE Instrumentation & Measurement Magazine* 21 (Feb. 2018), pp. 34–43. DOI: 10.1109/MIM.2018.8278808.

[5] O. Sawodny, H. Aschemann, and S. Lahres. "An automated gantry crane as a large workspace robot". In: *Control Engineering Practice* 10.12 (2002), pp. 1323–1338. ISSN: 0967-0661. DOI: https://doi.org/10.1016/S0967-0661(02)00097-7. URL: https://www.sciencedirect.com/science/article/pii/S0967066102000977.

[6] C. Zhimei et al. "Obstacle Avoidance Path Planning of Bridge Crane Based on Improved RRT Algorithm". In: *Journal of System Simulation* 33.8 (2021). DOI: 10.16182/j.issn1004731x.joss.20-0272. URL: https://www.china-simulation.com/EN/10.16182/j.issn1004731x.joss.20-0272.

[7] X. Du, X.-Z. Yu, and X.-Y. Yun. "Scenario modeling and operation path planning of autonomous driving tower cranes". In: *Journal of Physics: Conference Series* 2365.1 (Nov. 2022), p. 012020. DOI: 10.1088/1742-6596/2365/1/012020. URL: https://dx.doi.org/10.1088/1742-6596/2365/1/012020.

[8] J. Waikoonvet, N. Suksabai, and I. Chuckpaiwong. "Collision-free Path Planning for Overhead Crane System Using Modified Ant Colony Algorithm". In: *2021 9th International Electrical Engineering Congress (iEECON)*. Mar. 2021, pp. 325–328. DOI: 10.1109/iEECON51072.2021.9440291.

[9] M. Vu et al. “Fast motion planning for a laboratory 3D gantry crane in the presence of obstacles”. In: *IFAC-PapersOnLine* 53 (July 2020), pp. 9508–9514. DOI: 10.1016/j.ifacol.2020.12.2427.

[10] Z. Wu and X. Xia. “Optimal motion planning for overhead cranes”. In: *IET Control Theory & Applications* 8.17 (2014), pp. 1833–1842. DOI: https://doi.org/10.1049/iet-cta.2014.0069.

[11] X. Zhang, Y. Fang, and N. Sun. “Minimum-Time Trajectory Planning for Underactuated Overhead Crane Systems With State and Control Constraints”. In: *IEEE Transactions on Industrial Electronics* 61.12 (Dec. 2014), pp. 6915–6925. ISSN: 1557-9948. DOI: 10.1109/TIE.2014.2320231.

[12] N. Sun et al. “A Novel Kinematic Coupling-Based Trajectory Planning Method for Overhead Cranes”. In: *IEEE/ASME Transactions on Mechatronics* 17.1 (Feb. 2012), pp. 166–173. ISSN: 1941-014X. DOI: 10.1109/TMECH.2010.2103085.

[13] Y. Fang et al. “A Motion Planning-Based Adaptive Control Method for an Underactuated Crane System”. In: *IEEE Transactions on Control Systems Technology* 20.1 (Jan. 2012), pp. 241–248. ISSN: 1558-0865. DOI: 10.1109/TCST.2011.2107910.

[14] S. Nagai, A. Kaneshige, and S. Ueki. “Three-dimensional obstacle avoidance online path-planning method for autonomous mobile overhead crane”. In: *2011 IEEE International Conference on Mechatronics and Automation*. Aug. 2011, pp. 1497–1502. DOI: 10.1109/ICMA.2011.5985971.

[15] A. KANESHIGE et al. “Development of the Autonomous Overhead Travelling Crane with Real Time Path-Planning Based on Obstacle Information”. In: *IFAC Proceedings Volumes* 45.24 (2012). 13th IFAC Symposium on Control in Transportation Systems, pp. 30–35. ISSN: 1474-6670. DOI: https://doi.org/10.3182/20120912-3-BG-2031.00007. URL: https://www.sciencedirect.com/science/article/pii/S147466701534742X.

[16] V. Suvorov, M. Bahrami, and E. Akchurin. “Anti sway tuned control of gantry cranes”. In: *SN Applied Sciences* 3 (Mar. 2021), p. 729. ISSN: 2523-3971. DOI: 10.1007/s42452-021-04719-w.

[17] M. Solihin and A. Legowo. “Fuzzy-tuned PID Anti-swing Control of Automatic Gantry Crane”. In: *Journal of Vibration and Control* 16.1 (Jan. 2010), pp. 125–145. DOI: 10.1177/1077546309103421.

[18] K. L. Sorensen, W. Singhose, and S. Dickerson. “A controller enabling precise positioning and sway reduction in bridge and gantry cranes”. In: *Control Engineering Practice* 15.7 (2007). Special Issue on Award Winning Applications, pp. 825–837. ISSN: 0967-0661. DOI: https://doi.org/10.1016/j.conengprac.2006.03.005. URL: https://www.sciencedirect.com/science/article/pii/S0967066106000785.

[19] H. I. Jaafar et al. “PSO-tuned PID controller for a nonlinear gantry crane system”. In: *2012 IEEE International Conference on Control System, Computing and Engineering*. Nov. 2012, pp. 515–519. DOI: 10.1109/ICCSCE.2012.6487200.

[20] J. Kennedy and R. Eberhart. “Particle swarm optimization”. In: *Proceedings of ICNN'95 - International Conference on Neural Networks*. Vol. 4. Nov. 1995, 1942–1948 vol.4. DOI: 10.1109/ICNN.1995.488968.

[21] A. G. Gad. “Particle Swarm Optimization Algorithm and Its Applications: A Systematic Review”. In: *Archives of Computational Methods in Engineering* 29.5 (Nov. 2022), pp. 2531–2561. ISSN: 1886-1784. DOI: 10.1007/s11831-021-09694-4.

[22] S Girija and A. Joshi. “Fast Hybrid PSO-APF Algorithm for Path Planning in Obstacle Rich Environment”. In: *IFAC-PapersOnLine* 52.29 (2019). 13th IFAC Workshop on Adaptive and Learning Control Systems ALCOS 2019, pp. 25–30. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2019.12.616. URL: https://www.sciencedirect.com/science/article/pii/S2405896319325583.

[23] X. Cheng et al. “An Improved PSO-GWO Algorithm With Chaos and Adaptive Inertial Weight for Robot Path Planning”. In: *Frontiers in Neurorobotics* 15 (2021). ISSN: 1662-5218. DOI: 10.3389/fnbot.2021.770361. URL: https://www.frontiersin.org/articles/10.3389/fnbot.2021.770361.

[24] O. M. Nezami, A. Bahrampour, and P. Jamshidlou. “Dynamic Diversity Enhancement in Particle Swarm Optimization (DDEPSO) Algorithm for Preventing from Premature Convergence”. In: *Procedia Computer Science* 24 (2013). 17th Asia Pacific Symposium on Intelligent and Evolutionary Systems, IES2013, pp. 54–65. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2013.10.027. URL: https://www.sciencedirect.com/science/article/pii/S1877050913011691.

[25] D. Aleksendrić and P. Carlone. “5 - Composite materials – modelling, prediction and optimization”. In: *Soft Computing in the Design and Manufacturing of Composite Materials*. Ed. by D. Aleksendrić and P. Carlone. Oxford: Woodhead Publishing, 2015, pp. 61–289. ISBN: 978-1-78242-179-5. DOI: https://doi.org/10.1533/9781782421801.61. URL: https://www.sciencedirect.com/science/article/pii/B9781782421795500055.

[26] H. Miao and Y.-C. Tian. “Dynamic robot path planning using an enhanced simulated annealing approach”. In: *Applied Mathematics and Computation* 222 (2013), pp. 420–437. ISSN: 0096-3003. DOI: https://doi.org/10.1016/j.amc.2013.07.022. URL: https://www.sciencedirect.com/science/article/pii/S0096300313007728.

[27] Q. Zhu, Y. Yan, and Z. Xing. “Robot Path Planning Based on Artificial Potential Field Approach with Simulated Annealing”. In: *Sixth International Conference on Intelligent Systems Design and Applications*. Vol. 2. Oct. 2006, pp. 622–627. DOI: 10.1109/ISDA.2006.253908.

[28] M. Dorigo and L. Gambardella. "Ant colony system: a cooperative learning approach to the traveling salesman problem". In: *IEEE Transactions on Evolutionary Computation* 1.1 (Apr. 1997), pp. 53–66. ISSN: 1941-0026. DOI: 10.1109/4235.585892.

[29] K. Akka and F. Khaber. "Mobile robot path planning using an improved ant colony optimization". In: *International Journal of Advanced Robotic Systems* 15.3 (2018), p. 1729881418774673. DOI: 10.1177/1729881418774673. eprint: https://doi.org/10.1177/1729881418774673. URL: https://doi.org/10.1177/1729881418774673.

[30] Y. Z. Cong and S. G. Ponnambalam. "Mobile robot path planning using ant colony optimization". In: *2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*. July 2009, pp. 851–856. DOI: 10.1109/AIM.2009.5229903.

[31] K. Sedighi et al. "Autonomous local path planning for a mobile robot using a genetic algorithm". In: vol. 2. July 2004, 1338 –1345 Vol.2. ISBN: 0-7803-8515-2. DOI: 10.1109/CEC.2004.1331052.

[32] A. Petrowski and S. Ben Hamida. *Evolutionary Algorithms*. Jan. 2016.

[33] A. Slowik and H. Kwasnicka. "Evolutionary algorithms and their applications to engineering problems". In: *Neural Computing and Applications* 32 (Aug. 2020), pp. 12363–12379. DOI: 10.1007/s00521-020-04832-8.

[34] R. Kala. "Multi-robot path planning using co-evolutionary genetic programming". In: *Expert Systems with Applications* 39.3 (2012), pp. 3817–3831. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2011.09.090. URL: https://www.sciencedirect.com/science/article/pii/S0957417411014138.

[35] M. A. Contreras-Cruz, V. Ayala-Ramirez, and U. H. Hernandez-Belmonte. "Mobile robot path planning using artificial bee colony and evolutionary programming". In: *Applied Soft Computing* 30 (2015), pp. 319–328. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2015.01.067. URL: https://www.sciencedirect.com/science/article/pii/S1568494615000885.

[36] J. Zhao et al. "Local Path Planning Algorithm for UGV Based on Improved Covariance Matrix Adaptive Evolution Strategy". In: *2021 33rd Chinese Control and Decision Conference (CCDC)*. May 2021, pp. 1085–1091. DOI: 10.1109/CCDC52312.2021.9601399.

[37] X. Yu, C. Li, and J. Zhou. "A constrained differential evolution algorithm to solve UAV path planning in disaster scenarios". In: *Knowledge-Based Systems* 204 (2020). ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2020.106209. URL: https://www.sciencedirect.com/science/article/pii/S0950705120304263.

[38] L. Zadeh. "Fuzzy sets". In: *Information and Control* 8.3 (1965), pp. 338–353. ISSN: 0019-9958. DOI: https://doi.org/10.1016/S0019-9958(65)90241-X. URL: https://www.sciencedirect.com/science/article/pii/S001999586590241X.

[39] L. Zadeh. "Fuzzy logic". In: *Computer* 21.4 (Apr. 1988), pp. 83–93. ISSN: 1558-0814. DOI: 10.1109/2.53.

[40] A. Pandey et al. "Path planning navigation of mobile robot with obstacles avoidance using fuzzy logic controller". In: *2014 IEEE 8th International Conference on Intelligent Systems and Control (ISCO)*. Jan. 2014, pp. 39–41. DOI: 10.1109/ISCO.2014.7103914.

[41] M. Wang and Liu. "Fuzzy logic based robot path planning in unknown environment". In: *2005 International Conference on Machine Learning and Cybernetics*. Vol. 2. Aug. 2005, 813–818 Vol. 2. DOI: 10.1109/ICMLC.2005.1527055.

[42] A. Hentout, A. Maoudj, and M. Aouache. "A review of the literature on fuzzy-logic approaches for collision-free path planning of manipulator robots". In: *Artificial Intelligence Review* (2022). ISSN: 1573-7462. DOI: 10.1007/s10462-022-10257-7. URL: https://doi.org/10.1007/s10462-022-10257-7.

[43] Y. Hwang and N. Ahuja. "A potential field approach to path planning". In: *IEEE Transactions on Robotics and Automation* 8.1 (Feb. 1992), pp. 23–32. ISSN: 2374-958X. DOI: 10.1109/70.127236.

[44] Y. Wang and G. Chirikjian. "A new potential field method for robot path planning". In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*. Vol. 2. Apr. 2000, 977–982 vol.2. DOI: 10.1109/ROBOT.2000.844727.

[45] X. Lin, Z.-Q. Wang, and X.-Y. Chen. "Path Planning with Improved Artificial Potential Field Method Based on Decision Tree". In: *2020 27th Saint Petersburg International Conference on Integrated Navigation Systems (ICINS)*. May 2020, pp. 1–5. DOI: 10.23919/ICINS43215.2020.9134006.

[46] G. Abbas et al. "Constrained Model-Based Predictive Controller for a High-Frequency Low-Power DC-DC Buck Converter". In: *International Journal on Electrical Engineering and Informatics* 5 (Sept. 2013), pp. 316–339. DOI: 10.15676/ijeei.2013.5.3.6.

[47] F. Behrooz et al. "Review of Control Techniques for HVAC Systems—Nonlinearity Approaches Based on Fuzzy Cognitive Maps". In: *Energies* 11 (Feb. 2018), p. 495. DOI: 10.3390/en11030495.

[48] P. Airikka. "Advanced control methods for industrial process control". In: *Computing & Control Engineering Journal* 15 (July 2004), pp. 18–23. DOI: 10.1049/cce:20040303.

[49] *Design Neural Network Predictive Controller in Simulink*. URL: https://se.mathworks.com/help/deeplearning/ug/design-neural-network-predictive-controller-in-simulink.html.

[50] S. Johansen et al. "Control system for automation of overhead crane". MA thesis. Aalborg: Aalborg University, May 2022. URL: https://projekter.aau.dk/projekter/da/studentthesis/control-system-for-automation-of-overhead-crane(a193ac0d-0b00-4d7c-9ff5-c23a4a09ab1e).html.

[51] M. T. Incorporated. "ATmega640/1280/1281/2560/2561 Datasheet". In: (2020). URL: https://www.microchip.com/en-us/product/ATMEGA2560.

[52] J. D. Irwin, R. M. Nelms, and A. P. R. Mark Nelms. "Engineering Circuit Analysis". In: *Engineering Circuit Analysis*. Vol. 11. 2015.

[53] N. Blet et al. "NONLINEAR MPC VERSUS MPC USING ON-LINE LINEARISATION — A COMPARATIVE STUDY". In: *IFAC Proceedings Volumes* 35.1 (2002). 15th IFAC World Congress, pp. 147–152. ISSN: 1474-6670. DOI: https://doi.org/10.3182/20020721-6-ES-1901.00593. URL: https://www.sciencedirect.com/science/article/pii/S1474667015390145.

[54] T. M. Inc. *Optimization Toolbox version: 9.4 (R2022b)*. Natick, Massachusetts, United States, 2023. URL: https://se.mathworks.com/help/mpc/ug/optimization-problem.html.

[55] S. H. Zak. *An introduction to model-based predictive control (MPC)*. 2017.

[56] M. Grasmair. *MINIMIZERS OF OPTIMIZATION PROBLEMS*. 2023. URL: https://wiki.math.ntnu.no/_media/tma4180/2015v/existence.pdf.

[57] T. R. Kuphaldt. "Lessons In Electric Circuits, Volume I–DC". In: ().