Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

Even Rise Gregersen

Simultaneous Control of Crane Tip Position and Crane Load Oscillation Dampening on Shipboard Cranes

Master's thesis in Subsea Technology
Supervisor: Christian Holden
June 2023

NTNU
Norwegian University of
Science and Technology

Even Rise Gregersen

Simultaneous Control of Crane Tip Position and Crane Load Oscillation Dampening on Shipboard Cranes

Master's thesis in Subsea Technology
Supervisor: Christian Holden
June 2023

Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

NTNU
Norwegian University of
Science and Technology

Preface

This master thesis is written as a part of the course TPK4960 “Robotics and Automation, Master’s Thesis ” in the spring of 2023. This thesis constitutes the final part of my master’s degree in Subsea Technology at the Norwegian University of Science and Technology and is written in cooperation with Sintef Ocean.

First of all, I would like to thank my thesis supervisor Christian Holden, whose keen interest and thorough help have been invaluable. Linn Evjemo from Sintef Ocean deserves a big thanks for her interest and input. I would also like to thank Olav Egeland for providing some of the fundamental theories for this project.

Abstract

For operations in aquaculture and offshore industries shipboard cranes are extensively used. Waves induce disturbances into the crane through the ship and wind disturbances affect the crane load. These disturbances can cause pendulation in the crane load that can put equipment and more importantly personnel at risk. To minimise these unwanted pendulations a feedback control system can be employed.

In this master's thesis, such control systems are investigated. Three different control systems are designed together with dynamic and kinematic models for the crane-tip and crane load. The control goals are to create simultaneous control of the crane-tip position and crane load dampening. Different types of control are tried such as feedforward, feedback, cascade and nonlinear model predictive control in different combinations.

Both promising and non-promising results were found. Using a feedback controller for damping with feedforward control for the crane tip worked for the crane without disturbances. Using a controller with feedback for both crane load damping and crane tip position, and a controller with nonlinear model predictive control for crane tip position and Lyapunov-based damping showed less promising results. Evaluation of the dynamic model for the crane is promising there were, however, problems with the inverse kinematic solution of the third crane joint position together with the solution for the velocity and acceleration of the first joint.

Sammendrag

For operasjoner i akvakultur og offshore industriene er skipskraner til en stor grad tatt i bruk. Bølger har evnene til å indusere forstyrrelser fra skipet inn i kranen og vind påvirker kranlasten. Disse forstyrrelsene kan forårsake oscilleringer i kranlasten som kan sette utstyr og personal i fare. For å minimere disse uønskede oscilleringene kan en tilbakekoblingsregulator benyttes.

I denne masteroppgaven er slike kontrollsystem utforsket. Tre forskjellige kontrollsystemer er designet sammen med dynamiske og kinematiske modeller for kran og kranlast. Reguleringsmålet er å oppnå regulering av kran-tupp posisjonen og demping av kranlasten samtidig. Flere reguleringsmetoder er prøvd som fremoverkobling, tilbakekobling, kaskade og ikke linear modell prediktiv styring.

Både lovende og ikke lovende resultater ble oppnådd. Ved bruk av tilbakekobling for kranlastdemping og fremoverkobling for kran-tupp posisjon ble gode resultat oppnådd for kranen uten forstyrrelser. Tilbakekobling for både demping og posisjon samt ikke linear modell prediktiv styring for posisjon med Lyapunov-basert demping visste mindre lovende resultater. Evaluering av de dynamiske modellene for kran og kranlast er lovende, problemer med løsningen av posisjonen til kranens tredje ledd sammen med hastigheten og akselerasjonen til det første leddet ble funnet i invers kinematikken.

Contents

List of Figures

List of Tables

Chapter 1.

Introduction

1.1. Background and motivation

Cranes are an essential device for offshore and subsea operations and are often mounted to floaters and ships. Some common applications of cranes in these environments are the lifting of heavy loads, transfer of payloads from ship to ship and lowering of components for subsea installations down into the ocean.

When a load is hung from the crane tip by a wire, this load will swing in all directions like a spherical pendulum. When uncontrolled the pendulations of the load can be a significant risk for personnel and equipment. Wind and waves are common disturbances for these ship-mounted cranes. Waves will cause movements in the ship that will induce further movement into the crane and payload. Furthermore if one has a large and heavy crane on a smaller vessel, movement in the crane, for example, due to strong winds, can induce movements in the vessel. In a report published by General Electric, it is estimated that unexpected downtime accounts for roughly $49 million in losses annually in the offshore oil and gas industry [1]. Reducing crane pendulation can therefore be beneficial as it will make crane operations more reliable and safe for both personnel and equipment, and it could potentially reduce the operation time of the crane.

The last few decades have seen a lot of research in control systems for crane load dampening. Ramli et al. [2] and Cao and Li [3] have conducted comprehensive literature reviews on the topic of shipboard crane control. They both agree on the need for feedback control in this type of crane. Several types of control have been tried.

The most studied types of cranes are, however, land-based stationary cranes. These types of cranes do not experience the same amount of unpredictable disturbances like waves. The operator can because of this more easily predict the load pendulations counteracting them accordingly. This also means that controllers that work for this type of crane will not necessarily work for a ship-mounted crane. The unpredictability and sometimes the severity of the disturbances at sea will require feedback control, and precise models of the crane, load, ship and disturbances.

A vast array of controllers have been successfully employed for crane applications. A straightforward and effective starting point is the PID controller. These are, however, linear controllers which can prove a challenge when put against nonlinear systems. As a result, they can often be seen combined with other techniques to improve their perfor-

mance. For example, using Neural Network Self Tuning has been used to estimate and tune PD controller gains for gantry cranes. Other more complex controller schemes such as the Linear Quadratic Regulator (LQR) and Model Predictive Control (MPC) have also seen favourable results when used for crane control [2].

Sensor systems will also be important in the control of shipboard cranes. Tracking the payload for example using IMUs (Inertial Measuring Units) which can measure their own angular velocity and linear acceleration and computer vision has been tried. Using the same crane as used in this thesis the thesis " Crane Pose Estimation using IMU and Computer Vision" by Espen Nilsen covers the topic [4].

1.2. Problem description

The goal of the project is to automate a shipboard crane for use in aquaculture operations. The crane in question can be seen in figure 1.1, where it is placed upon the ship Torra. The crane is actuated using hydraulics and is of the model Maxilift ML 270L.2. Both the ship and crane are owned by Sintef Ocean, who is a collaborator in this project. The actual crane has as of now no sensor system or automatic control and is controlled manually using joysticks. This means the crane has to be retrofitted with a new sensor- and control system.

Figure 1.1.: The boat Torra, with the crane [4].

This thesis is a continuation of the work done in [5]. The research goal of the thesis is to achieve simultaneous control of the crane tip position and dampening of the load oscillations. Several controllers were designed and tried to achieve the above control goals. We also need a dynamic and kinematic model for the crane and a dynamic model for the crane load. Since this thesis will focus on the control system all measurements will be

considered perfect for the simulations. The research question formulated for the project is:

"Can a satisfactory controller be designed that dampens the crane load oscillations simultaneously with the crane tip position on shipboard cranes?"

The main goals with stretch goals of the thesis are given as

Main goals:

- Expand the simulation from [5] to include crane motion
- Test the expanded model in simulation
- Develop multiple control schemes. Test and compare their performance in simulation

Stretch goals:

- Expand the model to include wave and wind effects
- Test the expanded model in simulation
- Expand control schemes developed to counter-act the environmental effects on the load, test the results in simulation, and compare to the undisturbed case

1.3. Report outline

The Thesis is divided into five chapters:

Introduction Introduces the background and motivation for the thesis together with the research question and thesis goals.

Theory Relevant theories about the kinematic and dynamic modelling of cranes and crane load, and the control theory used are presented.

Method Here the relevant equations and material used to make the models are presented.

Results The results from the simulations are presented.

Analysis and Discussion The results are analysed and discussed to form a conclusion.

Conclusion and Future Work A conclusion is presented and future work needed to complete the project is discussed.

Chapter 2.

Theory

This chapter contains the most important theory used throughout the thesis. First, the theory for designing the dynamic and kinematic models is presented, and lastly, the control theory used is presented.

2.1. Crane

Cranes are mechanical devices and its most primary function is to lift up heavy objects and move them to another location. Abdel-Rahman describes in his journal paper "Dynamics and Control of Cranes: A Review" [6] three different types of cranes. These are the rotary-, gantry and boom cranes.

In figure 2.1 (b) we see the rotary crane. This crane moves the load by rotating on its vertical axis and translating along its horizontal axis. Figure 2.1 (c) shows the gantry crane, here translation along two axes is used to move the load. These two crane types will usually utilise variable cable lengths to lower the load when at the right position. The boom crane, depicted as (a) in figure 2.1, typically rotates its base and can lower its boom to place the load at the right position. It is also typically smaller and more mobile than the other crane types, often found on boats and various types of road vehicles. Usual additions to boom cranes are prismatic joints to extend the boom and variable cable length to lower the load.

The actuators used for cranes are typically done by electric, hydraulic or combustion motors or some combination of these. Because of the nature of cranes high output forces are usually preferred over speed, when it comes to actuation. This will ordinarily result in slower crane movements but makes the crane able to lift heavier loads.

Unless dampened by a control system, pendulum-like oscillations can persist for a considerable amount of time if not dampened by a control system. This is because of the low-dampening characteristic exhibited by loads suspended by a wire. The University of California conducted "Dry" tests of crane systems without a control system. This revealed damping values in the range of 0.1 % to 0.5 % of critical damping [7]. The importance of effective control systems for cranes is highlighted by these numbers.

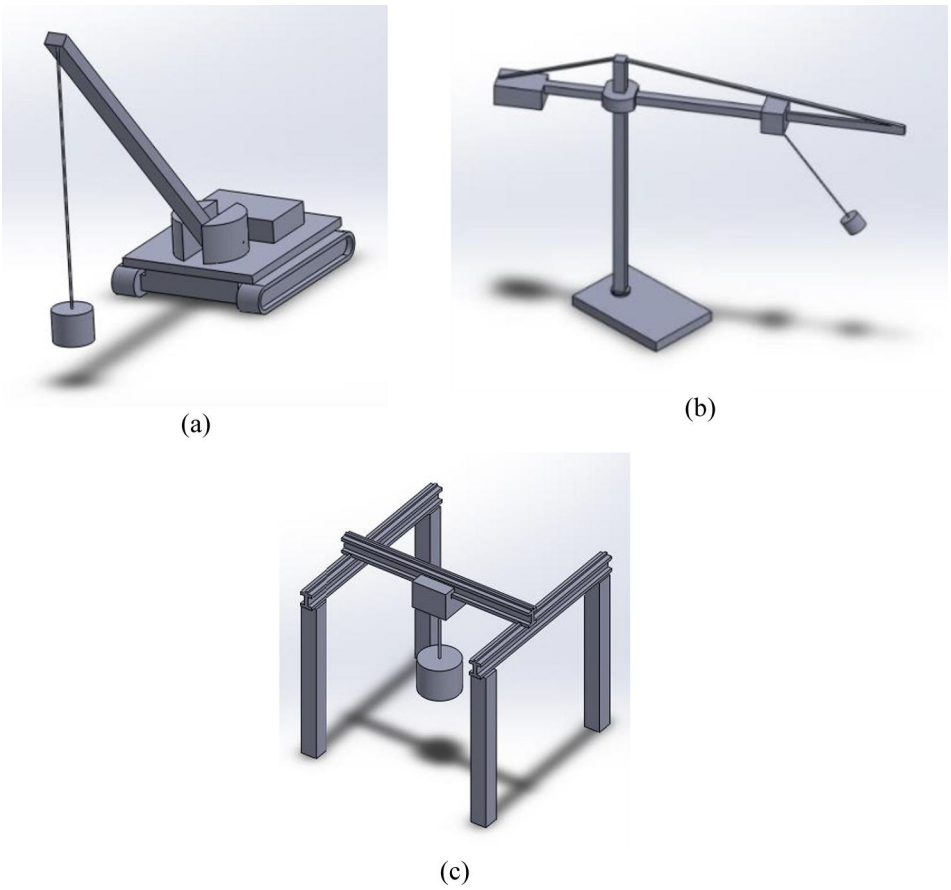


Figure 2.1.: Illustration of the crane types. (a): Boom crane. (b): Rotary crane. (c): Gantry crane. [2]

2.2. Pendulum

A pendulum can be described as a mass suspended by a line attached to a pivot. The pendulum swings freely around the pivot point. The bottommost position of the pendulum is called it equilibrium point. When the pendulum deviates from this point, gravity will induce a restoring force which will attempt to restore it to its equilibrium.

Here the pendulum is used as a model for the crane load, therefore, a pendulum-based model with a moving attachment point is derived. The moving attachment point will allow us to try and dampen out the oscillations of the pendulum with movements from the crane-tip. The crane-tip acceleration is provided as input to the pendulum merging the models. This also consequently gives us two control objectives: one is the dampening of the pendulum oscillations, and the second is to keep the attachment point, or the crane-tip in this case, at the position we want it to stay at.

The use of a pendulum-based model to simulate crane loads is not a novel concept. It has been explored by Olav Egeland in his work note " Crane-Load Dynamics and Control" [8] where several pendulum models both 1D and 2D with and without moving attachment points are derived by Lagrange Formulation. Geir Ole Tysse uses similar models in his PhD thesis "Modelling and Control of Ship-mounted Cranes" [9]. In the project report

preceding this thesis the concept of using a pendulum for the crane load simulation where explored with pendulation-reducing control systems [5].

2.3. Crane kinematics

The materials presented in 2.3 and 2.4 are taken from "Modern Robotics: Mechanics, Planning, and Control" by Lynch and Park [10] and " Robotics Modelling, Planning and Control" by Siciliano et al. [11].

It is possible to find the crane'sposiiton and orientation using a rigid body model. Using the joint variables q as an input, the goal is to create a function that finds the orientation and position of the crane end effector

$$f(q) = x = \begin{bmatrix} \vartheta_x \\ \vartheta_y \\ \vartheta_z \\ x \\ y \\ z \end{bmatrix} \tag{2.1}$$

Solving this problem is commonly known as forward kinematics. Solving it the other way, finding the joint variable using the end effector orientation and position is called inverse kinematics.

2.3.1. Useful Functions and Theory for Crane Kinematic and Dynamic Modelling

This section will provide some useful mathematical relationships, functions and theories for 2.3 and 2.4.

Kinematics uses coordinate frames and transformations between them to analyse a robot's motion.

A homogeneous transformation matrix is in the special Euclidean group $SE(3)$ and is always 4×4 matrix in $\mathbb{R}^3$. If one has two frames $\{s\}$ and $\{b\}$ where the first is a fixed frame and the latter a body frame. One can represent the position and orientation of $\{b\}$ in $\{s\}$-coordinates. This is done by using a rotation matrix $R \in SO(3)$ which represents the orientation of frame $\{b\}$ in $\{s\}$, and the vector $p \in \mathbb{R}^3$ which represents the origin of $\{b\}$ in $\{s\}$. The Transformation matrix will then have the following form.

$$T = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_1 \\ r_{21} & r_{22} & r_{23} & p_2 \\ r_{31} & r_{32} & r_{33} & p_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

Multiplication between transformation matrices causes the cancellation of subscripts as follows

$$T_1^0 T_2^1 = T_2^0 \tag{2.3}$$

This relationship allows us to represent frame 2 with respect to frame 0, this will prove valuable for calculating the forward kinematics.

Some other useful mathematical relations and functions are

Skew-Symmetric Matrix (3×3) The skew is denoted as $[x]$ and for the vector $x \in \mathbb{R}^3$ the skew-symmetric matrix is as follows

$$[x] = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix} \tag{2.4}$$

Skew-Symmetric Matrix of a Twist (6×6) Using the (3×3) skew-symmetric matrix the skew of a twist $\mathcal{V}$ is defined as

$$[\mathcal{V}] = \begin{bmatrix} [\omega] & v \\ 0 & 0 \end{bmatrix} \tag{2.5}$$

Adjoint The adjoint representation of a homogeneous transformation matrix T, $[Ad_T]$, is written as

$$[Ad_T] = \begin{bmatrix} R & 0 \\ [p]R & R \end{bmatrix} \tag{2.6}$$

Adjoint map For a twist $\mathcal{V}$ the adjoint map given a transformation matrix T can be written as

$$\mathcal{V}' = [Ad_T]\mathcal{V} \tag{2.7}$$

2.3.2. Forward Kinematics

The forward kinematics solves the position and orientation of the end effector based on the joint configuration. A typical open-chain robot is modelled as n rigid body links connected by joints. In figure 2.2, a 3R planar open-chain robot is shown, 3R means it has 3 revolute joints. Although it is possible to find the forward kinematics using basic trigonometry, this would become considerably more complicated the more joints there are. A more common and systematic way to solve the Forward kinematics is to attach a reference frame to each link. In figure 2.2 we have three link reference frames {1}, {2} and {3}. One can write the forward kinematics for this example as four homogeneous transformation matrices.

$$T_{04} = T_{01} T_{12} T_{23} T_{34} \tag{2.8}$$

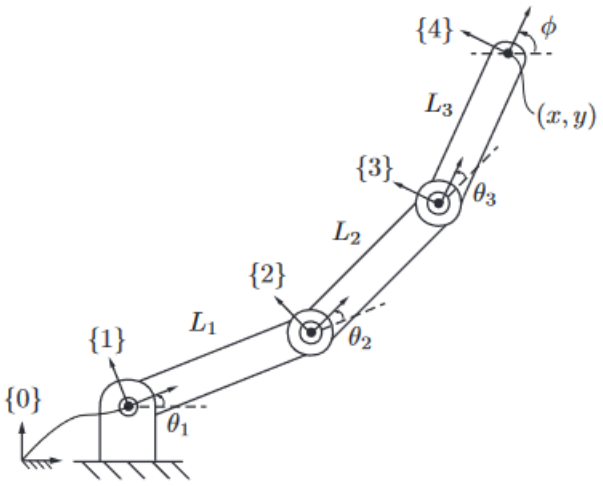


Figure 2.2.: 3R open-chain robot arm [10]

2.3.3. Product of Exponential Forward Kinematics

Using figure 2.2 another useful way to solve the forward kinematics is by the Product of Exponential Formula (PoE). Setting all joint angles to zero, the "zero" or "home" position, M can be defined as the position and orientation of frame {4}. In this example, M will be

$$M = \begin{bmatrix} 1 & 0 & 0 & L_1 + L_2 + L_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.9}$$

Each link is the considered as zero-pitch screw axis. Setting θ_1 and θ_2 to zero one can express the screw axis about joint 3 expressed in the {0} as

$$\mathcal{S}_3 = \begin{bmatrix} \omega_3 \\ v_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ -(L_1 + L_2) \\ 0 \end{bmatrix} \tag{2.10}$$

Confirming these results can be done by observations of figure 2.2. Imagine the arm is stretched out in the zero-configuration at a turntable rotating $\omega_3 = 1\,\text{rad/s}$ about joint 3's axis. At the origin of 0 the linear velocity v_3 is in the $\hat{y}_0$-direction at a rate of $L_1 + L_2$ units per second. This can be solved algebraically as $v_3 = -\omega_3 \times q_3$, where q_3 is any point on the axis of joint 3 expressed in {0}, in this case, $q_3 = (L_1 + L_2, 0, 0)^T$.

We can express the screw axis $\mathcal{S}_3$ in $se(3)$ matrix form as

$$[\mathcal{S}_3] = \begin{bmatrix} [\omega] & v \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & -(L_1 + L_2) \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.11}$$

For any θ_3, we can solve the matrix representation for screw motions as

$$T_{04} = e^{[\mathcal{S}_3]\theta_3} M \quad (for\ \theta_1 = \theta_2 = 0) \tag{2.12}$$

Provided $\mathcal{S}_1$ and $\mathcal{S}_2$ we then have the following equations

$$T_{04} = e^{[\mathcal{S}_2]\theta_2} e^{[\mathcal{S}_3]\theta_3} M \quad (for\ \theta_1 = 0) \tag{2.13}$$

and

$$T_{04} = e^{[\mathcal{S}_1]\theta_1} e^{[\mathcal{S}_2]\theta_2} e^{[\mathcal{S}_3]\theta_3} M \tag{2.14}$$

where in this example

$$[\mathcal{S}_2] = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & -L_1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.15}$$

$$[\mathcal{S}_1] = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.16}$$

In figure 2.3 an Illustration of PoE used on an n-link open-chain robot arm can be seen. It is also noteworthy to mention that in [10] θ is usually used to denote joint variables, as in this example. For most of this thesis, however, q will be used to denote the joint variables.

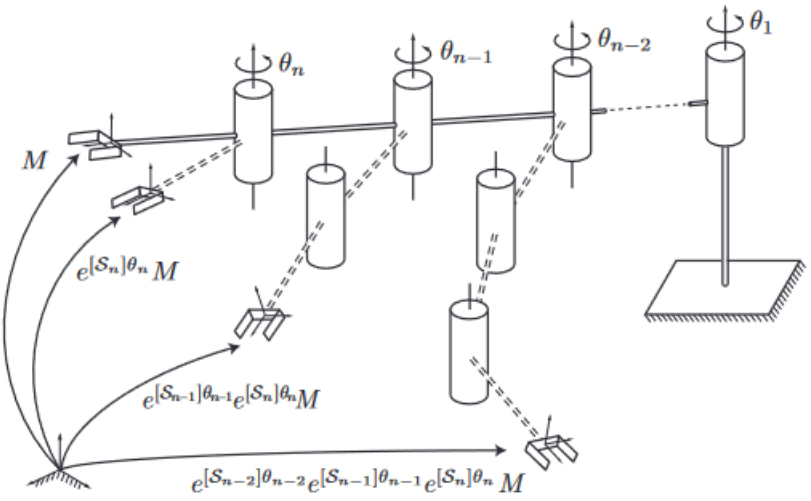


Figure 2.3.: PoE for an n-link spatial open-chain robot arm [10]

2.3.4. Velocity- and Acceleration Kinematics

Given a minimal set of coordinates $x \in \mathbb{R}^m$ and the velocity $\dot{x} = dx/dt \in \mathbb{R}^m$ the forward kinematics can be written as

$$x(t) = f(q(t)), \tag{2.17}$$

where $q \in \mathbb{R}^m$ is the joint variables. Using the chain rule, one can find the time derivative of this equation at time t as

$$\dot{x} = \frac{\partial f(q)}{\partial q}\frac{dq(t)}{dt} = \frac{\partial f(q)}{\partial q}\dot{q} = J(q)\dot{q} \tag{2.18}$$

where $J(q) \in \mathbb{R}^{m\times n}$ is the Jacobian. Here, the Jacobian represents the end-effector velocity $\dot{x}$ to the joint velocity $\dot{q}$ and is a function of the joint variables q.

To find the Jacobian of an open-chain robot one can consider the n-link open-chain forward kinematics in the product of exponential form

$$T(q_1, \ldots, q_n) = e^{[\mathcal{S}_1]q_1} e^{[\mathcal{S}_2]q_2} \ldots e^{[\mathcal{S}_n]q_n} M \tag{2.19}$$

With the spatial twist $\mathcal{V}_s$ given by $[\mathcal{V}_s] = \dot{T}T^{-1}$, where

$$\begin{aligned} \dot{T} &= \left(\frac{d}{dt}e^{[\mathcal{S}_1]q_1}\right) \ldots e^{[\mathcal{S}_n]q_n} M + e^{[\mathcal{S}_1]q_1}\left(\frac{d}{dt}e^{[\mathcal{S}_2]q_2}\right) \ldots e^{[\mathcal{S}_n]q_n} M \ldots \\ &= [\mathcal{S}_1]\dot{q}_1 e^{[\mathcal{S}_1]q_1} \ldots e^{[\mathcal{S}_n]q_n} M + e^{[\mathcal{S}_1]q_1}[\mathcal{S}_2]\dot{q}_2 e^{[\mathcal{S}_2]q_2} \ldots e^{[\mathcal{S}_n]q_n} M + \ldots \end{aligned} \tag{2.20}$$

and

$$T^{-1} = M^{-1}e^{-[\mathcal{S}_n]q_n} \dots e^{-[\mathcal{S}_1]q_1} \tag{2.21}$$

The product of $\dot{T}T^{-1}$ is then

$$[\mathcal{V}_s] = [\mathcal{S}_1]\dot{q}_1 + e^{[\mathcal{S}_1]q_1}[\mathcal{S}_2]e^{-[\mathcal{S}_1]q_1}\dot{q}_2 + e^{[\mathcal{S}_1]q_1}e^{[\mathcal{S}_2]q_2}[\mathcal{S}_3]e^{-[\mathcal{S}_2]q_2}e^{-[\mathcal{S}_1]q_1}\dot{q}_3 \dots \tag{2.22}$$

Which can also be expressed in the vector form by adjoint mapping as

$$\mathcal{V}_s = \underbrace{\mathcal{S}_1}_{J_{s1}}\dot{q}_1 + \underbrace{Ad_{e^{[\mathcal{S}_1]q_1}}(\mathcal{S}_2)}_{J_{s2}}\dot{q}_2 + \underbrace{Ad_{e^{[\mathcal{S}_1]q_1}e^{[\mathcal{S}_2]q_2}}(\mathcal{S}_3)}_{J_{s3}}\dot{q}_3 + \dots \tag{2.23}$$

It should be observed that $\mathcal{V}_s$ is a sum of n spatial twists of the form

$$\mathcal{V}_s = J_{s1}\dot{q}_1 + J_{s2}\dot{q}_2 + \dots + J_{sn}(q)\dot{q}_n \tag{2.24}$$

In matrix form

$$\mathcal{V}_s = \begin{bmatrix} J_{s1} & J_{s2} & \dots & J_{sn} \end{bmatrix} \begin{bmatrix} \dot{q}_1 \\ \vdots \\ \dot{q}_n \end{bmatrix} = J_s(q)\dot{q} \tag{2.25}$$

It should be noted that this is the space Jacobian i.e. the Jacobian in space frame coordinates.

Further, to find the acceleration of the end effector, taking the time derivative of equation 2.18 again will provide us with the equation

$$\ddot{x} = J(q)\ddot{q} + \dot{J}(q,\dot{q})\dot{q} \tag{2.26}$$

which can be used to find the acceleration of the end-effector provided the joint position, -velocity and -acceleration.

2.3.5. Inverse kinematics

The inverse kinematics of a robot arm gives us the joint coordinates from the end effector orientation and position. The inverse kinematics can be solved both analytically and numerically. One of the main issues with inverse kinematics is that a given end effector position can often be achieved with several different combinations of the joint variables. As can be seen in figure 2.4 even a simple open-chain robot as the 2R planar open-chain has two different solutions for a single end-effector position. The “lefty” and the “righty” solutions. Using the following forward kinematics ignoring the orientation of the robot arm

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} L_1 \cos\theta_1 + L_2 \cos(\theta_1 \ + \theta_2) \\ L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2) \end{bmatrix} \tag{2.27}$$

where we assume $L_1 > L_2$. An explicit solution $(\theta_1\theta_2)$ for given (x, y) can be found. Using figure 2.4 the angle β is restricted to the interval $[0, \pi]$ and can be found using the law of cosines

$$L_1^2 + L_2^2 - 2L_1L_2\cos\beta = x^2 + y^2 \tag{2.28}$$

solving fore β gives us

$$\beta = \cos^{-1}\left(\frac{L_1^2 + L_2^2 - x^2 - y^2}{2L_1L_2}\right). \tag{2.29}$$

Similarly for α

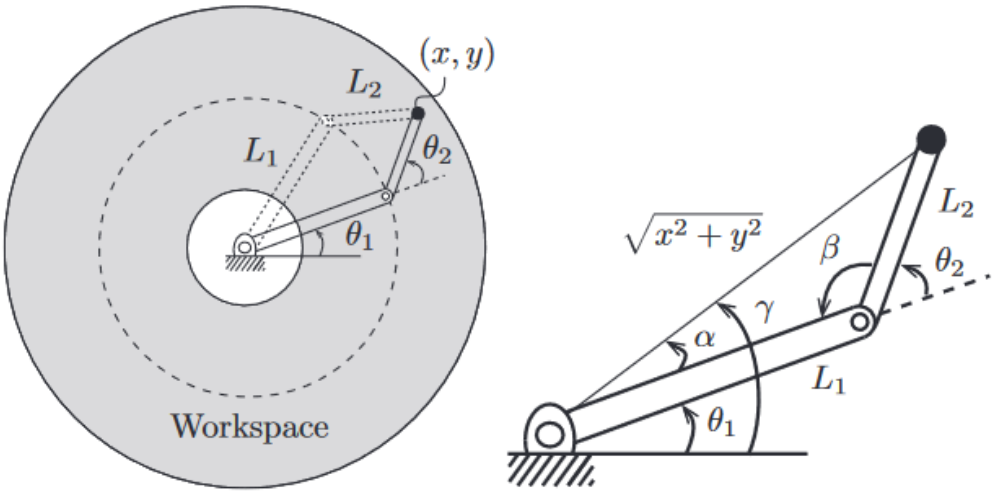
$$\alpha = \cos^{-1}\left(\frac{x^2 + y^2 + L_1^2 - L_2^2}{2L_1\sqrt{x^2 + y^2}}\right). \tag{2.30}$$

Defining an angle γ as $\gamma = atan2(y, x)$ the righty position an be found as follows

$$\theta_1 = \gamma - \alpha, \quad \theta_2 = \pi - \beta. \tag{2.31}$$

And for the lefty solution we have

$$\theta_1 = \gamma + \alpha, \quad \theta_2 = \beta - \pi. \tag{2.32}$$



(a) A workspace, and lefty and righty configurations.

(b) Geometric solution.

Figure 2.4.: 2R planar open chain inverse kinematics [10]

In this thesis, the inverse kinematics is solved using iterative numerical methods. These are used for situations where analytical solutions are difficult or impossible or just to

improve the accuracy of the solution. The numerical algorithm used is the Levenberg-Marquardt algorithm. The general algorithm is described below with information taken from " The Levenberg-Marquardt Algorithm: Implementation and Theory" by Moré [12].

Firstly we have $F : \mathbb{R}^n \to \mathbb{R}^m$ which is continuously differentiable. Moré then asks to consider the "nonlinear least squares problem of finding a local minimizer of"

$$\phi(x) = \frac{1}{2}\sum_{i=1}^{m} f_i^2(x) = \frac{1}{2}||F(x)||^2. \tag{2.33}$$

Chapter 2 of [12] describes the derivation of the Levenberg-Marquardt algorithm which results in the final cited algorithm below:

(a) Given $\Delta_k > 0$, find $\lambda_k \geq 0$ such that if

$$(J_k^T J_k + \lambda_k D_k^T D_k)P_k = -J_k^T f_k \tag{2.34}$$

Then either $\lambda_k = 0$ and $||D_k P_k|| \leq \Delta_k$ or $\lambda_k > 0$ and $||D_k P_k|| = \Delta_k$

(b) If $||F(x_k + p_k))|| < ||F(x_k)||$ set $x_{k+1} = x_k + p_k$ and evaluate J_{k+1}; otherwise set $x_{k+1} = x_k$ and $J_{k+1} = J_k$

(c) Choose Δ_{k+1} and D_{k+1}

2.3.6. Velocity and Acceleration Inverse Kinematics

For solving the inverse kinematics for finding the joint velocity we can solve equation 2.18 with respect to $\dot{q}$ as follows

$$\dot{q} = J(q)^{-1}\dot{x}. \tag{2.35}$$

And for the joint acceleration, the same can be done by solving equation 2.26 with respect to $\ddot{q}$

$$\ddot{q} = J(q)^{-1}(\ddot{x} - \dot{J}(q, \dot{q})\dot{q}). \tag{2.36}$$

Seeing how the Jacobian is only invertible if the robot has six joints, creating a 6x6 Jacobian, one might have to alter the Jacobian for this approach to work. One solution is to remove the variables you do not need. For example if one has an open-chain robot arm with three joint variables $q = [q_1 \quad q_2 \quad q_3]^T \in \mathbb{R}^3$ one can remove the rows of the Jacobian corresponding to angular velocities, which using the space Jacobian formulation form section 2.3.4 are the three first rows. Then only use the last three rows for solving the linear velocities as described below.

$$J(q) = \begin{bmatrix} J_{11} & J_{12} & J_{13} \\ J_{21} & J_{22} & J_{23} \\ J_{31} & J_{32} & J_{33} \\ J_{41} & J_{42} & J_{43} \\ J_{51} & J_{52} & J_{53} \\ J_{61} & J_{62} & J_{63} \end{bmatrix} \Rightarrow \begin{bmatrix} J_{41} & J_{42} & J_{43} \\ J_{51} & J_{52} & J_{53} \\ J_{61} & J_{62} & J_{63} \end{bmatrix} = J_{3\times3}(q) \tag{2.37}$$

Then the Jacobian is a 3×3 matrix and therefore invertible. There are also other methods of achieving this, such as the Moore-Penrose pseudoinverse.

2.4. Crane Dynamics

The dynamic model of the crane will, as the kinematic model, be a model of the motion of the crane. However, with the dynamic model, the forces and torques that cause the motions will be taken into account. The dynamic model for a crane is a set of second-order differential equations of the form

$$\tau = M(q)\dot{q} + C(q,\dot{q})\dot{q} + G(q) \tag{2.38}$$

where $q \in \mathbb{R}^n$ is a vector of joint variables, $\tau \in \mathbb{R}^n$ is a vector of joint forces and torques. The matrix $M(q) \in \mathbb{R}^{n\times n}$ is a symmetric positive definite mass matrix, $C(q,\dot{q})\dot{q} \in \mathbb{R}^n$ is the centripetal and Coriolis matrix and $G(q) \in \mathbb{R}^n$ is the gravity matrix.

2.4.1. Lagrange Formulation

A common way to solve dynamics is by Lagrange mechanics. In the Lagrangian formulation a set of independent coordinates $q \in \mathbb{R}^n$ are chosen to describe the configuration of the system. These are called generalised coordinates. The generalized forces $f \in \mathbb{R}^n$ can then be chosen. The forces are dual to the coordinate rates $\dot{q}$, and the product $f^T\dot{q}$ relates to power. The Lagrangian function $\mathcal{L}(q,\dot{q})$ is defines as

$$\mathcal{L}(q,\dot{q}) = \mathcal{K}(q,\dot{q}) - \mathcal{P}(q) \tag{2.39}$$

where $K(q,\dot{q})$ is the systems kinetic energy and $\mathcal{P}(q)$ is the systems potential energy. With the Lagrangian function, it is possible to find the systems equations of motions by the equation

$$f = \frac{d}{dt}\frac{\partial\mathcal{L}}{\partial\dot{q}} - \frac{\partial\mathcal{L}}{\partial q} \tag{2.40}$$

For an open-chain robot, the kinetic energy can be expressed as

$$\mathcal{K}(q,\dot{q}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} m_{ij}(q)\dot{q}_i\dot{q}_j = \frac{1}{2}\dot{q}^T M(q)\dot{q} \tag{2.41}$$

where m_{ij} is the (i,j)th element of the $n \times n$ mass matrix. The dynamics equations can be found by evaluating the right side of

$$\tau_i = \frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}}{\partial q_i}, \quad i = 1, ..., n \tag{2.42}$$

With the kinetic energy expressed as in equation 2.41, the dynamics can be written explicitly

$$\tau_i = \sum_{j=1}^{n} m_{ij}(q)\ddot{q}_j + \sum_{j=1}^{n}\sum_{k=1}^{n} \Gamma_{ijk}(q)\dot{q}_j\dot{q}_k + \frac{\partial \mathcal{P}}{\partial q_i}, \quad i = 1, ..., n \tag{2.43}$$

Γ_{ijk} is known as the Christoffel symbol of the first kind and is defined as

$$\Gamma_{ijk}(q) = \frac{1}{2}\left(\frac{\partial m_{ij}}{\partial q_k} + \frac{\partial m_{ik}}{\partial q_j} + \frac{\partial m_{jk}}{\partial q_i}\right) \tag{2.44}$$

Getting these dynamic equations as on the form seen in equation 2.38. G(q) is simply equal to $\frac{\partial \mathcal{P}}{\partial q}$. In this form $C(q, \dot{q}) \in \mathbb{R}^n$ is defined as

$$c_{ij}(q, \dot{q}) = \sum_{k=1}^{n} \Gamma_{ijk}(q)\dot{q}_k \tag{2.45}$$

where c_{ij} is the $(i, j)th$ element of $C(q, \dot{q})$.

With the model in equation 2.38 it is possible to solve the systems cranes forward and inverse dynamics, where $\ddot{q}$ is solved using $(q, \dot{q})$ and the joint forces and torques gives the forward dynamics

$$\ddot{q} = M^{-1}(q)(\tau - C(q, \dot{q})\dot{q} - G(q)) \tag{2.46}$$

and the inverse dynamics is solved using equation 2.38.

2.4.2. Inertia Matrix

To solve the dynamics an inertia matrix is needed, the spatial inertia matrix has the form

$$\mathcal{G}_b = \begin{bmatrix} \mathcal{I}_b & 0 \\ 0 & mI \end{bmatrix} \tag{2.47}$$

where I is the 3×3 identity matrix, m is the mass of the joint, and $\mathcal{I}_b$ is the inertia tensor. The easiest way to find the inertia tensor is to assume a basic shape of the joint and calculate its moment of inertia. In this thesis, all joints of the crane are assumed to be circular cylinders with uniform density. The only difference used in the calculation of the moment of inertia is where the centre of mass is.

This gives us two ways to calculate the moment of inertia. The first with the centre of mass is in the middle of the cylinder.

$$\mathcal{I}_b = \begin{bmatrix} \frac{1}{12}(3r^2 + h^2) & 0 & 0 \\ 0 & \frac{1}{12}(3r^2 + h^2) & 0 \\ 0 & 0 & \frac{1}{2}mr^2 \end{bmatrix} \tag{2.48}$$

And the second with the centre of mass is at the end of the cylinder.

$$\mathcal{I}_b = \begin{bmatrix} \frac{1}{3}(3r^2 + h^2) & 0 & 0 \\ 0 & \frac{1}{2}(3r^2 + h^2) & 0 \\ 0 & 0 & \frac{1}{3}(3r^2 + h^2) \end{bmatrix} \tag{2.49}$$

where m is the same mass as used in equation 2.47, h is the height of the joint and r is the radius of the joint.

2.4.3. Dynamic Equations in Closed Form

The dynamics in the model are solved using a recursive inverse dynamic algorithm and can be used to solve a closed-form set of dynamic equations

$$\tau = M(q)\ddot{q} + C(q, \dot{q}) + G(q) \tag{2.50}$$

Firstly this method requires proof that the kinetic energy can be written in the manner of equation 2.41. This is done by noting that $\mathcal{K}$ can be expressed as the sum of the kinetic energy of each link

$$\mathcal{K} = \frac{1}{2}\sum_{i=1}^{n} \mathcal{V}^T \mathcal{G}_i \mathcal{V}_i \tag{2.51}$$

where $\mathcal{V}_i$ is the twist from frame $\{i\}$ and $\mathcal{G}_i$ is the spatial inertia matrix of link i as defined by equation 2.47.

$$\mathcal{V}_i = J_{ib}(q)\dot{q}, \quad i = 1, ..., n \tag{2.52}$$

The kinetic energy with this be written as

$$\mathcal{K} = \frac{1}{2}\dot{q}^T \left(\sum_{i=1}^{n} J_{ib}^T(q) \mathcal{G}_i J_{ib}(q) \right) \tag{2.53}$$

where the terms in the parenthesis are the mass matrix $M(q)$:

$$M(q) = \sum_{i=1}^{n} J_{ib}^T(q) \mathcal{G}_i J_{ib}(q) \tag{2.54}$$

To derive the closed-form set of dynamic equations, we first start by defining stacked vectors

$$\mathcal{V} = \begin{bmatrix} \mathcal{V}_1 \\ \vdots \\ \mathcal{V}_n \end{bmatrix} \tag{2.55}$$

$$\mathcal{F} = \begin{bmatrix} \mathcal{F}_1 \\ \vdots \\ \mathcal{F}_n \end{bmatrix} \tag{2.56}$$

The following matrices are defined

$$\mathcal{A} = \begin{bmatrix} \mathcal{A}_1 & 0 & \dots & 0 \\ 0 & \mathcal{A}_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \dots & \mathcal{A}_n \end{bmatrix} \in \mathbb{R}^{6n \times n} \tag{2.57}$$

$$\mathcal{G} = \begin{bmatrix} \mathcal{G}_1 & 0 & \dots & 0 \\ 0 & \mathcal{G}_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \dots & \mathcal{G}_n \end{bmatrix} \in \mathbb{R}^{6n \times 6n} \tag{2.58}$$

$$[ad_{\mathcal{V}}] = \begin{bmatrix} [ad_{\mathcal{V}_1}] & 0 & \dots & 0 \\ 0 & [ad_{\mathcal{V}_2}] & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \dots & [ad_{\mathcal{V}_n}] \end{bmatrix} \in \mathbb{R}^{6n \times 6n} \tag{2.59}$$

$$[ad_{\mathcal{A}\dot{\theta}}] = \begin{bmatrix} [ad_{\mathcal{A}_1\dot{q}_1}] & 0 & \dots & 0 \\ 0 & [ad_{\mathcal{A}_2\dot{q}_2}] & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & \dots & [ad_{\mathcal{A}_n\dot{q}_n}] \end{bmatrix} \in \mathbb{R}^{6n \times 6n} \tag{2.60}$$

$$\mathcal{W}(q) = \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ [Ad_{T_{21}}] & 0 & \dots & 0 & 0 \\ 0 & [Ad_{T_{32}}] & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & [Ad_{T_{n,n-1}}] & 0 \end{bmatrix} \in \mathbb{R}^{6n \times 6n} \tag{2.61}$$

Finally, the following stacked vectors are defined

$$\mathcal{V}_{base} = \begin{bmatrix} Ad_{T_{10}}(\mathcal{V}_0) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^{6n} \tag{2.62}$$

$$\dot{\mathcal{V}}_{base} = \begin{bmatrix} Ad_{T_{10}}(\dot{\mathcal{V}}_0) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^{6n} \tag{2.63}$$

$$\mathcal{F}_{tip} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ Ad^T_{T_{n+1,n}}(\mathcal{F}_{n+1}) \end{bmatrix} \in \mathbb{R}^{6n} \tag{2.64}$$

The following inverse dynamics recursive algorithm can then be assembled

$$\mathcal{V} = \mathcal{W}(q)\mathcal{V} + \mathcal{A}\dot{q} + \mathcal{V}_{base}, \tag{2.65}$$

$$\dot{\mathcal{V}} = \mathcal{W}(q)\dot{\mathcal{V}}\mathcal{A}\ddot{q} - [ad_{\mathcal{A}\dot{q}}](\mathcal{W}(q)\mathcal{V} + \mathcal{V}_{base}) + \dot{\mathcal{V}}_{base}, \tag{2.66}$$

$$\mathcal{F} = \mathcal{W}^T(q)\mathcal{F} + \mathcal{G}\dot{\mathcal{V}} - [ad_{\mathcal{V}}]^T\mathcal{G}\mathcal{V} + \mathcal{F}_{tip}, \tag{2.67}$$

$$\tau = \mathcal{A}^T\mathcal{F}. \tag{2.68}$$

We can then define $\mathcal{L}(q) = (I - \mathcal{W}(q))^{-1}$. It can be verified by direct calculations that

$$\mathcal{L}(q) = \begin{bmatrix} I & 0 & 0 & \dots & 0 \\ [Ad_{T_{21}}] & I & 0 & \dots & 0 \\ [Ad_{T_{31}}] & [Ad_{T_{32}}] & I & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ [Ad_{T_{n1}}] & [Ad_{T_{n2}}] & [Ad_{T_{n3}}] & \dots & I \end{bmatrix} \in \mathbb{R}^{6n\times 6n} \tag{2.69}$$

Equation 2.65 to 2.68 can now be rewritten

$$\mathcal{V} = \mathcal{L}(q)(\mathcal{A}\dot{q} + \mathcal{V}_{base}), \tag{2.70}$$

$$\dot{\mathcal{V}} = \mathcal{L}(q)(\mathcal{A}\ddot{q} + [ad_{\mathcal{A}\dot{q}}]\mathcal{W}(q)\mathcal{V} + [ad_{\mathcal{A}\dot{q}}]\mathcal{V}_{base} + \dot{\mathcal{V}}_{base}, \tag{2.71}$$

$$\mathcal{F} = \mathcal{L}^T(q)(\mathcal{G}\dot{\mathcal{V}} - [ad_{\mathcal{V}}]^T\mathcal{G}\mathcal{V} + \mathcal{F}_{tip}), \tag{2.72}$$

$$\tau = \mathcal{A}^T\mathcal{F}. \tag{2.73}$$

From these equations, the mass-, Coriolis- and centripetal-, and gravity matrix can be expressed as

$$M(q) = \mathcal{A}^T\mathcal{L}^T(q)\mathcal{G}\mathcal{L}(q)\mathcal{A}, \tag{2.74}$$

$$c(q,\dot{q}) = -\mathcal{A}^T\mathcal{L}^T(q)(\mathcal{G}\mathcal{L}(q)[ad_{\mathcal{A}\dot{q}}]\mathcal{W}(q) + [ad_{\mathcal{V}}]^T\mathcal{G})\mathcal{L}(q)\mathcal{A}\dot{q}, \tag{2.75}$$

$$G(q) = \mathcal{A}^T\mathcal{L}^T(q)\mathcal{G}\mathcal{L}(q)\dot{\mathcal{V}}_{base}. \tag{2.76}$$

This is not the most computationally effective way of simulating the dynamic. It is, however, a straightforward and relatively easy method to code.

2.5. Control theory

In this section, the control theory used in the thesis is presented.

2.5.1. Feedforward Control

Feedforward control is a type of control that uses measurements of the most significant disturbance variables to achieve control. The concept is to take corrective measures before the disturbances upset the system. These corrective measures will, however, not occur until after the process has generated a non-zero error signal. Feedforward gets no feedback from the actual system making corrective measures needed for other causes than the disturbance model provided can render the controller incapable of correcting the errors. Combining feedforward with other types of control, often feedback control, can help feedforward control to compensate for modelling errors and unmeasured disturbances[13]. In figure 2.5 a fundamental block diagram of a typical setup of feedforward control is presented. Here D is the disturbance variable, U is the system input, Y is the system output and Y_{sp} is the set-point.

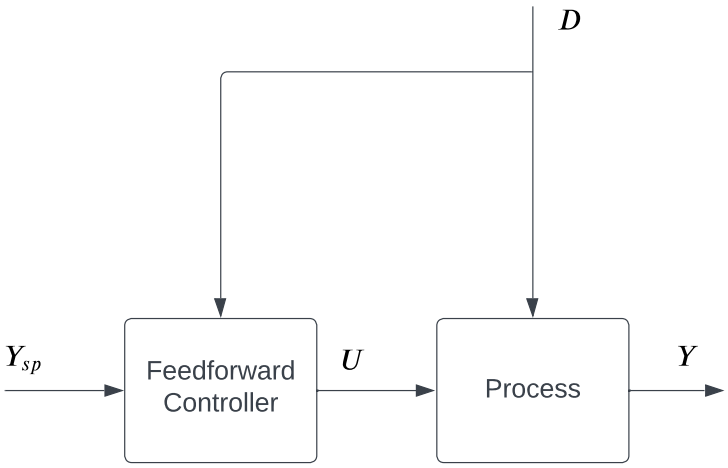


Figure 2.5.: Feedforward block diagram

2.5.2. Feedback Control

This section is based on the book "Multivariable feedback control" by Skogestad and Postelwaith [14].

In feedback control, a set-point value describes the value where you want the system to be. A measurement of the actual state value is subtracted from the set point giving us the system error

$$e = y_m - r \tag{2.77}$$

where y_m is the measurement $y_m = y + n$ where n is measurement noise, and r is the set-point. This is called a negative feedback loop, as the state that is used in the feedback loop is subtracted from the set point. Using the system error it can be sent to a controller K whose task is to drive the error to zero. When the system error is zero it means that there is no error between the desired state value and the measured state value. The controller gives us the model input as follows

$$u = K(s)(r - y - n). \tag{2.78}$$

Figure 2.6 shows a block diagram of a typical negative feedback loop, here G is the system model and G_d is the disturbance model.

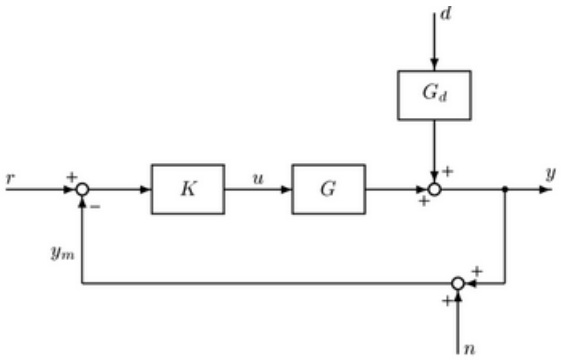


Figure 2.6.: Block diagram of a typical negative feedback loop [14].

It is often preferred to use feedback over feedforward. One of feedback control's major advantages is that it uses the system error always letting the controller know how far off from the desired setpoint the system is. This makes controlling systems with unpredictable disturbances easier, as it quickly notices changes in the system. Feedforward needs to correct for disturbances before they happen, as it only uses the desired set point and a model for the disturbance to create the controller input. This requires a very good model of the disturbance, which can be difficult if the disturbance is of an unpredictable nature like wind and wave disturbances. Making it mostly used in processes where good disturbance models can be designed. There are also problems such as model uncertainty and instability which feedback control handles better because of its use of the system error.

2.5.3. PID controller

One of the most common and versatile controllers used for feedback control is the PID controller. The controller uses three adjustable gains: the proportional gain (P), the integral gain (I) and the derivative gain (D). Depending on the system's need one can exclude or include any one of these gains, it is for example common to omit the derivative gain in process control because process plants usually are stable with an over-damped response [14]. A common way to formulate the PID controller is as follows

$$u(t) = k_p e(t) + k_i \int e(t)dt + k_p \frac{de}{dt}. \tag{2.79}$$

All the different gains have different contributions to the controller. Proportional to the error we have the proportional gain, which is the gain most seldom excluded in the controller. When a steady-state offset from the reference is present the integral gain is commonly applied. Integral gain is a good way to minimise the offset without the use of a very large proportional gain. To introduce an element of prediction into the controller the derivative gain is employed. The derivative gain uses the rate of change of the error. The derivative gain can amplify the noise, which is a common problem when differentiating noisy signals [15].

2.5.4. Cascade Control

Cascade control is a form of control where two nested feedback loops with each its own controller are used to make single system input. The first, or primary controller loop sets the set-point for the secondary controller which is used to improve upon the first setpoint. This gives the control two sensors and two controlled variables with a single manipulated variable.

The benefit of using cascade control is that a disturbance affecting the secondary controller has been corrected by this controller before it can affect the value of the primary controller. Generally, the second controller is tuned to be faster than the primary, meaning the process that needs the fastest control is in the inner loop [13]. An example of a typical cascade control structure can be seen in Chapter 3 in figure 3.1.

2.5.5. Lyapunov Stability

Lyapunov is commonly used for the analysis of stability for equilibrium points. Lyapunov is often used when ascertining the stability of nonlinear systems [14]. The following two theorems are taken from "Nonlinear Systems" by Khalil [16]. The first theorem is defined in Theorem 4.1 in [16].

Theorem 1 If we have the equilibrium point x for the system

$$\dot{x} = f(x) \tag{2.80}$$

and a domain $D \subset \mathbb{R}^n$ containing $x = 0$. Then we let $V : D \to R$ be continuously

differentiable such that

$$V(0) \quad \text{and} \quad v(X) > 0 \ \in D - \{0\} \tag{2.81}$$

$$\dot{V}(x) \leq 0 \in D \tag{2.82}$$

Then $x = 0$ will be stable. If

$$\dot{x}(x) < 0 \in D - \{0\} \tag{2.83}$$

then $x = 0$ will be asymptotically stable.

If a function $V(x)$ is a differentiable function satisfying equation 2.81 and 2.82 it is called a Lyapunov function. Another theorem given as Theorem 4.10 in [16] gives the conditions for exponential stability in terms of Lyapunov as

Theorem 2 We set x as an equilibrium point for

$$\dot{x} = f(t, x) \tag{2.84}$$

and $D \subset \mathbb{R}^n$ is a domain containing $x = 0$. We then have the continuously differentiable function $V : [0, \infty) \times D \to R$ such that

$$k_1||x||^a \leq V(t, x) \leq k_2||x||^a \tag{2.85}$$

$$\frac{\partial V}{\partial t} + \frac{\partial V}{\partial x} f(t, x) \leq -k_3||x||^a \tag{2.86}$$

Which holds $\forall t \geq 0$ and $\forall x \in D$. Here k_1, k_2, k_3 and a are positive constants. If this holds $x = 0$ is exponentially stable, and if the assumptions holds globally it is globally exponentially stable.

2.5.6. Nonlinear Model Predictive Control

Subsection 2.5.6 and 2.5.7 are based on "An Introduction to Nonlinear Model Predictive Control" by Findeisen and Allgöwer [17].

The basic principles of the model predictive control problem can be seen in figure 2.7. At the time t a measurement is obtained, this is used by the controller to predict the future dynamic behaviour of the system over a prediction horizon T_p. This in turn will determine, over a control horizon $T_c \leq T_p$, the input such that a predetermined open-loop performance objective function is optimised.

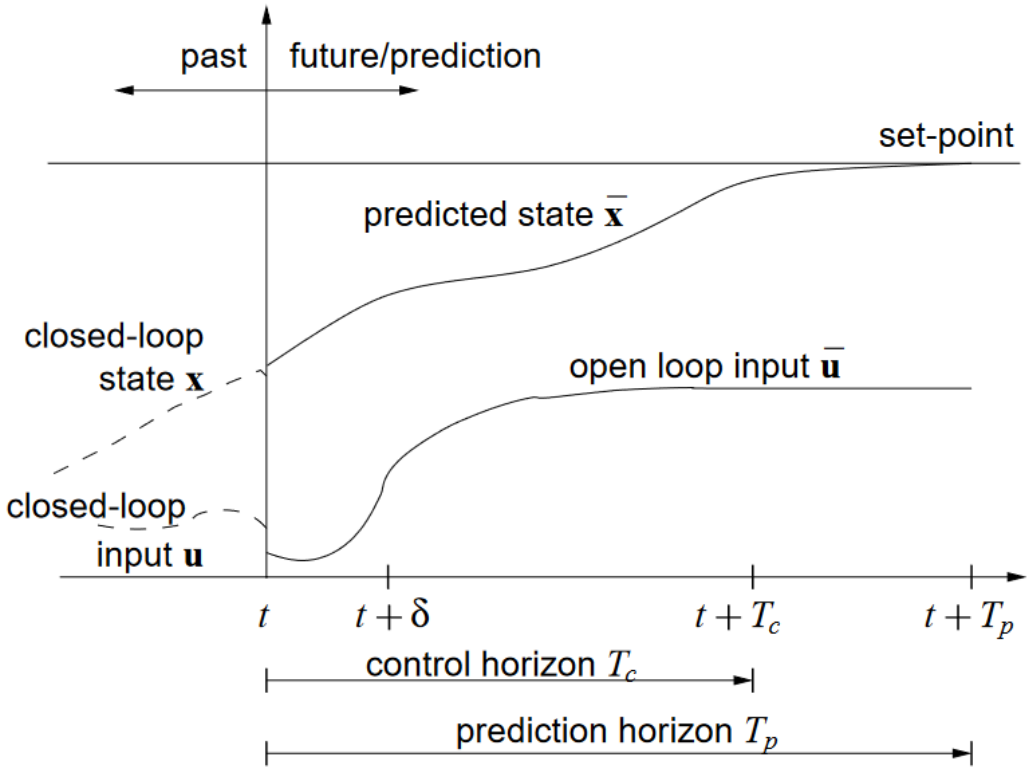


Figure 2.7.: The basic principle of model predictive control [17]

2.5.7. Mathematical Formulation of NMPC

Firstly, in the following equatins $||\cdot||$ denotes Euclidean vector norm in $\mathbb{R}^n$. The use of the semicolon ";" in a function argument indicates that the following symbols are parameters. For example, $f(x;\gamma)$ will mean that we have the function value at x with the parameter γ. For clarity, the internal variables are denoted with bars, for example, $\bar{x}$. This is to distinguish between the real system and the system used to predict.

To explain the mathematical formulation of the NMPC we start with a set of nonlinear differential equations

$$\dot{x}(t) = f(x(t), u(t)), \quad x(0) = x_0 \tag{2.87}$$

subject to input and state constraints

$$u(t) \in \mathcal{U}, \forall t \geq 0 \quad x(t) \in \mathcal{X} \times \mathcal{U}, \forall t \geq 0 \tag{2.88}$$

where the vector of states are given by $x(t) \in \mathcal{X} \subseteq \mathbb{R}^n$ and vector of inputs are denoted as $u(t) \in \mathcal{U} \subseteq \mathbb{R}^m$. Here a set of feasible states are denoted $\mathcal{X}$ and a set of feasible input values $\mathcal{U}$. It is assumed $\mathcal{X}$ and $\mathcal{U}$ follow the following assumptions.

Assumption 1 It is assumed $\mathcal{U} \subset \mathbb{R}^p$ is compact, $\mathcal{U} \subseteq \mathbb{R}^n$ is connected and $(0,0) \in \mathcal{X} \times \mathcal{U}$

A box constraint denotes $\mathcal{U}$ and $\mathcal{X}$ on its simplest form

$$\begin{aligned} \mathcal{U} &:= \{u \in \mathbb{R}^m | u_{min} \leq u \leq u_{max}\}, \\ \mathcal{X} &:= \{x \in \mathbb{R}^n | x_{min} \leq x \leq x_{max}\}. \end{aligned} \tag{2.89}$$

Where u_{min}, u_{max}, x_{min} and x_{max} are constant vectors.

Assumption 2 Vector field $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is continous satisfies $f(0,0) = 0$ and is Lipschitz continuous in x.

Assumption 3 The equation 2.87 in the region of interest, has unique continuous solutions for any initial conditions. This also applies to any piecewise continuous and right continuous input function $u(\cdot) : [0, T_p] \to \mathcal{U}$.

A typical formulation of the finite horizon open-loop optimal control problem is as follows:

Find

$$\min_{\bar{u}(\cdot)} J(x(t), \bar{u}(\cdot); T_c, T_p) \tag{2.90}$$

with

$$J(x(t), \bar{u}(\cdot); T_p, T_c) := \int_t^{t+T_p} F(\bar{x}(\tau), \bar{u}(\tau)))d\tau \tag{2.91}$$

subjected to

$$\bar{x}(\tau) = f(\bar{x}(\tau), \bar{u}(\tau)), \quad \bar{x}(t) = x(t) \tag{2.92}$$

$$\bar{u}(t) \in \mathcal{U}, \quad \forall \tau \in [t, t + T_c] \tag{2.93}$$

$$\bar{u}(\tau) = \tau + \bar{T}_c, \quad \forall \tau \in [t + T_c, t + T_p] \tag{2.94}$$

$$\bar{\tau} \in \mathcal{X}, \quad \forall \tau \in [t, t + Tp]. \tag{2.95}$$

The solution of equation 2.92 is driven by the input in equations 2.93 and 2.94 with initial conditions $x(t)$.

Below the stage cost is defined as function F, here the desired control performance is specified. The stage cost below is in the standard quadratic form as it is commonly used

$$F(x, u) = (x - x_s)^T \mathcal{Q}(x - x_S) + (u - u_s)^T + \mathcal{R}(u - u_s). \tag{2.96}$$

Here $\mathcal{Q}$ and $\mathcal{R}$ are the positive definite symmetric weighting matrices, and x_s and u_s are the given set-points.

If one assumes the existence of an optimal solution it is denoted as $\bar{u}^*(\cdot; x(t), T_p, T_c : [t, t + T_p] \to \mathcal{U}$. At the sampling interval $t = j\delta$, $j = 0, 1, \ldots$, the open-loop optimal control problem is solved repeatedly. The closed-loop control at the sampling instant is defined as

$$u^*(\tau) := \bar{u}^*(\tau; x(t), T_p, T_c), \tau \in [t, \delta]. \tag{2.97}$$

Finding the optimal value of the NMPC open-loop optimal control as a function of states can be done as follows

$$V(x; T_p, T_c) = J(x, \bar{u}^*(\cdot; x(t)); T_p, T_c). \tag{2.98}$$

This function serves as a Lyapunov function candidate and is important in proving the stability of various NMPC schemes.

Chapter 3.

Material and Method

In this chapter, the work done to make the different models are presented. Three models are presented each with a unique controller and two different crane models are used.

3.1. Matlab and Simulink

All simulations are done using Matlab® and Simulink®. Block diagrams were made in Simulink, with functions and scripts made with Matlab.

All the code and models created or discussed in this chapter are provided in the thesis' digital attachment. Some of the necessary toolboxes needed to run the models and code are given below. It should be noted that this may not include all the toolboxes required but merely the most important ones.

- Robotics System Toolbox
- Model Predictive Control Toolbox
- Simscape®

The last toolbox needed is not provided by MathWorks® but by the creators of the book "Modern Robotics: Mechanics, Planning and Control" [10]. Their software for Matlab can be downloaded from Github through the following URL `https://github.com/NxRLab/ModernRobotics`. This toolbox is needed for the crane dynamic and kinematic models for models 2 and 3.

All models and codes introduced in this chapter and used for Chapter 4 can be found in the digital attachment. In Appendix A all the contents in this attachment is described.

3.2. Control Scheme

The general scheme that all of the models try to follow can be seen in the block diagram in figure 3.1. The pendulum oscillation is controlled by an inner loop where the angle rate of the pendulum $\dot{\phi}_x$ and $\dot{\phi}_y$ is used as the feedback variable. The input of the pendulum damping controller is created by an outer loop controlling the crane position, here the feedback variables are the crane-tip position and -velocity provided by the forward

kinematics. The desired position, velocity and acceleration are then sent to the inverse kinematics block where the desired joint angles, -velocity and -acceleration are found. These then go to the crane controller which sends the correct torque to the crane dynamics model. The joint variables from the dynamic model are then sent to the forward kinematics block which gives the pendulum model the crane-tip accelerations $\ddot{x}_0$, $\ddot{y}_0$ and $\ddot{z}_0$ as input. As can be seen in the figure there is an inner and an outer loop which indicates cascade control, which is a control method where two feedback loops are combined into a single output. Where the output of the first controller (crane-tip position) adjusts the set-point of the second controller (crane load damping) [13]. It should be noted that this scheme is merely the first draft made and that the final models may differ to various degrees.

The basic principle of this control scheme is that the crane-load oscillations are dampened using movements in the crane tip, while it also places the crane tip at the correct x- and y- positions. For all the models perfect measurements are assumed, and all movements in the z-axis are also set to be constant with no change.

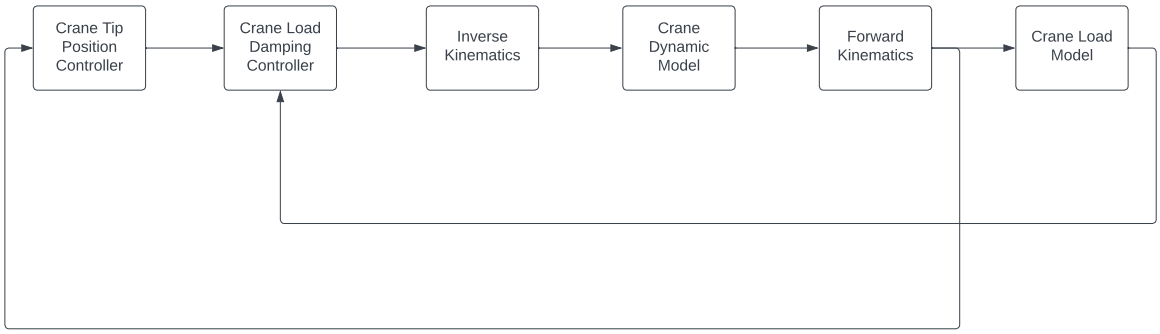


Figure 3.1.: General control scheme

3.3. Model 1

The first model uses the controller deduced in the preceding project report, the same applies to the pendulum model used to simulate the crane load [5]. The pendulum model and controller are based on the work note "Crane-Load Dynamics and Control" by Olav Egeland [8].

3.3.1. Pendulum Model

The spherical pendulum model is taken from Chapter 3.2 in Egeland's work note. This chapter describes the deduction of a 2D spherical pendulum model with a moving attachment point using Lagrange mechanics.

To make the model the the position of the mass is first defined as

$$\vec{r}_r = (x_0 - \sin\phi_y L)\vec{n}_1 + (y_0 + \sin\phi_x \cos\phi_y L)\vec{n}_2 + (z_0 - \cos\phi_x \cos\phi_y L)\vec{n}_3. \tag{3.1}$$

Here x_0, y_0 and z_0 are the position of the attachment point and ϕ_x and ϕ_y are the pendulum angles. It should be noted that z_0 is the vertical position.

Derivation of this position with respect to time gives us the velocity of the mass. The position and velocity can be used to find the kinetic and potential energy of the pendulum. The kinetic energy is

$$\begin{aligned}\mathcal{K} =& \frac{1}{2}mL^2\left(\dot{\phi}_x^2\cos^2\phi_y + \dot{\phi}_y^2\right) + \frac{1}{2}m(\dot{x}_0^2 + \dot{y}_0^2 + \dot{z}_0^2) - mL\dot{x}_0\dot{\phi}_y\cos\phi_y \\ &+ mL\dot{y}_0\left(\dot{\phi}_x\cos\phi_x\cos\phi_y - \dot{\phi}_y\sin\phi_x\sin\phi_y\right) \\ &+ mL\dot{z}_0\left(\dot{\phi}_x\sin\phi_x\cos\phi_y + \dot{\phi}_y\cos\phi_x\sin\phi_y\right),\end{aligned} \tag{3.2}$$

and the potential energy

$$\mathcal{P} = -mgL(\cos\phi_x\cos\phi_y - 1) \tag{3.3}$$

Then the Lagrangian can then be found by $\mathcal{L} = \mathcal{K} - \mathcal{P}$. By using the generalised coordinates ϕ_x and ϕ_y and using Euler-Lagrange equations of motion we get the following dynamic model for the crane load

$$\ddot{\phi}_x\cos\phi_y + \frac{g}{L}\sin\phi_x = 2\dot{\phi}_x\dot{\phi}_y\sin\phi_y - \frac{1}{L}\ddot{y}_0\cos\phi_x - \frac{1}{L}\ddot{z}_0\sin\phi_x \tag{3.4}$$

and

$$\begin{aligned}\ddot{\phi}_y + \frac{g}{L}\cos\phi_x\sin\phi_y =& -\dot{\phi}_x^2\sin\phi_y\cos\phi_y + \frac{1}{L}\ddot{x}_0\cos\phi_y \\ &+ \frac{1}{L}\ddot{y}_0\sin\phi_x\sin\phi_y - \frac{1}{L}\ddot{z}_0\cos\phi_x\sin\phi_y.\end{aligned} \tag{3.5}$$

3.3.2. Pendulum Control system

The control system is based on Chapter 3.3 in the work note.

With the assumption that the attachment point only is moving in the horizontal directions x_0 and y_0 and that $\ddot{z}_0 = 0$ we have the feedback damping controller

$$\ddot{y}_0 = 2\zeta\omega_0 L\dot{\phi}_x \tag{3.6}$$

and

$$\ddot{x}_0 = -2\zeta\omega_0 L\dot{\phi}_y - \frac{1}{\cos\phi_y}\ddot{y}_0\sin\phi_x\sin\phi_y \tag{3.7}$$

where ω_0 is given as

$$\omega_0 = \sqrt{\frac{g}{L}}. \tag{3.8}$$

In the project thesis, it was found that the cross term in equation 3.7 made little difference in the controller and sometimes made for worse control. The cross term will therefore be

omitted in this thesis [5].

For the crane tip position control, a similar controller to the one in Chapter 2.5 in the work note was suggested in the project report. This controller has the form

$$u_s = k_p(x_d - x_0) + k_d(\dot{x_d} - \dot{x_0}) \tag{3.9}$$

and

$$u_s = k_p(y_d - y_0) + k_d(\dot{y_d} - \dot{y_0}). \tag{3.10}$$

The position and velocity denoted with the subscript d are here the reference values and the ones with the subscript 0 are found by integrating equation 3.6 and 3.7.

Using the superposition principle the two controller outputs, here u_p for the pendulum damping controller and u_s for the suspension point controller make one common input.

$$\ddot{x_0} = u_p + u_s \tag{3.11}$$

Another part of the controller that was added to the damping controller in equation 3.6 and 3.7 is nonlinear damping. Since the damping is based on the velocity component of the pendulum we get the equation

$$u = \zeta_u |\dot{\phi_i}| \dot{\phi_i}. \tag{3.12}$$

Here ζ_u is the damper gain, absolute value is used so that when the velocity is negative the sign of the damping controller does not change when it should not.

In figure 3.2 the controller for the x-position in Simulink is shown, and the controller for the y-position is identical.

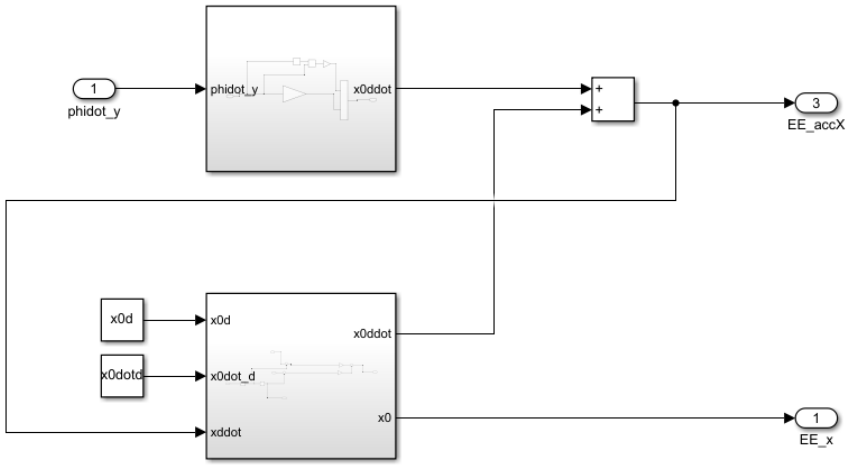


Figure 3.2.: The controller for the x-position, model 1

3.3.3. Crane model

The crane model used for model 1 is from Espen Nilsen's master thesis [4]. For a thorough review of this model, the thesis should be consulted. The crane model consists of a Simulink file and a rigid body tree.

The crane is modelled as an RRP robot arm as can be seen in Figure 3.3. The model was created by using a URDF file and using the smimport('model.urdf') function. URDF is an acronym standing for Universal Robot Description Format. In figure 3.4 the block diagram made from importing the URDF file into Simulink is shown. Several parameters can be set for each joint these include spring stiffness, a damping coefficient and limits for joint velocity and position. The model is controlled by setting a joint position for each joint. In this thesis, these a provided by an inverse kinematic block.

The rigid body tree is a representation of the structure of the robot. It can be used to represent various types of robots including manipulators and other kinematic trees. A rigid body tree consists of rigid bodies attached to one another through joints. A joint defines each rigid body's motion relative to its parent in the tree. By setting a fixed transformation on each joint the transformation from one body to the next is specified [18]. The rigid body tree made in Nilsen's thesis is called "modelcodegen.m" and is used in this thesis.

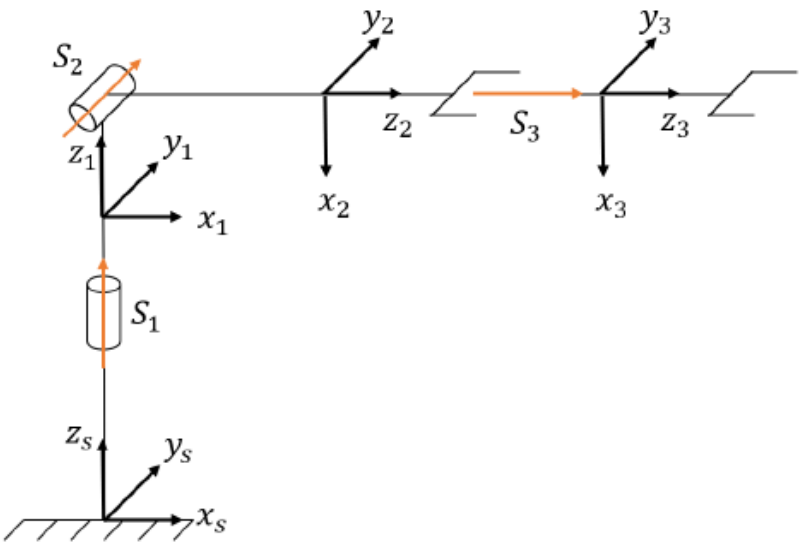


Figure 3.3.: Crane diagram [4]

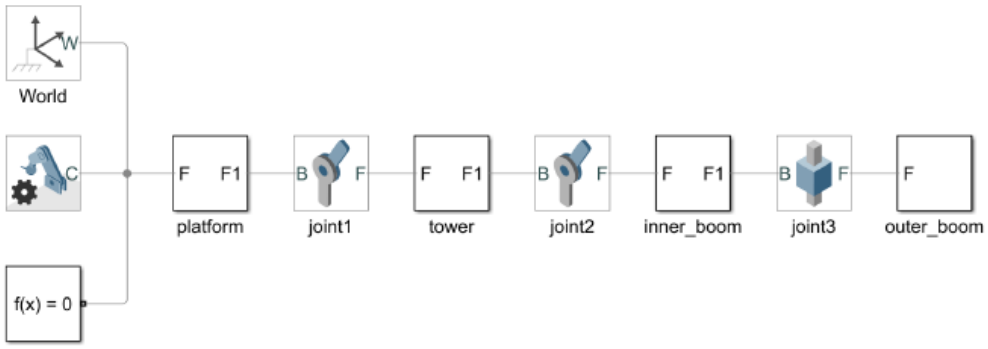


Figure 3.4.: Simscape block diagram [4]

3.3.4. Crane Kinematics

The forward kinematics in model 1 is solved internally in the model using the "Transform Sensor" block from Sismcape. The block takes in information from all the previous joints to calculate the position, velocity and acceleration of the crane end-effector.

The inverse kinematics is solved using the "Inverse Kinematics"-block from the Robotics Toolbox. This block uses the rigid body tree from Espen Nilsen's thesis "modelcode-gen.m". The block takes in the orientation and position of the end-effector which is provided by a "Coordinate Transformation Conversion"-block which takes in the crane-tip positions $[x; y; z]$ and provides a homogeneous transformation matrix. It must also be provided weights, here given as the vector $[0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1]$ this indicates that we want the position to weigh more than the orientation in the solver. An initial guess for the joint configuration is also provided with the vector $[0 \quad 0 \quad 0]$. The solver is selected as "Levenberg-Marquardt", and uses this algorithm to solve the inverse kinematics. The block returns the joint configuration of the crane q.

3.3.5. Model 1 Control Scheme

A block diagram depicting how model 1 is set up in Simulink can be seen in figure 3.5. As can be observed the controller has feedback in the crane-load damping controller and the crane-tip position control uses a sort of feedforward control where the feedback variable is provided by the damping controller instead of from the crane model.

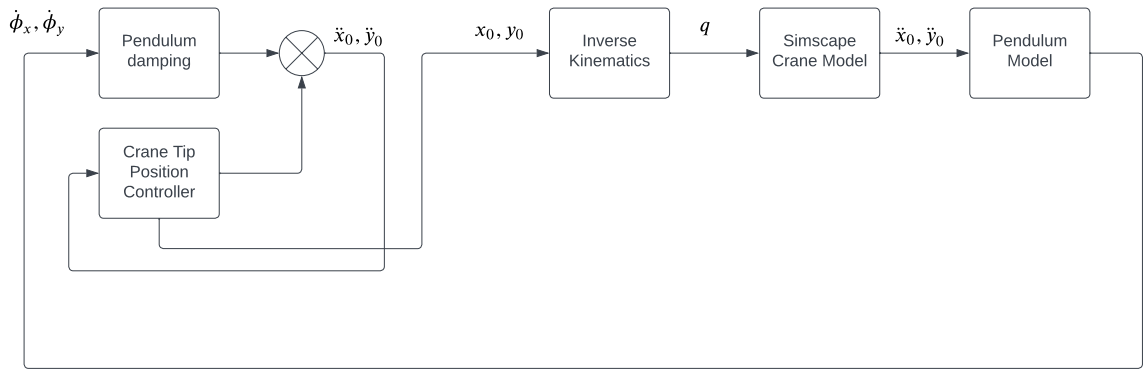


Figure 3.5.: Model 1 control scheme

3.4. Model 2

For the second model, a similar controller was designed only it also incorporates feedback for the position control. A new crane model was also made.

3.4.1. Crane model

To make the following crane model the software from "Modern Robotics: Mechanics, Planning, and Control" [10] was used. This software contained Matlab-functions that could calculate the Inertia matrix $M(q)$ the Coriolis and centripetal force matrix $C(q, \dot{q})\dot{q}$ and the gravity matrix $g(q)$ so that the following dynamic model could be derived

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau \tag{3.13}$$

In Simulink, the system is solved as

$$\ddot{q} = M(q)^{-1}(\tau - C(q, \dot{q})\dot{q} - g(q)) \tag{3.14}$$

The functions used to generate the different matrices were

- M = MassMatrix(thetalist,Mlist,Glist,Slist)
- c = VelQuadraticForces(thetalist,dthetalist,Mlist,Glist,Slist)
- grav = GravityForces(thetalist,g,Mlist,Glist,Slist)

where Mlist is a list of transforms $M_{i-1,i}$, Glist is a list of spatial inertia matrices G_i, Slist is a list of joint screw axes S_i expressed in the base frame and g is the gravitational constant. The lists thetalist and dthetalist are respectively the joint configuration q and joint velocities $\dot{q}$.

Mlist, Glist and Slist are all based on the same model Espen Nilsen used in his master thesis [4] and are

$$M_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.15}$$

$$M_2 = \begin{bmatrix} 0 & 0 & 1 & 0.3 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0.8 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.16}$$

$$M_3 = \begin{bmatrix} 0 & 0 & 1 & 0.3 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0.8 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.17}$$

$$M_{EE} = \begin{bmatrix} 0 & 0 & 1 & 0.6 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0.8 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.18}$$

$$S_{list} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & -0.8 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.19}$$

The spatial inertia matrices are a combination of the inertia tensor and the mass matrix and have the following form

$$G_i = \begin{bmatrix} \mathcal{I}_b & 0_{3x3} \\ 0_{3x3} & mI \end{bmatrix} \tag{3.20}$$

where

$$\mathcal{I} = \begin{bmatrix} \mathcal{I}_{xx} & 0 & \\ 0 & \mathcal{I}_{yy} & 0 \\ 0 & 0 & \mathcal{I}_{zz} \end{bmatrix} \tag{3.21}$$

All inertia tensors are modelled as solid cylinders, where for the first joint the centre of mass is at the centre of the cylinder giving us

$$\mathcal{I}_1 = \begin{bmatrix} \frac{1}{12}m(3r^2 + h^2) & 0 & 0 \\ 0 & \frac{1}{12}m(3r^2 + h^2) & 0 \\ 0 & 0 & \frac{1}{2}mr^2 \end{bmatrix} \tag{3.22}$$

while the two latter joints have their centre of masses at the end of the cylinder giving us

$$\mathcal{I}_{2,3} = \begin{bmatrix} \frac{1}{3}m(3r^2+h^2) & 0 & 0 \\ 0 & \frac{1}{2}m(3r^2+h^2) & 0 \\ 0 & 0 & \frac{1}{3}m(3r^2+h^2) \end{bmatrix} \tag{3.23}$$

The parameters used for the inertia matrices are based on the values found in "model-codegen.m" and can be found in table 3.1. The density this code uses for the joints is the density of carbon steel set to $\rho = 7800kg/m^3$.

Parameter	joint 1	joint 2	joint 3	unit
r_i	0.12	0.10	0.08	m
h_i	0.60	0.60	0.60	m
m_i	211.71	147.03	94.10	kg

Table 3.1.: Parameters for crane inertia matrices

How the model is set up in Simulink is shown in figure 3.6. The initial values used for the crane joints and -tip can be seen in table 3.2. We base the crane on the same parameters as the one used in Nilsen's thesis to continue to use the rigid body tree for future simulations. A rigid body tree is necessary for most of the blocks provided by the robotics toolbox in Simulink making this code valuable for the rest of the simulations.

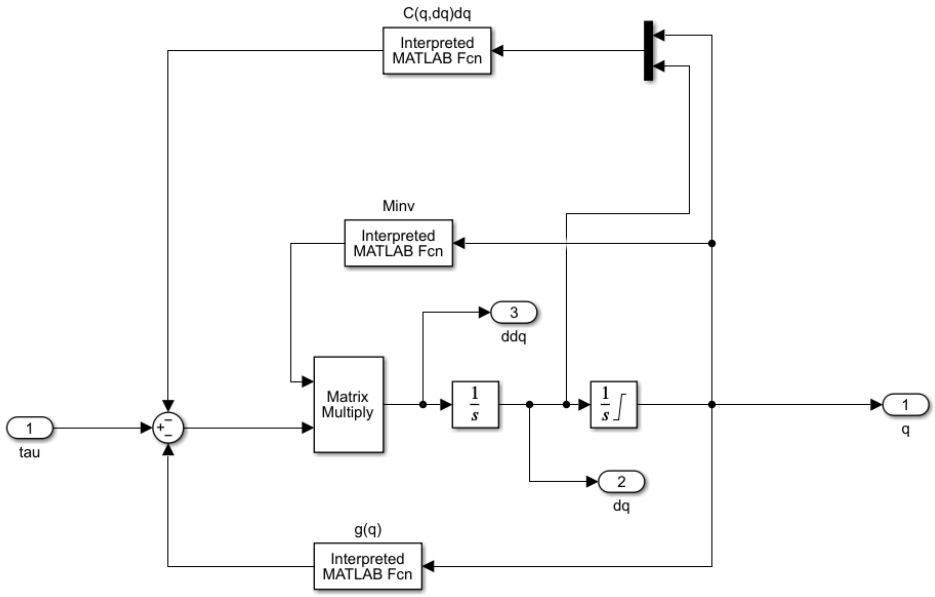


Figure 3.6.: Crane dynamics in Simulink

Parameter	Initial value	Unit
x_0	0.6	m
y_0	0	m
z_0	0.8	m
q_1	0	rad
q_2	0	rad
q_3	0	m

Table 3.2.: Initial values for the crane model

3.4.2. Crane control

A controller is also necessary for providing the crane dynamic model with a torque input. The controller is of type feedforward plus feedback linearisation. Given a typical PD controller with feedforward

$$(\ddot{q}_d - \ddot{q}) + k_d(\dot{q}_d - \dot{q}) + k_p(q_d - q) = 0. \tag{3.24}$$

This can be solved with respect to $\ddot{q}$

$$\ddot{q} = \ddot{q}_d + k_d(\dot{q}_d - \dot{q}) + k_p(q_d - q). \tag{3.25}$$

This can in turn be inserted into the equation 2.38 which gives the following controller, also known as the computed torque controller

$$\tau = M(q)(\ddot{q}_d + k_d(\dot{q}_d - \dot{q}) + k_p(q_d - q)) + C(q, \dot{q})\dot{q} + g(q). \tag{3.26}$$

In figure 3.7 one can see how this is implemented in Simulink. One of the drawbacks of such a controller is that it requires all three joint variables as input. This creates a need for inverse kinematics that solves the joint position, -velocity and -acceleration.

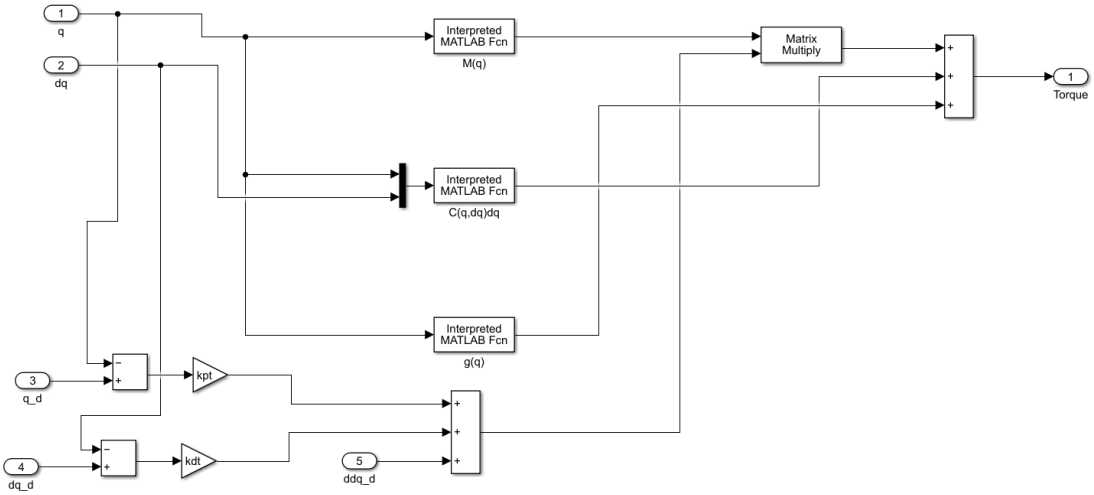


Figure 3.7.: Torque controller in Simulink

3.4.3. Crane Kinematics

The forward Kinematics is solved using the product of the exponential formula, the function used is taken from Nilsen's master thesis [4]. For this model, there is also a need to solve the forward kinematics for the crane-tip velocity and -acceleration. These are solved using the equation 2.18 and 2.26. The Jacobian is solved using a function from the modern robotics toolbox. The function is "Js = JacobianSpace(Slist,thetalist)", which takes in a list of joint screws and joint configuration to solve the space Jacobian $J_s(q)$. The "Derivative" block in Simulink is used to create the time derivative of the Jacobian $\dot{J}(q)$.

The second model uses the same inverse kinematics as the previous model for solving the joint position, This can be done since we base the crane dynamic model on the same parameters as the rigid body tree from model 1. However, this model also requires the joint velocity and -acceleration for the input to the crane controller. These are solved using equation 2.35 and 2.36. Since we only have 3 joints we have 6×3 Jacobian. In order to make the Jacobian invertible, the size of the Jacobian is altered in the same manner as equation 2.37. Since we only control the x- and $y-$ position, we technically could make do with a 2×2 Jacobian. However, the z-position is also included since this position will also have to be controlled in future. Solving the Jacobian with only these three positions in mind we get a 3×3 Jacobian which is invertible.

3.4.4. Crane-Tip Set-Points

To create the set-point for the crane position and if applicable the velocity and acceleration, transfer functions were used. With the help of a simple second-order transfer function with a step function input the position can be represented as

$$G_{pos}(s) = \frac{1}{T_2 s^2 + T_1 s + 1}. \tag{3.27}$$

The time derivative of position is velocity which can be represented as a transfer function as

$$G_{vel}(s) = \frac{s}{T_2 s^2 + T_1 s + 1}. \tag{3.28}$$

and the second time derivative of position is acceleration which is given as

$$G_{acc}(s) = \frac{s^2}{T_2 s^2 + T_1 s + 1}. \tag{3.29}$$

Where the "s" and "s^2" in the numerator denote the derivative and second derivative respectively. The functions can be tuned by adjusting T_1 and T_2. One issue with transfer functions is that one cannot set initial conditions, for variables where this is necessary, one can turn the transfer function into state space models by using the Matlab function "tf2ss". This function takes in the numerator adn denominator of the transfer function and provides us with the A, B, C and D matrices which are needed to make a state space model. These matrices are used in the "State-space"-block in Simulink in which initial conditions can be specified.

The benefit of setting the set points this way, contrary to using normal step functions, is that the velocity and acceleration set points make sense compared to the position we want to achieve. Setting the set-point of the velocity to a constant zero while the position is a step function from zero to one, will for example create a discrepancy between the position we want and the velocity wanting to stay at zero. Using these transfer functions, however, both the position and the velocity strive to achieve the same position.

3.4.5. Feedback controller

The feedback controller for crane-tip position control and damping of the crane load used in Model 2 is based on the article "Vision-Based Control of a Knuckle Boom Crane With Online Cable Length Estimation" by Geir O. Tysse [19]. It should be noted that cable length estimation will not be included. The same pendulum model as model 1 is used, this is slightly different from the one used in the paper which also includes variable cable length. A noteworthy difference from model 1 is that the crane tip position controller is in the outer loop and the pendulum damping controller is in the inner loop. Tysse relates this to there is more need for fast control for the pendulum oscillations than for the crane tip.

The following damping controller proposed

$$\begin{aligned} \ddot{x_0} &= 2L\zeta\omega_0\dot{\phi}_y + u_x, \\ \ddot{y_0} &= -2L\zeta\omega_0\dot{\phi}_x + u_y, \end{aligned} \tag{3.30}$$

where u_x and u_y are the crane-tip position controller, which is a PD controller

$$\begin{aligned} u_x &= k_p(x_d - x_0) + k_d(\dot{x}_d - \dot{x}_0), \\ u_y &= k_p(y_d - y_0) + k_d(\dot{y}_d - \dot{y}_0). \end{aligned} \tag{3.31}$$

The gains can be selected as $k_p = \omega_s^2$ and $k_d = 2\zeta_s\omega_s$. Where $\omega_s << \omega_0$ and ζ_s can be selected in the range $[0.7, 1]$. Setting $\omega_s = \omega_0/k_s$ where $k_s \geq 5$ is noted by Tysse as sufficient.

The paper notes that it in practice will be impossible to command the acceleration of the crane tip. It, therefore, suggests using the following equations for commanding the velocity instead

$$\dot{\omega}_x = \ddot{x}_0, \quad \dot{\omega}_y = \ddot{y}_0, \tag{3.32}$$

$$\begin{aligned} \dot{v}_x &= \frac{1}{T_v}(\omega_x - v_x), \\ \dot{v}_y &= \frac{1}{T_v}(\omega_y - v_y). \end{aligned} \tag{3.33}$$

T_v is here the bandwidth of the velocity loop it is proposed by Tysse that this should be sufficiently faster than the damping. It should be noted that in this model, countary to the one Tysse designed in [19], the joint position, velocity and acceleration needs to be provided to the inverse kinematics. This is for the crane torque controller. In Simulink equation 3.32 and 3.33 is therefore modelled as seen in figure 3.8, providing the crane-tip position, -velocity and -acceleration to the inverse kinematics.

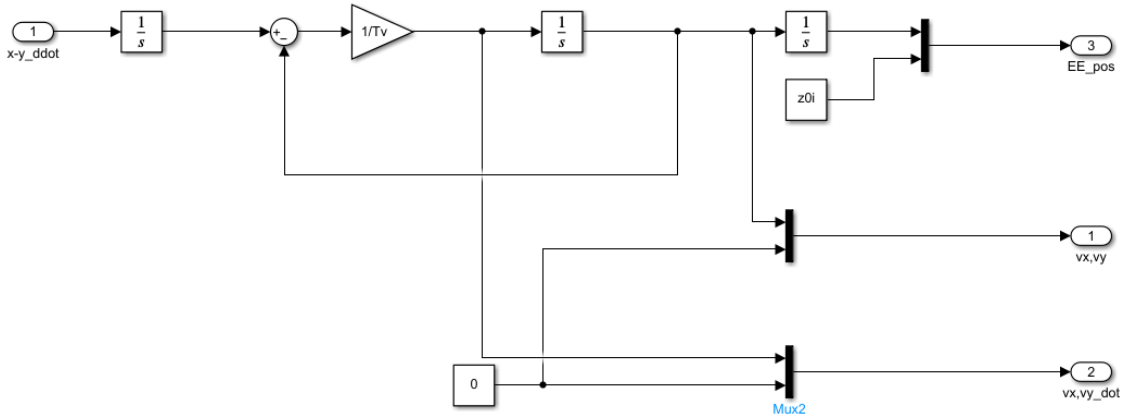


Figure 3.8.: Equations 3.32 and 3.33 as modelled in Simulink

The block diagram in figure 3.9 shows how model 2 is built in Simulink. This control scheme is a clear use of cascade control where we have two nested feedback loops. The damping controller is here the inner loop and the crane-tip position control is the outer loop.

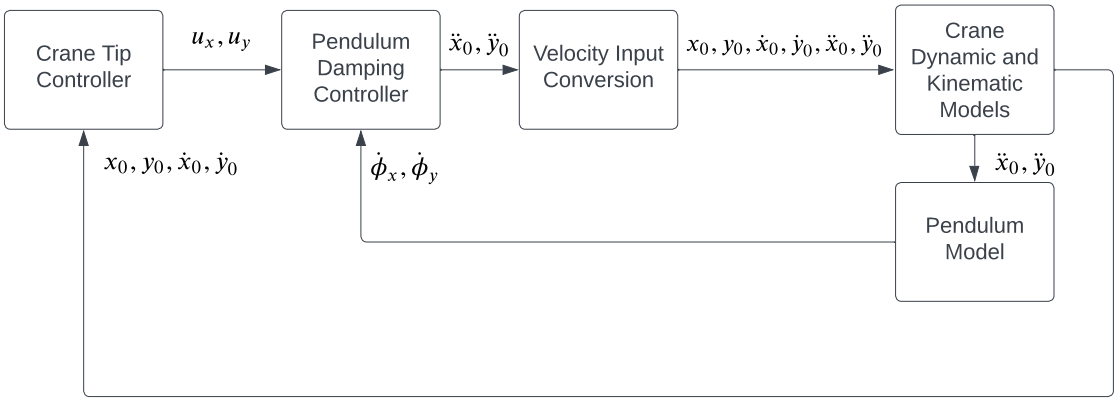


Figure 3.9.: Model 2 control scheme

3.5. Model 3

The same dynamic and kinematic models for the crane and crane load are used for the third model. As with model 2, this model is based on Geir Ole Tysse's PhD-thesis. This controller is based on the article "Crane load position control using Lyapunov-based pendulum damping and nonlinear MPC position control " [20], and the complete derivation of the controller can be found in this article.

The controller design starts with a basic damping controller as the feedback control laws, similar to the ones used in the preceding models. It should be noted that here s and c mean respectively sin and cos, the subscript x and y denote that we are either using the angle ϕ_x or ϕ_y

$$\ddot{y_0} = -2\zeta\omega_0 L\dot{\phi_x}, \tag{3.34}$$

$$\ddot{x_0} = 2\zeta\omega_0 L\dot{\phi_y} - \frac{\ddot{y_0}s_x s_y}{c_y}. \tag{3.35}$$

These equations can be inserted into the pendulum model. The pendulum model used by Tysse is similar to the ones from equations 3.4 and 3.5, but with $\ddot{z}_0$ set to zero

$$\ddot{\phi_x}c_y + 2\zeta\omega_0\dot{\phi_x}c_x + \omega_0^2 s_x = 2, \dot{\phi_x}\dot{\phi_y}s_y \tag{3.36}$$

$$\ddot{\phi_y} + 2\zeta\omega_0\dot{\phi_y}c_y + \omega_0^2 c_x s_y = \dot{\phi_x}^2 s_y s_y, \tag{3.37}$$

and then linearised with $\phi_x = \phi_y = 0$ and $\dot{\phi_x} = \dot{\phi_y} = 0$

$$\ddot{\phi_x} + 2\zeta\omega_0\dot{\phi_x} + \omega_0^2\phi_x = 0, \tag{3.38}$$

$$\ddot{\phi}_y + 2\zeta\omega_0\dot{\phi}_y + \omega_0^2\phi_y = 0. \tag{3.39}$$

When the suspension point is stationary, the pendulum's kinetic and potential energy can then be used to analyse the closed-loop dynamics of the nonlinear system.

$$V_d = \frac{1}{2}mL^2(\dot{\phi}_x^2 c_y^2 + \dot{\phi}_y^2) + mgL(1 - c_x c_y) \tag{3.40}$$

Which has the time derivative along the trajectories of motion

$$\dot{V}_d = mL\dot{\phi}_x\ddot{y}_0 c_y c_x + mL\dot{\phi}_y(\ddot{x}_0 c_y + \ddot{y}_0 s_x s_y) \tag{3.41}$$

Inserting equation 3.34 and 3.35 we then have its closed loop dynamics

$$\dot{V}_d = -2\zeta\omega_0 mL^2(\dot{\phi}_x^2 c_x c_y + \dot{\phi}_y^2 c_y) \tag{3.42}$$

Using LaSalle's theorem Tysse explains that at the equilibrium points, the same point as the system is linearised at, it is asymptotically stable.

These equation leads to the second step of the controller design which is to design an exponentially stable Lyapunov controller for the pendulum motions in ϕ_x and ϕ_y. The accelerations of the suspension point are considered control variables $\ddot{x}_0 = u_x$ and $\ddot{y}_0 = u_y$.

The feedback control laws used are

$$u_y = -\frac{Lc_y}{c_x}(k_d\dot{\phi}_x + k_p\phi_x) - \frac{2Ls_y}{c_x}\dot{\phi}_x\dot{\phi}_y + g\frac{s_x s_y^2}{c_x} \tag{3.43}$$

$$u_x = \frac{L}{c_y}(k_d\dot{\phi}_y + k_p\phi_y) - Ls_y\dot{\phi}_x^2 - \frac{u_y s_x s_y}{c_y} \tag{3.44}$$

The controller gains are here k_d and k_p. The feedback variables are the angle and the angle rate of the pendulum. The closed-loop dynamics of the system with the above controller laws are

$$\ddot{\phi}_x + k_d\dot{\phi}_x + k_p\phi_x + \omega_0^2 c_y s_x = 0 \tag{3.45}$$

$$\ddot{\phi}_y + k_d\dot{\phi}_y + k_p\phi_y + \omega_0^2 c_x s_y = 0 \tag{3.46}$$

The Lyapunov function can then be deduced, the state vector is set to

$$x = [\phi_x, \dot{\phi}_x, \phi_y, \dot{\phi}_y]^T \tag{3.47}$$

The Lyapunov function used is

$$V(x) = \frac{1}{2}x^T Px + mgL(1 - c_x c_y) \tag{3.48}$$

where P is the real symmetric positive definite matrix given by

$$P = \begin{bmatrix} p_{11} & p_{13} & 0 & 0 \\ p_{13} & p_{22} & 0 & 0 \\ 0 & 0 & p_{11} & p_{13} \\ 0 & 0 & p_{13} & p_{22} \end{bmatrix} \tag{3.49}$$

where

$$p_{11} = (k_p + ck_d)p_{22}, \quad p_{13} = cp_{22}, \quad p_{22} = mL^2 \tag{3.50}$$

c is here a positive constant satisfying $c < k_d$.

The following domain is considered

$$D = \{x| \quad |\phi_x| < \frac{\pi}{2} \quad \text{and } |\phi_y| < \frac{\pi}{2} - \delta\} \tag{3.51}$$

where $0 < \delta < \pi/2$. With further investigation of the Lyapunov function, Tysse finds the Lyapunov function's upper and lower bounds are bounded by

$$k_1||x||_2^2 \leq V(x) \leq k_2||x||_2^2. \tag{3.52}$$

Using the smallest eigenvalue of P λ_{min}, and $\tilde{P} = P + mgL\ diag(1, 0, 1, 0)$, we have that $k_1 = (1/2)\lambda_{min}(P)$ and $k_2 = (1/2)\lambda_{min}(\tilde{P})$.

The time derivative of the Lyapunov function along the solutions of the closed-loop system is found to be as follows

$$\begin{aligned} \dot{V}(X) = -p_{13}k_p(\phi_x^2 + \phi_y^2) - p_{22}k_d(\dot{\phi_x^2} + \dot{\phi_y^2}) \\ -p_{13}\omega_0^2(\phi_x s_x c_y + \phi_y s_y c_x) \end{aligned} \tag{3.53}$$

Tysse found with further analysis of the time derivative of the Lyapunov function that

$$\dot{V}(x) \leq -k_3||x||_2^2 \tag{3.54}$$

with $k_3 = \min\{p_{13}k_p, p_{22}k_d\}$ and concludes that equations 3.45 and 3.46 are exponentially stable in D.

Since it in practice will not be possible to use acceleration as a control input, non-vanishing perturbations are introduced. Non-vanishing perturbations can be explained as disturbances that do not vanish. In this case, they are deviations between the input value and the acceleration, and given as

$$\ddot{x_0} = u_x + g_x \tag{3.55}$$

$$\ddot{y_0} = u_y + g_y \tag{3.56}$$

where u_x and u_y are the acceleration according to the control laws from equation 3.43 and 3.5. The closed-loop dynamics with non-vanishing perturbations then becomes

$$\ddot{\phi}_x + k_d\dot{\phi}_x + k_p\phi_x + \omega_0^2 c_y s_x - \frac{c_x}{c_y L} g_y = 0 \tag{3.57}$$

$$\ddot{\phi}_y + k_d\dot{\phi}_y + k_p\phi_y + \omega_0 c_x s_y + \frac{c_y}{L} g_x + \frac{s_x s_y}{L} g_y = 0 \tag{3.58}$$

Tysse then uses Lyapunov stability to show that the perturbed system satisfies

$$||x(t)||_2 \leq k e^{-\varphi t} ||x(0)||_2. \quad \forall t < T \tag{3.59}$$

and

$$||x(t)||_2 \leq b, \quad \forall t \geq T \tag{3.60}$$

where T is finite and

$$k = \sqrt{\frac{k_2}{k_1}}, \quad \varphi = \frac{(1-\theta)k_3}{2k_2}, \quad b = \frac{k_4}{k_3}\sqrt{\frac{k_2}{k_1}}\frac{\gamma}{\theta} \tag{3.61}$$

The next step is to design the crane-tip position control, this is to be done by nonlinear MPC. Tysse suggests the following closed-loop dynamic system for the NMPC.

$$\ddot{x_0} = g_x + \frac{L(k_d\dot{\phi}_y + k_p\phi_y) - Lc_y s_y \dot{\phi}_x^2 - u_y s_x s_y}{c_y}, \tag{3.62}$$

$$\ddot{y_0} = g_y - \frac{Lc_y(k_d\dot{\phi}_x + k_p\phi_x) + 2Ls_y\dot{\phi}_x\dot{\phi}_y - g s_x s_y^2}{c_x}, \tag{3.63}$$

$$\ddot{\phi}_x = -k_d\dot{\phi}_x - k_p\phi_x - \omega_0^2 c_y s_x + \frac{c_x}{c_y L} g_y, \tag{3.64}$$

$$\ddot{\phi}_y = -k_d\dot{\phi}_y - k_p\phi_y - \omega_0^2 c_x s_y + \frac{c_y}{L} g_x - \frac{s_x s_y}{L} g_y. \tag{3.65}$$

Here the state vector is given as

$$z = [x_0 \quad \dot{x_0} \quad y_0 \quad \dot{y_0} \quad \phi_x \quad \dot{\phi}_x \quad \phi_y \quad \dot{\phi}_y]^T \tag{3.66}$$

the control vector is given by the non-vanishing perturbations g_x and g_y

$$w = [g_x \quad g_y]^T \tag{3.67}$$

The following state space model is introduced

$$\dot{z}(t) = f(z(t), w(t)), \quad z(0) = z_0 \tag{3.68}$$

where the perturbation is bounded by γ as follows

$$\mathbf{U} := \{w| \ |g_x| < \gamma \text{ and } |g_y| < \gamma\} \tag{3.69}$$

To solve this system in Simulink the nonlinear MPC block is used. For this block to work we need a state function and a state output function. The state function is based on 3.68 and solved as

$$\dot{z} = [x_2 \quad \ddot{x_0} \quad x_4 \quad \ddot{y_0} \quad x_6 \quad \ddot{\phi_x} \quad x_8 \quad \ddot{\phi_y}]^T \tag{3.70}$$

where the states are set to

$$z = [x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7 \quad x_8] \tag{3.71}$$

The output function is set to

$$y = z \tag{3.72}$$

so that each output can accept a reference.

In addition, the Jacobian for both the state- and output function is provided. The Jacobians are found using the following equation for the state matrix

$$A(i,j) = \frac{\partial z(i)}{\partial x(j)} \tag{3.73}$$

for the input matrix

$$B_{mv}(i,j) = \frac{\partial z(i)}{\partial w(j)} \tag{3.74}$$

and for the output function

$$C(i,j) = \frac{\partial y(i)}{\partial x(j)} = I_{8x8} \tag{3.75}$$

where I_{8x8} is the 8x8 identity matrix.

To create the inverse kinematics input a similar solution to model 2 is used, where the velocity is commanded instead of the acceleration. For this model the equation used are

$$\dot{\omega}_x = u_x + g_x, \quad \dot{\omega}_y = u_y + g_y, \tag{3.76}$$

$$\begin{aligned} \ddot{x}_0 &= \frac{1}{T_v}(\omega_x - \dot{x}_0) \\ \ddot{y}_0 &= \frac{1}{T_v}(\omega_y - \dot{y}_0) \end{aligned} \tag{3.77}$$

here $1/T_v$ is the bandwidth of the velocity loop. The weights of the NMPC are set to

$$Q = diag([Q_{x_0} \quad Q_{\dot{x}_0} \quad Q_{y_0} \quad Q_{\dot{y}_0} \quad 0 \quad 0 \quad 0 \quad 0]) \tag{3.78}$$

and the reference is set to

$$z_r = [x_{0r} \quad \dot{x}_{0r} \quad y_{0r} \quad \dot{y}_{0r} \quad 0 \quad 0 \quad 0 \quad 0]^T \tag{3.79}$$

Some bounds were also set on the crane-tip position and velocity in the NMPC controller. The velocity is bounded by $|v| = 3.5\gamma$ where gamma is the same as used for the perturbations. The position is bounded by ± 1.2 which is the maximum reach of the crane in both directions.

The Nonlinear MPC block requires a nonlinear object that defines the parameters and functions that should be used by the block. The object and functions are provided in the digital attachment. For more information about the NMPC block and nonlinear object, the following MathWorks articles should be consulted [21, 22]. It should be noted that the NMPC block needs a discrete input, usually, this is provided by state estimation, by for example a Kalman filter. In this model the measurements are however assumed perfect, the "Zero-Order Hold"-block is therefore used to make the discrete input. This block provides the NMPC with an input that matches its own sample time.

The block diagram in figure 3.10 shows the setup of model 3 in Simulink. This also applies a form of cascade where the outer loop is the NMPC position control and the inner loop is the crane load damping controller.

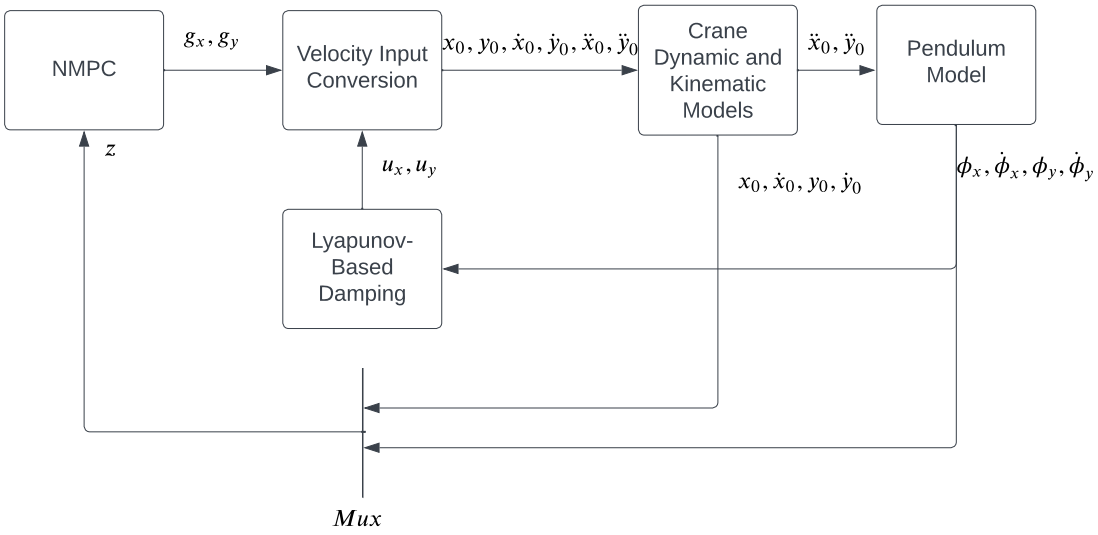


Figure 3.10.: Model 3 control scheme

Chapter 4.

Results

4.1. Model 1

In this section, the results from the simulation of model 1 are presented. This controller uses only feedback for the pendulum damping controller and employs feedforward from this controller into the crane tip position controller.

4.1.1. Model 1 Test 1

The first test was conducted using the parameters in table 4.1. In this test, the crane load starts with no pendulation and there are steps in the crane-tip position control.

Parameter	Value	Unit
k_p	1.6	N/A
k_d	5	N/A
ζ	0.2	N/A
ζ_u	0.25	N/A
x_{0i}	0.6	m
y_{0i}	0	m
z_{0i}	0.8	m
x_{0d}	0.7	m
y_{0d}	0.6	m
ϕ_{xi}	0	degrees
ϕ_{yi}	0	degrees

Table 4.1.: Parameters for controller, model 1 test 1

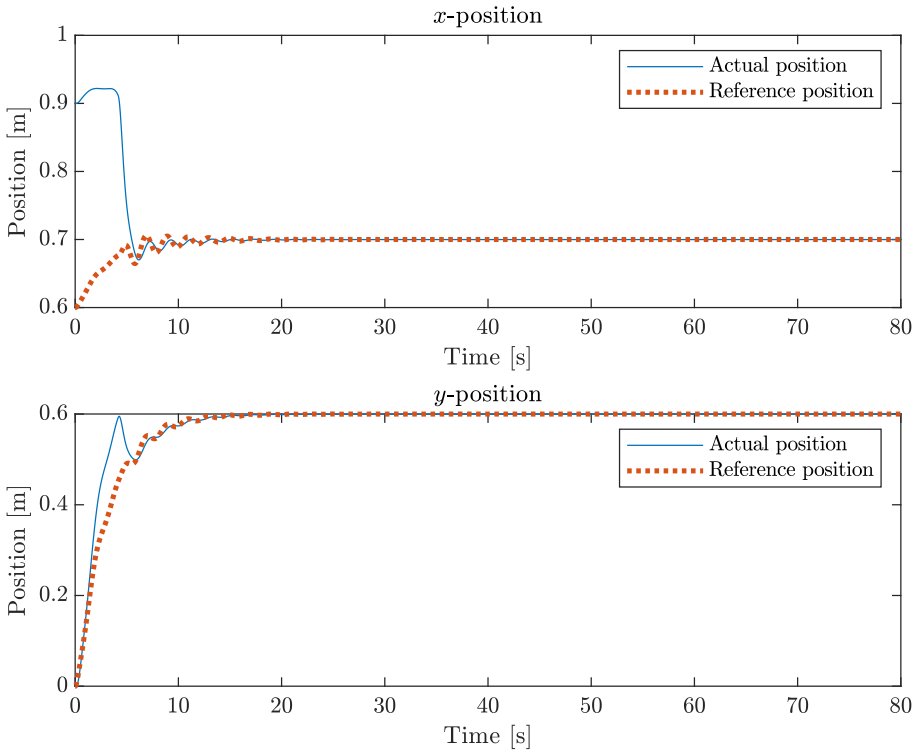


Figure 4.1.: Model 1, test 1 x- and y-position

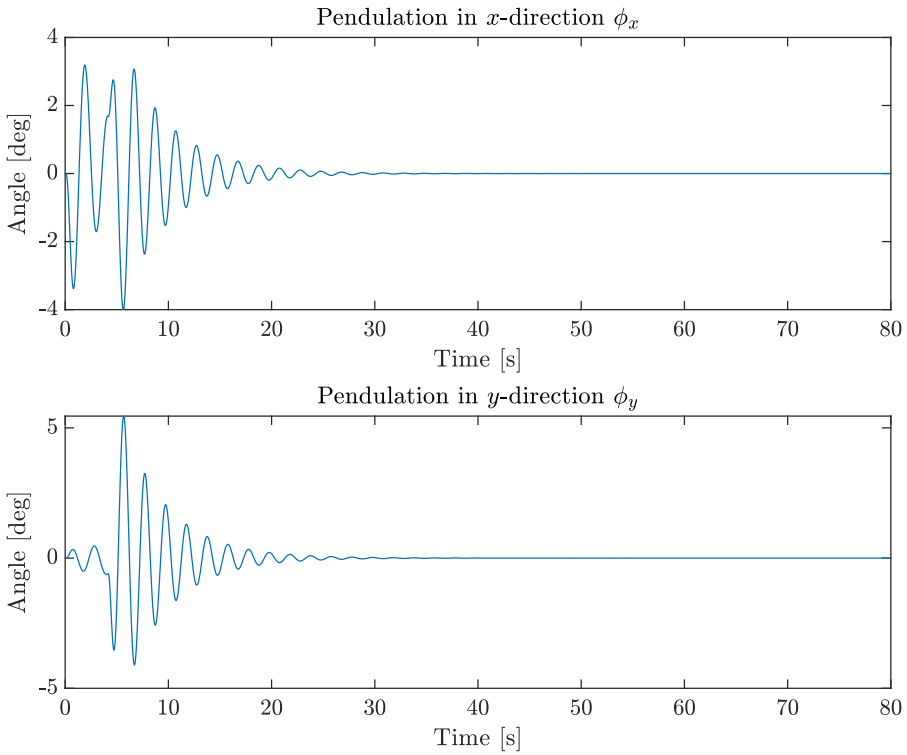


Figure 4.2.: Model 1, test 1 ϕ_x and ϕ_y

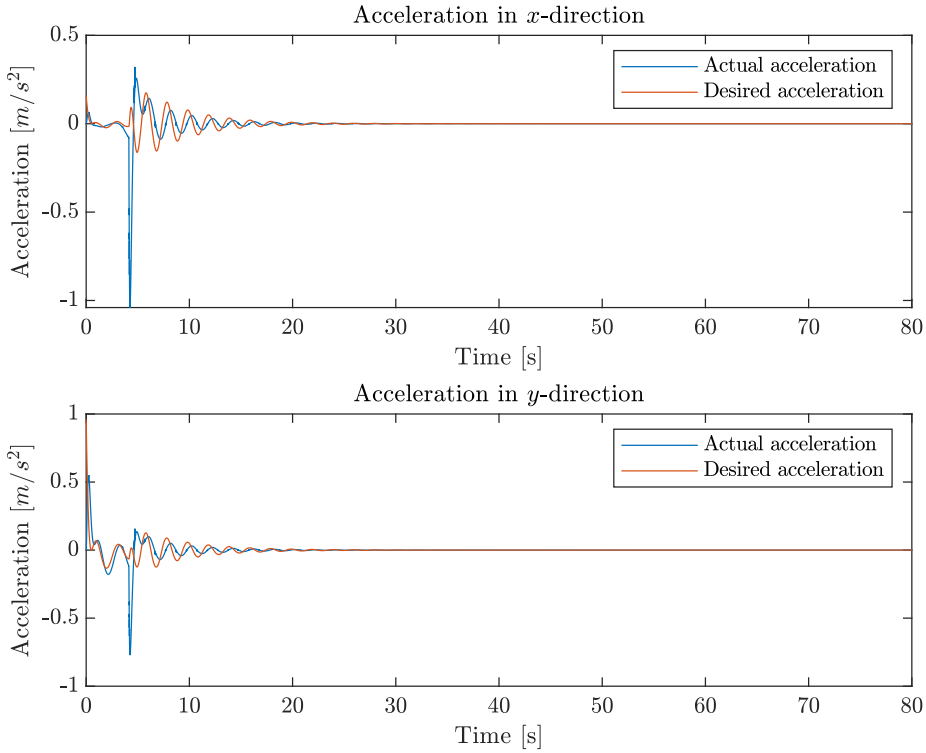


Figure 4.3.: Model 1, test 1 acceleration in x- and y- direction

4.1.2. Model 1 Test 2

Test 2 uses the same parameters as test 1 but with initial conditions of the pendulum angle set to $-20°$ and $15°$ for ϕ_x and ϕ_y respectively.

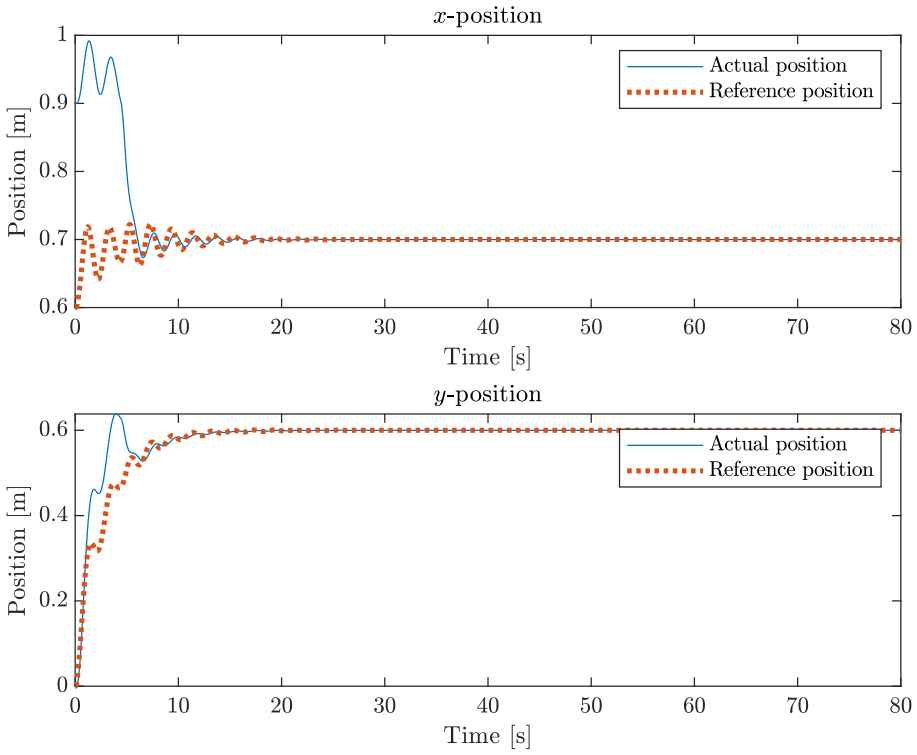


Figure 4.4.: Model 1, test 2 x- and y-position

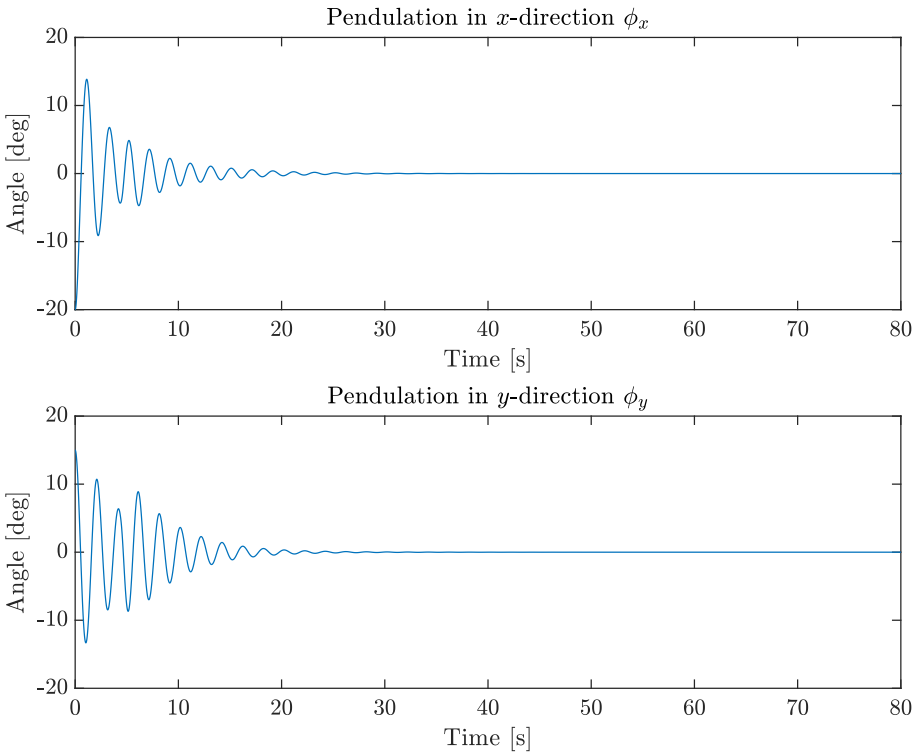


Figure 4.5.: Model 1, test 2 ϕ_x and ϕ_y

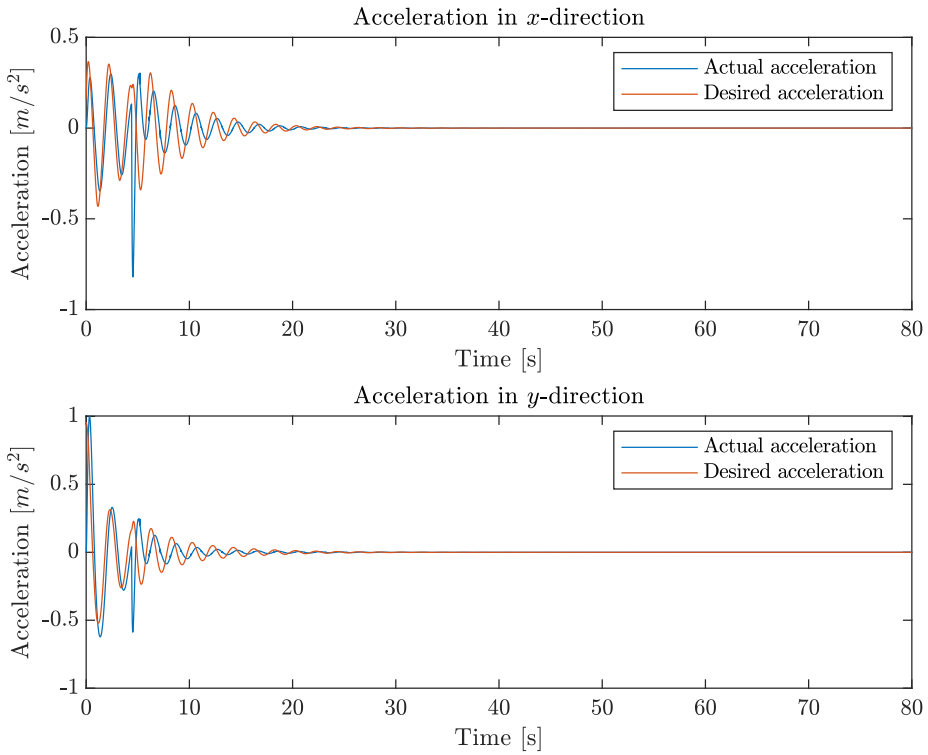


Figure 4.6.: Model 1, test 2 acceleration in the x- and y- directions

4.2. Model 2

For model 2 there is first a test of the new crane model. This model is then tested with the new controller that has feedback for both the crane load damping and crane-tip position controller.

4.2.1. New Crane Model

The crane model and torque controller are tested with a trajectory input where q, $\dot{q}$ and $\ddot{q}$ are provided. This trajectory is taken from the Espen Nilsens's master thesis [4]. With k_p set to 10 and k_d set to 15, we get the difference between the actual and desired position seen in figure 4.7 and the joint torque inputs seen in figure 4.8.

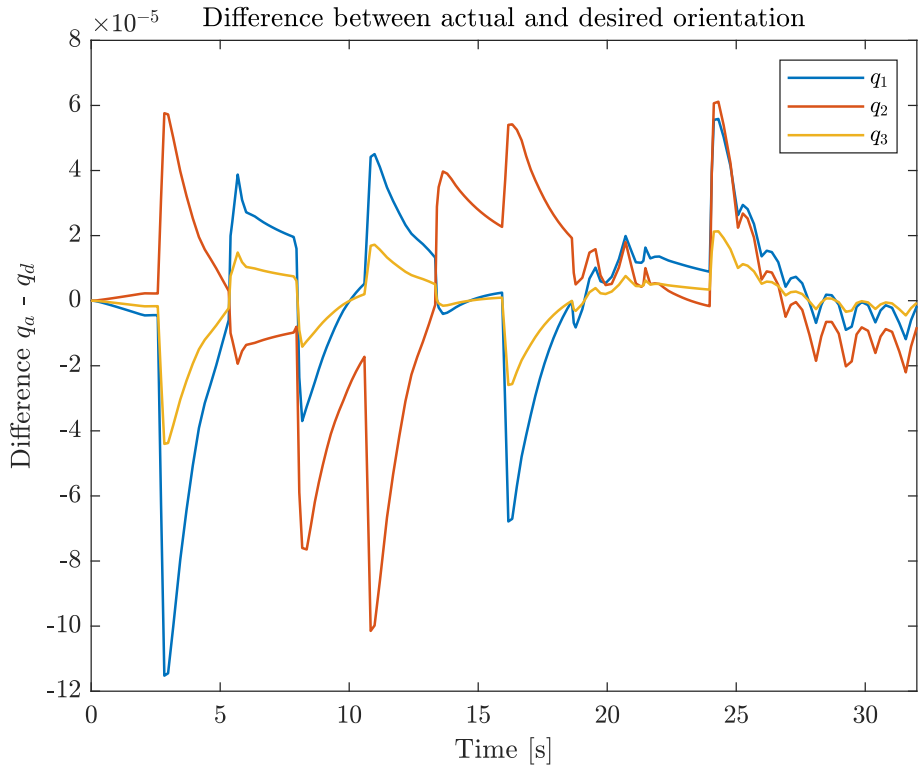


Figure 4.7.: Difference between the actual and desired orientation of crane joints

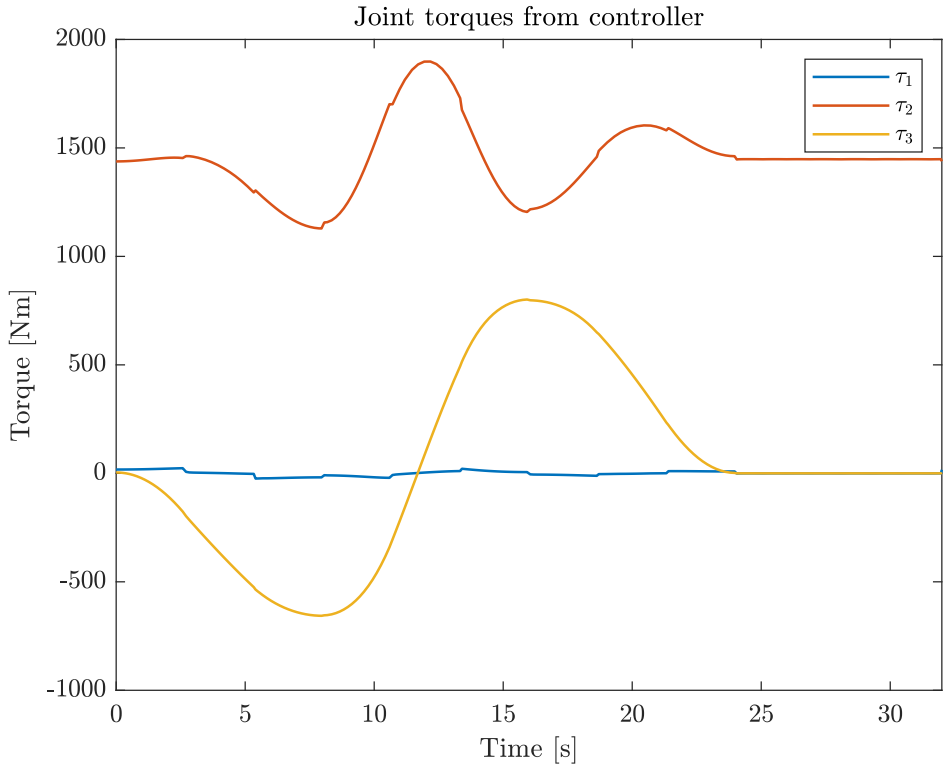


Figure 4.8.: Joint torque input from controller

4.2.2. Model 2 Test 1

The first test of the new controller was conducted with the parameters seen in table 4.2. Here there are steps in the crane-tip position, but the crane load starts with no pendulation. The steps occur at time 30 s to allow the system to stabilise first.

Parameter	Value	Unit
ζ	0.2	N/A
ζ_s	0.8	N/A
ω_s	$\frac{\omega_0}{5}$	N/A
k_{pt}	10	N/A
k_{dt}	15	N/A
T_v	5	s
x_{0i}	0.6	m
y_{0i}	0	m
z_{0i}	0.8	m
x_{0d}	0.7	m
y_{0d}	0.6	m
ϕ_{xi}	0	degrees
ϕ_{yi}	0	degrees

Table 4.2.: Parameters for controller, model 2 test 1

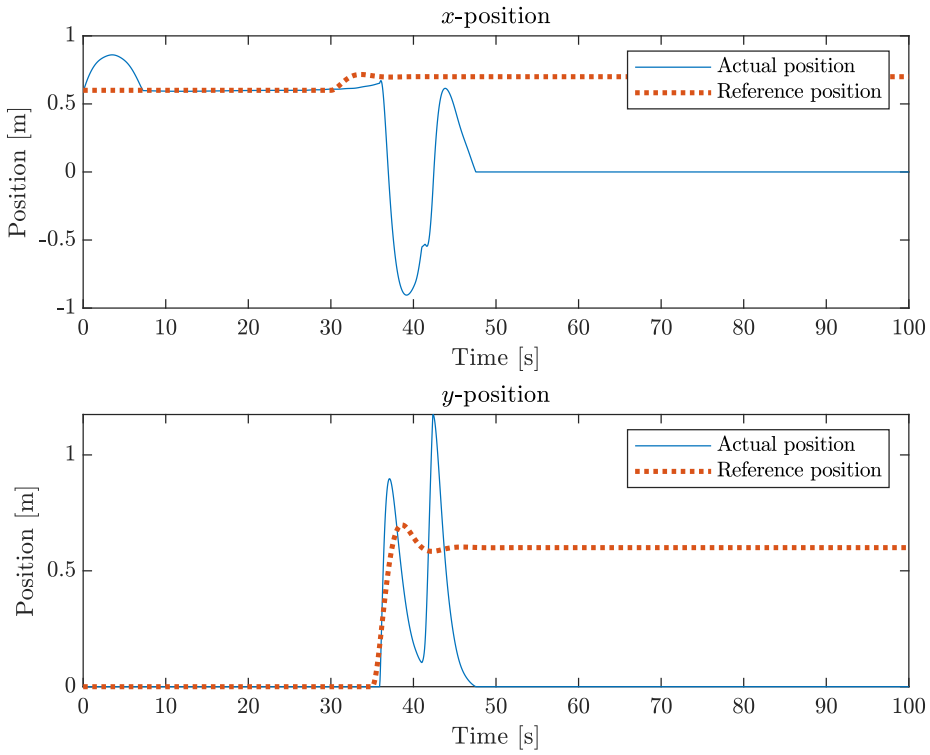


Figure 4.9.: Model 2, test 1 actual and reference x- and y-position of the crane-tip

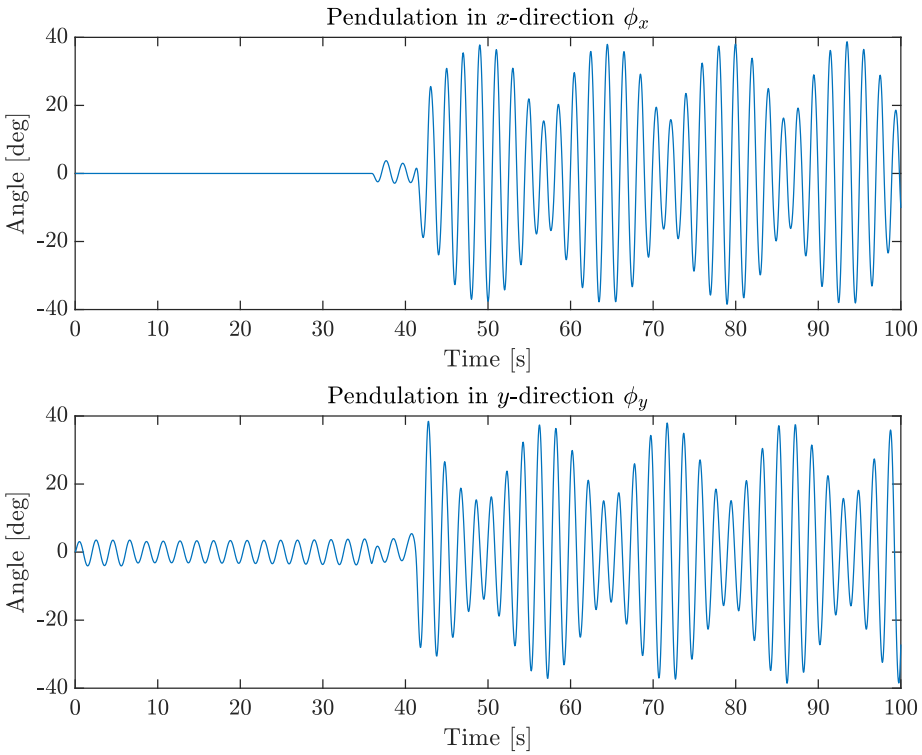


Figure 4.10.: Model 2 test 1 ϕ_x and ϕ_y

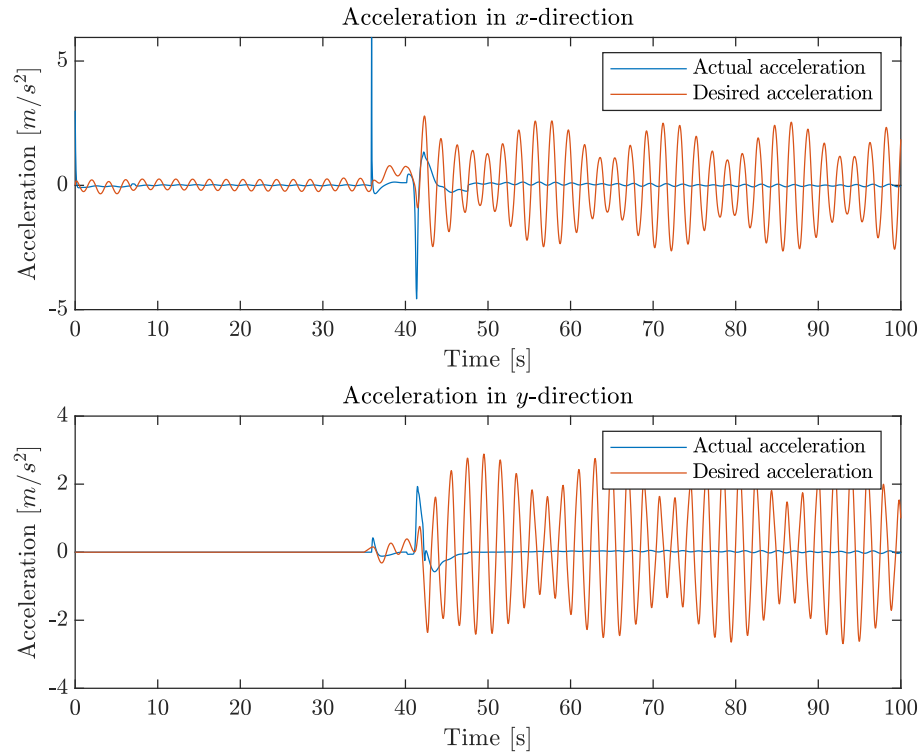


Figure 4.11.: Model 2, test 1 actual and desired x- and y - acceleration

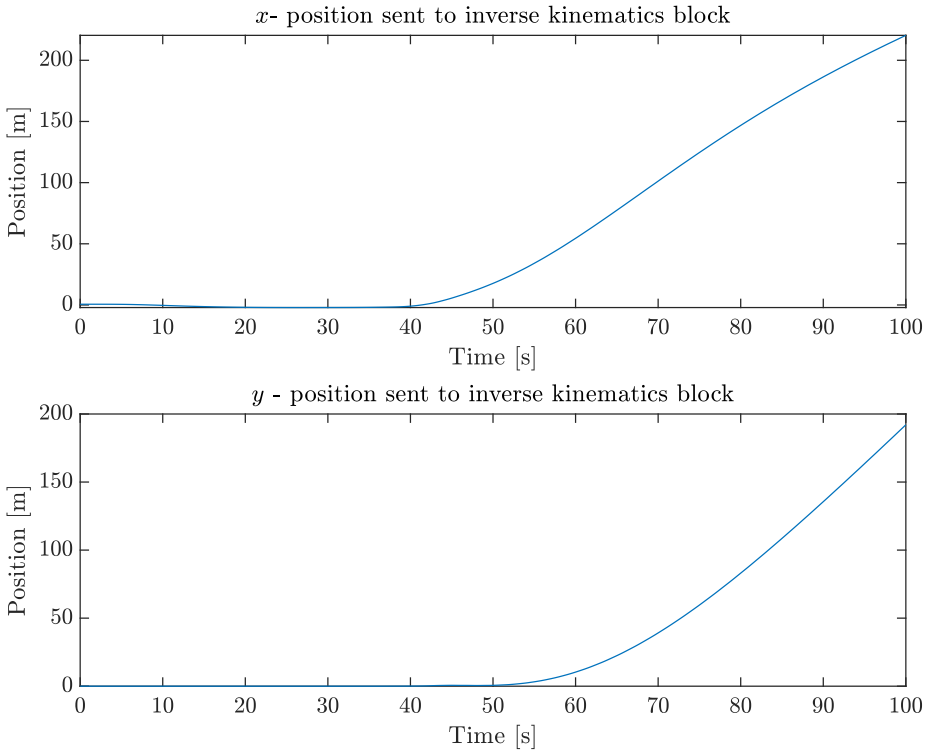


Figure 4.12.: Model 2, test 1 desired trajectory

4.2.3. Model 2 Test 2

The second test is made with the same condition as in table 4.2, with the initial conditions for the crane load ϕ_{xi} and ϕ_{yi} changed to $-20°$ and $15°$ respectively.

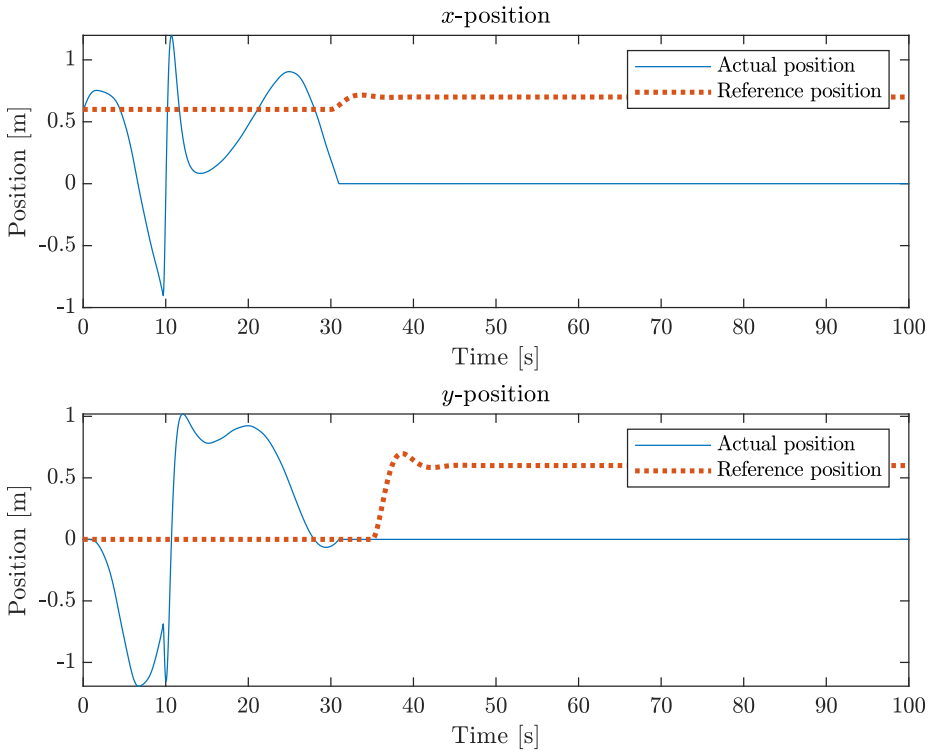


Figure 4.13.: Model 2, test 2 actual and reference x- and y-position of the crane-tip

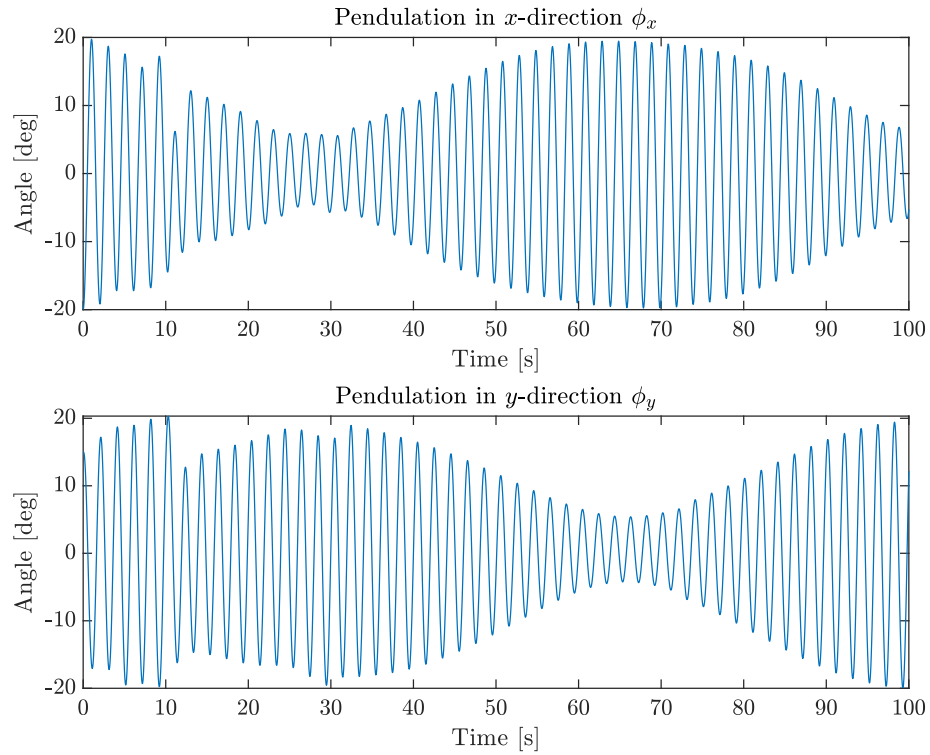


Figure 4.14.: Model 2, test 2 ϕ_x and ϕ_y

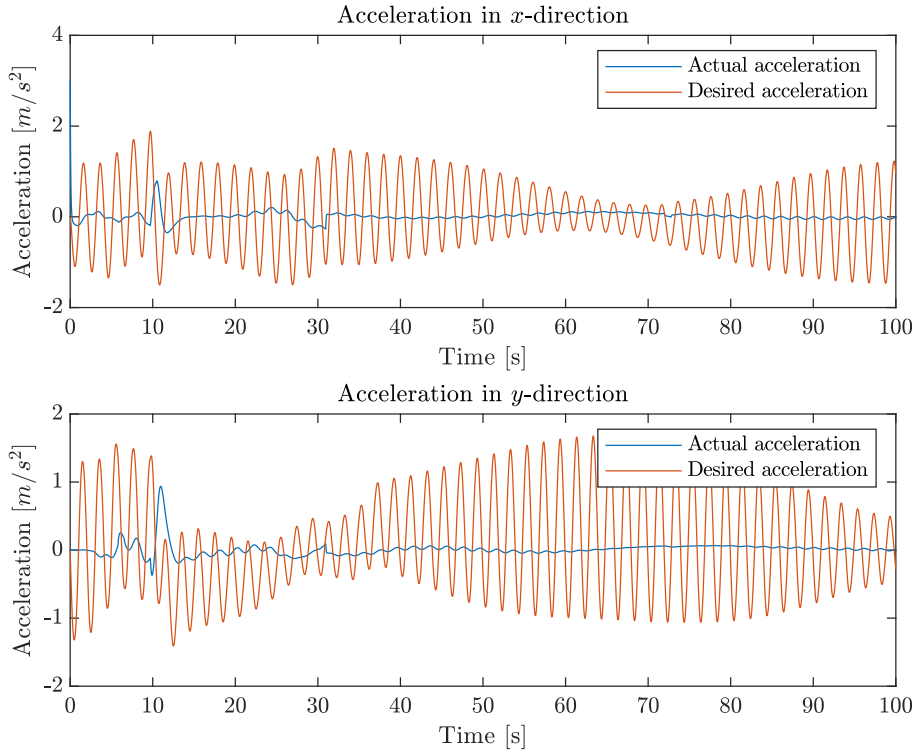


Figure 4.15.: Model 2, test 2 actual and desired x- and y - acceleration

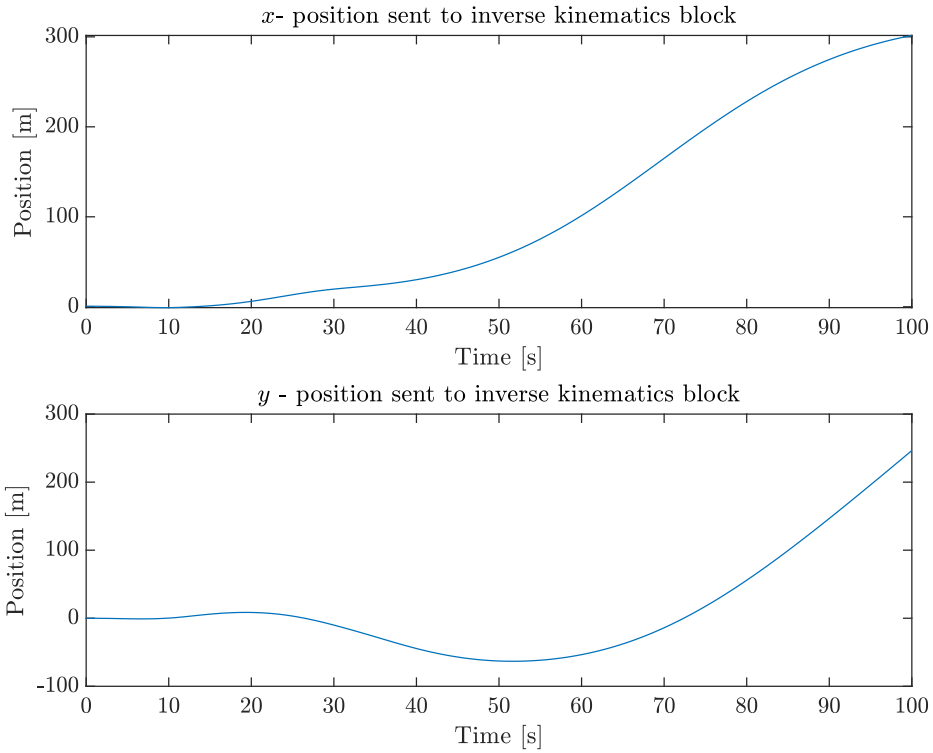


Figure 4.16.: Model 2, test 2 desired trajectory

4.2.4. Model 2 Test 3

The third test is conducted with the same parameters as in table 4.2 but with x_{0d} and y_{0d} set equal to their initial conditions x_{0i} and y_{0i}.

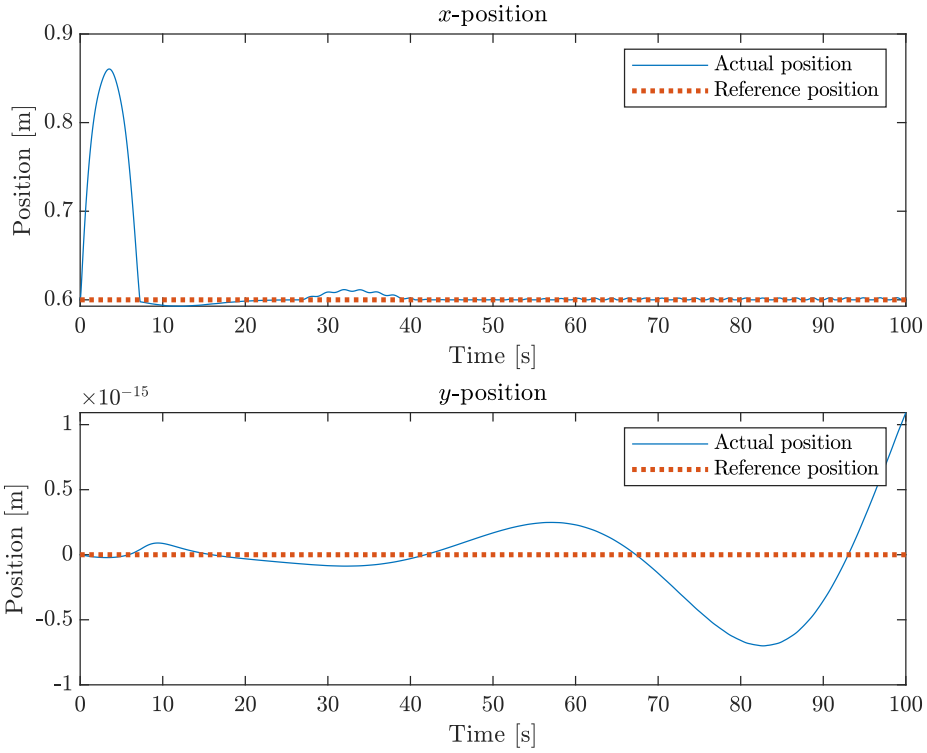


Figure 4.17.: Model 2, test 3 actual and reference x- and y-position of the crane-tip

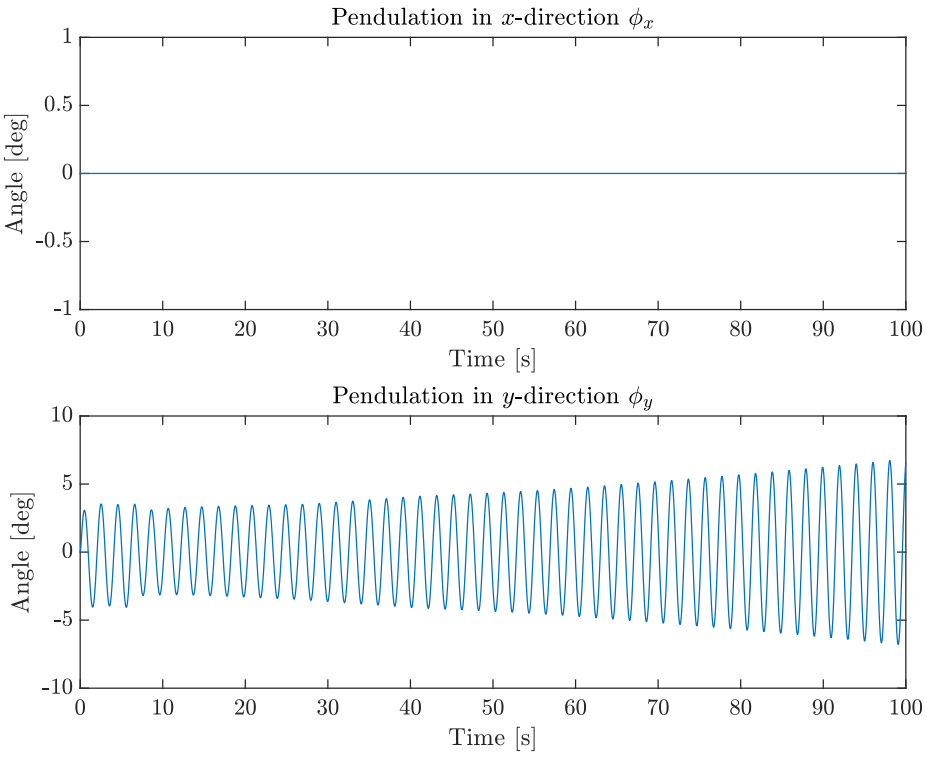


Figure 4.18.: Model 2, test 3 ϕ_x and ϕ_y

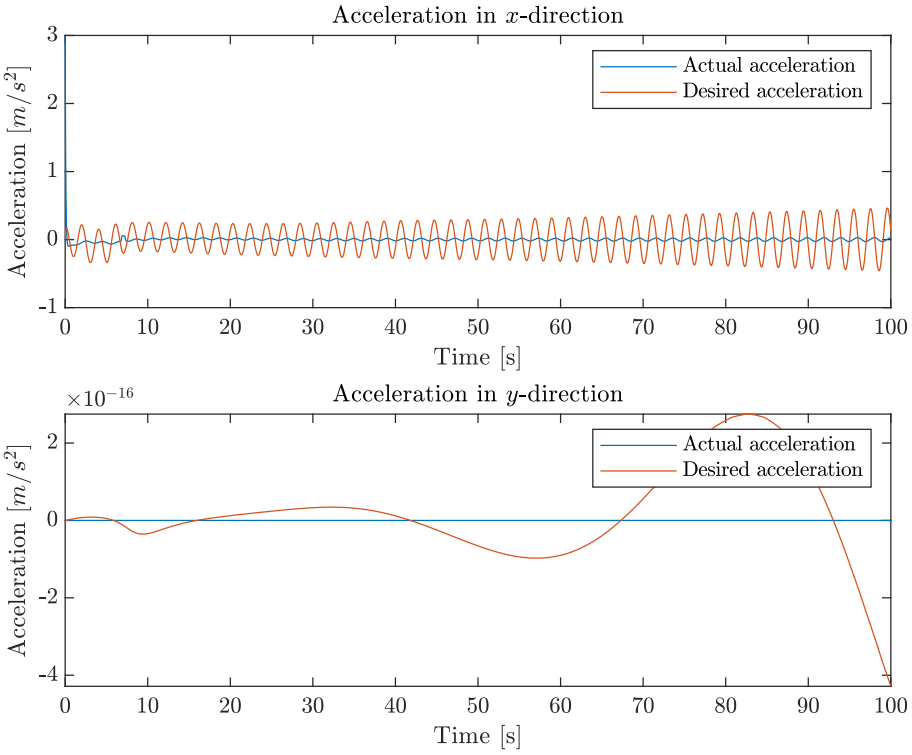


Figure 4.19.: Model 2, test 3 actual and desired x- and y - acceleration

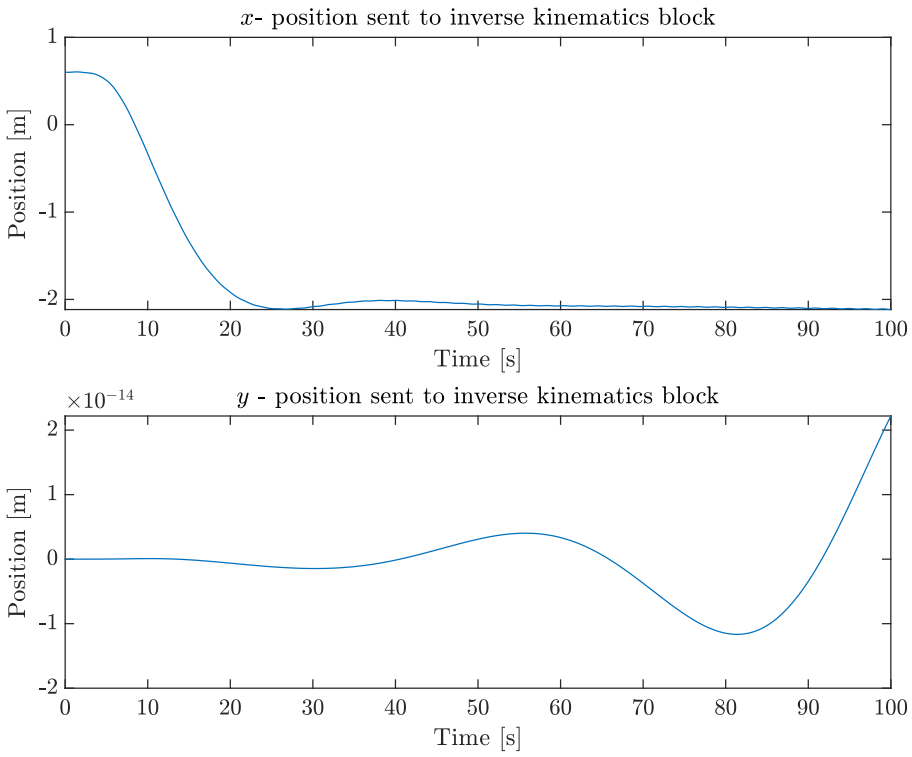


Figure 4.20.: Model 2, test 3 desired trajectory

4.3. Model 3

The third model uses Lyapunov-based damping for the crane load and NMPC for the crane-tip position control. The dynamic and kinematic models are the same as in model 2.

4.3.1. Model 3 Test 1

In test 1 the tuning parameters from table 4.3 are used. In this simulation, all references are set equal to the system's initial conditions. Additionally, this test has no bounds on the manipulated variables $\gamma = 0$. However, the output variables are all bounded using $\gamma = 1$.

Tuning parameter	Value	Unit
T_s	0.1	s
T_c	5	s
T_p	15	s
T_v	200	s
k_{pt}	10	N/A
k_{dt}	15	N/A
k_p	2	N/A
k_d	1	N/A
x_{0i}	0.6	m
y_{0i}	0	m
ϕ_{xi}	0	degrees
ϕ_{yi}	0	degrees

Table 4.3.: Parameters for controller, model 3 test 1

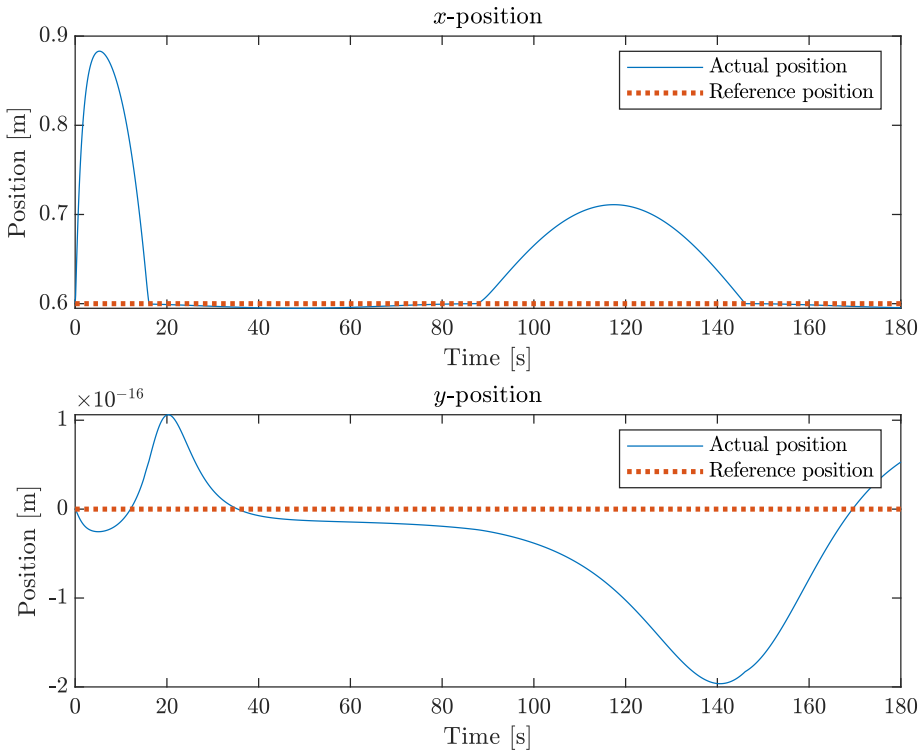


Figure 4.21.: Model 3, test 1 actual and reference x- and y position of the crane-tip

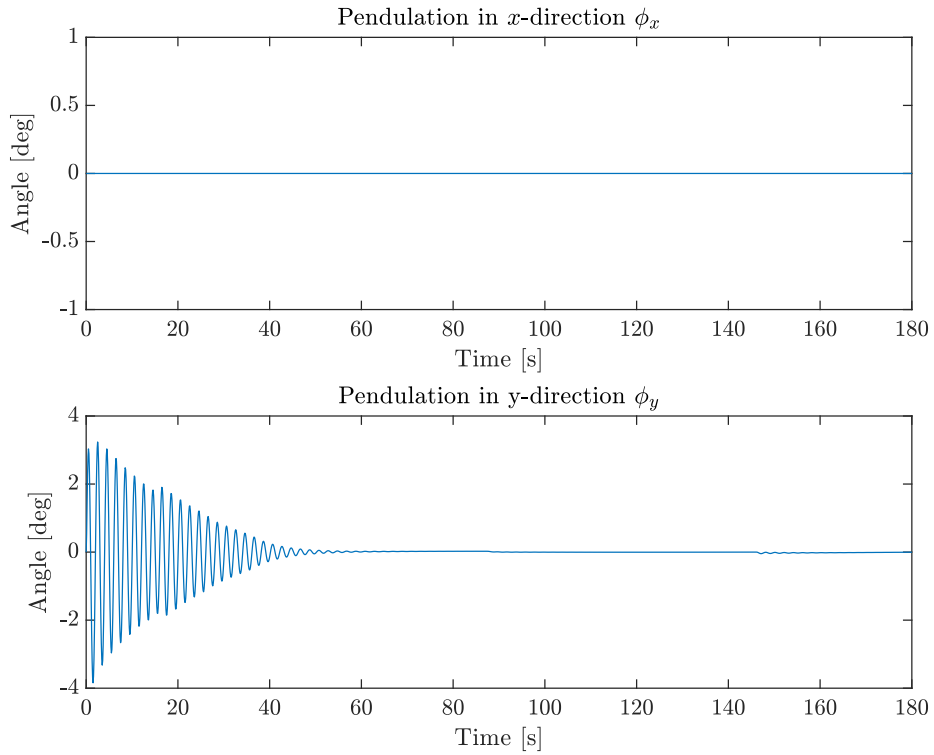


Figure 4.22.: Model 3, test 1, ϕ_x and ϕ_y

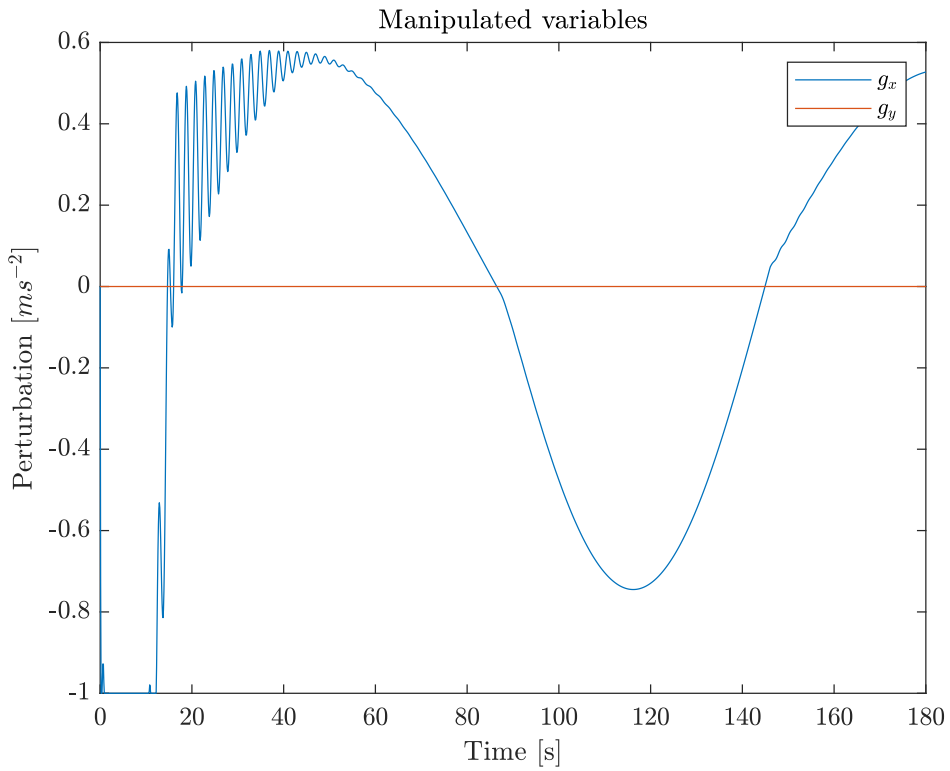


Figure 4.23.: Model 3, test 1 manipulated variables g_x and g_y

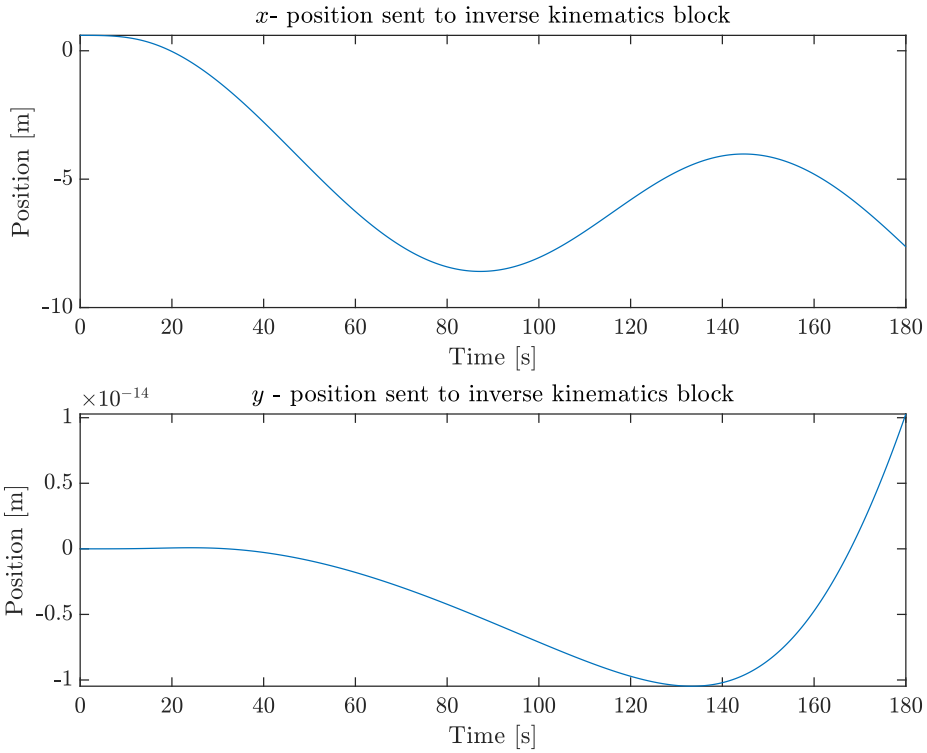


Figure 4.24.: Model 3, test 1 x- and y-position sent into the inverse kinematics block

4.3.2. Model 3 Test 2

The second test uses similar tuning parameters as the first but now with a step in both the x- and y-position reference at the time of 40 seconds. The desired x-position is set to 0.7 m and the desired y-position is set to 0.6 m.

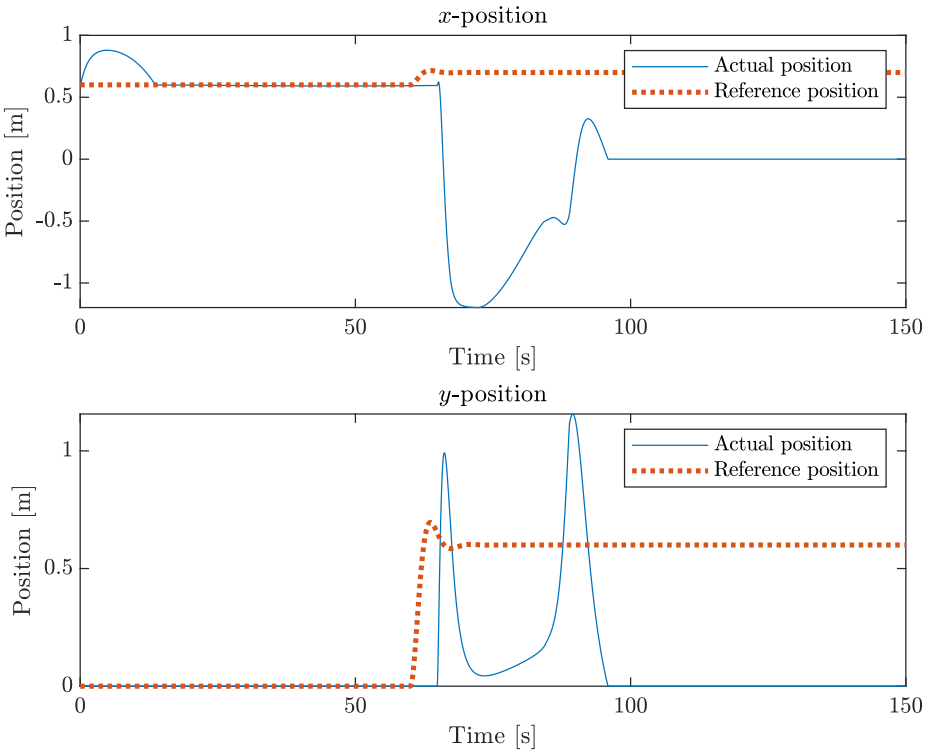


Figure 4.25.: Model 3, test 2 actual and reference x- and y position of the crane-tip

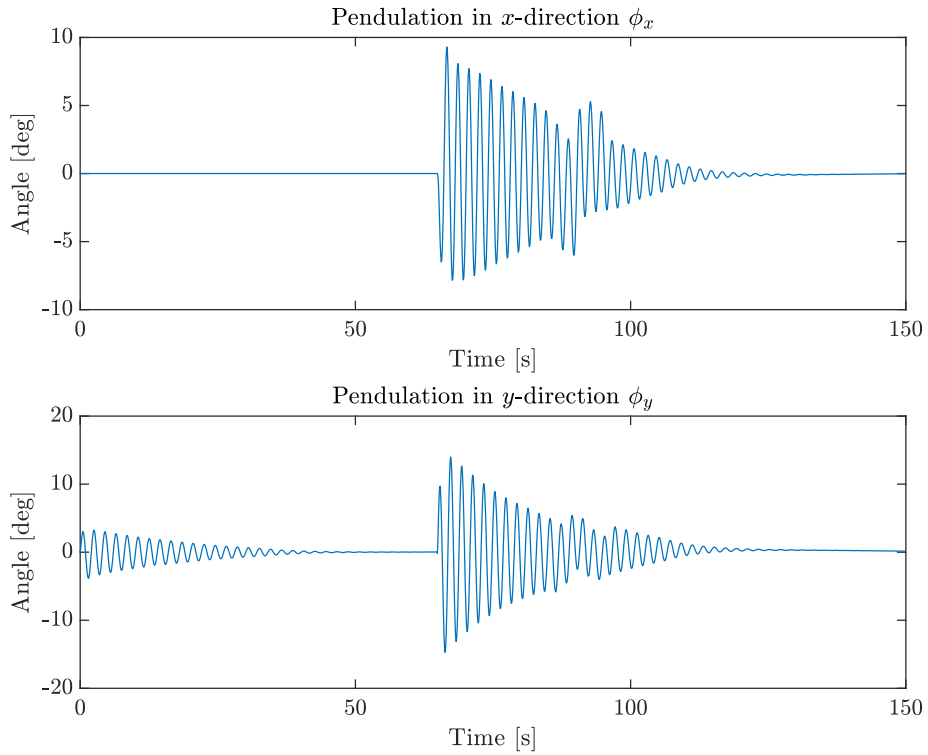


Figure 4.26.: Model 3, test 2 ϕ_x and ϕ_y

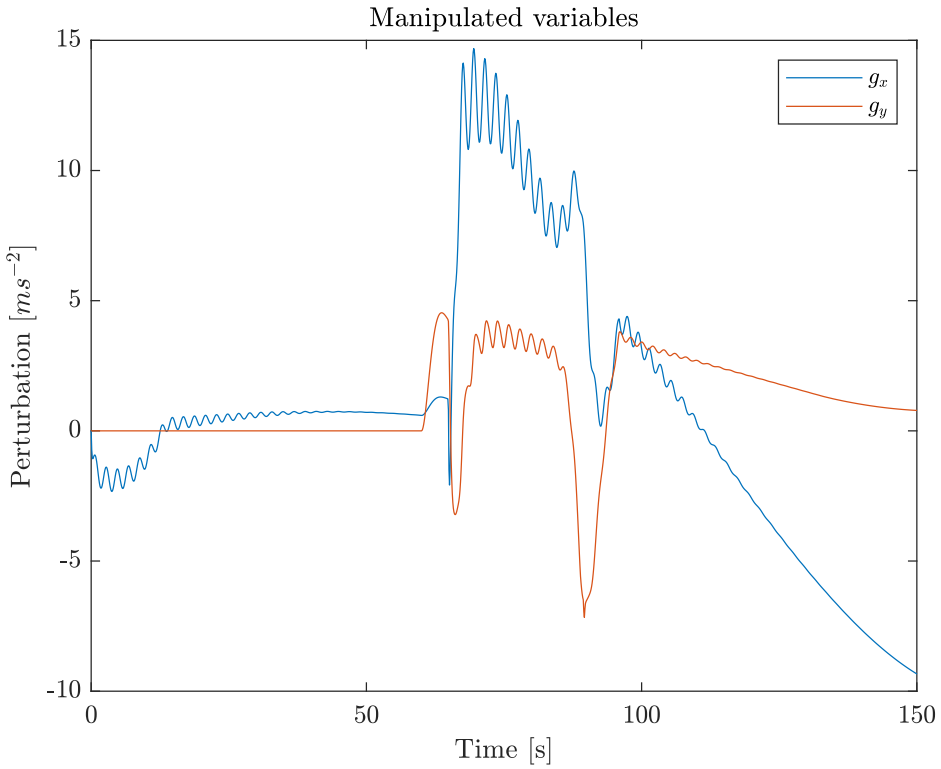


Figure 4.27.: Model 3, test 2 manipulated variables g_x and g_y

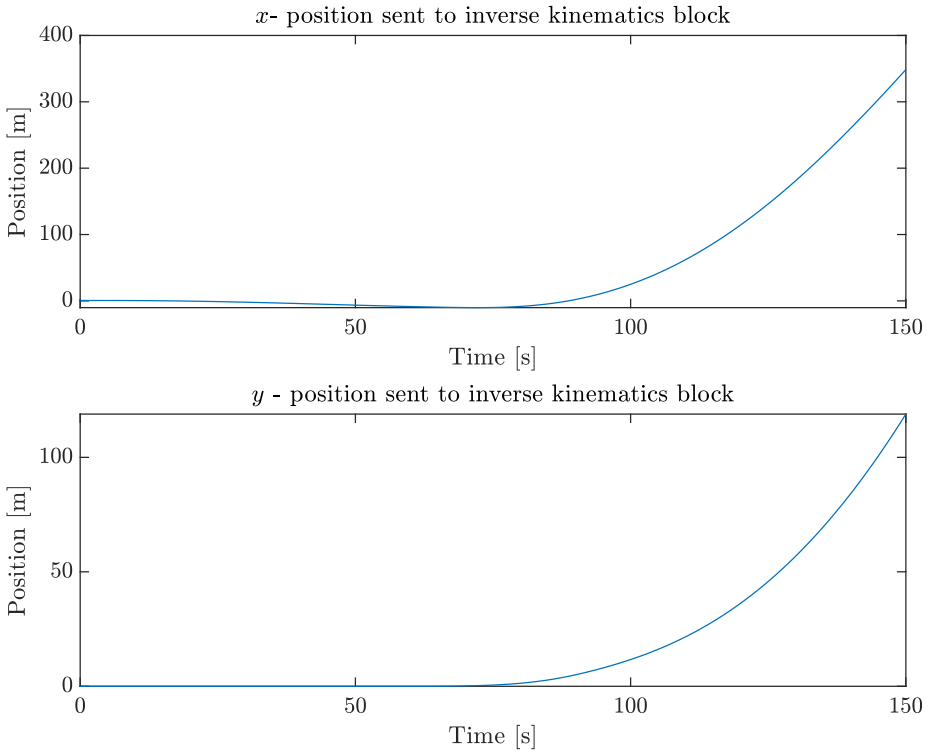


Figure 4.28.: Model 3, test 2 x- and y position sent into the inverse kinematics block

4.3.3. Model 3 Test 3

Test 3 is conducted with the same conditions as test 2 but now with initial conditions for ϕ_x and ϕ_y changed to $-20°$ and $15°$ respectably. There is also set a limit γ equal to 1 on the non-vanishing perturbations g_x and g_y.

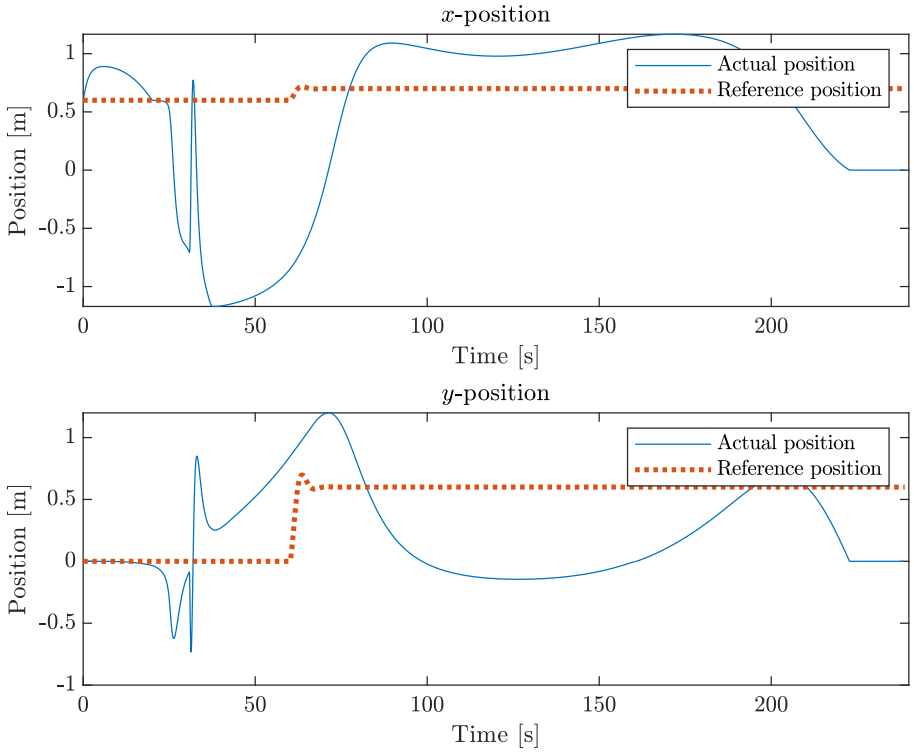


Figure 4.29.: Model 3, test 3 actual and reference x- and y position of the crane-tip

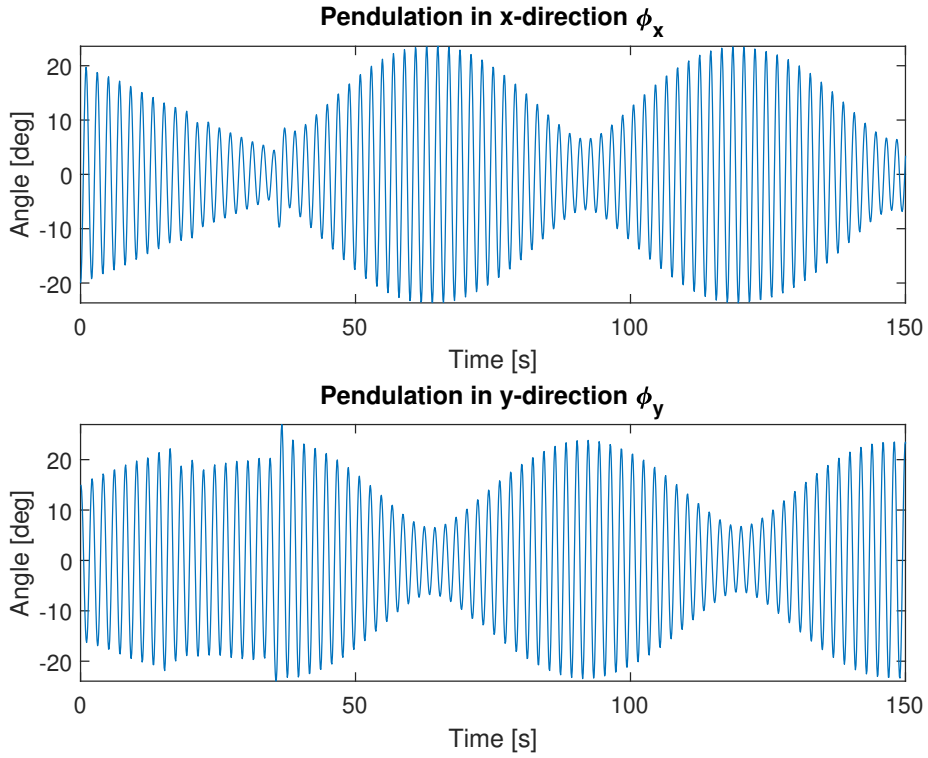


Figure 4.30.: Model 3, test 3 ϕ_x and ϕ_y

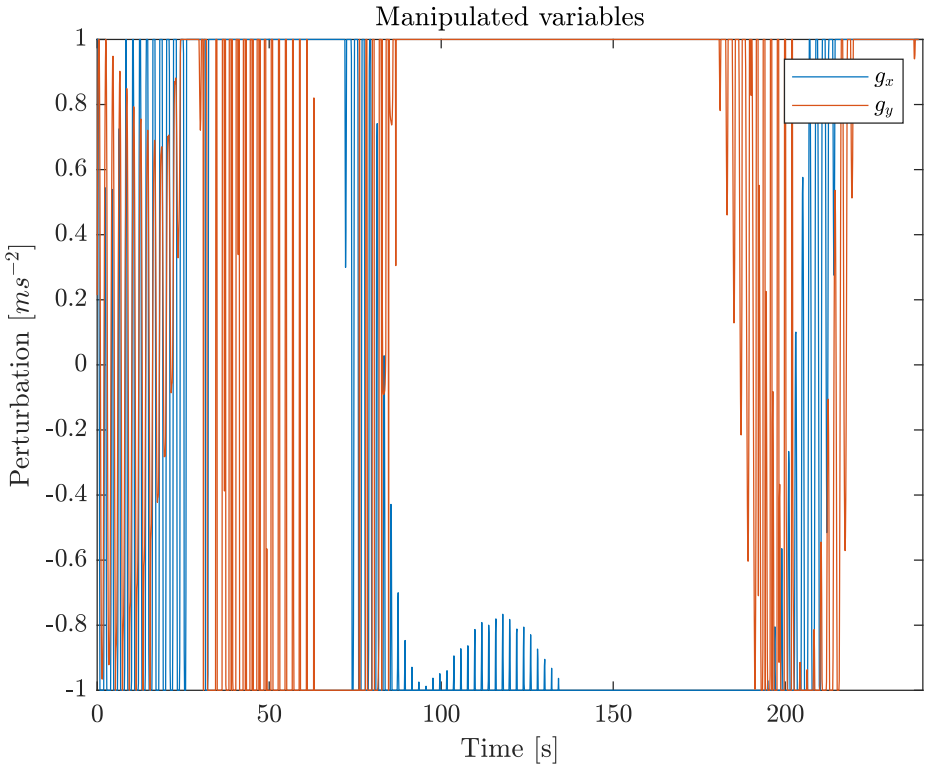


Figure 4.31.: Model 3, test 3 manipulated variables g_x and g_y

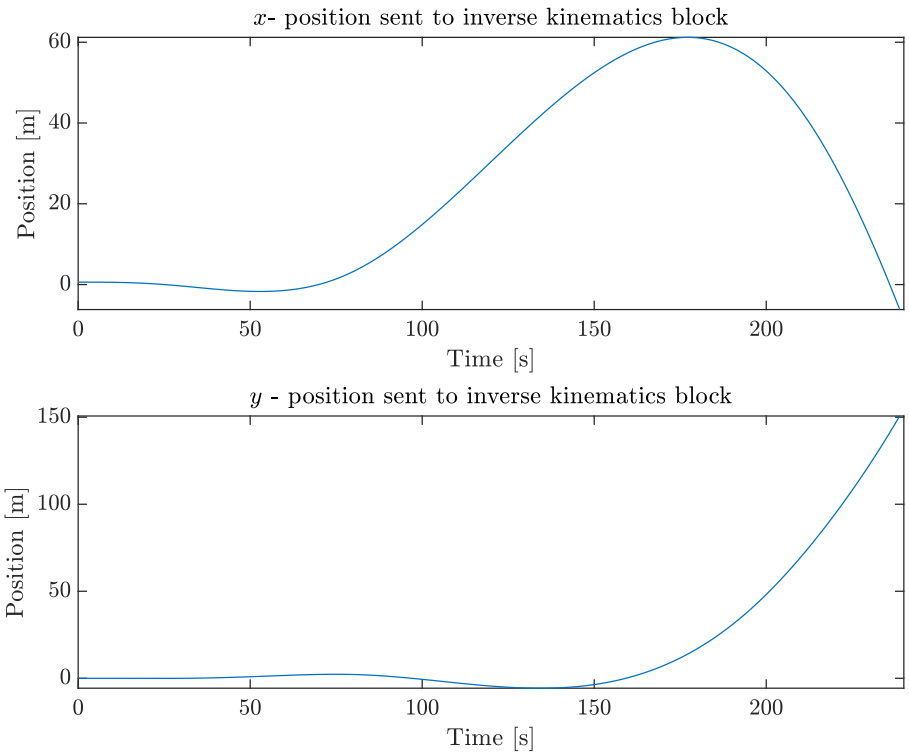


Figure 4.32.: Model 3, test 3 x- and y-position sent into the inverse kinematics block

4.4. Kinematics Test

A final test was conducted using only kinematic models, to rule out any problems with the kinematics. The Simulink model used can be seen in Figure 4.33. A fixed trajectory providing q, $\dot{q}$ and $\ddot{q}$ is given to the crane controller, and the dynamics are then solved. The joint variables are then sent to the forward kinematics which solves the end-effector position, -velocity and acceleration and the inverse kinematic turns these back into joint variables. What we are interested in seeing here is if the joint variables solved by the inverse kinematics match the reference trajectory. The trajectory used is the same as in 4.2.1. Some of the plots show the difference between the reference and final joint variables, this is because the plots were too similar to properly distinguish between them. The controller gains are the same as for models 2 and 3, $k_{pt} = 10$ and $k_{dt} = 15$. The initial conditions of the joint positions q in the dynamic model are here set to $q_0 = [0 \quad 0 \quad 0]$.

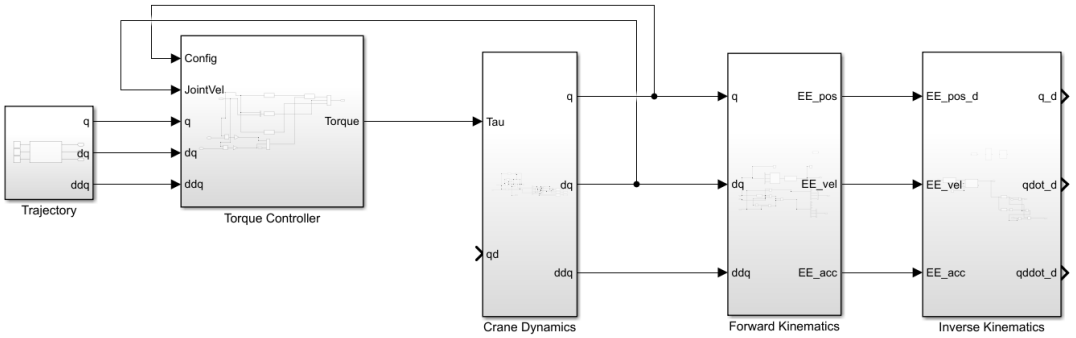


Figure 4.33.: Kinematic test setup in Simulink

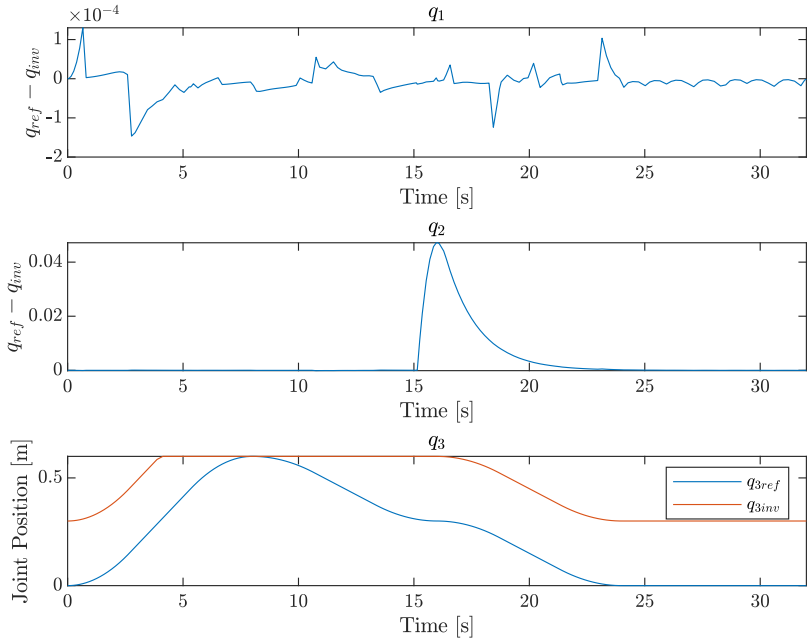


Figure 4.34.: Joint position reference value and joint position solved by inverse kinematic

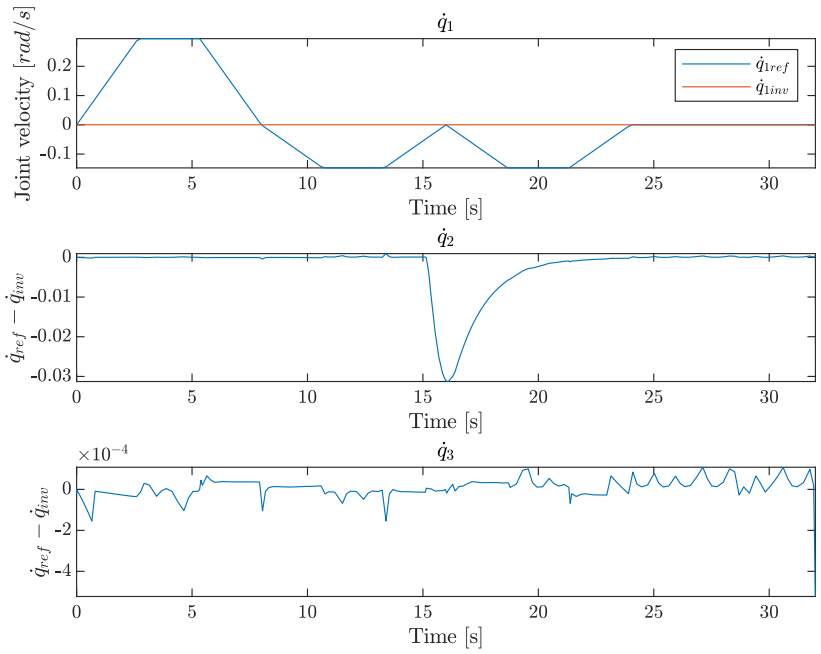


Figure 4.35.: Joint velocity reference value and joint velocity solved by inverse kinematic

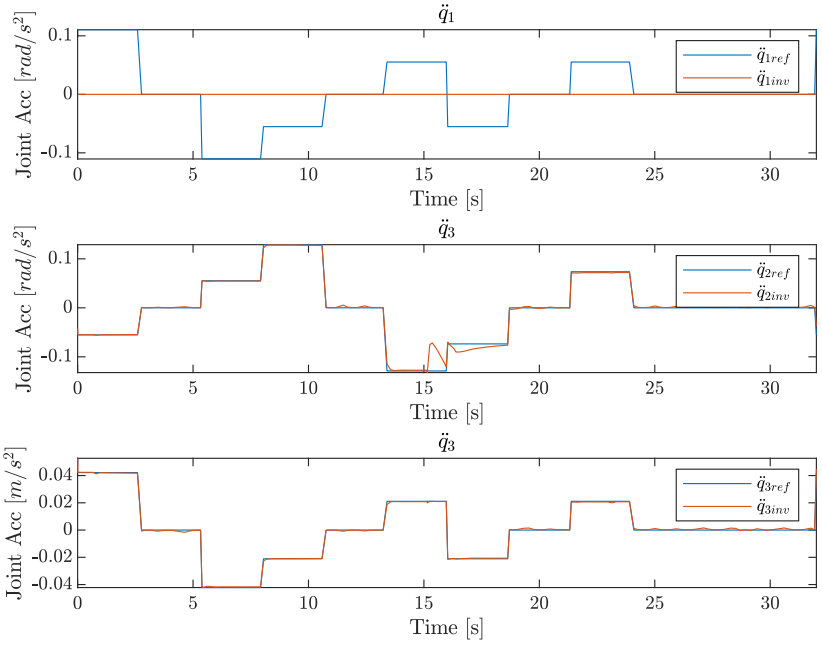


Figure 4.36.: Joint acceleration reference value and joint acceleration solved inverse kinematic

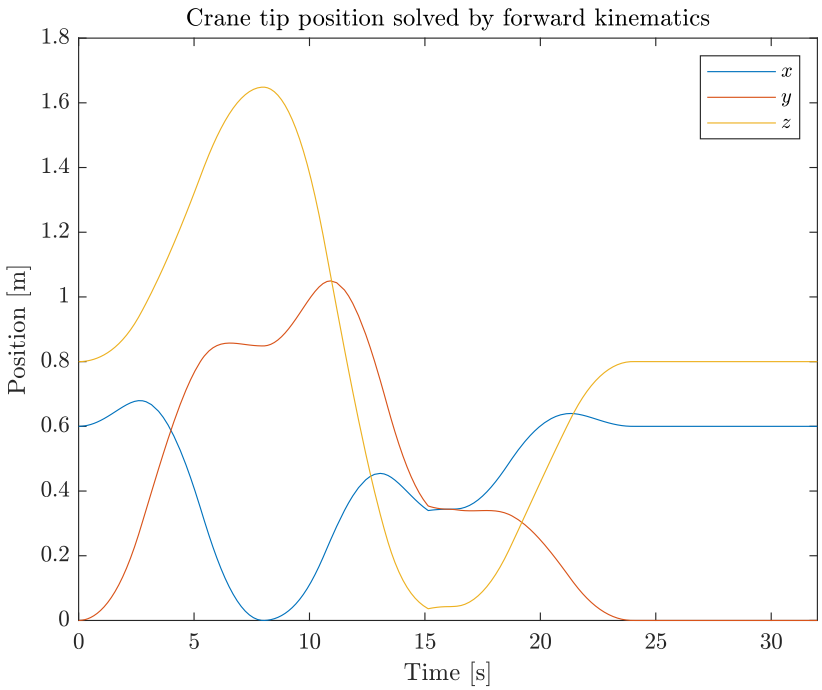


Figure 4.37.: Crane-tip position solved by forward kinematics

Chapter 5.

Analysis and Discussion

In this chapter, the results are analysed and discussed. There are also discussed possible solutions for improving the current models.

5.1. Model Analysis

5.1.1. Analysis Model 1

Two tests were conducted with the first model one with a step in the position and with an initial value of the pendulum angle ϕ_x and ϕ_y set to 0° which can be seen in figure 4.1 to 4.3. And a second test with the same conditions but with the initial value of the pendulum angles ϕ_x and ϕ_y set to $-20°$ and 15° respectively, can be seen in figure 4.4 to 4.6.

As can be seen in the figures 4.1 to 4.6 the controller does meet the requirements set for the controller. It dampens the pendulum oscillations to 0° and manages to put and keep the end effector at the right x- and y-positions. An error in the initial conditions of the x-position can be observed in both figure 4.1 and 4.4. Where the actual crane starts at 0.9 m while the desired position starts at 0.6 m. This deviation is, however, handled well by the controller and in both cases, they quickly meet up at the set-point. One can also observe in figure 4.3 and 4.6 that the desired and actual acceleration follows a similar pattern. Although the actual acceleration does in both cases have a slightly lower amplitude and some more significant deviation spikes which can be observed at roughly time 5 seconds from both figures and for both accelerations.

5.1.2. Analysis Model 2

New Crane Dynamic- and Kinematic Model

The second model has a new dynamic model for the crane and a new controller which implements feedback for both the pendulum damping and position control. Forward kinematic models for finding the end effector position, -velocity and -acceleration, together with inverse kinematic models for finding joint velocity and -acceleration were also added.

As can be seen in figure 4.7 the error between the desired and actual orientation of the crane joints is as small as 10^{-5}. Which makes the difference between the two practically

negligible. Figure 4.8 shows the torque input from the crane controller to the crane dynamic model given in Nm, it can here be observed that the torque given to joint 3 ends up at a value just below 1500 Nm, this is necessary for the actuators of the crane to hold the crane up at a constant height of 0.8 m.

Later tests of the kinematic model as seen in figures 4.34-4.36, show that the inverse kinematic models struggle with solving the joint 3 position and joint 1 velocity and acceleration. Even though joint 1 velocity and acceleration are off the joint 1 position is on target with a small error in the 10^{-4}-range. It can also be observed by the end-effector positions solved by the forward kinematic seen in figure 4.37 that the forward kinematic that finds the crane-tip position uses the correct initial values for the end-effector position.

Controller Model 2

Three tests were conducted on the controller the first one with a step in the x- and y direction in time 30 s. As can be seen in figure 4.9 to 4.12. In figure 4.9 it can be observed that the position in both directions is stable at its initial condition until the step occurs, both directions then take around 40 s to stabilise at 0 m. In figure 4.10 it can be observed that even though the pendulum starts out with an angle of 0° they start to pendulate quite quickly after the step happens at time 30 s and in the case of the y-direction it already has small pendulation from the beginning. After the step, both directions reach a maximum pendulation of about 40°. In figure 4.11 it can be observed that the difference in shape and amplitude of the desired and actual acceleration is great. Figure 4.12 shows that desired position sent into the inverse kinematics block is quite far off from the actual and reference position seen in figure 4.9 seemingly exponentially increasing, instead of settling at a steady value.

Test 2 had many of the same problems as Test 1. The difference between the test is the pendulum is given an initial condition of $-20°$ for ϕ_x and 15° for ϕ_y. As can be seen in figure 4.13 the position will, because of the initial movement of the pendulum, not stay in its own initial conditions. And in the same fashion as test 1 it never reaches its new set point after time 30 s but instead ends up at 0 m. As seen in 4.15, the pendulum angles never go to 0°. However, it never goes above or below 20° either, which is an improvement from Test 1. The acceleration shown in figure 4.15 has similar problems as Test 1 where the amplitude is lower and the general shape of the actual acceleration is off from the desired acceleration. Similarly, the desired position shown in figure 4.16 is far from what is represented in figure 4.13, and exponentially increasing.

A third test was made to see how the controller reacts when there are no changes in the crane-tip position and the pendulum angles' initial conditions are set to 0°. Figure 4.17 shows that in these conditions the position remains mostly on the mark, with a 0.25 m deviation in the first 10 s in the x-position. A similar deviation at the beginning of the simulation in the x-position can be seen in figure 4.9. The y-position, however, is on the mark with some minor deviation at the end. These deviations are, however, so small that they are considered negligible. In figure 4.18 it is shown that the ϕ_x angle is steady at 0°, the ϕ_y angle slowly grows more unstable reaching a max amplitude of 6° at the end of the simulation. The desired and actual acceleration seen in 4.19 does not match, for the y-acceleration this is not an issue considering how small the desired acceleration is. For the actual x-acceleration, it is possible to see some very small oscillations resembling the

one in the desired acceleration. However, it is still far off from the desired acceleration. Note the desired position in 4.20 is more realistic to the actual movements of the crane where the x-position still does not match. The y-position is a close match in shape, the desired position, however, has an amplitude in the 10^{-14}-range while the actual position is in the 10^{-15}-range. These values are so small that they both can be approximated to zero.

5.1.3. Analysis Model 3

The third model uses the same dynamic and kinematic models as model 2. The controller is however changed to use Lyapunov-based pendulum dampening and nonlinear MPC position control. Three tests were conducted on the model.

The first test was conducted with the initial value of the pendulum angle set to 0° and with no change in position. The goal of this test was to see if the controller could hold the crane and pendulum steady with no change in the system. As can be seen in figure 4.21 and 4.22 the x-position has some initial movement but eventually settles at its initial value, and yet again at time 90 s to 145 s it deviates but goes back to its initial value. The y-position keeps itself steady at 0 m with only very small deviations in the 10^{-16}-range, which are negligible. The sudden movement in the x-position at the time 0 s leads to a small pendulation in ϕ_y, this does eventually go the zero after about 50 s and is only a minor deviation with a maximum amplitude of ±4°. In figure 4.23 we can see the manipulated variables of the system. It can be observed that there is only movement in g_x. Figure 4.24 the positions sent into the inverse kinematic block can be seen. The shape and magnitude for the x-position do not represent what can be seen in figure 4.21. Here the deviations in the y-position are also negligible as it is in the 10^{-14}-range, which is greater than the actual position but still negligible.

Test 2 was conducted with the same parameters as test 1, but in addition, a step in x- and y-position was made. In figure 4.25 and 4.26 the end-effector position and pendulum angles can be observed. The position holds to its initial value until the step is made at time 60 s then both position eventually settles at 0 m at about time 95 s. This is not the desired position. The x-position experiences an initial bump at time 0 s, which results in a pendulation in ϕ_y, the pendulum also starts oscillating in both directions after the position is changed at time 60 s. All three oscillations of the pendulum eventually die out. In figure 4.27 one can see that both manipulated variables are in movement and neither g_x or g_y are settling at zero. It can also be observed in figure 4.28 that the position sent into the inverse kinematics block steadily increases to over 340 m for the x-position and 120 m for the y-position, similar to the exponential rise in the desired position as seen in model 2.

The third test was done with initial conditions in ϕ_x and ϕ_y set respectively to −20° and 15° and with a limit on manipulated variables $\gamma = 1$. In figure 4.29 we can see a similar trend as test 2. The x-position only briefly settles at its initial value, and it has the same initial bump as can be seen in figure 4.21 and 4.25. The y-position spends around 20 s at its initial position before it also starts to deviate. Both positions reach their maximum reach of ±1.2 m before settling at 0 m as test 2 did. Figure 4.30 shows that the oscillation of the pendulum alternates several times between growing worse and dampening down,

but never settling at 0°. The maximum amplitude of either pendulation direction does not exceed ±25°. The manipulated variable, seen in figure 4.31 quickly saturates at the limit that's been set for it and never goes to zero. Figure 4.32 shows the same problem as test 2 where the position grows far greater than it should.

5.2. Discussion and Model Comparison

Model 1 worked well and managed to achieve all the control goals. The reason this model was not built further on is that the crane tip position controller uses feedforward instead of feedback control. It was thought that this type of position control would not be good enough for an actual shipboard crane and that feedback from the actual crane-tip position would be essential for good control of the crane tip. Since the input of the controller is the crane tip acceleration created using crane loads angle rate and then integrated twice to make the x- and y-positions used in the controller. The crane controller does not have any reference of where the crane tip is and only knows for certain where we want it to be. Especially if there are heavy disturbances from wind and waves further distorting the measurements the feedforward controller can be vulnerable. With vulnerable it is meant that the crane will struggle to end up at the reference position and start deviating. Achieving feedback in the following controllers, therefore, became a major priority as can be seen in both models 2 and 3.

Models 2 and 3 encountered many of the same problems. The discussion of these problems is therefore merged where it is appropriate. For model 2 test 3 and model 3 test 1 both managed to keep the position steady when there were no steps in position and the initial condition for the pendulum is 0°. Both did, however, perform quite poorly with steps in the crane tip position and non-zero initial conditions in the crane load. Model 3 did more consistently manage to dampen the pendulum oscillations, only not managing it in the third test where the MV's were bounded by $\gamma \pm 1$, and the initial condition for the pendulum was higher.

An anomaly can be seen in both models 2 and 3 where there are steps in the crane tip position. In these cases, the position never reaches the new set-point or the old initial position but rather settles at 0 m. The desired position in these cases all grows to very high values, which are impossible for the actual crane to reach. Looking at the desired positions one would assume these ended up at one of the limits ±1.2 m. There are a few theories as to why these errors happen.

The most likely cause is as stated above, the desired position never resembles the actual or reference position. A cause for this can be that to create this position input we have to integrate values three times as can be seen in figure 3.8. When integrating oscillations that do not oscillate around zero, the integration can cause an exponential increase. The velocity created here has feedback which can stabilise the integration, the position however falls outside this feedback loop. Without feedback, the integration can be allowed to increase indefinitely under the right conditions. A solution to this may be to not use position in the control but rather use the velocity and acceleration together or alone to control the crane. The actual crane has a form of velocity control this was, however,

discovered too late to implement[1]. One could also try to formulate these equations to command the position instead of the velocity. As mentioned by Tysse in his PhD thesis the crane-tip acceleration cannot in reality be used as an input [9], using the acceleration will therefore only be a solution that works in simulations.

Another theory is that errors are collected throughout the model. Each subsystem has an error that for the subsystem itself may not be too significant. It can, however, become quite significant when all subsystems have a specific error or if one block has a large error, these errors can then propagate throughout the system. A solution for this problem could be better fine-tune every part of the model. Using an error measuring tool like RMS-error, like in the project report [5], could help to uncover where the errors originate. This is useful for finding more suitable gains. By process of elimination, one could look at the results in 4.2.1 and see that the error most likely does not occur in the crane dynamic model, this could also be said of the dynamic model for pendulum as it works quite well for model 1 and in the project report [5]. This leaves us with the kinematic models and controllers as the most likely sources for such errors. Which aligns quite well with the first theory of integration difficulties. Problems have been discovered with the initial condition of joint 3 and the x-position. As can be seen in figure 4.1 and 4.4 and see that the initial condition here is set to 0.9 m whereas, in the subsequent models, it is set to 0.6 m. The reason for this is that when the initial condition where set to 0.9 m in the controller the initial condition of the actual position went up by around 0.3 m. This discrepancy is one of the reasons a new crane model was tried. A residual of this can be seen in the x-position in the subsequent models, especially in model 2 test 3 and model 3 test 1 where the position seems to try to go up before settling at 0.6 m. As is seen in section 4.4 these problems most likely originate in the inverse kinematics which sets a different initial condition for joint 3 than the dynamic model.

Together with the initial condition problem, a problem with achieving feedback control is a reason for making a new dynamic model for the crane. The Simscape model from Espen Nilsen's thesis [4] had a problem where algebraic loops would occur when the calculated crane-tip position and -velocity was used in a feedback controller. This both made the control quite bad together with simulation time going up drastically. It was tried using unit delays to fix the algebraic loops. The "Unit delay"-block delays the input by the sample time. This means you use the measurement one time step back instead of the current one. This did rid the system of algebraic loops but did not improve the control or the simulation time. Algebraic loops are usually a symptom that something is wrong in the model, and since it proved difficult to find these issues for the Simscape crane a new model was seen as the simplest solution to make a new dynamic model.

The new crane model works quite well when the trajectory is predefined with joint position, -velocity and -acceleration provided to the torque controller. Simulations of the kinematic models and the dynamic model showed that some of the joint angles and velocities were not solved correctly. As can be seen in figure 4.34 -4.36, q_3, $\dot{q}_1$ and $\ddot{q}_1$ deviates from the reference value to an unacceptable degree. The other joint variables have smaller sometimes negligible deviations from the reference joint variables. In figure 4.34 the reason for the deviation in joint 3 seems to be that the initial condition of the inverse kinematic solver for the joint positions sets for q_3 is different from the one set by the dynamic model

[1]Conversation with Linn Danielsen Evjemo, a researcher for Sintef Ocean

and the trajectory. The inverse kinematics sets the initial condition to 0.3 m, while the crane model, sets it to 0 m. It also flattens out after time 4 s to 17 s because 0.6 is the joint limit. Other than the difference in the initial condition and that it saturates, joint 3 seems to have a similar shape to the reference trajectory. The velocity and acceleration of joint 1 are also inaccurate according to the reference the position of joint 1, however, still remains very close to the reference. While the velocity and acceleration for joint 3 are all close on the reference but the position of joint 3 is off. The reason for the deviations in the first joint can have the same origin as the deviation in joint 3. This fault can, however, also have its origin in the 3×3 Jacobian inverse which is used for both velocity and acceleration inverse kinematics. As mentioned in the theory, the only time the manipulator Jacobian is naturally invertible is when the robot has six joints, which is why it is reduced to a 3×3 matrix in this case. There are other solutions for inverting such matrices such as the Moore-Penrose pseudoinverse [10], however, reducing the Jacobian seemed like the more computationally effective approach. To make the time derivative of the Jacobian necessary for the forward and inverse kinematics for the joint- and end-effector acceleration the "Derivative"-block in Simulink is used. Numerical derivation can amplify noise in the signals, it could therefore be beneficial for the system to solve $\dot{J}(q)$ analytically and then make a function that solves it directly without the need for numerical derivation. This could improve both the crane-tip and joint acceleration.

5.3. Possible Improvements

The suggested improvement for model 1 is to add feedback to the crane tip position control, which is why models 2 and 3 were made. Therefore this section will mainly focus on the two latter models. Both these models suffer from similar problems.

One of the first improvements that should be attempted is to get the velocity input to work correctly. This can be attempted by changing the crane controller to only take in velocity as input and providing the joint variables to the Jacobian by feedforward from the crane dynamic model. Removing the need for integrating this input more than necessary. Another possible solution is to try and command the position instead of the velocity. However, since the actual crane has a type of velocity control, the first solution might be preferred.

A second improvement that can be made in improving the inverse kinematic solution. Preferably making an inverse kinematic solver that is independent of the rigid body tree. So that the joint 3 position and joint 1 velocity and acceleration are solved correctly. A good starting point here could be to alter the rigid body tree "modelcodegen.m" and try to change the initial conditions here. Finding another way to solve the inverse Jacobian, and analytically solving the time derivative of the Jacobian removing the need for numerical derivation could also as mentioned improve the inverse kinematics solutions.

If none of these is the main cause of the problems looking at the controllers themselves can be of use. And alternatively, design a whole new controller that employs feedback for both the crane load and crane tip position. Feedforward works well for model 1 possibly adding some feedforward control to the scheme proposed in figure 3.1 can better the controller already proposed. One can also look into controlling the z-direction, as of now the crane load model is underactuated as we only control the x and y-direction. Adding control to

the z-direction will add another element of control that can be beneficial. Adding control to the z-position of the crane-tip will eventually be necessary anyway to be able to lower the crane load.

Even though the current crane does not have variable cable length, this can be retrofitted to the crane, adding another controlled variable that can benefit the control system. The control system used for model 2 was initially designed for a crane with variable cable length making this an ideal start candidate for such control.

Chapter 6.

Conclusion and Future Work

6.1. Conclusion

Three main goals was set for the thesis. The first was to expand the model from the project report [5] to include crane motion. This was done using two models one made with Simscape and the other using differential equations. Both models worked as intended when attached to the crane load model, however, the Simscape model had problems implementing feedback making the differential equation approach the preferred option. The forward kinematic model worked as intended, the inverse kinematic model experienced some problems specifically in solving the position for joint 3 and the velocity and acceleration for joint 1.

Three controllers were proposed and the control goals set was to achieve simultaneous control of the crane-tip position and crane load damping. The first model achieved the control goals, however, because of the use of feedforward in the crane-tip control instead of feedback it was deemed unsuitable for shipboard crane control. Neither the second nor third model managed to achieve the control goals with model 3 performing somewhat better in the crane load damping control. Both these models used feedback control with the second model using cascade control where the inner loop controlled crane load damping and the outer loop the crane tip position. The third model used nonlinear MPC for the position control and Lypaunov-based damping for the crane load. None of the stretch goals was attempted in this thesis, the main goals were all achieved with varying results.

To answer the research question "Can a satisfactory controller be designed that dampens the crane load oscillations simultaneously with the crane tip position on shipboard cranes?" the question cannot be satisfyingly answered in this thesis. Model 1 did, however, achieve all the control goals and cannot be fully discounted for use on shipboard cranes without being tested for it. Models 2 and 3 might work if the suggested improvements were to be done. However, none have been tested with the necessary disturbances, and if the controller does not work without disturbances and with perfect measurements, it will not work with the disturbances added.

6.2. Future Work

There is still a lot of work that needs to be made to make a satisfying control system for the crane. This goes for both theoretical work and practical work on the crane itself.

A working control system with feedback for both the pendulum dampening and the crane tip position controller should be designed. Possible improvements for the models used in this thesis are described in chapter 5.3. It can also be desirable to try out new ideas for the control system.

With a working control system, work should be done on testing it with wave and wind disturbances. Preferably the crane should be attached to a ship model so that wave disturbances can be induced from the ship into the crane. The wind disturbance should mainly affect the crane load. Testing this model with simulations should then be done to verify if the controller works in these conditions. At this stage, state estimation can be employed for the estimation system states.

With a control system that can handle the wave and wind disturbances, the model should be tried with a sensor system like the one in [4]. It could also be worth simulating the crane dynamics with hydraulic actuators as the actual crane has. A controller for the z-position for lowering the load when at the correct x- and y-position will also have to be designed.

Sintef Ocean has plans to have the option of remote controlling the crane, which is located in Frøya, from Trondheim. This will introduce transport delays in the system. the control system will have to be fast enough to handle these as well. Some tests of the control system from the project report [5] were done with both transport delay and mechanical delay added. When all needed models and simulations are done the crane can be retrofitted with a new control and sensor system and testing of the real crane can begin.

Bibliography

[1] GE Oil Gas. *The Impact of Digital on Unplanned Downtime*. Available at https://www.ge.com/digital/sites/default/files/download_assets/ge-the-impact-of-digital-on-unplanned-downtime.pdf (2022/12/14). 2016.

[2] Liyana Ramli, Zaharuddin Mohamed, Auwalu M Abdullahi, Hazriq Izzuan Jaafar, and Izzuddin M Lazim. "Control strategies for crane systems: A comprehensive review". In: *Mechanical Systems and Signal Processing* 95 (2017), pp. 1–23.

[3] Yuchi Cao and Tieshan Li. "Review of antiswing control of shipboard cranes". In: *IEEE/CAA Journal of Automatica Sinica* 7.2 (2020), pp. 346–354.

[4] Espen M. Nilsen. "Crane Pose Estimation using IMU and Computer Vision". MA thesis. Trondheim No., 2022.

[5] Even Rise Gregersen. *Crane Load Pendulation Control*. 2022.

[6] Eihab M Abdel-Rahman, Ali H Nayfeh, and Ziyad N Masoud. "Dynamics and control of cranes: A review". In: *Journal of Vibration and control* 9.7 (2003), pp. 863–908.

[7] MD Todd, ST Vohra, and F Leban. "Dynamical measurements of ship crane load pendulation". In: *Oceans' 97. MTS/IEEE Conference Proceedings*. Vol. 2. IEEE. 1997, pp. 1230–1236.

[8] Olav Egeland. *Crane-Load Dynamics and Control. Working Note*. 2022.

[9] Geir O. Tysse. "Modelling and Control of Ship-mounted Cranes". PhD thesis. Trondheim, No, 2020.

[10] Kevin M Lynch and Frank C Park. *Modern robotics*. Cambridge University Press, 2017.

[11] Lorenzo Sciavicco and Bruno Siciliano. *Modelling and control of robot manipulators*. Springer Science & Business Media, 2001.

[12] Jorge J Moré. "The Levenberg-Marquardt algorithm: implementation and theory". In: *Numerical Analysis: Proceedings of the Biennial Conference Held at Dundee, June 28–July 1, 1977*. Springer. 2006, pp. 105–116.

[13] Dale E Seborg, Thomas F Edgar, Duncan A Mellichamp, and Francis J Doyle III. *Process dynamics and control*. John Wiley & Sons, 2016.

[14] Sigurd Skogestad and Ian Postlethwaite. *Multivariable feedback control: analysis and design*. john Wiley & sons, 2005.

[15] Michael A Johnson and Mohammad H Moradi. *PID control*. Springer, 2005.

[16] Hassan Khalil. *Nonlinear Systems*. Pearson Education, 2002.

[17] Rolf Findeisen and Frank Allgöwer. "An introduction to nonlinear model predictive control". In: *21st Benelux meeting on systems and control*. Vol. 11. Veldhoven. 2002, pp. 119–141.

[18] MathWorks. *Rigid Body Tree Robot Model*. Available at https://se.mathworks.com/help/robotics/ug/rigid-body-tree-robot-model.html (2023/06/16).

[19] Geir Ole Tysse, Andrej Cibicik, and Olav Egeland. "Vision-Based Control of a Knuckle Boom Crane With Online Cable Length Estimation". In: *IEEE/ASME Transactions on Mechatronics* 26.1 (2021), pp. 416–426. DOI: 10.1109/TMECH.2020.3024637.

[20] Geir Ole Tysse and Olav Egeland. "Crane load position control using Lyapunov-based pendulum damping and nonlinear MPC position control". In: *2019 18th European Control Conference (ECC)*. 2019, pp. 1628–1635. DOI: 10.23919/ECC.2019.8796215.

[21] MathWorks. *Nonlinear MPC Controller*. Available at https://se.mathworks.com/help/mpc/ref/nonlinearmpccontroller.html (2023/06/23).

[22] MathWorks. *Specify Prediction Model for Nonlinear MPC*. Available at https://se.mathworks.com/help/mpc/ug/specify-prediction-model-for-nonlinear-mpc.html (2023/06/23).

Appendix A.

Digital Attachments

A zip file is provided with digital attachments this folder contains the models, scripts and some of the literature used in the thesis. It should be noted that for models 2 and 3, and the dynamic and kinematic model the crane folder needs to be added to the path. The rigid body tree "modelcodegen.m" should also be added to the path for all models. The list below describes all the contents of the different folders in the attachment.

- Literature

  Crane - Olav Egeland Olav Egeland's working note "Crane-Load Dynamics and Control" [8], provided here with Egeland's permission.

  Project report Even Gregersen (the author's) project report "Crane Load Pendulation Control" [5], provided here since it is not published.

- Model 1

  ScriptModel1.m Script containing parameters and variables for the controller.

  constants.m constants for the crane model provided by Espen Nilsen's thesis [4].

  IMUFunctionGeneration.m Script initialising the IMUs in the crane model, is necessary for the crane model to work properly. This is also provided by Espen Nilsen.

  modelcodegen.m The rigid body tree, used for the inverse kinematics block, is also provided by Espen Nilsen's thesis. This also needs to be on the path for the succeeding models.

  ForwardKinematics.m Forward kinematics solved using product of exponential formula, the function is created in the "IMUFunctionGeneration.m" script. This also needs to be on the path for the succeeding models.

  crane.slx Simulink model of the crane, pendulum and controller.

  readmeMod1.txt Text file containing correct run order for the scripts.

- Model 2

  ScriptModel2.m Script containing the controller parameters and variables.

  Model2.slx Simulink model of the crane, pendulum and controller.

- Model 3

  ScriptNMPC.m Script containing the controller parameters and variables.

  NonLinearObject.m Nonlinear object used by the NMPC block in Simulink.

  ninLinMPC.m State function used by the nonlinear object.

  outputfunc.m Output function used by the nonlinear object.

  JacobianStateFunc.m Jacobian of the state function use by the nonlinear object.

  Model3_NMPC.slx Simulink model of the crane, pendulum and controller.

  readmeMod3.txt Text file containing the correct load order of the scripts and model.

- Crane

  c_matrix.m Calculates the $C(q, \dot{q})\dot{q}$-matrix for the crane dynamic model and controller.

  ForwardKinSpace.m Alternative Forward Kinematic solution, also based on the product of exponentials formula.

  gravityMat.m Calculates the $G(q)$-matrix for the crane dynamic models and controller.

  jacobianSpace.m Calculates the space Jacobian for the velocity and acceleration kinematics.

  jacobianSpaceThreebyThree Function solving the 3×3 space Jacobian for the inverse kinematics.

  jacobianSpaceInverse.m Function calculating the inverse of the 3×3 Space Jacobian.

  M_inverse.m Function solving the inverse of the $M(q)$-matrix for the crane dynamics.

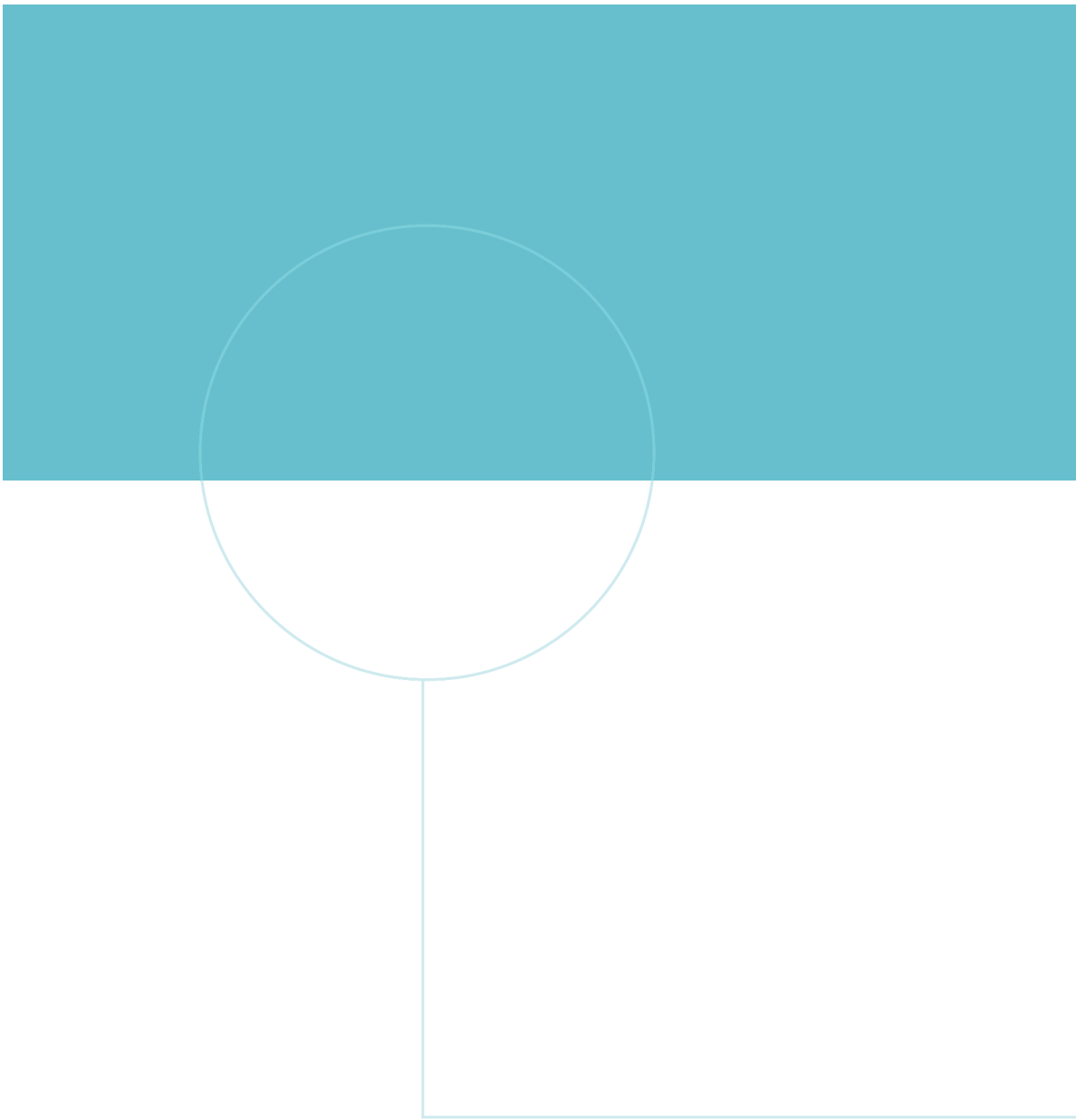
  M_matrix Calculating the $M(q)$-matrix for the crane controller.

- Dynamic and Kinematic Model

  KinDynMod.slx Simulink model for simulating the dynamic and kinematic models used in models 2 and 3.

  ScriptKinTest.m Script with controller parameters for KinDynMod.slx.

  readmeKinDyn.txt Text file describing how to initialise this model.



NTNU

Norwegian University of
Science and Technology