NTNU

Norwegian University of
Science and Technology

Planning and Control of Energy Efficient Manipulation Task for ABB 1600 robot

Lars Roen Hansen

Master of Science in Cybernetics and Robotics
Submission date: December 2016
Supervisor: Anton Shiriaev, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Problem formulation

The problem formulation given by Professor Anton Shiriaev were: *The project is aimed at developing a motion planning algorithm that recovers an energy-efficient velocity profile of robot tool movement along a predescribed path. In the task the priority should be given to smooth behaviors consistent with contraints imposed on velocities and accelerations as well as jerk of joint variables. The result are supposed to be tested and validated in experiments. Comprehensive comparison with performance of ABB motion planner is to be given.*

Preface

This Master's thesis was carried out in the autumn of 2016, in order to finalize my study of Engineering Cybernetics at the Norwegian University of Science and Technology (NTNU) in Trondheim, and to acquire the degree of Master of Science. The project was thought of and carried out under the supervision of Professor Anton Shiriaev.
This thesis was not a continuation of a project thesis.

Trondheim, 2016-12-20

Lars Roen Hansen

Acknowledgment

I would like to thank my supervisor Professor Anton Shiriaev for his help and contribution during the course of this Master's thesis. I would also like to thank PhD Candidate Torleif Anstensrud for his help with the lab equipment, and Engineer Stepan Pchelkin, Ph.D., for his help using External Control and performing a torque experiment. I must also thank my family who have always supported me throughout my life and education, and a special thanks to my sister Mari Roen Hansen who has spent several hours proofreading this thesis.

L.R.H.

Abstract

Due to the ever increasing number of industrial robots in the world, working day and night producing and manufacturing, a need has arisen to make sure that these robots work energy efficient. This is both to make robots cost effective for businesses and environmentally friendly.

During the course of this thesis a method for planning an energy efficient velocity profile for an ABB IRB 1600 robot has been developed. This method does not only look at the assignment of velocity and acceleration along a path, but also the gain of energy efficiency that optimizing the position and orientation of a manipulation task have.

The result of this method shows that the optimal position and orientation gives a minimization of the movement of the second joint, and that at this optimal position the gain in energy efficiency of the optimal orientation is 161.1% from the horizontal plane. The optimal trajectory also have a 16.3% smaller energy consumption than a trajectory made by ABBs own motion planner in the optimal plane. Thus, the work done in this thesis optimize energy in two ways.

Experiments were also performed to find the maximum and minimum torque for the first three joints of the robot. This was supposed to be used when constraining acceleration and jerk. These constraints were however excluded from the optimization scheme as the energy efficient trajectories obtained were nowhere close to the torque bounds. These experimental results are however greatly of interest when looking at time-optimal trajectories, and can be reused for this purpose.

Sammendrag (Norsk)

På grunn av det stadig økende tallet av industriroboter, som jobber dag og natt i produksjon rundt om i verden, har det blitt nødvendig å sørge for at disse robotene jobber energieffektivt. Dette er både for å gjøre robotene kostnadseffektive for bedrifter og for å gjøre robotene miljøvennlige.

Gjennom arbeidet med denne masteroppgaven har en metode som planlegger en energieffektiv hastighetsprofil for en ABB IRB 1600 robot blitt utviklet. Denne metoden ser ikke bare på tildeling av hastighet og akselerasjon langs en bane, men også på hvordan en optimal posisjon og orientering av denne banen kan videre minske energiforbruket.

Resultatet av denne metoden viser at gjennom å optimere posisjon og orientering, så minimeres bevegelsen til det andre leddet til roboten. I den optimale posisjonen er gevinsten i energieffektivitet av den optimal orienteringen 161.1% i forhold til å gjøre den samme oppgaven i det horisontale planet. Den optimale hastighetsprofilen er også 16.3% mer energieffektiv enn det ABBs egen baneplanlegger klarer i det samme planet. Dermed energieffektiverer denne løsningsmetoden banen på to måter.

Eksperimenter for å finne maksimalt og minimalt dreiemoment for de tre første leddene til roboten, er også gjennomført. Disse resultatene skulle bli brukt for å begrense akselerasjon og rykk, men i og med at de optimale resultatene var langt unna de maksimale og minimale dreiemomentene ble disse bregrensningene sett bort i fra. Resultatene av disse eksperimentene er derimot av stor interesse for tidsoptimal baneplanlegging og kan bli gjenbrukt for denne hensikten.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The industrial world is becoming more and more automated, with industrial robots taking over manufacturing and assembly tasks in all parts of the industry. According to a summary of world robotics in 2015, done by the International Federation of Robotics, [9], industrial robot sales increased by 15% this year, and a further 61.3% increase is predicted until 2019. At the same time increasing emission of green house gases are endangering the world as we know it.

Industrial robots working day and night assembling, cutting, milling and loading material use a lot of electricity. When a statistic of the worlds key electricity trends, [10], shows that 67.2 % of the worlds electricity generation comes from burning fossil fuels such as coal, oil and natural gases, it is apparent that the more ineffective the industry is with its electricity, the higher emission becomes. It's clear that the need for energy efficient industrial robots is big.

The subject of energy efficient robotics is of great consideration in research all over the world. Especially the Areus Project which is a cooperation between several universities across Europe, such as Chalmers University in Sweden and DTU in Denmark, and several companies, such as KUKA robotics, are concentrating on making automation and robotics sustainable for manufacturing. During their research they have found that smooth trajectories could reduce energy consumption by up to 40 %, [11]. At NTNU this topic has also been looked at, where a previous

Master's thesis, Breistøl [7], have performed a 7% decrease in energy consumption using numerical optimization.

What hasn't been investigated at great extent is how much of an effect changing the position and orientation of a manipulation task around in the work space of the robot, have on the energy consumption. This could be of great interest in production and manufacturing around the world, not only to save the environment, but also because saving energy also save electricity which again save money. This area will be investigated in this thesis.

1.2 Objectives

1. The main objective of this thesis is as stated in the problem formulation to develop an algorithm or a method that returns an energy efficient velocity profile, constrained in velocity, acceleration and jerk.

2. The obtained velocity profile must be tested and validated in experiments, and must be compared to some norm made by ABBs own motion planner.

1.3 Limitations

The trajectories made in this thesis only use the first three joints of the robot. The last three joints, which controls the orientation of the robot tool are unactuated in zero position. It is however thought that since these three joints are the ones consuming the lowest amounts of energy, then it will not affect the end result as much as one could fear. This is therefore a small limitation of this thesis.

The trajectories from ABBs own motion planner however must be programmed using all six joints. It is impossible for the last three joints to be unactuated and to be in zero position. Therefore there will be some offset between the movement of the first three joints programmed with External Control and the first three joints programmed with ABBs own path planner. This will be further discussed later in the thesis.

1.4 Approach

In working with this thesis it was chosen to build on the work done by previous NTNU students, Breistøl [7] and Strandbråten [8]. They worked with the same robot and together they identified a lot of robot parameters and created a model for the robot, which will be used in this thesis. Breistøl [7] also worked in the area of energy efficient trajectories. Many changes and extensions has been made to further improve the results obtained in this thesis.

One of the biggest changes that was made in this thesis was to not only look at an optimal set of velocities and accelerations, but also what contribution changing the position and orientation of a path could give when trying to make the trajectory as energy efficient as possible. In order to make the trajectories smoother, a constraining of the acceleration of the different joints were performed. In addition to this a different choice of the expression for the velocity profile was made, as well as fixing the circle path previously modelled. Experiments obtaining an approximation of the torque bounds for the different joints in order to constrain the jerk and acceleration, was also conducted.

1.5 Structure of the Report

The report is structured as follows. After this introduction to the thesis, in **Chapter 2**, an introduction to the robot at hand is given, in addition to the two different ways of programming and controlling the robot. After that, in **Chapter 3**, the robot theory needed in this thesis will be presented including Kinematics, Dynamics and Motion Planning. In **Chapter 4**, the path will be described mathematically and the optimization problem will be set up. In **Chapter 5** some parameters for the optimization problem will be identified and the practical optimization scheme will be explained. This include the experiment obtaining maximum and minimum torques for the ABB 1600 robot. Explanation of the robot programming used to validate the results are also given here. The results will be presented in **Chapter 6**, and will be further discussed in **Chapter 7** where a conclusion and summary will also be presented. In **Appendix A** abbreviations are explained, in **Appendix B** some additional plots excluded from the main report are shown and in **Appendix C** the included files are explained.

Chapter 2

Robot and robot motion programming

A lot of the time working on this thesis were spent in one of the robot labs in the EL building at NTNU, more specifically antennehallen(the antenna hall). This time was used to set up the equipment, to gain experience using the equipment as well as performing experiments to obtain torque bounds and to validate results. The antenna hall includes several robots such as ABB IRB IRB 140-6/0.8, ABB IRB 1600-6/1.2, ABB IRB 4600-60/2.05, a KUKA LBR4+ robot and a butterfly robot, and different equipment such as a NIKON Metrology camera system. In this chapter a short description of the equipment and software that was used in this thesis will be given, but first a short introduction to industrial robotics.

2.1 Industrial robots

Industrial robots are robot systems mainly used in manufacturing and production, for applications including welding, painting and assembly. An industrial robot system can carry out the same movements over and over 24/7 for several years, without making errors or needing to stop, [1]. Such a robot typically consists of six rotational joints or axes, as the articulated robots shown in Figure 2.3, where three joints control the position of the robot end effector and the other three joints control the orientation of the end effector. Other common types of industrial robots are SCARA (Selective Compliance Articulated Robot Arm) robots shown in Figure 2.1, typically used for "pick and place" applications, and Cartesian robots shown in Figure 2.2 also used for this purpose.

Figure 2.1: SCARA manipulator, courtesy of [1]

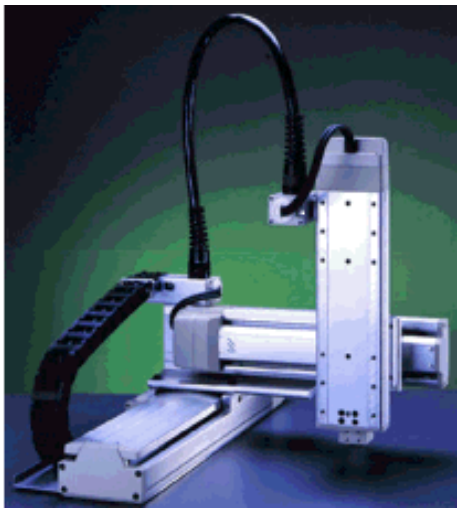

Figure 2.2: Cartesian manipulator, courtesy of [1]

In the present production environments industrial robots have taken over much assembly and manufacturing from human personnel, [12]. In an assembly line situation, like the one seen in Figure 2.3, several robots work together to assemble and produce products, in a much more effective manner that what could be achieved by human mechanics.

Figure 2.3: Production environment, courtesy of [2]

The top three industrial robot producers in the world are Swedish-Swiss ABB, Japanese Yaskawa, both with over 300000 robots installed world wide, [13], and German KUKA. All these three producers deliver several different solutions from large scale robots to small scale robots.

2.2 ABB IRB 1600

The industrial robot used in this thesis is ABBs IRB 1600-6/1.2 robot. With its weight of 250 kg and reach of 1.2 meters, it is considered a quite small articulated industrial robot, compared to ABBs' biggest robot weighing up to 800 kg with a reach of 4.2 m. Because of its size, and with a position repeatability of 0.02mm and a path repeatability of 0.13mm, it is ideal for tasks such as Arc Welding, Machine Tending, Material Handling, Gluing, Deburring and Grinding applications, [3]. The robot consists of six revolute joints, and as can be seen in the name it can handle payloads of up to 6 kg. In Figure 2.4 the robot and the reach of it in millimeters are shown.

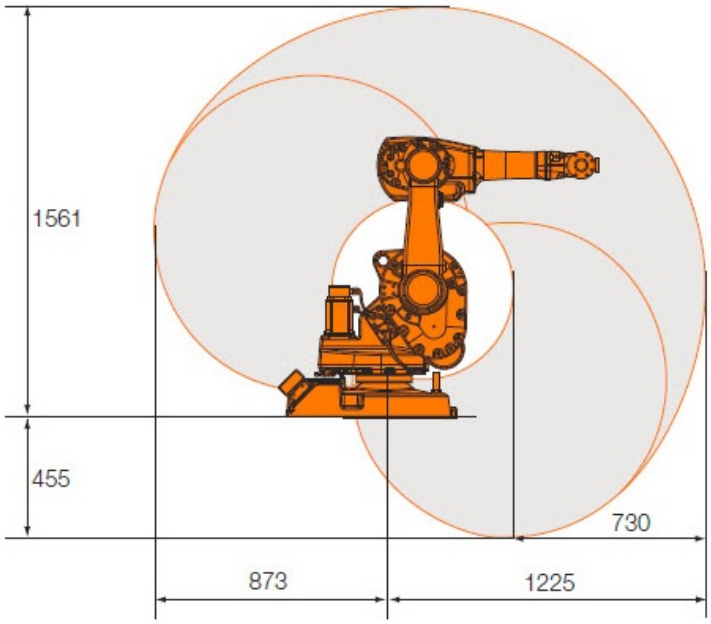

Figure 2.4: IRB 1600-6/1.2, courtesy of [3]

The working range and the maximum angular velocities of the robot are shown in the Tables 2.1 and 2.2

Table 2.1: Working range of robot, courtesy of [3]

Axis 1	$+180^\circ$, -180°
Axis 2	$+136^\circ$, -63°
Axis 3	$+55^\circ$, -235°
Axis 4	$+200^\circ$, -200° def.
	$+/-190^\circ$ revolution
Axis 5	$+115^\circ$, -115°
Axis 6	$+400^\circ$, -400° def.
	$+/-288^\circ$ revolution

Table 2.2: Maximum angular velocity, courtesy of [3]

Axis 1	$150°/s$
Axis 2	$160°/s$
Axis 3	$170°/s$
Axis 4	$320°/s$
Axis 5	$400°/s$
Axis 6	$480°/s$

2.2.1 RobotStudios and RAPID programming

In order to program paths for the ABB IRB 1600 robot, a program called RobotStudios can be used. The programming language used in RobotStudios is called RAPID, and the basic RAPID syntax used in this thesis are found in the RAPID introduction [14].

In RobotStudios every ABB robot can be chosen, and by using a 3D model of the robot it is possible to program movements defining points and paths between these points in the robots' work space. RobotStudios then creates RAPID code using these points and paths, through so called Move instructions. There are several different move instructions, such as MoveL(linear movement), MoveJ(non-linear quickest path), MoveC(circular movement) and MoveAbsJ(specific joint movement).
A typical such move instruction are given below

MoveC p_1**,** p_2**, v100 \T:=5, z10, tool0**

In this move instruction MoveC describes the type of instruction, here circular. p_1 describes an intermediate point used in some movements, while p_2 describes the end point of the movement. v100 expresses a maximum linear velocity of 100 mm/s and T:=5 describes a execution time of 5 seconds. z10 describes the so called zone data which explains that corners can be cut when 10mm from the corner, and tool0 is the chosen tool.

An example of two paths programmed in RobotStudios can be seen in Figure 2.5. These paths are one square and one circle. Here, the blue arrows represents the z-axis of the end-effector at a certain point, the red arrows represents the y-axis, the green arrows represents the x-axis, while the yellow arrows show the direction of the movement between different points.

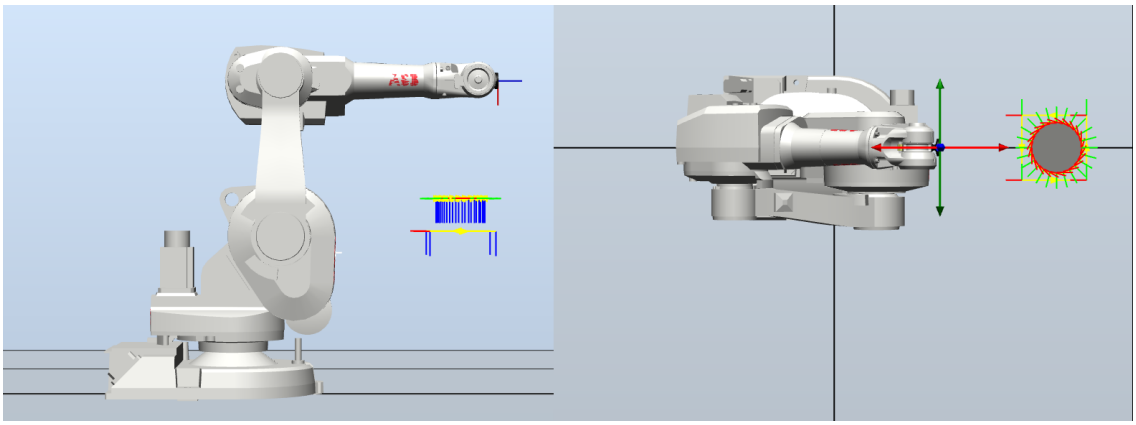

Figure 2.5: Example of two paths programmed in RobotStudios, from the side and top

2.3 External Control - extctr

In order to work around the ABB RAPID controller and path planner, an external control interface called External Control is used. External Control opens up the opportunity to model a controller in Simulink which is integrated in the robot control using Real-Time Workshop, [4]. Real-Time Workshop, which in recent MATLAB versions has been renamed Simulink Coder, is a code generator functionality in MATLAB, which provides C/C++ code from a simulink model. This c-code is compiled to an executable file that can be run through an opcom interface on a computer which is coupled with the robots' main and axis computer, through both an RS-232 Serial connection and an Ethernet connection.

Figure 2.6 shows a schematic of how the Simulink controller is used in External Control. The data from the robots main computer is here redirected in to the simulink controller and manipulated there, before being sent to the axis computer. The four states needed to load the controller are explained in Table 2.3

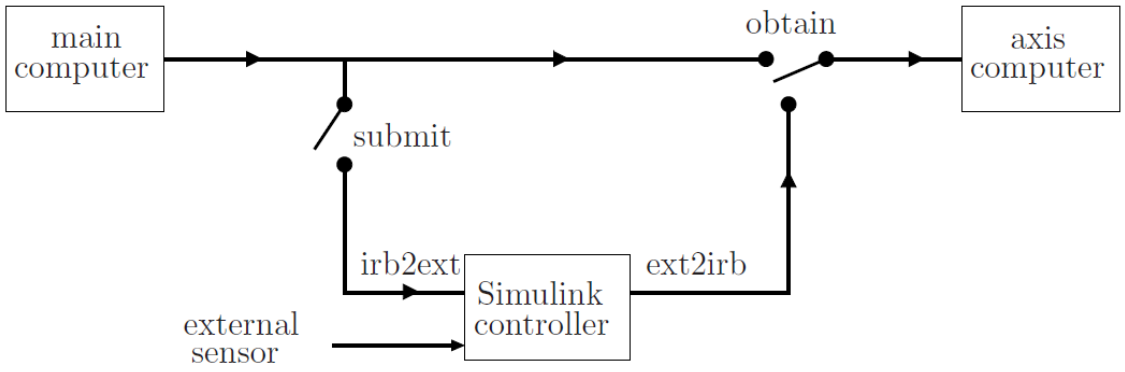

Figure 2.6: Path of robot variables from main to axis computer, courtesy of [4]

Table 2.3: States for loading controller

State	Explanation
unload	Direct communication between ABBs main and axis computer, and no communication with simulink controller
load	Controller is set up, but no communication with either main or axis computer
submit	The controller now listens to the robot data from the main computer, but it cannot send data to the axis computer. Can be used for logging ABB programs
obtain	Two way communication. The simulink controller is now receiving data from the main computer and communicating manipulated data to the axis computer

The robot variables are made available for the simulink model, by being defined in a .lc-file, which along with the c-code of the simulink model become part of the executable file used in the opcom interface. These variables are accessed defining input signals and output signals, in the simulink model, with names on the forms *irb2ext.** and *ext2irb.**, see Figure 2.6, where * defines what variable that is going to be accessed. Some of the signals of interest in this thesis are presented in Table 2.4.

Table 2.4: Signals of interest

Input		Output	
Variable	Explanation	Variable	Explanation
posRawFb	measured position	posRef	position reference
velRaw	measured velocity	velRef	velocity reference
trqRefFlt	applied torque	trqFfw	feed forward torque
parKp & parKv	controller gains	parKp & parKv	controller gains

Chapter 3

Robot theory, kinematics, dynamics, motion planning and controller

In this chapter some robot preliminaries will be presented. These preliminaries are basic robot math and modelling used to describe the complex robotic systems. This include a way of computing the position and orientation of the end effector using the configuration values of the robot, a way of computing the robot configuration values given the end effector position and orientation, and a way of calculating torques using the current state of the robot. These techniques are known respectively as the Forward Kinematics, the Inverse Kinematics and the Robot Dynamics. In addition to this some basic theory regarding robot motion planning will be presented, as well as a proposed way of controlling the robot.

3.1 Forward Kinematics and The Denavit-Hartenberg Convention

The Forward Kinematics is the method of calculating the end-effector position of a robot, knowing the joint configurations. A joint configurations is an angle for a rotational joint and an extension for a prismatic joint. A common convention used to select frames of reference and to calculate the forward kinematics of a robot is the Denavit-Hartenberg convention (the DH convention). This convention was introduced by Jacques Denavit and Richard Hartenberg in 1955.

Here each transformation matrix, $\mathbf{A}_i$, between frame $i-1$ and i, is a series of four basic transformations, defined in Spong et al. [15] as follows

$$\begin{aligned}\mathbf{A}_i &= \mathbf{Rot}_{z,\theta_i} \cdot \mathbf{Trans}_{z,d_i} \cdot \mathbf{Trans}_{x,a_i} \cdot \mathbf{Rot}_{x,\alpha_i} \\ &= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} c_{\alpha_i} & s_{\theta_i} s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i} c_{\alpha_i} & -c_{\theta_i} s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}\end{aligned} \tag{3.1}$$

In Equation 3.1 $\mathbf{Rot}_{*,*}$ represents a rotation about either the z- or the x-axis with an angle of either θ_i or α_i, and $\mathbf{Trans}_{*,*}$ represents a translation along either the z- or the x-axis by either d_i or a_i. The DH variables, a_i, α_i, d_i and θ_i, used in the transformation matrices above, each has a specific definition that needs to be upheld if the calculation between frames are to be done correctly, and are in Spong et al. [15], respectively given the names **link length**, **link twist**, **link offset** and **joint angle**. The definitions of these variables are shown in Figure 3.1, and further explained in Table 3.1.

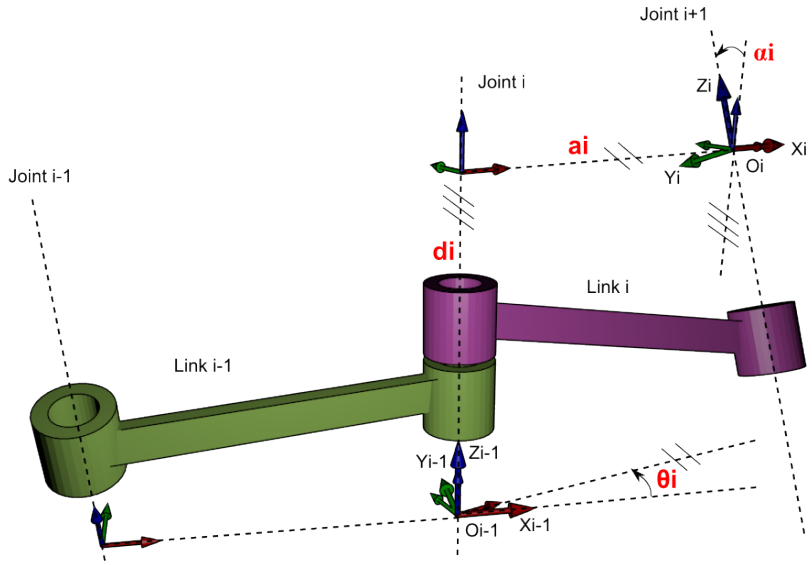

Figure 3.1: The classic DH parameters shown in red, courtesy of [5]

Table 3.1: Explanation of DH parameters

Parameter	Explanation
d_i	offset along z_{i-1} to common normal of frames z_{i-1} and z_i
θ_i	angle about z_{i-1} from x_{i-1} to x_i
a_i	distance along common normal, x_i, from frame $i-1$ to i
α_i	angle about x_i from z_{i-1} to z_i

3.1.1 ABB IRB1600 DH parameters

Selecting the frames of reference for the ABB IRB1600 robot according to the DH convention, results in the assignment in Figure 3.2.

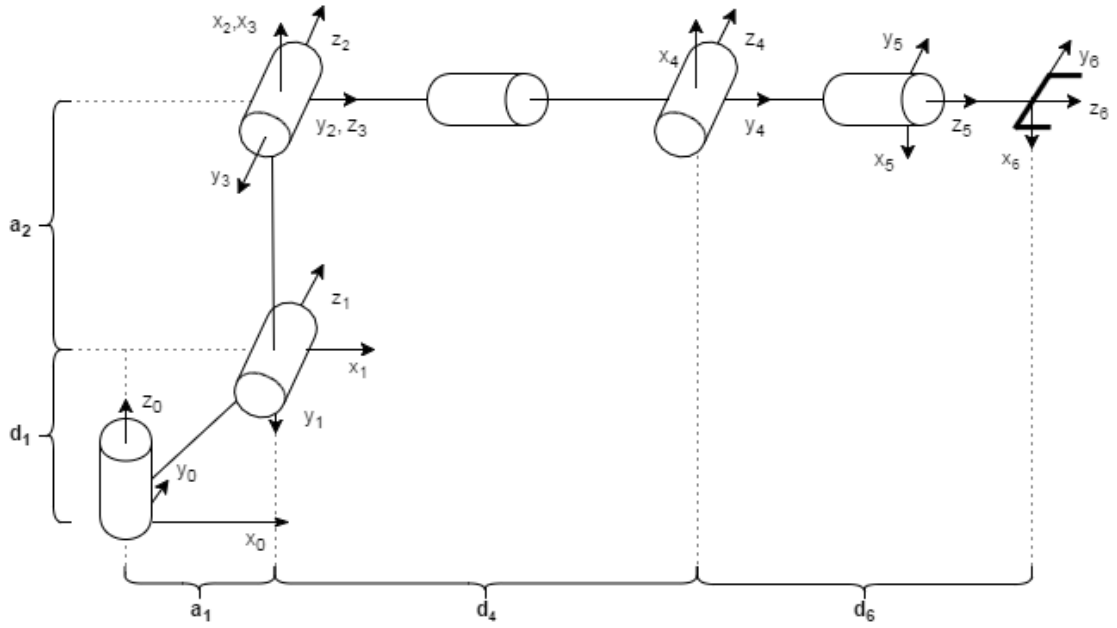

Figure 3.2: Denavit-Hartenberg frame assignment for ABB 1600 robot

Using this frame assignment and the robot dimension values given in the product specification [16], provides the DH Table 3.2. Inserting these values into the transformation matrices from Equation 3.1, gives the matrices in Equation 3.2

Table 3.2: ABB IRB 1600 robot DH parameters

Link	θ_i	d_i	a_i	α_i
1	$\theta_1 = q_1$	$d_1 = 486.5mm$	$a_1 = 150mm$	$\alpha_1 = -90°$
2	$\theta_2 = q_2 - 90°$	$d_2 = 0mm$	$a_2 = 475mm$	$\alpha_2 = 0°$
3	$\theta_3 = q_3$	$d_3 = 0mm$	$a_3 = 0mm$	$\alpha_3 = -90°$
4	$\theta_4 = q_4$	$d_4 = 600mm$	$a_4 = 0mm$	$\alpha_4 = 90°$
5	$\theta_5 = q_5 + 180°$	$d_5 = 0mm$	$a_5 = 0mm$	$\alpha_5 = 90°$
6	$\theta_6 = q_6$	$d_6 = 65mm$	$a_6 = 0mm$	$\alpha_6 = 0°$

$$\mathbf{A}_1 = \begin{bmatrix} c_{q_1} & 0 & -s_{q_1} & a_1 c_{q_1} \\ s_{q_1} & 0 & c_{q_1} & a_1 s_{q_1} \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{A}_2 = \begin{bmatrix} s_{q_2} & c_{q_2} & 0 & a_2 s_{q_2} \\ -c_{q_2} & s_{q_2} & 0 & -a_2 c_{q_2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

$$\mathbf{A}_3 = \begin{bmatrix} c_{q_3} & 0 & -s_{q_3} & 0 \\ s_{q_3} & 0 & c_{q_3} & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{A}_4 = \begin{bmatrix} c_{q_4} & 0 & s_{q_4} & 0 \\ s_{q_4} & 0 & -c_{q_4} & 0 \\ 0 & 1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{A}_5 = \begin{bmatrix} -c_{q_5} & 0 & -s_{q_5} & 0 \\ -s_{q_5} & 0 & c_{q_5} & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{A}_6 = \begin{bmatrix} c_{q_6} & -s_{q_6} & 0 & 0 \\ s_{q_6} & c_{q_6} & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Simplified robot

As mentioned in the introduction, the last three joints will be unactuated and set to zero when planning motions and when using External Control. Therefore some simplifications can be made to the DH matrices and the last four DH-matrices can be restated as one single matrix, where the last transformation is a rotation about z_2 with $q_3 + 90^\circ$ and a translation along x_3 with $d_4 + d_6$, resulting in

$$\mathbf{A}_{3-6} = \begin{bmatrix} -s_{q_3} & -c_{q_3} & 0 & -(d_4 + d_6) s_{q_3} \\ c_{q_3} & -s_{q_3} & 0 & (d_4 + d_6) c_{q_3} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

3.1.2 Forward kinematics

The forward kinematics is achieved by multiplying all the transformation matrices with each other as

$$\mathbf{T}_6^0 = \mathbf{A}_1\mathbf{A}_2\mathbf{A}_3\mathbf{A}_4\mathbf{A}_5\mathbf{A}_6 \tag{3.4}$$

The resulting matrix has the form

$$\mathbf{T}_6^0(\mathbf{q}) = \begin{bmatrix} \mathbf{R}_6^0(\mathbf{q}) & \mathbf{o}_6^0(\mathbf{q}) \\ \mathbf{0} & 1 \end{bmatrix} \tag{3.5}$$

with

$$\mathbf{q} = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 & q_5 & q_6 \end{bmatrix}^T \tag{3.6}$$

In Equation 3.5, $\mathbf{R}_6^0(\mathbf{q})$ represents the orientation of the end effector, while $\mathbf{o}_6^0(\mathbf{q})$ is the position of the end effector in relation to the base frame, i.e. the forward kinematics. This position is calculated to be

$$\mathbf{o}_6^0(q) = \begin{bmatrix} o_{ex} \\ o_{ey} \\ o_{ez} \end{bmatrix} \tag{3.7}$$

where

$$\begin{aligned} o_{ex} =& a_1 c_{q1} + d_6(c_{q5}(c_{q1}c_{q2}c_{q3} - c_{q1}s_{q2}s_{q3}) - s_{q5}(s_{q1}s_{q4} + c_{q4}(c_{q1}c_{q2}s_{q3} + c_{q1}s_{q2}c_{q3}))) \ldots \\ & \ldots + d_4(c_{q1}c_{q2}c_{q3} - c_{q1}s_{q2}s_{q3}) + a_2 c_{q1}s_{q2} \end{aligned} \tag{3.8}$$

$$\begin{aligned} o_{ey} =& a_1 s_{q1} + d_6(c_{q5}(c_{q2}c_{q3}s_{q1} - s_{q1}s_{q2}s_{q3}) + s_{q5}(c_{q1}s_{q4} - c_{q4}(s_{q1}c_{q2}s_{q3} + s_{q1}s_{q2}c_{q3}))) \ldots \\ & \ldots + d_4(s_{q1}c_{q2}c_{q3} - s_{q1}s_{q2}s_{q3}) + a_2 s_{q1}s_{q2} \end{aligned} \tag{3.9}$$

$$o_{ez} = d_1 + a_2 c_{q2} - d_4(c_{q2}s_{q3} + s_{q2}c_{q3}) - d_6(c_{q5}(c_{q2}s_{q3} + s_{q2}c_{q3}) + c_{q4}s_{q5}(c_{q2}c_{q3} - s_{q2}s_{q3})) \tag{3.10}$$

The forward kinematics for the simplified robot stays the same as long as the last three joint angles are set to zero.

3.2 Inverse Kinematics

The inverse kinematics is the method of computing the joint configurations of an n DOF robot, given the Cartesian position and orientation of the robot end effector. It is important to notice that this exercise is increasingly extensive with the number of joints of a robot, and that few positions of an end effector are specific to a single set of joint configurations, as one example in Figure 3.3 shows.

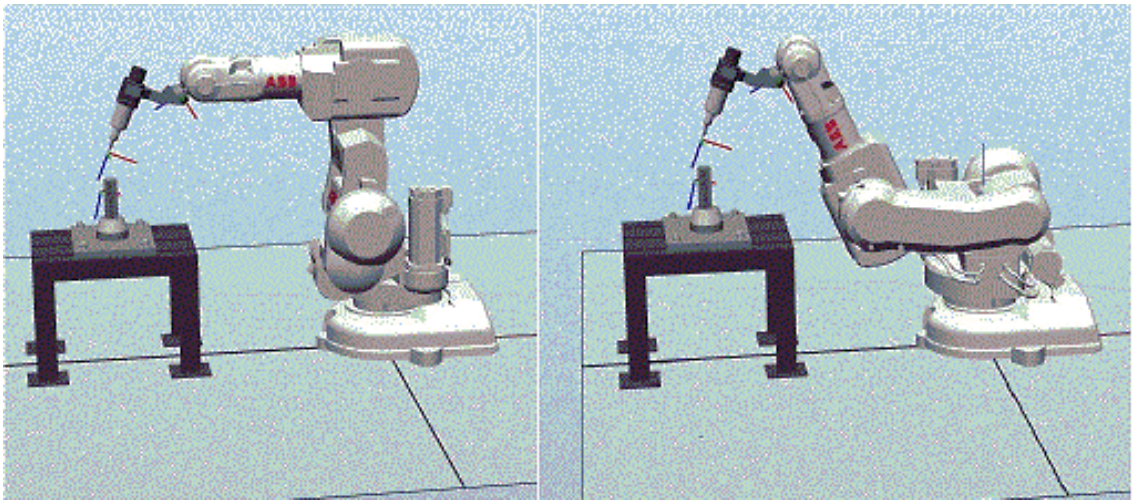

Figure 3.3: One end effector position, two different joint configurations, courtesy of [6]

3.2.1 Kinematics decoupling for ABB IRB1600

For robots having six joints, where the last three joints intersect at a point, as for the ABB IRB 1600 robot used in this thesis, it is possible to split the inverse kinematics problem into two simpler problems, [15]. Here one problem looks only at the first three joint configurations and the other one looks at the last three joint configurations.

As can be seen in Figure 3.2 the axes $\mathbf{z}_3$, $\mathbf{z}_4$ and $\mathbf{z}_5$, all intersect at a point, from now called $\mathbf{o}_c = [x_c, y_c, z_c]^T$, representing the center of the robot wrist. The origins $\mathbf{o}_4$ and $\mathbf{o}_5$ will always be at $\mathbf{o}_c$, and this position can therefore be calculated using only the first three robot configuration angles, q_1, q_2 and q_3. For this reason this problem has been called the inverse position kinematics, [15]. The end effector position $\mathbf{o}$ is achieved by translating from $\mathbf{o}_c$ along $\mathbf{z}_5$ the

distance d_6 using the formula

$$\mathbf{o} = \mathbf{o}_c + d_6\mathbf{R}\begin{bmatrix}0\\0\\1\end{bmatrix} \tag{3.11}$$

where **R** is the orientation of the end effector. The last three joints control the orientation of the end effector, therefore this problem of computing these joint configurations is called the inverse orientation kinematics.

The inverse position kinematics

The inverse position kinematics are often calculated by geometrically investigating the position, $\mathbf{o}_c$. Firstly, q_1 is found using the x and y position of $\mathbf{o}_c$, see Figure 3.4, and using the two-argument arctangent function like:

$$q_1 = atan2(y_c, x_c) \tag{3.12}$$

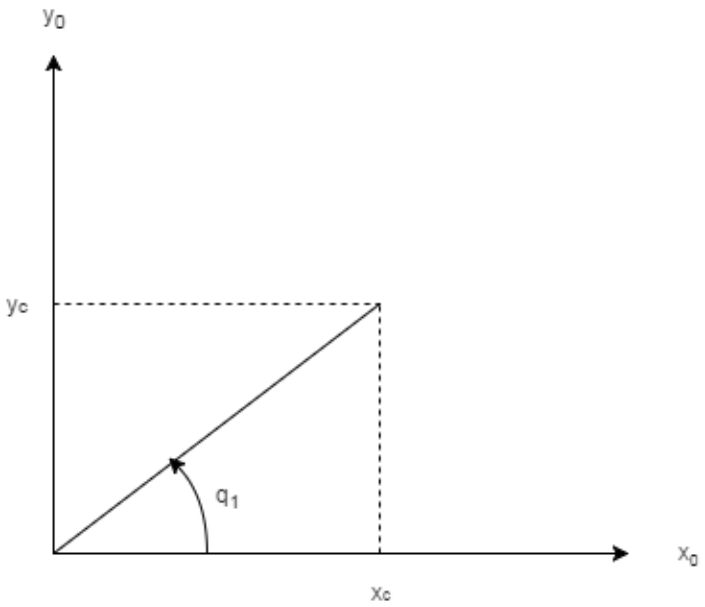

Figure 3.4: Angle q_1 in $x_0 - y_0$-plane

The reason that atan2 is used, instead of the classic arctangent function is because atan2

takes in two arguments, x_c and y_c, in stead of one fraction, $\frac{y_c}{x_c}$, and uses the individual signs of these arguments in order to find the correct quadrant of the angle.

Next it is wanted to find q_2 and q_3. Now the horizontal plane is no longer of interest. The vertical plane composed of z^* which is parallel to z_0 and x^* parallel to the horizontal plane, is used. This coordinate system has its origin in the second joint as can be seen in Figure 3.5.

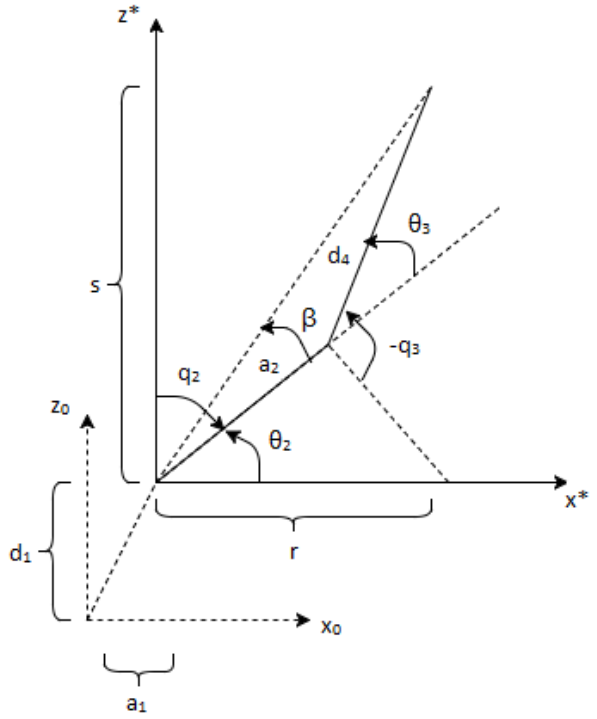

Figure 3.5: Angles q_2 and q_3 in $x^* - z^*$-plane

The first task is to calculate the distance r along x^* from the center of joint 2 to $\mathbf{o}_c$, and the distance s along z^* also from the center of joint 2 to $\mathbf{o}_c$. This is done in Equations 3.13 and 3.14

$$r = \sqrt{(x_c - a_1 cos(q_1))^2 + (y_c - a_1 sin(q_1))^2} \tag{3.13}$$

$$s = z_c - d_1 \tag{3.14}$$

From the law of cosines the calculation of θ_3 can be performed giving the expression for q_3 in Equation 3.18

$$cos(\theta_3) = \frac{r^2 + s^2 - a_2^2 - d_4^2}{2a_2 d_4}$$
$$= \frac{(x_c - a_1 cos(q_1))^2 + (y_c - a_1 sin(q_1))^2 + (z_c - d_1)^2 - a_2^2 - d_4^2}{2a_2 d_4} := D \tag{3.15}$$
$$sin(\theta_3) = \pm\sqrt{1 - D^2} \tag{3.16}$$
$$\theta_3 = atan2\left(\pm\sqrt{1 - D^2}, D\right) \tag{3.17}$$
$$q_3 = -\theta_3 - \frac{\pi}{2} \tag{3.18}$$

Equation 3.18 is compensating for the zero position of the third joint, which is rotated $\frac{\pi}{2}$rad from the zero position used to calculate θ_3. q_3 is also defined negative relative to θ_3. Because of the $\pm$-sign in Equation 3.17 there are two different solutions of q_3 corresponding to either elbow up or elbow down.

q_2 can now be calculated as in Equation 3.21, through a calculation of θ_2. θ_2 is the difference between the angle between the x^*-axis and the stapled line from joint 2 to $\mathbf{o}_c$, and the angle β, seen in Figure 3.5. The resulting equations becomes

$$\theta_2 = atan2(s, r) - atan2(a_2 + d_4 cos(\theta_3), d_4 sin(\theta_3)) \tag{3.19}$$
$$\theta_2 = atan2(z_c - d_1, \sqrt{(x_c - a_1 cos(q_1))^2 + (y_c - a_1 sin(q_1))^2})$$
$$- atan2(a_2 + d_4 cos(\theta_3), d_4 sin(\theta_3) \tag{3.20}$$
$$q_2 = \frac{\pi}{2} - \theta_2 \tag{3.21}$$

also using the fact that θ_2 are defined as negative relative to q_2, and rotated $\frac{\pi}{2}$ rad from the zero position in Figure 3.5.

The inverse orientation kinematics

To find the orientation angles, the fact that

$$\mathbf{R}_6^3(\mathbf{q}) = (\mathbf{R}_3^0(\mathbf{q}))^T \mathbf{R}_6^0(\mathbf{q}) \tag{3.22}$$

is used, [15]. Since the first three joint angles are already known, as well as some fixed orientation $\mathbf{R} = \mathbf{R}_6^0(\mathbf{q})$ of the end effector, the orientation angles can be found investigating Equation 3.22. From the DH matrices these matrices are calculated to

$$\begin{bmatrix} s_{q4}s_{q6} - c_{q4}c_5c_{q6} & s_{q4}c_{q6} + c_{q4}c_{q5}s_{q6} & -c_{q4}s_{q5} \\ -c_{q4}s_{q6} - s_{q4}c_{q5}c_{q6} & -c_{q4}c_{q6} + s_{q4}c_{q5}s_{q6} & -s_{q4}s_{q5} \\ -s_{q5}c_{q6} & s_{q5}s_{q6} & c_{q5} \end{bmatrix} = \begin{bmatrix} c_{q1}s_{q2}c_{q3} + c_{q1}c_{q2}s_{q3} & s_{q1}s_{q2}c_{q3} + s_{q1}c_{q2}s_{q3} & c_{q2}c_{q3} - s_{q2}s_{q3} \\ s_{q1} & -c_{q1} & 0 \\ -c_{q1}s_{q2}s_{q3} + c_{q1}c_{q2}c_{q3} & -s_{q1}s_{q2}s_{q3} + s_{q1}c_{q2}c_{q3} & -c_{q2}s_{q3} - s_{q2}c_{q3} \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{3.23}$$

When investigating these matrices, the cosine of q_5 is found in the lower right corner, and is calculated through matrix multiplication to be

$$c_{q5} = r_{13}(-c_{q1}s_{q2}s_{q3} + c_{q1}c_{q2}c_{q3}) + r_{23}(-s_{q1}s_{q2}s_{q3} + s_{q1}c_{q2}c_{q3}) + r_{33}(-c_{q2}s_{q3} - s_{q2}c_{q3}) \tag{3.24}$$

The sine of q_5 is gotten from the other two elements on column three, and calculated to be

$$\begin{aligned} \pm s_{q5} &= \sqrt{(-c_{q4}s_{q5})^2 + (-s_{q4}s_{q5})^2} \\ &= \sqrt{(r_{13}(c_{q1}s_{q2}c_{q3} + c_{q1}c_{q2}s_{q3}) + r_{23}(s_{q1}s_{q2}c_{q3} + s_{q1}c_{q2}s_{q3}) + r_{33}(c_{q2}c_{q3} - s_{q2}s_{q3}))^2} \\ &\overline{+(r_{13}s_{q1} - r_{23}c_{q1})^2} \end{aligned} \tag{3.25}$$

The angle q_5 can then be calculated using these expressions through

$$q_5 = atan2(\pm s_{q5}, c_{q5}) \tag{3.26}$$

The last two orientation angles can now be calculated, knowing q_5. Looking at the third row and column of Equation 3.23, it is seen that these angles can be calculated, depending on if s_{q5} is

positive or negative respectively, through

$$q_4 = atan2(\pm(-s_{q4}s_{q5}), \pm(-c_{q4}s_{q5})) \tag{3.27}$$

$$q_6 = atan2(\pm(s_{q5}s_{q6}), \mp(-s_{q5}c_{q6})) \tag{3.28}$$

inserting the elements on the right side of Equation 3.23 corresponding to the left side elements in Equations 3.27 and 3.28. As can be seen from these expressions $q_5 = 0°$ results in a wrist singularity, as both arguments of the atan2 function is zero, and it is impossible to calculate q_4 and q_6 accurately.

3.3 Robot Dynamics

The standard equation for robot dynamics is defined as, [15]

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \boldsymbol{\tau} \tag{3.29}$$

This equation is used to consider the torques and forces that are producing a robot motion, [15]. In Equation 3.29, $\mathbf{M}(\mathbf{q})$ is the mass inertia matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is the coriolis/centrifugal matrix and $\mathbf{G}(\mathbf{q})$ is the gravity matrix. These matrices and Equation 3.29 can be calculated using one of two different approaches; Euler-Lagrange and Newton-Euler. Both approaches leads to the same result of Equation 3.29, but as the Newton-Euler approach investigates each link individually and recursively, the Euler-Lagrange approach rely on the theory of using one Lagrangean for the system. Some of the most essential details for both approaches will be presented here, and are more accurately described in Spong et al. [15].

3.3.1 Euler-Lagrange Equations

As mentioned, the **Euler-Lagrange equations** of motion are based on the use of a **Lagrangean**, L. A Lagrangean is defined as the difference between the kinetic energy, K, and the potential energy, P. When deriving these energies, the Lagrangean and the Euler-Lagrange equations of motion for an n DOF robot, it is as stated in Spong et al. [15], easier to first look at Newton's

Second Law,

$$f = m \cdot a \tag{3.30}$$

for a one dimensional system, where a particle can move only in the y-direction. Newton's Second Law for this particle of mass, m, is given by

$$m\ddot{y} = f - mg \tag{3.31}$$

where f is the force, g is the gravitational constant and $\ddot{y}$ is the acceleration. The left side of this equation can easily be rewritten as

$$\frac{d}{dt}(m\dot{y}) = \frac{d}{dt}\frac{\partial}{\partial \dot{y}}\left(\frac{1}{2}m\dot{y}^2\right) \tag{3.32}$$

where $K = \frac{1}{2}m\dot{y}^2$ is the kinetic energy of the system. To obtain the potential energy of the system the gravitational force on the left side of Equation 3.31 can be rewritten in the same manner as

$$mg = \frac{\partial}{\partial y}(mgy) \tag{3.33}$$

where $P = mgy$. Now that the ingredients of the **Lagrangean** are obtained, it can be written as

$$L = K - P = \frac{1}{2}m\dot{y}^2 - mgy \tag{3.34}$$

Since the kinetic energy only rely on the velocity of the particle and the potential energy only rely on the position of the particle, it is obvious that differentiating the Lagrangean w.r.t. the velocity is the same as differentiating the kinetic energy w.r.t. the velocity,

$$\frac{\partial L}{\partial \dot{y}} = \frac{\partial K}{\partial \dot{y}} \tag{3.35}$$

and that differentiating the Lagrangean w.r.t. the position is the same as differentiating the negative potential energy w.r.t. the position

$$\frac{\partial L}{\partial y} = -\frac{\partial P}{\partial y} \tag{3.36}$$

Using these facts Equation 3.31 can be rewritten as

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{y}} - \frac{\partial L}{\partial y} = f \tag{3.37}$$

which is the so called **Euler-Lagrange equations** of motion for the one particle system. The **Euler-Lagrange equation** for a robot manipulator with n DOFs have the same form and is written as

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = \tau_i \tag{3.38}$$

By choosing

$$K = \frac{1}{2}\sum_{i,j}^{n} m_{ij}(\mathbf{q})\dot{q}_i\dot{q}_j \tag{3.39}$$

and $P_i = g^T r_{ci} m_i$ where r_{ci} is the distance to the c.o.m. of link i and m_i is the mass of the same link, it is shown in Spong et al. [15, p. 200-201] how using the equations above to end up with

$$\sum_i m_{ij}(\mathbf{q})\ddot{q}_j + \sum_{i,j} c_{ijk}(\mathbf{q})\dot{q}_i\dot{q}_j + g_k(\mathbf{q}) = \tau_k \tag{3.40}$$

where $k = 1 \dots n$ and

$$c_{ijk} = \frac{1}{2}\left\{\frac{\partial m_{kj}}{\partial q_i} + \frac{\partial m_{ki}}{\partial q_j} - \frac{\partial m_{ij}}{\partial q_k}\right\} \tag{3.41}$$

is the Christoffel Symbols. This derivation is quite extensive and require a lot of equations. It is therefore chosen to exclude this from the report. It is however easy to see that Equation 3.40 has the form of Equation 3.29.

3.3.2 Newton-Euler formulation

The second approach of obtaining the Dynamics Equation 3.29 is Newton-Euler. As mentioned earlier this approach treats every link of the robot individually. As each link of the robot is coupled to other links, the equations for force or torque for a certain link will contain so called coupling forces and torques. The Newton-Euler equations can because of these coupling forces and torques, be derived using forward and backward recursion.

When deriving the Newton-Euler equations some special notation are used. Explanations of this notation are found in Table 3.3, [15].

Table 3.3: Newton-Euler vectors and notation

$\mathbf{a}_{c,i}$	=	the acceleration at link i's c.o.m.
$\mathbf{a}_{e,i}$	=	the acceleration at joint i+1, or at the end of the i-th link
$\boldsymbol{\omega}_i$	=	the angular velocity of the i-th frame w.r.t. the base frame
$\boldsymbol{\alpha}_i$	=	the angular acceleration of the i-th frame w.r.t. the base frame
$\mathbf{g}_i$	=	the gravitational acceleration in the i-th frame
$\mathbf{f}_i$	=	the force from link i-1 on link i
$\boldsymbol{\tau}_i$	=	the torque from link i-1 on link i
$\mathbf{R}_i^{i+1}$	=	the rotation matrix between frame i+1 and i
m_i	=	mass of the i-th link
$\mathbf{I}_i$	=	inertia matrix of link i
$\mathbf{r}_{i,ci}$	=	vector from joint i to the c.o.m. of link i
$\mathbf{r}_{i+1,ci}$	=	vector from joint i+1 to the c.o.m. of link i
$\mathbf{r}_{i,i+1}$	=	vector from joint i to i+1

Deriving forces and torques

The first equation to be presented, when deriving the dynamics equation, is the force balance for link i,

$$\mathbf{f}_i - \mathbf{R}_i^{i+1}\mathbf{f}_{i+1} + m_i\mathbf{g}_i = m_i\mathbf{a}_{c,i} \tag{3.42}$$

Here, it is already apparent what is meant by the coupling forces, as Equation 3.42 include the element f_{i+1} which is the the force for the next link. This represent the backward recursion of the Newton-Euler approach.

Next, the moment balance is written down. Moment is defined by a force f around a point, $f \times r$, with r being the distance between the point where the force is applied and the point where the moment is calculated. Equation 3.43 shows this

$$\boldsymbol{\tau}_i - \mathbf{R}_i^{i+1}\boldsymbol{\tau}_{i+1} + \mathbf{f}_i \times \mathbf{r}_{i,ci} - (\mathbf{R}_i^{i+1}\mathbf{f}_{i+1}) \times \mathbf{r}_{i+1,ci} = \boldsymbol{\alpha}_i + \boldsymbol{\omega}_i \times (\mathbf{I}_i\boldsymbol{\omega}_i) \tag{3.43}$$

Here, both a coupling torque, τ_{i+1}, and a coupling force, f_{i+1}, are present.

Now, in order to calculate the forces, $\mathbf{f}_i$, and the torques, $\boldsymbol{\tau}_i$, the unknowns of Equations 3.42 and 3.43, $\boldsymbol{\omega}_i$, $\mathbf{a}_{c,i}$ and $\boldsymbol{\alpha}_i$, needs to be calculated. In these equations elements of the last link, $i-1$, will be present. This represent the forward recursion of the approach.
First, the angular velocity will be calculated

$$\boldsymbol{\omega}_i^{(0)} = \boldsymbol{\omega}_{i-1}^{(0)} + \mathbf{z}_{i-1}\dot{q}_i \tag{3.44}$$

where (0) in $\boldsymbol{\omega}_i^{(0)}$ refers to the base frame. Expressing Equation 3.44 in frame i gives

$$\boldsymbol{\omega}_i = (\mathbf{R}_{i-1}^i)^T \boldsymbol{\omega}_{i-1} + \mathbf{b}_i \dot{q}_i \tag{3.45}$$

where $\mathbf{b}_i$ is joint i's axis of rotation in the i-th frame

$$\mathbf{b}_i = (\mathbf{R}_0^i)^T \mathbf{z}_{i-1} \tag{3.46}$$

Having represented the angular velocity, the angular acceleration is given by

$$\boldsymbol{\alpha}_i = (\mathbf{R}_0^i)^T \dot{\boldsymbol{\omega}}_i^{(0)} \tag{3.47}$$

where Equation 3.44 is differentiated w.r.t. time to get

$$\dot{\boldsymbol{\omega}}_i^{(0)} = \dot{\boldsymbol{\omega}}_{i-1}^{(0)} + \mathbf{z}_{i-1}\ddot{q}_i + \boldsymbol{\omega}_i^{(0)} \times \mathbf{z}_{i-1}\dot{q}_i \tag{3.48}$$

giving the updated equation for the angular acceleration

$$\boldsymbol{\alpha}_i = (\mathbf{R}_{i-1}^i)^T \boldsymbol{\alpha}_{i-1} + \mathbf{b}_i \ddot{q}_i + \boldsymbol{\omega}_i \times \mathbf{b}_i \dot{q}_i \tag{3.49}$$

Lastly, the expressions for linear velocity and acceleration in the c.o.m. of the i-th link are calculated

$$\mathbf{v}_{c,i}^{(0)} = \mathbf{v}_{e,i-1}^{(0)} + \boldsymbol{\omega}_i^{(0)} \times \mathbf{r}_{i,ci}^{(0)} \tag{3.50}$$

$$\mathbf{a}_{c,i}^{(0)} = \mathbf{a}_{c,i-1}^{(0)} \times \mathbf{r}_{i,ci}^{(0)} + \boldsymbol{\omega}_i^{(0)} \times (\boldsymbol{\omega}_i^{(0)} \times \mathbf{r}_{i,ci}^{(0)}) \tag{3.51}$$

Expressing Equation 3.51 in the i-th frame gives

$$\mathbf{a}_{c,i} = (\mathbf{R}_{i-1}^{i})^T \mathbf{a}_{e,i-1} + \dot{\boldsymbol{\omega}}_i \times \mathbf{r}_{i-1,ci} + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \mathbf{r}_{i-1,ci}) \tag{3.52}$$

where

$$\mathbf{a}_{e,i} = (\mathbf{R}_{i-1}^{i})^T \mathbf{a}_{e,i-1} + \dot{\boldsymbol{\omega}}_i \times \mathbf{r}_{i-1,i} + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \mathbf{r}_{i-1,i}) \tag{3.53}$$

Hence the Recursive Newton-Euler equations are derived. The fact that there is no link 0 or link $n+1$, is now used to derive the torques and forces for each individual link. Starting with the initial conditions $\boldsymbol{\omega}_0$, $\boldsymbol{\alpha}_0$, $\mathbf{a}_{c,0}$ and $\mathbf{a}_{e,0}$ equal to zero, and then solving the equations for $\boldsymbol{\omega}_i$, $\boldsymbol{\alpha}_i$, $\mathbf{a}_{e,i}$ and $\mathbf{a}_{c,i}$ in that order from $i = 1$ increasing to $i = n$, followed by setting the terminal conditions $\mathbf{f}_{n+1}$ and $\boldsymbol{\tau}_{n+1}$ equal to zero and then solving the equations for the force $\mathbf{f}_i$ and the torque $\boldsymbol{\tau}_i$ in that order from $i = n$ decreasing to $i = 1$, results in the Newton-Euler equations for each link. These expressions can then be re-formulated as Equation 3.29.

3.3.3 Identification of ABB IRB1600 robot equation

In the expressions above, there are a couple of robot parameters that needs to be identified. These variables are the masses of the links, the center of mass of the links and the inertia matrices of the links. These variables were identified and validated in Breistøl [7] and Strandbråten [8] and are given below.

Mass, center of mass and inertia matrices

Table 3.4 shows the masses and the distance to the c.o.m. defined from the links original coordinate frame:

Table 3.4: Masses and center of masses for the robot

Link	Mass [g]	x [mm]	y [mm]	z[mm]
1	104393.43	51.98	-343.26	-11.92
2	20452.12	201.93	-0.15	-182.8
3	50588.7	-11.12	15.28	96.81

The inertia matrices are

$$\mathbf{I}_1 = \begin{bmatrix} 1077815.9 & 1939.4 & 24207.0 \\ 1939.4 & 1010823.7 & -13131.7 \\ 24207.0 & -13131.7 & 225407.6 \end{bmatrix} \tag{3.54}$$

$$\mathbf{I}_2 = \begin{bmatrix} 688775700.03 & -2153419.93 & -83961.29 \\ -2153419.93 & 52950112.68 & -48598599.09 \\ -83961.29 & -48598599.09 & 703024768.84 \end{bmatrix} \tag{3.55}$$

$$\mathbf{I}_3 = \begin{bmatrix} 249444.6 & 0 & 0 \\ 0 & 305986.7 & -0.1 \\ 0 & -0.1 & 88391.5 \end{bmatrix} \tag{3.56}$$

The mass inertia, coriolis/centripetal and gravity matrices are from this calculated using the Newton-Euler approach, as described in Subsection 3.3.2. This modelling were, as mentioned, performed in Strandbråten [8] and Breistøl [7], and their model is reused in this thesis. Some analysis of their expressions were performed, but no questionable errors were found.

Modelled friction

Breistøl [7] and Strandbråten [8] also performed identification of the friction forces in the different joints. As there were small possibilities of obtaining a new identification in this thesis, these results are also reused. This identification was performed running each individual joints at different constant angular velocities, while logging the torque applied. Using this torque and compensating for the centripetal/coriolis and gravity forces, gave the friction force models shown in Figure B.1, Figure B.2 and Figure B.3, and an expression for $\boldsymbol{\tau}_f(\dot{\mathbf{q}})$ used in Equation 3.57. It is worth noting that this function is discontinuous which will be further discussed later.

Including friction forces, the robot dynamics equation becomes

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) + \boldsymbol{\tau}_f(\dot{\mathbf{q}}) = \boldsymbol{\tau} \tag{3.57}$$

3.4 Robot Motion Planning

Trajectory planning for robots is complicated, as assigning positions, velocities and accelerations to the robot needs to be done avoiding collisions and without breaking the robots' physical constraints. It is also important to avoid singularities, which are positions and orientations where the robot lose one DOF, and thus can be described with infinitely many configurations, as described in the end of Subsection 3.2.1. Some basic theory regarding planning of robot motion will therefore be presented in this section. First, a definition of a central concept in robotics, before two different approaches for obtaining trajectories are described briefly.

3.4.1 The configuration space

The configuration space is the set of all possible configurations, where a configuration is a complete set of every joint variable, $q_i, i \in \{1, n\}$, for a robot. To plan a collision free trajectory every configuration along a path must be in the configuration space, and for the robot used in this thesis the configurations space consists of every configuration in the robots' work space, except the space occupied by the 50 cm high plinth at which the robot is mounted.

3.4.2 n-th order trajectory planning

A trajectory is often developed as an n-th order polynomial. Here every joint configuration along a path, $q(t)$ or $q(\theta(t))$, is described as a polynomial of a certain order. For a cubic polynomial the position has the form, [15]

$$q(t) = a + bt + ct^2 + dt^3 \tag{3.58}$$

where t is the time in seconds, and a, b, c and d are motion descriptors. The velocity has the form

$$\dot{q}(t) = \frac{dq(t)}{dt} = b + 2ct + 3dt^2 \tag{3.59}$$

while the acceleration and jerk profiles are the differentiated and double differentiated versions of Equation 3.59. From the equations above one can plan different motions from start, t_0, to finish, t_f, by changing the variables $\{a, b, c, d\}$.

3.4.3 Trajectories based on Fourier Series Expansion

A different way of describing trajectories is based on Fourier Series. The Fourier Series Expansion has the form, [17]

$$x(t) = \frac{1}{2}a_0 + \sum_{k=1}^{\infty}[a_k cos(k\omega_0 t) + b_k sin(k\omega_0 t)] \tag{3.60}$$

Here ω represent the fundamental frequency of the function, while a_k and b_k represents even and odd fourier coefficients respectively. $x(t)$ can be either a joint position q_i or velocity $\dot{q}_i$ or something else depending what is going to be planned. Fourier Series Expansion Trajectories are often used, when trying to minimize the amplitude of the acceleration profile, to for example obtain smooth trajectories, and to plan periodic motions.

Examples of Fourier Series trajectories are Gutman 1-3 and Freudenstein 1-3, [17], which both only include odd fourier coefficients and sine elements for position expressions, and thus only cosine elements for the velocity expressions.

3.5 Inverse Dynamics Controller

The controller used in this thesis is an Inverse Dynamics Controller, as the one presented in Pchelkin et al. [18]. For such a controller the control input of the Dynamics Equation 3.29, $\boldsymbol{\tau}$, has the form

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})[\mathbf{a}(\theta) - \mathbf{K}_p\mathbf{y} - \mathbf{K}_d\boldsymbol{\omega}] + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) \tag{3.61}$$

where the angular position error $\mathbf{y}$, the angular velocity error $\boldsymbol{\omega}$ and the angular acceleration error $\mathbf{a}(\theta)$ for the joints are vectors with elements

$$y_i = q_i - \phi_i(\theta) \tag{3.62}$$

$$\omega_i = \dot{q}_i - \phi_i^{'}(\theta)h(\theta) \tag{3.63}$$

$$a_i = [\phi_i^{''}(\theta)h(\theta) + \phi_i^{'}(\theta)h^{'}(\theta)]h(\theta) \tag{3.64}$$

where θ is the path variable, $h(\theta)$ is the rate of change of the path variable, and $\phi_i(\theta)$ is a function mapping the path variable θ to configuration value q_i. The function mapping used in this thesis will be presented in Subsection 4.1. $\mathbf{K}_p$ and $\mathbf{K}_d$ are the proportional and derivative gain matrices of the controller.

Equation 3.61 can be rewritten to

$$\boldsymbol{\tau} = [\mathbf{M}(\mathbf{q})\mathbf{a}(\theta) + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q})] + [-\mathbf{M}(\mathbf{q})\mathbf{K}_p]\mathbf{y} + [-\mathbf{M}(\mathbf{q})\mathbf{K}_d]\boldsymbol{\omega} \tag{3.65}$$

where $\mathbf{K}_y = -\mathbf{M}(\mathbf{q})\mathbf{K}_p$ and $\mathbf{K}_{\boldsymbol{\omega}} = -\mathbf{M}(\mathbf{q})\mathbf{K}_d$. Inserting 3.65 into the Dynamics Equation 3.29 gives

$$[\ddot{\mathbf{q}} - \mathbf{a}(\theta)] + \mathbf{K}_d\boldsymbol{\omega} + \mathbf{K}_p\mathbf{y} = 0 \tag{3.66}$$

The experimental setup of this controller will be presented later in Subsection 5.2.1.

Chapter 4

Planning the robot movement

The overall purpose of this thesis is to plan a velocity profile, which is energy efficient and constrained in velocity, acceleration and jerk. In order to do this, a path had to be chosen in addition to an expression for the velocity profile along this path, and lastly the optimization problem had to be set up with constraints. This will all be presented in this chapter.

4.1 The path

The path to be optimized were chosen to be a circle oriented in space. Every point on this circle can be expressed by

$$\mathbf{P} = \mathbf{P}_0 + r \cdot \mathbf{R}_y(-\alpha) \cdot \begin{bmatrix} cos(\theta) \\ sin(\theta) \\ 0 \end{bmatrix} \tag{4.1}$$

where

$$\mathbf{P}_0 = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} \tag{4.2}$$

is the center of the circle.

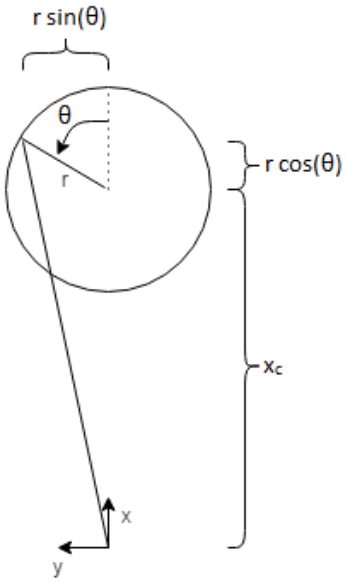

Figure 4.1: View of the x-y-plane with the circle, here α is zero

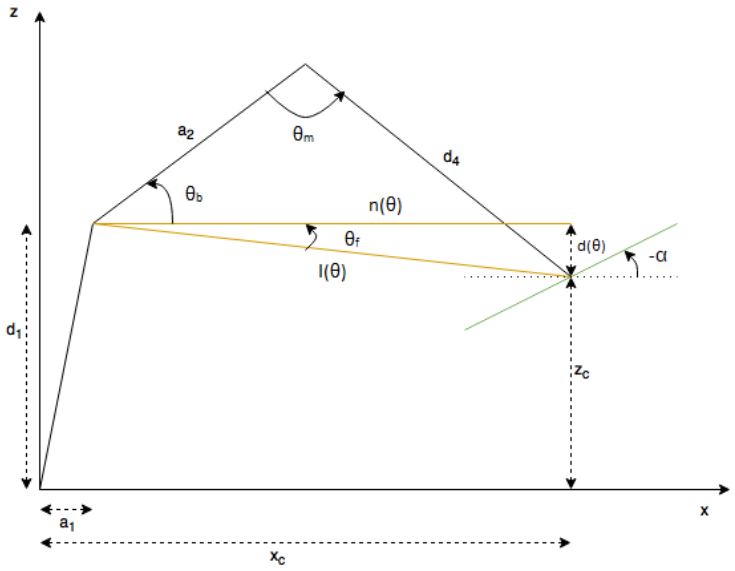

Figure 4.2: View of the x-z-plane where the circle is flipped around local y-axis by $-\alpha$

$$\mathbf{R}_y(-\alpha) = \begin{bmatrix} cos(-\alpha) & 0 & sin(-\alpha) \\ 0 & 1 & 0 \\ -sin(-\alpha) & 0 & cos(-\alpha) \end{bmatrix} \tag{4.3}$$

is a rotation matrix rotating the circle plane around the y-axis with the angle $-\alpha$, see Figure 4.2. A negative α is used as the rotation of the circle plane is defined negative using the right hand rule for the original y-axis. The constant r is the radius of the circle while $\theta = 0 \ldots 2\pi$ is the angle, ie. the path variable, along the circle, see Figure 4.1. The resulting expression for the position of each point along the circle then becomes

$$\mathbf{P} = \begin{bmatrix} x_0 + r \cdot cos(\theta) \cdot cos(-\alpha) \\ y_0 + r \cdot sin(\theta) \\ z_0 - r \cdot cos(\theta) \cdot sin(-\alpha) \end{bmatrix} \tag{4.4}$$

This position is used when calculating the joint configurations through inverse kinematics. The first joint angle is calculated using the x- and y-position as in Section 3.2

$$q_1 = atan2(y_c + r \cdot sin(\theta), x_c + r \cdot cos(\theta) \cdot cos(-\alpha)) \tag{4.5}$$

where y_c and x_c is the center of the circle for the wrist center calculated from Equation 3.11 using $\mathbf{P}_0$ as the position $\mathbf{o}$.

The values of q_2 and q_3 can be found evaluating Figure 4.2. From Pythagoras Theorem for the triangle composed of $l(\theta)$, $n(\theta)$ and $d(\theta)$, we have that

$$l(\theta) = \sqrt{(d_1 - z_c + r \cdot cos(\theta) \cdot sin(-\alpha))^2 + \left(\sqrt{(x_c + r \cdot cos(\theta) \cdot cos(-\alpha))^2 + (y_c + r \cdot sin(\theta))^2} - a_1\right)^2} \tag{4.6}$$

In previous thesises, Breistøl [7], Strandbråten [8] and Røv [19], $y_c + r \cdot sin(\theta)$ was omitted from this expression. This made the horisontal circles look more like eggs, and gave the circle a bend when rotating it.

The next variable to be calculated is the angle θ_m between the second and third link of the robot. Here the law of cosine is used and this gives

$$\theta_m = arccos\left(\frac{a_2^2 + d_4^2 - l(\theta)^2}{2a_2 d_4}\right) \tag{4.7}$$

θ_b is the angle between the vector along $n(\theta)$, and the second link. To calculate this variable, the law of sine is used.

$$\theta_b = arcsin\left(\frac{d_4 sin(\theta_m)}{l(\theta)}\right) - \theta_f \tag{4.8}$$

where

$$\theta_f = atan2\left((d_1 - z_c + r \cdot cos(\theta) \cdot sin(-\alpha)), \left(\sqrt{(x_c + r \cdot cos(\theta) \cdot cos(-\alpha))^2 + (y_c + r \cdot sin(\theta))^2} - a_1\right)\right) \tag{4.9}$$

q_2 and q_3 can now be expressed with θ_b and θ_m

$$q_2 = \frac{\pi}{2} - \theta_b \tag{4.10}$$

$$q_3 = \frac{\pi}{2} - \theta_m \tag{4.11}$$

Orientation

The orientation configuration angles, q_4, q_5 and q_6, can be calculated using Equations 3.26 to 3.28, inserting a fixed orientation. As this thesis only looks at the first three DOFs this will not be done. This also means that in the equations above every d_4 can be replaced with the sum $(d_4 + d_6)$, and $\mathbf{o}_c$ can be replaced by $\mathbf{P}_0$.

Derivatives of the circle motion

Having expressed the joint configurations as functions of the path variable θ,

$$q_i = \phi_i(\theta) \tag{4.12}$$

it is important, in order to plan movements for every joint, to also represent the angular velocities and accelerations as functions of θ. Using the chain rule and the product rule of differentiation, these expressions become

$$\begin{aligned} \dot{q}_i &= \frac{d\phi_i(\theta)}{dt} = \frac{d\phi_i(\theta)}{d\theta}\frac{d\theta}{dt} \\ &= \phi_i'(\theta)\dot{\theta} \end{aligned} \tag{4.13}$$

and

$$\ddot{q}_i = \phi_i''(\theta)\dot{\theta}^2 + \phi_i'(\theta)\ddot{\theta} \tag{4.14}$$

In addition to these expressions, an expression for the jerk of each joint is needed in order to see that the jerk constraints are not violated. The jerk of a joint is the third derivative of the position, and the rate of change of the acceleration. Thus, differentiating Equation 4.14, gives Equation 4.15

$$\begin{aligned} \dddot{q}_i &= \phi_i'''(\theta)\dot{\theta}^3 + 2\phi_i''(\theta)\dot{\theta}\ddot{\theta} + \phi_i''(\theta)\dot{\theta}\ddot{\theta} + \phi_i'(\theta)\dddot{\theta} \\ &= \phi_i'''(\theta)\dot{\theta}^3 + 3\phi_i''(\theta)\dot{\theta}\ddot{\theta} + \phi_i'(\theta)\dddot{\theta} \end{aligned} \tag{4.15}$$

4.1.1 Velocity Profile

The trajectory planning techniques described in Section 3.4 were both dependent on time. This will not be the case here. Since the robot must follow the circle, it is so called path constrained. To plan a motion for such paths and because the controller described in Section 3.5 is dependent on a path variable not time, the trajectory should be dependent on this path variable θ. As seen in the last subsection expressions for $\dot{\theta}$, $\ddot{\theta}$ and $\dddot{\theta}$ are needed.

The choice of the expression for the velocity profile, $\dot{\theta}$, is based on Fourier Series Expansion and looks like

$$\dot{\theta} = h(\theta) = \frac{1}{2}a_0 + \sum_{i=1}^{n-1} a_i cos(i\omega\theta) \tag{4.16}$$

where $i \in 0 \dots n-1$ is the index of the optimizing fourier coefficients a_i, and where n is the order of the velocity profile.

As Equation 4.16 only include cosine elements, and no sine elements as in Equation 3.60, the rate of change of θ along the circle is mirrored in the two half circles (from 0 to π and from π to 2π), which is a wanted behavior for a circle motion. This also reduces the number of optimization variables for the trajectory, which makes the optimization task smaller for the solver. Breistøl [7] also used a velocity profile based on Fourier Series, but this one included sine elements which made the trajectories quite uneven. It was also observed that removing the sines improved the energy efficiency.

Differentiating Equation 4.16 w.r.t. time, results in

$$\begin{aligned}\ddot{\theta} &= \frac{d}{dt}\dot{\theta} = \frac{d}{dt}h(\theta) = h^{'}(\theta)\dot{\theta} = h'(\theta)h(\theta) \\ &= \left(\sum_{i=1}^{n-1}(-i\omega a_i sin(i\omega\theta))\right)\left(\frac{1}{2}a_0 + \sum_{i=1}^{n-1} a_i cos(i\omega\theta))\right)\end{aligned} \tag{4.17}$$

These values for $\dot{\theta}$ and $\ddot{\theta}$ are then used when calculating $\dot{q}_i$ and $\ddot{q}_i$. The triple derivative of θ, which is needed for jerk calculation, is obtained by differentiating Equation 4.17

$$
\begin{aligned}
\dddot{\theta} &= \frac{d}{dt}\ddot{\theta} = \frac{d}{dt}h'(\theta)h(\theta) \\
&= h''(\theta)\dot{\theta}h(\theta) + h'(\theta)h'(\theta)\dot{\theta} = h''(\theta)h(\theta)^2 + h'(\theta)^2 h(\theta) \\
&= \left(\sum_{i=1}^{n-1}(-i^2\omega^2 a_i cos(i\omega\theta))\right)\left(\frac{1}{2}a_0 + \sum_{i=1}^{n-1}(a_i cos(i\omega\theta))\right)^2 \\
&+ \left(\sum_{i=1}^{n-1}(-i\omega a_i sin(i\omega\theta))\right)^2\left(\frac{1}{2}a_0 + \sum_{i=1}^{n-1}(a_i cos(i\omega\theta))\right)
\end{aligned} \tag{4.18}
$$

4.2 Optimization problem

In order to find a best possible trajectory, some optimization is needed. In Nocedal and Wright [20], the general optimization problem is defined as

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x) \\ \text{subject to} \quad & c_i(x) = 0,\ i \in \varepsilon \\ & c_i(x) \leq 0,\ i \in I \end{aligned} \tag{4.19}$$

where $f(x)$ is the function to be minimized, $c_i(x)$ are nonlinear equality constraints when $i \in \varepsilon$ and inequality constraints when $i \in I$, and x is the variable used to minimize f. The following subsections will present the optimization problem used in this thesis, on this form.

4.2.1 Objective function

In order to obtain energy efficient trajectories, the energy consumed along a circle motion has to be minimized. Energy is the integrated version of power over time. Power is defined for a rotational systems as $P_{rot} = |\tau \cdot \omega|$, where ω is the angular velocity. For a robot this angular velocity are the joint rates $\dot{q}_i$.

As seen in the previous sections $\dot{\mathbf{q}}$ are functions of $\dot{\theta}$, which again are functions of the trajectory optimizers a_i. Thus, as seen below, also the energy is a function of these optimizers

$$\begin{aligned} E &= \int \left|\boldsymbol{\tau}\dot{\mathbf{q}}\right| \\ &= \int \left|\boldsymbol{\tau}\boldsymbol{\phi}'(\theta)\dot{\theta}\right| \\ &= \int \left|\boldsymbol{\tau}\boldsymbol{\phi}'(\theta)\left(\frac{1}{2}a_0 + \sum_{i=1}^{n-1} a_i cos(i\omega\theta)\right)\right| \end{aligned} \tag{4.20}$$

The torque, τ, is, as can be seen from the section about Newton Euler and Euler Lagrange, highly dependent on the positional joint angles $\mathbf{q}$, the joint velocities $\dot{\mathbf{q}}$ and the joint accelerations $\ddot{\mathbf{q}}$. Thus changing the trajectory optimizers a_i as well as the positional optimizers, presented in the

next subsection, will not only change the joint rates, but also the torques.

4.2.2 Optimization variables for position and orientation

As mentioned in the beginning, it is wanted to see how the change of position and orientation of the circle can be used to further improve the energy efficiency of the robot. A typical job for the ABB 1600 robot could be to paint, cut or mill a circle in a conveyor belt situation. Something that can be very useful is to know exactly where and how the material with the circle should be placed, in order to consume as low amounts of energy as possible. Three variables are used to describe this placement, and are described in Table 4.1.

Table 4.1: Explanation of optimization variables for position and orientation

Variable	Explanation
α	Describes the orientation of the circle in space, shown in Figure 4.2
x_c	The horizontal position of the center of the circle, with $y_c = 0m$
z_c	The vertical position of the center of the circle

It is chosen not to optimize the position y_c, as changing only x_c will be enough to find an optimal horizontal position. In addition to the angle α, an additional angle γ which describes the rotation of the circle about the x-axis could be optimized. This is however not done in this thesis as it is thought that the rotation with α better represents a realistic conveyor belt situation, where a rotation about the x-axis would mean that the circle plane could be in collision with the belt direction.

In stead of optimizing all three of these variables, it is of course also possible to only optimize one or two of them, depending on the specifications of a production environment. If the position has to be fixed, one could only optimize the angle α, and if the orientation has to be fixed, one could only optimize the position. This way this method could be used to find the best solution for a given production environment.

4.2.3 Constraints

In order to find a feasible trajectory of the robot, it has to be constrained in its configuration space, as well as w.r.t. its limits on angular velocity, angular acceleration and jerk. The joint position constraints look like this

$$q_{i,min} \leq q_i \leq q_{i,max} \tag{4.21}$$

and are found in Table 2.1. The joint rate constraints have a similar look

$$\dot{q}_{i,min} \leq \dot{q}_i \leq \dot{q}_{i,max} \tag{4.22}$$

and are found in Table 2.2, however as the friction experiments performed in Strandbråten [8] and Breistøl [7], were only done for angular velocities $|q_i| \leq 1$rad/s, the angular velocities are constrained below this threshold.

To ensure smooth behaviors the acceleration are set to be below some maximum and above some minimum as

$$\ddot{q}_{i,min} \leq \ddot{q}_i \leq \ddot{q}_{i,max} \tag{4.23}$$

where $\ddot{q}_{i,min/max}$ are the minimum and maximum allowed acceleration, which will be derived later in Section 5.1.1. The jerk of the robot joints must also be between some thresholds

$$\dddot{q}_{i,min} \leq \dddot{q}_i \leq \dddot{q}_{i,max} \tag{4.24}$$

where $\dddot{q}_{i,min/max}$ are the minimum and maximum allowed jerk, which will also be derived later in Section 5.1.1.

The first four constraints controls the physical constraints of the robot. In addition to this, it is important to avoid really slow trajectories to also constrain the time used by the robot. This constraint can be set by one of the trajectory coefficients, namely a_0 which controls the average velocity along the circle. This can be seen when summarizing every value of $\dot{\theta}$ along the circle.

The result, using Equation 4.16, is $\sum_{i=1}^{N} \dot{\theta} = N \cdot \frac{1}{2} \cdot a_0$ where N is the number of steps when θ goes from 0 to 2π. Thus the average velocity is $\frac{1}{2}a_0$. The time constraint is therefore dependent on this variable and become

$$t_{min} \leq t(a_0) \leq t_{max} \tag{4.25}$$

where $t_{min} = 9s$ and $t_{max} = 10s$ represents a chosen time interval that cannot be broken. As the step size of the simulink model, used in this thesis to simulate the circle, is not constant this time constraint is not 100% accurate, but with addition of another constraint it does the job. The problem with the step size of the simulink model, is that it might take long steps at zero velocity to avoid using any energy. Therefore to further help the solver a constraint is also set on $\dot{\theta}$. It is set to be above a certain threshold in order to make sure that the robot never stands still, or for some reason tries to go in a negative direction. The need for this constraint was discovered in [7]. This minimum threshold were chosen to be $\dot{\theta}_{min} = 0.1 rad/s$ giving the constraint

$$\dot{\theta} \geq \dot{\theta}_{min} \tag{4.26}$$

The optimization variables on position and orientation must also be constrained. The angle, α, should be in the range between $\alpha_{min} = 0°$ and $\alpha_{max} = 90°$. It is thought that any angle $\alpha_{90°+}$ above $\alpha = 90°$ will give the same energy contribution as the angle $\alpha = 180° - \alpha_{90°+}$. This constraint therefore becomes

$$\alpha_{min} \leq \alpha \leq \alpha_{max} \tag{4.27}$$

The position of the circle is constrained using the distance, $D = \sqrt{(x_t - a_1)^2 + (z_t - d_1)^2}$, from the center of joint 2 to the end effector. Two points on the circle are of interest. The point where the tool is closest to joint 2 at $\theta = \pi[rad]$, and the point where the tool is furthest away from joint 2 at $\theta = 0[rad]$. The distances to these points are given by the equations below

$$D_{max} = \sqrt{(x_c + r \cdot cos(-\alpha) - a_1)^2 + (z_c - r \cdot sin(-\alpha) - d_1)^2} \tag{4.28}$$

$$D_{min} = \sqrt{(x_c - r \cdot cos(-\alpha) - a_1)^2 + (z_c + r \cdot sin(-\alpha) - d_1)^2} \tag{4.29}$$

and are constrained to

$$D_- \leq D_{max} \leq D_+ \tag{4.30}$$

$$D_- \leq D_{min} \leq D_+ \tag{4.31}$$

where, according to Figure 2.4, $D_- = 495\text{mm} - a_1$ is as close a point on a circle can be and $D_+ = 1225\text{mm} - a_1$ is as far away a point on the circle can be. The DH variable $a_1 = 150mm$ from Table 3.2 is subtracted from the distances in Figure 2.4 because the D_*'s are defined from joint 2, not joint 1. It is also worth noting that these constraints assume that the working range of the robot is perfectly spherical. This is not the case, but as the error is only a few millimeters it is thought to be close enough.

One last constraint is needed to make sure that the optimal path is not planned in collision with the plinth. This constraint is only active when the robot tools' z-position is below zero. When this is the case then the robot tools' x-position must be larger than $300mm$, which is approximately the distance from the robot base to the end of the plinth. This gives us a constraint

$$x_{robot,min} \geq x_{max,plinth} \tag{4.32}$$

whenever

$$z_{robot,min} \leq 0mm \tag{4.33}$$

This constraint and the fact that the trajectory will be planned in front of the robot, makes sure

that the discontinuous part of the spherical work range in Figure 2.4 is not in consideration.

4.2.4 Resulting problem

The resulting problem with the objective function and the constraints is given here:

$$
\begin{aligned}
\min_{a_i,\alpha,x_c,z_c\in\mathbb{R}^n} \quad & E(a_i,\alpha,x_c,z_c) \\
\text{subject to} \quad & q_{i,min} \le q_i \le q_{i,max}, && i \in \{1\dots3\} \\
& \dot{q}_{i,min} \le \dot{q}_i \le \dot{q}_{i,max}, && i \in \{1\dots3\} \\
& \ddot{q}_{i,min} \le \ddot{q}_i \le \ddot{q}_{i,max}, && i \in \{1\dots3\} \\
& \dddot{q}_{i,min} \le \dddot{q}_i \le \dddot{q}_{i,max}, && i \in \{1\dots3\} \\
& t_{min} \le t(a_0) \le t_{max} \\
& \dot{\theta} \ge \dot{\theta}_{min} \\
& \alpha_{min} \le \alpha \le \alpha_{max} \\
& D_- \le D_j(x_c,z_c,\alpha) \le D_+, && j \in \{max,min\} \\
& x_{robot,min} \ge x_{max,plinth}, && \text{if } z_{robot,min} \le 0mm
\end{aligned}
\tag{4.34}
$$

Chapter 5

Optimization and experimental setup

In this chapter the practical details of the optimization problem will be presented, including the chosen solver and the experiments obtaining the bounds for the acceleration and jerk. In addition to this the programming of the external control circle and the ABB circles will be presented.

5.1 Optimization setup

5.1.1 Finding bounds for acceleration and jerk

Torque experiment

As acceleration and jerk bounds aren't provided by ABB, they must be calculated from the bounds put on torque. The torque bounds are not supplied by ABB either, but approximations can be obtained doing some experiments.

These experiments were performed by running each of the first three joints individually at as high velocities as possible. The movements were done in the most outstretched configurations possible to make the load as large as possible for the joint moving. The velocity setting, v7000, which represent a maximum linear velocity of 7000 mm/s, were used in MoveAbsJ instructions programming the robot to go back and forth between two joint configurations. The applied torque was logged using External Control in Submit mode, and plotted using MATLAB. The torques in addition to the measured velocity in the torque experiments are shown in Figures

5.1, 5.2, 5.3, 5.4, 5.5 and 5.6. From these figures the values for the torque bounds were set to

$$\boldsymbol{\tau}_{max} = \begin{bmatrix} \tau_{1,max} \\ \tau_{2,max} \\ \tau_{3,max} \end{bmatrix} = \begin{bmatrix} 533.2552 \\ 1128.6251 \\ 287.2087 \end{bmatrix} [Nm] \tag{5.1}$$

$$\boldsymbol{\tau}_{min} = \begin{bmatrix} \tau_{1,min} \\ \tau_{2,min} \\ \tau_{3,min} \end{bmatrix} = \begin{bmatrix} -526.0914 \\ -1112.8514 \\ -297.8804 \end{bmatrix} [Nm] \tag{5.2}$$

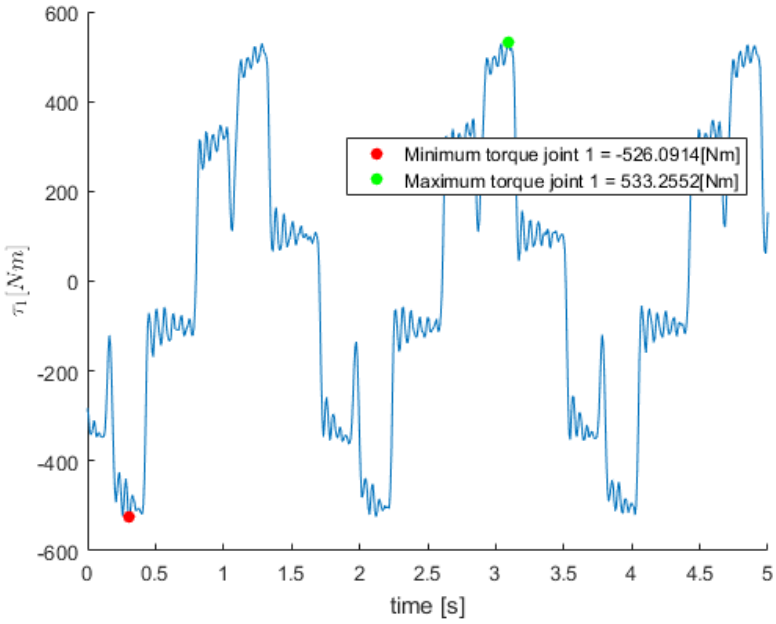

Figure 5.1: Torque experiment joint 1

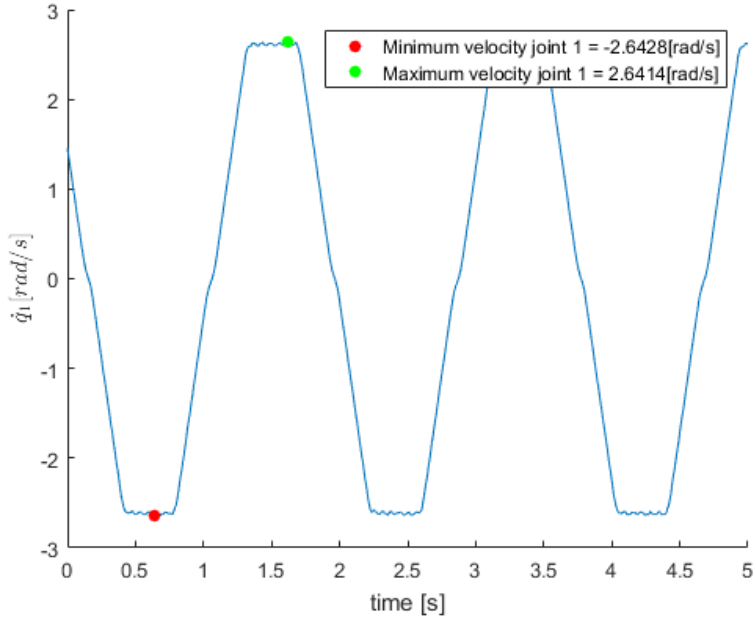

Figure 5.2: Velocity of joint 1 during torque experiment

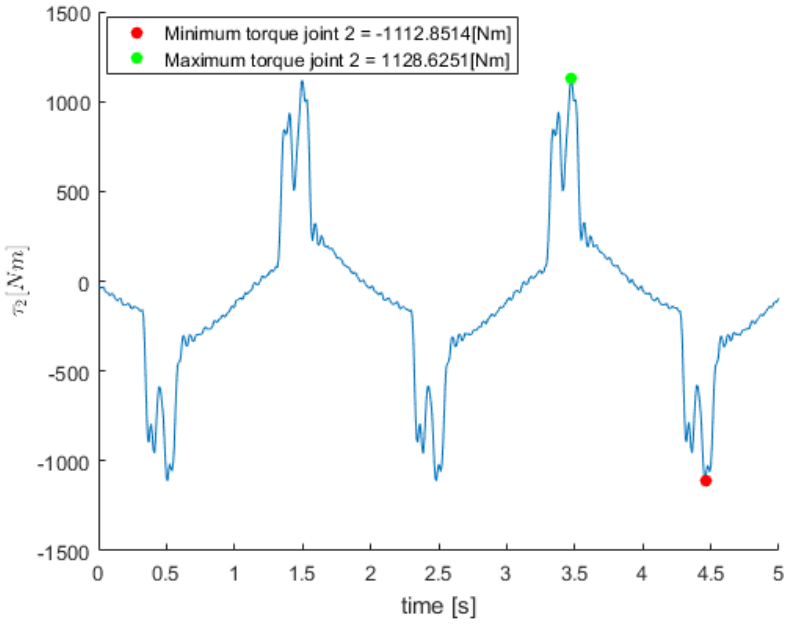

Figure 5.3: Torque experiment joint 2

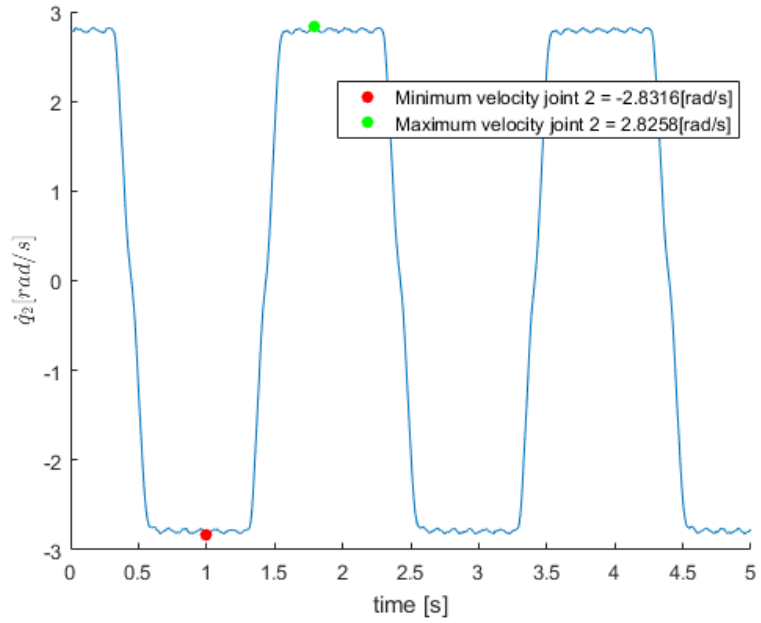

Figure 5.4: Velocity of joint 2 during torque experiment

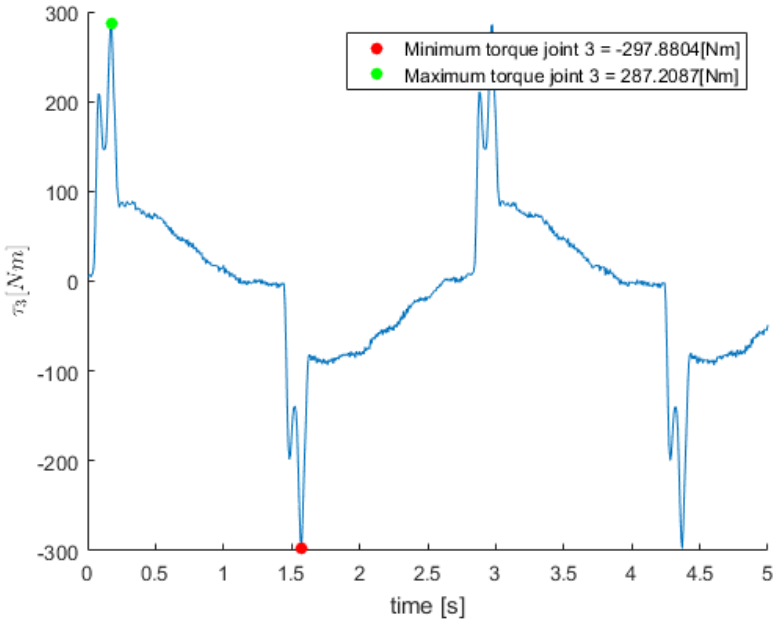

Figure 5.5: Torque experiment joint 3

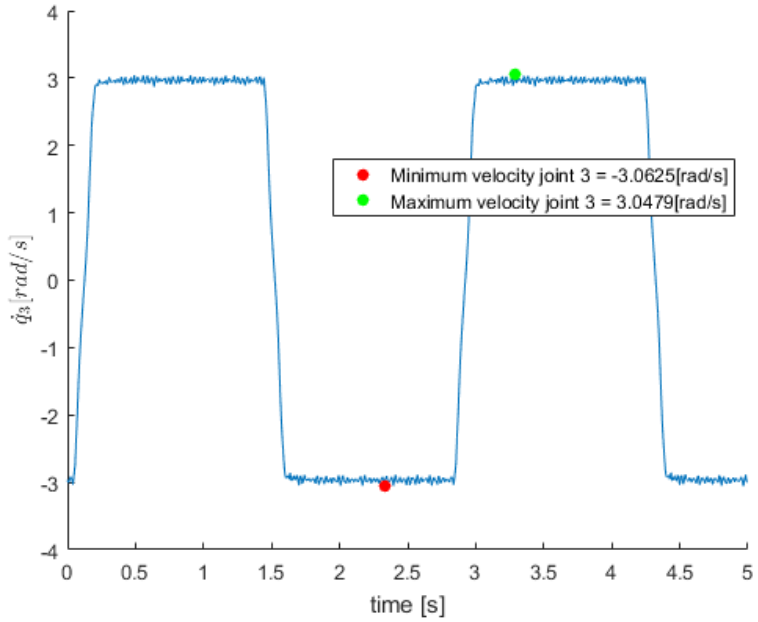

Figure 5.6: Velocity of joint 3 during torque experiment

To see if these values actually represent the torque bounds some analysis is performed. Looking at the figures showing the joint rates during the experiments one can see that the obtained velocities are actually above the maximum and minimum angular velocities for the different joints. For joint 1 the angular velocities of

$$\dot{q}_{1,max} = 2.6414 rad/s$$
$$\dot{q}_{1,min} = -2.6428 rad/s$$

were obtained, which is above the maximum of $150°/s = 2.618 rad/s$ given in Table 2.2. These angular velocities are in the most outstretched configuration equal to linear velocities for the robot end effector of

$$v_{1,max} = (a_1 + a_2 + d_4 + d_6) \cdot 2.6414 rad/s = 1290 mm \cdot 2.6414 rad/s = 3407.4 mm/s$$
$$v_{1,min} = -3409.2 mm/s$$

Since these values are not even half the programmed maximum, in addition to the joint rate being at a maximum this can be interpreted as the maximum and minimum linear velocities the robot is able to perform here. Thus the maximum and minimum torques along this movement can be interpreted as the torque bounds of joint 1.

The same exercise can be done for the second and the third joint. The minimum and maximum angular velocities here were calculated to be

$$\dot{q}_{2,max} = 2.8258 rad/s$$
$$\dot{q}_{2,min} = -2.8316 rad/s$$
$$\dot{q}_{3,max} = 3.0479 rad/s$$
$$\dot{q}_{3,min} = -3.0625 rad/s$$

which again are slightly above the maximum of respectively $160°/s = 2.79 rad/s$ and $170°/s =$

2.967rad/s given for joint 2 and 3 in Table 2.2. This gives linear velocities of

$$v_{2,max} = (a_2 + d_4 + d_6) \cdot 2.8258 rad/s = 3221.4 mm/s$$
$$v_{2,min} = -3228.0 mm/s$$
$$v_{3,max} = (d_4 + d_6) \cdot 3.0479 rad/s = 2026.9 mm/s$$
$$v_{3,min} = -2036.6 mm/s$$

None of these values are close to the programmed maximum of 7000 mm/s either, so one can say that these values represent a highest possible linear velocities, and the torque applied are interpreted as bounds.

Deriving expressions for acceleration and jerk bounds

From the Dynamics Equation 3.57, and the experimental results above, the acceleration bounds are

$$\ddot{\mathbf{q}}_{max} = \mathbf{M}^{-1}(\mathbf{q})(\boldsymbol{\tau}_{max} - \boldsymbol{\tau}_f(\dot{\mathbf{q}}) - \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{G}(\mathbf{q})) \tag{5.3}$$
$$\ddot{\mathbf{q}}_{min} = \mathbf{M}^{-1}(\mathbf{q})(\boldsymbol{\tau}_{min} - \boldsymbol{\tau}_f(\dot{\mathbf{q}}) - \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{G}(\mathbf{q})) \tag{5.4}$$

The jerk can be bounded by the derivative of $\ddot{\mathbf{q}}$ and inserting the torque bounds, giving us

$$\begin{aligned}\dddot{\mathbf{q}}_{max} = &\frac{d}{dt}\mathbf{M}^{-1}(\mathbf{q})\dot{\mathbf{q}}(\boldsymbol{\tau}_{max} - \boldsymbol{\tau}_f(\dot{\mathbf{q}}) - \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{G}(\mathbf{q})) \\ &+ \mathbf{M}^{-1}(\mathbf{q})\left(-\frac{d}{dt}\boldsymbol{\tau}_f(\dot{\mathbf{q}})\ddot{\mathbf{q}} - \frac{d}{dt}\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\ddot{\mathbf{q}} - \frac{d}{dt}\mathbf{G}(\mathbf{q})\dot{\mathbf{q}}\right)\end{aligned} \tag{5.5}$$
$$\begin{aligned}\dddot{\mathbf{q}}_{min} = &\frac{d}{dt}\mathbf{M}^{-1}(\mathbf{q})\dot{\mathbf{q}}(\boldsymbol{\tau}_{min} - \boldsymbol{\tau}_f(\dot{\mathbf{q}}) - \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{G}(\mathbf{q})) \\ &+ \mathbf{M}^{-1}(\mathbf{q})\left(-\frac{d}{dt}\boldsymbol{\tau}_f(\dot{\mathbf{q}})\ddot{\mathbf{q}} - \frac{d}{dt}\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\ddot{\mathbf{q}} - \frac{d}{dt}\mathbf{G}(\mathbf{q})\dot{\mathbf{q}}\right)\end{aligned} \tag{5.6}$$

The expressions for these bounds are calculated using a maple script, see Appendix C. As the equations for the maximum and minimum jerk include a differential of the discontinuous expressions for the friction forces, it was chosen to exclude these constraints. Discontinuous function does not have derivatives at all points, and the discontinuous point of the friction forces are at $\dot{\mathbf{q}} = \mathbf{0} rad/s$, which is a point that will be crossed many times. Since the energy efficient trajectories obtained in this thesis were not to rapid, and the load on the end effector was very small it should not matter. When obtaining time optimal trajectories the values for acceleration and jerk bounds are more of interest.

Some bound were however set on the acceleration to obtain smooth trajectories, given below

$$\ddot{q}_{1,min} = -0.25 rad/s^2 \leq \ddot{q}_1 \leq 0.25 rad/s^2 = \ddot{q}_{1,max} \tag{5.7}$$

$$\ddot{q}_{2,min} = -0.1 rad/s^2 \leq \ddot{q}_2 \leq 0.1 rad/s^2 = \ddot{q}_{2,max} \tag{5.8}$$

$$\ddot{q}_{3,min} = -0.25 rad/s^2 \leq \ddot{q}_3 \leq 0.25 rad/s^2 = \ddot{q}_{3,max} \tag{5.9}$$

These values were chosen by some trial and error, and gave good and smooth trajectories which avoid rapid accelerations.

5.1.2 Penalizing low velocities

In the optimization problem it is chosen to penalize angular velocities below $0.1 rad/s$ as at these velocities unmodelled phenomenas such as dry friction, or stiction, occur and it is wanted to find trajectories which avoids these velocities as much as possible. This problem was identified in Breistøl [7] and updated friction models used to avoid this can be seen in the appendix in Figures B.4, B.5 and B.6.

5.1.3 Solver

The fmincon function of Matlab, [21], was used to solve the optimization problem. Using its Inerior Point Algorithm it searches for a minimum by changing the optimization variables. It will not necessarily find the global minimum, because of the non-convexity of the problem at

hand, but it is able to find local minimas.

fmincon takes in a problem, and what it should minimize for. It also needs some start condition, x_0, for the trajectory optimizers a_i, and the position and orientation optimizers x_c, z_c and α. These have to be chosen with care as it might affect the output, with what local minimum it finds.

A Simulink model is made that, for each suggestion of the coefficients made by fmincon, simultates the entire circle calculating joint angles, joint velocities, joint accelerations, torques and power in order to calculate the energy. This model uses the updated friction model introduced in the last subsection. Thus, this calculated energy is not the true energy. Therefore whenever the fmincon algorithm gives a solution, a second model, using the original friction model, is simulated to calculate the true energy consumption. An additional search is done from these optimal coefficients to see if the algorithm finds a better solution in the vicinity of the obtained solution, which for some start points where the case. The non-linear constraints in the problem 4.34 are also made as a function.

Choice of start point

a_0 must be chosen so that $t_{min} \leq t(a_0) = \frac{2\pi}{\frac{1}{2}a_0} \leq t_{max}$. The rest of the trajectory coefficients are chosen to be zero. The angle α is chosen at random in its feasible area, while the two variables representing position are chosen to be in a position in front of the robot, thus x_c is positive, and in a height $z_c \leq 0.6m$. If z_c is chosen too big, the algorithm might converge to a position straight above the robot, which is not wanted or ideal.

5.2 Experimental setup

5.2.1 Controller setup

The external control controller used to test the circle is an inverse dynamics controller developed by Stepan Pchelkin and Breistøl [7] for his tests. The external control simulink model is further developed in this thesis to be able to handle oriented circles, and to run the new velocity profile. The controller gains worked for the purpose and were not tuned.

Figures 5.7 and 5.8 show the simulink interface of the controller. As mentioned in the beginning C code is generated from this and run through the external control opcom interface.
The functionality of the control block in Figure 5.8 is as follows. It uses the measured joint angles to calculate the position of the end effector using forward kinematics. The angle θ along the circle is then using this position calculated. For the oriented circle this is done by the formula

$$\theta = atan2\left(y \cdot cos(-\alpha_{opt}), x - x_{c,opt}\right) \tag{5.10}$$

When this is done the value of θ is integrated using the expression for the velocity profile $\dot{\theta}$ along the circle. This is done three samples ahead, as there is a delay between input to output of $3 \cdot 0.004032s = 0.012096s$. This value of θ along with the measured joint angles and joint velocities are used to calculate new references for position and velocity of the different joints. These references are chosen from a long list of values for θ, q_i and $\dot{q}_i$ for the optimal trajectory, in the block "DesiredTrajectory".

These references are then sent to the axis computer along with a feed forward torque using the Dynamics Equation 3.57, and the controller gains as described in Section 3.5. The blocks Arm2Motor and Motor2Arm are used to change between motor angles and joint angles, using the gear ratios.

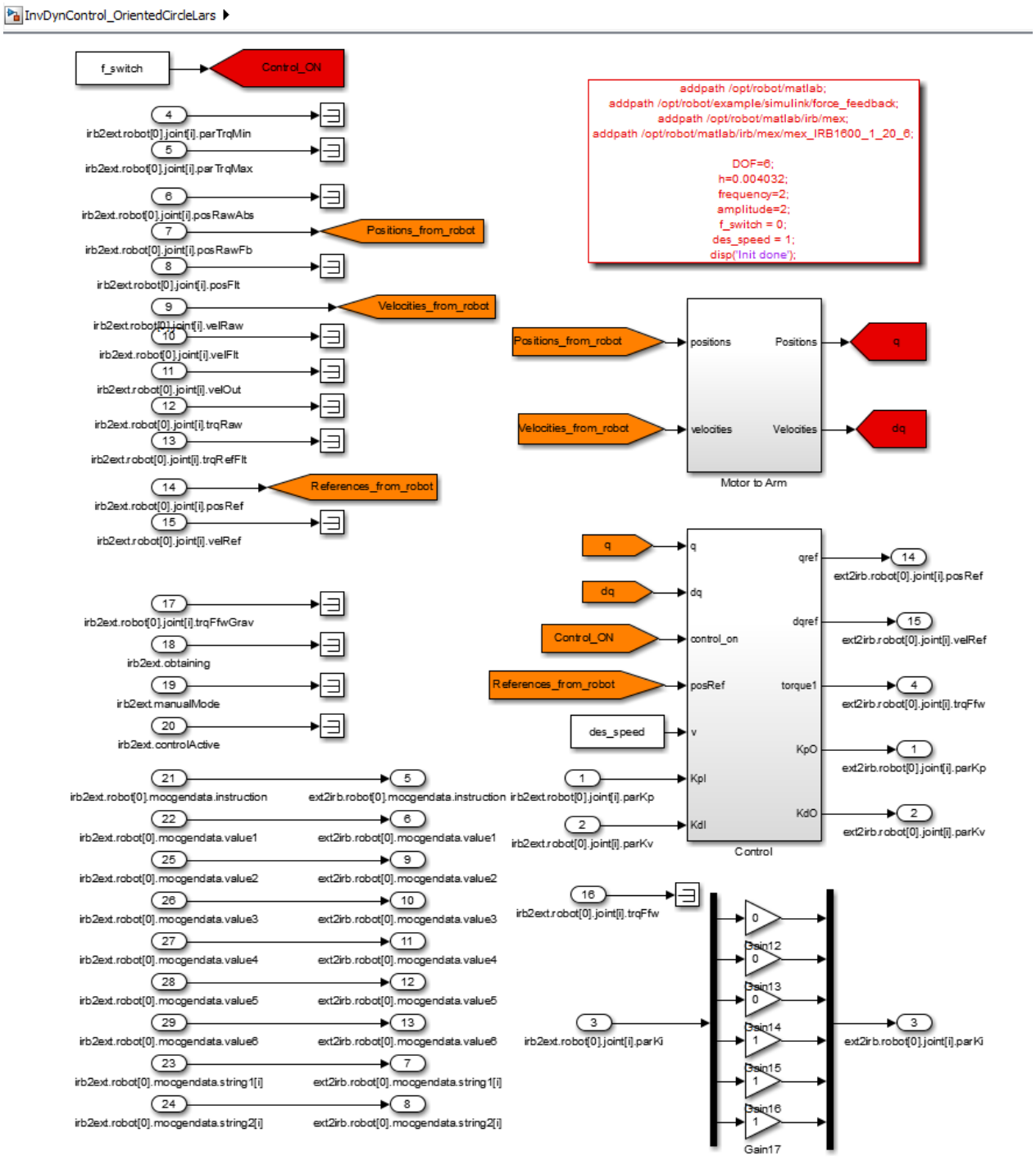

Figure 5.7: Inverse Dynamics Controller in Simulink, used to test the circle in External Control

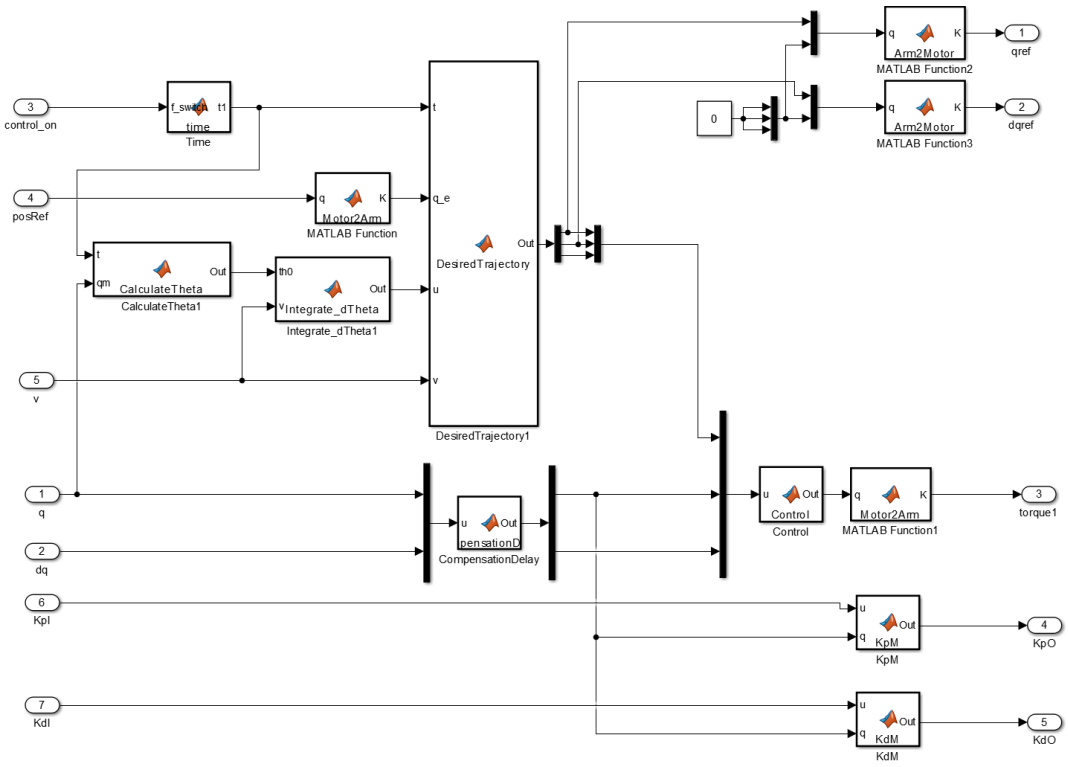

Figure 5.8: Control Block in Simulink, used to test the circle in External Control

5.2.2 ABB RAPID circles

To get a measure of the gain in energy efficiency from the optimized external control trajectory, it was compared to some norm created by the ABB motion planner. Two such norm trajectories were created, and they can be seen in Figure 5.9. The first norm were a circle motion in the same position and plane as the circle obtained in the optimization problem. The second norm were a circle motion with the same center point, but in the horizontal plane. This way it was possible to see both how the optimized trajectory coefficients a_i's improved the energy consumption, and also how the changing of the plane of the circle changed the energy consumption.

The RAPID circles were programmed using two MoveC instructions, each requiring two points; one end-point of the movement, and one intermediate point that the circle motion will be planned through. These points were calculated at $\theta = 90°$ and $\theta = 180°$ for the first half circle, and at $\theta = 270°$ and $\theta = 0°$ for the second half circle, shown in Figure 5.9. The path was programmed to use 10 seconds around the circle, thus each half circle use 5 seconds. A maximum TCP velocity of 100mm/s and zone data z10 were chosen.

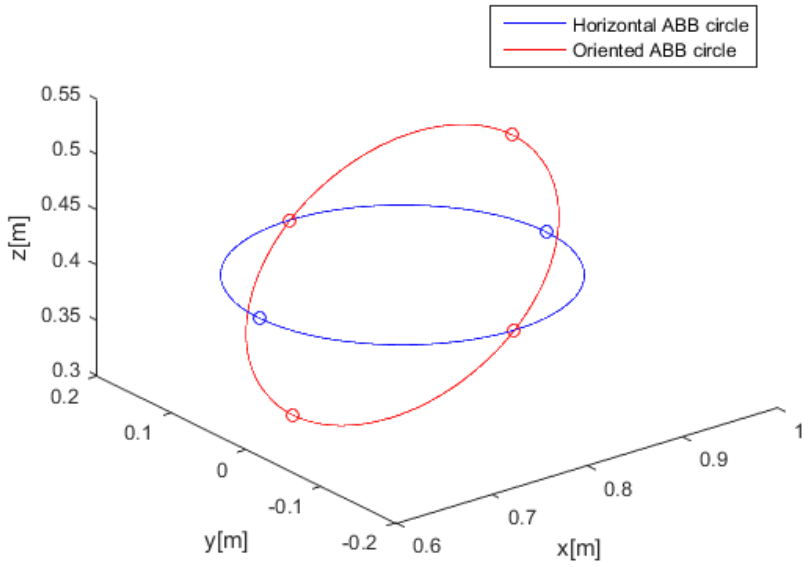

Figure 5.9: The two ABB circles

Chapter 6

Results

When optimizing three different types of circles were tried. Two of these circles optimized position and orientation, where one had a radius of 10cm and the second had a radius of 15cm. The third circle had a radius 15 cm, but here the orientation $\alpha = 0$ were kept fixed. This was to make sure that the optimization algorithm didn't just end up at the same point.

For all the circles the optimization algorithm converged to different small areas of the variables α, x_c and z_c, depending on the allowed time of execution. For the circle with radius 10 cm, with execution time between 5 and 6 seconds, it was $\alpha_{opt} \approx 34^\circ$, $x_{c,opt} \approx 0.86m$ and $z_{c,opt} \approx 0.25m$. For the circle with radius 15 cm, with execution time between 9 and 10 seconds, it was $\alpha_{opt} \approx 40^\circ$, $x_{c,opt} \approx 0.76m$ and $z_{c,opt} \approx 0.4m$. And for the fixed orientation circle, with execution time between 9 and 10 seconds it was at the position $x_c \approx 0.68m$ and $z_c \approx -0.15m$. These values were obtained as long as the start point were chosen as described earlier. It can therefore be said with some confidence that these areas are optimal for the different circle movements. The optimal trajectory coefficients differed a bit, but the resulting energy was not very different from time to time.

For all these approaches the velocity of the second joint was minimized. As can be seen in the next section the velocities for joint 2 proposed by the optimization algorithm are below the velocities to be penalized by the updated friction models. The only movement in joint 2 is the minimum requirement to correct the circle motion. These low velocities makes sense as can be

seen from the friction models, the inertia matrices and the maximum torque experiments. The torque needed to move the second joint is much bigger than for the other two joints. Therefore the energy consumption of this joint is also much bigger.
As the inertia of the second joint is so much larger than of the other two joints, it should be easier to control for small movements. It is therefore chosen to go on with this behavior.

6.1 Optimization results

The oriented circle of radius 15 cm were tested in an experiment, therefore these optimization results are shown below in Figures 6.1, 6.2, 6.3 and 6.4. An optimal path for the 10cm circle is shown in Appendix B in Figures B.7 and B.8, and the optimal horizontal path is shown in Figures B.9 and B.10 in the same appendix.

The torques for this circle, are as can be seen in Figure 6.3, calculated to be much lower than the maximum and much higher than the minimum obtained in the torque experiments. Therefore the bounds for acceleration and jerk are not broken.

The calculated energy were $E = 70.3751J$. This was obtained using a fifth order trajectory, with coefficients $a_{opt} = [1.4048, 0.0779, -0.2354, 0.0319, 0.0081]$, $x_{c,opt} = 0.7631m$, $z_{c,opt} = 0.4154m$ and $\alpha_{opt} = 40.3345°$. It was also tried to find an optimum using a seventh order trajectory, but this gave no improvement.

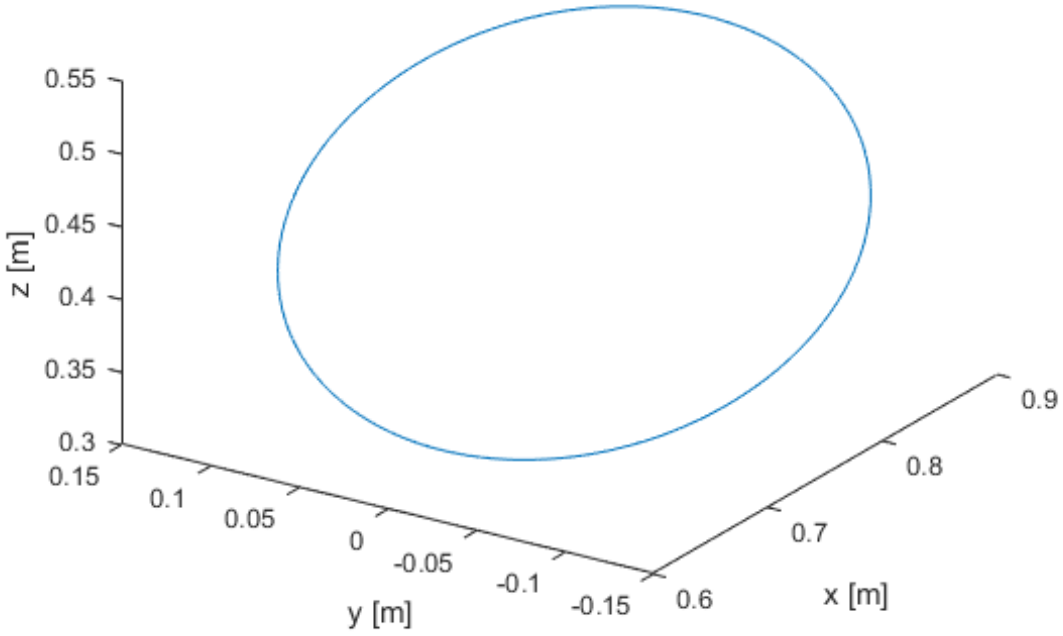

Figure 6.1: The circle path at optimal position $\alpha = 40.3345°$, $x_c = 0.7631m$ and $z_c = 0.4154m$

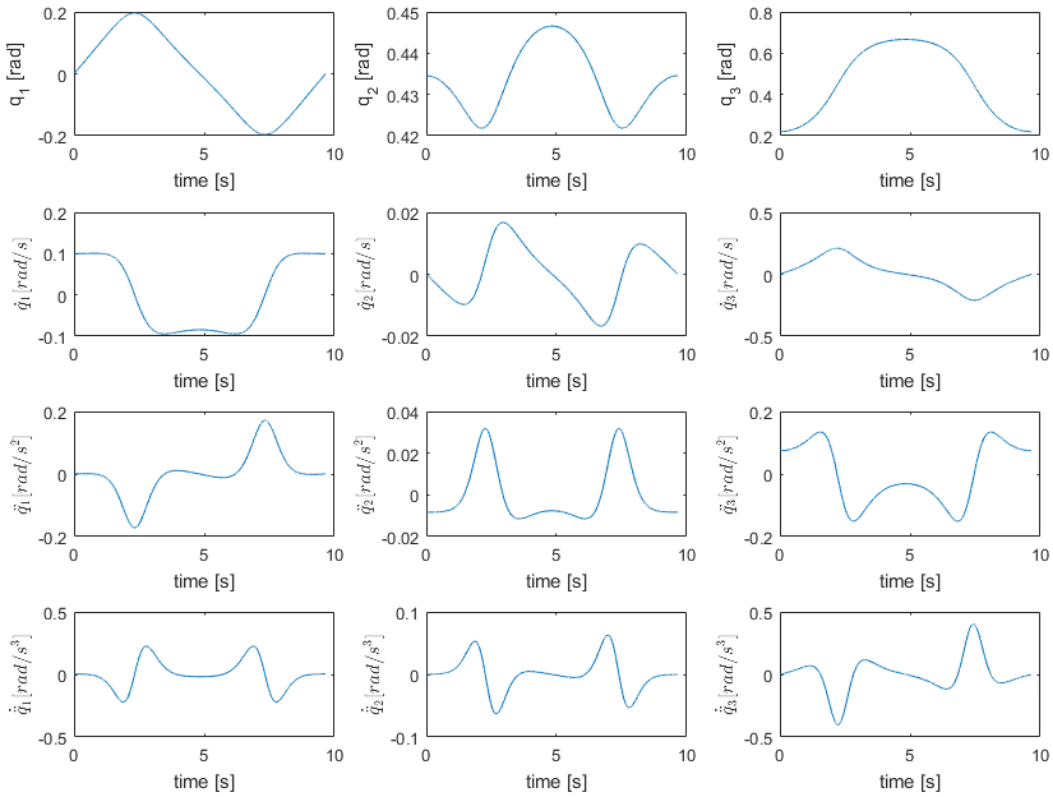

Figure 6.2: Angular position, velocity, acceleration and jerk at optimal position

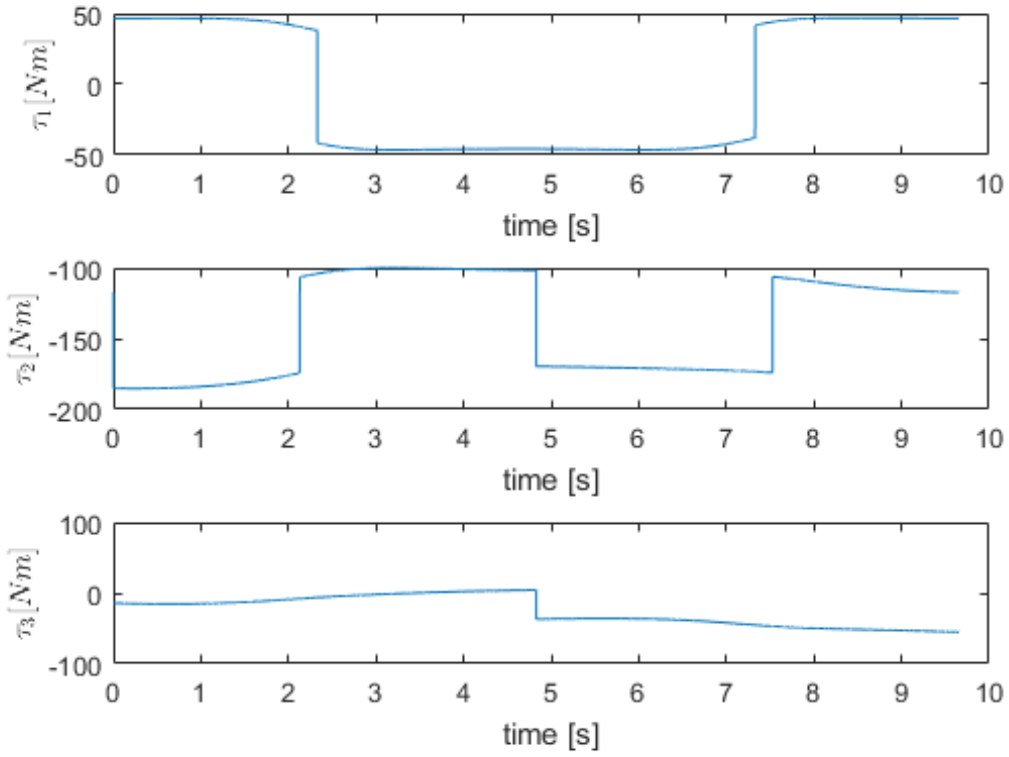

Figure 6.3: Torque calculation for joint 1, 2 and 3

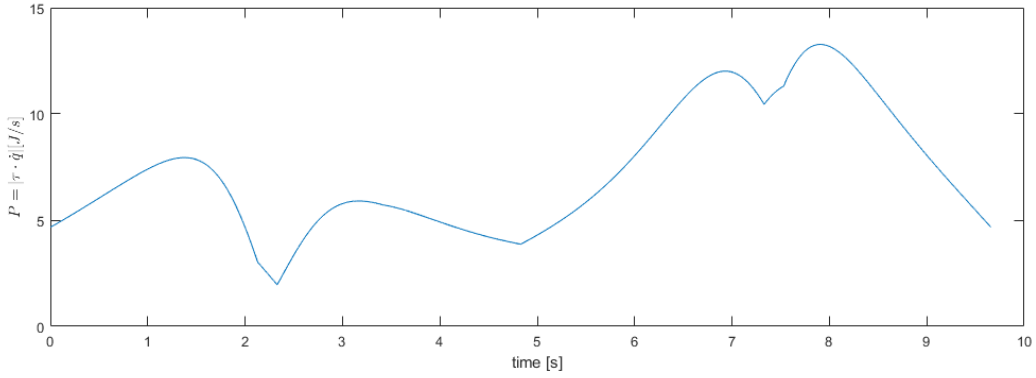

Figure 6.4: Power comsumption along trajectory

6.2 Experimental results

The circle motion obtained by the optimization problem was tested using external control, and the controller seen in Figures 5.7 and 5.8, and were compared to two circles made by ABBs own motion planner. The positions, velocities and torques were logged using an external control logging program, and the values of **irb2ext.robot[0].joint[i].posRawFb**, **irb2ext.robot[0].joint[i].velRaw** and **irb2ext.robot[0].joint[i].trqRefFlt** were read by a MATLAB script. The power along the circle path were calculated by the absolute value of the product between the velocities of the joints and the torques applied.

6.2.1 Optimized path

In Figures 6.5, 6.6 and 6.7 the experimental trajectory, torque and power consumption for the optimal path are shown.

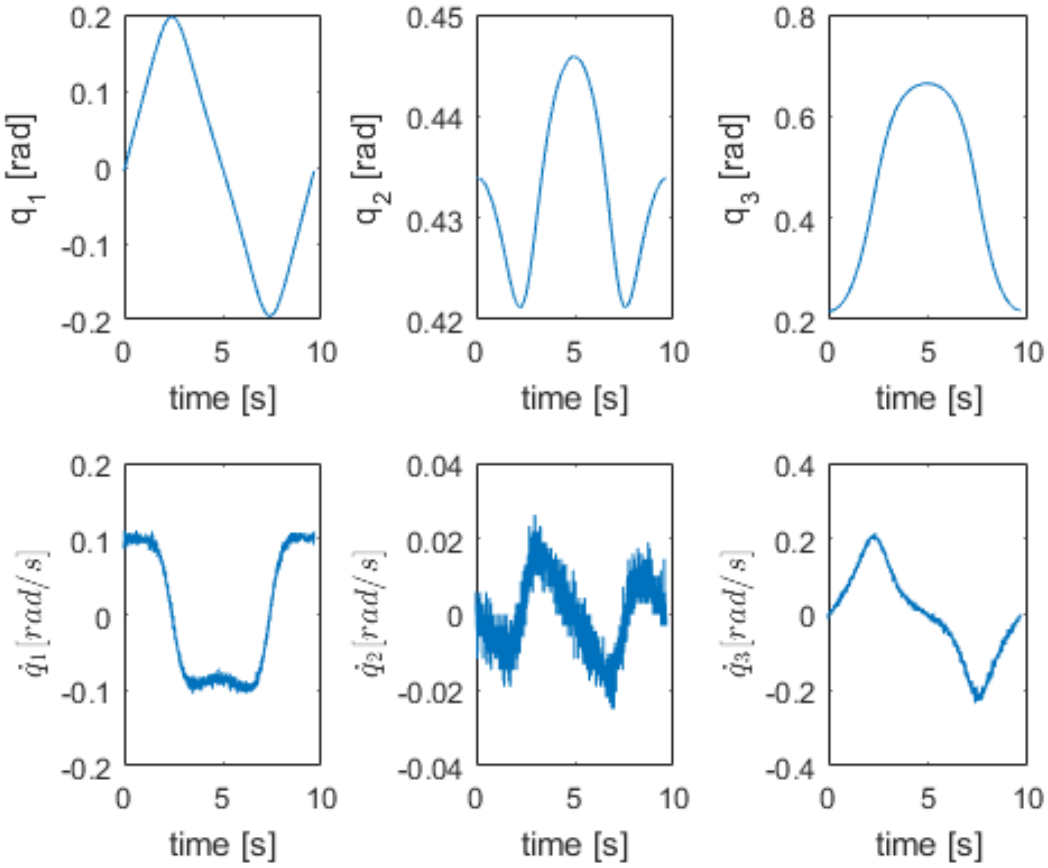

Figure 6.5: Angular position and velocity during experiment

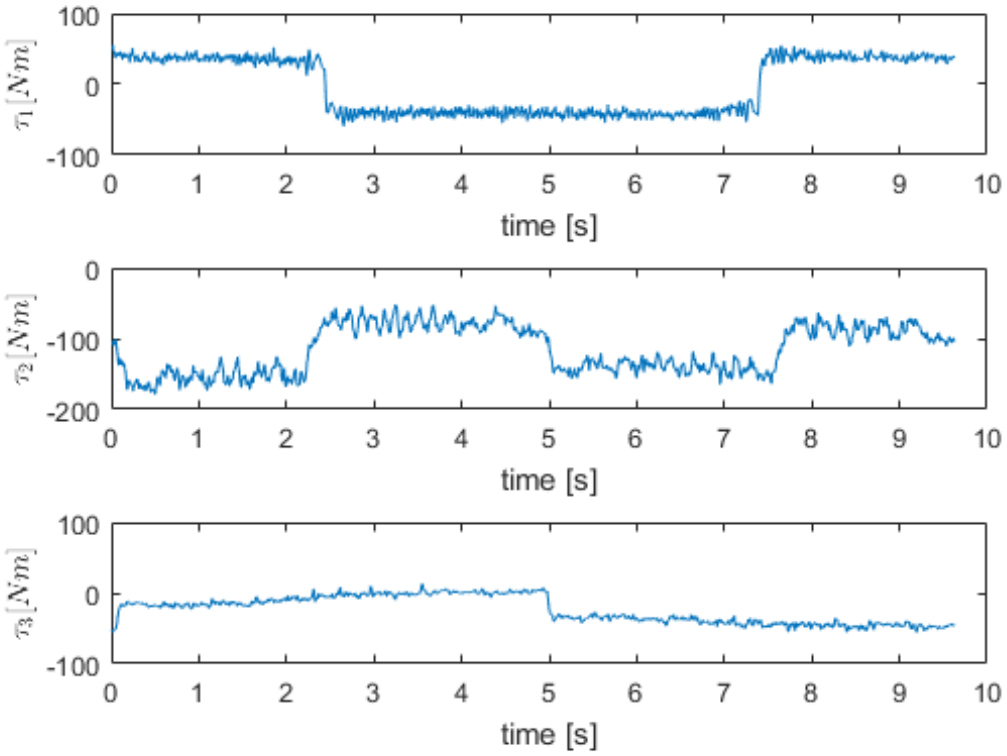

Figure 6.6: Torque calculation for joint 1, 2 and 3 during experiment

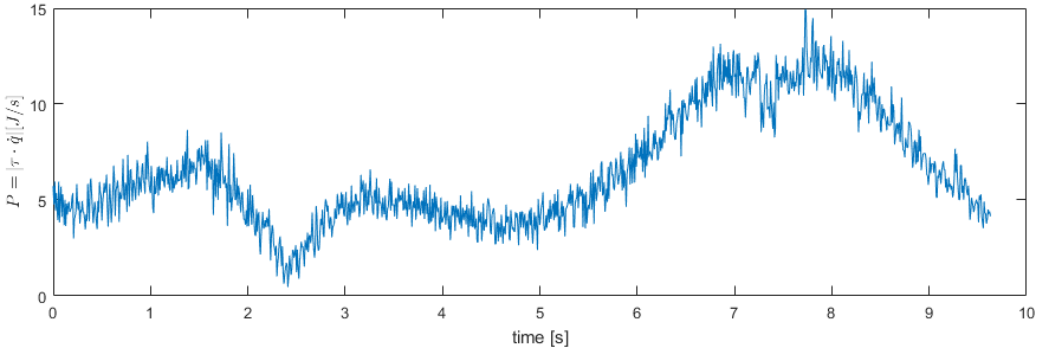

Figure 6.7: Power comsumption during experiment

Comparing these experimental results with the theoretical results, it is easy to see that trajectory is followed. The torque and power are a bit off, and the calculated energy of the optimal circle is $E = 62.6668J$ which is approximately 12.3% less than the theoretical value.

Reasons for this offset might be some error in the model parameters, unmodelled phenomenas and the fact that the control torque does a job that couldn't be calculated theoretically. The DH parameters, masses, c.o.m.s and inertia matrices were all obtained using ABBs general information for the ABB 1600 robot. These data sheets and CAD models does not take into account small differences between each produced robot. Therefore some offset must be expected between theory and experiment.

6.2.2 ABB oriented circle

In Figures 6.8, 6.9 and 6.10 the trajectory, torque and power consumption of the oriented ABB circle are shown. Here it is worth noting that there is some offset between this path and the external control path, as it is not possible to program paths in RAPID without movement in joints 4, 5 and 6. It is also not possible for joint 5 to have angle $q_5 = 0°$, as this will result in a wrist singularity, see Section 3.2. Therefore when programming this circle, joints 1, 2 and 3 will not have the optimal positions to compare with the circle programmed with External Control. It was however tried to make this difference small.

The calculated energy of the oriented ABB path is $E = 72.8885J$ which is approximately 16.3% more than the path programmed with External Control.

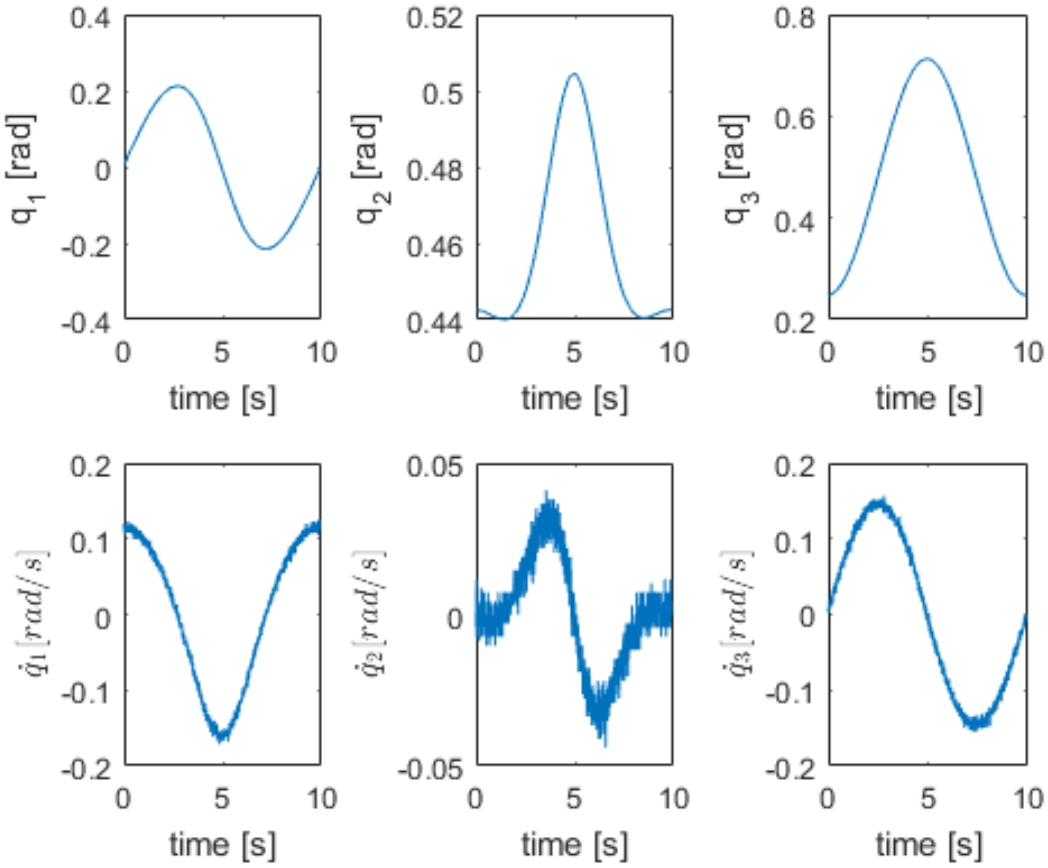

Figure 6.8: Angular position and velocity from ABBs motion planner in oriented circle plane

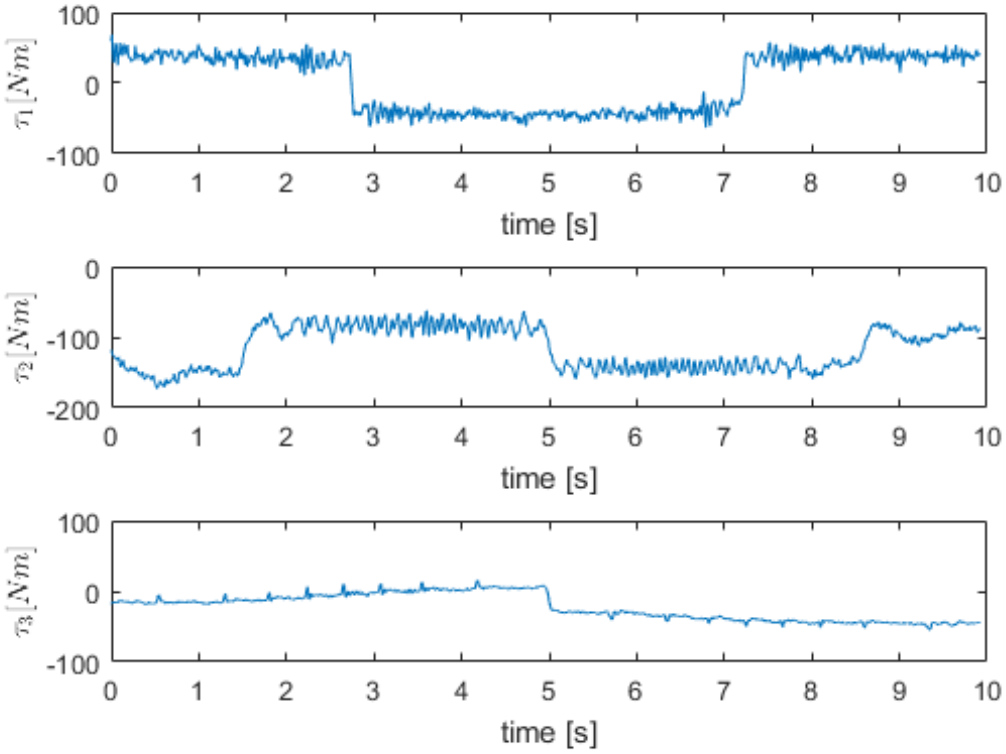

Figure 6.9: Torque calculation for joint 1, 2 and 3 for ABBs oriented circle

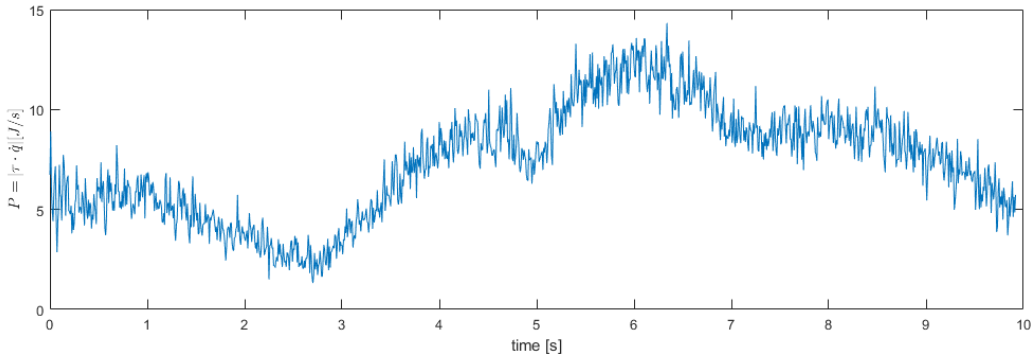

Figure 6.10: Power comsumption for ABBs oriented circle

6.2.3 ABB horisontal circle

In Figures 6.11, 6.12 and 6.13 the measured values of the horisontal ABB circle are shown. This is where the results become interesting. As can be seen from the power consumption and the resulting energy calculation, $E = 163.6110J$, this trajectory consume about 161.1% more energy than the optimal trajectory and about 124.5% more than the oriented ABB circle, which means that by simply rotating the plane of a manipulation task it is possible to more than half the energy consumed by the robot.

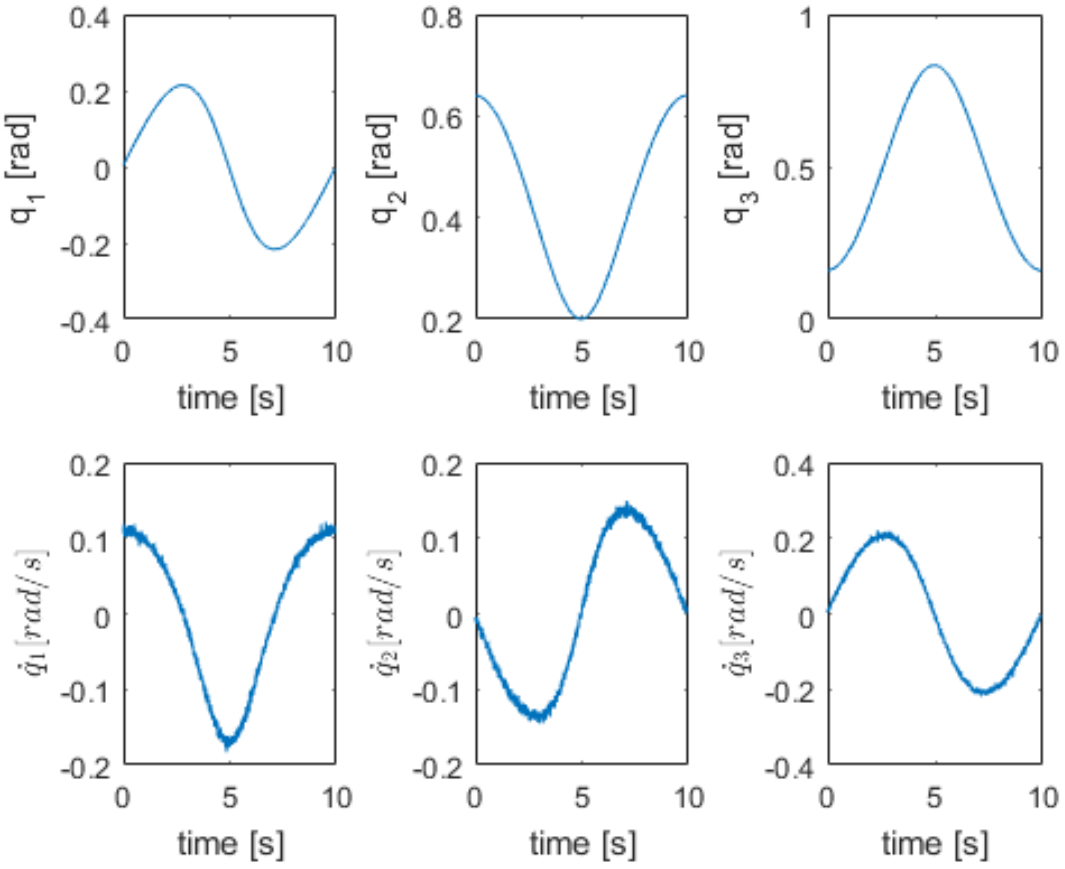

Figure 6.11: Angular position and velocity from ABBs motion planner in horisontal circle plane

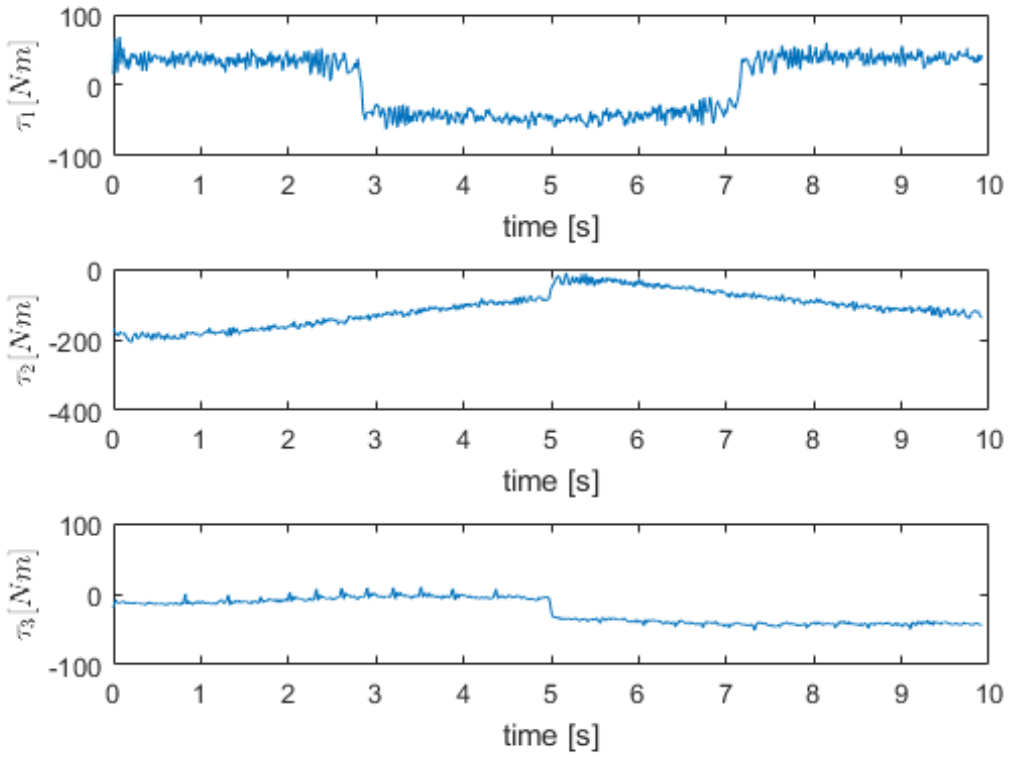

Figure 6.12: Torque calculation for joint 1, 2 and 3 for ABBs horisontal circle

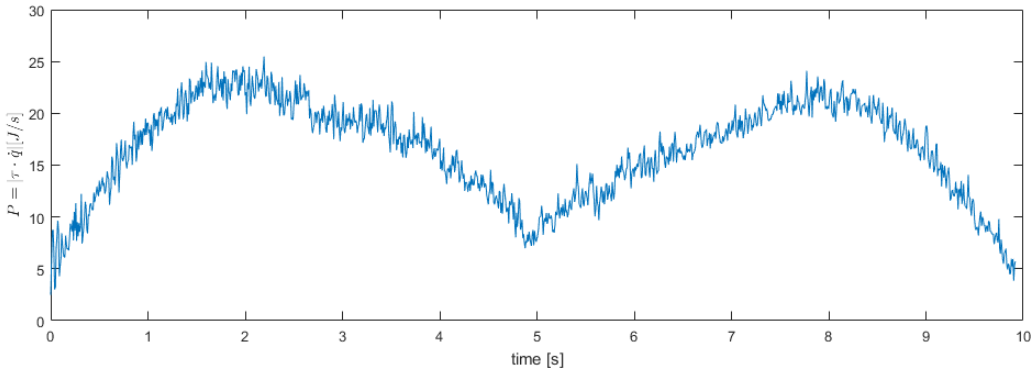

Figure 6.13: Power comsumption for ABBs horisontal circle

Chapter 7

Summary, Discussion and Further Work

7.1 Summary and Conclusion

During the course of this thesis a method for planning of an energy efficient velocity profile for an ABB IRB 1600 robot has been developed. Unlike previous approaches the method used in this thesis does not only look at the velocity and acceleration assignment, but also finds the optimal position and orientation of a manipulation task. This is an approach that can be used for businesses when setting up production.

To plan these motions an optimization scheme was developed where a path is simulated for different choices of optimization variables, returning the consumed energy. This energy is minimized by an optimization algorithm. The optimization scheme needed to be constrained so that a path couldn't be planned outside the feasible area of the robot, or in collision with the plinth of the robot. In addition to this, constraints were put on the acceleration of the robot in order to obtain smooth trajectories and time to avoid really slow trajectories.

Experiments were performed to find the maximum and minimum torque possible of the first three joints of the robot. This was supposed to be used when constraining acceleration and jerk. These constraints were however excluded from the optimization scheme as the energy efficient trajectories obtained were nowhere close to the torque bounds. These experimental results are however greatly of interest when looking at time-optimal trajectories, and can be reused for this

purpose.

The main results of this thesis shows that an optimization of position and orientation of a manipulation task can improve the energy consumption by at least 161.1%, and that by using the velocity profile obtained by Section 4.2.4s optimization scheme energy consumption can be reduced by approximately 16.3%. Analysing the results of the optimization algorithm compared to those trajectories made by ABBs motion planner, it is seen that minimizing the movement of joint 2 is the biggest contributor to the minimization of energy, as the movements of joint 1 and 3 do not differ much in the three approaches.

In conclusion I would say that even though my contribution to the problem of energy effective trajectory planning might seem small, the effect of doing this type of optimization of the position and orientation of the robot movement is big. This type of analysis can be used for businesses the world over to find a plane for a manipulator task that use the second joint as little as possible, and in this way use as little energy as possible.

7.2 Discussion

The results presented in the previous chapter are quite strong. Even though the optimization scheme presented in Chapter 4 does not provide the same solution for every start point, it does provide paths in the same area, with close to similar energy consumptions, as long as the start point is chosen with some analysis.

The results show the huge impact just rotating the plane of the manipulation task have on the energy consumption, no matter how sub-optimal the solution is. Therefore, such an analysis could be useful for production environments. Of course such a moving and rotation of material could mean big changes in existing production environments, but for new businesses setting up production it is useful to both save money and to protect the environment.

The sub-optimality

The fact that the solution differs for every new start point can be explained by the non-convexity of the problem. A non-convex problem have many local minimas, and with so many optimization variables changing just one of these variable by a small increment, could force the optimization algorithm to search in another area. This again can result in a small difference in local minimum. Also, it makes perfect sense that a change in position and orientation would result in a new "optimal" velocity profile, and vice versa.

Improvement from previous thesis

In addition to showing how much can be gained by rotating and moving a manipulation task, an improvement of the 16.3% has been performed relative to the ABB motion planner. In Breistøl [7] a 6.98% improvement was achieved.

Reasons for this can be split into three parts. First of all the modelling of the circle in the previous thesis was wrong and the robot drew an egg shape instead of a circle. The movements along this egg shape differed a lot. In the pointy top of the egg shape accelerations and velocities were generally much higher, and thus more energy inefficient, than in the flat bottom of the egg shape. A second improvement is the expressions for the velocity profile. In the previous approach a fourier series velocity profile using both cosine and sine parts were used. This lead to less smooth behaviour than obtained using only the cosine parts. Lastly, the acceleration bound put in the optimization problem in this thesis removed the risk of costly behavior.

What can be gained by this approach?

To calculate how much money that can be saved from this approach, Bryan et al. [22] say that an industrial robot in USA on average consume approximately 300 kWh every day. In Norway, the average price for electricity in December 2016 were around, 30.5 øre/kWh, [23]. The price of running a robot for a day is therefore 91.5 kroner. For a year running everyday, the price be-

comes 33397.5kroner/robot. Let's say that this is a norm for the horizontal movement. Table 7.1 shows a comparison of running a robot using the three approaches seen in this thesis. Note that this comparison, assume that every movement done by the robot have the same energy efficiency, and that all electricity are used on movements, which is not the case. This comparison is regardless of this, good in order to see the potential of energy efficient robotics.

Table 7.1: Cost comparison

Circle	Electricity cost per year [23]
ABB horisontal	33397.50 kroner/robot/year
ABB oriented	14876.39 kroner/robot/year
External Control	12791.08 kroner/robot/year

To get a look of the environmental gains of this approach, US Environmental Protection Agency [24], calculates the amount of emission due to green house gases per kwh consumed in the US. It claims that a robot using 300 kwh per day is responsible for an emission of 77 metric tons of CO_2 every year. This amount of electricity generation also corresponds to the emission of green house gases, obtained by driving 16.3 passenger vehicles for a year. The reduction of energy consumption can therefore be of great interest in this aspect also, and therefore a reduction 161.1% is a big improvement.

7.3 Future Work

There are several approaches that can be used in order to further improve the subject of energy efficient trajectory planning.

A first short-term approach is to add a rotation matrix about the x-axis, to see how this will affect the energy consumption. It is believed that, as the rotation about y lead to a minimization of the movement in the second joint, a rotation about x will lead to a minimization of the movement of the first joint. The matrix for a rotation with, γ, about x is

$$R_x(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\gamma) & -sin(\gamma) \\ 0 & sin(\gamma) & cos(\gamma) \end{bmatrix} \tag{7.1}$$

and this could be multiplied with the rotation about y in Equation 4.1 in order to model the new circle. Constraints on γ would also need to be added. As the second joint is the joint consuming the most energy, the improvement of this approach should not be as big, but could be of interest nevertheless.

It could also be interesting to see the effect this approach have on paths with corners, such as a square. Here the constraints on acceleration will be more effective as the robot would have to stop at each corner unlike the circle movement used in this thesis.

To further improve the accuracy and validity of the method and optimization scheme, some calibration of the robot parameters can be performed. Kolyubin et al. [25] presents a technique for improving a model without relying on inaccurate CAD data etc. This could also be done on the ABB 1600 robot.

A long-term approach would be to extend the work done here to 6DOFs. This require a lot of work especially as friction experiments will have to be performed for the last three joints. This isn't as straight forward as the experiments performed for the first three joints, as the last three joints are controlled by the same motor. It is therefore difficult to know exactly how much torque is required to control one specific joint when several joints are moving together. A second difficulty here is that when adding three more joints then the recursive equations for the robot dynamics would become much larger. Lastly, since the last three joints have much smaller inertias, controlling these would lead to some difficulties and tuning the gains for the controller could be quite extensive.

Appendix A

Abbreviations

DH = Denavit-Hartenberg

DoF = Degree of Freedom

TCP = Tool Center Point

C.O.M. = Center Of Mass

$c_{*_i} = cos(*_i)$

$c_{*_i} = sin(*_i)$

Appendix B

Additional plots

Friction model

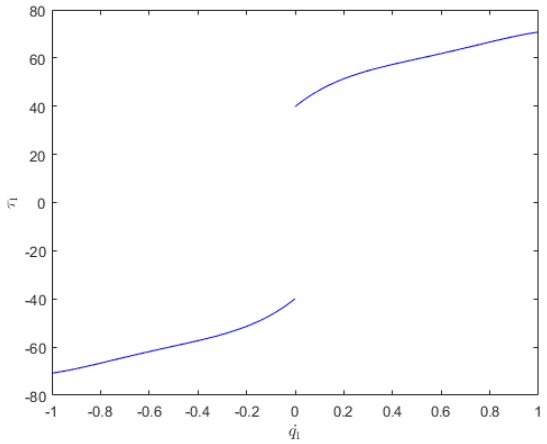

Figure B.1: Friction model for joint 1 obtained in [7] and [8]

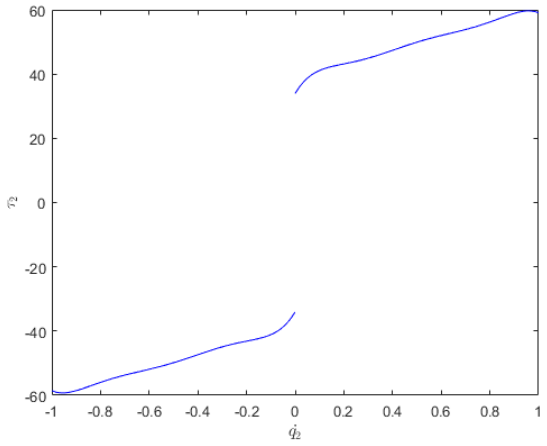

Figure B.2: Friction model for joint 2 obtained in [7] and [8]

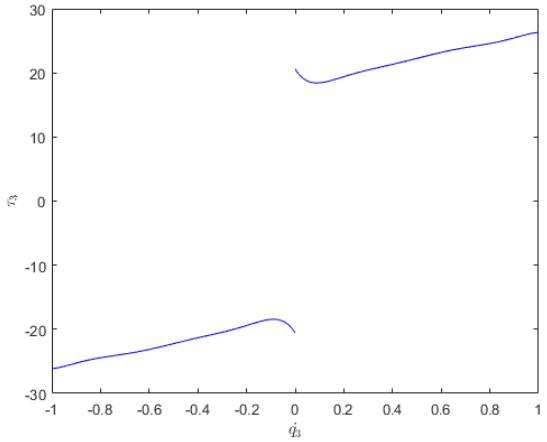

Figure B.3: Friction model for joint 3 obtained in [7] and [8]

Updated friction model

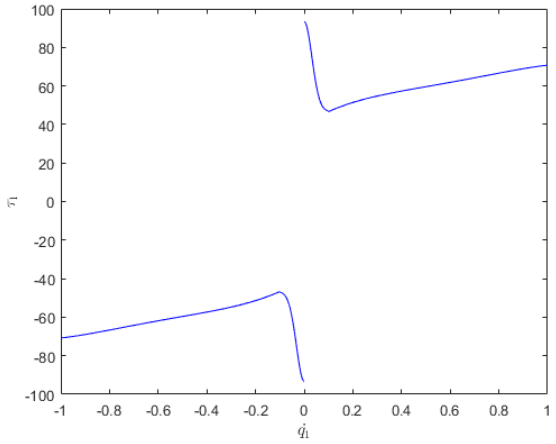

Figure B.4: Friction model used to penalize low velocities for joint 1 obtained in [7]

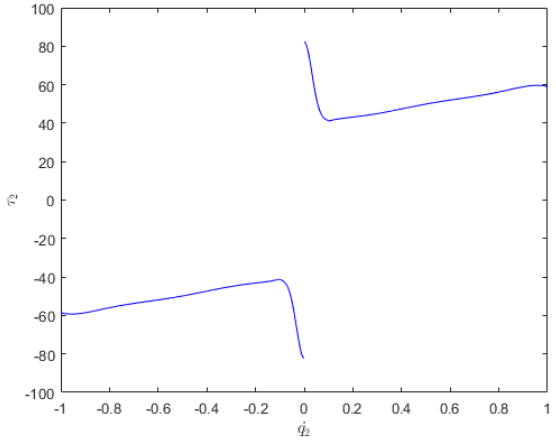

Figure B.5: Friction model used to penalize low velocities for joint 2 obtained in [7]

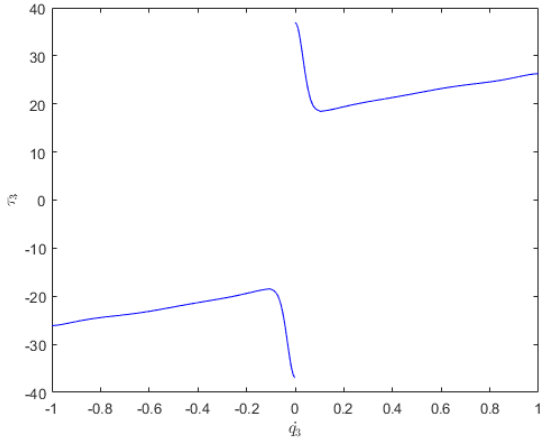

Figure B.6: Friction model used to penalize low velocities for joint 3 obtained in [7]

Optimal path 10cm circle

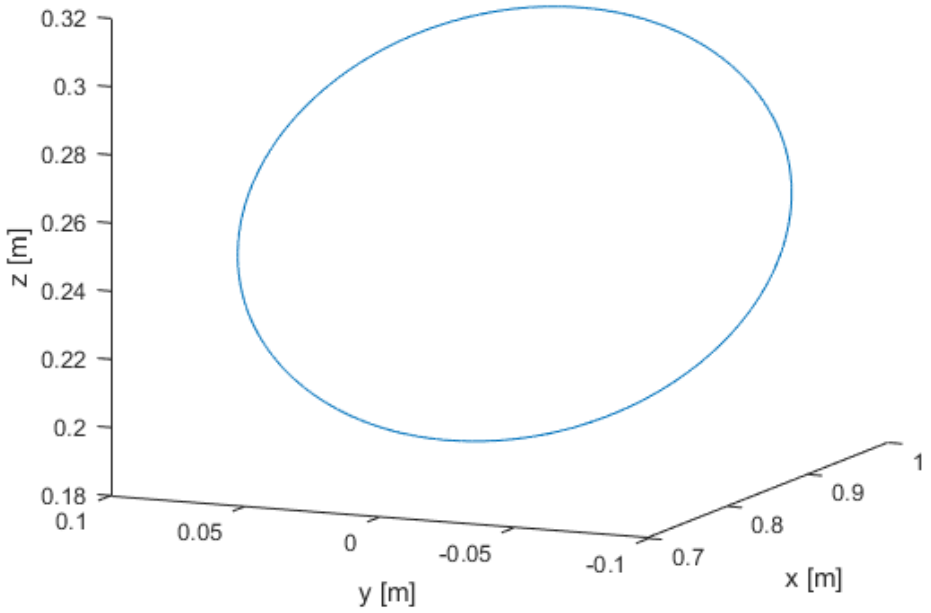

Figure B.7: The circle path at optimal position $\alpha = 33.8452°$, $x_c = 0.8684m$ and $z_c = 0.2502m$ for circle with radius 10cm

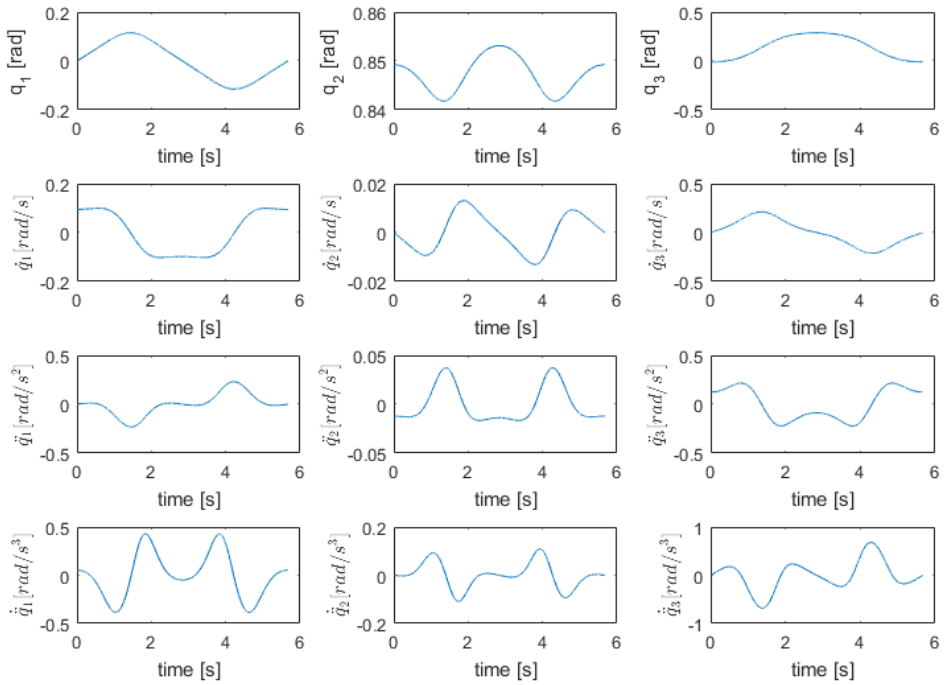

Figure B.8: Angular position, velocity, acceleration and jerk at optimal position for circle with radius 10cm

Optimal horizontal path

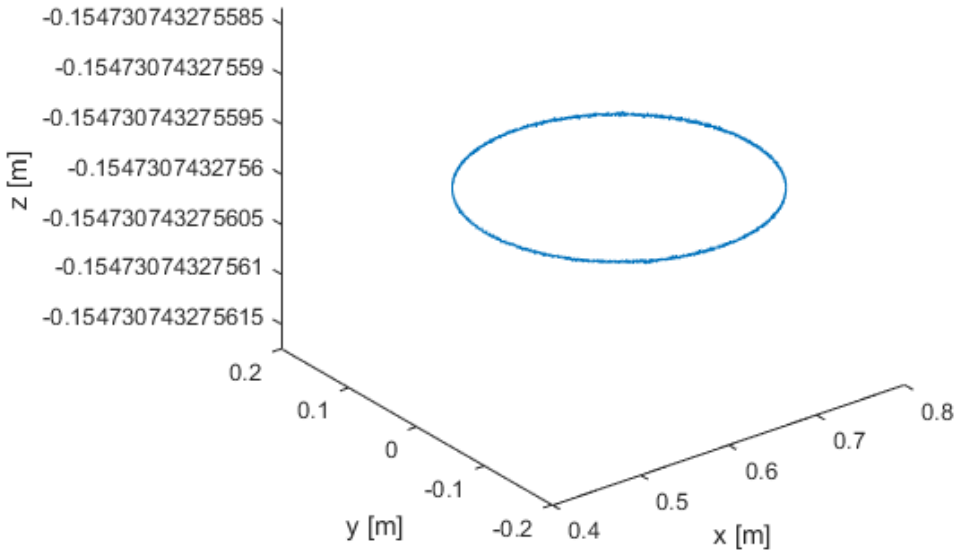

Figure B.9: The circle path at optimal position $\alpha = 0°$, $x_c = 0.68m$ and $z_c = -0.15m$ for circle with radius 15cm

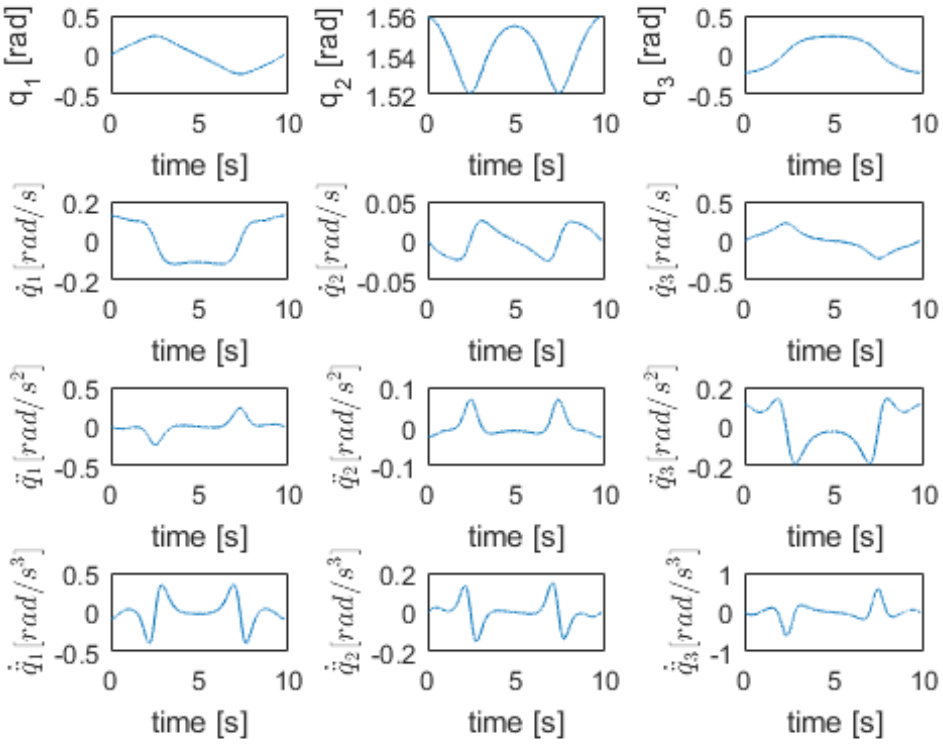

Figure B.10: Angular position, velocity, acceleration and jerk at optimal position for horizontal circle with radius 15cm

Appendix C

Files

startOptimization.m

This file sets up the optimization problem with an initial guess of the different optimization variables, and runs the fmincon optimization

returnEnergy.m

Function that returns the energy of a simulated model for a certain guess of optimization variables.

model.slx

Simulink model that simulates the circle motion and calculates the energy using the updated friction model. This simulink block diagram is quite similar to one made by [7], but code in the blocks is different with a new circle motion, additional optimization variables and a new velocity profile

model2.slx

Simulink model that simulates the circle motion and calculates the actual energy using the original friction model. Also used to make the vectors for θ, $\mathbf{q}$ and $\dot{\mathbf{q}}$ to be copied in to the external control simulink controller.

makeConstraints.m

A function that checks the nonlinear constraints for the circle motion, for the fmincon optimization. Similar to a function made by [7], but with the new and improved circle motion and additional constraints on angles, acceleration, position and orientation.

accelerationConstraints.m

Function that returns the acceleration constraints.

frictionModel.m

Function that returns the torque needed to overcome friction, obtained in [7] and [8].

forwardKinematics.m

Calculates the position using the current joint configuration

printPath.m

Plots the position, velocity, acceleration, jerk, torque and power for the optimized path.

plotExperiment.m

Plots the measured experimental values obtained using External Control and ABB Rapid programming.

matrixMultiplication.m

File computing the Forward Kinematics symbolically

trqTestPlot.m

Plots the torque experiment

InvDynControl_OrientedCircleLars.slx

Inverse Dynamics Controller used in the experiments with the optimal oriented circle. Similar to the one used in [7], but with small changes to use the new velocity profile and to handle oriented circles.

derivativesCircleMotion.mw

This file calculates the derivatives of $\phi_i(\theta)$ used for the calculation of the velocities, accelerations and jerks of the different joints, inspired by a function made by [8] and [7].

FindAccJerkConstraints.mw

Calculates the expressions for the acceleration and jerk constraints, built upon a function made by [8] and [7] which calculates the dynamics equations.

Log files

A folder including the logged files from all the experiments

Bibliography

[1] Rich Hooper. Industrial robots. URL http://www.learnaboutrobots.com/industrial.htm.

[2] Robots.com. Robot integration. URL https://www.robots.com/images/RobotIntegration.jpg.

[3] *IRB 1600 Data Sheet.* ABB robotics, 2014. URL https://library.e.abb.com/public/3b0491a94bd700a248257c71004ef393/PR10282EN_R8.pdf.

[4] Isolder Dressler. *Force control interface for ABB S4/IRC5*, 2009.

[5] Wikipedia. Denavit–hartenberg parameters, 2016. URL https://en.wikipedia.org/wiki/Denavit-Hartenberg_parameters.

[6] *Introduction to RobotStudios - Terms and concepts.* ABB Robotics, 2011. URL http://developercenter.cloudapp.net/BlobProxy/manuals/RobotStudioOpManual/doc11.html.

[7] Kristian Breistøl. Developing energy efficient scenarios of work for industrial robot manipulators. Master's thesis, Norwegian University of Science and Technology, 2015.

[8] Eirik Lie Strandbråten. Issues in trajectory planning and controlling an industrial robot's tool position for accurate passage through sharp corners of a nominal path. Master's thesis, Norwegian University of Science and Technology, 2015.

[9] *Execurive Summary World Robotics 2016 Industrial Robots*. IFR - International federation of robotics, 2016. URL http://www.ifr.org/fileadmin/user_upload/downloads/World_Robotics/2016/Executive_Summary_WR_Industrial_Robots_2016.pdf.

[10] *Key Electricity Trends*. IEA - International Energy Agency, 2015. URL http://www.iea.org/publications/freepublications/publication/Electricitytrends.pdf.

[11] Chalmers University. Smooth robot movements reduce energy consumption by up to 40 percent. URL https://www.chalmers.se/en/departments/s2/news/Pages/Smooth-robot-movements-reduce-energy.aspx.

[12] Sarah Knapton. Robots will take over most jobs within 30 years, experts warn. 16. February 2016. URL http://www.telegraph.co.uk/news/science/science-news/12155808/Robots-will-take-over-most-jobs-within-30-years-experts-warn.html.

[13] China.org.cn. Top 10 industrial robotic companies in the world. URL http://www.china.org.cn/top10/2015-09/17/content_36613597.htm.

[14] *Introduction to RAPID*. ABB robotics, 2007.

[15] Mark W. Spong, Seth Hutchinson, and M. Vidayasagar. *Robot Modelling and Control, First edition*. John Wiley Sons, Inc., 2006.

[16] *Product specification IRB 1600*. ABB robotics, 2010.

[17] Luigi Biagiotti and Claudio Melchiorri. *Trajectory Planning for Automatic Machines and Robots*. Springer, 2008.

[18] Stepan S. Pchelkin, Anton S. Shiriaev, Anders Robertsson, Leonid B. Freidovich, Sergey A. Kolyubin, Leonid V. Paramonov, and Sergey V. Gusev. On orbital stabilization for industrial manipulators: Case study in evaluating performances of modified pd+ and inverse dynamics controllers. 2016.

[19] Marius Nordheim Røv. Time optimal motion planning and motion control for industrial manipulators. Master's thesis, Norwegian University of Science and Technology, 2014.

[20] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization 2nd Edition*. Springer, 2006.

[21] MathWorks. fmincon, 2016. URL `https://se.mathworks.com/help/optim/ug/fmincon.html`.

[22] Cory Bryan, Mitch Grenwalt, and Adam Stienecker. Energy consumption reduction in industrial robots. *ASEE North Central Sectional Conference*, 2010.

[23] Statistisk Sentral Byrå. Elektrisitetspriser, 3. kvartal 2016, 2016. URL `https://www.ssb.no/energi-og-industri/statistikker/elkraftpris`.

[24] US Environmental Protection Agency. Greenhouse gas equivalencies calculator, 2016. URL `https://www.epa.gov/energy/greenhouse-gas-equivalencies-calculator`.

[25] Sergey Kolyubin, Anton Shiriaev, and Anthony Jubien. Consistent calibration of serial redundant manipulators: Case study in kuka lwr4+ sequential kinematics and dynamics identification. 2016.