NTNU – Trondheim
Norwegian University of
Science and Technology

Issues in Trajectory planning and controlling an industrial Robot's Tool position for accurate passage through sharp Corners of a nominal Path

Eirik Lie Strandbråten

Master of Science in Cybernetics and Robotics
Submission date: June 2015
Supervisor: Anton Shiriaev, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Abstract

Robot replaces manual labor all over the world every day. The demand for robots is extremely high in the ever growing automated society. The purpose of this master thesis is to improve the performance of an industrial robot manipulator, the ABB IRB 1600, at paths including sharp corners. In an industrial perspective there are several scenarios where precise robots at sharp corners are needed. An example is laser cutting of squares, where a deviation of millimeters gives poor result.

In this project the precision is far behind what the human eye is capable to see. Therefore, the Nikon K610 camera equipment is used to track the position of the robot. This is a high precision measurement device that could measure positions very accurate.

A dynamic model of the robot is developed using estimation tools and the data available from ABB. In addition to the dynamic model, a friction model is developed from experiments. This friction model gives the torque needed to overcome friction given the joint velocity.

Optimization is used to find a motion generator that, given different start perturbations, results in the smallest error along the path. In most robotics scenarios, classical asymptotic stabilization, known as reference tracking control, is used in order to follow a predefined motion. In this project orbital stabilization is used, which means convergence of solutions of a closed-loop system to an orbit of a nominal motion. This means that instead of using the classical tracking error in the controller, the error is redefined to be the error to the nominal orbit. This could probably improve the absolute path accuracy of the robot. To perform orbital stabilization, the Inverse Dynamics Controller is redesigned.

All the outcome of this report is tested at the NTNU Industrial Robotics Lab. The performance of the robot taking advantage of the developed friction model resulted in an improvement of 30 to 45 percent compared to a standard proportional-derivative controller. Compared to a good designed proportional-integral-derivative controller it is hard to see any improvement from the friction model.

The new controller together with the optimized motion generator is tested at the lab, performing a square. The path accuracy performance of the robot measured by the camera equipment shows an improvement of 36 percent compared with the state-of-the-art ABB controller. The maximum path error from the ABB controller is measured to be 2.2 millimeters, while the new controller and motion design gives an error of 1.4 millimeters.

Sammendrag (Norwegian)

Menneskelige arbeidsoppgaver blir til stadighet overtatt av roboter. Dagens store fokus på automatisering i industrien, gir en økt etterspørsel etter roboter. Denne oppgaven skal se nærmere på presisjonen til ABB roboten IRB 1600, langs baner som inkluderer skarpe hjørner. Det finnes mange industrielle problemstillinger hvor presisjon i skarpe hjørner er viktig. Et eksempel er laserkutting av kvadrater. En feil i størrelsesorden millimeter vil gi et veldig dårlig resultat.

Presisjonen det jobbes for å oppnå i denne oppgaven er mye høyere enn hva et øye kan se. Derfor brukes Nikon K610 kamerautstyr til å måle presisjonen langs banen. Dette er høypresisjons måleutstyr som gir en veldig nøyaktig måling.

Ved hjelp av estimasjonsverktøy og databladet til roboten, er en dynamisk modell av roboten utviklet. Eksperimenter har ført frem til en friksjonsmodell for roboten. Denne modellen gir ut kraften som trengs for å overkomme friksjonen gitt farten til den enkelte joint.

Optimalisering er brukt til å finne den bevegelsesgeneratoren som gir minst feil langs banen, gitt en startfeil. Innenfor kontroll av roboter er det vanligste å bruke klassisk asymptotisk stabillisering, også kalt referansefølging, for å følge en definert bevegelse. I denne oppgaven er orbital stabilisering brukt. Dette betyr konvergens av en løsning til et lukket sløyfe-system, til en orbit av den nominelle bevegelsen. Dette betyr at man i stedet for å definere feilen som avstanden til referansen, definerer feilen som avstanden til den nominelle banen. Det forventes at dette kan gi forbedring i presisjonen. Regulatoren som er tilpasset til dette er en Inverse Dynamic controller.

Alle steg gjort i denne rapporten er testet på NTNU sin industrielle robotikk lab. Ved å bruke den utviklede friksjonmodellen ser man en forbedring av presisjonen på 30 til 45 prosent sammenliknet med en standard proporsjonal derivat kontroller. Sammenliknet med en godt tilpasset proporsjonal integral derivat kontroller, er det vanskelig å se noen forbedring.

Den nye kontrolleren sammen med den optimaliserte bevegelsesgeneratoren ble satt til å kjøre i en firkant i rommet. En forbedring på 36 prosent ble oppnådd, sammenliknet med ABB sin egen kontroller. Maksimal banefeil for den nye kontrolleren ble målt til 1.4 millimeter mot ABB sine 2.2 millimeter.

Preface

This master thesis is a part of a masters degree in Engineering Cybernetics at Norwegian University of Science and Technology (NTNU). This thesis is a continuation of the project work from fall 2014.

It's been a pleasure to work in such a highly equipped laboratory. During the time at the lab I have not only learned a lot about the topics presented in this report, but also a lot about trouble shooting and always having a solution oriented mind.

First of all i want to thank my supervisor at the Department of Engineering Cybernetics, professor Anton Shiriaev, for his support and always open door during this spring semester. I also would like to thank Leonid Paramonov for valuable discussions regarding results. Next I would like to thank Sergey Kolyubin for support with the camera equipment at the lab. At last but not least i want to thank Stepan Pchelkin for his support with controller design and implementation issues, without his support this project would not be realizable.

A great thank to my colleague Kristian Breistøl for cooperation with the friction experiment, and for hours of experimental work at the lab. Also a great thank to Birgit Jegleim for proof reading and support this semester.

Trondheim, June 2015
Eirik Lie Strandbåten

Table of Contents

List of Tables

List of Figures

Abbreviations

atan2(y,x)	=	Definition used in writing
s_x	=	$sin(q_x)$
c_x	=	$cos(q_x)$
DOF	=	Degree Of Freedom
Task Space	=	Room defined by the coordinate system at the robot base
Joint Space	=	The space defined by each joint angle
PID	=	Proportional Integral Derivative
PD	=	Proportional Derivative
LED	=	Light Emitting Diode
ExtCtrl	=	External Control

Chapter 1

Introduction

1.1 Problem Description

Below is the problem description provided by Professor Anton Shiriaev:

”It is common to use VE envelops and a geometrical representation of a robot tool motion for developing a program performed later by the robot. Such software (e.g. RAPID) and human machine interface (e.g. Robot Studio) can be sufficient for realization of many robot operations in various scenarios. However, it has a number of deficiencies that become apparent when an absolute accuracy of the performed by robot motion is among the key specifications. The project is aimed at analysis and development of alternative motion planning algorithms for one of challenging motions in this category. Namely, it is focused on the way how a robots tool should pass through a corner, i.e. a point where the tool should stop and change a direction to go. The project assumes both theoretical, software developing and experimental parts to validate such alternative algorithms and implementations.”

1.2 Motivation

The demand for robot increases all over the world. To be able to compete in a global market the importance of new and more efficient technology increases. For high-cost countries like Norway adopting new technology is essential to keep the same level of prosperity. New technology does not only implies making new and fancy machines that could do a very specific task, it could also involve small improvements on already invented technologies such that they can be used in new settings and new environments.

An industrial robot is defined by ISO 8373 [1] as an automatically controlled, reprogrammable multipurpose manipulator with three or more axes. The first ABB robot, at that time ASEA, was delivered in 1974 [2]. This robot was the first microcomputer controlled electric industrial robot [2]. From this early beginning the ABB robots have entered a lot of new industries. Classical robotics tasks have been related to moving of components, simple machinery, car painting etc. Typical tasks that for humans is very tiring, and where robots is both faster and more precise. Robots are capable of doing a number of different tasks, only limited by the fantasy.

Although the performance of the robots are good, there are room for improvements. One of the greatest constraints for robots to enter new industries is the precision. It is well known that the absolute path accuracy for the best-on-the-market multifunctional robotic manipulators can hardily be reduced below 2-3 millimeters [9]. This excludes robot manipulators from a number of tasks like laser cutting, polishing, milling, cutting, welding etc. This means that specialized machines has to be made to automate tasks as mentioned. Compared to the already existing multipurpose robot technology, the specialized equipment is very expensive and the implementation requires a lot of work. This makes it motivating to improve the multipurpose robot such that it can perform these challenging tasks.

From earlier experiments in the project task [12], it is shown that the robot has a great deviation from the path at corners. For all the tasks mentioned above, a path through a sharp corner will often occur. A hypothetical situation is laser cutting of a square with sides of 10 centimeters. If the deviation from the desired path is in magnitude 2-3 millimeters the result is too bad for most applications. Therefore, improvement of the robots behavior at sharp corners is important for further introduction of robotics in such industries.

1.3 Objectives

The main contribution of this thesis is to find an optimal motion generator together with an advanced controller, in order to improve the precision of the robot at paths including sharp corners. The basic kinematics and a dynamical model of the robot is found from parameters available. In order to further improve the controller the force to overcome friction is experimentally found and implemented in the modified controller. This project uses a lot of advanced equipment, therefore a lot of time is used to trail and error.

From a easy-to-compute transverse coordinate the error along the path could be estimated given a small perturbation at the start of the of the motion. This perturbation representing the single point error of the robot. Then optimization is used to find the best possible motion generator, minimizing the error along the path. After finding the optimal motion generator for the given challenge in task space, the motion generator together with an advanced controller is implemented and tested. The performance of the new implementation is compared with the state-of-the-art commercial implementation.

1.4 Outline of Report

The report is laid out in the following way: Chapter 2 gives a description of the equipment used in this project. In chapter 3 the basic theory and concepts used in the rest of the report are described. Chapter 4 identifies a dynamic model of the robot from estimation tools and the data sheet, while experiments leads to a friction model for the robot. In chapter 5 a optimization problem is stated in order to optimize on the error along a path, given an initial error. In this chapter a controller is redesigned to achieve orbital stabilization, while in chapter 6 the controller is implemented. Chapter 7 gives the main results of the thesis, and chapter 8 gives some concluding remarks and some recommendations for further work.

Chapter 2

Equipment

The industrial robotics lab at NTNU is a well equipped lab. There are four different robots present at the lab. Three ABB robots, IRB 140-6/0.8, IRB 4600-60/2.05 and IRB 1600/6/1.2, all equipped with additional software that makes it possible to implement paths and control strategies in Simulink and MATLAB. IRB 4600 is also equipped with a moving belt conveyor BGL 23. The last robot is a KUKA Light Weight Robot with 7 DOF. The latest addition to the lab is a advanced camera measurement system from Nikon Metrology. For this project the IRB 1600 and the camera measurement equipment is used. Below is a description of each of the system.

In addition to the physical equipment at the lab different computer software is used. A briefly description of what is used, and for what purpose is described in this section.

2.1 Robot ABB IRB 1600 - 6/1.2

The robot IRB 1600 - 6/1.2 is used in this project, from now called IRB 1600. The robot is a 6 DOF industrial robot, that is equipped with a tracking tool. This tool is described in section 2.2. The 6 joints of the robot is powered by AC motors. This robot handles a payload of 6 kg, and reaches up to 1.2 m.

The robot is controlled by a IRC5 controller cabinet, in which a complete model of the robot dynamics exists. This is together with the path planning strategies the state-of-the-art of ABB robots, therefore ABB does not publish this information. Neither they makes it possible to modify it. To make a model of the robot and include our own path planning strategies the dynamics of the robot has to be found through experiments. The IRC5 controller could mainly be divided into to parts, i.e. the main computer and the axis computer. The main computer is the part of the controller that plans paths and all the signals sent to the axis computer. The main computer calculates the position and velocity,

Gain	1. Joint	2. Joint	3. Joint	4. Joint	5. Joint	6. Joint
Kp	20	20	20	20	20	20
Kd	1.8	1.4	0,3	0.1	0.14	0.1
Ki	18	14	5	2	2.8	2.2

Table 2.1: Controller gains of IRB1600 Axis Computer

Link i	$q_{i,min}$	$q_{i,max}$	$\dot{q}_{i,min}$	$\dot{q}_{i,max}$
1	-1.9199	1.9199	-2.6180	2.6180
2	-1.0996	2.3736	-2.7925	2.7925
3	-4.1015	0.9599	-2.9671	2.9671
4	-3.4907	3.4907	-5.5851	5.5851
5	-2.0071	2.0071	-6.9813	6.9813
6	-6.9813	6.9813	-8.0285	8.0285

Table 2.2: Joint position and velocity limits

but also torque needed to overcome friction, gravity etc.The main computer is not possible to look up, neither to modify. The axis computer is a PID controller as controlling each joint individually. This controller also has a feed forward term that makes it possible to compensate for known disturbances. This controller is shown i figure 2.2. A PID controller is a proportional-integral-derivative controller. This is a widely used controller that is easy to use, and is able to handle challenging task, but it has some disadvantages. This controller is based on time planned paths, in the scope of precision, time is not of interests. The standard settings of this controller is shown in Table 2.1. A feed forward term is a connection to the controller that sends a additional control signal to the controller. This signal is shown i figure 2.2, labeled ”trgFfq”.

Each joint of the robot has limited reach and velocity. This data is presented in Table 2.2. In all experiments this table would be used to avoid joint out of reach. The first joint has from the data sheet a reach of $\pm\pi$ radians. Since there is placed a computer behind the robot physical limits of ± 1.9199 radians is set. These limits are pins placed at the sides of the robot, that physically prevents the robot to pass. When the robot reach these pins, safety functions stops the robot. The scope of this project is accuracy, therefore this project will never illuminate the highest velocities for each joint.

2.1.1 RobotStudio

RobotStudio is ABB PC software for modeling, offline programming and simulation of robots. For this project RobotStudio is used for implementation of paths used for calibration. This software enable the user to model a robot system as it is in the lab. This enables the user to make paths and simulate them without spending time with the real robots. This is less time consuming and the risk of damaging the equipment is eliminated. The software is based on drag and drop in a graphical user interface. From this graphical interface RAPID code is generated, which is the control language of the robot. This software has

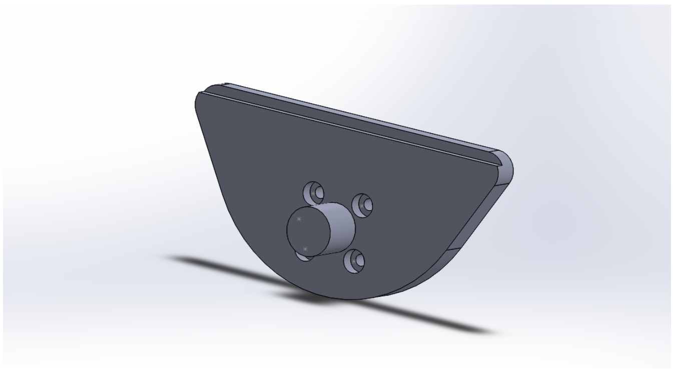


Figure 2.1: The tracking tool designed in SolidWorks

a lot of robot systems and equipment in its library, in addition it have the possibility to implement customized objects.

RAPID is the programming language of ABB robots. This language describes each target, and the path between targets. Different settings could be set to get the desired behavior. In this project precision is very important so the setting ”fine” is used. This means that the robot goes exactly to the desired target. Other settings could be used, for instance ”‘z = 5”’ then the robot is closer than 5 mm to the target. Both velocity and the motion it selves could be described in instructions in RAPID.

2.2 Tracking Tool

This project uses an external camera system to track the end effector relative to the base of the robot. Since this system is tracking LED’s a tool to mount on the robot’s end effector is designed. The following criteria is set for the tool:

- Light weight to save 3D printing costs
- Possible to place 3 LEDs
- Large spread between LEDs
- At least one LED at a different level
- Need a cylinder to define frame
- Need a line to define frame
- A rigid design

The tool is designed in SolidWorks, and printed on a 3D printer. The tool is shown in figure 2.1. All the elements from the list above is included in the design. The printed tools is rigid and easy to place the LEDs on.

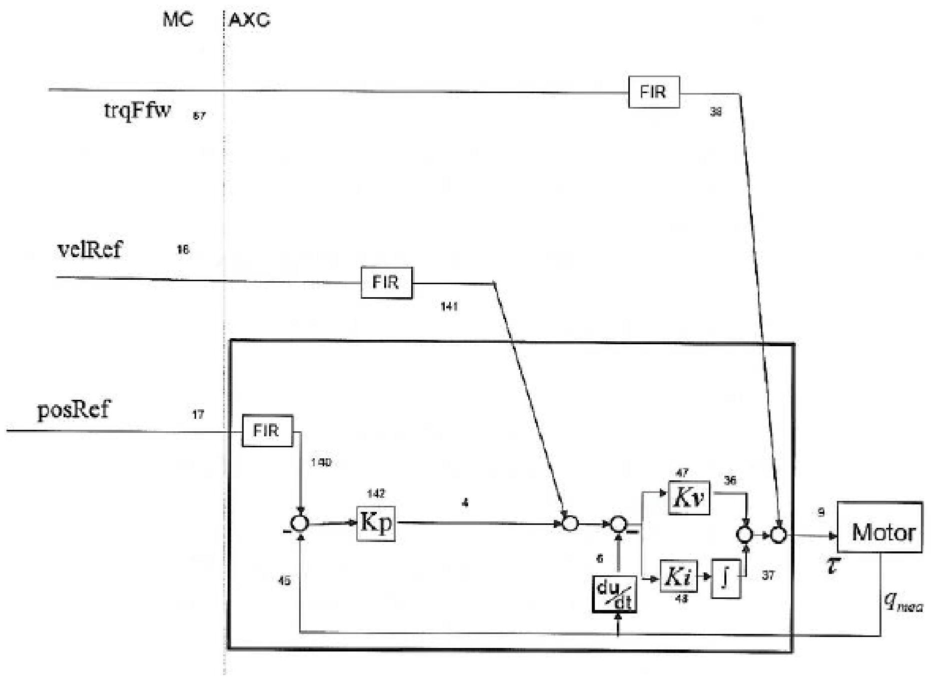


Figure 2.2: PID controller used by ExtCtrl

2.3 External Control

The principle of external control is to go around the standard ABB path planning system. The external control makes it possible to read and change positions, velocities and torques to each of the joint, sent from the main computer to the axis computer. The main computer is the ABB controller, that performs the path planning and sends velocity and position references to the axis computer. The path planning is the secret of ABB robots, and is not modifiable. Therefore, new path planning algorithms have to be developed to make an improved controller. The axis computer is a standard PID controller, that controls each joint individually. The gains of the PID controller is adjustable, and it is possible to include torque feed forward compensation. For example the feed forward makes it possible to compensate for gravity and friction forces. The controller is shown in figure 2.2. This tool is made in the scope of easy testing of new path planning strategies, and is therefore a tool made for an experimental environment. There is only three places in the world where External Control is used to control ABB robots.

The external control software, ExtCtrl, runs on a Linux Fedora computer. For communication with the IRC5 robot controller a serial connection and a ethernet cable is used. This computer is equipped with MATLAB and Simulink software with the Real Time Workshop installed. Then paths could be planned in MATLAB and Simulink and sent to the robot threw ExtCtrl.

When using external control most of the safety from ABB is turned off. What's left is a safety limit for too large steps in the position. This means that every planned path has to be tested carefully in simulations, and all limits on joint angles and velocities has to be checked toward the limits of the actual robot. It is also important to use low speed during testing, and not work on the limits of what is possible.

External Control in practice

Since this software removes many safety features of ABB, the use of the software requires attention from the user. The four steps of the program is described below:

Load, loading a chosen program into ExtCtrl. This program is made in MATLAB and Simulink, and build with the Real Time Workshop extension.

Unload, unload the current program from ExtCtrl.

Switching to "submit" means that all signals from the robot's main controller could be read. These signals are measurements of joint angle and velocities, regulator gains, torques and state of the robot. These signals are provided both filtered and unfiltered. Switching to "submit" is described in figure 2.3, where the signals from the main computer is sent to the extctrl. This picture is taken from reference [5], where extended information about the setup can be read. When the external control is in submit, the joint position and velocity signals can be tested, using the current position of the robot as initial position, without doing any movements on the robot. The main computer is still controlling the robot, while the Simulink model is not connected.

Switching to "obtain" means that the main computer is completely turned off, while the external control is activated. Figure 2.3 shows how the signal from the main computer is switch off, and the signal from the external controller is switched on. Now the paths made in Simulink is sent to the axis computer and the robot will perform the desired path.

The extctrl has an option to set parameters that is defined in the Simulink model. Then experiments with different velocities could be done without compiling the model again. A parameter to start the clock, and then start the path planning is also used.

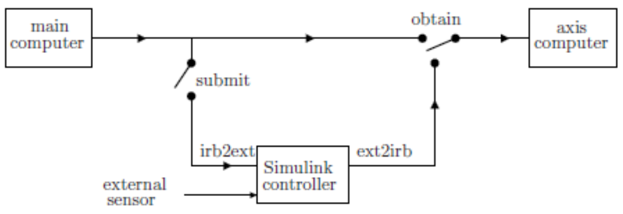


Figure 2.3: Schematic of reference value path from main to axis computer

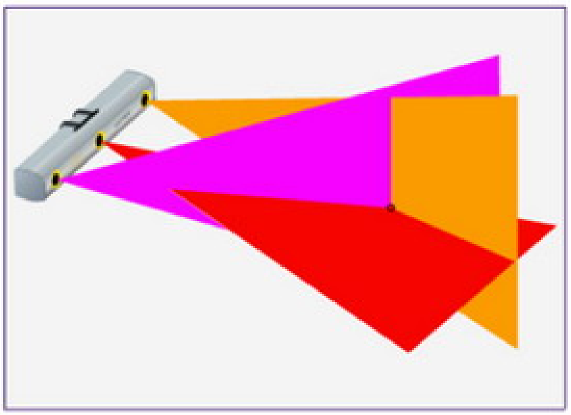


Figure 2.4: Triangulation detecting the position of a LED, from [6]

2.4 Nikon Metrology K610

The robotics lab at NTNU is equipped with a Nikon Metrology K-Series Optical CMM camera. This is a system that uses three linear CCD cameras to measure the 3D position of a LED. This is calculated through the principles of triangulation. Figure 2.4, shows how a LED is detected by the three cameras. When a LED is detected the position is calculate from the angle between the point and each of the three cameras.This means that the relative position between each camera has to be very accurate. To handle deviation in the construction regarding changes in temperature etc., the camera system has a calibration routine that adjusts for external influences. A very accurate bar with LEDs is measured in 24 different positions to calibrate the camera. This is a routine that is done weekly in testing periods, to ensure good enough accuracy. This camera system enable precision far beyond both the human eyes and what the robot is able to perform. The camera operates with single point accuracy up to $37\mu m$. For a measurement of a volumetric element the accuracy is $60\mu m$. Specific data on the camera equipment is taken from the reference [6], where also more information is available.

The camera is able to find positions of LEDs, but also to track moving LEDs. This makes it possible to use the camera both for measuring positions and elements in 3D, but also tracking of moving elements. For this report the tracking of moving elements is the main purpose, therefore most of the descriptions is focused on this purpose. To be able to track a moving object, frames have to be defined. The camera is able to find the position of a frame relative to different objects. This could either be other dynamic frames, a static frame measured in the room or the camera itself. The camera can also measure positions of points in the room with a Space Probe. From points frames is defined, or other geometric elements.

Space Probe

The Space Probe is a hand held device with nine LEDs attached to it. With such a high number of LEDs, and the rigid body, the Space Probe ensures high precision measurements. The Space Probe have tips of different size and length, such that the user can measure positions with different shape. When changing a tip, the new tip has to be calibrated. This is done by measuring the same position with different angles of the Space Probe relative to the camera position. Different types of tips exist, but for this project the 5 and 10 millimeter balls are used.

Geoloc and defining frames

The robot's end effector should be measured in coordinates of the base frame. In order to do that the camera system requires frames to be defined both at the end effector and at the base of the robot. From those two frames, the relative position between them can be measured. To do this a program called Geoloc is used. This makes it possible to create coordinate frames from different features in the room. There are two main methods to define frames. Either as a direct way, i.e. defining real elements of the coordinate system, or in an indirect way by measuring a cloud of points in the room.

Direct definition of frames

First the direct way will be described and later on the indirect way. To be able to directly make a frame, the following has to be defined:

- Direction of primary axis of the frame
- Direction of secondary axis of the frame
- Known position in the frame

In Geoloc different features can give us this. To make a feature in Geoloc the Space Probe is used to measure points. For example if a line should be made, points along the line is measured with the Space Probe. Different Geoloc features are shown in figure 2.5, with positive orientation shown.

- Point
- Line
- Plane
- Circle
- Cylinder

A easy definition of a frame is shown in figure 2.6, where a line defines positive x-direction. A plane normal defines the positive y-axis. And a known point at $z = 500$ defines the origin of the frame.

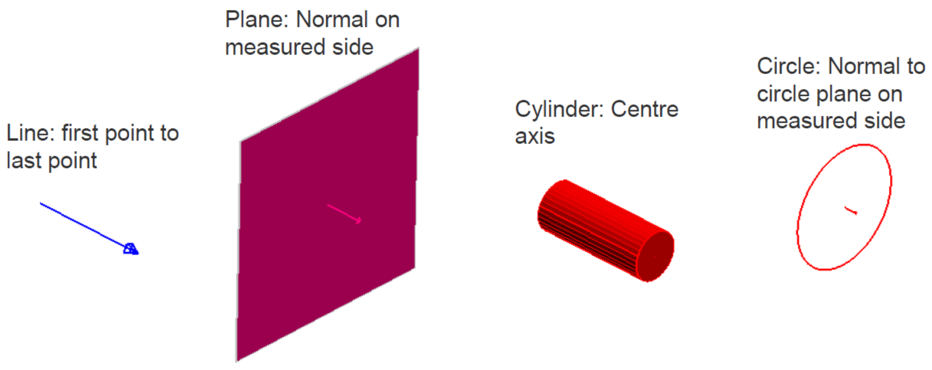

Plane: Normal on measured side
Line: first point to last point
Cylinder: Centre axis
Circle: Normal to circle plane on measured side

Figure 2.5: Geoloc features, from [7]

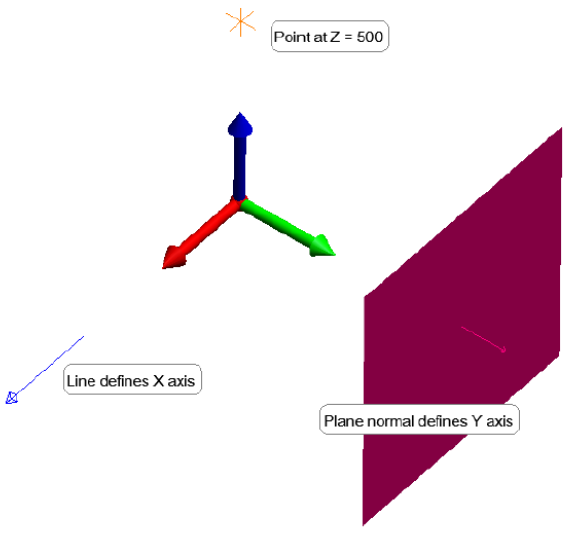


Figure 2.6: Direct definition of frames, from [7]

To define the robot's base frame different approaches are tested out, but the best results is achieved with the following solution. The positive z-direction is defined with a cylinder. A cylinder exists of two different elements, a plane and a circle. The cylinder is made by running the robot in a circle movement. A program for defining frames is made in Robot Studio with "fine" precision settings at each point. To define the plane different solutions are tested. Since the precision of this work is very important, small deviations of the plane will result in a large error of the robot's behavior measured by the camera. Therefore it is very important to get all measurements with the Space Probe correct. A lot of testing is done, how to measure this plane. Both strategies with measuring a lot of points on the base of the robot, and more selective strategies with specific point measurements are done. Since the base of the robot is not a perfect plane many point measuring strategies gives some error. The best result is achieved by measuring 3 points around each of the three feet of the robot. Since this is the contact points between the robot and the base this gives the most correct plane.

This cylinder both gives the positive direction of the z-axis and the origin of the base frame, so the next thing needed is either the positive direction of the y or x-axis. Two different strategies to define positive x-direction is tested out. First three screws on the side of the robot is used to define the positive x-direction. Second the positive x-direction is found by driving the robot in a line movement in positive x-direction. For this a Robot Studio program with "fine" precision settings is used. The results are similar, so most used is the screws.

To be able to track the end effector, a frame at the end also has to be defined. The tool in figure 2.1 is designed to make the frame definition easy. Therefore it has a cylinder in positive z-direction, to define the z-axis and the origin of the frame. A horizontal trace at the top of the tool defines the positive y-axis. The orientation of the tool only counts when the orientation of the end effector is essential in the path planning. For this project only the position of the end effector is of interest.

Indirect definition of frames

A frame could also be defined indirectly. This means that a cloud of known points relative to the frame, that should be defined, is measured in the room. from this the frame is calculated. For defining the robot's base frame this is a very effective tool. To use this tool a number of positions are made in Robot Studio. All positions have the precision settings "fine", such that the robot will relay only on the precision ABB provides for a single point. The program generated in Robot Studio runs, and all the positions are measured with the Space Probe. The Robot Studio file is attached, see appendix A.3. The path of the robot is shown in figure 2.7. All the small coordinate systems in the figure represent a measured point. All the position data from Robot Studio is punched into Geoloc, and a mapping tool generates the frame based on this data. The disadvantage of this tool is that it relies on the DH parameters of the robot being correct. This is not tested in this project, and to improve performance further the DH parameters could be verified with an external measurement device.

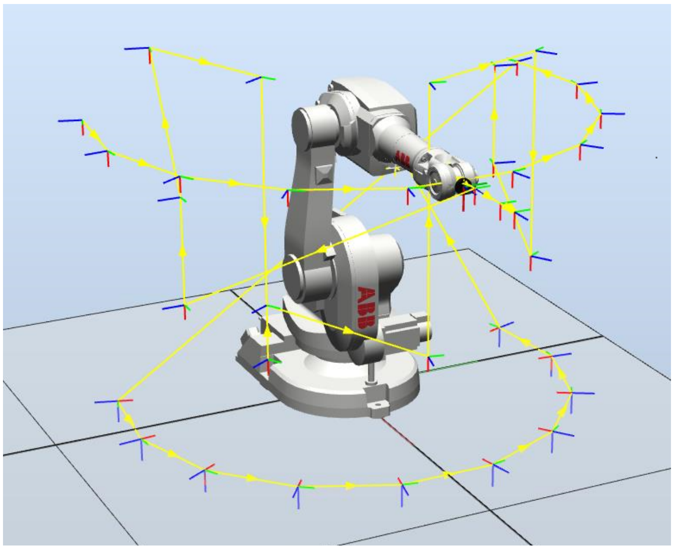


Figure 2.7: Path for indirect definition of base frame

2.4.1 Placement of LEDs

After defining frames, the camera knows the position of the frame in the coordinate system of the camera. Since the camera is moveable, it is important to make the frames dynamic before moving anything at all. Making a frame dynamic means that LEDs is attached to the object that is measured with the Space Probe. For example is LEDs attached to the end effector tool, after measuring the elements described above. Both the base frame, and the end effector is made dynamic. The end effector is dynamic because this is the moving element that should be tracked. The base is made dynamic to make it possible to do changes of the camera position, without destroy the frame definitions. Tracking the robot's end effector the position of interest is not the position in the cameras coordinate system, rather the position of the end effector in the base coordinate system. Since the base of the robot never moves, and the interest of the end effector only is relative to the base frame, the camera can move without loosing any information.

To attach LEDs a glue gun is used. To get best possible visibility of the LEDs they are pointed towards the camera. The angle of the them towards the camera has to be less than 30 degrees. The LEDs are placed with sufficient distance between them, and at different distances toward the camera. In figure 2.8 the placement of the LEDs on the base is shown. For the end effector the LED placement is shown in figure 2.9

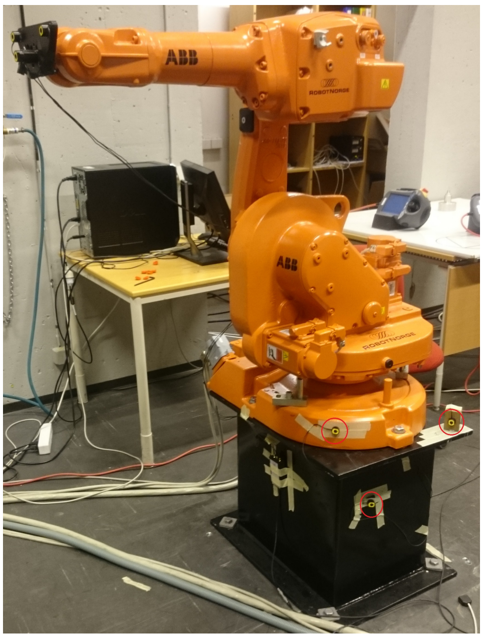


Figure 2.8: Placement of LEDs on the base. LEDs inside red circle

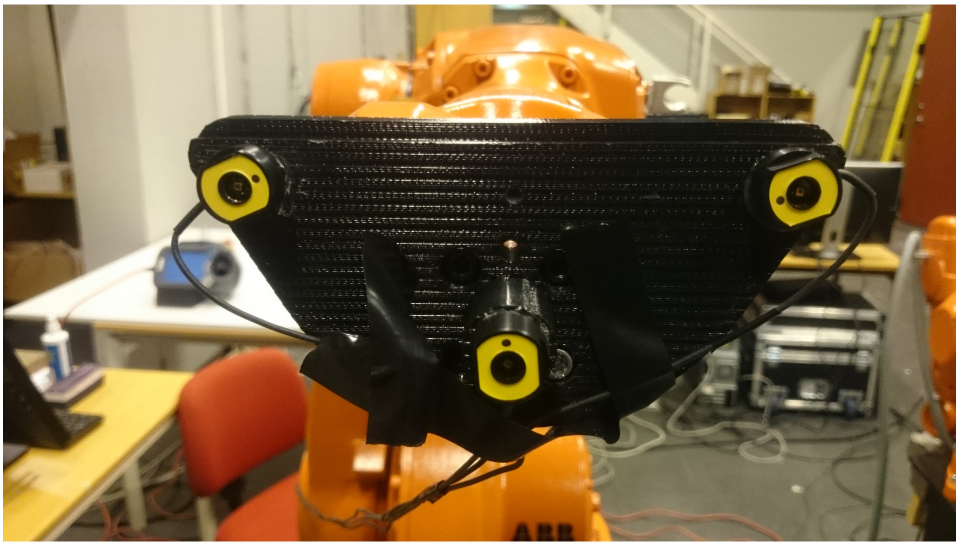


Figure 2.9: Placement of LEDs on the end effector

2.5 Computer Software

MATLAB R2014a

For calculations, simulations and making plots MATLAB is used. All the controllers for external control are also designed and implement in MATLAB with the Real Time Workshop. For controlling of the robot an older version of MATLAB is used, i.e. R2011a.

Maple 2015

For symbolic calculations Maple is used.

SolidWorks 2014 Dassault Systems

For estimation of different robot parameters, SolidWorks and the CAD drawings of the robot is used. SolidWorks is also used for designing of the 3D printed tool.

Chapter 3

Concepts and Theory

This chapter describes the basic theory and consepts used in this project. The background theory and the matemathical scripts from this chapter is used in advance in the further work with the project.

3.1 Kinematics

To be able to plan paths for the robot it was necessary to develop tools to convert between the joint frame and the base frame. Joint frame means the angle of each joint on the robot, while base frame is the position of the end effector in coordinates relative to the base of the robot. To control the robot we need to give it positions and velocities in the joint frame, but the motion that the robot should perform is often described in the base frame.

To develop this the DH (Denavit-Hatenberg) convention is used. It is possible to solve this without this convention, but for a system with complexity like this, it is much easier to use the DH convention. This convention is based on four basic transformations:

$$A_i = Rot_{z,\theta_i} Trans_{z,d_i} Trans_{x,a_i} Rot_{x,\alpha_i} \tag{3.1}$$

$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} c_{\alpha_i} & s_{\theta_i} s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i} c_{\alpha_i} & -c_{\theta_i} s_{\alpha_i} & a_i s_{\alpha_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

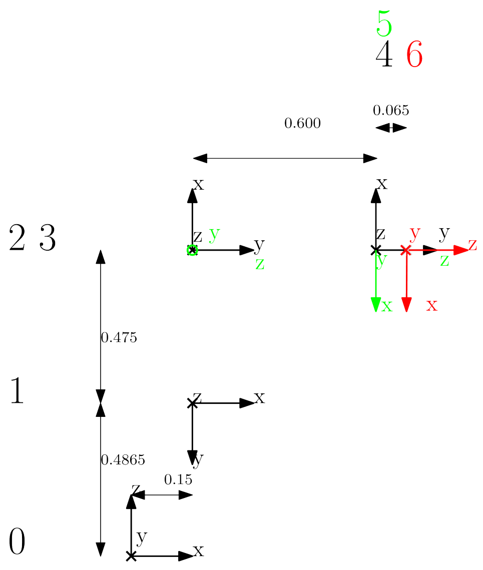


Figure 3.1: DH coordinate frame assignment for IRB1600

Where $\theta_i, d_i, a_i, \alpha_i$ is the parameter associated with link i and joint i. θ_i is the joint angle, d_i is the link offset, a_i is link length and α_i is the link twist. The translations of each joint is shown in figure 3.1, and the parameters for each joint i is written in table 3.1. Combining these parameters with the A_i matrix shown above, results in a translation matrix for each joint of the robot.

i	d_i	θ_i	a_i	α_i
1	$d_1 = 0.4865$	q_1	$a_1 = 0.15$	$-\pi/2$
2	0	$-\pi/2 + q_2$	$a_2 = 0.475$	0
3	0	q_3	0	$-\pi/2$
4	$d_4 = 0.600$	q_4	0	$\pi/2$
5	0	$\pi + q_5$	0	$\pi/2$
6	$d_6 = 0.065$	q_6	0	0

Table 3.1: DH parameters for IRB1600

Using convention:

$$c_i = \cos(q_i)$$

$$s_i = \sin(q_i)$$

Using that:

$$\sin(-\pi/2) = -1 \tag{3.4}$$

$$\cos(-\pi/2) = 0 \tag{3.5}$$

$$A_1 = \begin{bmatrix} c_1 & 0 & -s_1 & a_1c_1 \\ s_1 & 0 & c_1 & a_1s_1 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

Using that:

$$\cos(q_i - \pi/2) = \sin(q_i) \tag{3.7}$$

$$\sin(q_i - \pi/2) = -\cos(q_i) \tag{3.8}$$

$$A_2 = \begin{bmatrix} s_2 & c_2 & 0 & a_2s_2 \\ -c_2 & s_2 & 0 & -a_2c_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.9}$$

Using equation 3.4 and 3.5:

$$A_3 = \begin{bmatrix} c_3 & 0 & -s_3 & 0 \\ s_3 & 0 & c_3 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.10}$$

Using that:

$$\sin(\pi/2) = 1 \tag{3.11}$$

$$\cos(\pi/2) = 0 \tag{3.12}$$

$$A_4 = \begin{bmatrix} c_4 & 0 & s_4 & 0 \\ s_4 & 0 & -c_4 & 0 \\ 0 & 1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.13}$$

Using equation 3.11, 3.12 and the following:

$$\cos(q_i + \pi) = -\cos(q_i) \tag{3.14}$$

$$\sin(q_i + \pi) = -\sin(q_i) \tag{3.15}$$

$$A_5 \begin{bmatrix} -c_5 & 0 & -s_5 & 0 \\ -s_5 & 0 & c_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.16}$$

$$A_6 \begin{bmatrix} c_6 & -s_6 & 0 & 0 \\ s_6 & c_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.17}$$

3.1.1 Forward Kinematics

To be able to get the position of the end effector, given the joint angles, the forward kinematics is necessary. The transformation matrix from base frame to end effector has the form of 3.18 with $i = 6$. Here o_6 represents the position of the end effector in coordinates of the base frame. Therefore o_6 is used to calculate the position of the end effector from the joint angles. The calculation of o_6 is shown in attached files see appendix A.1.

$$T_0^i = \begin{bmatrix} R_0^i & o_i \\ 0 & 1 \end{bmatrix} \tag{3.18}$$

$$o_6 = \left[\begin{matrix} a_1c_1 + d_6(c_5(c_1c_2c_3 - c_1s_2s_3) - s_5(s_1s_4 + c_4(c_1c_2s_3 + c_1c_3s_2))) + \dots \\ a_1s_1 + d_6(c_5(c_2c_3s_1 - s_1s_2s_3) + s_5(c_1s_4 - c_4(c_2s_1s_3 + c_3s_1s_2))) + \dots \\ d1 + a_2c_2 - d_4(c_2s_3 + c_3s_2) - \dots \end{matrix}\right.$$

$$\left.\begin{matrix} \dots d_4(c_1c_2c_3 - c_1s_2s_3) + a_2c_1s_2 \\ \dots d_4(c_2c_3s_1 - s_1s_2s_3) + a_2s_1s_2 \\ \dots d_6(c_5(c_2s_3 + c_3s_2) + c_4s_5(c_2c_3 - s_2s_3)) \end{matrix}\right] \tag{3.19}$$

Inserting the joint angles $q_i i \in [1, 6]$ in equation 3.19 returns the $[x, y, z]^T$ position of the end effector.

3.1.2 Jacobian

To transform from velocities in the base frame to velocities for each joint, the Jacobian matrix has to be calculated for both the linear and the angular velocity. From reference [11] chapter 4.6 more information about derivation of the Jacobian could be found. The Jacobian is divided into a upper and a lower part, where the upper part (J_v) represents the linear velocity and the lower half (J_ω) represents the angular velocity.

Linear velocity

$$J_v = [J_{v_1} \dots J_{v_n}] \tag{3.20}$$

Where

$$J_v = \begin{cases} z_{i-1} \times (o_n - o_{i-1}) & \text{for revolute joint i} \\ z_{i-1} & \text{for prismatic joint i} \end{cases} \tag{3.21}$$

Angular velocity

$$J_\omega = [J_{\omega_1} \dots J_{\omega_n}] \tag{3.22}$$

Where

$$J_\omega = \begin{cases} z_{i-1} & \text{for revolute joint i} \\ 0 & \text{for prismatic joint i} \end{cases} \tag{3.23}$$

Here o_i $i \in [1, 6]$ is the translation from joint angles to position of joint i in base frame. z is the z part of the rotation matrix from the transformation matrix 3.18

$$R = [x, y, z] \tag{3.24}$$

For the IRB1600 there are only revolute joints. The calculation of the matrix is done symbolically using MATLAB. The file is attached, see appendix A.1.

3.1.3 Inverse Kinematics

The inverse kinematics is used to calculate the joint positions, given the manipulator position in the base frame. To make paths it is necessary with a tool that enable us to both convert from joint frame to base frame, and from base frame to joint frame. In the earlier section it is shown how to compute the manipulators positions from the joint angles. This turned out to be easy calculations that gives one particular solution. This section will show the complexity going from the base frame to the joint frame. The complexity appears because multiple solutions often will occur, and the fact that a solution not necessary is feasible.

Kinematics decoupling

The inverse problem is difficult to solve in general, but for the 6 DOFs manipulator IRB 1600 it could be simplified. This is because the last three joints intersect at one point. This makes it possible to decouple the problem into two simpler problems, the inverse position problem and the inverse orientation problem.

The first problem is to find the position of the wrist center. This means the intersection point of the wrist axes. After finding the position of this point it is possible to calculate the angle of the first three joints. After that the the orientation of the wrist is calculated.

For a given rotation and orientation:

$$R_6^0(q_1, \dots, q_6) = R \tag{3.25}$$

$$o_6^0(q_1, \dots, q_6) = o \tag{3.26}$$

Defining the position of the wrist center as o_c. Since the first three joints give the position of the wrist, the position of the end effector is only a translation of distance d_6 along z_5:

$$o = o_c^0 + d_6 R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.27}$$

$$o_c^6 = o - d_6 R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.28}$$

Then we have the position of wrist center $o_c = (x_c, y_c, z_c)$.

In the same manner the rotation could be decoupled. Since we can find the position of the first three joints it is also possible to find the rotation (R_3^0).

$$R = R_3^0 R_6^3 \tag{3.29}$$

$$R_6^3 = (R_3^0)^{-1} R = (R_3^0)^T R \tag{3.30}$$

Finding q_1

To find the position of the first joint, the (x,y) position of the wrist center we found above is used. From figure 3.2 it is clear that the angle has to be as follows:

$$q_1 = atan2(y_c, x_c) \tag{3.31}$$

Since the robot could rotate ± 180 degrees, there exist two solutions:

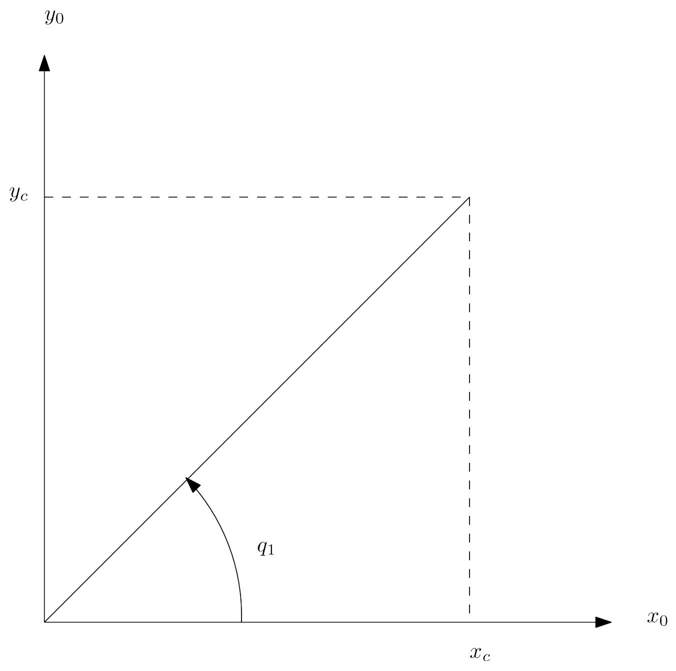


Figure 3.2: Wrist center in x_0 - y_0 plane

$$q_1 = \pi + atan2(y_c, x_c) \tag{3.32}$$

Finding q_2 and q_3

The next step is to find the third joint angle, because this angle decides whether the solution is an elbow up or elbow down solution. Figure 3.3 shows that the positive direction of the robot is opposite for what is naturally, therefore the negative sign appears. This figure does not show the rotation of the axis, this will be handled below.

The first step is to find the distances s and r from figure 3.3.

$$r = \sqrt{(x_c - a_1\cos(q_1))^2 + (y_c - a_1\sin(q_1))^2} \tag{3.33}$$

$$s = z_c - d_1 \tag{3.34}$$

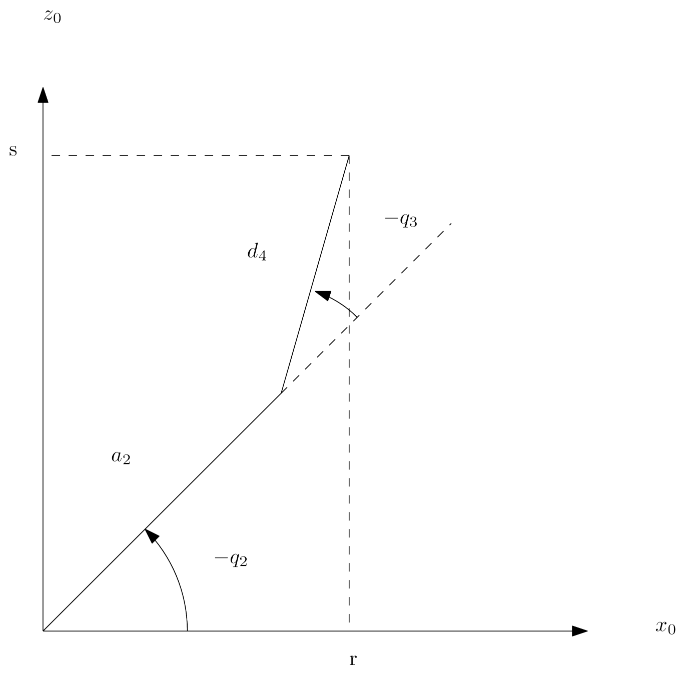


Figure 3.3: Plane formed by q_1 and q_2

Joint q_3

this gives us the x component:

$$\cos(q_3) = \frac{r^2 + s^2 - a_2^2 - d_4^2}{2a_2 d_4} = D \tag{3.35}$$

the corresponding y component is:

$$\sin(q_3) = \pm\sqrt{1 - D^2} \tag{3.36}$$

Compensating for the $90\,\deg$ deviation from zero position showed in figure 3.3 gives:

$$-q_3 = atan2(\pm\sqrt{1 - D^2}, D) + \frac{\pi}{2} \tag{3.37}$$

$$q_3 = -atan2(\pm\sqrt{1 - D^2}, D) - \frac{\pi}{2} \tag{3.38}$$

The two different solutions correspond to an elbow up and an elbow down solution. The negative sign gives elbow up, while the positive sign gives the elbow down solution.

Joint q_2

$$-q_2 = atan2(s,r) - atan2\left(d_4 \sin\left(-q_3 - \frac{\pi}{2}\right), a2 + d_4 \cos\left(-q_3 - \frac{\pi}{2}\right)\right) - \frac{\pi}{2} \quad (3.39)$$

$$q_2 = -atan2(s,r) + atan2\left(d_4 \sin\left(-q_3 - \frac{\pi}{2}\right), a2 + d_4 \cos\left(-q_3 - \frac{\pi}{2}\right)\right) + \frac{\pi}{2} \quad (3.40)$$

Finding q_4, q_5 and q_6

For now, the position of the wrist center is determined. The next step is to find the orientation of the end effector. As mentioned earlier the three last joints determines this orientation. Therefore the orientation of the end effector could be found solving the inverse of the rotation matrix from link three to six:

$$\begin{aligned}
T_6^3 &= A_4 A_5 A_6 && (3.41)\\
&= \begin{bmatrix} R_6^3 & o_6^3 \\ 0 & 1 \end{bmatrix} && (3.42)\\
&= \begin{bmatrix} s4s6 - c4c5s6 & c6s4 + c4c5s6 & -c4s5 & -c4d6s5 \\ -c4s6 - c5s4s6 & c5s4s6 - c4c6 & -s4s5 & -d6s4s5 \\ -s5s6 & s5s6 & c5 & d4 + c5d6 \\ 0 & 0 & 0 & 1 \end{bmatrix} && (3.43)
\end{aligned}$$

From R_6^3 we see that q_5s x component could be found from the element r_{33} i.e:

$$\cos(q_5) = r33 \quad (3.44)$$

From R_6^3 we see that q_5s y component could be found from a combination of r_{13} and r_{23}, i.e.:

$$\begin{aligned}
\sqrt{r_{13}^2 + r_{23}^2} &= \sqrt{(\cos(q_4)\sin(q_5))^2 + (\sin(q_4)\sin(q_5))^2} && (3.45)\\
&= \sqrt{\sin(q_5)^2(\cos(q_4)^2 + \sin(q_4)^2)} && (3.46)\\
&= \sqrt{\sin(q_5)^2} && (3.47)\\
&= \pm\sin(q_5) && (3.48)
\end{aligned}$$

Then we have that:

$$q_5 = -atan2\left(\pm\sqrt{r_{13}^2 + r_{23}^2}, r_33\right) \quad (3.49)$$

If $\sin(q_5) > 0$

$$q_4 = atan2(r_{23}, r_{13}) \tag{3.50}$$
$$q_6 = atan2(r_{32}, -r_{31}) - \pi \tag{3.51}$$

If $\sin(q_5) < 0$

$$q_4 = atan2(-r_{23}, -r_{13}) \tag{3.52}$$
$$q_6 = atan2(-r_{32}, r_{31}) - \pi \tag{3.53}$$

3.2 Robot Dynamics

A dynamical model of the robot is required when an optimal trajectory should be found. The dynamical model of the robot has the form:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) \tag{3.54}$$

This equation describes the dynamics of the system. Where τ is control torque applied at the joints, M is the inertia matrix, the vector $C(q, \dot{q})$ represents Coriolis and centrifugal generalized forces, and G is due to gravity.

There are mainly two methods to determine the dynamic equation of the robot, namely the Lagrangian formulation and the Newton-Euler formulation. The methods are equivalent in almost all aspects, although the methods are build upon two different philosophies. In practice the Lagrangian formulation treats the the manipulator as a whole and using the Lagrangian function. The Newton-Euler formulation looking at each joint by themselves and writing down the equations for linear and angular motion. For this case the Newton-Euler formulation will be used. The steps is detailed described in [11], only the most important parts will be illuminated in this report:

The set of parameters and vectors below is used in the Newton-Euler method:

Newtons second law gives the force balance for a single joint i:

$$f_i - R^i_{i+1} f_{i+1} + m_i g_i = m_i a_{c,i} \tag{3.55}$$

The moment balance for link i needs to be found. The moment exerted by a force around a point is given by $f \times r$ where r is the vector from the point where the force is applied to the point where the moment is computed. This gives:

$$\tau_i - R^i_{i+1}\tau_{i+1} + f_i \times r_{i-1,ci} - (R^i_{i+1} f_{i+1}) = \omega_i \times (I\omega_i) + I_i \alpha_i \tag{3.56}$$

The challenge of the Newton-Euler formulation are to find the vectors f_i and τ_i that corresponds to a given configuration of the robot $q, \dot{q}, \ddot{q}$. To determine f_i and τ_i the parameters

$a_{c,i}$	acceleration of the center of mass of link i
$a_{e,i}$	acceleration of the end of link i
ω_i	angular velocity of frame i with respect to frame 0
α_i	angular acceleration of frame i with respect to frame 0
g_i	acceleration due to gravity
f_i	the force exerted by link $i-1$ on link i
τ_i	torque exerted by link $i-1$ on link i
R_{i+1}^i	rotation matrix from frame i to $i+1$
z_i	axis of actuation of frame i with respect to frame 0
m_i	the mass of link i
I_i	inertia tensor of link i about a frame parallel to frame i
$r_{i-1,ci}$	vector from origin to frame $i-1$ to center of mass of link i
$r_{i,ci}$	vector from the origin of frame i to the center of mass of link i
$r_{i-1,i}$	vector from the origin of frame $i-1$ to the origin of frame i

Table 3.2: Notation

$a_{c,i}, \omega_i, \alpha_i$ have to be found. This is done in a recursive manner where i increases. After these parameters is found the f_i and τ_i could be found recursively for decreasing i.

Starting with the angular velocity of joint i expressed inertial frame (superscript 0):

$$\omega_i^{(0)} = \omega_{i-1}^{(0)} + z_{i-1}\dot{q}_1 \tag{3.57}$$

Changing to frame i:

$$\omega_i = (R_i^{i-1})^T \omega_{i-1} + b_i \dot{q}_1 \tag{3.58}$$

Where b_i is defined as follows:

$$b_i = (R_i^0)^T R_{i-1}^0 z_0 \tag{3.59}$$

Taking the time derivative of 3.57 to find the angular acceleration in the inertial frame:

$$\dot{\omega}_i^{(0)} = \dot{\omega}_{i-1}^{(0)} + z_{i-1}\ddot{q}_1 + \omega_i^{(0)} \times z_{i-1}\dot{q}_i \tag{3.60}$$

Changing to frame i:

$$\alpha_i = (R_i^{i-1})^T \alpha_{i-1} + b_i \ddot{q}_1 + \omega_i \times b_i \dot{q}_i \tag{3.61}$$

Now the angular velocity and acceleration are found, then the next step is to find the linear acceleration of the center of the mass of link i. Starting with the linear velocity expressed in the inertial frame:

$$v_{c,i}^{(0)} = v_{e,i-1}^{(0)} + \omega_i^{(0)} \times r_{i-1,ci}^{(0)} \tag{3.62}$$

This gives the following acceleration

$$a_{c,i}^{(0)} = a_{e,i-1}^{(0)} \times r_{i-1,ci}^{(0)} + \omega_i^{(0)} \times (\omega_i^{(0)} \times r_{i-1,ci}^{(0)}) \tag{3.63}$$

Changing to frame i:

$$a_{c,i} = (R_i^{i-1})^T a_{e,i-1} + \dot{\omega}_i \times r_{i-1,ci} + \omega_i \times (\omega_i \times r_{i-1,ci}) \tag{3.64}$$

where $a_{e,i}$ is simply the same equation changing from $r_{i-1,ci}$ to $r_{i-1,i}$:

$$a_{e,i} = (R_i^{i-1})^T a_{e,i-1} + \dot{\omega}_i \times r_{i-1,i} + \omega_i \times (\omega_i \times r_{i-1,i}) \tag{3.65}$$

These equations represent the recursive Newton-Euler formulation. For a n-linked industrial robot manipulator the problem is solved by setting all initial conditions equal zero, i.e. $\omega_0 = \alpha_0 = a_{c,0} = ae, 0 = 0$. And the terminal conditions equal to zero, i.e. $f_{n+1} = \tau_{n+1} = 0$. Then the forward recursion is applied by solving equation 3.58, 3.61, 3.65 and 3.64 in the given order for each link $i = 0 \ldots n$. After this the backward recursion is performed by solving equation 3.55 and 3.56 for each link $i = n \ldots 1$. When this is completed equation 3.54 could be formed.

3.3 Third Order Path Planning

In robotics one of the main challenges is how to plan a path. One of the basics methods for planning path is to plan a time dependent for each joint of the robot. This project will later on use more complex path planning strategies, but the basic principles described below is used for small experiments during the project.

The polynomial has the form:

$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 \tag{3.66}$$

$$\dot{q}(t) = a_1 + 2a_2 t + 3a_3 t^2 \tag{3.67}$$

$$\ddot{q}(t) = 2a_2 + 6a_3 t \tag{3.68}$$

At initial time t_i we have:

$$q(t_i) = a_0 + a_1 t_i + a_2 t_i^2 + a_3 t_i^3 = q_i \tag{3.69}$$

$$\dot{q}(t_i) = a_1 + 2a_2 t_i + 3a_3 t_i^2 = \dot{q}_i \tag{3.70}$$

Assuming $t_i = 0$:

$$a_0 = q_i \tag{3.71}$$

$$a_1 = \dot{q}_i \tag{3.72}$$

At final time t_f we have:

$$q(t_f) = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 = q_f \tag{3.73}$$

$$\dot{q}(t_f) = a_1 + 2a_2 t_f + 3a_3 t_f^2 = \dot{q}_f \tag{3.74}$$

Using what we found for a_i $i \in [0, 1]$

$$q(t_f) = q_i + \dot{q}_i t_f + a_2 t_f^2 + a_3 t_f^3 = q_f \tag{3.75}$$

$$\dot{q}(t_f) = \dot{q}_i + 2a_2 t_f + 3a_3 t_f^2 = \dot{q}_f \tag{3.76}$$

Solving the equations gives:

$$a_3 = \frac{2(q_i - q_f) + (\dot{q}_i + \dot{q}_f)t_f}{t_f^3} \tag{3.77}$$

$$a_2 = \frac{-(\dot{q}_f + 2\dot{q}_i)t_f - 3(q_i - q_f)}{t_f^2} \tag{3.78}$$

We have assumed that $t_i = 0$. Now we want a general case for arbitrary t_i and t_f. Defining $T = t_f - t_i$ gives us the same a_0, a_1 as before but:

$$a_3 = \frac{2(q_i - q_f) + (\dot{q}_i + \dot{q}_f)T}{T^3} \tag{3.79}$$

$$a_2 = \frac{-(\dot{q}_f + 2\dot{q}_i)T - 3(q_i - q_f)}{T^2} \tag{3.80}$$

Generalizing the position and velocity as well:

$$q(t) = a_0 + a_1(t - t_i) + a_2(t - t_i)^2 + a_3(t - t_i)^3 \tag{3.81}$$

$$\dot{q}(t) = a_1 + 2a_2(t - t_i) + 3a_3(t - t_i)^2 \tag{3.82}$$

Chapter 4

Identification

4.1 Friction Identification

In mechanical systems friction will occur. For the robot's case friction occurs in the gearboxes, motors and in the joints itself. Identification experiments in [8] have showed that the friction has a nonlinear shape. Detecting the force to overcome friction in an experiment, and using this result to make a feed forward compensation for the friction could help us improve precision. Since the friction will change with time and the identification is done offline, an easy and fast script for detecting the force to overcome friction has to be developed. In this case, offline means that an identification experiment is done prior to the execution.

To detect the force to overcome friction a number of experiments are done. These experiments will detect the force necessary to overcome static friction. Equal experiments are done in [8], where the scope is a general identification for the robot IRB 140. For this project's purpose more limitations are set. Since the scope of this work is precision, high velocities will not be investigated. To detect the force to overcome friction, experiments are done where each joint runs with constant speed one at a time. The goal of this experiment is to make a mapping between velocities and the torque to overcome friction.

The velocity of each joint is planned as follows:

$$\dot{q}_{i,j} = \begin{cases} a_{1,j}t, & t \leq t_{1,j} \\ a_{2,j}, & t_{1,j} < t \leq \frac{T_j}{2} - t_{1,j} \\ -a_{1,j}t, & \frac{T_j}{2} - t_{1,j} < t \leq \frac{T_j}{2} + t_{1,j} \\ -a_{2,j}, & \frac{T_j}{2} + t_{1,j} < t \leq T_j - t_{1,j} \\ a_{1,j}t, & T_j - t_{1,j} < t \leq T_j \end{cases} \tag{4.1}$$

Where $a_{1,j}$ is the acceleration and $a_{2,j}$ is the desired constant velocity. T_j is the time

j	1	2	3	4	5	6	7
$\dot{q}_{i,j}$	0.05	0.1	0.15	0.2	0.25	0.3	0.35

j	8	9	10	11	12	13	14	15
$\dot{q}_{i,j}$	0.4	0.45	0.5	0.6	0.7	0.8	0.9	1

Table 4.1: Joint velocities used in friction experiment

for one period, and $t_{1,j}$ is the time used to accelerate from, in this case, zero velocity to $a_{2,j}$. Where j is the number of the experiments, and i is the joint. The velocities for this project is limited to those presented in table 4.1. Since the last three joints of the robot are controlled together by three motors and a complex gearing system ,it is more difficult to find torque to overcome friction. Therefore this project focuses on the three first joints.

The torque sent to each motor is measured and logged. To get the mapping between velocity and torque, the measured control signal to the motors is multiplied by the inverse gear ratio. Calculated velocity from measurements of position are provided from encoders placed on the motor side. This data is also logged. Since the measurements are on the motor side, they have to be multiplied by the gear ratio to get a mapping between joint velocity and torque. Another velocity measurement is provided by the external camera. The Nikon K610 system is used to track the position of the end effector. From this measurement the velocity of the joint is calculated. This is used as another approach to the velocity - torque mapping. Since the camera measurement also take the dynamics of the robot into account, this could possibly improve the performance of the robot's end effector. The controller used to control the robot in this experiment is attached, see appendix A.1.

Starting with the Robot equation:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau - \tau_{fr} \tag{4.2}$$

Where τ_{fr} is the signal needed to overcome the friction force. This is the signal that should be found. The measured torque τ is the control signal provided to the engine. We assume that the force to overcome friction is independent of direction of rotation. When the torque to overcome friction is detected there is constant velocity, which means zero acceleration:

$$\dot{q} = q_c \tag{4.3}$$

$$\ddot{q} = 0 \tag{4.4}$$

if

$$\dot{q}_c > 0 \tag{4.5}$$

$$\dot{q}_c = c \tag{4.6}$$

$$\rightarrow C(q,c)c + G(q) = \tau_+ - \tau_{fr}(c) \tag{4.7}$$

else

$$\dot{q}_c < 0 \tag{4.8}$$
$$\dot{q}_c = -c \tag{4.9}$$
$$\rightarrow -C(q,-c)c + G(q) = \tau_+ - \tau_{fr}(-c) \tag{4.10}$$
$$C(q,c)c + G(q) = \tau_+ + \tau_{fr}(c) \tag{4.11}$$
$$\tag{4.12}$$

Then we have that

$$\tau_+ - \tau_{fr}(c) = \tau_+ + \tau_{fr}(c) \tag{4.13}$$
$$\tau_{fr}(c) = \frac{\tau_+ - \tau_-}{2} \tag{4.14}$$

The assumption above makes it possible to distinguish the gravitational force and the Coriolis and centrifugal forces, from the force to overcome friction. These forces could also be found in individual identification experiments.

4.1.1 Results

The results from the experiment are a static mapping between applied force to overcome friction and velocity. The result is presented in figure 4.1, 4.2 and 4.3. Both the static friction map with velocities from the encoder and the camera are presented in the figures. As the figures show, the results are quite similar. It will be interesting to see, from validation, which compensation that gives best performance.

Since the physics is different for each joint, the result is also quite different for each joint. Therefore, different orders of polynomials are used to represent the force to overcome friction. For the first joint a polynomial of order 4 is used, for the second joint a polynomial of order 6 is used and for the third joint a polynomial of order 7 is used. The same order is used both for the positive and negative velocity.

To improve the performance of the robot the offline calculation of the control force to overcome friction is used in a feed forward control. This is done by modifying the feed forward signal sent to the controller in figure 2.2. This makes it possible for the controller to take advantage of the offline calculated knowledge about the friction. It is important to notice that the parameters of the feed forward are only reliable in a period because of changes in the dynamics of gear oil etc.

Figure 4.1 to 4.3 shows that for velocities close to zero there is a step in the force. To avoid problems with switching, it could be smart to rely on the reference signal instead of the estimated velocity. Since there is no velocity measurement, it has to be estimated based on the position measurements from the encoders. This could result in some noise and error, therefore a method for choosing which signal to rely on is needed. Different techniques are tested. First a pure switching algorithm is used for velocities close to zero.

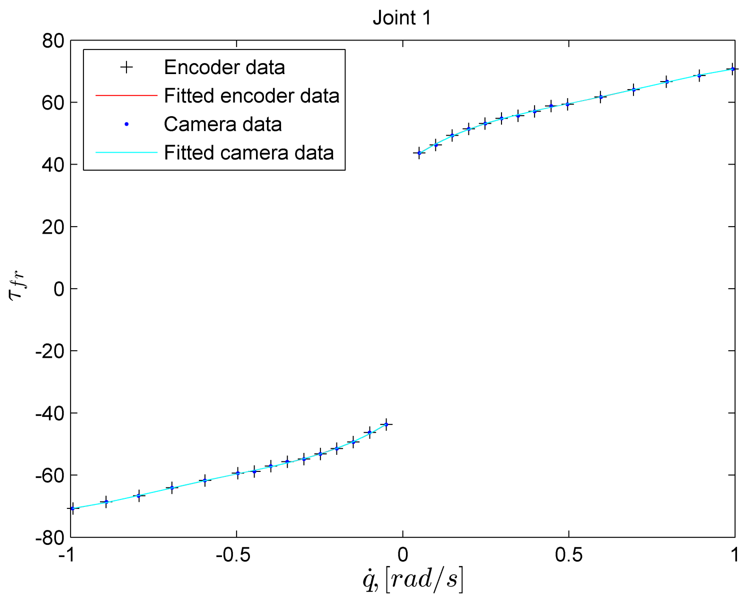


Figure 4.1: Static friction map first joint

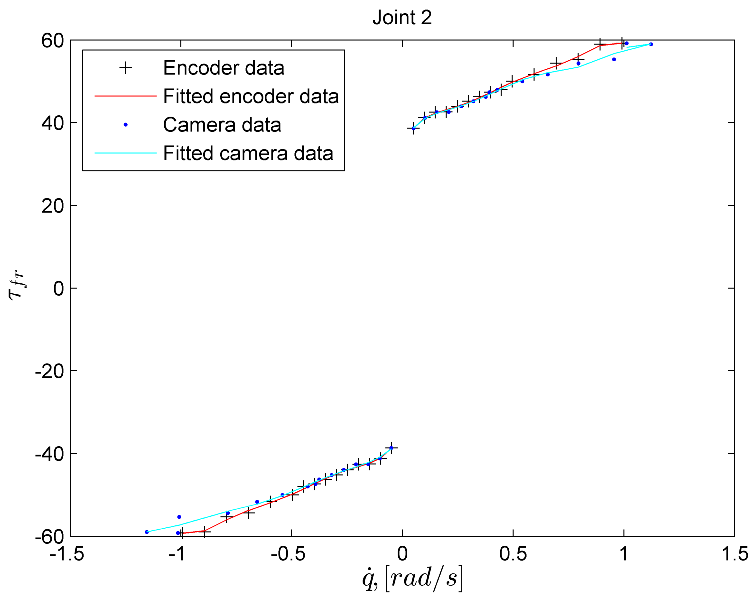


Figure 4.2: Static friction map second joint

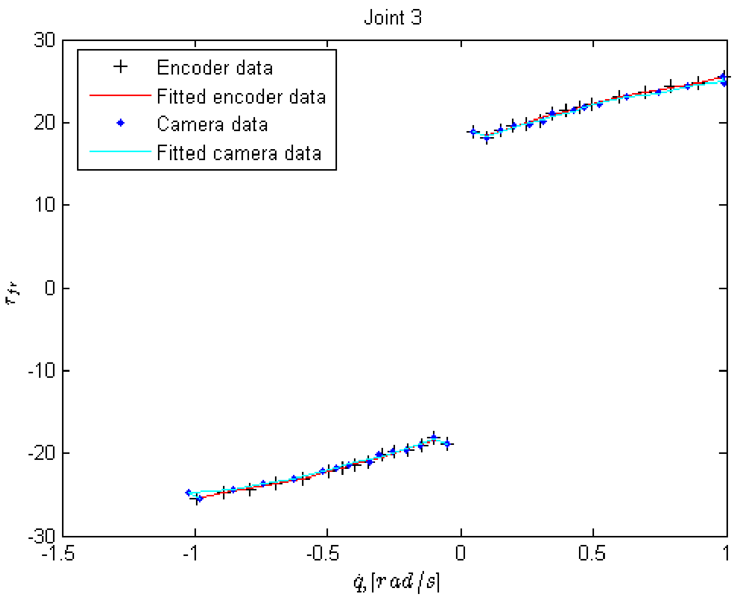


Figure 4.3: Static friction map third joint

This gave acceptable results, but to further improve the performance a bell curve strategy is implemented. The bell curve is shown in figure 4.5, and the following formula is used:

$$dq = \alpha dq_{ref} + (1 - \alpha)dq_{encoders} \tag{4.15}$$

The alpha function is calculated as follows:

$$\alpha = a \exp\left(-\frac{dq_{ref}^2}{2c^2}\right) \tag{4.16}$$

where $a = 1$ and $c = 1/2, 1/4, 1/8$. The bell curves are shown in figure 4.4 to 4.6. Experiments showed that best results are achieved with $c = 1/4$.

From this calculated velocity the feed forward controller did the following:

```
if dq < 0
        calculate Feed Forward using lower polynomial
else
        calculate Feed Forward using upper polynomial
end
```

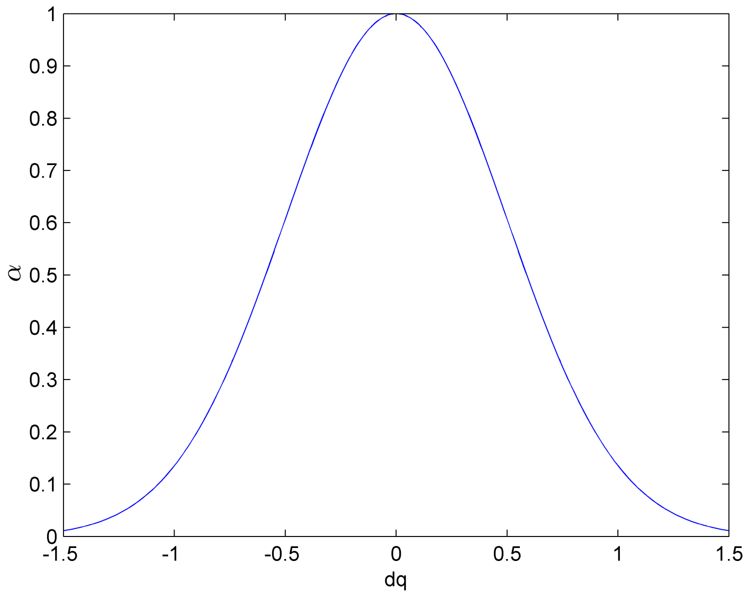


Figure 4.4: Bell curve used for choosing velocity signal c = 1/2

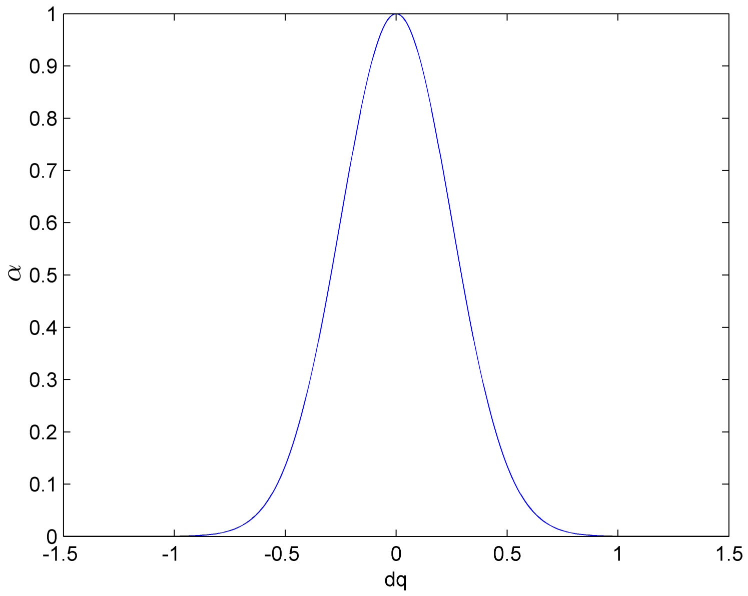


Figure 4.5: Bell curve used for choosing velocity signal c = 1/4

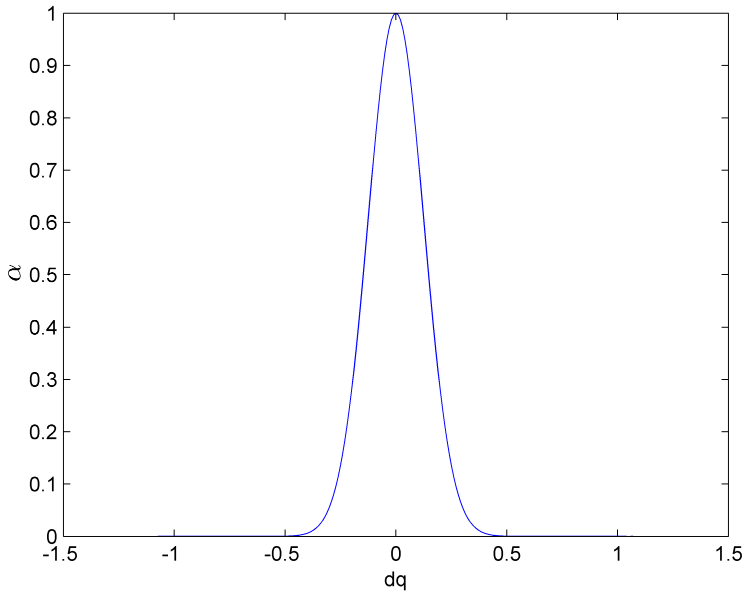


Figure 4.6: Bell curve used for choosing velocity signal c = 1/8

This feed forward signal is calculated in a Simulink block, and sent to the controller shown in figure 2.2. This controller is called ”frictionExperimentValidation.mdl”, and a attached in a file. See appendix A.1.

4.1.2 Validation of Results on Single Joints

To validate the static friction map experiments is done. The purpose of the validation is to show that by adding the friction compensation to the controller the performance of the robot increase both on the motors side measured by encoders, and at the end effector measured by the external camera software. To do this, each joint is tested individually with a sinus path, planned as follows for each joint q_i individually. Each joint will be tested with three different maximum velocities, i.e. $a = 0.1, 0.5, 1 rad/s$.

Selecting velocity to be:

$$\dot{q}_i = a \sin(\omega t) \tag{4.17}$$

then we have

$$q_i = -\frac{-a}{\omega} \cos(\omega t) + q_i(0) + q_{max} \tag{4.18}$$

Where $q_i(0)$ is the start position of the robot in the current experiment, and q_{max} is the peak amplitude, i.e. half of the travel distance in radians. The controller is attached, see appendix A.1.

For each joint and velocity it is tested with three different configurations:

- Without feed forward
- With feed forward from encoders mapping
- With feed forward from camera mapping

Testing reveals that it is difficult to show the improvement of feed forward on an individual joint. The fact that a good PID controller is developed for the external control software makes it hard to see any change on the robot performance neither on the encoder measurement nor by the camera measurements. Since the purpose of this experiment is to validate the friction model, and not to validate the PID controller, it is decided to turn of the integral part. From table 2.1 it is shown that the integral term of the PID controller is very high. This will most likely cancel out the signal from the feed forward term. For the PID controller the feed forward term will occur as a constant error. With a high integral term the controller will cancel out constant error very fast. Therefore it is reasonable to set the integral gain equal zero, in the purpose of validating the feed forward term.

Different methods for evaluating the data from the validation experiments are tested out. Due to sampling time and delay in mechanisms the measured joint position from encoders is approximately $12-16ms$ behind the reference. The sampling time is $4.032ms$, therefore the signals is compared with an offset of 3 samples. This is tested and validated to give the best result. Due to the fact that the signals are not continuous, there is still a shift in time between the reference and the measured signal. This will result in a intersection between the reference and the measurement, which will result in zero error. Due to the sinus reference this occurs at turning point of the path, i.e. zero velocity points. Therefore, some of the results presented here gets zero error at the start and at the end, this is only a result of sampling and is not representable for the error along a path for the robot. Actually, it is assumed that the greatest error will occur at the turning points of the sinus path. This is related to backlash in gears, that will occur when switching driving direction of the motors.

The results presented here will only be the middle velocity $0.5rad/s$. The results from the experiments with $0.1rad/s$ and $1rad/s$ i presented in appendix B. Since the lowest velocity is very close to zero velocity, where the model is badly defined, and the highest velocity is up to the maximum of what the model is made for, the middle velocity is assumed to give the best picture of the performance.

Both results from encoder measurements and camera measurements is presented here. The comparison is done in radians for simplicity. Later on when evaluating the robot precision the comparison is done in x,y,z coordinates.

Joint 1 Encoder Results

For the first joint the results are presented in figure 4.7, 4.8 and 4.9. The result in figure 4.7 shows the performance of the robot with only the PD controller. The maximum difference in error is shown at $-0.8rad$ approximately. Here the error is between $-3.8 \cdot 10^{-4}rad$ and $3.4 \cdot 10^{-4}rad$, in total $7.2 \cdot 10^{-4}rad$.

In figure 4.8 a feed forward term from the encoder friction model is used in addition to the PD controller. It is clear that there is an improvement from the result without any friction compensation. Here the error is between $-2.1 \cdot 10^{-4}rad$ and $3.0 \cdot 10^{-4}rad$, in total $5.1 \cdot 10^{-4}rad$. This means an improvement of approximately 30 percent. From the figure we see that the improvement of the error is approximately the same all along the path. We also see a high improvement at the start and end of the path, at $1.3rad$ and $0.15rad$. Where the error without friction compensation has a blunt shape at the start and end the error with friction compensation is sharper.

In figure 4.9 a feed forward term from the camera friction model is used in addition to the PD controller. The improvement of this friction compensation model is approximately the same as for the encoder model. The error is between $-2.4 \cdot 10^{-4}rad$ and $2.8 \cdot 10^{-4}rad$, in total $5.2 \cdot 10^{-4}rad$. This is the same as for the encoder model. There is a difference between the result in figure 4.8 and 4.9, that is a shifting around zero. A great number of experiments are done and it seems like this is happening randomly. A theory could be that because of the canceling of the integrator term it appear a stationary error in the process of switching between the ABB controller and our own made controller. Since there is no integral part in the controller this will not be canceled out. The process of switching between ABB controller and our own controller includes the step of release breaks. This means that the controller joy stick of the robot has to be pushed, which could result in a very small movement of the robot, that could give us this stationary deviation.

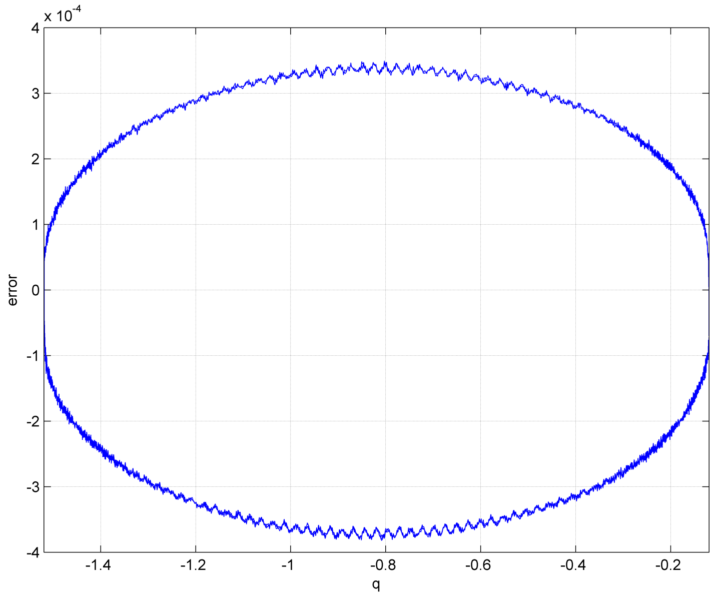


Figure 4.7: 1. Joint — No feed forward

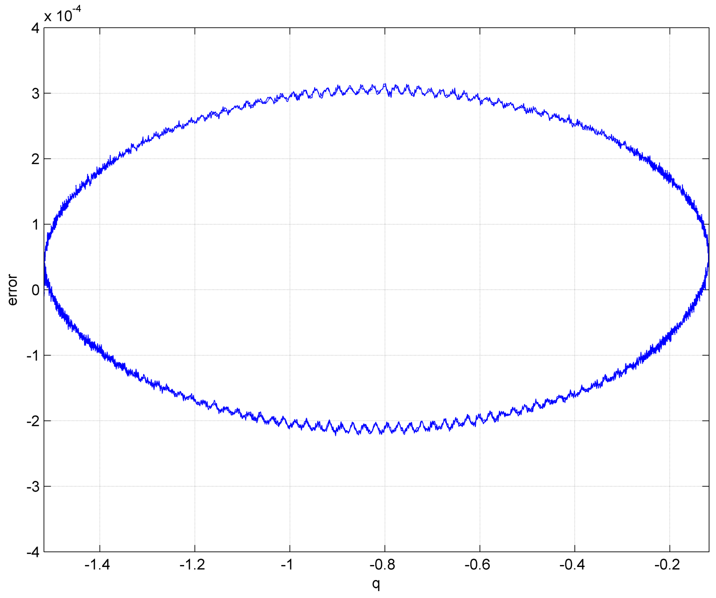


Figure 4.8: 1. Joint — Encoder feed forward

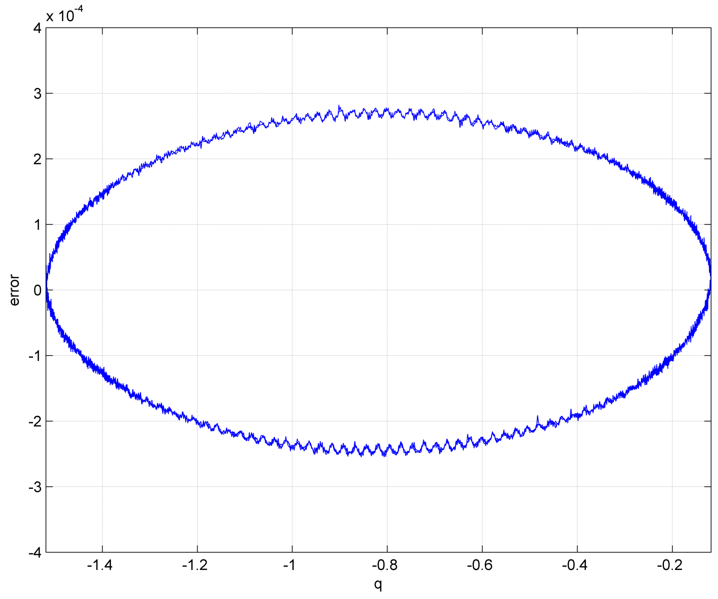


Figure 4.9: 1. Joint — Camera feed forward

Joint 2 Encoder Results

For the second joint the results are presented in figure 4.10, 4.11 and 4.12. The result in figure 4.10 shows the performance of the robot with only a PD controller, and no friction compensation. Here the error is between $-2.5 \cdot 10^{-4} rad$ and $3.5 \cdot 10^{-4} rad$, in total $6 \cdot 10^{-4} rad$.

In figure 4.11 a feed forward term from the encoder friction model is used in addition to the PD controller. Here the error is between $-1.0 \cdot 10^{-4} rad$ and $3.0 \cdot 10^{-4} rad$, in total $4 \cdot 10^{-4} rad$. This is an improvement of approximately 33 percent. Also here we see an improvement both at the middle of the motion, at q equal $1.5 rad$, and at the start and end, at $1.05 rad$ and $1.9 rad$. We see that the improvements are most on the negative side. The positive and the negative sign of the error represents positive and negative direction of the motion. This could either be related to the stationary deviation discussed for joint 1, or the fact that the friction model is better for one of the directions.

In figure 4.12 a feed forward term from the camera friction model is used in addition to the PD controller. The results are very equal to the one in figure 4.11. The only viewable difference is the shift of the center. Also here an improvement of 33 percent occurs when implementing the friction compensation model.

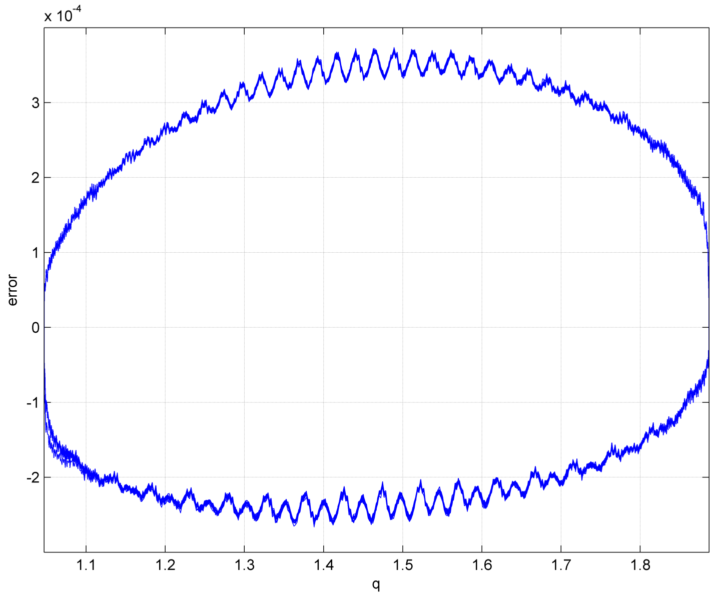


Figure 4.10: 2. Joint — No feed forward

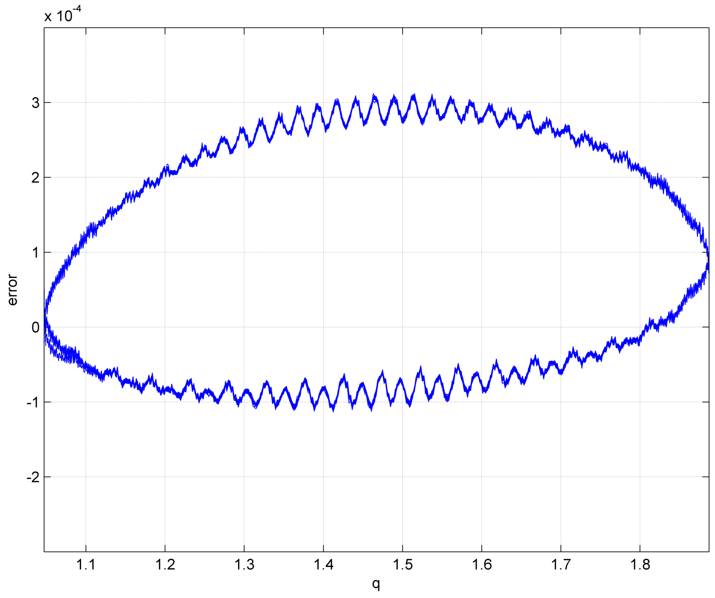


Figure 4.11: 2. Joint — Encoder feed forward

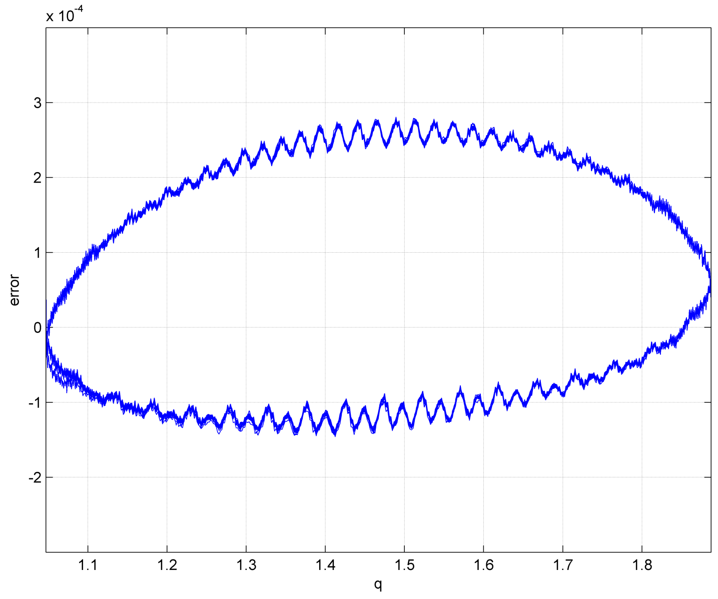


Figure 4.12: 2. Joint — Camera feed forward

Joint 3 Encoder Results

For the third joint the results are presented in figure 4.13, 4.14 and 4.15. The result in figure 4.13 shows the performance of the robot with only a PD controller, and no friction compensation. Here the error is between $-7 \cdot 10^{-4} rad$ and $7 \cdot 10^{-4} rad$, in total $14 \cdot 10^{-4} rad$.

In figure 4.11 a feed forward term from the encoder friction model is used in addition to the PD controller. Here the error is between $-3.5 \cdot 10^{-4} rad$ and $4.2 \cdot 10^{-4} rad$, in total $7.7 \cdot 10^{-4} rad$. This is an improvement of approximately 45 percent. It is interesting to see that the friction compensation for the last joint is significantly better than the friction compensation for the first two joints. There is different shape of the error depending on the sign. This means that the error is different depending on the direction of the motion. This could be a result of a better compensation model for one or the other direction, but since the error with no feed forward has the same shape it is probably just related to the motion itself.

In figure 4.15 a feed forward term from the camera friction model is used in addition to the PD controller. The results are very equal the one in figure 4.14. The only viewable difference is a very small shift of the center. Also here a improvement of 45 percent occurs when implementing the friction compensation model.

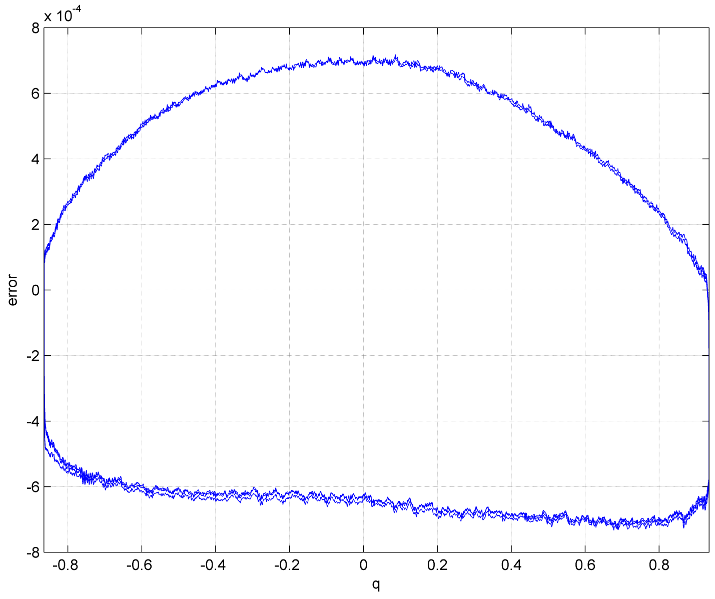


Figure 4.13: 3. Joint — No feed forward

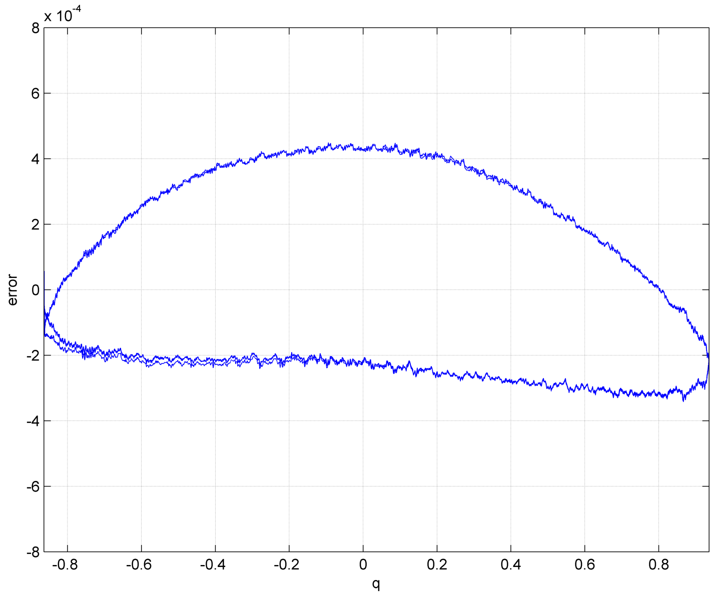


Figure 4.14: 3. Joint — Encoder feed forward

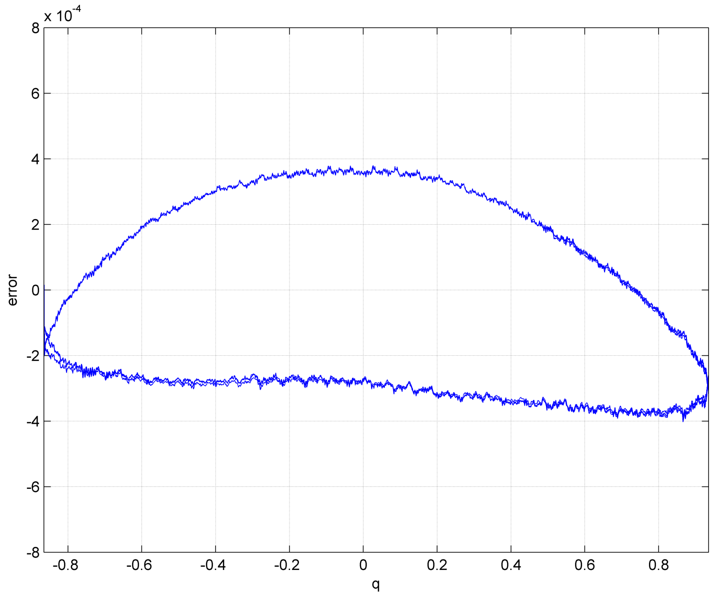


Figure 4.15: 3. Joint — Camera feed forward

Joint 1 Camera Results

For the first joint the camera measurement results are presented in figure 4.16, 4.17 and 4.18. The improvement shown earlier from the encoder measurement is no longer present. Both with compensation from camera friction model and from encoder friction model the result is almost identical to the result from the experiment without compensation. The 30 percent improvement is not visible for the camera. It is worth mentioning that the total error in this case is $12.5 \cdot 10^{-4} rad$. This is significantly higher than the error of $6 \cdot 10^{-4} rad$ from the result in figure 4.10.

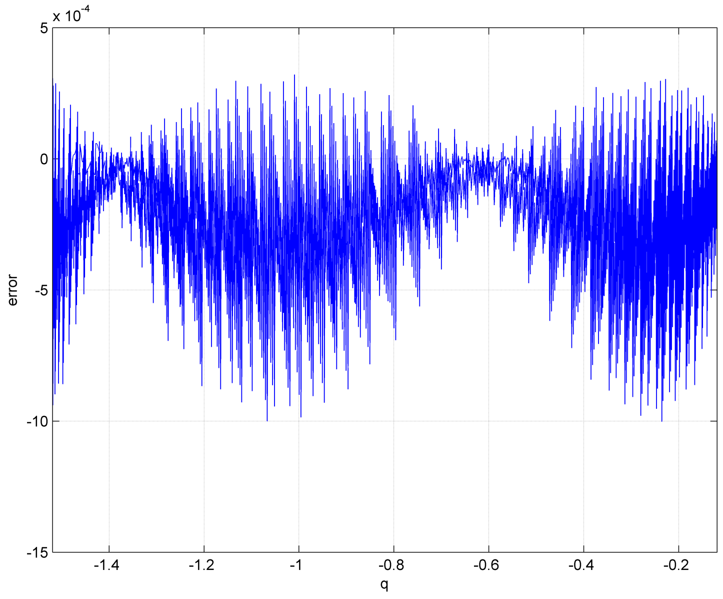


Figure 4.16: 1. Joint — No feed forward

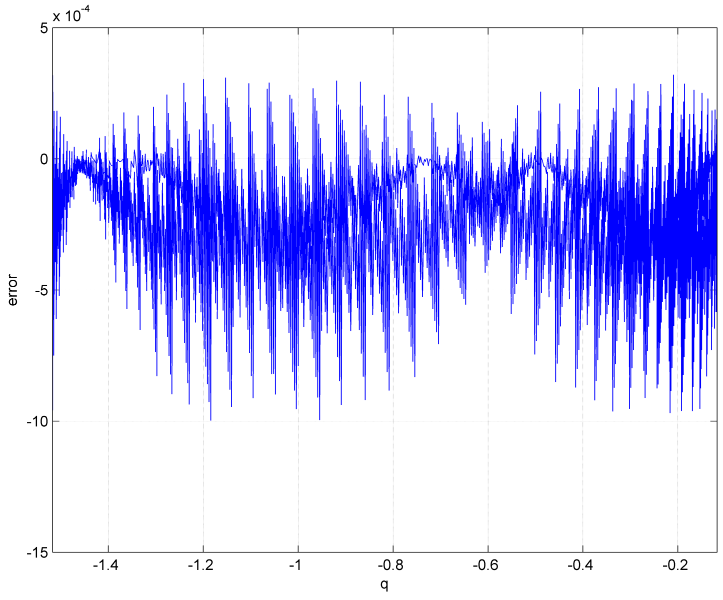


Figure 4.17: 1. Joint — Encoder feed forward

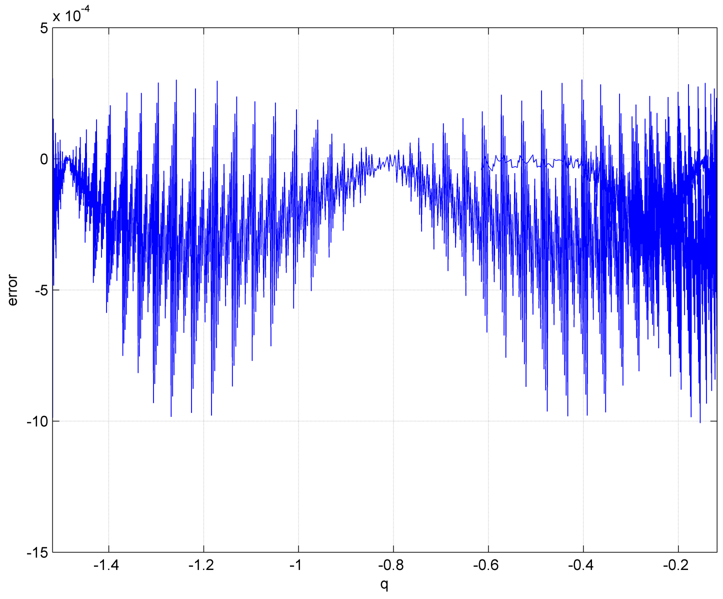


Figure 4.18: 1. Joint — Camera feed forward

Joint 2 Camera Results

For the second joint the the camera measurement results are presented in figure 4.19, 4.20 and 4.21. The improvement shown earlier from the encoder measurement is neither here present. Both with compensation from camera friction model and from encoder friction model the result is almost identical to the result from the experiment without compensation. The total error in this case is $19 \cdot 10^{-4} rad$. This is significantly higher than the error of $5.2 \cdot 10^{-4} rad$ from the result in figure 4.7.

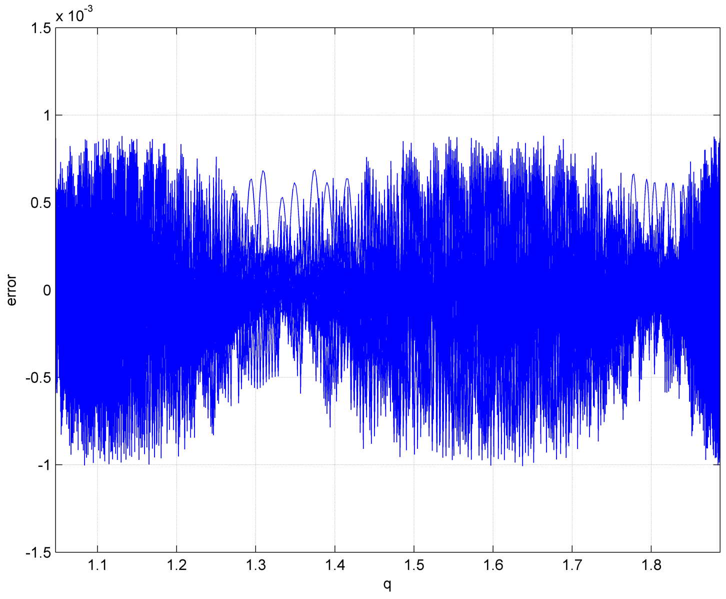


Figure 4.19: 2. Joint — No feed forward

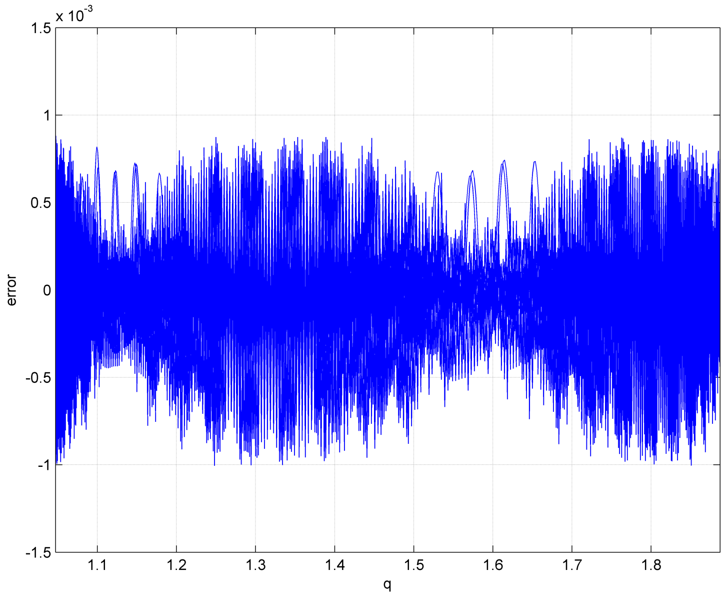


Figure 4.20: 2. Joint — Encoder feed forward

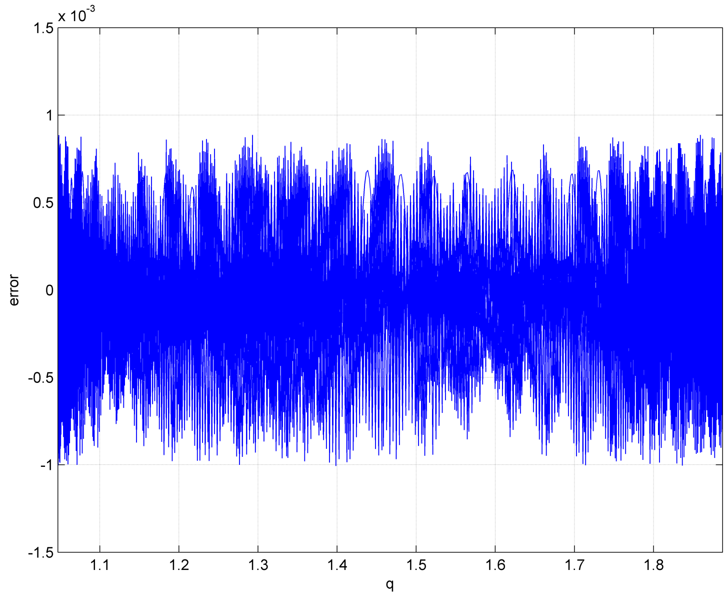


Figure 4.21: 2. Joint — Camera feed forward

Joint 3 Camera Results

For the third joint the camera measurement results are presented in figure 4.22, 4.23 and 4.24. Neither here any significant improvement is visible. The fact that on the encoder measurement the improvement was approximately 45 percent makes it strange that no improvement is visible on the camera measurement.

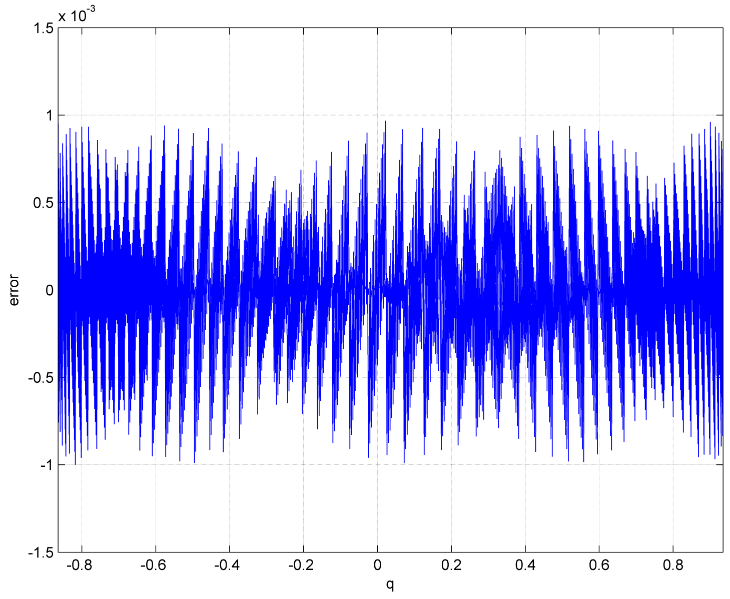


Figure 4.22: 3. Joint — No feed forward

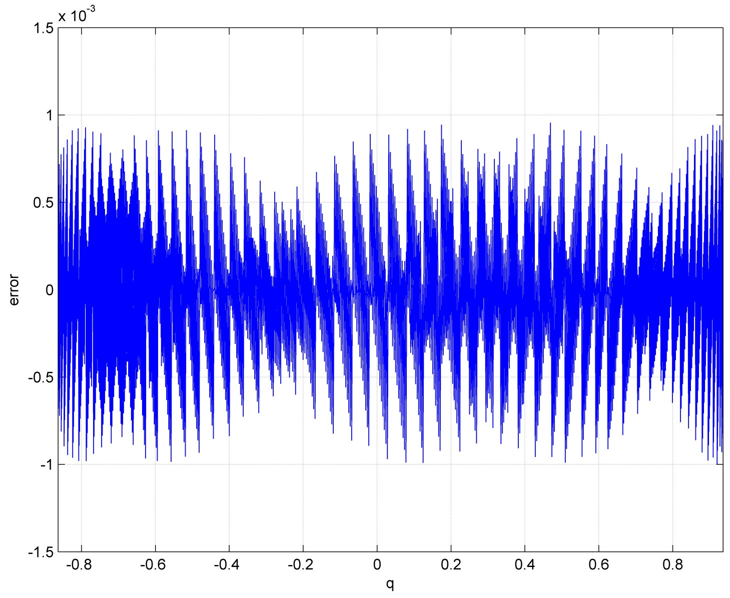


Figure 4.23: 3. Joint — Encoder feed forward

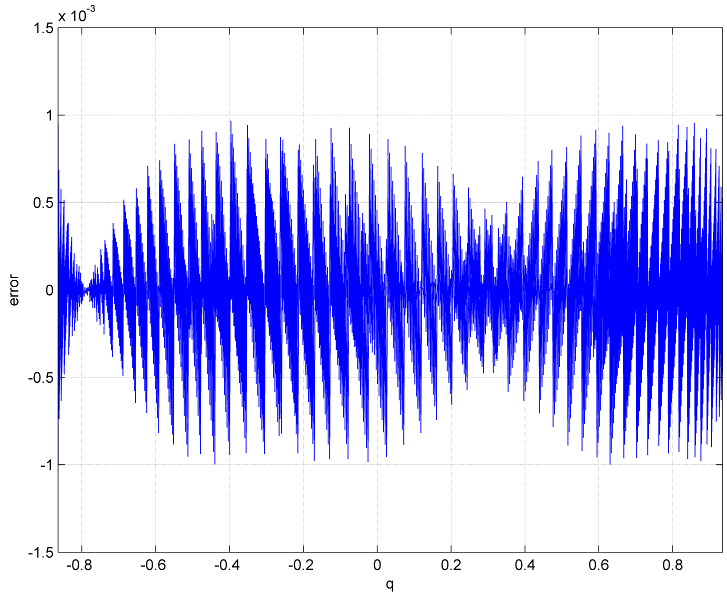


Figure 4.24: 3. Joint — Camera feed forward

4.1.3 Validation of Results Circle Movement

The above results confirm that the new friction model improves the performance of a single joint. It also shows that it is hard to see any improvement on the camera side. Therefore it is reasonable to try another approach to validate the friction models that are developed. Since the task for this report will involve collaborative work by all three joints, it is reasonable to take a look at the increase of performance in collaborative work. For this validation experiment a circle motion is chosen, where the radius (R) of the circle is set to $15cm$.

The motion of the robot is planned as a motion in task space, and then using trigonometry rewritten in joint space. The generalized coordinate is chosen to be θ which is the angle of the position of the robot on the circle.

Starting with describing theta using third order path planning as described in section 3.3.

$$\theta = k_0 + k_1(t - ti) + k_2(t - ti)^2 + k_3(t - ti)^3 \tag{4.19}$$

$$d\theta = k_1 + 2k_2(t - ti) + 3k_3(t - ti)^2 \tag{4.20}$$

The path could be parametrized as:

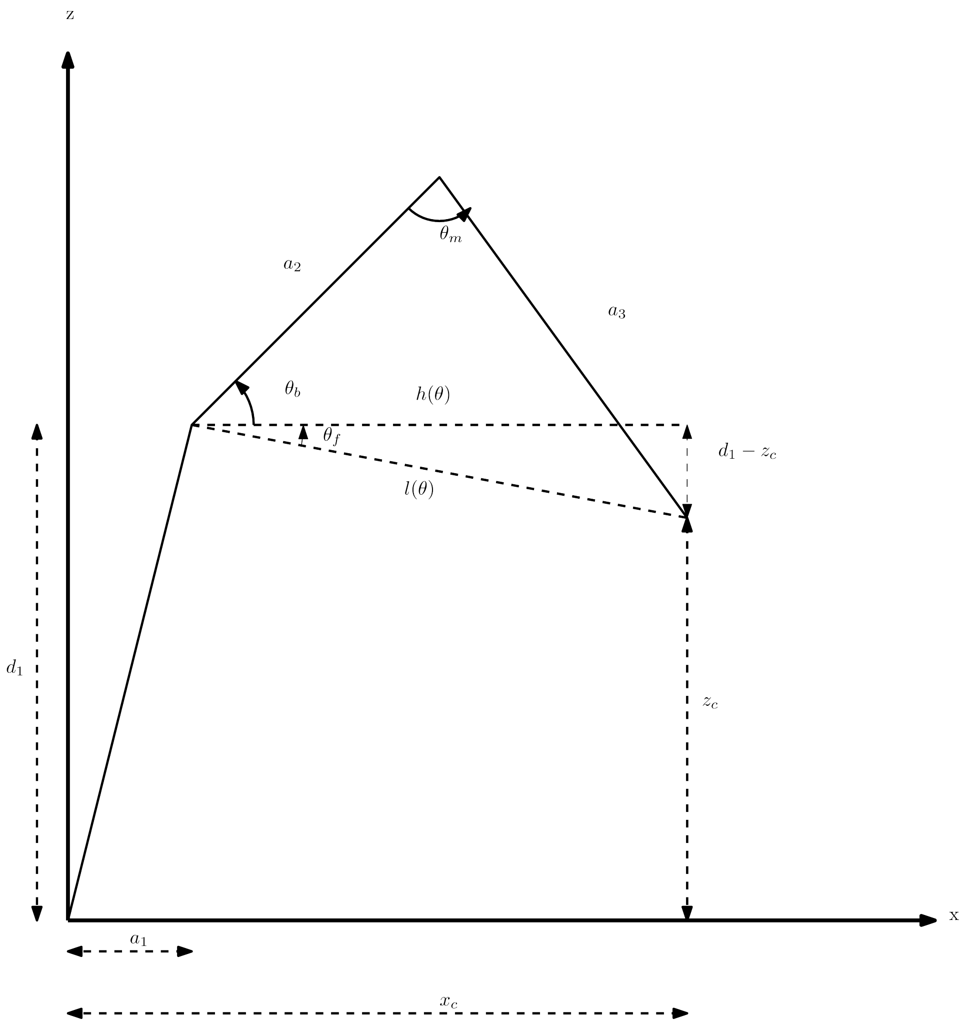


Figure 4.25: 3 DOF robot model Sketch

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} R\cos(\theta) + x_c \\ R\sin(\theta) \\ z_c \end{bmatrix} \tag{4.21}$$

Where x_c and z_c are parameters to be chosen. For this experiment we have

$$x_c = 0.6 \tag{4.22}$$

$$z_c = 0.13 \tag{4.23}$$

$$R = 0.15 \tag{4.24}$$

The next parameters are calculated from trigonometry from figure 4.25. The parameters a_1, a_2, a_3 and d_1 are the DH parameter of the 3 DOF model.

$$q1 = \arctan\left(\frac{R\sin(\theta) + y_c}{R\cos(\theta) + x_c}\right) \tag{4.25}$$

$$h(\theta) = x_c - a1 + R\cos(\theta) \tag{4.26}$$

$$l(\theta) = \sqrt{(d_1 - z_c)^2 + (h(\theta))^2} \tag{4.27}$$

$$\theta_m = \arccos\left(\frac{a_3^2 + a_2^2 - l^2}{2a_2a_3}\right) \tag{4.28}$$

$$\theta_f = \arctan\left(\frac{d_1 - z_c}{x_c - a_1 + R\cos(\theta)}\right) \tag{4.29}$$

$$theta_b = \arcsin\left(\frac{a_3\sin(\theta_m)}{l}\right) - \theta_f \tag{4.30}$$

$$q2 = \frac{\pi}{2} - \theta_b \tag{4.31}$$

$$q3 = \frac{\pi}{2} - theta_m \tag{4.32}$$

From these equations Maple is used to calculate the derivatives. This file is attached see appendix A.2.

For the experiments there are used two different travel times of the motion, i.e. 10 and 15 seconds. Both measurements from encoders and from the camera are recorded and compared. The controller used to perform the motion is attached as a Simulink file named ”FrictionExperimentValidation_CircleMotion.mdl”, see appendix A.1.

To compare the performance of the robot the shortest distance in task space is used:

$$error = \sqrt{(x_{meas} - x_{ref})^2 + (y_{meas} - y_{ref})^2 + (z_{meas} - z_{ref})^2} \tag{4.33}$$

The results are plotted against time, and therefore it is not comparable along the x-axis.

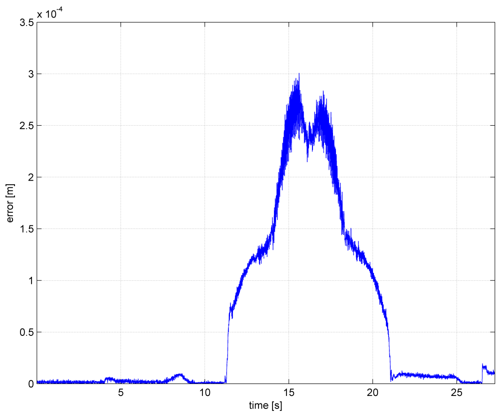
High velocity, 10 s travel time

In figure 4.26a,4.27a and 4.28a the results measured by the encoder are presented. The first figure is without friction compensation, the second is with friction compensation from encoder model and the last is with friction compensation from camera model. It shows an improvement in the size of what we found on the single joint. In percent it is a little over 30 percent, which is reasonable taken into account that the improvement on the single joints 1,2,3 were 30, 30 and 45 percent. In numbers the error without feed forward is mostly around 260 microns, while with feed forward it is reduced to 180 microns. From the plots it seems like the result in figure 4.27a is the best. This means that the encoder friction compensation model give the best result.

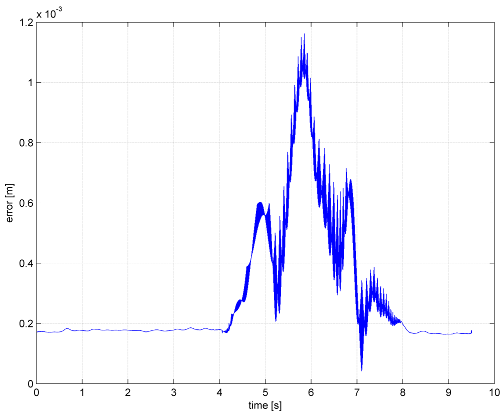
In figure 4.26b,4.27b and 4.28b the results measured by the camera are presented. The first figure is without friction compensation, the second is with friction compensation from encoder model and the last is with friction compensation from camera model. Neither here any improvement occurs using any friction compensation model. It is interesting that there is a error before and after the motion on the camera measurements. This is probably related to errors in the DH parameters of the robot. If those are wrong the reference and the camera measurement will deviate even if the robot is at a stationary position. The difference is approximately 200 microns, which is quite high. ABB claims that the position repeatability of the robot is 20 microns, which is much less than the deviation shown. The total error measured by the camera is at maximum around 1.1 millimeters. This is much more than the 180 microns measured by the encoders, so it is clear that another measurement device is needed to get a picture of the performance of the robot.

Low velocity, 15 s travel time

In figure 4.29a,4.30a and 4.31a the results measured by the encoder are presented. The first figure is without friction compensation, the second is with friction compensation from encoder model and the last is with friction compensation from camera model. The improvement from the friction compensation model is even better for this lower velocity, than what we saw with the high velocity. The error without feed forward is mostly around 220 microns, while with feed forward it is reduced to 120 microns. This is an improvement of approximately 45 percent, which shows that the friction model definitely improves the performance of the controller. It is difficult to see any large differences between the camera friction model and the encoder friction model. From the figures 4.30a and 4.31a it seems like the encoder compensation is a little better, but not any significant difference.

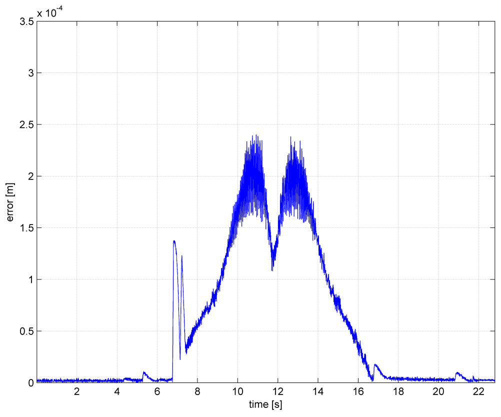


(a) Encoder Measurement

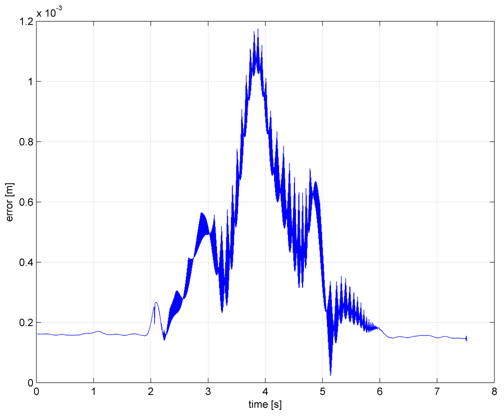


(b) Camera Measurement

Figure 4.26: Path travel time 10 s — No Feed Forward



(a) Encoder Measurement

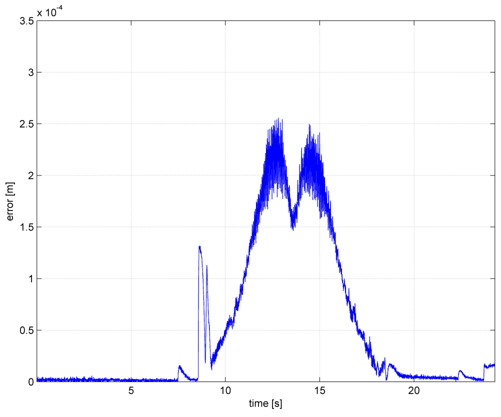


(b) Camera Measurement

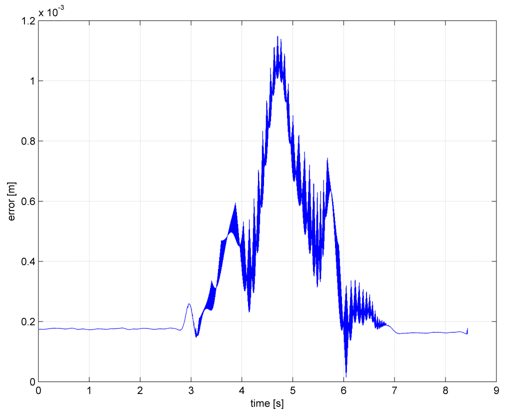
Figure 4.27: Path travel time 10 s —Encoder Feed Forward

In figure 4.29b,4.30b and 4.31b the results measured by the camera are presented. The first figure is without friction compensation, the second is with friction compensation from encoder model and the last is with friction compensation from camera model. Neither here any significant improvement occurs. The error is approximately the same as for the high velocity case, up to 1.1 millimeters. The same stationary error occurs before and after the motion, as described in the low velocity case.

The results from the camera measurement does not show any significant improvement neither on the single joint experiments nor on the collaborative experiment. The camera

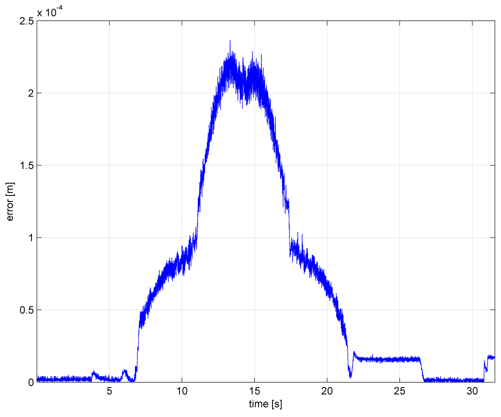


(a) Encoder Measurement

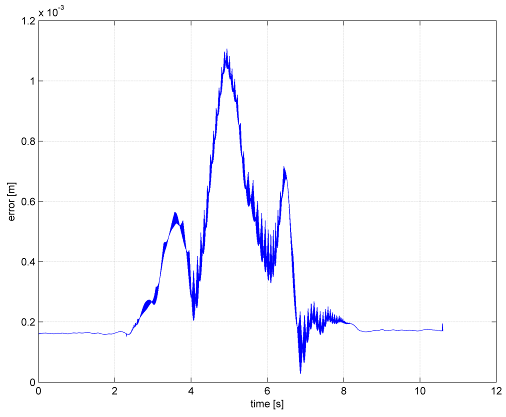


(b) Camera Measurement

Figure 4.28: Path travel time 10 s —Camera Feed Forward



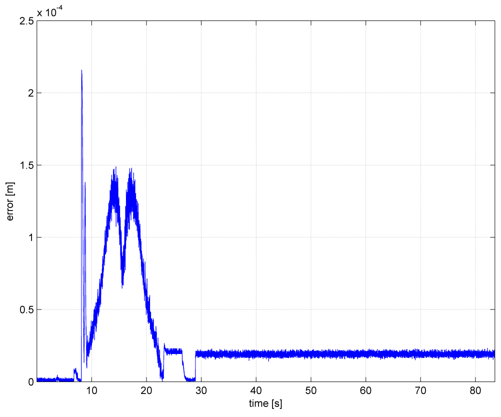
(a) Encoder Measurement



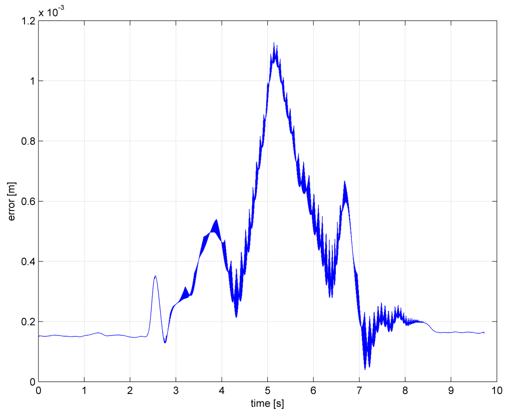
(b) Camera Measurement

Figure 4.29: Path travel time 15 s — No Feed Forward

software has, as discussed earlier, very high precision and it would be reasonable to expect that any improvement shown on the encoder measurement also should be visible from the camera measurement. A number of experiments, not presented here, are completed trying to get any improvement also on the camera measurement. Different methods for definition of frame, as discussed in section 2.4, is tested. To validate that the feed forward term actually effect the controller, all controller gains are set to zero. Then the robot's first joint is run only controlled by the friction compensation model. This experiment validated that the feed forward term affects the controller. A very large number of experiments are done, in order to see if the results are consistent. All the results give the same picture of the

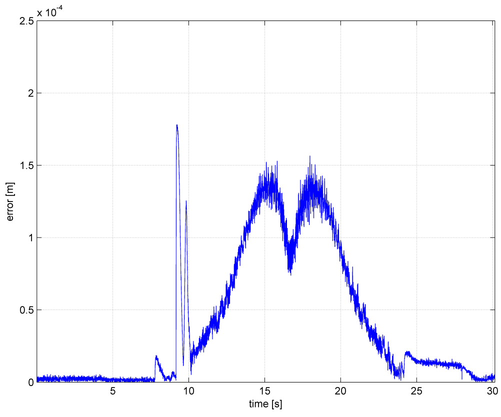


(a) Encoder Measurement

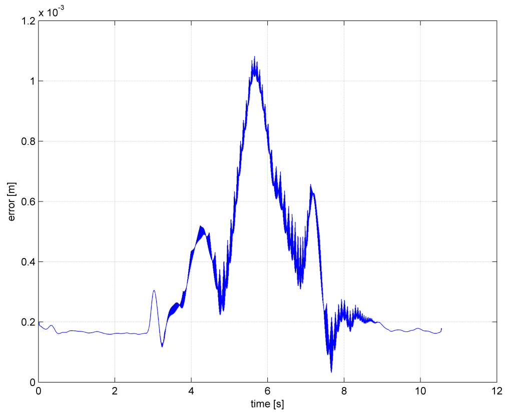


(b) Camera Measurement

Figure 4.30: Path travel time 15 s —Encoder Feed Forward



(a) Encoder Measurement



(b) Camera Measurement

Figure 4.31: Path travel time 15 s —Camera Feed Forward

situation, it is difficult to see any improvement on the camera measurement. Even though the problem is widely tested it could be some causes of error. It is worth mentioning that the process of defining frames, as described in section 2.4, is very challenging. Very small movement of the camera under the defining process could for example give error in the result. The tool, placed at end of the robot, could also cause errors if it is not rigid enough. The fact that the camera sees the end effector position implies that it sees all the dynamics of the robot. The measurement from the encoder , which is placed on the motor side, does not see any of the dynamics of the robot. This measurement only represents the position of the motor. There are both a gear box and the robot arm dynamics on the other side of the motor that the encoder measurement does not take into account. It is a challenging task using this system to confirm any improvement on the robot performance in this experiment, hopefully we will see some greater improvements measured by the camera later on in this report.

The results from the camera measurements indicated that there is an error when the robot is in a fixed position, i.e. before and after the motion starts. This is related to a difference between the reference and the measured position by the camera. Since the robot itself has very high accuracy at such points, 20 microns, it is reasonable to believe that there is another error. Using the fact that the reference position is calculated through the DH parameter of the robot, it is reasonable to believe that those parameters are not accuracy enough. A solution to this is to use the camera equipment to find more precise DH parameters of the robot. Another explanation could be error in the definition of frames on the camera system. A small deviation of the angle of the base coordinate system will give sufficiently deviation at a long arm of the robot. These are topics that could be investigated further.

The results presented in this chapter show that taking advantage of the knowledge about friction, the performance of the robot increases by 30 to 45 percent depending on velocities and which joint you are looking at. It is shown improvement on performance on each single joint, and it is also shown an improvement in a collaborative experiment. The increase in performance for the collaborative experiments shows highest improvement for the low velocity case. Since the scope of this report is precision of the robot, velocity is not important at all.

4.2 Identification of the Robot Equation

The robot equation 3.54 has to be found in order to formulate an optimization problem of the performance of the robot. The robot equation will also be used to make a more advanced controller that takes the dynamics of the robot into account. This is done by using the Newton-Euler formulation described in section 3.2.

To be able to calculate the dynamic robot equation, the center of masses and inertia of each link has to be found. To achieve the most accurate model of the robot the center of masses and inertia should be found experimentally. In [8] methods for experimentally determining center of masses and inertia are shown. Due to limited time, such experiments are not used here. The producer of the robot does not provide information about the center of masses or the inertia. They only provides a CAD model of the robot. Using this model, SolidWorks and the ”Mass Properties” function, the center of masses and inertias are found.

4.2.1 Masses and Center of Masses

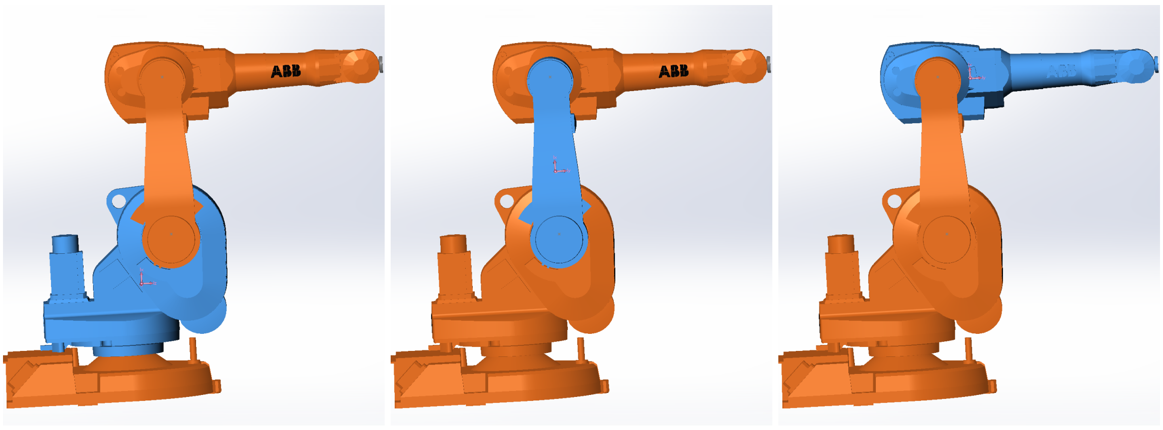
To use the SolidWorks for estimation of masses, center of masses and inertias the SolidWorks model is downloaded from the ABB web page, the full link is provided here [3].

Link	Mass [kg]	Mass Center x [m]	Mass Center y [m]	Mass Center z [m]
1	104.3934	0.052	-0.3433	-0.0119
2	20.4525	0.2019	-0.0002	-0.1828
3	50.5887	0.0968	0.0111	0.0153

Table 4.2: Identification from SolidWorks

SolidWorks can estimate the mass of the object based on the volume and the material. From default the mass calculations estimate a total mass of the robot of 546.5725 kilograms. From the data sheet [4] the total mass of the robot is claimed to be 250 kilograms. To get the correct mass of the robot the mass of each element is multiplied by the factor $\frac{546.572}{250} = 2.186288$. The masses of each single joint are presented in table 4.2.

To get the correct distances two coordinate systems are made in SolidWorks, one at the second and one at the third joint. The distances are calculated from the starting point of the link and to the center of mass, while the distances is expressed in the coordinate frame at the end of the link. The center of masses is shown in figure 4.32. This gives us the distances shown in table 4.2.



(a) 1. Link **(b)** 2. Link **(c)** 3. Link

Figure 4.32: Center of mass and Inertia Tensor

From table 4.2 the vectors below are made:

$$\vec{r}_{0,c1} = \begin{bmatrix} 0.052 & -0.3433 & -0.0119 \end{bmatrix}^T \tag{4.34}$$

$$\vec{r}_{1,c2} = \begin{bmatrix} 0.2019 & -0.0002 & -0.1828 \end{bmatrix}^T \tag{4.35}$$

$$\vec{r}_{2,c3} = \begin{bmatrix}0.0968 & 0.0111 & 0.0153\end{bmatrix}^T \tag{4.36}$$

The following vectors come from the DH parameters of the 3 DOF model:

$$\vec{r}_{0,1} = \begin{bmatrix}a_1 & -d_1 & 0\end{bmatrix}^T \tag{4.37}$$

$$\vec{r}_{1,2} = \begin{bmatrix}a2 & 0 & 0\end{bmatrix}^T \tag{4.38}$$

$$\vec{r}_{2,3} = \begin{bmatrix}a3 & 0 & 0\end{bmatrix}^T \tag{4.39}$$

4.2.2 Inertia

In the same manner as for the masses, SolidWorks and the ”Mass Properties” function is used to find the inertia matrix. Each link is selected separately and the frame at the start of the link is selected as the coordinate system that the values are relative to. The option that is used is the ”Taken at the center of mass and aligned with the output coordinate system”. Here the output coordinate system is the coordinate system at the start of the link. Figure 4.32 shows each of the principal axes of inertia. The results are presented in equation 4.40 to 4.42. In the inertia matrices the larges elements are at the diagonal. That is reasonable since all of the elements is almost a cylinder.

$$I_1 = \begin{bmatrix}1077815.9 & 1939.4 & 24207.0 \\ 1939.4 & 1010823.7 & -13131.7 \\ 24207.0 & -13131.7 & 225407.6\end{bmatrix} \tag{4.40}$$

$$I_2 = \begin{bmatrix}688775026.5 & -2153417.8 & -83961.2 \\ -2153417.8 & 52950060.9 & -48598551.6 \\ -83961.2 & -48598551.6 & 703024081.4\end{bmatrix} \tag{4.41}$$

$$I_3 = \begin{bmatrix}249444.6 & -0.0 & 0.0 \\ -0.0 & 305986.7 & -0.1 \\ 0.0 & -0.1 & 388391.5\end{bmatrix} \tag{4.42}$$

Then all of the unknown elements needed to perform the forward and backward recursion of the Newton-Euler formulation described in section 3.2 are retrieved A Maple script developed by Marius Nordheim Røv is modified and used. The script is attached, see appendix A.2.

Chapter 5

Path Planning

Path planning is the description of how the states of the robot should evolve. In many scenarios not only the position at the start and end point is of interest, but the position of the robot along a predefined trajectory in task space. Most of todays industrial robot manipulators are controlled by reference tracking control. This means that a asymptotically stabilization is used to follow a predefined time dependent motion. The disadvantage of such controllers is the fact that they are dependent on time. In the case of making the robot follow an path as accurate as possible, the time is not of interests.

To improve absolute path accuracy and robustness to uncertainties in the system a new controller is tested, namely the orbital stabilization. Orbital stability means convergence of a solution of a closed-loop system to an orbit of a nominal motion. This means that the controller, instead of tracking a reference point, tries to converge to an orbit of the nominal motion.

In this chapter a controller that achieves asymptotic orbital stabilization of motions and a novel tool for analysis and re-design of systems dynamics using an excessive set of easy-to compute transverse coordinates will be used. This experiment is based on the paper [9], where a more detailed description of the calculations and proofs are shown. The scope of this chapter will be implementation and modification of the methods described in [9].

5.1 Mathematical definitions

In the paper [9] a tool for analysis and re-design of system's dynamics is introduced. This tool uses an excessive set of easy-to-compute transverse coordinates. The details of the development of the tool is shown in [9], only the main steps and the configurations for this particular case are illuminated in this thesis.

Earlier the dynamics of the robot were presented as:

$$M(q)\ddot{q} + C(q,\dot{q}) + G(q) = \tau \tag{5.1}$$

5.1.1 Motion Description

First assumption

It is assumed that a smooth set of functions define the motion, the functions are shown below. Those equations is feasible and chosen as nominal to be performed by the system 5.1.

$$q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} q_1^*(t) \\ q_2^*(t) \\ q_3^*(t) \end{bmatrix} \tag{5.2}$$

Second assumption

The next assumption is that a generator of the nominal motion $h(\theta)$ is known. This means that there are:

- Smooth scalar functions $\phi_1(\cdot)$, $\phi_2(\cdot)$, $\phi_3(\cdot)$ and $h(\cdot)$.
- A smooth scalar function of time $\theta^*(t)$ that satisfies the differential relation

$$\frac{d}{dt}\theta^*(t) = h(\theta^*(t)) \tag{5.3}$$

All this together makes it possible to make the angle of each joint only dependent on the variable $\theta^*(t)$.

$$\begin{bmatrix} q_1^*(t) \\ q_2^*(t) \\ q_3^*(t) \end{bmatrix} = \begin{bmatrix} \phi_1(\theta^*(t)) \\ \phi_2(\theta^*(t)) \\ \phi_3(\theta^*(t)) \end{bmatrix} \tag{5.4}$$

5.1.2 Controller Description

Above it is defined nominal functions that describe the motion of the robot, both in task space $\theta^*(t)$ and in joint space $q^*(t)$. Knowing the nominal behavior of the robot it is possible to determine the nominal control torque $\tau^*(t)$ which is needed in order to follow the nominal trajectory, i.e.

$$\tau = U(q,\dot{q}) \tag{5.5}$$

where

$$U(q,\dot{q})\Bigg|_{\substack{q = q^*(t) \\ \dot{q} = \dot{q}^*(t)}} = \tau^*(t) \tag{5.6}$$

The challenge is to find such an operator $U(\cdot)$ that satisfies the relation in 5.6 for the nominal motion 5.2. Therefore, in order to achieve orbital stabilization, the following controller is suggested in [9]:

$$\tau = \tilde{U}(q,\dot{q}) = U(q,\dot{q}) + K_y(q,\theta)y + K_w(q,\theta,\dot{q})w + K_{\dot{y}}(q,\theta,\dot{q})\dot{y} \tag{5.7}$$

for the 3 DOF system the $y, q, \dot{y}$ are defined as follows

$$\begin{array}{ccc} y_1 = q_1 - \phi_1(\theta), & y_2 = q_2 - \phi_2(\theta), & y_3 = q_3 - \phi_3(\theta) \\ \frac{d}{dt}y_1 = \dot{q}_1 - \phi_1'(\theta)\dot{\theta}, & \frac{d}{dt}y_2 = \dot{q}_2 - \phi_2'(\theta)\dot{\theta}, & \frac{d}{dt}y_3 = \dot{q}_3 - \phi_3'(\theta)\dot{\theta} \\ w_1 = \dot{q}_1 - \phi_1'(\theta)h(\theta), & w_2 = \dot{q}_2 - \phi_2'(\theta)h(\theta), & w_3 = \dot{q}_3 - \phi_3'(\theta)h(\theta) \end{array} \tag{5.8}$$

Where θ is defined as a result of a smooth projection of configuration vector q onto the orbit of the nominal motion:

$$\theta = P(q) \tag{5.9}$$

This equation must be twice continuous differentiable and recovering the time evolution of the variable $\theta^*(t)$

In this case $P(q)$ is chosen to be the forward kinematics for the θ variable, as defined earlier. So when moving in the x-direction $P(q)$ is the x component of the forward kinematics for the 3 DOF model. See appendix A.2 for a maple script that calculates this variable.

5.1.3 Transverse Linearization

The goal is to find a way to analytically compute a linear time-periodic system, with dimensions greater than (2n-1). Showing asymptotic stability of this linear time-periodic system, as argued in [9], implies orbital stability of the close-loop system. From Theorem 2 in [9], the following is stated:

"Consider the dynamics of the closed-loop system 5.1, 5.7 i a tubing vicinity of the nominal motion 5.2. There exists a time-periodic matrix $A(t)$, such that the dynamics of the variations of the variables y and w introduced in 5.8 along the solution $q = q^*(t)$ of the closed-loop system 5.1, 5.6, 5.7 can be computed directly by solving the linear equation:"

$$\frac{d}{dt}\begin{bmatrix}\delta y \\ \delta w\end{bmatrix} = A(t)\begin{bmatrix}\delta y \\ \delta w\end{bmatrix} \tag{5.10}$$

Theorem 3 in [9] gives the stability statement of the chosen controller:

"Consider the dynamics of the close-loop system 5.1, 5.7 in a tubing vicinity of the nominal motion 5.2, if the linear system with periodic coefficients 5.10 is asymptotically stable, then the nominal periodic motion 5.2 of the closed-loop system 5.1, 5.7 is asymptotically orbital stable."

5.2 Inverse Dynamics Control

The classic PID controller is based on an always existing error to the given reference trajectory. This requires a very fast and precise controller. From the robot's dynamics we know the nominal behavior of the robot, from the friction model developed earlier the static friction that occurs is also known. This information gives the nominal behavior of the robot, given a desired trajectory. Then the controller only have to compensate for the tracking error from the nominal motion of the robot. This prevents the controller to be unnecessary requested to correct the error, as the case is for the classical PID controller. Therefore, in this case, the error used for feedback will be redefined to be the distance to the orbit.

In order to achieve a better performance one of the available controllers that achieve asymptotic stability is redesigned to ensure orbital stability. As the title suggests, the inverse dynamics controller is chosen. A PD+ controller is another good alternative, but for this project only one controller is studied. As discussed above, to ensure asymptotic orbital stabilization for the motion, convergence to zero of a full set of transverse coordinates is required. A more detailed presentation of the controller redesign is shown in [9], this report only focuses on the main aspects.

The Inverse Dynamic Controller has the form:

$$\tau = M(q)\left[a(\theta) - K_p y - K_d w\right] + C(q,\dot{q})\dot{q} + G(q) \tag{5.11}$$

where θ is defined by 5.9 and y and w is defined by 5.8. The gains K_p and K_d are constant diagonal $(3x3)$ matrices. The vector a is:

$$a = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \phi_1''(\theta)h(\theta)^2 + \phi_1'(\theta)h'(\theta)h(\theta) \\ \phi_2''(\theta)h(\theta)^2 + \phi_2'(\theta)h'(\theta)h(\theta) \\ \phi_3''(\theta)h(\theta)^2 + \phi_3'(\theta)h'(\theta)h(\theta) \end{bmatrix} \tag{5.12}$$

Rewriting the controller in 5.11 to show that it is in the family of the controller in 5.7

$$\tau = \underbrace{\left[M(q)a(\theta) + C(q,\dot{q})\dot{q} + G(q)\right]}_{U(q,\dot{q})} + \underbrace{\left[-M(q)K_p\right]}_{K_y} y + \underbrace{\left[-M(q)K_d\right]}_{K_w} w \tag{5.13}$$

Above it is argued that the sufficient conditions for orbital stability for the closed loop system in 5.1 and 5.11 requires analysis of an auxiliary linear system 5.43 to be met. Below is a suggestion for the matrix $A(t)$ for the dynamics 3.54 and 5.11:

$$A(t) = D_1 + D_2(t) \tag{5.14}$$

Where D_1 only depend on the controller gains:

$$D_1 = \begin{bmatrix} \mathbb{O}_{3x3} & \mathbb{O}_{3x3} \\ -K_p & -K_d \end{bmatrix} \tag{5.15}$$

$D2(t)$ is defined as a solution of the linear equation below. Since the determinant of $\mathcal{T}(t)$ could go to zero this could not be solved analytically, therefore a linear solver is used.

$$D_2(t)\mathcal{T}(t) = C(t) \tag{5.16}$$

$$\mathcal{T}(t) = \begin{bmatrix} \mathbb{I}_{3x3} - \phi'(\theta)\frac{\partial}{\partial q}P(q) & \mathbb{O}_{3x3} \\ -\mathcal{F}_1(\theta)\frac{\partial}{\partial q}P(q) & \mathbb{I}_{3x3} \end{bmatrix} \tag{5.17}$$

$$C(t) = \begin{bmatrix} -\mathcal{F}_1(\theta)\frac{\partial}{\partial q}P(q) - \phi'(\theta)\mathcal{F}_2(\theta, q, \dot{q}) & \mathbb{I}_{3x3} - \phi'(\theta)\frac{\partial}{\partial q}P(q) \end{bmatrix} \tag{5.18}$$

where

$$\phi'(\theta) = \left[\frac{d}{d\theta}\phi_1(\theta), \frac{d}{d\theta}\phi_2(\theta), \frac{d}{d\theta}\phi_3(\theta)\right]^T \tag{5.19}$$

$$\phi''(\theta) = \left[\frac{d^2}{d\theta^2}\phi_1(\theta), \frac{d^2}{d\theta^2}\phi_2(\theta), \frac{d^2}{d\theta^2}\phi_3(\theta)\right]^T \tag{5.20}$$

$$\frac{\partial}{\partial q}P(q) = \left[\frac{\partial}{\partial q_1}P(q), \frac{\partial}{\partial q_2}P(q), \frac{\partial}{\partial q_3}P(q)\right] \tag{5.21}$$

$$\mathcal{F}_1(\theta) = \phi''(\theta)h(\theta) + \phi'(\theta)h'(\theta) \tag{5.22}$$

$$\mathcal{F}_2(\theta, q, \dot{q}) = \dot{q}^T \frac{\partial^2 P(q)}{d\partial} - h'(\theta)\frac{\partial P(q)}{\partial q} \tag{5.23}$$

$$\frac{\partial^2}{\partial q^2}P(q) = \begin{bmatrix} \frac{\partial^2 P(q)}{\partial q_1^2} & \frac{\partial^2 P(q)}{\partial q_1 \partial q_2} & \frac{\partial^2 P(q)}{\partial q_1 \partial q_3} \\ \frac{\partial^2 P(q)}{\partial q_2 \partial q_1} & \frac{\partial^2 P(q)}{\partial q_2^2} & \frac{\partial^2 P(q)}{\partial q_2 \partial q_3} \\ \frac{\partial^2 P(q)}{\partial q_3 \partial q_1} & \frac{\partial^2 P(q)}{\partial q_3 \partial q_2} & \frac{\partial^2 P(q)}{\partial q_3^2} \end{bmatrix} \tag{5.24}$$

5.3 Motion Generator

The path for this experiment is a square. A square will give four sharp corners where we can investigate the performance of the robot. To make such a path a parameter in task space has to be chosen. For this experiment the velocity will work as a motion generator. Since time is not of interest this motion generator will only depend on the position of the robot. The challenge of this experiment is to find an optimal motion generator, through investigation of orbital stabilization of the motion.

The motion generator must fulfill the second assumption made in section 5.1.

First the motion is defined as a function of the position of the robot:

$$\dot{\theta} = h(\theta) = \sum_{i=1}^{n} \alpha_i \theta^i \tag{5.25}$$

Where n is the chosen order of the polynomial. For this case a polynomial of order 7 will be tested. θ is defined as the position in task space. For this experiment it will be the position of the robot's end effector in a given direction relative to a start position. For the case with a square with sides of 10 centimeters in x-y plane, theta will be either x or y, depending on which side of the square the robot is at.

The motion generator must ensure that the robot starts and stops. To ensure that the robot starts the acceleration of θ has to be greater than zero at $\theta = 0$, i.e.

$$\ddot{\theta} = h'(\theta)\dot{\theta} = h'(\theta)h(\theta) \tag{5.26}$$

To ensure the above

$$h(0) > 0 \tag{5.27}$$

$$h'(0) > 0 \tag{5.28}$$

The next condition of the motion generator is related to the stop of the motion. To ensure that the robot stops at $\theta = 0$ the velocity and acceleration must be zero. From equation 5.26 it is shown that if $\dot{\theta}$ is equal zero, the acceleration $\ddot{\theta}$, by definition, also is zero. Then to stop the motion the following condition should hold:

$$h(0.1) = 0 \tag{5.29}$$

Path Constraints

Since the motion is defined as a velocity, constraints are used to make the robot manipulator follow the desired path. This is called path-constrained trajectory planning. Defining θ as the direction of the motion, the constraints are the inverse kinematics for the 3 DOF model. This principle is defined for motion in x-direction below. For the line motion in y-direction the variable dependent on θ changes from x to y.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \theta + x_0 \\ y_0 \\ z_0 \end{bmatrix} \tag{5.30}$$

Where x_c and z_c are parameters to be chosen. For this experiment we have

$$x_0 = 0.45 \tag{5.31}$$

$$y_0 = 0 \tag{5.32}$$

$$z_0 = 0.13 \tag{5.33}$$

The next parameters are calculated from trigonometry from figure 4.25. The parameters a_1, a_2, a_3 and d_1 are the DH parameter of the 3 DOF model.

$$q1 = \arctan\left(\frac{y}{x}\right) \tag{5.34}$$

$$h = \sqrt{x^2 + y^2} - a1 \tag{5.35}$$

$$l = \sqrt{(d_1 - z)^2 + h^2} \tag{5.36}$$

$$\theta_m = \arccos\left(\frac{a_3^2 + a_2^2 - l^2}{2a_2a_3}\right) \tag{5.37}$$

$$\theta_f = \arctan\left(\frac{d_1 - z}{h}\right) \tag{5.38}$$

$$\theta_b = \arcsin\left(\frac{a_3 \sin(\theta_m)}{l}\right) - \theta_f \tag{5.39}$$

$$q2 = \frac{\pi}{2} - \theta_b \tag{5.40}$$

$$q3 = \frac{\pi}{2} - \theta_m \tag{5.41}$$

Redefining the above as a function ϕ only dependent on the variable $\theta(t)$, in this case $\theta(t) = x(t) + x_c$:

$$q_1(t) = \phi_1(\theta(t)), \quad q_2(t) = \phi_2(\theta(t)), \quad q_3(t) = \phi_3(\theta(t)) \tag{5.42}$$

For the velocity calculations Maple is used. Differentiating on the correct variable, representing θ. See appendix A.2 for information about the attached Maple script.

5.4 Optimization

In order to find an optimal motion generator a optimization problem is stated. The optimization problem uses the transverse coordinates defined above to find one of the best motion generators to follow the nominal trajectory.

The problem to be optimized is:

Point	δx_0	δy_0	δz_0
1	0.01	0.0125	0.12
2	-0.1	0.012	-0.0125
3	0.1	-0.002	0.01
4	0.1	0.07	0.01
5	-0.07	0.001	-0.01
6	0.012	0.07	0.01
7	-0.012	0.07	-0.01
8	-0.1	-0.002	-0.01
9	0.1	0.07	-0.01
10	0.1	-0.01	0.07

Table 5.1: Initial Perturbation

$$\frac{d}{dt}\begin{bmatrix}\delta y\\ \delta w\end{bmatrix} = A(t)\begin{bmatrix}\delta y\\ \delta w\end{bmatrix} \tag{5.43}$$

The initial condition of δy and δw is taken from the path accuracy of the robot. From the data sheet [4] the path repeatability accuracy is said to be $0.13mm$. For the optimization problem ten points in the room, with total error $\sqrt{x^2+y^2+z^2} < 0.13$, are chosen.

From the values in table 5.1 the corresponding error in joint space is calculated using inverse kinematics. This is used as the initial condition for numerical solving of equation 5.43. The model is implemented in Simulink and solved with ODE45 solver.

The parameter to be optimized is the α parameters of the motion generator 5.25. This means that the optimization problem to be solved consists of 7 unknown parameters.

In addition to the optimization problem and the initial condition there are several constraints, both necessary constraints, and constraints that could be included to probably increase performance.

Nonlinear Constraints

The first constraint is related to the physical limitations of the robot. From table 2.2 the limits of each of the joint velocities are set.

Since the function to optimize is the velocity in task space, some calculations are used to ensure that the velocities in joint space are lower than the limits. A nonlinear constraint is made as a for loop, the function is shown below. In the function below the velocities from table 2.2 are used. The other elements of the function are defined above.

```
function [c, ceq] = constraint(alpha)
for theta = 0:0.1
  h(theta) = sum from i = 0 to 7 (alpha(i)*theta^i);
  dphi = derivative of phi;
  q_velocity(theta) = [abs(dhpi*h) - [2.6180 2.7925 2.9671]']
end
c = q_velocity
c = [c; -h(11:end-10) + 0.0025];
c = [c; -h(end-9:end)
```

Since the motion only should go in one direction, the $h(\theta)$ function should be positive for all θ. Therefore $-h(\theta)$ is added to the nonlinear inequality constraint. In addition a constant with size 0.0025 is added for θ greater than 0.01 and less than 0.09. A full stop will result in backlash in gears, which is not taken into account in this report. Therefore it is assumed that a full stop between two corners will result in large error.

Linear Inequality Constraint

The condition in equation 5.28 is realized as a inequality constraint. To get the motion started $h(0)$ has to be greater than zero, but not too large either. The following limits are used:

$$1e - 8 < \alpha(0) < 1e - 5 \tag{5.44}$$

Setting up the constraints on matrix form:

$$Ax = b \tag{5.45}$$

where x is α

$$A = \begin{bmatrix} 0000000-1 \\ 00000001 \end{bmatrix} \tag{5.46}$$

$$b = \begin{bmatrix} -1e-8 \\ 1e-5 \end{bmatrix} \tag{5.47}$$

Linear Equality Constraint

To stop the motion the constraint in equation 5.29 must be fulfilled. This is realized as an equality constraint.

$$A_{eq}x = b_{eq} \tag{5.48}$$

Where

$$A_{eq} = \begin{bmatrix} \theta_{max}^7 \theta_{max}^6 \theta_{max}^5 \theta_{max}^4 \theta_{max}^3 \theta_{max}^2 \theta_{max} 1 \end{bmatrix} \tag{5.49}$$

$$b_{eq} = 0 \tag{5.50}$$

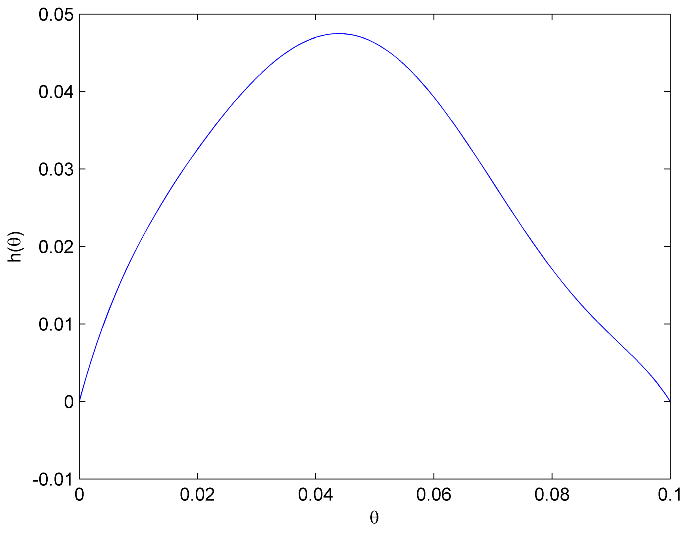


Figure 5.1: Positive X motion generator

Implementation

The problem is implemented in Matlab and Simulink. The function ”fmincon” is used, which is a Matlab function that finds the minimum of a constrained nonlinear multi variable function. The default settings of fmincon are used, which means that the optimization algorithm that is used is the interior point algorithm. This algorithm is not a global minimizer, but it will give a local minimum of the objective function. In appendix A.1 it is described a folder named ”PathPlanning” that is attached. This folder contains all the files for the optimization problem.

The optimization algorithm is used to find motion generator for each of the sides of the square. This results in four different velocity profiles. The generalized coordinate for each side is the coordinate representing the direction of the motion, i.e. x and y.

5.4.1 Optimization Results

Using the optimization algorithm on the problem stated above resulted in many different results. Depending on the start point the result of the optimization problem gives different results. In other words the problem consists of many local minimum.

Below is the result of the optimization problem shown for each of the sides of the square:

In figure 5.1 to 5.4 the results of the optimization are shown. Since the results is optimized for the controller on the given configuration in joint space it is reasonable that they are a little different. In magnitude the maximum velocity is almost equal; for the first side,

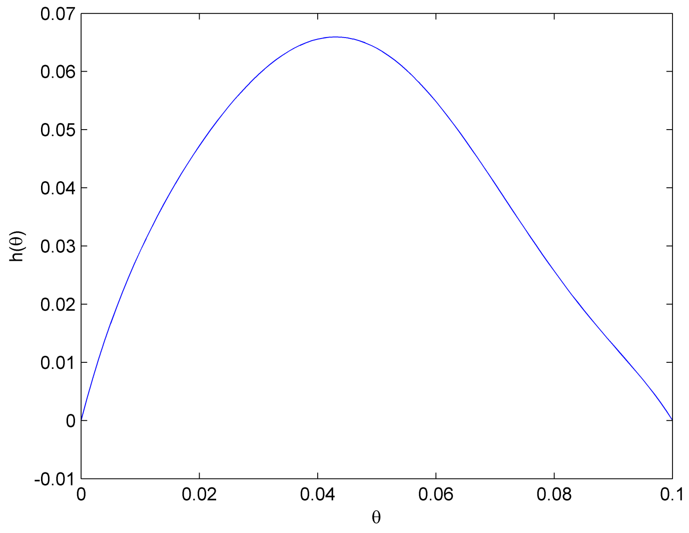


Figure 5.2: Positive Y motion generator

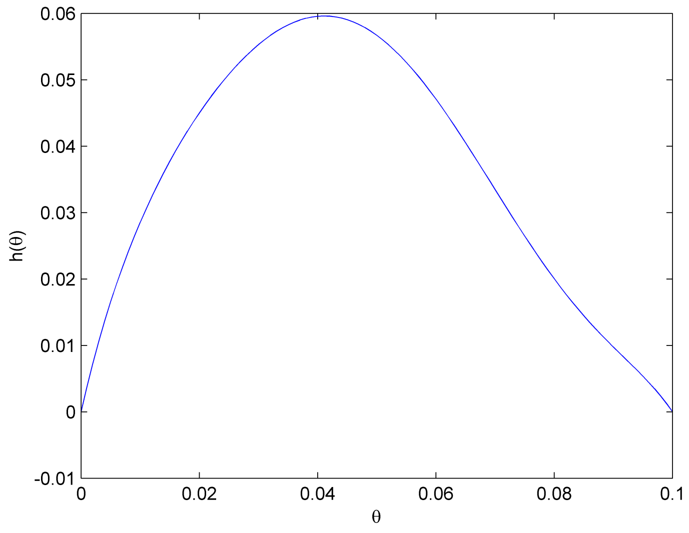


Figure 5.3: Negative X motion generator

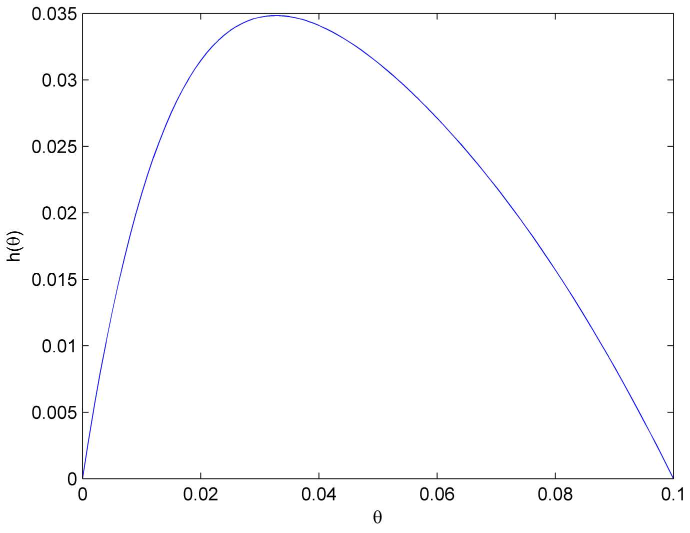


Figure 5.4: Negative Y motion generator

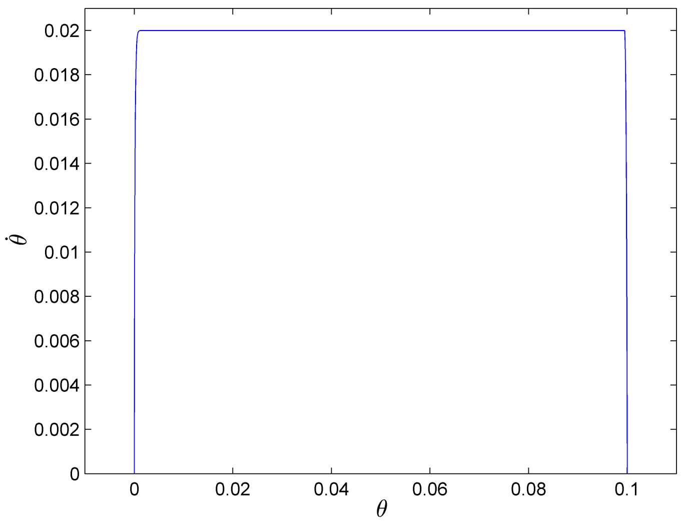


Figure 5.5: Position - Velocity plot for all sides by standard ABB Controller

figure 5.1, it is approximately $0.047 m/s$, the second side, figure 5.2, it is approximately $0.065 m/s$, for the third side, figure 5.3, it is approximately $0.059 m/s$ and for the last side, figure 5.4, it is approximately $0.035 m/s$. These velocities does not say anything about the joint velocity at the given configuration, so it is reasonable that they differ.

Opposed to the standard ABB controller, shown in figure 5.5, this method takes the dynamics of the robot into account and finds a optimal profile for each part of the motion. It is interesting to see that the velocity profiles found are quite different from what ABB uses on the same path. The profile from the ABB path planner uses almost a step and keeps the velocity constant. Testing will show which gives the best performance at the robot's end effector.

As mentioned earlier the optimization algorithm only finds local minimum. In order to achieve better results the algorithm could be used with a number of different initial points, with the goal of finding the best of all the local minimums. This could probably give other results, that could be tested on the robot to tell if it really gives an improvement of the end effector performance. For now, this result, together with the controller designed in the next chapter, is used to evaluate the performance of the robot at sharp corners.

Chapter 6

Controller Design

In chapter 5 optimal paths are found to ensure orbital stabilization of the Inverse Dynamics controller. In this chapter the realization of the Inverse Dynamics controller is in focus. This controller is based on a Simulink controller developed by Stepan Pchelkin in his work with the paper [8]. The controller used by Stepan was designed for the robot ABB IRB 140, and for circle motions. Therefore a lot of changes are made on the controller to make it perform a square, and to make it work on another robot.

6.1 Redesign of the External Control Axis Controller

From section 5.1.2 the following controller is suggested:

$$\tau = M(q)\left[a(\theta) - K_p y - K_d w\right] + C(q,\dot{q})\dot{q} + G(q) \tag{6.1}$$

From before the axis controller of ABB is used in experiments, this controller is shown in figure 2.2. There is two main methods of realize the new Inverse Dynamics control on the, already existing, axis controller. Either to send the equation 6.1 on the feed forward port and set all gains of the axis controller to zero, or to modify the gains of the axis controller on the fly to fit the new controller design in 6.1.

It is assumed that the easiest way of implementing the controller is to use the feed forward port on the axis controller and set all axis controller gains to zero. This is tested, and the results are strange. A difference between the control signal sent on the feed forward port, and the real control signal sent to the robot occurred. After discussion with one of the inventors of the external control, Anders Robertsson at Lund University, it is clear that an integrator gain always will occur on the axis controller even if the integrator gains are set to zero. This describes the difference between the signal generated by the Inverse Dynamics control and the actual signal sent to the robot. Therefore it was decided to design the axis controller to fit the new Inverse Dynamics controller. The ”hidden” integrator term will still affect, but the error seen by the integrator will be less using the axis controller design as it is.

The redesign of the axis controller is as follows. The control law of the axis controller is:

$$\begin{aligned} \tau = K_{d,ac}\left(K_{p,ac}(posRef - q_{meas}) + (velRef - \dot{q}_{meas})\right) + \dots \\ \dots + K_{i,ac}\int \left(K_p(posRef - q_{meas}) + (velRef - \dot{q}_{meas})\right)dt + trqFfw \end{aligned} \tag{6.2}$$

where:

- $K_{d,ac}$ is the velocity gain
- $K_{p,ac}$ is the position gain
- posRef is the joint position reference
- velRef is the joint velocity reference
- trqFfw is the controller feed forward

The redesign of the axis controller follows from the Inverse Dynamics controller design:

$$\begin{aligned} \tau &= M(q)\left[a(\theta) - K_p y - K_d w\right] + C(q,\dot{q})\dot{q} + G(q) \\ \tau &= M(q)\left[a(\theta) - K_p\left(q - \phi(\theta)\right) - K_d\left(\dot{q} - \phi'(\theta)h(\theta)\right)\right] + C(q,\dot{q})\dot{q} + G(q) \\ \tau &= M(q)K_p\left(\phi(\theta) - q\right) + M(q)K_d\left(\phi'(\theta)h(\theta) - \dot{q}\right) + M(q)a(\theta) + C(q,\dot{q})\dot{q} + G(q) \\ \tau &= \underbrace{M(q)K_d}_{K_{d,ac}}\left[\underbrace{K_d^{-1}K_p}_{K_{p,ac}}\left(\underbrace{\phi(\theta)}_{posRef} - q\right) + \left(\underbrace{\phi'(\theta)h(\theta)}_{velRef} - \dot{q}\right)\right] + \dots \\ &\dots + \underbrace{M(q)a(\theta) + C(q,\dot{q})\dot{q} + G(q)}_{trqFfw} \end{aligned} \tag{6.3}$$

Since this project only involves the three first joints the modifications used here are only done for the three first joints. The other three use the standard axis controller.

6.2 Simulink Implementation

The implementation of the controller is shown in figure 6.1, where all the signals from External Control are available. A closer description of each signal is available in [13] and will not be described in detail here. The controller is attached, see appendix A.1. The implementation consists of a block setting up the basic setting. The next block converting all signals from the measured motor side to the corresponding arm side signal, this block is label ”Motor to Arm”. The block below is the main block where all the control is implemented, this block is labeled ”Control”. The last element is the modification of the integral gain, where the gains for the three first joints are set to zero.

The Simulink diagram shown in figure 6.2 shows the structure of the controller. Below is a short description of each block:

Time

When the signal f_switch is set, the clock starts and the motion begins.

CalculateTheta

This block calculates the θ from the measured position using the forward kinematics. Each time $\theta = 0.1$ the variable currentSide is incremented. This holds control of which side of the square the motion is at such that the correct motion generator is used.

Integrate_dTheta

This block integrates the θ variable three times to compensate for the delay in the system, using the motion generator $h(\theta)$ that is found in the previous chapter. In the paper [8] the delay was observed from experiments to be approximately three samples.

DesiredTrajectory

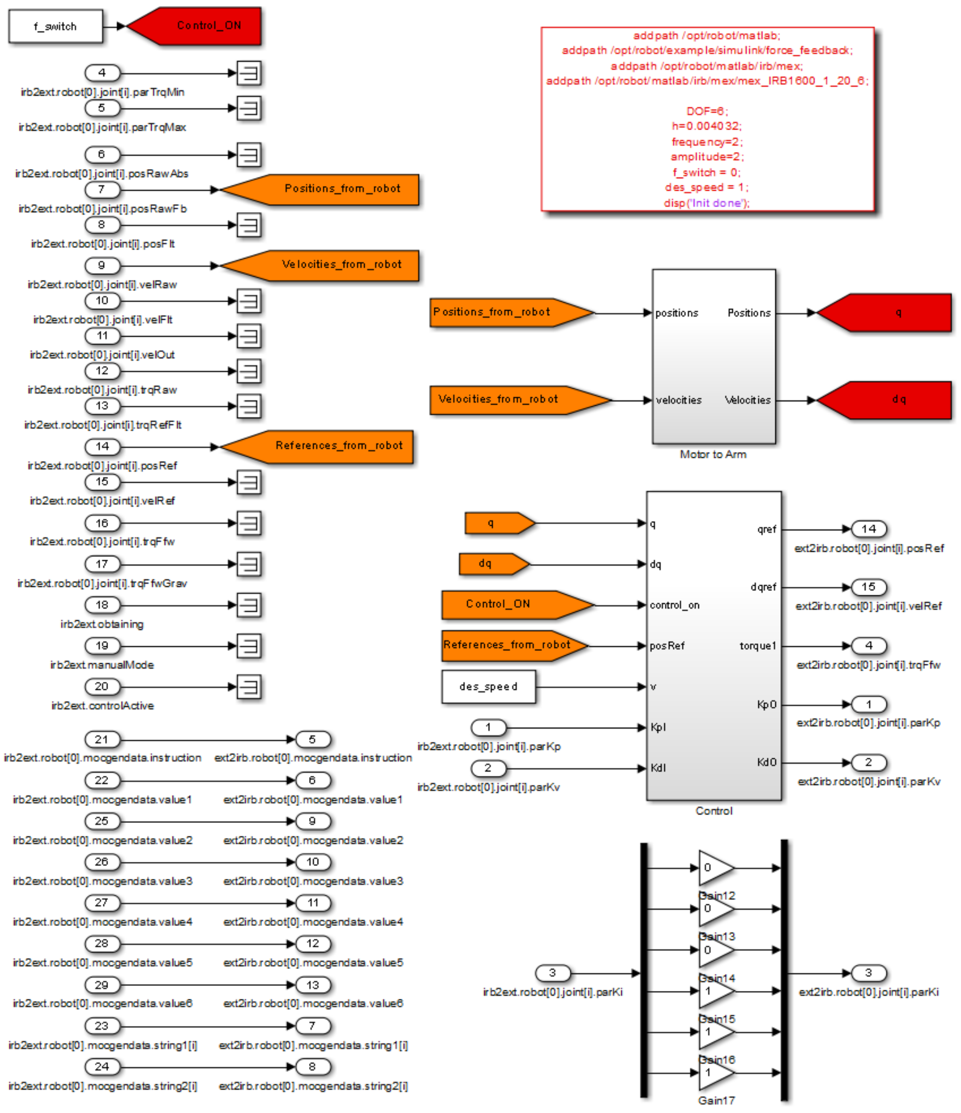
From the velocity profiles found in chapter 5, all variables $\theta, q, \dot{q}, \ddot{q}$ are calculated with a sampling rate equal the sampling rate of the robot. These variables are placed in this block as look up tables to improve the execution time of the script.

For the first half second of execution time this block uses fourth order path planning to plan a path from the current position to the start position of the motion, i.e. the first elements of the look up tables. Since the motion starts with a small velocity this ensures a smooth start, and it ensures that the robot is at the start position of the motion.

After the first half second the first element greater than the current θ is found in the look up table. Then interpolation is used between the two closest elements in the look up table to find the next position, velocity and acceleration of each joint.

CompensationDelay

Compensation on the measurement signal for the delay in the system.



```
addpath /opt/robot/matlab;
addpath /opt/robot/example/simulink/force_feedback;
addpath /opt/robot/matlab/irb/mex;
addpath /opt/robot/matlab/irb/mex/mex_IRB1600_1_20_6;

DOF=6;
h=0.00432;
frequency=2;
amplitude=2;
f_switch = 0;
des_speed = 1;
disp('Init done');
```

f_switch
Control_ON
irb2ext.robot[0].joint[i].parTrqMin
irb2ext.robot[0].joint[i].parTrqMax
irb2ext.robot[0].joint[i].posRawAbs
irb2ext.robot[0].joint[i].posRawFb
Positions_from_robot
irb2ext.robot[0].joint[i].posFlt
irb2ext.robot[0].joint[i].velRaw
Velocities_from_robot
irb2ext.robot[0].joint[i].velFlt
irb2ext.robot[0].joint[i].velOut
irb2ext.robot[0].joint[i].trqRaw
irb2ext.robot[0].joint[i].trqRefFlt
irb2ext.robot[0].joint[i].posRef
References_from_robot
irb2ext.robot[0].joint[i].velRef
irb2ext.robot[0].joint[i].trqFfw
irb2ext.robot[0].joint[i].trqFfwGrav
irb2ext.obtaining
irb2ext.manualMode
irb2ext.controlActive
irb2ext.robot[0].mocgendata.instruction
ext2irb.robot[0].mocgendata.instruction
irb2ext.robot[0].mocgendata.value1
ext2irb.robot[0].mocgendata.value1
irb2ext.robot[0].mocgendata.value2
ext2irb.robot[0].mocgendata.value2
irb2ext.robot[0].mocgendata.value3
ext2irb.robot[0].mocgendata.value3
irb2ext.robot[0].mocgendata.value4
ext2irb.robot[0].mocgendata.value4
irb2ext.robot[0].mocgendata.value5
ext2irb.robot[0].mocgendata.value5
irb2ext.robot[0].mocgendata.value6
ext2irb.robot[0].mocgendata.value6
irb2ext.robot[0].mocgendata.string1[i]
ext2irb.robot[0].mocgendata.string1[i]
irb2ext.robot[0].mocgendata.string2[i]
ext2irb.robot[0].mocgendata.string2[i]
positions
Positions
velocities
Velocities
q
dq
Motor to Arm
control_on
posRef
v
KpI
KdI
qref
dqref
torque1
Kp0
Kd0
des_speed
Control
ext2irb.robot[0].joint[i].posRef
ext2irb.robot[0].joint[i].velRef
ext2irb.robot[0].joint[i].trqFfw
ext2irb.robot[0].joint[i].parKp
ext2irb.robot[0].joint[i].parKv
irb2ext.robot[0].joint[i].parKp
irb2ext.robot[0].joint[i].parKv
irb2ext.robot[0].joint[i].parKi
ext2irb.robot[0].joint[i].parKi
Gain12
Gain13
Gain14
Gain15
Gain16
Gain17

Figure 6.1: Inverse Dynamics Control implemented in Simulink

Control

Implementation of the feed forward part of the controller. In this block the part of the equation 6.3 labeled ”trqFfw” is implemented. In addition the friction compensation developed in section 4.1 is implemented in this block.

Kp

The proportional gain of the controller is implemented in this block. From the equation 6.3 this is labeled ”$K_{p,ac}$”.

KdM

The derivative gain of the controller is implemeted in this block. From the equation 6.3 this is labeled ”$K_{d,ac}$”.

Motor2Arm / Arm2Motor

Conversion between motor and arm side.

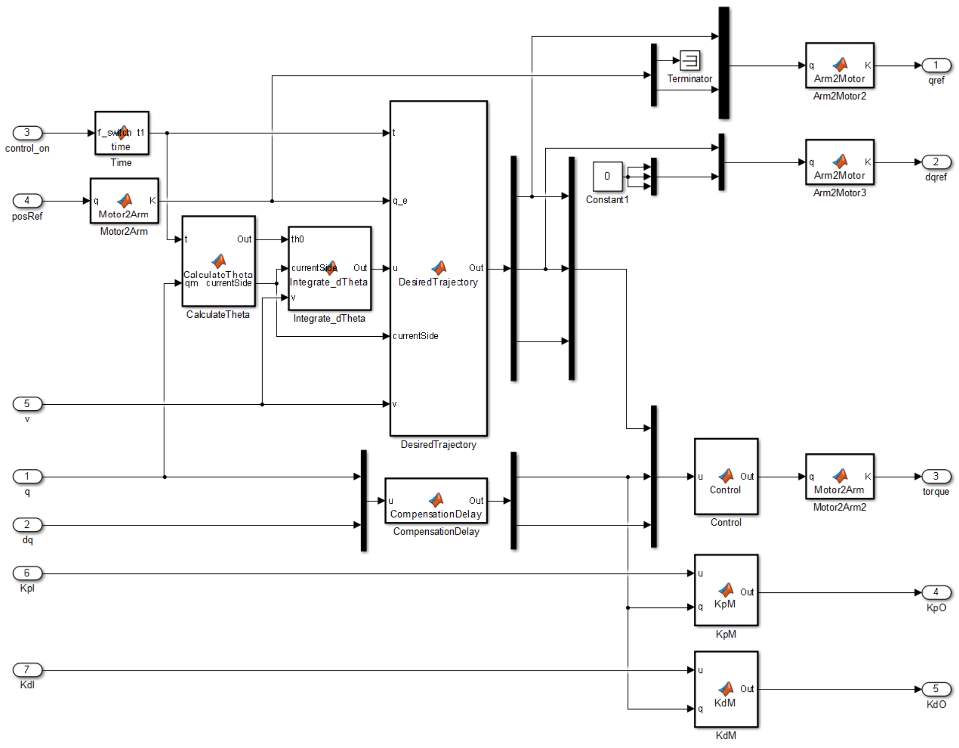


Figure 6.2: Control block in figure 6.1

Chapter 7

Main Results

This chapter presents the main results of this project. The performance of the robot's end effector with the motion generator developed in chapter 5 and the controller developed in chapter 6, is compared with the performance of the state-of-the-art ABB robot controller. To evaluate the performance, the Nikon camera equipment described in section 2.4 is used together with the performance measured by the encoders on the robot.

7.1 Experiment Setup

7.1.1 Inverse Dynamics Controller

A square with sides of 10 cm should be performed by the robot. The starting point is at $x = 0.45m$, $y = 0m$ and $z = 0.13m$. The 5. joint of the robot is flipped up to make the tracking tool shown in section 2.2 visible for the camera. The controller gains i set to:

$$K_d^{-1}K_p = \begin{bmatrix} 125 & 0 & 0 \\ 0 & 150 & 0 \\ 0 & 0 & 100 \end{bmatrix} \tag{7.1}$$

$$K_d = \begin{bmatrix} 0.2400 & 0 & 0 \\ 0 & 0.1350 & 0 \\ 0 & 0 & 1.4700 \end{bmatrix} \tag{7.2}$$

Different controller gains are tested with different results, but the best end effector performance is achieved with these gains. The experiment is repeated to verify the results.

7.1.2 ABB Controller

A path performing the same path as the Inverse Dynamics controller is implemented in Robot Studio. Since the end effector is flipped up, the path described in robot studio departs from the numbers above. That is because Robot Studio gives the position of the end effector of the 6 DOF model, while Inverse Dynamics control only counts for the 3 DOF model.The Robot Studio file is attached, see appendix A.3.

7.2 Experiment Results

The performance of the robot is measured by both encoders and the camera. Since the encoders is placed on the motor side of the robot, this measurement is not very reliable in this scope of precision. Therefore the camera measurement is considered as the most validated measurement. The error is measured as the three dimensional distance to the orbit:

$$error = \sqrt{(x - x_{ref})^2 + (y - y_{ref})^2 + (z - z_{ref})^2} \tag{7.3}$$

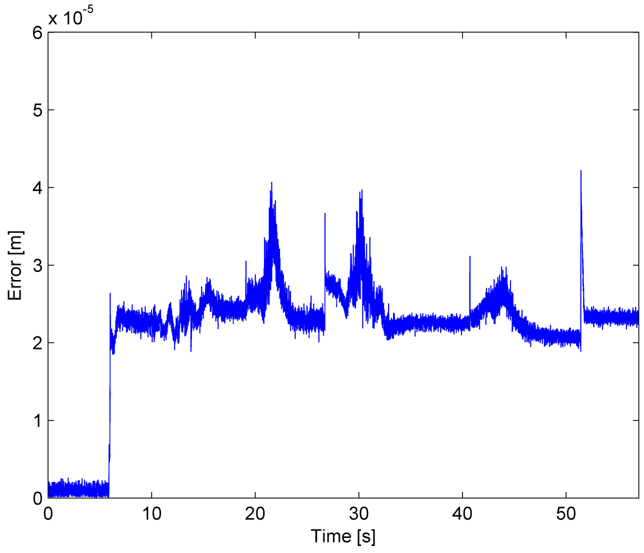


Figure 7.1: Inverse Dynamics Control Encoder Measurement

The results are shown in figure 7.1 to 7.4. From figure 7.1 the performance of the motion and controller designed in this project measured by the encoders is approximately $4.23 \cdot 10^{-5}m$. The comparable performance of the state-of-the-art ABB controller is shown in figure 7.3. The maximum error is measured to $5.64 \cdot 10^{-5}m$. This shows an

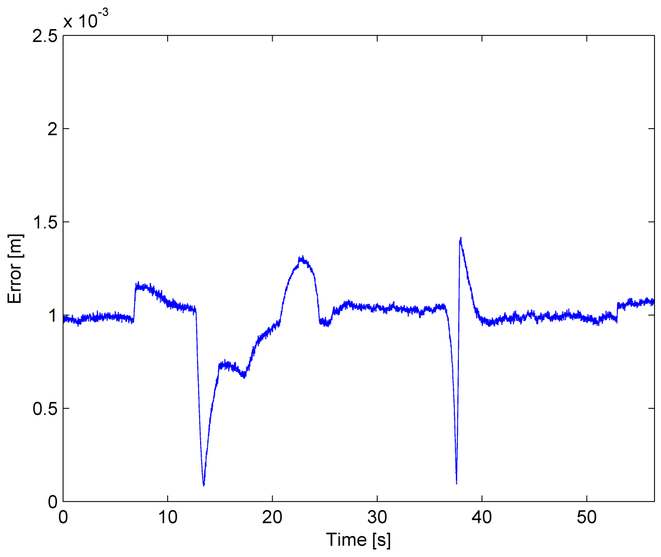


Figure 7.2: Inverse Dynamics Control Camera Measurement

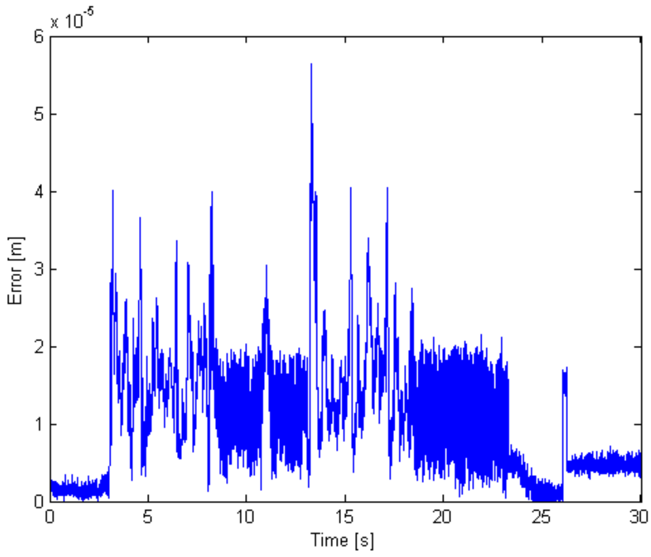


Figure 7.3: ABB Control Encoder Measurement

improvement on the encoder measurement of 25 percent, with the new controller design and the optimized motion.

In figure 7.2 and 7.4 the performance of the Inverse Dynamics control and the ABB con-

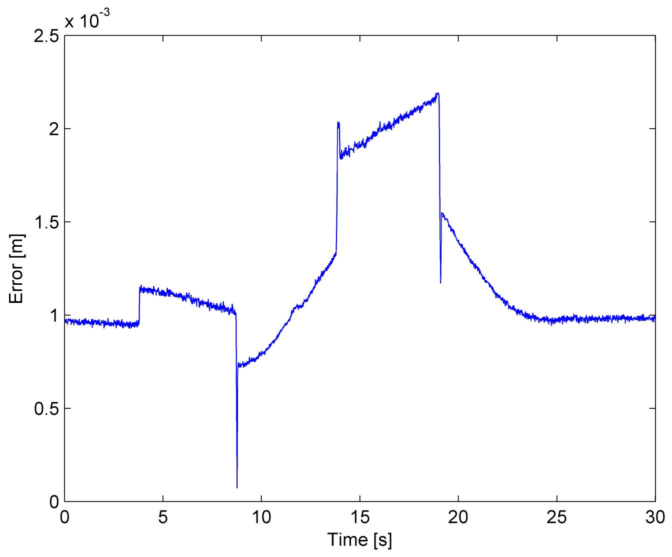


Figure 7.4: ABB Control Camera Measurement

troller, measured by the camera, is shown. The maximum error for the result of this report shows a maximum error of $1.4 \cdot 10^{-3} m$. The ABB controller performs $2.2 \cdot 10^{-3} m$. This shows that the new design gives an improvement of the robots end effector performance at a path with sharp corner of 36 percent, measured by external measurement device.

7.3 Discussion

From figure 7.2 and 7.4 it seems like a constant error of approximately one millimeter occurs. This could either be a correct picture of the performance, or it could be related to error in for example the DH parameters of the robot. To detect this, the DH parameters could be found from experiments on the actual robot. It could also be related to errors in the process of defining frames. Since this process, as described in section 2.4, is closely related to the DH parameters both these possible reasons could be investigated. It is important to notice that, even if this constant error is related to DH parameter the performance of the robot is better with the new path and controller design. The distance from the one millimeter line to the maximum outlier is less for the new design.

The increased performance with the new controller design is 25 percent measured by encoders. Measured by the camera the increase in performance is 36 percent. Since the controller only takes the encoder measurements into account it is reasonable to assume that there is still room for improvement in tuning of controller. From figure 7.2 it is shown that a constant error occurs on the encoder measurement. The error is very small but it is reasonable to assume that tuning improvement and improvement of the dynamic model of the robot and the friction model could improve the result further.

Chapter 8

Conclusion and Further Work

8.1 Conclusion

This report has shown that an improvement of the performance of the ABB IRB 1600 of 36 percent is possible using the Inverse Dynamic controller and a sub optimal motion design. This result is achieved using an external measurement device, namely the Nikon K610 camera measurement system.

In this report a friction model for the robot is developed in order to use it in the modified controller. This friction model is a function of velocity. From the current velocity the function calculates the torque needed to overcome friction. Validation of this model showed that compared to a standard PD controller an improvement from 30 to 45 percent occurred. This improvement is only visible on the encoder measurement. The camera measurement showed no significantly improvement. Two different friction models are developed, one with velocities from encoder measurements, and one with velocity measurements from camera. There is no significant difference between those two models, so in the rest of the report the encoder model is used. Another remark is that a good PID controller gives almost the same performance as a PID controller with friction compensation.

Path constrained trajectory planning is used to remove the dependency of time from the path planning. A motion generator that gives the velocity along the path using generalized coordinates, is found using optimization. The objective function in the optimization problem is defined using linearized transverse dynamics of the robot, and evaluating the error along the motion using 10 different initial perturbations. The task for the robot is chosen to be a square with sides of 10 cm. A sub optimal motion generator is found for each side of the square. The project shows that it is possible to find sub optimal motion generators, which giv good results on the performance in real experiments. The fact that only one sub optimal motion generator is found, opens for further experiments trying to find a solution closer to the global optimum of the optimization problem.

For control a Inverse Dynamics controller is modified for orbital stabilization of the motion. The controller is implemented, tuned and verified. The system with sub optimal motion generator and the Inverse Dynamics controller shows even better performance than the state-of-the-art ABB controller. All though the achieved performance is good, there is probably room for further improvement of the controller.

8.2 Further Work

For further work it is recommended to take a closer look at the optimization problem. In this project only one sub optimal solution to each side of the motion is tested. Since the optimization algorithm only finds sub optimal solutions it could be interesting to run the algorithm for a number of initial points in order to achieve a result closer to the global optimum.

The optimization problem could also be extended with more constraints, such that the task for the controller is easier. For example constraints on the joint acceleration could be implemented.

From the experiment it seems like there is still potential for improvements on the controller gain tuning. It is difficult to find good tuning gains for such controllers, and in this project it is based on trial and error.

The dynamics of the robot is very different for different positions in the room. In addition to the velocity profile optimization, it could be optimized on the start position of the motion.

From the friction validation experiment it is hard to see any improvement on the camera side. It is argued that this could be related to the DH parameters. It would be interesting to use the camera equipment to detect the DH parameters experimentally, and compare them with what the producer provides.

The dynamical equation of the robot is found using the CAD model of the robot. To get a better model of the robot, and further improve the performance, this equation could be found through experiments.

Bibliography

[1] ISO Standard 8373:1994. Manipulating industrial robots - vocabulary.

[2] ABB. More than 30 years with abb robotics. `http://www.abb.com/product/ap/seitp327/583a073bb0bb1922c12570c1004d3e6b.aspx`. [Online, accessed 03-June-2015].

[3] ABB. Irb 1600 cad models. `http://new.abb.com/products/robotics/industrial-robots/irb-1600/irb-1600-cad`, 2015. [Online, accessed 24-May-2015].

[4] ABB. Irb 1600 the highest performance 10 kg robot. `http://www05.abb.com/global/scot/scot241.nsf/veritydisplay/3b0491a94bd700a248257c71004ef393/$file/PR10282EN_R8.pdf`, 2015. [Online, accessed 24-May-2015].

[5] Isolde Dressler. *Force control interface for ABB S4/IRC5*, 2009.

[6] Nikon Metrology NV. *Focus 10 Overview and training notes*, 2013.

[7] Nikon Metrology NV. *Introduction to frames*, 2013.

[8] S.S. Pchelkin, A. Shiriaev, A. Robertsson, and L. B. Freidovich. *Analysing and Comparison of PD+ and Inverse Dynamics Controls for Orbital Stabilisation a Full-Actuated System*. PhD thesis, Norwegian University of Science and Technology, 2013.

[9] S.S. Pchelkin, A. Shiriaev, A. Robertsson, L. B. Freidovich, Paramonov L. V., and Gusev S. V. On orbital stabilization for industrial manipulators: case study in evaluating performances of modified pd+ and inverse dynamics controllers.

[10] M. N. Røev. Time optimal motion planning and motion control for industrial manipulators. Master's thesis, Norwegian University of Science and Technology, 2014.

[11] M.W. Spong, S. Hutchinson, and M. Vidyasagar. *Robot Modeling and Control.* John Wiley and Sons, Inc, 2006.

[12] E. L. Strandbraaten. Case studies in trajectory design for industrial robot manipulators. Master's thesis, Norwegian University of Science and Technology, 2014.

[13] D. Strke, A. Robertsson, and B Olofsson. *External Control - ExtCtrl*, 2011.

Appendix

A Attached Files

In addition to this paper there is attached a ZIP-file with digital attachments. In this folder files for symbolic calculation, optimization and control of robot is attached. Below is a description of each file in the directory.

A.1 MATLAB

Findo6.m

Symbolic calculation of the o6 matrix, i.e. the transformation matrix.

JaconbianSymbolic.m

Symbolic calculation of the Jacobian.

IndetificationConstantVelocity.mdl

Robot controller used for identification of the friction model. Running each joint separately with constant speed.

FrictionExperimentValidation.mdl

Robot controller to validate the performance of the friction model. This controller runs a sinus at each joint individually.

FrictionExperimentValidation_CircleMotion.mdl

Robot controller to validate the performance of the friction model. This controller runs all the first three joints simultaneously in a circle path.

PathPlanning

Files used to find a optimal path, and prepare vectors to the controller.

findInitError.m Converting 10 initial error positions to task space

optimization.m Main file for running optimization problem

errorfun.m Objective function of the optimization problem

errorModel.slx Simulink model used by errorfun.m to solve the transverse coordinates

constraint.m Constraint function of the optimization problem

makeThetaModel.slx Makes θ for the optimized motion generator with sampling 0.004032, equal the robot sampling rate

makeQFromTheta.m Calculates position, velocity and accelration for each joint, for the given motion generator

InvDynControl.mdl

The final controller, performing a square with sides of 10 cm, using Inverse Dynamics Control.

A.2 Maple

FindDynamicModel.mw

Calculates the dynamic model of the robot based on estimated input. Script is based on the work by Marius Nordheim Røv in [10]

FindP(q).mw

Calculates the function P(q)

FindPhi.mw

Calculates the functions $\phi(\theta)$, $\phi'(\theta)$ and $\phi''(\theta)$

circleHorisontal_dqCalc

Calculates dq for the horisontal circle used in friction validation experiment.

A.3 RobotStudio

DefinitionOfFrames.rsstn

Path for defining the base frame of the robot in Nikon system.

Square.rsstn

Path used to compare with the new controller developed in this thesis.

B Friction Validation Results

A high number of experiments is done in the validation process of the friction compensation model. Only portions of the results is presented in the report, and here are some extensions.

The figures below shows the error compute as the distance in joint angle. Due to sampling and delay the joint angle reference is 3 samples in front of the measured joint angle. Therefore the two signals is compared with this offset. For comparing the measured joint positions from encoders without any filtering is used. Three different methods is compared in the figures below. Joint number and maximum velocity of the sinus curve is stated for each figure.

Top	No feed forward term added
Middle	Feed forward compensation from encoder model added
Bottom	Feed forward compensation from camera model added

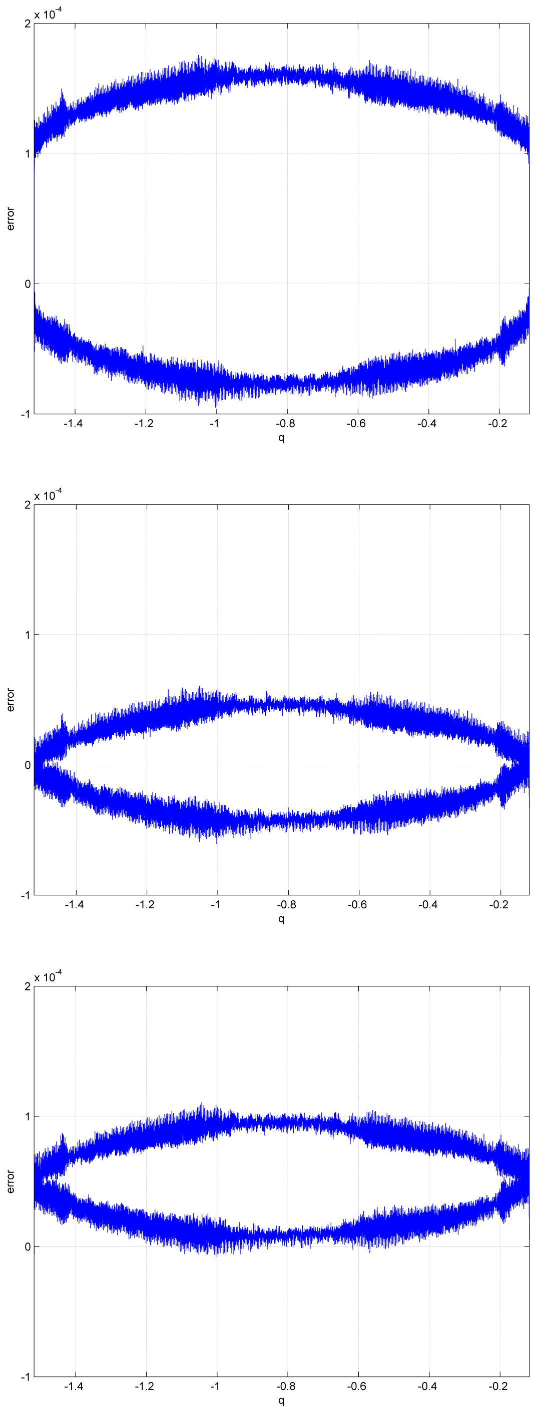
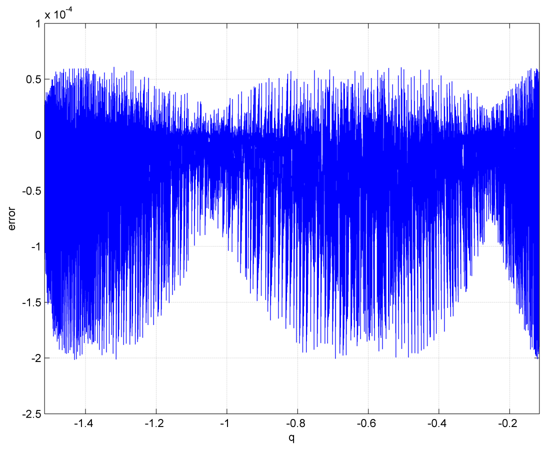
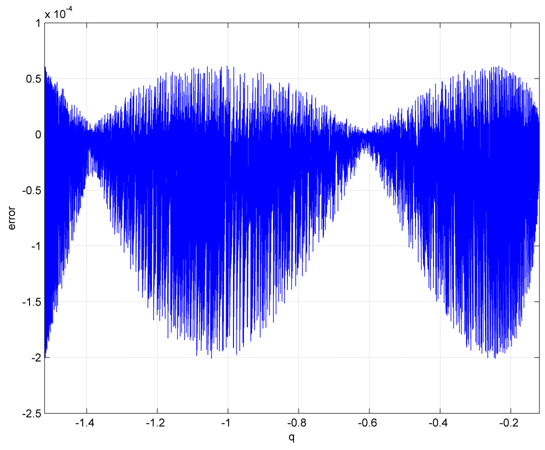


Figure 8.1: Encoder — Joint 1, velocity 0.1 rad/s — Top no FF, Middle Enc FF, Bottom Cam FF





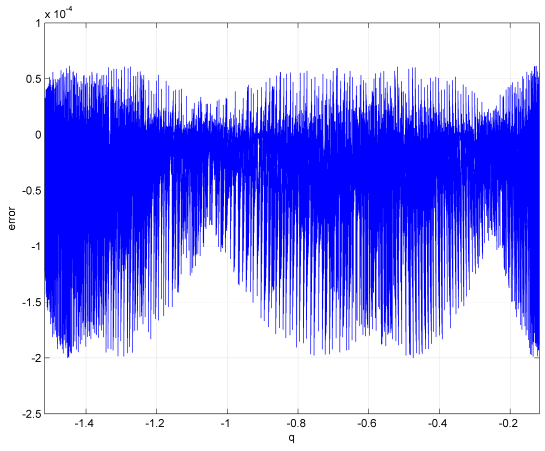


Figure 8.2: Camera — Joint 1, velocity 0.1 rad/s — Top no FF, Middle Enc FF, Bottom Cam FF

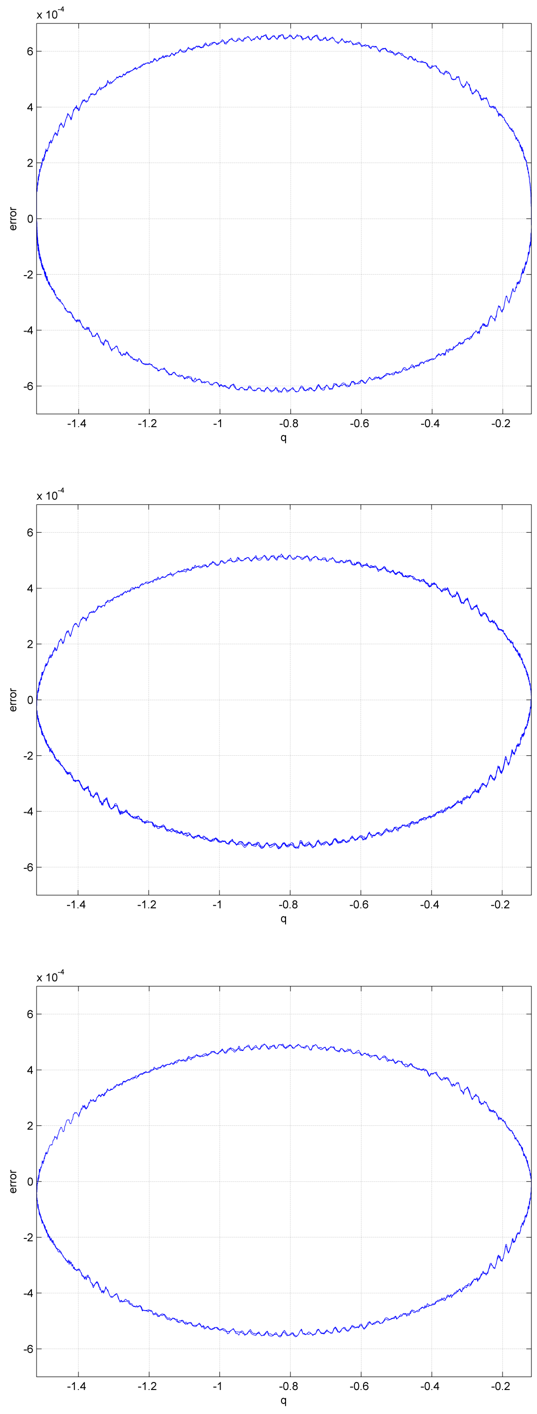
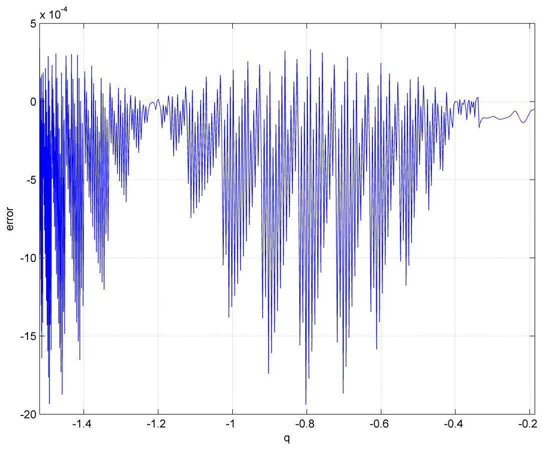
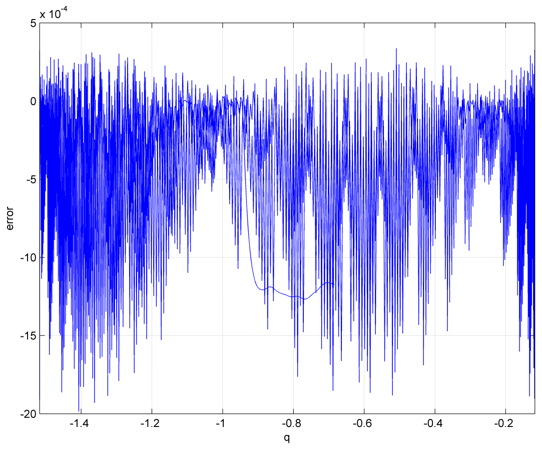


Figure 8.3: Encoder — Joint 1, velocity 1 rad/s — Top no FF, Middle Enc FF, Bottom Cam FF





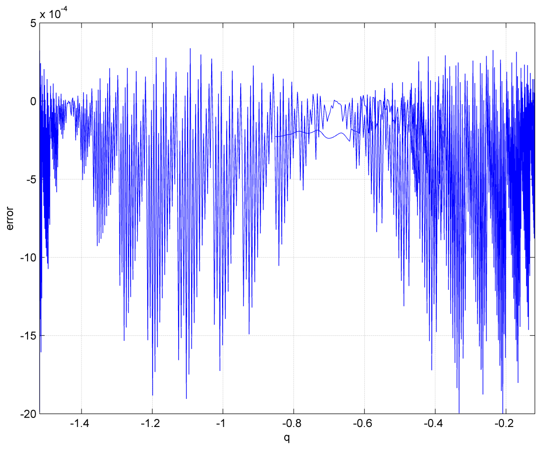


Figure 8.4: Camera — Joint 1, velocity 1 rad/s — Top no FF, Middle Enc FF, Bottom Cam FF

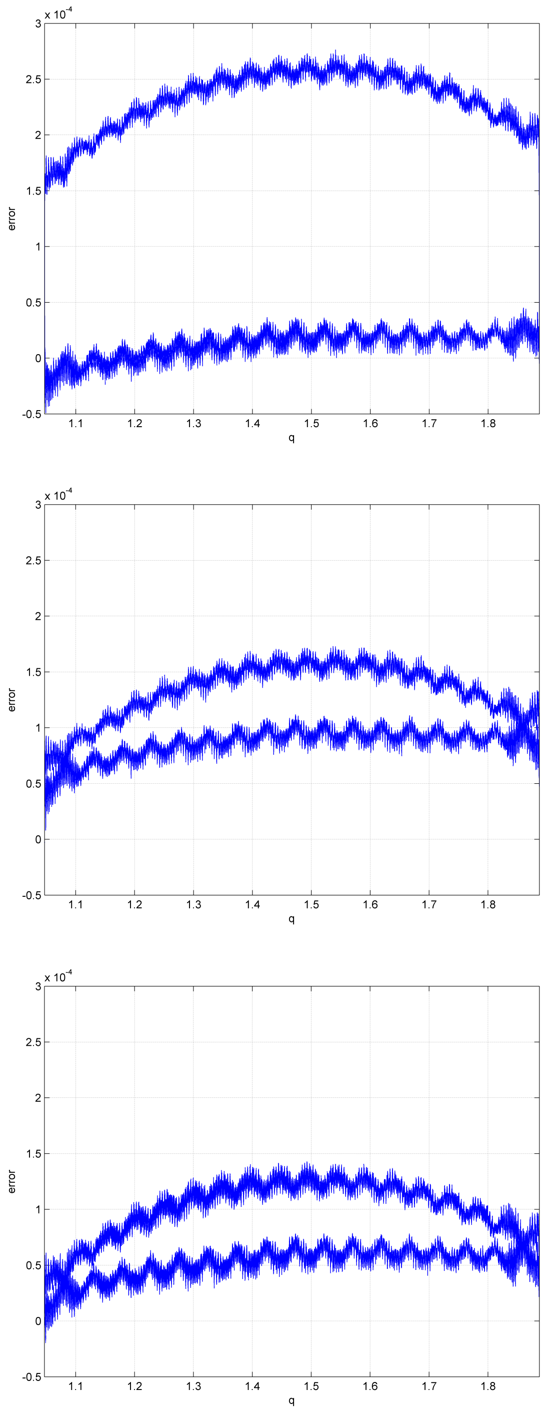


Figure 8.5: Encoder — Joint 2, velocity 0.1 rad/s — Top no FF, Middle Enc FF, Bottom Cam FF

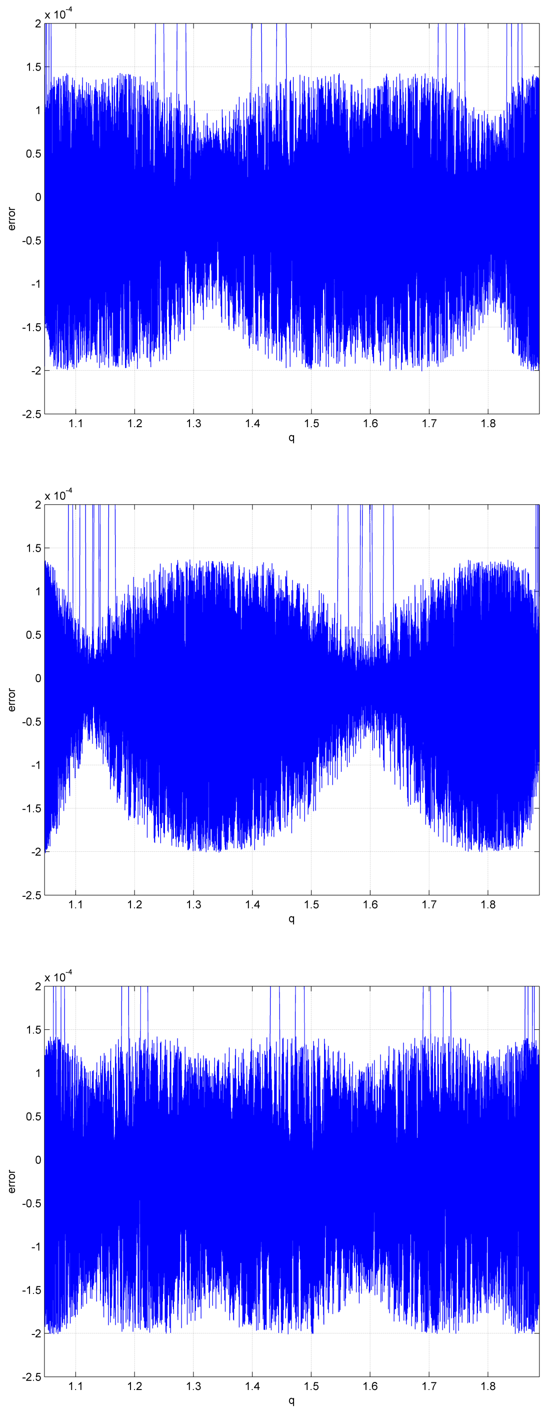
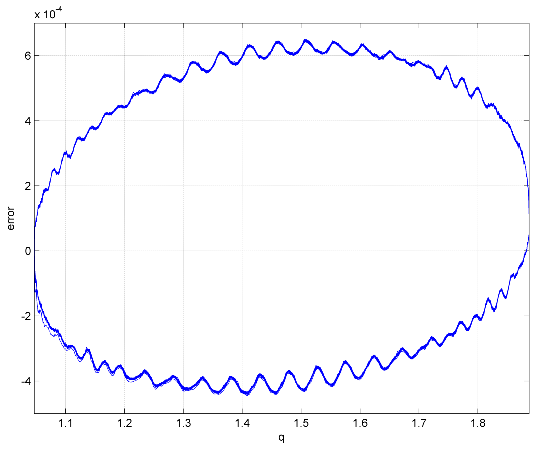
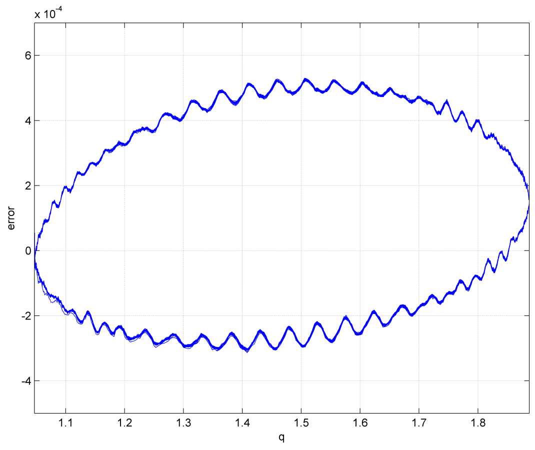


Figure 8.6: Camera — Joint 2, velocity 0.1 rad/s — Top no FF, Middle Enc FF, Bottom Cam FF





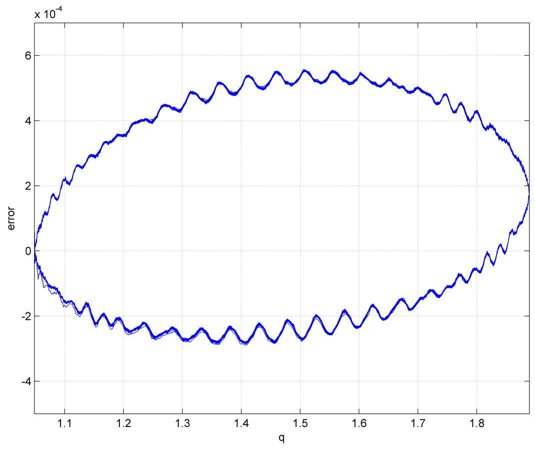
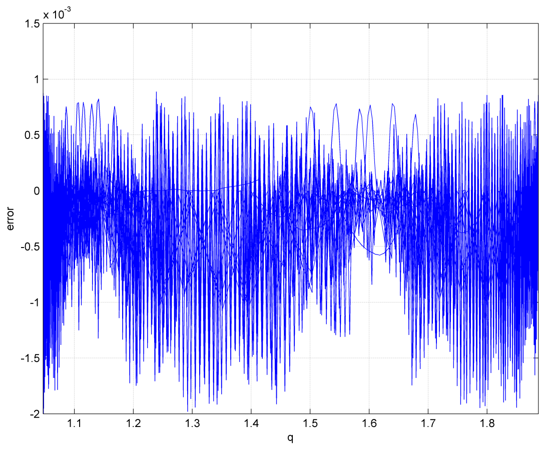
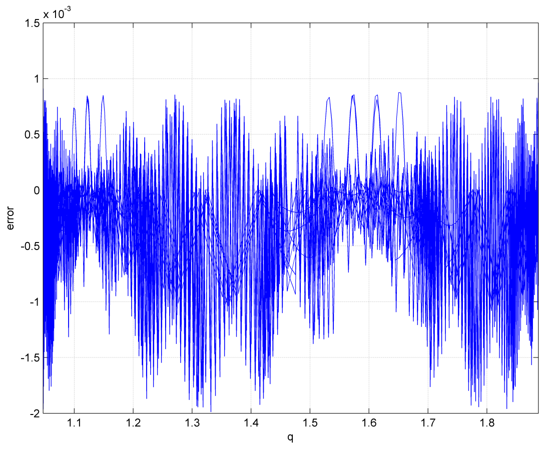


Figure 8.7: Encoder — Joint 2, velocity 1 rad/s — Top no FF, Middle Enc FF, Bottom Cam FF





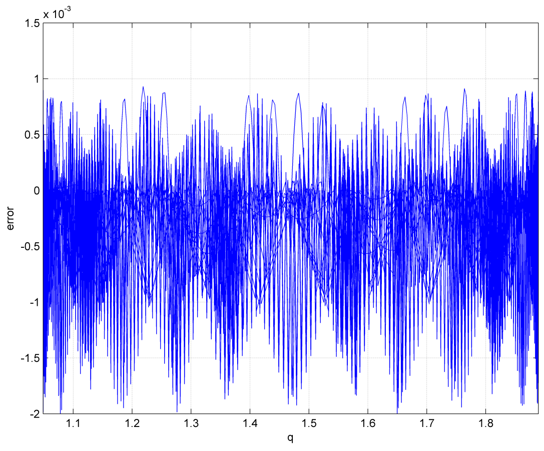


Figure 8.8: Camera — Joint 2, velocity 1 rad/s — Top no FF, Middle Enc FF, Bottom Cam FF

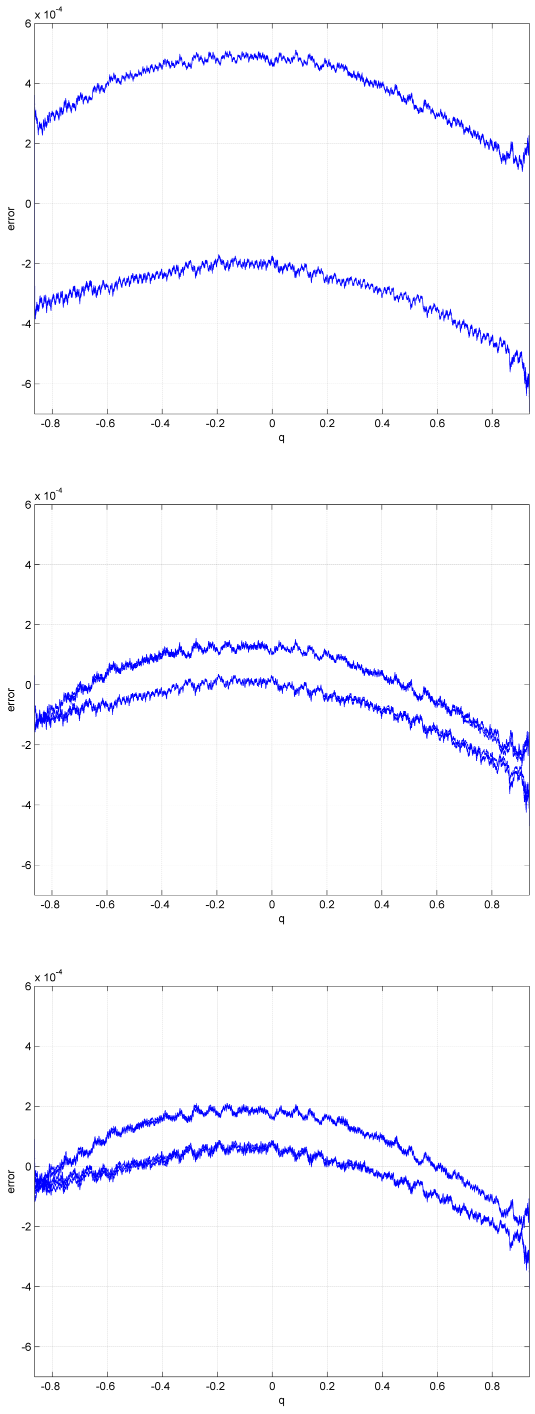


Figure 8.9: Encoder — Joint 3, velocity 0.1 rad/s — Top no FF, Middle Enc FF, Bottom Cam FF

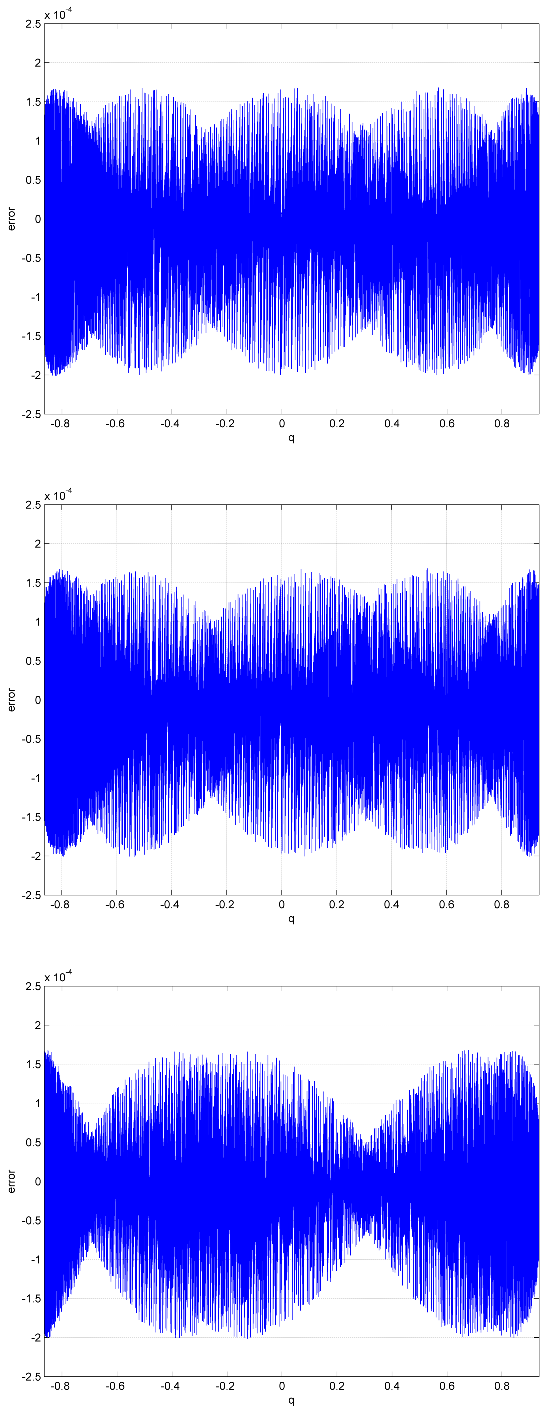


Figure 8.10: Camera — Joint 3, velocity 0.1 rad/s — Top no FF, Middle Enc FF, Bottom Cam FF

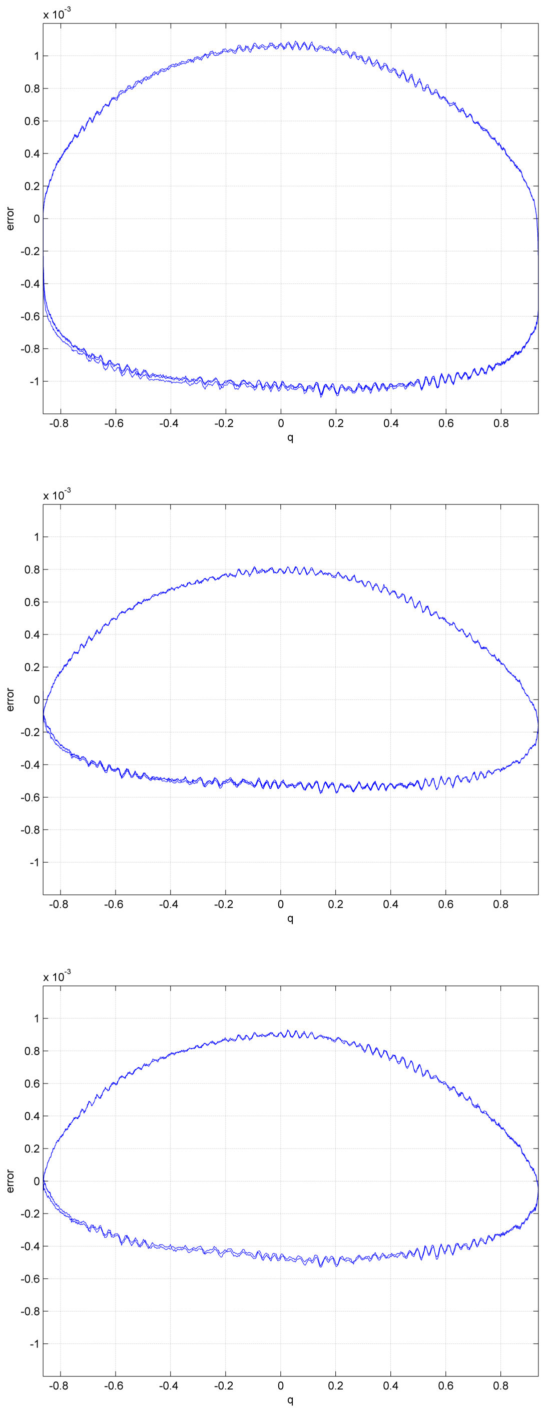


Figure 8.11: Encoder — Joint 3, velocity 1 rad/s — Top no FF, Middle Enc FF, Bottom Cam FF

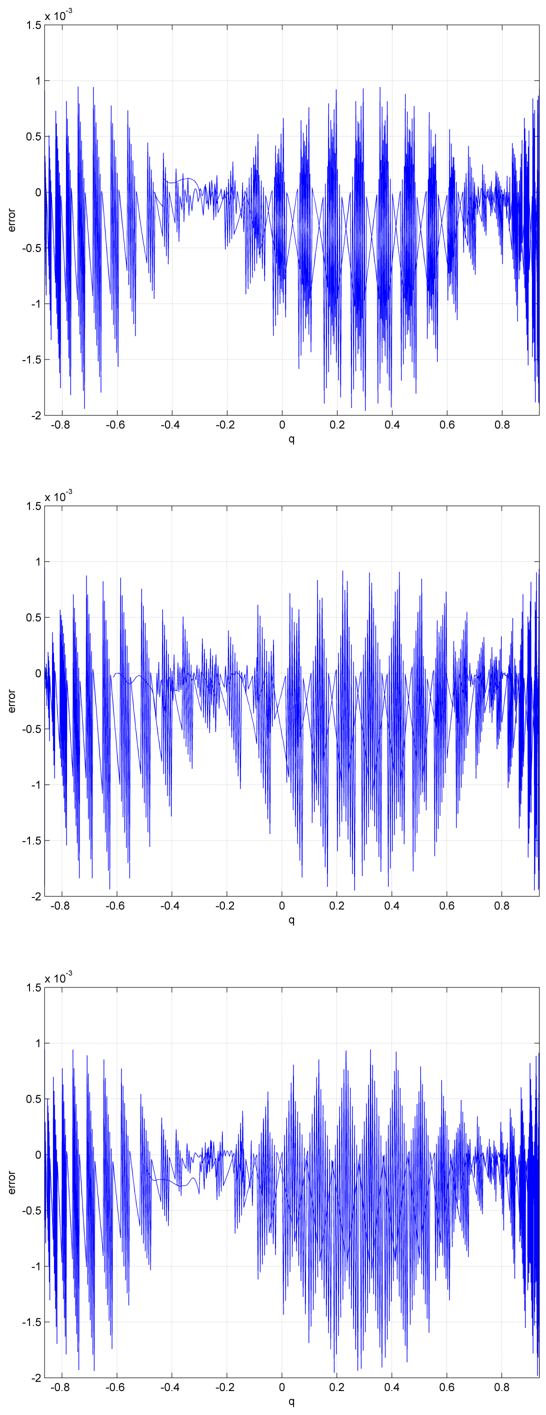


Figure 8.12: Camera — Joint 3, velocity 1 rad/s — Top no FF, Middle Enc FF, Bottom Cam FF