# Modeling and Control of ROV Manipulator

Morten Haugen

Marine Technology
Submission date: June 2012
Supervisor: Asgeir Johan Sørensen, IMT

Norwegian University of Science and Technology
Department of Marine Technology

# Problem Description

Remotely Operated Vehicles (ROVs) are common in deep water industries. As the oil industry, and also other relevant industries, moves into deeper water, the use of sub sea technology has increased. Due to safety and practical reasons it is not convenient to use manned diving for maintenance and surveys. The solution is to use ROVs that are unoccupied, highly maneuverable and operated by a person on board a vessel. The ROV is normally equipped with one or several manipulators so that it can perform simple tasks such as pulling cables, opening valves and handle different tools. It is important that the manipulators are easy maneuverable, highly accurate and has a quick response because this will both save money and also make it possible to perform more difficult tasks. In addition, the use of a good manipulator can prevent damage on the environment. The objective of this thesis is to investigate mathematical models and control methods for the manipulator on the ROV SF 30K, conduct simulations and tests, and also develop an interface for control of the manipulator and LabVIEW. Full scale tests of the derived control system will also be conducted to test and tune the controller.

**Scope of work**

- Review relevant literature on ROV and manipulators.
- Review documentation for the manipulator on the ROV SF 30K.
- Formulate a kinematic and dynamic model of the manipulator.
- Formulate inverse kinematics for manipulator control.
- Investigate control principals for position control.
- Formulate a proper control algorithm for the model.
- Conduct simulation for controller tuning and testing.
- Develop interface for control of manipulator and LabVIEW.
- Carry out full scale experiments to demonstrate the control system for the Raptor manipulator.
- Document each step in the process.

The report shall be written in English and edited as a research report including literature survey, description of mathematical models, description of control algorithms, simulation results, model test results, discussion and a conclusion including a proposal for further work. Source code should be provided in a digital folder with code listing enclosed in appendix. It is supposed that Department of Marine Technology, NTNU, can use the results freely in its research work, unless otherwise agreed upon, by referring to the student's work. The thesis should be submitted within June 10th.

| | |
|---|---|
| Supervisor: | Professor Asgeir Johan Sørensen (NTNU) |
| Advisors: | Mauro Candeloro (NTNU) |
| | Martin Ludvigsen (NTNU) |
| | Fredrik Dukan (NTNU) |

# Abstract

The main objective of this thesis is to investigate and present the most relevant techniques and topics within the field of robot modeling and control. The studies will then be used to develop a working control system for the 'Raptor' manipulator stationed on the ROV 'SubFighter 30K'.

Due to insufficient information, a simplified model is made to resemble the actual manipulator. This model forms the foundation of all subsequent actions, including the model based control design. The dynamic model is developed by the well known method of Euler-Lagrange. Since this is an energy based method, both the kinetic and the potential energy of the system must be calculated. Systematic procedures are given to clarify the process of these calculations.

In this thesis, a sliding-mode controller is derived and proposed as a suitable controller for the given manipulator. The control objective is to force the manipulator to track a time dependent, desired path in the joint space. However, since it is inconvenient for the operator to specify joint space trajectories, several inverse kinematics algorithms are suggested. Due to the kinematic structure of the manipulator, no closed-form solutions can be obtained. The focus is thus directed towards numerical Jacobian based methods.

A full-scale implementation requires a working interface between the developed control system and the manipulator system. For that reason, the main concepts of digital communication are presented. Although no communication data is logged from the Raptor, this presentation will pose an advantage if the work is continued.

When no control forces are applied to the dynamic model, the manipulator model is expected to behave like a multi joint, three dimensional pendulum. The simulations corresponds to this assumptions, thus the model is assumed to be correct and valid. Simulations of the complete system shows that the sliding-mode controller works as intended. Two chosen IK algorithms are then implemented and compared through simulations. The DLS method proves to be superior to the simple inverse Jacobian method.

Finally, the control system is implemented in LabVIEW and thus prepared for full-scale testing.

# Acknowledgements

It gives me great pleasure in acknowledging the support and help of my supervisor, Professor Asgeir J. Sørensen. His enthusiastic appearance and great knowledge has both inspired me and and given me insight to understand the advanced topics of this thesis. I would also like to thank Professor Roger Skjetne for his patience and for giving me knowledge of nonlinear control design. When it comes to the topic of robot modeling, I would like to honor the help of Professor Eilif Pedersen and Research Scientist Uwe Mettin. Their vast knowledge has guided me through the large topic of robotics and given me great wisdom.

I would also like to share the credit of my work with Ph.D. candidate Mauro Candeloro. This thesis would not have been possible without his help. Thanks also to Ph.D. candidate Christoffer Fredrik Lid Thorvalden, Ph.D. candidate Fredrik Dukan, and Post. Doc. Martin Ludvigsen for help and comments on my work, and my co-students for supporting me through discussions and comments.

Lastly, I would like to thank my loving and caring wife, Monika Sletteberg Haugen, whose faithful support during this semester is so appreciated.

Morten Sletteberg Haugen
Trondheim, June 10, 2010

# Contents

# Nomenclature

**Abbreviations**

| | |
|---|---|
| BFM | Bi-Phase Mark |
| CAD | Computer-Aided-Design |
| CFD | Computational Fluid Dynamics |
| CLF | Control Lyapunov Function |
| D-H | Denavit-Hartenberg |
| DLS | Damped Least-Squares |
| DOF | Degrees-of-Freedom |
| EMI | Electromagnetic Interference |
| FM-1 | Frequency Modulation 1 |
| IK | Inverse Kinematics |
| RFFM | Raptor Force Feedback Manipulator |
| RMRC | Resolved Motion Rate Control |
| ROV | Remotely Operated Vehicle |
| RSD | Remote Servo Driver |
| SF | SUB-fighter |
| SVD | Singular Value Decomposition |
| VIs | Virtual Instruments |
| VSS | Variable System Structure |

## Symbols

| | |
|---|---|
| $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ | Centrifugal and Coriolis matrix |
| $\boldsymbol{D}(\boldsymbol{q})$ | Inertia matrix |
| $\boldsymbol{\delta}_0$ | Vector of unmodeled dynamics |
| $\boldsymbol{\delta}_\tau$ | Vector of additive disturbance terms |
| $\boldsymbol{F}(\dot{\boldsymbol{q}})$ | Friction vector |
| $\boldsymbol{G}(\boldsymbol{q})$ | Buoyancy and gravity vector |
| $\boldsymbol{H}$ | Homogeneous transformation matrix |
| $\boldsymbol{I}_b$ | Body fixed inertia tensor |
| $\boldsymbol{J}$ | Jacobian matrix |
| $\mathcal{K}$ | Kinetic energy |
| $\mathcal{L}$ | Lagrangian |
| $\boldsymbol{\omega}$ | Angular velocity |
| $\mathcal{P}$ | Potential energy |
| $\boldsymbol{\psi}$ | Alternative signum function |
| $\boldsymbol{q}$ | Joint angles |
| $\boldsymbol{R}$ | Rotational matrix |
| $\boldsymbol{S}$ | Skew symmetric matrix |
| $\boldsymbol{T}$ | Transformation matrix |
| $\boldsymbol{\tau}$ | Torque vector |
| $\boldsymbol{\Theta}_e$ | Minimal representation of the orientation |
| V | Lyapunov function |
| $\boldsymbol{v}$ | Linear velocity |
| $\|p\|$ | The absolute value of a scalar p |
| $\|\boldsymbol{x}\|$ | The Euclidean norm of a vector x |
| $\|\boldsymbol{x}\|_p$ | The p-norm of a vector x |
| $\xi$ | High-pass filter |
| y | System output |
| $y_d$ | Desired system output |
| z | Sliding surface |

# Chapter 1

# Introduction

The field of robotics has evolved rapidly over the past twenty years. Advances in computer and sensor technology has made it possible to create more accurate and faster robots for a great number of applications. While the term robot often has been widely used, this thesis will treat the operator controlled manipulator. This is a mechanical arm that can perform humanoid tasks in cases where the use of humans is considered either unpractical or uneconomical. In the marine industry, the computer controlled manipulator is often used in connection with an ROV to maintain and perform tasks at sub sea installations. These devices are extremely complicated and a great deal of knowledge of robotic theory is therefore required in order to give an analytical description of the system. This thesis will give a presentation of the fundamentals of robot modeling, including inverse kinematics and dynamics, and nonlinear robot control. The final goal is to guide the reader through the making of a dynamic model, an inverse kinematics algorithm, and a model based control system for the Raptor manipulator system. Simulations are then carried out to visualize the result and check the validity of both model, controller, and inverse kinematics algorithms.

This chapter will first give an introduction to the motivation of this thesis. Next, an outline and the contributions of this thesis will be presented.

## 1.1 Motivation

The term Remotely Operated Vehicles (ROV) refers to an underwater vehicle physically linked trough a tether to an operator on board a ship or installation. The vehicle is thus powered, and control signals are sent, through this tether. The fact that ROVs are unoccupied and highly maneuverable makes them ideal to perform operations that are hazardous or impossible to perform by humans. Throughout the history, professional divers have been used to perform some of the same tasks. Unfortunately, this resulted in heavy after effects for the divers. Also, as the oil and gas industry moves out into deeper water, the use of professional divers becomes impossible. Design and control of ROVs are therefore highly relevant for the modern industry.

Most ROVs are equipped with one or two manipulators to expand the capability of the ROV. These manipulators are typically fitted with a specialized tool or a gripper. The gripper are able to perform a variety of tasks, and special tools are made to match the gripper. For instance, if a bolt is to be loosened, the ROV will be equipped with a torque tool made such that the gripper easily can handle it. The manipulator can also be used to perform other tasks, such as collecting samples, picking things up, or perform maintenance.

There exists numerous types of robot manipulators, all depending on application. Most small industrial manipulators are actuated by electric motors. Manipulators that are used under water, are normally hydraulically powered. These manipulators are unrivaled in their speed of response and torque producing capability. On the other side, the drawbacks of hydraulic powered manipulators are that they tend to leak hydraulic fluid, requires more maintenance, and are noisy. Electric driven manipulators are both cheaper, cleaner, and less noisy. When it comes to geometry, most manipulators have six or fewer DOF. Also, there are not that many different kinematic compositions. In fact, the evolution of manipulator is partly a result of the difficulties involved in the problem of inverse kinematics.

From a control perspective, modeling of robot manipulators are important because it enables the creation and testing of different controllers. These controllers can be used to make the manipulator perform advanced and accurate tasks, and that with a user-friendly interface.

## 1.2 Outline

- **Chapter 2:** The concept of telerobotics is given as a closer introduction to the topic of interest. The chapter includes both a historical perspective on the concept and a summary of the different control architectures involved.
- **Chapter 3:** The subject of differential kinematics and dynamics is first presented to form a theoretical foundation. Then, a simplified model of the Raptor manipulator is introduced. The dynamics of this model is derived using the method of Euler-Lagrange.
- **Chapter 4:** The topic of inverse kinematics is thoroughly explained. Different solutions, depending on the kinematic structure of the manipulator, are suggested and introduced.
- **Chapter 5:** This chapter deals with the problem of controlling the manipulator. An introduction to general control is given, the control objective is defined, and a model based nonlinear control algorithm is designed.
- **Chapter 6:** The relevant concepts of digital communication are explained to facilitate the development of the interface between manipulator and user computer.

- **Chapter 7:** The derived model, control algorithm, and inverse kinematics solutions are verified through simulations. The model calculations are first controlled by simulating the model when no control forces are applied. Two chosen IK algorithms are then tested and compared. Next, the full system, including the sliding-mode algorithm, is simulated to verify and tune the controller. Finally, a brief presentation of the full-scale LabVIEW implementation of the control system is given.
- **Chapter 8:** A short discussion of different issues that have been encountered and different choices that have been made during the work are presented.
- **Chapter 9:** The work done throughout this thesis is concluded and proposals for further work are stated.

## 1.3 Contributions

- **Chapter 2:** A compact and concise overview of the topic of telemanipulators are presented.
- **Chapter 3:** A simplified model of the Raptor manipulator is suggested. The dynamic model of this manipulator is then derived. A step by step procedure for calculating the involved matrices are given.
- **Chapter 4:** Different inverse kinematics solutions are presented in a clear manner and suggested for the given application.
- **Chapter 5:** A model based sliding-mode controller that handles parameter uncertainties and unknown disturbance is derived.
- **Chapter 6:** The characteristics of the communication system are summarized and clarified.
- **Chapter 7:** Simulink models are made, and simulations performed, for the complete system. The control system is rewritten into LabVIEW code and thereby prepared for full-scale testing.

## 1.4 Software

Three different computer programs have mainly been used to solve the assignment of this thesis. A brief introduction to these programs and their area of utilizations will be given beneath.

#### Maple 14

Maple is a technical computing software for doing symbolic, numeric and graphical computations. The program is developed and sold by Waterloo Maple Inc., also known as Maplesoft [15]. Because of the efficiency and flexibility in symbolic computations, Maple has been used to derive the dynamic model and kinematics of the robot manipulator.

### MATLAB R2011b with Simulink 7.8

MATLAB, or Matrix Laboratory, is a high-level programming language and a numerical computing environment developed by MathWorks [17]. MATLAB allows algorithm development, data analysis, visualization, and numerical computation faster than traditional computer languages. In addition, MATLAB also offers a tight integration with other MathWorks products, such as Simulink. Simulink is a tool for modeling, simulation, and analyzing multi domain dynamic systems. Its primary interface is a graphical block diagramming tool and a customizable set of block libraries. MATLAB and Simulink has been used to simulate the dynamic model of the manipulator, and to present the results graphically.

### LabVIEW 2011

LabVIEW is a system design software developed by National Instruments [10]. The name LabVIEW is an abbreviation for Laboratory Virtual Instrumentation Engineering Workbench. The programming language in LabVIEW is often referred to as G and is a so called data flow, or visual, programming language. Execution is thereby determined by the structure of a graphical block diagram on which the programmer connects different functions by drawing wires. These wires propagate variables, and any function can be executed as soon as all its input data are available. LabVIEW programs, or subroutines, are called virtual instruments (VIs). Each VI consists of a block diagram, a front panel, and a connector panel. The front panel is a user interface to the block diagram, while the latter is used to represent the VI in the block diagram of other VIs. Controls and indicators on the front panel allows the user to extract data from or input data to a running VI. However, the front panel can also serve as a programmatic interface. A virtual instrument can therefore be run as a single program or as a subroutine within another block diagram. This means that subroutines easily can be tested before being embedded into a larger program. In this thesis, LabVIEW has been used to make a software suitable for controlling the robot manipulator in real time.

# Chapter 2

# Telerobotics

Telerobotics simply refers to a robot controlled by a human operator (Siciliano and Khatib [24]). Normally, the operator and controlled robot are separated by some sort of barrier. This barrier may be imposed by distance, but also by for instance hazardous environments. In other words, telerobotics deals with overcoming these barriers by remote-controlling a robot at the environment, as shown in Figure 2.1.

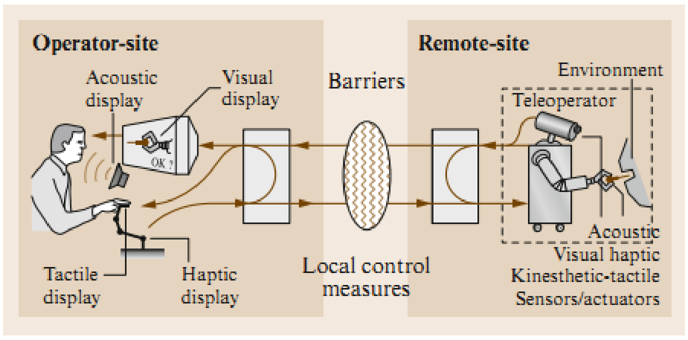

Figure 2.1: Concept of a telerobotic system, adapted from Ferrell and Sheridan [6]

As Figure 2.1 also shows, a telerobotic system is often split into two different sites. The human operator, and all components to support the interaction between man and system, are referred to as the local site. Examples of these components are joysticks, keyboards, and monitors. The remote site, on the other hand, contains the remote-controlled robot, sensors and control elements.

This chapter will give further introduction to the field of telerobotics. First, the historical development of the telerobotic system will be presented. Next, some important control architectures within the field will be introduced.

## 2.1 Historical Perspective

Ever since prehistoric times, humans have developed tools to ease and enable daily life tasks such as hunting and preparation of food. In modern times, tools are used for a wide range of applications. Treatment of hazardous goods are certainly one application that became relevant as the military began to work on nuclear substances in the 1940s (Mollet [20]). This lead to the development of a new type of tool, namely the telemanipulator. These manipulators were made to increase the dexterity and range of the human operator. In this way, one could perform simple tasks without risking human lives. The first telemanipulators were in fact electrical, controlled by simple relays(Siciliano and Khatib [24]). Unfortunately, they were slow and hard to operate. Raymond C. Goertz therefore developed a pair of mechanically linked master-slave robots. The intention was to connect the remote device as tightly and as compatibly as possible with the operator. In this way, the operator should feel and perform the tasks as if he was present at the remote site (Ferrell and Sheridan [6]). This system allowed the operator to use muscle force and hand movements to operate the slave robot while having direct view of it. On the negative side, the system had some clear limitations in both workload and distance between operator and slave. Goertz soon improved these limitations by discovering the advantage of electrically coupled manipulators. This discovery is said to be the foundation of modern telerobotics.

Later, in the 1960s, a minimal distance between the master and the slave was imposed. This was done to protect the operator from the great forces of the slave manipulator. However, this lead to another problem, namely time delays. To deal with this problem the concept of supervisory control was introduced. The approach allowed the operator to specify tasks at a high level, which again helped reducing the problem of delays. The concept of supervisory control also inspired the following development of theoretical based control.

## 2.2 Control Architectures

Normal robotic systems performs movements and actions based on an automated process. Telerobotic systems, however, require commands from and provide sensor information to the user (Siciliano and Khatib [24]). The control architecture of the telerobotic system can be categorized according to the style and level of this connection. The three main control architecture categories are direct control, shared control and supervisory control. These categories will be explained beneath.

### 2.2.1 Supervisory Control

The supervisory control was introduced by Ferrell and Sheridan [6] in 1967. As mentioned, the architecture was introduced to deal with the problem of time delays in telerobotic systems. Supervisory control therefore implies that user's commands

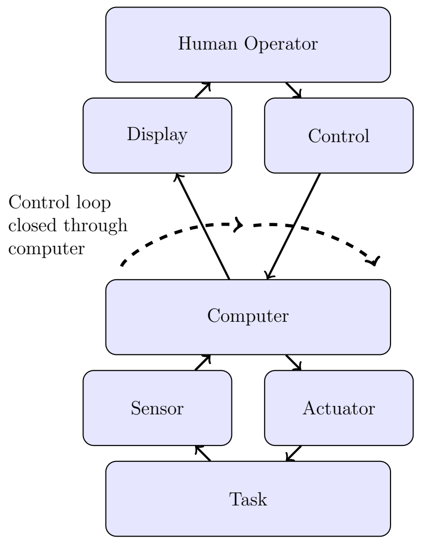

Figure 2.2: General model of supervisory control system

and feedbacks are given at a very high level. This requires a significant amount of autonomy and intelligence in the system. In Figure 2.2, a general model of supervisory control, as described by Sheridan [22], is given. In comparison, a fully automated control scheme would have no user commands sent from the operator, while a direct control scheme would have no significant computer assistance.

### 2.2.2 Shared Control

In some situations it can be beneficial to allow the human operator and the embedded controller to share control over the slave manipulator. In risky operations or long distance applications this can help to increase the safety of the remote operation. This type of control is referred to as shared control and is somewhere between supervisory and direct control when it comes to degree of automation. In a shared control robotic system, the human operator does most of the work by guiding the master manipulator as wanted. At the same time, the system monitors the operator's performance and provides stability and support through active constraints. If these constraints are superimposed into the visual or haptic scene of the operator, they are called virtual fixtures. These fixtures can help humans perform very precise robot assisted tasks by limiting the movement into restricted regions or influencing movement along a desired path (Abbott et al. [1]). The well balanced combination of human control and autonomy is well suited for tasks that requires both accuracy and the ability to make logical decisions. Such applications includes robot-assisted surgery and manipulation tasks in hazardous environment.

### 2.2.3 Direct Control

As the name suggests, direct or manual control implies no automation within the system (Siciliano and Khatib [24]). In that way, the slave motion is directly controlled by the operator via the master interface. The fact that all control commands are sent to the remote site via communication channels makes direct control unsuitable in some cases (Urbancsek [32]). Depending on the distance between local and remote site, the signals might come with a fairly significant delay. However, to avoid complications in creating local autonomy, most telerobotic systems include some degree of direct control.

Roughly speaking, direct control can be divided into two sub categories: unilateral and bilateral control. The basic ideas and issues connected to these two concepts will be explained beneath.

#### Unilateral Control

Unilateral control means that the telemanipulator is controlled directly by a master input device. This device is normally a spring centered joystick and the operator commands are thereby proportional to the joystick displacement. The slave manipulator will only respond to movement of the master controller (TeleRobotics [29]). Ergo, pushing or pulling the slave manipulator will not affect the master joystick, hence unilateral control.

The three most common control modes for controlling telemanipulators with joysticks are: position, rate, and acceleration control. The two latter modes can require substantial effort for the operator to reach and hold a given target position. However, these modes requires no kinematic coupling or synchronization of the master and the slave. Position control of a telemanipulator normally means that the operator can specify the position of the end-effector by using a joystick. The biggest challenge connected to this control mode is the mapping between master and slave positions. It is important to keep in mind that the master and the slave might not always be coupled. When the system is turned on, the master and slave robots may be positioned differently. Also, some systems allows a temporary disconnection between the two sites. This is known as clutching or indexing (Siciliano and Khatib [24]). A solution to this problem is to allow for offsets between the master and the slave.

#### Bilateral Control

In bilateral control, the human synchronously manipulates and perceives the resulting reaction force through direct feedback. If this perception happens through the master robot, i.e. the operator feels the reacting force, it is referred to as force feedback. When using force feedback, both sites of a master-slave system will respond to movement of either the master or the slave (Hirche et al. [9]). This feature is incorporated to increase the sense of being present at the remote environment. The final objective is to gain the operators ability to perform complex tasks. On the negative side, bilateral control involves a lot of communication and thereby often delays. This may cause challenging stability issues.

There are two basic force feedback architectures that are used, namely position-position and position-force. The first represents the simplest case of master-slave coupling. Here, both robots are equipped with a tracking controller in order to follow each others positions. Both the master and the slave will be subjected to forces. Whether or not these forces are equal in magnitude or scaled depends on the similarities of the two robots. It is also worth noting that the master manipulator also will be subjected to inertial, friction, and other dynamic forces from the slave. This issue is not present at the position-force architecture. Here, a force sensor placed on the slave's end-effector provides the force feedback. The human is thereby able to only experience the external forces acting on the slave. Unfortunately, this architecture has bigger stability issues.

## 2.3 The Raptor Manipulator System

The Raptor is a teleoperated master-slave manipulator system designed for usage under water or in other human hostile environments [30]. The system includes an optional position-position force feedback capability. As said, this means that both static and dynamic forces acting on the slave manipulator are reflected back to the operator through the force feedback mini-master as scaled forces. The build in control system also includes automatic manipulator stove and deploy functions, automatic self test, and programmable task execution. A complete description of the system and start up procedures can be found in the Raptor System Manual [30].

Figure 2.3 gives a description of the standard configuration of the system.

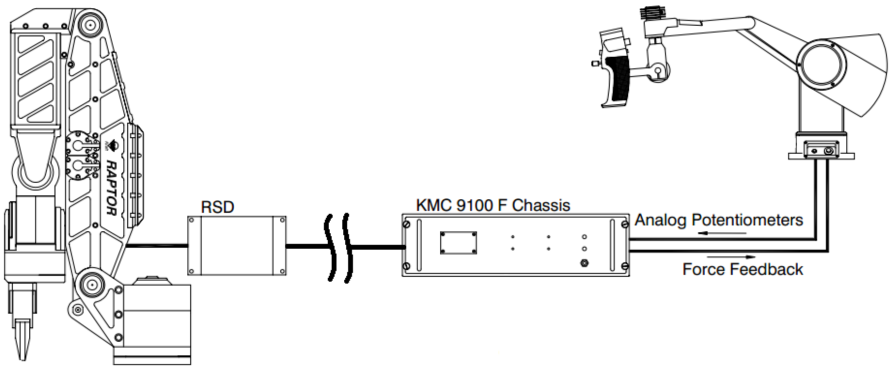

Figure 2.3: Standard configuration of the Raptor manipulator system

# Chapter 3

# Robot Modeling

The Raptor manipulator is produces by Kraft TeleRobotics [29]. Their website presents a data sheet containing relevant, but very limited information about the manipulator. This information is not sufficient to derive an accurate model of the Raptor manipulator, and thus a simplified model must be made and appropriate assumptions must be done. This chapter will present some of the given relevant information. An introduction on how to derive the differential kinematics equation will then be presented. Next, the simplified model will be introduced and the equations of motion for this model will be derived. The derivations will result in very large expressions, thus these are not given in the text. However, a precise procedure on how the model has been derived is included.

## 3.1 Notation

Since this thesis involves a lot of mathematical expressions, a brief introduction to the notation might be useful. First, all vectors and matrices are printed in bold face, and all vectors and points are equipped with a superscript to denote the reference frame. A coordinate system is written as $o_i x_i y_i z_i$, where the subscript i refers to the frame number. A robot manipulator variable is represented by $q_i$, where i refers to the joint number. The set of joint variables is given as $\boldsymbol{q} = [q_1, q_2, \cdots, q_n]^T$. A rotational matrix is written as $\boldsymbol{R}_j^i$. The subscript j denotes the frame that will be rotated, while the superscript i gives the resulting frame of the rotation. In some derivation, the arguments of the matrices have been left out for convenience. In addition, most symbols are explained in the nomenclature.

## 3.2 Operational Space, Workspace and Joint Space

The end-effector, or the last link of a serial robotic manipulator, is designed to interact with the environment. The exact nature of the end-effector depends on the application of the robot, but in most cases it consists of a gripper or a tool. If the manipulator is set to execute a task, the end-effector pose is important (Mathia [16]). In the three-dimensional case, the pose consists of both a $3 \times 1$ position vector $\boldsymbol{p}$, often given in Cartesian coordinates, and a $3 \times 1$ orientation vector $\boldsymbol{\Theta}$, given in Euler angles. Thus the end-effector pose is given as

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{p} \\ \boldsymbol{\Theta} \end{bmatrix}, \qquad \boldsymbol{x} \in \mathcal{R}^6 \tag{3.1}$$

This representation is referred to as the operation space (Khatib [12]) because it is defined in the space in which the manipulator operations are specified in.

The joints of a robot manipulator are usually physically constrained and thereby restricted from moving arbitrarily. As an example, a revolute joint might not be able to rotate a full 360 degrees due to its physical environment. These constraints leads to a subcategory of the operation space, namely the workspace. This is defined as the total volume swept out by the end-effector when the manipulator executes all possible motions. In other words, this region is limited by the upper and lower limits of the joints, and the manipulator geometry. Further, the workspace is often divided into reachable workspace and dextrous workspace (Spong et al. [27]). The dextrous workspace is defined as the set of point the manipulator can reach with an arbitrary orientation, while the reachable workspace includes all points the manipulator can reach without considering the orientation. The reachable workspace for the RFFM is given in Figure 3.1.

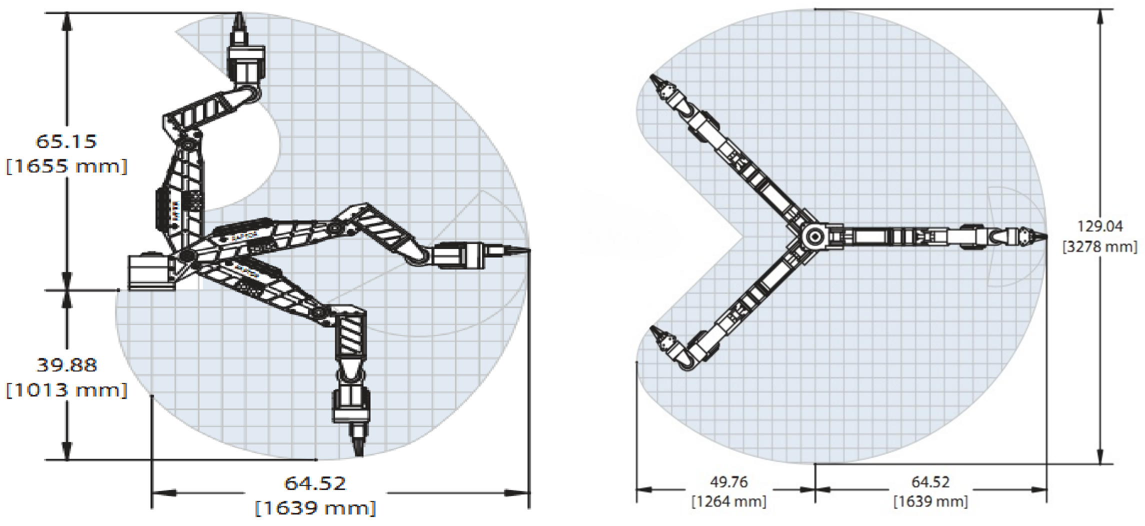

Figure 3.1: Workspace of the Raptor manipulator

The joint space, also referred to as the configuration space, is the the space in which all configurations are defined in. Due to the assumption of joints with only a single DOF, a robot manipulator with n number of joints has n DOF (Campa et al. [4]). The configuration of the robot is thereby described by a set of n joint coordinates:

$$\boldsymbol{q} = \begin{bmatrix} q_1 & q_2 & \cdots & q_n \end{bmatrix}^T, \qquad \boldsymbol{q} \in \mathcal{R}^n \tag{3.2}$$

If a robot is set to perform a task specified in the three-dimensional space with a given position and orientation, the task can be specified by m DOF. To ensure the accomplishment of the desired task, the DOF of the manipulator must satisfy $n \geq m$. In addition, if $n > m$, the manipulator is redundant, i.e. has more degrees of freedom then those required to perform the given task.

## 3.3 Differential Kinematics

In the project thesis [8], a relationship between joint positions and the end-effector pose was derived. These derivations can be used to develop a tool that connects the joint velocities to the linear and angular velocities of the end-effector. This tool is called geometric Jacobian and is a matrix that depends on the manipulator configuration. The Jacobian is considered to be one of the most important quantities within robot analysis and control.

If the end-effector pose is given by a minimal description, the Jacobian can be calculated by differentiation of the forward kinematics function with respect to the joint variables. This will result in a different Jacobian matrix called analytical Jacobian.

### 3.3.1 The Geometric Jacobian Matrix

The geometric Jacobian, or simply Jacobian, is included in the differential kinematics equations as follows

$$\dot{\boldsymbol{X}} = \begin{bmatrix} \boldsymbol{v}_n^0 \\ \boldsymbol{\omega}_n^0 \end{bmatrix} = \begin{bmatrix} \boldsymbol{J}_v \\ \boldsymbol{J}_\omega \end{bmatrix} \dot{\boldsymbol{q}} = \boldsymbol{J}_n^0(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{3.3}$$

Since there are both three angular and three linear velocities, the Jacobian matrix $\boldsymbol{J}_n^0$ will be a $6 \times n$ matrix, where n is the number of links.

The $\boldsymbol{J}_\omega$ matrix can be derived using some simple considerations. First, if joint i rotates, link i will experience an angular velocity. Following the Denavit-Hartenberg convention, this angular velocity can be expressed in the frame $i-1$ by

$$\boldsymbol{\omega}_i^{i-1} = \dot{q}\boldsymbol{k} \tag{3.4}$$

where $\boldsymbol{k} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ is the unit vector in $z_{i-1}$-direction.

The total angular velocity of the end-effector $\boldsymbol{\omega}_n^0$ can now be written as

$$\boldsymbol{\omega}_n^0 = \sum_{i=1}^{n} \rho_i \dot{q}_i \boldsymbol{k}^0 \tag{3.5}$$

The presence of $\rho_i$ is simply a way to exclude all prismatic joints as these will not influence the angular velocity. This means that $\rho_i = 0$ if joint i is prismatic, and $\rho_i = 1$ is joint i is revolute. The unit coordinate vector $\boldsymbol{k}$ needs to be expressed relative to the base frame, thus $\boldsymbol{k}^0 = \boldsymbol{R}_{i-1}^0 \boldsymbol{k}$. This means that the $\boldsymbol{J}_\omega$ matrix is given as

$$\boldsymbol{J}_\omega = \begin{bmatrix} \rho_1 \boldsymbol{k} & \rho_2 \boldsymbol{R}_1^0 \boldsymbol{k} & \cdots & \rho_n \boldsymbol{R}_{n-1}^0 \boldsymbol{k} \end{bmatrix} \tag{3.6}$$

The upper part of the Jacobian matrix $\boldsymbol{J}_v$ is now found by differentiating the linear velocity of the end-effector frame $\dot{o}_n^0$

$$\dot{o}_n^0 = \sum_{i=1}^{n} \frac{\partial o_n^0}{\partial q_i} \dot{q}_i \tag{3.7}$$

The i-th column of $\boldsymbol{J}_v$, denoted as $\boldsymbol{J}_{vi}$, is then given by

$$\boldsymbol{J}_{vi} = \frac{\partial o_n^0}{\partial q_i} \tag{3.8}$$

Unfortunately, this expression is not all simple to use. The expression only gives the linear velocity of the end-effector with all joints but the i-th fixed and the i-th joint actuated at unit velocity. It is therefore necessary to evaluate all joints separately to produce the upper part of the Jacobian $\boldsymbol{J}_v$.

However, according to Spong et al. [27], these derivations will result in the expression given in (3.9) if the type of joint is taken into consideration.

$$\boldsymbol{J}_i = \begin{bmatrix} \boldsymbol{J}_{v,i} \\ \boldsymbol{J}_{\omega,i} \end{bmatrix} = \begin{cases} \begin{bmatrix} \boldsymbol{z}_{i-1} \times (o_n - o_{i-1}) \\ \boldsymbol{z}_{i-1} \end{bmatrix} & \text{if joint i is revolute} \\ \begin{bmatrix} \boldsymbol{z}_{i-1} \\ \boldsymbol{0} \end{bmatrix} & \text{if joint i is prismatic} \end{cases} \tag{3.9}$$

Since all joints on the Raptor manipulator is revolute, the geometric Jacobian matrix is of the form

$$\begin{aligned}\boldsymbol{J}_6^0(\boldsymbol{q}) &= \begin{bmatrix}\boldsymbol{J}_1 & \boldsymbol{J}_2 & \boldsymbol{J}_3 & \boldsymbol{J}_4 & \boldsymbol{J}_5 & \boldsymbol{J}_6\end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{z}_0 \times (o_6 - o_0) & \boldsymbol{z}_1 \times (o_6 - o_1) & \cdots & \boldsymbol{z}_5 \times (o_6 - o_5) \\ \boldsymbol{z}_0 & \boldsymbol{z}_1 & \cdots & \boldsymbol{z}_5 \end{bmatrix}.\end{aligned} \tag{3.10}$$

### 3.3.2 The Analytical Jacobian Matrix

Normally, the end-effector pose is described by a position vector and the three unit vectors of the attached frame ($\boldsymbol{n}_e$,$\boldsymbol{s}_e$,$\boldsymbol{a}_e$)(Siciliano and Sciavicco [25]). This representation is quite difficult and may be inconvenient to use in practical situations. Therefore, the end-effector pose is at times specified in terms of a minimal number of parameters. The position of the end-effector reference frame $\boldsymbol{p}_e$ is still described by a minimal number of coordinates. However, now the orientation relative to the base frame is specified in terms of three Euler angles $\boldsymbol{\Theta}_e = [\phi \;\; \theta \;\; \psi]^T$ (Siciliano and Sciavicco [25]). Consequently, it is possible to describe the end-effector pose by the following vector

$$\boldsymbol{X}_e = \begin{bmatrix} \boldsymbol{p}_e \\ \boldsymbol{\Theta}_e \end{bmatrix} \tag{3.11}$$

If the end-effector pose is specified in these manners, the Jacobian can be obtained by differentiation of the forward kinematics equations with respect to the joint variables:

$$\dot{\boldsymbol{p}}_e = \frac{\partial \boldsymbol{p}_e}{\partial \boldsymbol{q}} \dot{\boldsymbol{q}} \qquad \dot{\boldsymbol{\Theta}}_e = \frac{\partial \boldsymbol{\Theta}_e}{\partial \boldsymbol{q}} \dot{\boldsymbol{q}} \tag{3.12}$$

$$\Downarrow$$

$$\dot{\boldsymbol{X}}_e = \begin{bmatrix} \dot{\boldsymbol{p}}_e \\ \dot{\boldsymbol{\Theta}}_e \end{bmatrix} = \begin{bmatrix} \frac{\partial \boldsymbol{p}_e}{\partial \boldsymbol{q}} \\ \frac{\partial \boldsymbol{\Theta}_e}{\partial \boldsymbol{q}} \end{bmatrix} \dot{\boldsymbol{q}} = \boldsymbol{J}_A(\boldsymbol{q}) \dot{\boldsymbol{q}} \tag{3.13}$$

The resulting matrix is called an analytical Jacobian and is in general different from the geometric Jacobian. For a given set of orientation angles, it is possible to find a relationship between the angular velocity $\boldsymbol{\omega}_e$ and the rotational velocity $\dot{\boldsymbol{\Theta}}$. This relationship can be derived in several ways, for instance (Fossen [7]):

$$\boldsymbol{\omega}_e = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \boldsymbol{R}_\phi^T \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \boldsymbol{R}_\phi^T \boldsymbol{R}_\theta^T \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} = \boldsymbol{T}_{\boldsymbol{\Theta}}^{-1} \dot{\boldsymbol{\Theta}} \tag{3.14}$$

where, in this case,

$$\boldsymbol{T}_{\boldsymbol{\Theta}}^{-1} = \begin{bmatrix} 1 & 0 & -s_\theta \\ 0 & c_\phi & c_\theta s_\phi \\ 0 & -s_\phi & c_\theta c_\phi \end{bmatrix} \tag{3.15}$$

$$\Downarrow$$

$$\boldsymbol{T}_{\boldsymbol{\Theta}} = \begin{bmatrix} 1 & s_\phi t_\theta & c_\phi t_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi / c_\theta & c_\phi / c_\theta \end{bmatrix} \tag{3.16}$$

Once this transformation matrix is given, the analytical Jacobian can be calculated by the geometric Jacobian as

$$\boldsymbol{J}_A(\boldsymbol{q}) = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{T}_{\boldsymbol{\Theta}} \end{bmatrix} \boldsymbol{J}(\boldsymbol{q}) \tag{3.17}$$

The analytical Jacobian is often used in situations where it is necessary to use differential quantities of variables defined in the operational space.

## 3.4 Dynamics

So far, the kinetic equations has been used to describe the motion of the robot manipulator. These equations does not take into account the forces that causes the motion. The dynamic equations, on the other hand, describes a connection between force and motion, and is thereby decisive when analyzing a dynamical system. There are several ways of deriving the dynamic equations of a mechanical system, but this thesis will use the well known Euler-Lagrange method.

### 3.4.1 Euler-Lagrange Equations of Motions

The Euler-Lagrange equations of motions describes the dynamic behavior of a mechanical system subjected to holonomic, or geometric, constraints. Holonomic constraints represents a restriction in the configuration space of the mechanical system (Mettin [19]), i.e. the links are not free to move in an arbitrary direction, but are constrained by the joints. The equations can be derived using the principle of virtual displacements and the principle of virtual work. These derivations are thoroughly covered in Spong et al. [27] and will therefore not be treated in this paper.

The Euler-Lagrange equations of motions takes the following form

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}}{\partial q_i} = \tau_i \qquad i = 1, ..., n \tag{3.18}$$

where

$$\mathcal{L} = \mathcal{K} - \mathcal{P} \tag{3.19}$$

is known as the Lagrangian, while $\mathcal{K}$ is the kinetic energy and $\mathcal{P}$ is the potential energy of the system. $q_i$, $i = 1, ..., n$ denotes the set of generalized coordinates. In this case these are given by the joint angles. Generalized coordinates are simply the minimum set of variables needed to successfully define any point in a rigid body (Spong et al. [27]).

The formulation is often referred to as the Lagrangian formulation and provides a fairly simple and systematic procedure for deriving the dynamics of a robot manipulator. The algorithm is, due to the presence of kinetic and potential energy, an energy based method. Another key aspect of this method is the independence of reference frames. As long as the Lagrangian of the system has been derived, the equations of motion can be found without regards of the body fixed coordinate frames.

Therefore, when using this method, the real problem comes down to determining the kinetic and potential energy of the system. A concise explanation on how these quantities are found will thus be presented in section 3.4.2 and 3.4.3

### 3.4.2 Kinetic Energy

The kinetic energy of a robot manipulator with rigid links can be found as the sum of the kinetic energy of each link

$$\mathcal{K} = \sum_{i=1}^{n} \mathcal{K}_i \tag{3.20}$$

Further, it is well known that the translational contribution can be given as

$$\mathcal{K}_{trans,i} = \frac{1}{2}\int_{V_i} \boldsymbol{v}_i^T \boldsymbol{v}_i \rho \, \mathrm{d}V_i = \frac{1}{2} m_i \boldsymbol{v}_i^T \boldsymbol{v}_i \tag{3.21}$$

The rotational contribution can be found by considering a simple, scalar case. Assume that there exist a rigid pendulum that moves in a pure rotation about a fixed axis. In this way, every point on the body moves in a circle and the linear velocity at a distance r from the origin can be found as

$$v = r\omega \tag{3.22}$$

A combination of (3.22) and the expression for translational kinetic energy gives

$$\mathcal{K}_{rot,i} = \frac{1}{2}m_i v^2 = \frac{1}{2}m_i r^2 \omega^2 = \frac{1}{2}I_i \omega^2 \tag{3.23}$$

where $I_i = m_i r^2$ is the moment of inertia of the body. In the non-scalar case, this expression becomes

$$\mathcal{K}_{rot,i} = \frac{1}{2}\boldsymbol{\omega}^T \boldsymbol{I}_i \boldsymbol{\omega} \tag{3.24}$$

The $\boldsymbol{I}_i$ matrix denotes the inertia tensor of link i, given with respect to the fixed global reference frame. The inertia tensor is normally calculated in the body fixed frame, but since all linear and angular velocities are given with respect to the fixed inertial frame, the inertia tensor must also be given in this reference frame. It can be shown that the body fixed inertia tensor $\boldsymbol{I}_{i,b}$ can be represented in the global coordinate frame by the relation given in (3.25).

$$\boldsymbol{I}_i = \boldsymbol{R}_i^0 \boldsymbol{I}_{i,b} \boldsymbol{R}_i^{0T} \tag{3.25}$$

The kinetic energy of each rigid link can now be stated as

$$\mathcal{K}_i = \frac{1}{2}m_i \boldsymbol{v}_i^T \boldsymbol{v}_i + \frac{1}{2}\boldsymbol{\omega}_i^T \boldsymbol{R}_i \boldsymbol{I}_{i,b} \boldsymbol{R}_i^T \boldsymbol{\omega} \tag{3.26}$$

Combining (3.26) with former derived expressions gives the kinetic energy of an n-link robot manipulator

$$\mathcal{K} = \frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{D}(\boldsymbol{q}) \dot{\boldsymbol{q}} \tag{3.27}$$

where

$$\boldsymbol{D}(\boldsymbol{q}) = \sum_{i=1}^{n} [m_i \boldsymbol{J}_{vi}(\boldsymbol{q})^T \boldsymbol{J}_{vi}(\boldsymbol{q}) + \boldsymbol{J}_{\omega_i}(\boldsymbol{q})^T \boldsymbol{R}_i(\boldsymbol{q}) \boldsymbol{I}_{i,b} \boldsymbol{R}_i(\boldsymbol{q})^T \boldsymbol{J}_{\omega_i}(\boldsymbol{q})] \tag{3.28}$$

is an $n \times n$ symmetric positive definite matrix and is referred to as the inertia matrix.

### 3.4.3 Potential Energy

Since this is a system of rigid bodies, the only contribution to the potential energy is the gravity. This means that the potential energy can be stated as

$$\mathcal{P} = \sum_{i=1}^{n} g^T r_{ci} m_i \tag{3.29}$$

where g is the gravitational vector given with respect to the inertial frame, $m_i$ is the mass of link i, and $r_{ci}$ is the vector from the origin to the center of mass of link i.

### 3.4.4 Equations of Motion

In the attempt of deriving the equations of motion, the kinetic energy is rewritten as

$$\mathcal{K} = \frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{D}(\boldsymbol{q})\dot{\boldsymbol{q}} = \frac{1}{2}\sum_{i,j=1}^{n} d_{ij}(\boldsymbol{q})\dot{q}_i\dot{q}_j \tag{3.30}$$

The Lagrangian is then given as

$$\begin{aligned}\mathcal{L} &= \mathcal{K} - \mathcal{P} \qquad (3.31)\\ &= \frac{1}{2}\sum_{i,j=1}^{n} d_{ij}(\boldsymbol{q})\dot{q}_i\dot{q}_j - \mathcal{P}\end{aligned}$$

The terms in (3.18) are now calculated as

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}_k} = \sum_j d_{kj}\ddot{q}_j + \sum_{i,j}\frac{\partial d_{kj}}{\partial q_i}\dot{q}_i\dot{q}_j \tag{3.32}$$

$$\frac{\partial \mathcal{L}}{\partial q_k} = \frac{1}{2}\sum_{i,j}\frac{\partial d_{ij}}{\partial q_k}\dot{q}_i\dot{q}_j - \frac{\partial \mathcal{P}}{\partial q_k} \tag{3.33}$$

Thus, the equations of motion can be written

$$\sum_{j=1}^{n} d_{kj}\ddot{q}_j + \sum_{i,j=1}^{n}\left[\frac{\partial d_{kj}}{\partial q_i} - \frac{1}{2}\frac{\partial d_{ij}}{\partial q_k}\right]\dot{q}_i\dot{q}_j - \frac{\partial \mathcal{P}}{\partial q_k} = \tau_k \tag{3.34}$$

By rearranging terms, the equation can be written as

$$\sum_{j=1}^{n} d_{kj}\ddot{q}_j + \sum_{i,j=1}^{n} \frac{1}{2}\left[\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}\right]\dot{q}_i\dot{q}_j - \frac{\partial \mathcal{P}}{\partial q_k} = \tau_k \tag{3.35}$$

where the terms

$$c_{ijk} = \frac{1}{2}\left[\frac{\partial d_{kj}}{\partial q_i} + \frac{\partial d_{ki}}{\partial q_j} - \frac{\partial d_{ij}}{\partial q_k}\right] \tag{3.36}$$

are referred to as Christoffel symbols of first kind. On matrix form, equation (3.35) takes the following form

$$\boldsymbol{D}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{\tau} \tag{3.37}$$

where $\boldsymbol{q} \in \mathcal{R}^n$ is the generalized coordinates, or joint positions; $\boldsymbol{D}(\boldsymbol{q}) \in \mathcal{R}^{n\times n}$ is the symmetric, bounded and positive definite inertia matrix; vector $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} \in \mathcal{R}^n$ contains centrifugal and Coriolis terms; $\boldsymbol{G}(\boldsymbol{q}) \in \mathcal{R}^n$ is the vector of buoyancy and gravity forces; and $\boldsymbol{\tau} \in \mathcal{R}^n$ is the torque vector applied at the joints.

In addition to the mentioned properties, both $\boldsymbol{D}(\boldsymbol{q})$ and $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ in (3.37) satisfies

$$\boldsymbol{x}^T\left[\dot{\boldsymbol{D}} - 2\boldsymbol{C}\right]\boldsymbol{x} = 0, \qquad \forall \boldsymbol{x} \in R^n \tag{3.38}$$

## 3.5 Information from Data Sheets

The manipulator has a total of six hydraulically powered revolute joints, excluding the gripper, hence six DOF. The total mass of the manipulator, including the base frame, is 75 kg, and the maximum lift capacity is 227 kg. Some applicable dimensions of the manipulators are given in Figure 3.2 and 3.3. Other relevant information, such as maximum range of motion of the joints can be seen in the digital PDF file given in Appendix A.

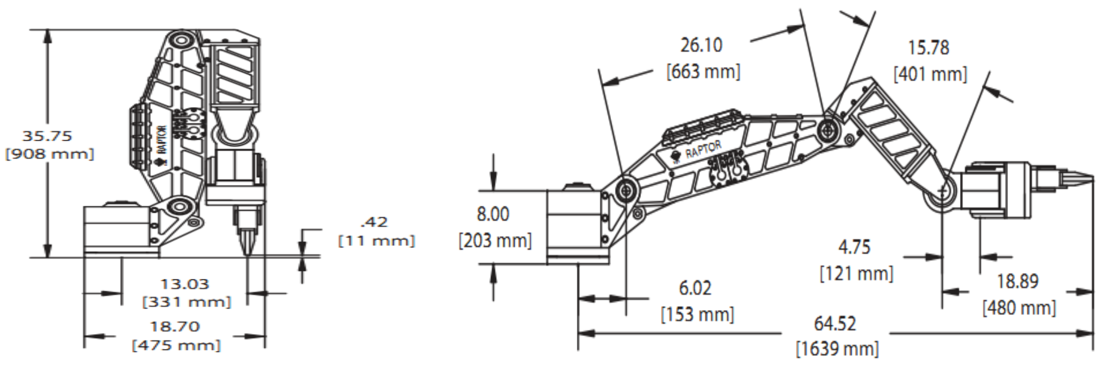

Figure 3.2: Outline drawings of the Raptor Force Feedback Manipulator, side view

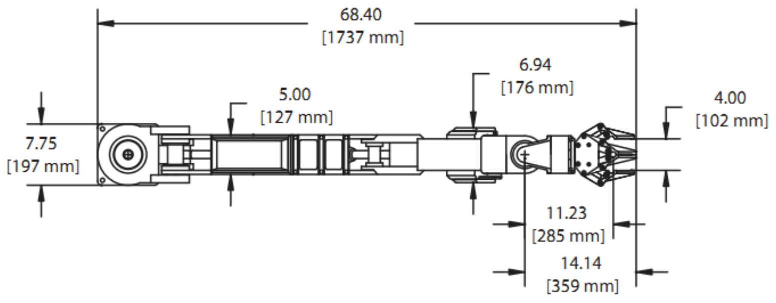

Figure 3.3: Outline drawing of the Raptor Force Feedback Manipulator, top view

## 3.6 Modeling Set-up

The information available in the data sheets is not sufficient to derive an accurate dynamic model of the Raptor manipulator. No dynamic parameters for the links are given, and there is not enough information to estimate these properly. Also, the mass and mass distribution of each link is not given. Ideally these quantities should be calculated by alternative identification methods like, for instance, CAD modeling, but this requires more detailed information about the geometry and materials of the manipulator. Due to these limitations, this thesis will propose a simplified model and make some rough estimations of the mass. Necessary assumptions are made and explained during the modeling set-up.

### 3.6.1 Simplified Model

The simplified model, given in Figure 3.4, is made to resemble the actual manipulator is the best possible way with the limited information available. Each link is drawn as a rectangle and all joints are represented by a cylinder. This is consistent with the symbolic representation explained in the project thesis [8]. The attached reference frames are chosen according to the D-H convention, and all reference frames follows the right-hand rule. Also, a rotation is considered to be positive if it follows the right-hand rule. The length of each link is denoted as $L_i$ for $i = 1, \cdots, 6$ and given in Table 3.1

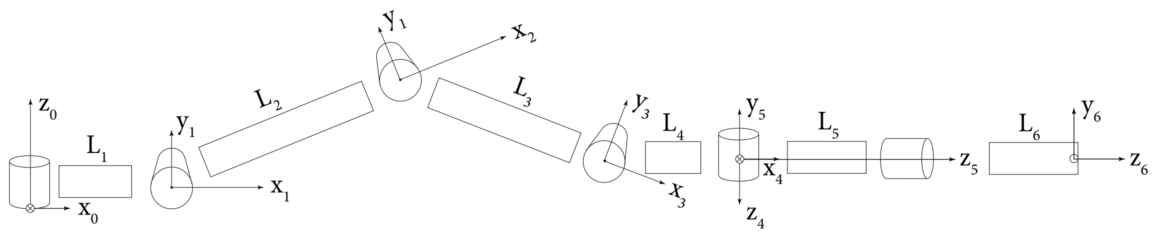

Figure 3.4: Symbolic representation of simplified manipulator model

| | |
|---|---|
| $L_1$ | 153 mm |
| $L_2$ | 663 mm |
| $L_3$ | 401 mm |
| $L_4$ | 121 mm |
| $L_5$ | 179 mm |
| $L_6$ | 179 mm |

Table 3.1: Link length of simplified manipulator model

### 3.6.2 Estimation of Mass

The mass of each link is estimated by assuming that there exist a connection between volume a mass. The volume of each link is therefore roughly calculated on the basis of the real manipulator. Knowing that the total mass of the manipulator is 75 kg, the mass of link i is calculated as

$$\text{Mass}_i = \frac{\text{Volume}_i}{\text{Total Volume}} \cdot 75\text{kg} \tag{3.39}$$

The calculated masses are given in Table 3.2.

### 3.6.3 Inertia matrix

Due to the mentioned lack of geometry information, the links are modeled as homogeneous, rectangular beams with a quadratic cross section. The symmetric property

| | |
|---|---|
| $m_1$ | 7.32 kg |
| $m_2$ | 31.90 kg |
| $m_3$ | 19.30 kg |
| $m_4$ | 13.30 kg |
| $m_5$ | 1.2 kg |
| $m_6$ | 1.95 kg |

Table 3.2: Estimated mass of each link

of these links, shown in Figure 3.5, implies that the inertia tensor becomes diagonal and simple to calculate. The inertia tensor of the beam in Figure 3.5 is shown in (3.40) and gives the structure of all inertia tensors for this manipulator. It has also been assumed that all cross sections are equal, ie. width = height = 127mm. This is, of course, not correct since the weight of links with same length are assumed to be different, but since the assumptions already are rough and inaccurate, this is not taken account for.

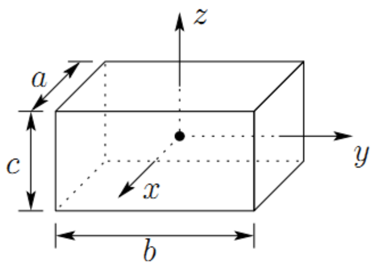

Figure 3.5: Uniform Rectangular Solid

$$\boldsymbol{I} = \begin{bmatrix} \frac{1}{12}m_i(b^2+c^2) & 0 & 0 \\ 0 & \frac{1}{12}m_i(a^2+c^2) & 0 \\ 0 & 0 & \frac{1}{12}m_i(a^2+b^2) \end{bmatrix} \tag{3.40}$$

### 3.6.4 Dynamic Model of Simplified System

The dynamic model of the simplified manipulator is derived in the same way as described in section 3.4. The D-H parameters for the simplified model are given in Table 3.3 and forms the base of these derivations.

The great number of DOF makes the calculations very extensive, and the resulting matrices are thus very large. Therefore, these calculations are performed by the help of the computational program Maple 14. Due to the great size of the resulting $\boldsymbol{D}(\boldsymbol{q})$ and $\boldsymbol{C}(\boldsymbol{q},\dot{\boldsymbol{q}})$, the terms and derivations are not included in this section, but are given in Appendix A. Also, a complete procedure on how the calculations are performed is given beneath.

**Procedure for calculating $\boldsymbol{D}(\boldsymbol{q})$ and $\boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})$:**

1. Form the homogeneous transformation matrices $\boldsymbol{A}_i$ by using the D-H convention.
2. Form the transformation matrices $\boldsymbol{T}_i^0 = \boldsymbol{A}_1 \cdots \boldsymbol{A}_i$.
3. Establish the origin of each reference frame $\boldsymbol{o}_i$. These are given by the top three elements in the fourth column of $\boldsymbol{T}_i^0$.
4. Establish the z-vectors. These are given by the top three elements in the third column of $\boldsymbol{T}_i^0$.
5. Form the upper column vector of the Jacobian. With all revolute joints, these are given as $\boldsymbol{j}_{vi} = \boldsymbol{z}_{i-1} \times (\boldsymbol{o}_n - \boldsymbol{o}_{i-1})$.
6. Form the upper Jacobian matrix for each link. For example, $\boldsymbol{J}_{vn} = \begin{bmatrix} \boldsymbol{j}_{v1} \cdots \boldsymbol{j}_{vn} \end{bmatrix}$.
7. Form the lower Jacobian matrix for each link. With all revolute joints, these are simply given as $\boldsymbol{J}_{\omega n} = \begin{bmatrix} \boldsymbol{z}_0 \cdots \boldsymbol{z}_n \end{bmatrix}$.
8. Establish the rotation matrices $\boldsymbol{R}_i^0$. These comes from the $3 \times 3$ matrix in the top left corner of $\boldsymbol{T}_i^0$.
9. Establish the inertia tensors $\boldsymbol{I}_i$ for each link.
10. Calculate the stiffness matrix for each link: $\boldsymbol{D}_i = m_i \boldsymbol{J}_{vi}^T \boldsymbol{J}_{vi} + \boldsymbol{J}_{\omega i}^T \boldsymbol{R}_i \boldsymbol{I}_i \boldsymbol{R}_i^T \boldsymbol{J}_{\omega i}$.
11. Calculate the complete stiffness matrix $\boldsymbol{D}$
12. Calculate Christoffel symbols: $c_{ijk} = \frac{1}{2}\left( \frac{\partial \boldsymbol{D}(k,j)}{\partial q_i} + \frac{\partial \boldsymbol{D}(k,i)}{\partial q_j} - \frac{\partial \boldsymbol{D}(i,j)}{\partial q_k} \right)$.
13. Determine the elements in the Coriolis/centrifugal matrix: $\boldsymbol{C}(k,j) = \sum_{i=1}^{n} c_{ijk} \dot{q}_i$

| **Link** | $\boldsymbol{a_i}$ | $\boldsymbol{\alpha_i}$ | $\boldsymbol{d_i}$ | $\boldsymbol{\theta_i}$ |
|---|---|---|---|---|
| 1 | $a_1 = 153$mm | 90 | 0 | $\theta_1$ |
| 2 | $a_2 = 663$mm | 0 | 0 | $\theta_2$ |
| 3 | $a_3 = 401$mm | 0 | 0 | $\theta_3$ |
| 4 | $a_4 = 121$mm | 90 | 0 | $\theta_4$ |
| 5 | 0 | -90 | 0 | $\theta_5$ |
| 6 | 0 | 0 | $d_6 = 480$mm | $\theta_6$ |

Table 3.3: The D-H Parameters of the Simplified Model

### Calculations of $\boldsymbol{G}(\boldsymbol{q})$

The potential energy of the system is found by summing the potential energy contributions of all links. By defining the base frame as the neutral point, it comes clear that link 1 will give no contribution to the potential energy. The potential energy of the remaining links is calculated beneath.

$$\mathcal{P}_2 = m_2 \cdot g \cdot \frac{L_2}{2} \cdot sin(q_2) \tag{3.41}$$

$$\mathcal{P}_3 = m_3 \cdot g \cdot \left( L_2 \cdot sin(q_2) + \frac{L_3}{2} \cdot sin(q_2 + q_3) \right) \tag{3.42}$$

$$\mathcal{P}_4 = m_4 \cdot g \cdot \left( L_2 \cdot sin(q_2) + L_3 \cdot sin(q_2 + q_3) + \frac{L_4}{2} \cdot sin(q_2 + q_3 + q_4) \right) \tag{3.43}$$

$$\begin{aligned} \mathcal{P}_5 = & m_5 \cdot g \cdot \Bigg( L_2 \cdot sin(q_2) + L_3 \cdot sin(q_2 + q_3) \\ & + \left[ L_4 + \left( \frac{L_5}{2} \cdot sin(-q_5) \right) \right] \cdot sin(q_2 + q_3 + q_4) \Bigg) \end{aligned} \tag{3.44}$$

$$\begin{aligned} \mathcal{P}_6 = & m_6 \cdot g \cdot \Bigg( L_2 \cdot sin(q_2) + L_3 \cdot sin(q_2 + q_3) \\ & + \left[ L_4 + \left( \left( L_5 + \frac{L_6}{2} \right) \cdot sin(-q_5) \right) \right] \cdot sin(q_2 + q_3 + q_4) \Bigg) \end{aligned} \tag{3.45}$$

The ith row of the gravity and buoyancy vector $\boldsymbol{G}(\boldsymbol{q})$ is now found by differentiating the total potential energy, given in (3.46), with respect to $q_i$. This can be seen in (3.47).

$$\mathcal{P} = \sum_{i=1}^{6} \mathcal{P}_i \tag{3.46}$$

$$\boldsymbol{G}(\boldsymbol{q}) = \begin{bmatrix} \frac{\partial \mathcal{P}}{\partial q_1} & \frac{\partial \mathcal{P}}{\partial q_2} & \frac{\partial \mathcal{P}}{\partial q_3} & \frac{\partial \mathcal{P}}{\partial q_4} & \frac{\partial \mathcal{P}}{\partial q_5} & \frac{\partial \mathcal{P}}{\partial q_6} \end{bmatrix}^T \tag{3.47}$$

The calculations of the elements in this vector can be seen in Appendix A.

# Chapter 4

# Inverse Kinematics

When controlling a robot manipulator it is often necessary to find the joint variables as a function of the position and orientation of the end-effector. In other words, the joint values needs to be decided in order to position the end-effector at a desired target pose. This is known as the problem of inverse kinematics (IK) and stands in contrast to the forward kinematics problem.

Generally, the IK problem can be stated in matrix form as (Spong et al. [27])

$$\boldsymbol{T}_n^0(q_1, ..., q_n) = \boldsymbol{A}_1(q_1) \cdots \boldsymbol{A}_n(q_n) = \boldsymbol{H} \tag{4.1}$$

Where the matrix $\boldsymbol{H}$ is a homogeneous transformation and represents the desired position and orientation of the end-effector. On component form, this results in twelve nonlinear equations in n unknown variables. The complexity of these equations makes them hard to solve directly. In addition, there is an uncertainty regarding the existence and uniqueness of the solution. The existence of a solution depends not only on mathematical considerations, but also the physical limitations of the manipulator. Revolute joints are, for instance, often restricted from rotating a full 360 degrees. Accordingly, after solutions to the mathematical equations are identified, the solution must be checked to see if it fits with the physical constraints of the manipulator. Also, if a solution exist, it may not be unique, as implied by Figure 4.1.

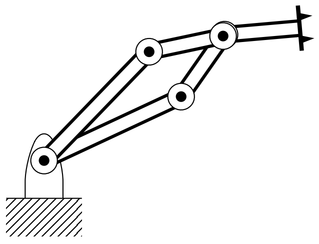

Figure 4.1: Two alternative solutions to the IK problem

Depending on the joint axis geometry of the manipulator system, closed-form or different numerical solutions to the IK problem can be determined. Generally, closed-form solutions are preferable for two reasons. First, they allows one to generate rules for choosing one solution over another. In this way, multiple solutions to the problem will not cause any trouble. Second, a closed-form expression is a lot quicker to compute than a heavy iterative search. Therefore, the robot is allowed to move at greater velocities.

This chapter will first give a further presentation to closed-form solutions of the IK problem. Next, an introduction to numerical methods for solving the IK problem will be given. Since there exist an incredible number of numerical methods, this thesis will only focus on methods based on the differential kinematics.

## 4.1 Closed-Form Solutions

In the attempt of finding a closed-form solution to the problem of IK, it is often useful to consider the particular kinematic structure of the manipulator. By exploiting this structure the problem can be decoupled into two simpler problems, namely inverse position kinematics and inverse orientation kinematics. The simplification is unfortunately only valid for manipulators having six joints, with the last three joints intersecting at a point. However, this is a common kinematic arrangement and is for that reason of great interest. In fact, the problem of IK has had a great influence on how the manipulator design has evolved and thus most manipulators are kinematically simple. Also, since there are few general methods for finding a closed-form solution, this part will give a brief introduction on how to solve the mentioned arrangement. In literature, this is known as a geometric approach.

The fact that the axes $z_3$, $z_4$, and $z_5$ intersects at a point means that there is a spherical wrist at the intersection point. According to the Denavit-Hartenberg convention, the origins $o_4$ and $o_5$ will then always be at the wrist center $o_c$. Consequently, any motion of the three last links about these axes will not influence the position of $o_c$. In addition, if $z_5$ and $z_6$ are the same axis, and the distance from $o_c$ to the desired end-effector position o is given by $d_6$, the coordinates of the wrist center can be written as

$$o_c^0 = o - d_6\boldsymbol{R}\begin{bmatrix}0\\0\\1\end{bmatrix} \tag{4.2}$$

Using that $o_c^0 = \begin{bmatrix}x_c & y_c & z_c\end{bmatrix}^T$ and $o = \begin{bmatrix}o_x & o_y & o_z\end{bmatrix}^T$ gives

$$\begin{bmatrix}x_c\\y_c\\z_c\end{bmatrix} = \begin{bmatrix}o_x - d_6r_{13}\\o_y - d_6r_{23}\\o_z - d_6r_{33}\end{bmatrix} \tag{4.3}$$

Now the orientation transformation $\boldsymbol{R}_3^0$ can be found by determining the values of the first three joint variables. Further, this can be used to derive the orientation of the end-effector relative to $o_3x_3y_3z_3$ as follows

$$\boldsymbol{R}_6^3 = (\boldsymbol{R}_3^0)^T\boldsymbol{R} \tag{4.4}$$

Exactly how these computations are performed will not be further explained as these can be seen in Spong et al. [27].

There also exist a second group of methods capable of providing closed-form solutions. This type is called algebraic methods and involves algebraic manipulations of the kinematic equations. It is clear that this method is mostly used on simple kinematic structures with few degrees of freedom.

## 4.2 Numerical Solutions

For manipulators with high degrees of freedom, or for redundant manipulators, a closed-form solution to the IK problem is very difficult. In some cases it might even be impossible to obtain a closed-form solution, thus different numerical techniques must be used (Meredith and Maddock [18]). These techniques are convenient due to their scalability. They are not robot dependent and can thereby be applied to any kinematic structure (Siciliano and Sciavicco [25]).

There exist a variety of numerical methods for solving the problem inverse kinematics. However, due to the large extent of the topic, this thesis will only focus on methods based on the inversion of the velocity mapping described in section 3.3:

$$\dot{\boldsymbol{X}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{4.5}$$

### Jacobian Inverse

If the Jacobian matrix is square, i.e. $m = n$, the joint velocities can be found by simply inverting the Jacobian:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{-1}(\boldsymbol{q})\dot{\boldsymbol{X}} \tag{4.6}$$

The corresponding joint positions $\boldsymbol{q}(t)$ are then obtained by integration of $\dot{\boldsymbol{q}}(t)$ for a given $\boldsymbol{q}(0)$. If joint accelerations are required, these can be obtained by differentiation of joint velocities. In addition, a low pass filter should be included to remove noise. The proposed solution is presented graphically in Figure 4.2. Of course, this is only valid if $\boldsymbol{J}(\boldsymbol{q})$ is invertible. The method is often referred to as Resolved Motion Rate Control (RMRC) (Whitney [35]).

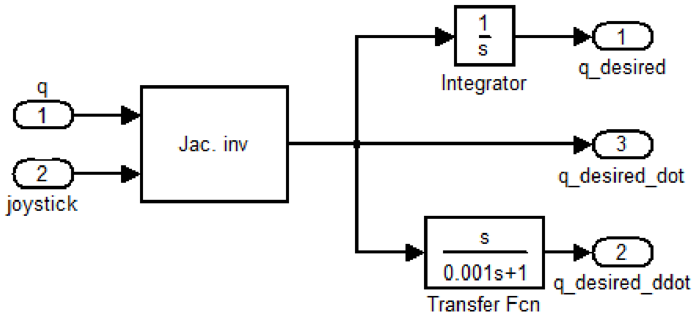

Figure 4.2: A proposed solution on how to derive the joint positions, velocities, and accelerations by the RMRC method.

### Jacobian Pseudoinverse

If the number of internal DOF is greater than the dimension of task space ($n > m$), the manipulator is said to be redundant. The Jacobian of a redundant manipulator will always be non-square, thus no inverse exist. However, equation (4.5) can still be solved by using an alternative or generalized Jacobian inverse. This generalized Jacobian is often denoted $\boldsymbol{J}^+$.

$$\boldsymbol{J}\boldsymbol{J}^+\boldsymbol{J} = \boldsymbol{J} \tag{4.7}$$
$$\boldsymbol{J}^+\boldsymbol{J}\boldsymbol{J}^+ = \boldsymbol{J}^+ \tag{4.8}$$
$$(\boldsymbol{J}^+\boldsymbol{J})^* = \boldsymbol{J}^+\boldsymbol{J} \tag{4.9}$$
$$(\boldsymbol{J}\boldsymbol{J}^+)^* = \boldsymbol{J}\boldsymbol{J}^+ \tag{4.10}$$

If $\boldsymbol{J}^+$ satisfies all four above mentioned relationships, it is called a Moore-Penrose pseudoinverse or simply pseudoinverse (Buss and Kim [3]). This matrix can be used instead of $\boldsymbol{J}^{-1}$, as shown in (4.11), to obtain the best possible solution to (4.5) in the sense of least-squares.

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^+(\boldsymbol{q})\dot{\boldsymbol{X}} \tag{4.11}$$

For redundant manipulators, there will always exist a null space due to difference in space dimensions ($n \neq m$). This null space is the set of task space velocities that gives no joint space velocities at the current manipulator configuration. A common way to include this null space in the solution was introduced by Liégeois [14] and is given by

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^+(\boldsymbol{q})\dot{\boldsymbol{X}} + (\boldsymbol{I} - \boldsymbol{J}^+\boldsymbol{J})\boldsymbol{b} \tag{4.12}$$

where $\boldsymbol{b} \in \Re^n$ is an arbitrary vector.

The pseudoinverse is normally calculated as shown in (4.13) and has some properties that are similar to inverse matrices. In fact, if $\boldsymbol{J}$ is square and invertible, then $\boldsymbol{J}^+ = \boldsymbol{J}^{-1}$.

$$\boldsymbol{J}^+ = \boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{J}^T)^{-1} \tag{4.13}$$

**Jacobian Transpose**

A third alternative within RMRC is a solution based on the transpose of the Jacobian. The principle is simple: use the transpose $\boldsymbol{J}^T$ instead of the inverse $\boldsymbol{J}^{-1}$ to obtain the solution of (4.5). It is clear that $\boldsymbol{J}^T \neq \boldsymbol{J}^{-1}$, however the use of the transpose can be justified in terms of virtual forces (Buss and Kim [3]). The method is considered to be stable and fast due to its low computational cost, but has poor quality (Siciliano [23]). Another advantage of the solution is that it may avoid numerical issues caused by kinematic singularities. On the negative side, the solution tends to oscillate when target position is unreachable.

**Damped Least-Squares**

The pseudoinverse method is great for many applications, but it also suffers in terms of stability near singularities. This occurs when the end-effector moves in a direction not achievable by changes in joint angles. As a result, the joint velocities can become arbitrarily large near these singular configurations. To prevent this from happening, Wampler [34] and Nakamura and Hanafusa [21] came up with an approach called the damped least-squares (DLS) method. This method is based on finding the solution that minimizes the sum

$$\left\|\dot{\boldsymbol{X}} - \boldsymbol{J}\dot{\boldsymbol{q}}\right\|^2 + \lambda^2 \left\|\dot{\mathbf{q}}\right\|^2 \tag{4.14}$$

where $\lambda \geq 0$ and is known as the damping factor. In most of the literature, the solution to this minimization problem takes the following form.

$$\dot{\boldsymbol{q}} = (\boldsymbol{J}^T\boldsymbol{J} + \lambda^2\boldsymbol{I})^{-1}\boldsymbol{J}^T\dot{\boldsymbol{X}} \tag{4.15}$$

This corresponds to a modified Jacobian matrix that is nonsingular throughout the whole workspace. The problem is now to choose an appropriate damping factor $\lambda$. Small values will give accurate solutions, but reduced robustness near singular points. Large $\lambda$ values will result in low tracking accuracy even when accurate solutions are possible. Note that in (4.15), $\lambda$ must be non-zero when the manipulator is redundant.

However, using singular value decomposition, it can be shown that the solution to the minimization problem also can be expressed as

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{J}^T + \lambda^2\boldsymbol{I})^{-1}\dot{\boldsymbol{X}} \tag{4.16}$$

This solution is also valid for $\lambda = 0$, provided $\boldsymbol{J}$ has full rank. In fact, it can be seen that when no damping is applied ($\lambda = 0$), the case corresponds to the pseudoinverse solution

$$\begin{aligned} &\lambda = 0 \\ &\;\Downarrow \\ &\dot{\boldsymbol{q}} = \boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{J}^T)^{-1}\dot{\boldsymbol{X}} = \boldsymbol{J}^+(\boldsymbol{q})\dot{\boldsymbol{X}} \end{aligned} \tag{4.17}$$

# Chapter 5

# Control of Robot Manipulator

The control problem for robot manipulators is the problem of determining the time history of joint torques required to give the end-effector a desired motion. There exist several control techniques and methodologies that can be applied to robot manipulators, but this thesis focuses on the sliding-mode control technique. This chapter will first give an introduction to the general control problem and try to clarify the differences between joint space control and operational space control. Next, the term tracking is explained, since this is chosen as the control objective for the given manipulator. Finally, the basic idea and background of sliding-mode control is presented, a proper sliding-mode controller is derived and proven stable, and the term scattering is explained.

## 5.1 The Control Problem in General

In general, most robotic manipulators are fully autonomous, computer controlled, and used in factories to perform repetitive tasks such as painting, welding, or transportation of objects. The reference generation for these manipulators is fairly easy because the given path or trajectory will be repeated several times and can be designed in advance. The motion of the manipulator can also be programmed, or taught, by physically moving the arm through the desired path and thereby logging the time evolution of the joint angles. Other manipulators, such as the RFFM, might be controlled by an operator. This means that the trajectories of the system must be generated continuously. There are several ways of specifying these trajectories and controlling the manipulator accordingly, all depending on whether the dynamics are formulated in the operational space or, as in this thesis, in the joint space.

If the controller is based on the joint space dynamics, i.e. a joint space controller, the desired trajectory must be given in terms of joint angles. This is often inconvenient with regard to the operator, since these trajectories are not task oriented (Mettin [19]). Thus, an operator will often prefer to specify the position of the end-effector in the operational space, by for instance using a joystick. However, since the control action is carried out in the joint space, an inverse kinematics algorithm is needed to transform the motion requirements $\boldsymbol{x}_d$ into a corresponding motion $\boldsymbol{q_d}$ given in the joint space (Siciliano and Sciavicco [25]). A summary of suitable IK algorithms are given in Chapter 4. The general scheme of joint space control can be seen in Figure 5.1.

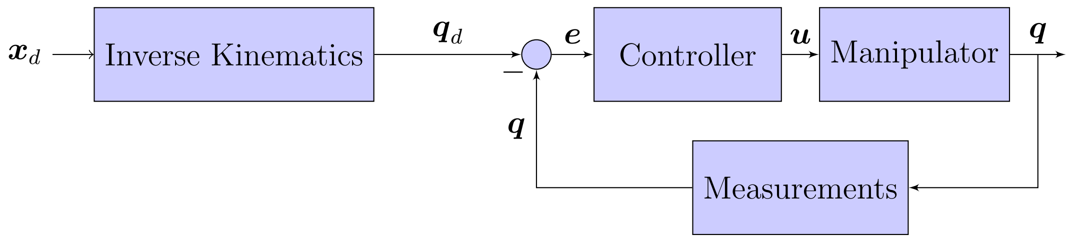

Figure 5.1: General scheme of joint space control

Operational space controllers are more complex due to the fact that the inverse kinematics are embedded into the feedback loop. This type of controllers have a conceptual advantage in the way that they can act directly on operational space variables. The general scheme of operational control is given in Figure 5.2.

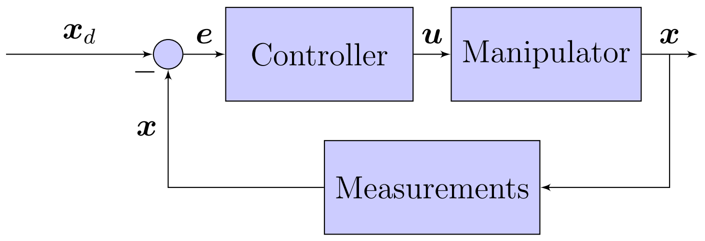

Figure 5.2: General scheme of joint space control

## 5.2 Tracking

According to Skjetne [26] and Fossen [7], the problem of tracking involves steering the systems output $y(t) \in \mathcal{R}^n$ to a time dependent, desired output $y_d(t) \in \mathcal{R}^n$. The desired output can be seen as a point in $\mathcal{R}^n$ moving as function of time, thus a trajectory equal to the one we want our system to follow is traced out by this point. The velocity and acceleration along this path is found by taking the time derivative of $y_d(t)$. This means that tracking both will give a desired path and make sure that the dynamics, i.e. the velocity and acceleration, along this path is satisfied. Tracking is therefore more restrictive than other control objective, such as path following, where the dynamics along the path is not important.

## 5.3 Sliding-Mode Control

A robust nonlinear design technique for controlling complex nonlinear systems is the sliding-mode control. This technique has been studied since the early sixties by the name variable system structure (VSS). The name variable structure system stems from the fact that the state-feedback control law is not a continuous function of time. Instead, the controller can switch between continuous structures, depending on the current position in the state space. The control structure of sliding-mode control is designed such that the trajectories will move towards a sliding surface or manifold. It is important that the trajectory converges to this surface in finite time. As it reaches this manifold, the dynamics is ruled by a new control structure and the trajectory will slide along this sliding surface. This means that the closed-loop response becomes insensitive to disturbances and other uncertainties. The method is therefore known for its robustness when it comes to modeling uncertainties, system parameters variation, external disturbances. One negative aspect of sliding-mode is that digital implementation of the controller can lead to a phenomenon called chattering, and also possible instability with large gains. Chattering occurs due to the limitations in sampling interval of switching devices. There are ways to reduce, or possibly remove, this phenomenon. Some of these will be presented in this paper.

### 5.3.1 The Concept of Sliding-Mode Control

By the definitions of Khalil [11], the basic idea of sliding-mode control is to design a controller that constraints the motion of a system to a manifold $\boldsymbol{z}$. If the given system is SISO and written in the same form as (5.5) and (5.6), the manifold will typically be

$$z = ax_1 + x_2, \qquad a > 0 \tag{5.1}$$

On this manifold, i.e. $z = 0$, the dynamics of the system is ruled by $\dot{x_1} = -ax_1$, since $z = 0 \Rightarrow x_2 = -ax_1$ and $\dot{x_1} = x_2$. This means that $x(t)$ converges to zero at time goes to infinity, and the motion on the manifold is therefore stable. This phase is sometimes denoted as the sliding phase.

The next task is therefore to control the system such that it reaches this manifold in finite time, also known as the reaching phase. Note that asymptotic stability is not sufficient because this means that the system never will reach the manifold, only converge towards it. The reaching phase is performed by making a sliding mode controller that ensures that $\dot{z}$ goes to zero in finite time and stays there. These sliding mode controllers often include a signum function that can cause a phenomena called scattering. An introduction to this phenomena and a proposed solution will be presented in section 5.3.4.

### 5.3.2 Problem Statement

The mathematical model of the robot manipulator has been derived previously and is given in (3.37). However, in order to make the model as true as possible, friction forces $\boldsymbol{F}(\dot{\boldsymbol{q}})$ and disturbances $\boldsymbol{\delta_0}$ are added. The friction force includes viscous friction proportional to the joint velocity. Even though no information about the possible friction force is given, it is assumed to be known. The disturbances acting on the system are unknown, but assumed to be bounded by a known scalar. The final dynamics then becomes

$$\boldsymbol{D}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{C}(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{F}(\dot{\boldsymbol{q}}) + \boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{\tau} + \boldsymbol{\delta_0} \tag{5.2}$$

Also, uncertainties and unmodeled dynamics in the actuators are assumed to be captured by a multiplicative term $\boldsymbol{K} \in R^{n \times n}$ and an additive term $\boldsymbol{\delta_\tau}$. In (5.3) $\boldsymbol{\tau_0}$ gives the actual torque input.

$$\boldsymbol{\tau} = \boldsymbol{K}\boldsymbol{\tau_0} + \boldsymbol{\delta_\tau} \tag{5.3}$$

For convenience, the states are rewritten as

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x_1} & \boldsymbol{x_2} \end{bmatrix}^T = \begin{bmatrix} \boldsymbol{q} & \dot{\boldsymbol{q}} \end{bmatrix}^T \tag{5.4}$$

A combination of (5.2),(5.3), and the new notation in (5.4) gives the following dynamic expressions

$$\dot{\boldsymbol{x_1}} = \boldsymbol{x_2} \tag{5.5}$$

$$\boldsymbol{D}\dot{\boldsymbol{x}}_{\boldsymbol{2}} = -\boldsymbol{C}\boldsymbol{x_2} - \boldsymbol{F} - \boldsymbol{G} + \boldsymbol{K}\boldsymbol{\tau_0} + \boldsymbol{\delta} \tag{5.6}$$

where the arguments of $\boldsymbol{D}(\boldsymbol{x_1})$, $\boldsymbol{C}(\boldsymbol{x_1}, \boldsymbol{x_2})$, $\boldsymbol{F}(\boldsymbol{x_2})$, and $\boldsymbol{G}(\boldsymbol{x_1})$ have been left out for practical reasons, and $\boldsymbol{\delta}$ is given as

$$\boldsymbol{\delta} = \boldsymbol{\delta_0} + \boldsymbol{\delta_\tau} \tag{5.7}$$

### 5.3.3 Derivation of Sliding-Mode Controller

As described in section 5.3.1 , a sliding manifold, or sliding surface, $\boldsymbol{z}$ must be defined to ensure that the motion on this manifold converges to zero when time goes to infinity. In this case, a proper sliding surface is given by

$$\boldsymbol{z} = \boldsymbol{x_2} - \boldsymbol{\alpha}(\boldsymbol{x_1}, t) \tag{5.8}$$

Now, $\boldsymbol{z} = 0$ gives

$$\boldsymbol{x_2} = \boldsymbol{\alpha}(\boldsymbol{x_1}, t) \tag{5.9}$$
$$\Downarrow$$
$$\dot{\boldsymbol{x}}_{\boldsymbol{1}} = \boldsymbol{\alpha}(\boldsymbol{x_1}, t) \tag{5.10}$$

Since this is a tracking problem, the tracking error $\boldsymbol{e}_1$ must be defined as

$$\boldsymbol{e_1} = \boldsymbol{x_1} - \boldsymbol{q_d} \tag{5.11}$$

The time derivative of the error $\dot{\boldsymbol{e}}_{\boldsymbol{1}}$ then becomes

$$\dot{\boldsymbol{e}}_{\boldsymbol{1}} = \boldsymbol{\alpha} - \dot{\boldsymbol{q}}_{\boldsymbol{d}} \tag{5.12}$$

If $\boldsymbol{\alpha}_1$ is chosen as

$$\boldsymbol{\alpha_1} = -\boldsymbol{A}(\boldsymbol{x_1} - \boldsymbol{q_d}) + \dot{\boldsymbol{q}}_{\boldsymbol{d}} = -\boldsymbol{A}\boldsymbol{e_1} + \dot{\boldsymbol{q}}_{\boldsymbol{d}} \tag{5.13}$$

the error dynamics takes the following form

$$\dot{\boldsymbol{e}}_{\boldsymbol{1}} = -\boldsymbol{A}\boldsymbol{e_1} \tag{5.14}$$

The error dynamics is thus asymptotically stable if and only if for all eigenvalues $\lambda$ of $\boldsymbol{A}$, $Re(\lambda) > 0$. Choosing the $\boldsymbol{A}$ matrix as a diagonal matrix with only positive constants will certainly give the system asymptotic stability.

The time derivative of $\dot{\boldsymbol{\alpha}}$ will come in handy later on and is therefore calculated as

$$\dot{\boldsymbol{\alpha}}_{\boldsymbol{1}} = -\boldsymbol{A}\dot{\boldsymbol{e}}_{\boldsymbol{1}} + \ddot{\boldsymbol{q}}_{\boldsymbol{d}} = \boldsymbol{A}^2\boldsymbol{e_1} + \ddot{\boldsymbol{q}}_{\boldsymbol{d}} \tag{5.15}$$

As convergence in the sliding phase is proven, it is time to develop a proper sliding mode controller. First, the time derivative of the sliding surface can be written as

$$\dot{\boldsymbol{z}} = \dot{\boldsymbol{x_2}} - \dot{\boldsymbol{\alpha}} \tag{5.16}$$

Multiplying all terms with $\boldsymbol{D}$ and inserting the expression for $\boldsymbol{D}\dot{\boldsymbol{x}}_{\boldsymbol{2}}$ given in (5.6) leads to

$$\boldsymbol{D}\dot{\boldsymbol{z}} = -\boldsymbol{C}\boldsymbol{x_2} - \boldsymbol{F} - \boldsymbol{G} + \boldsymbol{K}\boldsymbol{\tau_0} + \boldsymbol{\delta} - \boldsymbol{D}\dot{\boldsymbol{\alpha}} \tag{5.17}$$

For simplicity, some terms are gathered and denoted as

$$\boldsymbol{\phi} = -\boldsymbol{F} - \boldsymbol{G} - \boldsymbol{D}\dot{\boldsymbol{\alpha}} \tag{5.18}$$

which gives the following equation

$$\boldsymbol{D}\dot{\boldsymbol{z}} = \boldsymbol{\phi} + \boldsymbol{K}\boldsymbol{\tau_0} + \boldsymbol{\delta} - \boldsymbol{C}\boldsymbol{x_2} \tag{5.19}$$

In order to make a sliding-mode controller and prove that the system converges to the sliding surface in finite time, a Control Lyapunov Function (CLF) will be used. We suggests the CLF given in (5.20) for this purpose. This CLF satisfies the required properties of Lyapunov functions given in (5.21) and (5.22). This is due to the stated fact that the $M$ matrix is positive definite.

$$V = \boldsymbol{z}^T\boldsymbol{D}\boldsymbol{z} \tag{5.20}$$
$$V(0) = 0 \tag{5.21}$$
$$V(x) \leq 0 \quad \forall\ \boldsymbol{x} \in [R^n - \{0\}] \tag{5.22}$$

The derivative of V along the trajectories of the system is calculated as

$$\begin{aligned}\dot{V} &= 2\boldsymbol{z}^T\boldsymbol{D}\dot{\boldsymbol{z}} + \boldsymbol{z}^T\dot{\boldsymbol{D}}\boldsymbol{z} \\ &= 2\boldsymbol{z}^T\left[\boldsymbol{\phi} + \boldsymbol{K}\boldsymbol{\tau_0} + \boldsymbol{\delta} - \boldsymbol{C}\boldsymbol{x_2}\right] + \boldsymbol{z}^T\dot{\boldsymbol{D}}\boldsymbol{z} \\ &= 2\boldsymbol{z}^T\left[\boldsymbol{\phi} + \boldsymbol{K}\boldsymbol{\tau_0} + \boldsymbol{\delta} - \boldsymbol{C}(\boldsymbol{z} + \boldsymbol{\alpha_1})\right] + \boldsymbol{z}^T\dot{\boldsymbol{D}}\boldsymbol{z} \\ &= 2\boldsymbol{z}^T\left[\boldsymbol{\phi} + \boldsymbol{K}\boldsymbol{\tau_0} + \boldsymbol{\delta} - \boldsymbol{C}\boldsymbol{\alpha_1}\right] - 2\boldsymbol{z}^T\boldsymbol{C}\boldsymbol{z} + \boldsymbol{z}^T\dot{\boldsymbol{D}}\boldsymbol{z} \\ &= 2\boldsymbol{z}^T\left[\boldsymbol{\phi} + \boldsymbol{K}\boldsymbol{\tau_0} + \boldsymbol{\delta} - \boldsymbol{C}\boldsymbol{\alpha_1}\right] + \boldsymbol{z}^T\left[\dot{\boldsymbol{D}} - 2\boldsymbol{C}\right]\boldsymbol{z}\end{aligned} \tag{5.23}$$

Note that the final term in the last line corresponds to the property given in (3.38), thus the final expression for $\dot{V}$ is

$$\dot{V} = 2\boldsymbol{z}^T \left[\boldsymbol{\phi} + \boldsymbol{K}\boldsymbol{\tau_0} + \boldsymbol{\delta} - \boldsymbol{C}\boldsymbol{\alpha_1}\right] \tag{5.24}$$

In order to prove that $\boldsymbol{z}$ goes to zero at a finite time, some assumptions on the uncertainties must be done. First, it is assumed that the $\boldsymbol{K}$ matrix, representing the multiplicative term in the actuator disturbances, is a simple diagonal matrix with all positive constants. Also, the diagonal terms are assumed to be bounded by a positive, real number $k_0$ that is smaller than each term and a maximum value $k_{max}$, thus

$$\boldsymbol{K} = \text{diag}(k_1, k_2, \cdots, k_n) \tag{5.25}$$
$$0 < k_0 < k_i < k_{max} \quad i = 1, 2, \cdots, n \tag{5.26}$$

The disturbance vector $\boldsymbol{\delta}$ is assumed to be bounded by

$$\|\boldsymbol{\delta}\| \leq \rho \tag{5.27}$$

The vector $\boldsymbol{\psi}$ has the following form

$$\boldsymbol{\psi} = \begin{bmatrix} \psi_1(z_1) \\ \psi_2(z_2) \\ \vdots \\ \psi_n(z_n) \end{bmatrix} \tag{5.28}$$

where $\psi_i$, $i = 1, 2, \cdots, n$, is an alternative to the mentioned signum function and calculates by

$$\psi_i = (1 + \epsilon_1) \tanh\left(\frac{z_i}{\epsilon_2}\right) \tag{5.29}$$

Here $z_i$ correspond to the i-th row of the $\boldsymbol{z}$ vector.

Further, there exists two constant matrices $\boldsymbol{L}$ and $\boldsymbol{H}$ given by

$$\boldsymbol{L} = \text{diag}(l_1, l_2, \cdots, l_n) \tag{5.30}$$
$$\boldsymbol{H} = \text{diag}(h_1, h_2, \cdots, h_n) \tag{5.31}$$

where the diagonal terms are larger or equal to a positive, real number $l_0$ and $h_0$, respectively.

The sliding-mode controller is then chosen to be

$$\boldsymbol{\tau_0} = -\boldsymbol{L}\boldsymbol{z} - \sigma \boldsymbol{H}\boldsymbol{\psi}(\boldsymbol{z}) \tag{5.32}$$

This means that the derivative of V becomes

$$\dot{V} = 2\boldsymbol{z}^T \left[-\boldsymbol{K}\boldsymbol{L}\boldsymbol{z} - \sigma \boldsymbol{K}\boldsymbol{H}\boldsymbol{\psi} + \boldsymbol{\phi} + \boldsymbol{\delta} - \boldsymbol{C}\boldsymbol{\alpha_1}\right] \tag{5.33}$$

Since all terms in $\boldsymbol{K}$ and $\boldsymbol{L}$ are assumed to be larger than $k_0$ and $l_0$, respectively, it is easy to see that the first term in (5.33) becomes

$$-2\boldsymbol{z}^T \boldsymbol{K}\boldsymbol{L}\boldsymbol{z} \leq -2l_0 k_0 \left\|\boldsymbol{z}\right\|^2 \tag{5.34}$$

The second term in (5.33) satisfy

$$\begin{aligned} -2\sigma \boldsymbol{z}^T \boldsymbol{K}\boldsymbol{H}\boldsymbol{\psi} &\leq -2\sigma k_0 h_0 \left\|\boldsymbol{z}\right\| \left\|\boldsymbol{\psi}\right\| && (5.35) \\ &\leq -2\sigma k_0 h_0 \frac{1}{\sqrt{n}} \left\|\boldsymbol{z}\right\| && (5.36) \\ &\forall \left\|\boldsymbol{z}\right\| \geq \sqrt{n}\epsilon \end{aligned}$$

when $\epsilon = \epsilon_1 \tanh^{-1}(\frac{1}{1+\epsilon_2})$.

The first equivalence, (5.35), is a result of (5.37), where the $\boldsymbol{KH}$ matrix must be positive definite and diagonal (Skjetne [26]). This is certainly the case here, since both $\boldsymbol{K}$ and $\boldsymbol{H}$ are diagonal and only contains positive numbers. Also, the expression only holds if the signs of $\boldsymbol{z}$ and $\boldsymbol{\psi}$ are equal, but since $\boldsymbol{\psi}$ denotes the signs of $\boldsymbol{z}$, this is not an issue for us.

$$\boldsymbol{z}^T \boldsymbol{K}\boldsymbol{H}\boldsymbol{\psi} \geq k_o h_0 \left\|\boldsymbol{z}\right\| \left\|\boldsymbol{\psi}\right\| \tag{5.37}$$

Equation (5.36) can be shown by the use of Lemma 4.5 in Skjetne [26]:

$$\frac{1}{\sqrt{n}} \left\|\boldsymbol{z}\right\| \leq \boldsymbol{z}^T \boldsymbol{\psi}(\boldsymbol{z}) \leq \left\|\boldsymbol{s}\right\| \left\|\boldsymbol{\psi}(\boldsymbol{z})\right\| \tag{5.38}$$

This is only true when $\left\|\boldsymbol{z}\right\| \geq \sqrt{n}\epsilon$.

To prove this, one have to use the equivalence between norm given in (5.39) and (5.40)

$$\|\boldsymbol{x}\|_2 \leq \|\boldsymbol{x}\|_1 \leq \sqrt{n}\,\|\boldsymbol{x}\|_2 \tag{5.39}$$
$$\|\boldsymbol{x}\|_\infty \leq \|\boldsymbol{x}\|_2 \leq \sqrt{n}\,\|\boldsymbol{x}\|_\infty \tag{5.40}$$

Based on the requirement that the expression only holds when $\|\boldsymbol{z}\| \geq \sqrt{n}\epsilon$, we have

$$\|\boldsymbol{z}\|_2 \geq \sqrt{n}\epsilon \tag{5.41}$$
$$\Downarrow$$
$$\|\boldsymbol{z}\|_\infty \geq \epsilon \tag{5.42}$$

Then, if $z_i$ represents the largest value in $\boldsymbol{z}$, (5.40) gives

$$\|\boldsymbol{z}\|_\infty = |z_i| \tag{5.43}$$
$$\Downarrow$$
$$\boldsymbol{z}^T\boldsymbol{\psi}(z) = z_1\psi(z_1) + z_2\psi(z_2) + \cdots + z_n\psi(z_n) \tag{5.44}$$
$$\geq \|\boldsymbol{z}\|_\infty \geq \frac{1}{\sqrt{n}}\|\boldsymbol{z}\|_2 \tag{5.45}$$

In sum, these derivations shows that

$$\begin{aligned}\dot{V} \leq &- 2l_0k_0\|\boldsymbol{z}\|^2 - \frac{2\sigma}{\sqrt{n}}k_0h_0\|\boldsymbol{z}\| \\ &+ 2\|\boldsymbol{z}\|\,\|\boldsymbol{\phi} - \boldsymbol{C\alpha}\| + 2\|\boldsymbol{z}\|\,\|\boldsymbol{\delta}\|\end{aligned} \tag{5.46}$$

$$\leq -2\|\boldsymbol{z}\|\left\{\frac{k_0h_0}{\sqrt{n}}\sigma - \|\boldsymbol{\phi} - \boldsymbol{C\alpha}\| - \rho\right\} \tag{5.47}$$

Now it is desirable to design $\sigma$ such that $\boldsymbol{z}$ goes to zero in finite time and stays there. This is done by choosing $\sigma$ as

$$\sigma = \frac{\sqrt{n}}{k_0h_0}\{P + \|\boldsymbol{\phi} - \boldsymbol{C\alpha}\| + \rho\} \tag{5.48}$$

where P is a positive constant.

The resulting, and final, $\dot{V}$ is then

$$\dot{V} \leq -2P \|\boldsymbol{z}\| \qquad \forall \|\boldsymbol{z}\| \geq \epsilon \tag{5.49}$$

Hence, by Barbalat's lemma (Khalil [11]), $\boldsymbol{z}$ converges to zero in finite time and stays there.

### 5.3.4 Chattering

In practical applications of sliding-mode control, one may experience an undesired phenomenon called chattering. This is caused by limitations and delays in the control devices when the control algorithm is implemented digitally. Figure 5.3 shows an example on how these delays can cause chattering. In an ideal world, the trajectory of the system should start sliding on the sliding surface as soon as the trajectory reaches the manifold. Unfortunately, this is not the case in the real world. Due to the finite sampling rate of digital controllers and the fact that the control is constant within a sampling interval, the sign of z changes before the controller switches, Khalil [11]. During this delay, the trajectory of the system crosses the sliding surface into the $z < 0$ area. As the trajectory moves back towards the sliding surface, the same procedure occurs over and over again.

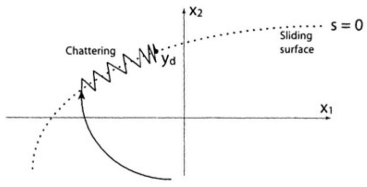

Figure 5.3: An illustration of the scattering phenomena

Another possible reason for this phenomenon can, according to Utkin and Lee [33], be that fast dynamics of the system was neglected in the ideal model when designing the controller.

Chattering is damaging because it results in low control accuracy, high wear and tear of mechanical parts, and high heat losses in electrical power circuits.

There exist methods that will reduce or possibly remove the chattering phenomenon. For instance, Khalil [11] presents a method of dividing the control into continuous and switching components. The general idea is to reduce the amplitude of the switching component and thereby reducing the switching. Another way of eliminating the chattering problem is to use an alternative signum function, such as a saturation function, given in (5.51), where $\epsilon$ denotes the slope. This will give a smoother transition between the signs of z and thereby remove the problem. In this paper, the hyperbolic tangent function given in (5.52) is used for this purpose. Here

$1/\epsilon_2$ denotes the slope of the curve through the origin, while $\epsilon_1$ ensures that the function exceeds the signum function when $\|\boldsymbol{z}\| \geq \epsilon_1 \tanh^{-1}(\frac{1}{1+\epsilon_2}) = \epsilon$. Both $\epsilon_1$ and $\epsilon_2$ are small positive numbers chosen by design.

In Figure 5.4, the conventional signum function is shown as a red line, the high-slope saturation function as a green line, and the hyperbolic tangent function, as a blue line. One can easily see that the hyperbolic function exceeds the signum function and that this function makes the transitions smoother.

$$\psi = \operatorname{sgn}(z) \tag{5.50}$$

$$\psi = \mathrm{SAT}\left(\frac{z}{\epsilon}\right) \tag{5.51}$$

$$\psi = (1+\epsilon_1)\tanh\left(\frac{z}{\epsilon_2}\right) \tag{5.52}$$

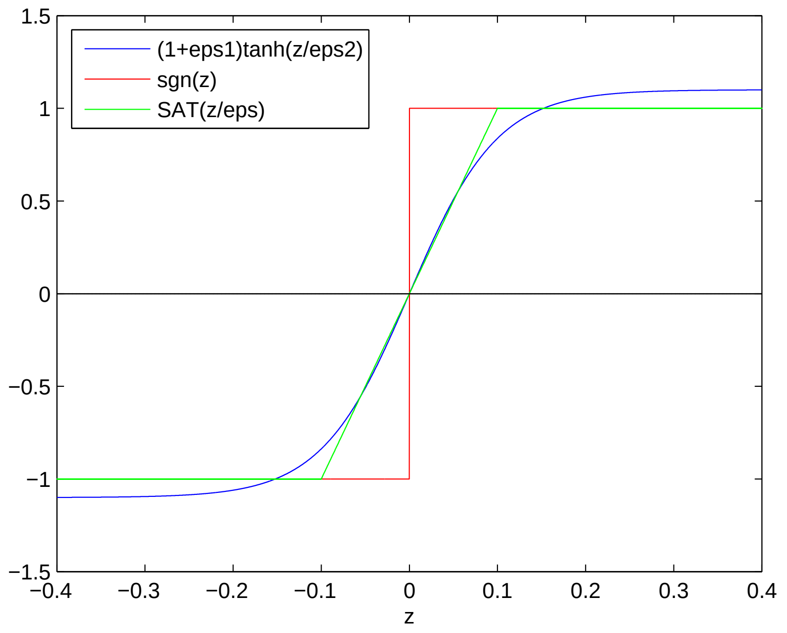

Figure 5.4: Plot of alternatives to avoid chattering.

# Chapter 6

# Communication

Data communication is the transfer of data or information between a source and a receiver. There exist a wide number of different data transfer methods, all depending on the transmission media, mode, protocols, and so on. As described in Chapter 2, all telerobotic systems depends on some sort of communication between the remote site and the local site. Therefore, data communication is considered to be a very important aspect of telerobotics.

According to the Raptor System Manual [30], the local site and the remote site communicates through a half-duplex RS-485 serial link. These terms will all be explained during this chapter. Also, since the system uses digital communication, a brief introduction to the binary numeral system will be given.

## 6.1 Binary Numeral System

In computer technology, data is represented by a collection of on and off statements, or switches (Kjos et al. [13]). This representation is convenient because it is easier to design electronic systems when dealing with only two states. Numerically, the states of these switches are represented by the digits 0 and 1, respectively, and since there are two different states they are said to be binary. Further, a single memory location in a typical computer is comprised of eight of these switches, or bits, and is referred to as a byte of data. Since each bit contains two states, a byte has $2^8 = 256$ possible states. Each of these states corresponds to an integer between 0 and 255 inclusive.

Numbers that are used in everyday life are written in what is known as decimal notation. Each successive digit in a decimal format corresponds to the next higher power of ten. For instance, the number 386 can be written as

$$\begin{aligned} 386 &= (3 \cdot 100) + (8 \cdot 10) + (6 \cdot 1) \\ &= (3 \cdot 10^2) + (8 \cdot 10^1) + (6 \cdot 10^0) \end{aligned} \tag{6.1}$$

Similarly, each successive digit in a binary format corresponds to the next higher power of two, as shown in (6.2).

$$\begin{aligned} 11001 &= (1 \cdot 2^4) + (1 \cdot 2^3) + (0 \cdot 2^2) + (0 \cdot 2^1) + (1 \cdot 2^0) \\ &= 16 + 8 + 0 + 0 + 1 = 25 \end{aligned} \tag{6.2}$$

Likewise, the byte which consists of all ones, i.e. the largest numbers that can be represented in one byte, represents the number 255.

$$\begin{aligned} 11111111 &= 2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 \\ &= 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255 \end{aligned} \tag{6.3}$$

## 6.2 Hexadecimals

Since it is long and cumbersome to write eight digits in a byte, computer scientists prefer a more economical representation. Breaking a byte into two, gives two four bits sets. These sets are referred to as nibbles. A nibble, or four bits, can take 16 distinct configurations. Each of these 16 configurations can be represented by a single digit character as given in Table 6.1. Since there are 16 different configurations, this representation is called hexadecimal.

| | |
|---|---|
| 0000 - 0 | 1000 - 8 |
| 0001 - 1 | 1001 - 9 |
| 0010 - 2 | 1010 - A |
| 0011 - 3 | 1011 - B |
| 0100 - 4 | 1100 - C |
| 0101 - 5 | 1101 - D |
| 0110 - 6 | 1110 - E |
| 0111 - 7 | 1111 - F |

Table 6.1: Hexadecimal representation of nibbles

With hexadecimal, a byte can be represented with two digits instead of eight, like shown in (6.4).

$$\underbrace{0\,1\,0\,0}_{4}\ \underbrace{1\,1\,0\,1}_{D} \tag{6.4}$$

In the same way as shown for the decimals and binary numbers, the value of a hexadecimal number can be represented by multiples of the power of 16. Therefore,

$$4D = (4 \cdot 16^1) + (13 \cdot 16^0) = 77 \tag{6.5}$$

Finally, it is worth noting that for programming purposes the code 0x is propound to hexadecimal numbers. This is just a way to distinguish between hexadecimals and base 10 decimals.

## 6.3 Serial Communication

The concept of serial communication is fairly simple. Data consisting of multiple bits are divided into single bits and then transmitted sequentially over a communication channel (Zheng and Gallager [36]). The individual bits are then reassembled at the receiving end to form the original data. As seen in Figure 6.1, this stands in contrast to the concept of parallel communication, where a group of bits are sent simultaneously through several channels. Although, serial transmission is slower than parallel transmission, it is simpler to use and can be used for long distance communication because it reduces noise.

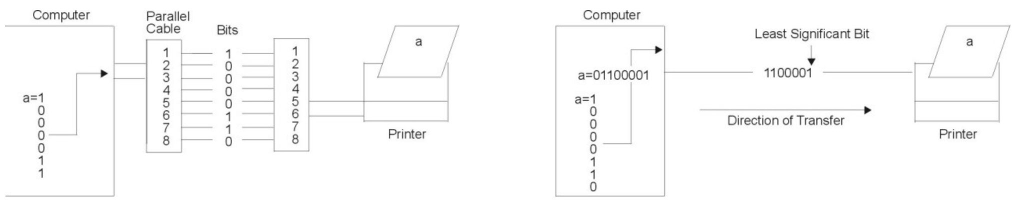

Figure 6.1: Comparison of parallel (left) and serial communication (right)

Serialized data are generally not sent through the communication channels at a uniform rate. Usually a group of regularly spaced data bits are followed by a pause indicating the end of the data packet. Small packets of variable length are sent in these manners until the message is fully transmitted. However, if the data packets are to be combined into the original message, one need to known exactly when each packet begins and how much time elapses between each bit. If this information is known, the receiver is said to be synchronized with the transmitter. A basic technique to ensure this synchronization is described in section 6.4.

The Raptor manipulator system uses a well known serial communication architecture called RS-485. This and the technology behind it will be explained in section 6.5 and section 6.6.

## 6.4 Encoding

Encoding is the process of organizing a sequence of letters, numbers, and punctuations into a specialized format for efficient transmission or storage. Decoding is the opposite process, namely to convert the specialized format back into the original sequence. A simple and nearby example of encoding is the human way of translating oral language into a written sequence of letters.

According to TeleRobotics [29], the RFFM uses Bi-Phase Mark (BPM), or Frequency Modulation 1 (FM-1), encoding. This encoding combines data with clock signals to form a single self-synchronizing data stream. Also, instead of encoding the signal directly, the BPM encoding uses a technique called differential coding. This technique utilizes the presence or absence of transitions to indicate logical values. In order to include clock information, the symbol rate of this encoding is twice the data rate. In other words, there are two transmission bits for every data bit. The encoding is said to be inefficient, but yet well suited for long distance applications.

As Figure 6.2 shows, every data bit interval begins with a clock edge. When the data bit to be encoded is a one, a transition happens during the bit interval. Likewise, the absence of a transition during the bit interval indicates an encoded zero value.

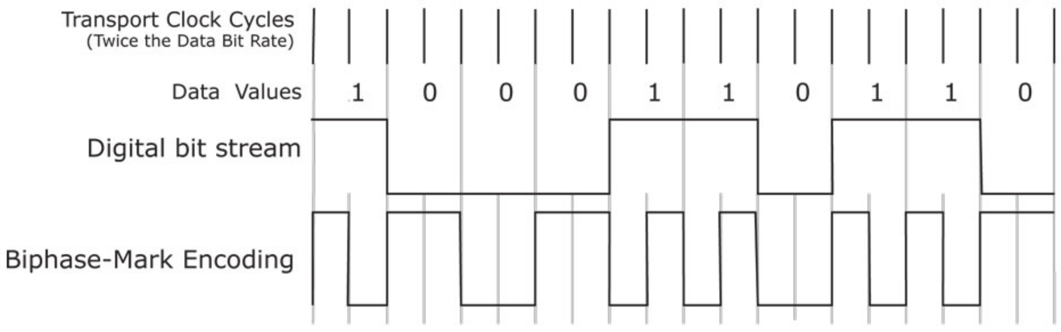

Figure 6.2: Example of Bi-Phase Mark encoding

## 6.5 Differential Signaling

Electronic data communication between devices will often fall into two different categories, namely single-ended or differential. Single-ended signaling is the simplest and most common method for transmitting signals through conductors. Here, the signal is represented by a varying voltage carried through on wire. A differential signal, on the other hand, is one that represents a value as the difference between two physical quantities (Brooks [2]). Practically, this means that the signal is carried on two conductors, and the signal value is the difference between the individual voltage on each conductor. The two conductors are labeled $V+$ and $V-$ respectively. When $V+ > V-$, the resulting signal is defined as positive, while when $V+ < V-$, the resulting signal is defined to be negative. In Figure 6.3, two sinusoidal signals are oscillating about an average voltage of 2V. Offsetting the signal pair in this way provides the maximum range of signal swing when the individual members of the

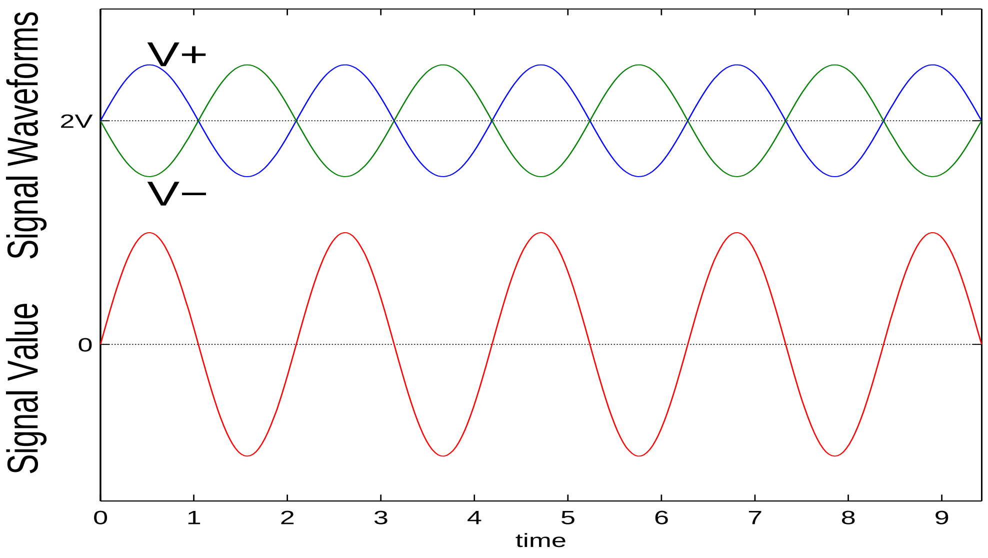

Figure 6.3: Basic idea of differential signal

pair are limited to a 0 − 4V excursion. The resulting differential signal is given by the red graph.

When moving from a single-ended to a differential signaling scheme, one replaces a single wire with a pair of wires, and thereby doubles the complexity of the associated interface circuitry. However, there are several benefits of differential signaling that outweighs the increased complexity. First, all voltages are measured with respect to another voltage, for instance with respect to the ground voltage. If there is a difference between the reference voltage in the transmitter and in the receiver, this will change a signal carried on one conductor. The further apart the signal source and the signal receiver are, the greater the likelihood of different reference signal. A differential signal will not be affected by this phenomena since the difference between the two signals will still be the same.

A second major advantage of differential signaling is that it is highly immune to outside electromagnetic interference (EMI) and crosstalk from nearby signal conductors. An interference will affect both conductors equally, like shown in Figure 6.4, if they are routed close enough. This means that the resulting differential signal not will be affected. Also, differential signals tend to produce less EMI than single-ended signals. This is because the two conductors create opposing electromagnetic fields that often cancel each other out.

In the electronics industry there is an increasing tendency to lower the supply voltage in order to save power and reduce unwanted emitted EMI. Normally, low supply voltage causes problems because it reduces noise immunity. However, due to the increased tolerance to EMI and noise, differential signaling is also suitable for low supply voltages.

In summary, when communicating at high data rates, or over long distances, or using low supply voltage, differential signaling is preferred in most cases.

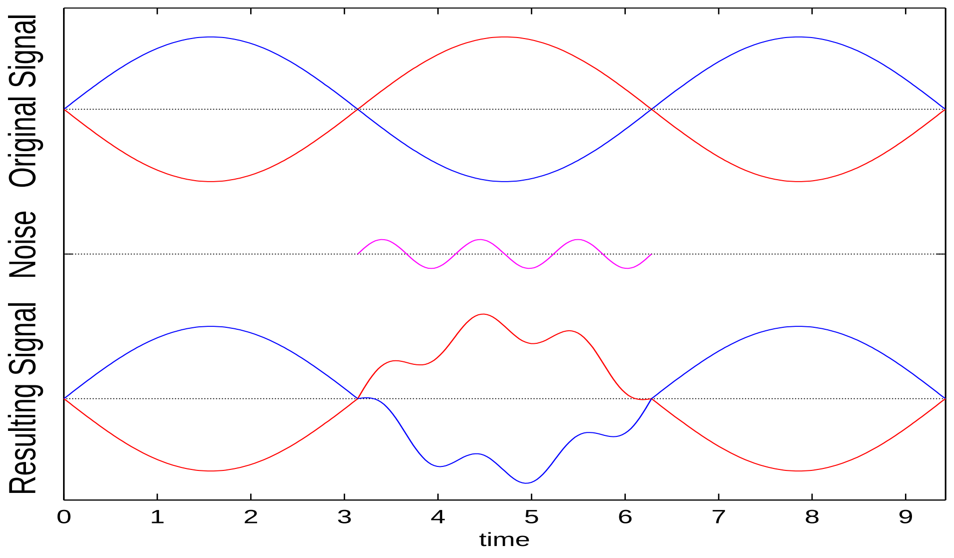

Figure 6.4: Effect of noise on differential signal

## 6.6 RS-485

In telecommunication, RS-485 is a standard for binary serial communications between devices. This standard, and several others, have been developed to insure compatibility between devices provided by different manufacturers. The standard is published by the Telecommunications Industry Association (TIA) [31] and is an updated version of the well known and common RS-232 standard. The latter is used in several devices on NTNU's two ROVs, Minerva and SF 30K. RS-232 is a single-ended standard that only allows the connection of two devices through a serial link. RS-485, on the other side, uses differential signaling, which means that this standard allows communication over longer distances than achievable with RS-232. Also, as mentioned, the nature of differential signaling makes the RS-485 standard less exposed to noise. It is also worth noting that the RS-485 interface supports 32 driver/receiver pairs in a so called multi-drop mode.

These standards states, among others, what types of connectors must be used and which pins in those connectors must be used for each function. In Table 6.2 other characteristics of the RS-485 communication standard are given.

| | |
|---|---|
| Differential | Yes |
| Max number of drivers | 32 |
| Max number of receivers | 32 |
| Network topology | Multipoint |
| Max distance | 1200m |
| Max speed at 12m | 35Mbs |
| Max speed at 1200m | 100kbs |
| Receiver input resistance | $\geq 12k\Omega$ |
| Driver load impedance | $54\Omega$ |
| Receiver input sensitivity | $\pm 200mV$ |
| Receiver input range | $-7 \ldots 12V$ |
| Max driver output voltage | $-7 \ldots 12V$ |
| Min driver output voltage | $\pm 1.5V$ |

Table 6.2: Characteristics of RS-485

# Chapter 7

# Simulations and Results

The dynamic model, sliding-mode controller, and inverse kinematics algorithms derived previously will now be verified through simulations. First, the manipulator is simulated when no control forces are applied. The manipulator are then expected to collapse under gravity and behave like a multi joint pendulum. Second, two different IK algorithms are put to the test by using a joystick simulator. The full system, including sliding-mode controller, is then simulated to verify the controller and tune the control parameters. Finally, the basics of the real-time implementation in LabVIEW is presented.

## 7.1 Simulation Structure

The model, including controller, is implemented in MATLAB/Simulink and uses several 'Level-2 MATLAB S-Functions' to incorporate the dynamics. These functions makes it possible to use basic MATLAB language to create custom blocks with multiple input and output ports. For every time step of the simulation, the 'Level-2 MATLAB S-Functions' calculates the proper output for the given input, thus the simulation might be a bit slow. The desired measurements are sent from Simulink to the MATLAB interface through a 'To Workspace' block. This makes it possible to present the results graphically.

## 7.2 Model Verification

A simulation of the manipulator model when no control forces are applied will give an idea on whether the model is correct or not. A manipulator model with no dissipative forces will give a chaotic response. Therefore, the friction force introduced in section 5.3.2 is included. The initial conditions for the simulation are

$$\boldsymbol{q}_{init} = \left[0, 0, 0, 0, -\tfrac{\pi}{2}, 0\right]^T \tag{7.1}$$
$$\dot{\boldsymbol{q}}_{init} = \left[0, 0, 0, 0, 0, 0\right]^T \tag{7.2}$$

According to the D-H convention, this implies that the robot manipulator starts in the horizontal position shown in Figure 7.1. This visualization is made using the 'Robotics Toolbox' of Corke [5].

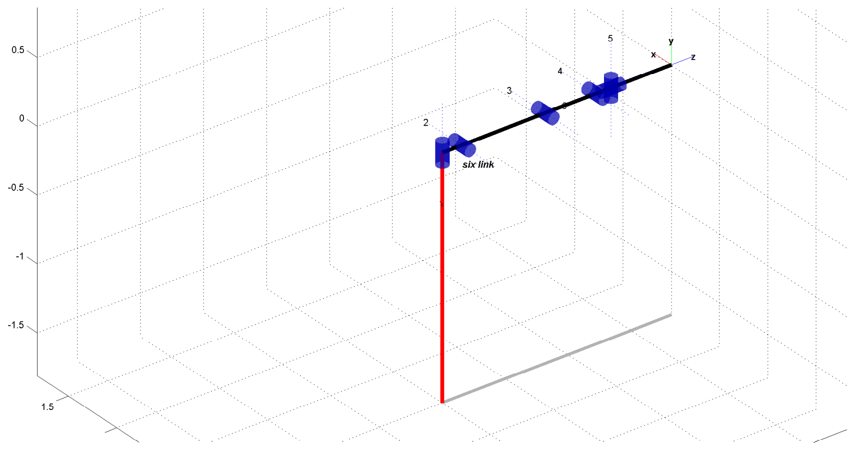

Figure 7.1: Initial configuration of robot manipulator

The response of this simulation is given in Figure 7.3 and 7.4. As expected, the manipulator will act like a multi joint pendulum. While the gravity compels the manipulator to accelerate from the initial position, the friction force dissipates the energy, leading the manipulator to settle at equilibrium position. Due to the geometry of the manipulator, this means that the five outer links will settle down in a vertical position. Since the first joint only allows rotations about the z-axis, the first link will settle at a horizontal position as shown in Figure 7.2. Since the given response matches the expected behavior, it is assumed that this model is correct.

## 7.3 Trajectory Generation

Normally the Raptor manipulator system is controlled by an advanced master controller with force feedback capabilities. This thesis, on the other hand, has chosen to use a joystick and rate control mode in combination with IK algorithms to state the operator commands. As described in Chapter 2, this means that the joystick displacement is proportional to the end-effector velocities. This setup calls for real-time simulations, which Simulink is not well suited for. Therefore, some alternative solutions to generate joint space trajectories are proposed.

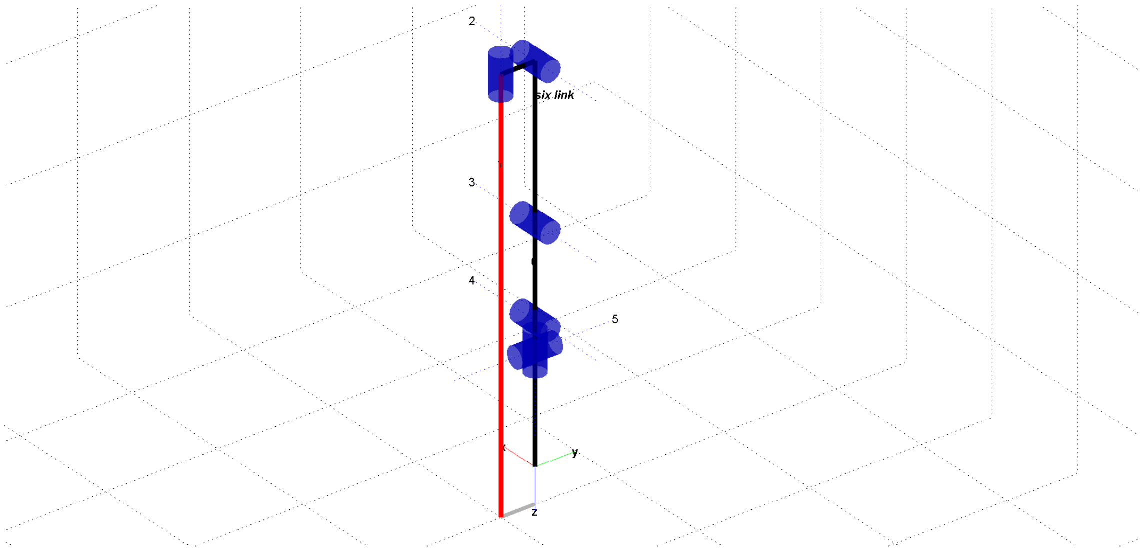

Figure 7.2: Manipulator configuration at equilibrium

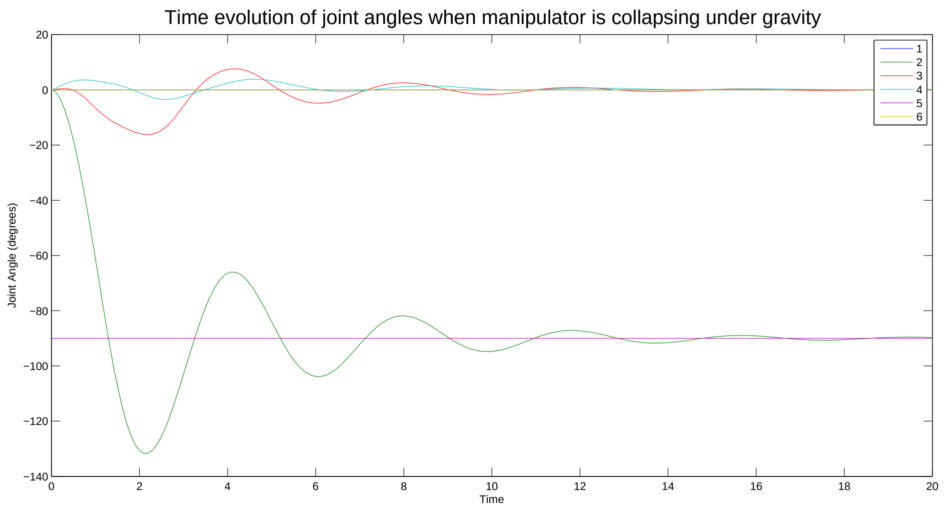

Figure 7.3: Time evolution of joint angles when manipulator is collapsing under gravity

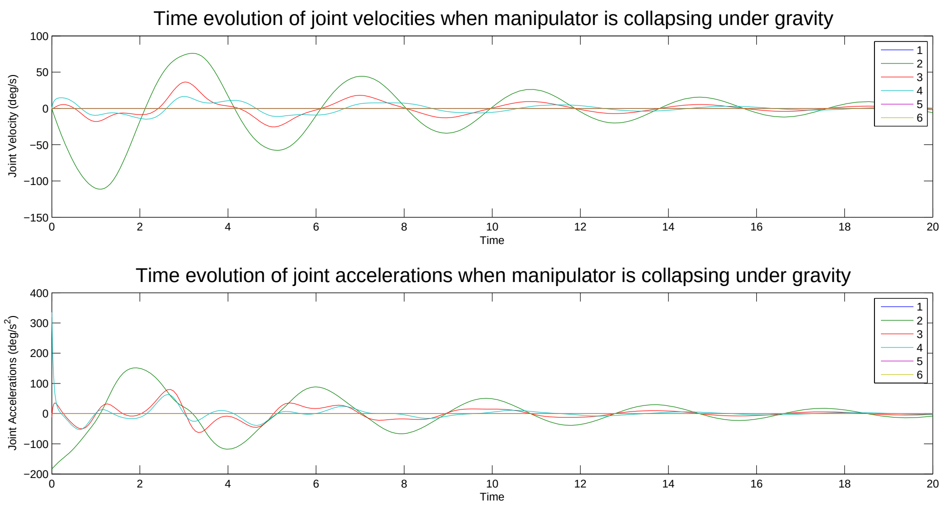

Figure 7.4: Time evolution of joint velocities (top) and accelerations (bottom) when manipulator is collapsing under gravity

In the project thesis [8], the reference signals were generated directly in joint space by a simple reference model adopted from Fossen [7]. The model consisted of a first-order low pass filter cascaded with a mass-damper-spring system. This time, a reference model is used to make a velocity trajectory in the operational space, thus mimicking the behavior of a joystick. The operational space reference is then transformed into joint space by an IK algorithm. The structure of the joystick simulator can be seen in Appendix B, while the simulated joystick signals can be seen in Figure 7.6.

Two different IK algorithms will now be verified through simulations. By integrating the joystick velocity outputs for a given initial position, the desired end-effector position is obtained. The joint space trajectories derived from the chosen IK algorithm are then transformed back to operational space by a forward kinematics algorithm. A comparison of the two operational space trajectories will give an idea on the quality and validity of the IK algorithms. This test setup is shown in Figure 7.5.

First, the DLS solution shown in (4.16) is implemented and tested with $\lambda = 0.01$ and $\lambda = 0.1$. The resulting comparison plots are given in Figure 7.7 and 7.8. Seeing that $\lambda = 0.01$ gives a perfect match, it is clear that low values of $\lambda$ gives more accurate solutions. This is consistent with the theory described in Chapter 4.

Second, the simple method of inverting the Jacobian is tested. The resulting response is given in Figure 7.9. The plot shows that the operational space trajectories are quite inaccurate compared to the results from the DLS method.

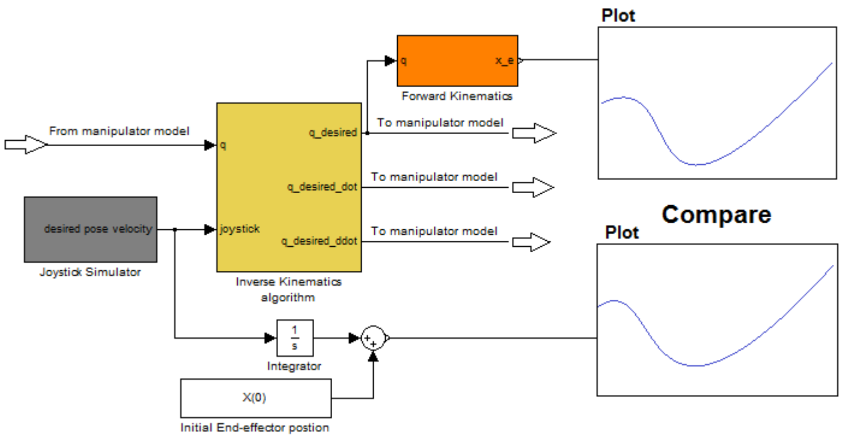

Figure 7.5: A conceptual sketch of the IK algorithm test.

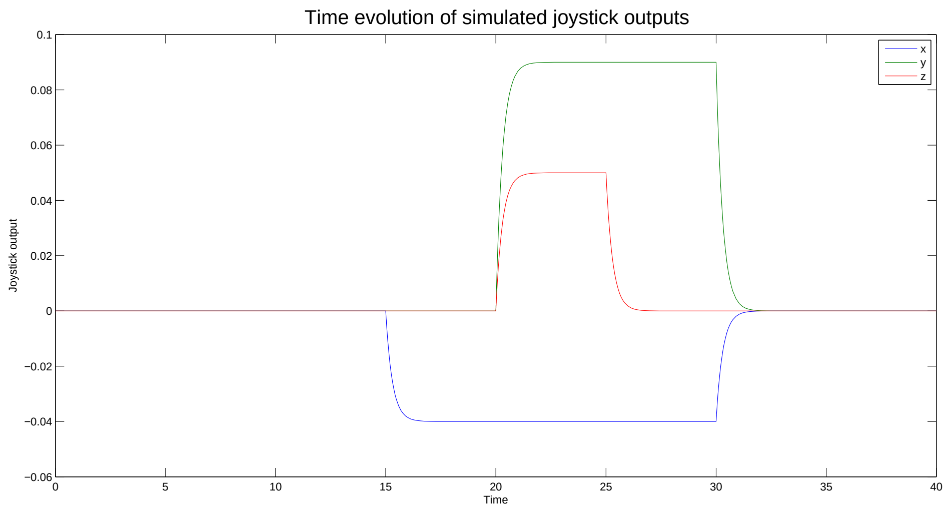

Figure 7.6: Simulated joystick signals.

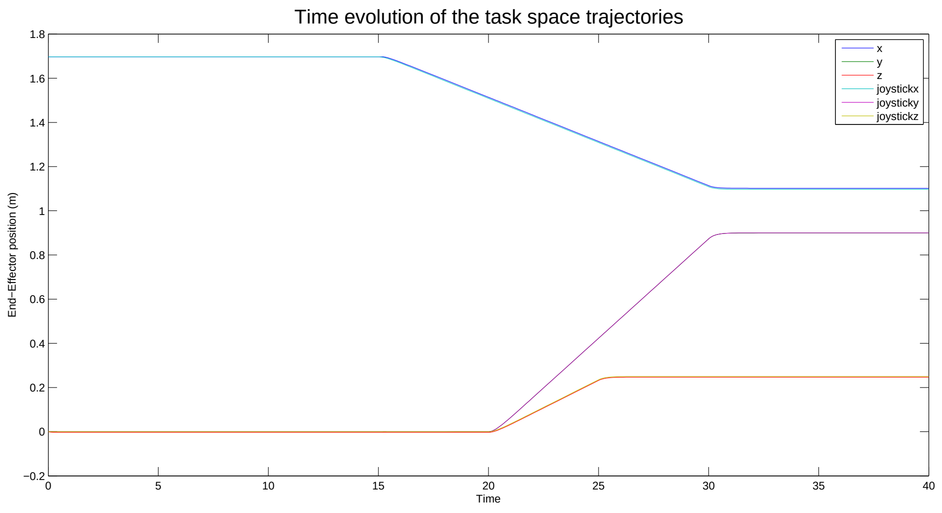

Figure 7.7: End-effector position versus integrated joystick signals, DLS method with $\lambda = 0.01$.

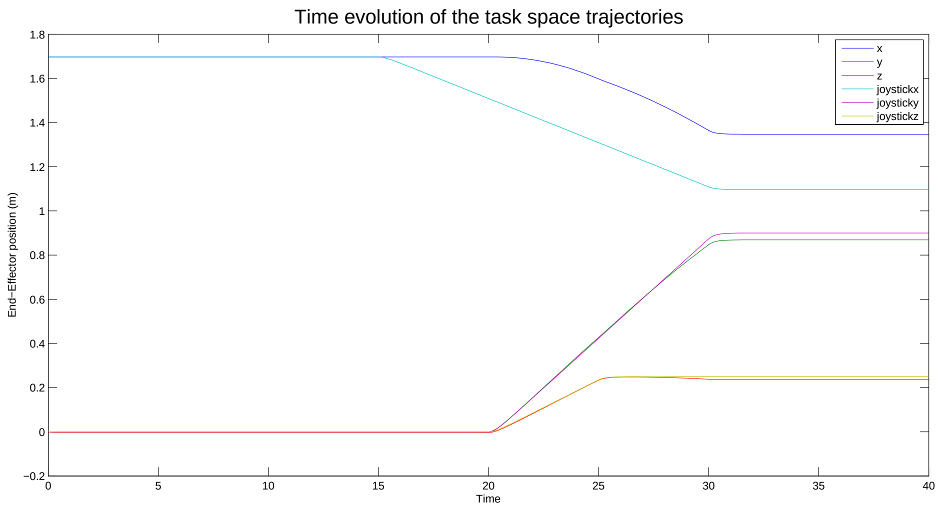

Figure 7.8: End-effector position versus integrated joystick signals, DLS method with $\lambda = 0.1$.

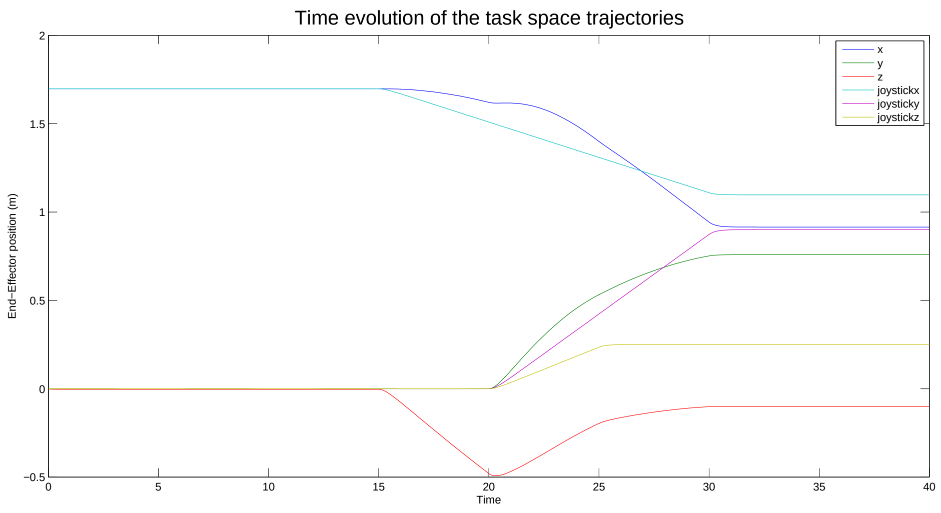

Figure 7.9: End-effector position versus integrated joystick signals, inverse Jacobian method.

## 7.4 Control Design Verification

Now that the model has been assumed correct and the trajectory generation has been explained, the sliding-mode controller will be the subject of investigation. The control system derived in Chapter 5 is therefore implemented in Simulink. A top-level view of the complete Simulink system can be seen in Figure 7.10.

The system will be simulated with the joint trajectories generated from the DLS method with $\lambda = 0.01$. This will move the manipulator from the initial configuration given in (7.1) to the final configuration given in (7.3). The final configuration of the simulation is also shown graphically in Figure 7.11. Controller gains and other parameters relevant for the simulation are given below.

$$\boldsymbol{q}_{final} = \left[0.8805, 0.6590, -1.0639, 0.4013, -0.6905, 0.0028\right]^T \tag{7.3}$$

The derivations in Chapter 5 shows that the dynamics on the sliding-surface is governed by

$$\dot{\boldsymbol{e}}_{\mathbf{1}} = -\boldsymbol{A}\boldsymbol{e}_1 \tag{7.4}$$

Hence, if the $\boldsymbol{A}$ matrix is positive definite, the dynamics will be asymptotically stable. Therefore, $\boldsymbol{A}$ is chosen as

$$\boldsymbol{A} = \mathrm{diag}(10, 10, 10, 10, 10, 10) \tag{7.5}$$

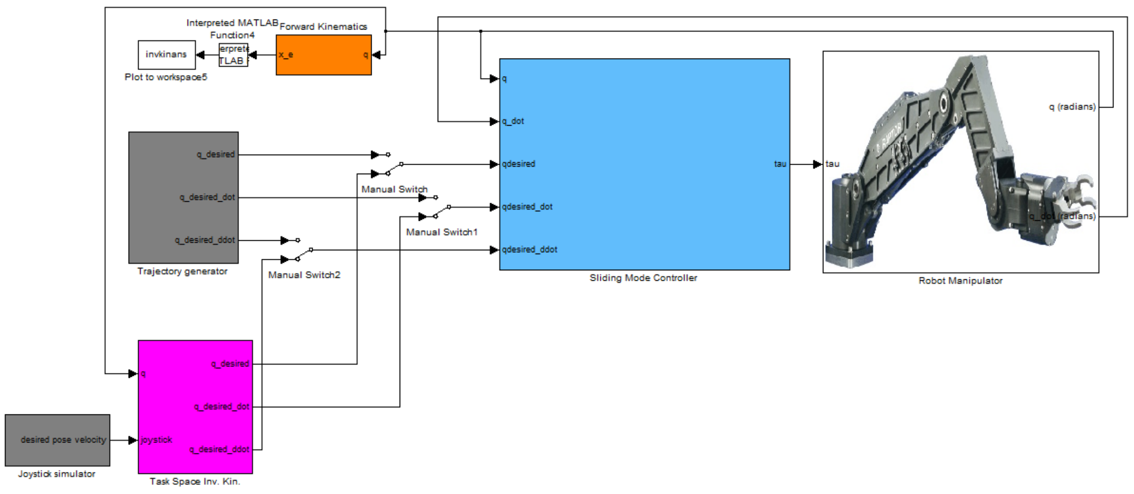

Figure 7.10: Complete Simulink diagram, main view

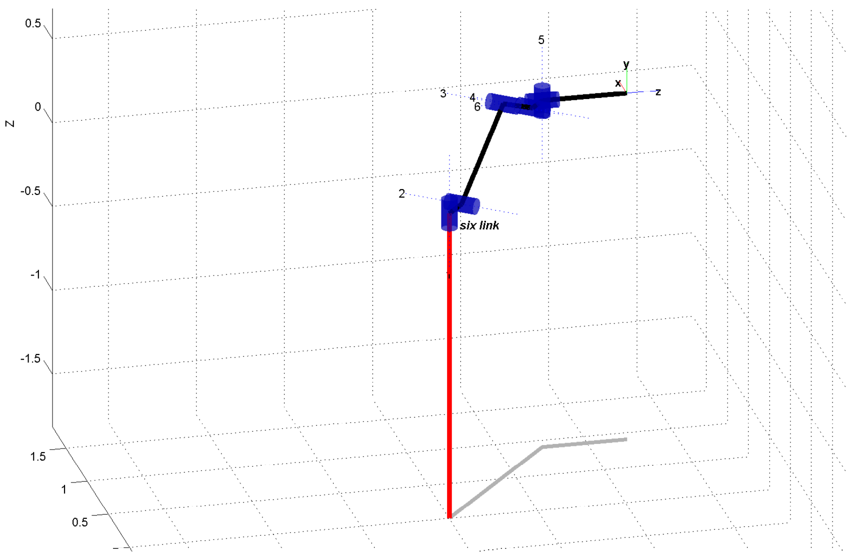

Figure 7.11: The final robot configuration

The unknown disturbance $\delta$ is simply modeled as a sine wave with amplitude of 0.1 [rad] acting on all joints. This disturbance is assumed to be bounded by

$$\|\boldsymbol{\delta}\| \leq \rho \tag{7.6}$$

Therefore, $\rho = 1$ will be true to the assumptions.

The friction force is said to be proportional with the joint velocity. There are no given information of the magnitude of this friction force, but normally the friction in joints are quite low. In this simulation, the friction force is modeled as

$$\boldsymbol{F}(\dot{\boldsymbol{q}}) = \text{diag}(50, 50, 50, 50, 50, 50)\dot{\boldsymbol{q}} \tag{7.7}$$

Other relevant control parameters are given beneath

$$\boldsymbol{H} = \text{diag}(2, 2, 2, 2, 2, 2) \tag{7.8}$$
$$\boldsymbol{L} = \text{diag}(2, 2, 2, 2, 2, 2) \tag{7.9}$$
$$h_0 = l_0 = 1.5 \tag{7.10}$$
$$k_0 = 0.5 \tag{7.11}$$
$$P = 10 \tag{7.12}$$
$$\boldsymbol{\epsilon}_1 = \boldsymbol{\epsilon}_2 = [0.1, 0.1, 0.1, 0.1, 0.1, 0.1]^T \tag{7.13}$$
$$\boldsymbol{K} = \text{diag}(0.8, 0.8, 0.8, 0.8, 0.8, 0.8) \tag{7.14}$$

Figure 7.12 shows that the manipulator is able to follow the desired joint angles in a good manner. There are also very small discrepancies between the desired joint velocities and the real joint velocities, as shown in Figure 7.13. The sliding-mode controller is therefore assumed to be correct and nicely tuned.

## 7.5 Full-Scale Testing

The final goal of this thesis is to conduct full-scale testing of the control system on the Raptor manipulator. To do so, the control system must be transfered to a programming language that is better suited for real-time control and hardware interfacing than Simulink. Also, an interface between this software and the manipulator software must be made. The chosen programming language is called LabVIEW. All generated code can be found in Appendix A.

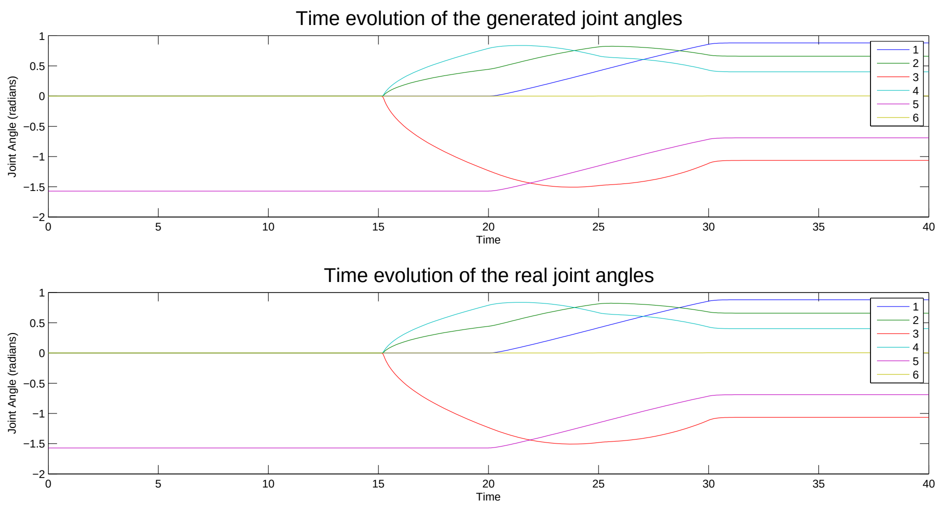

Figure 7.12: Time evolution of generated joint angles (top) and real joint angles (bottom)

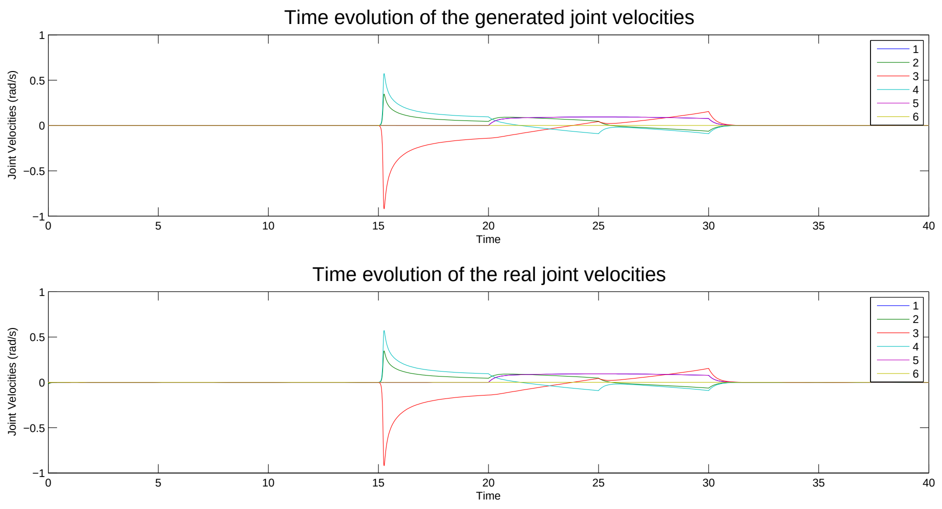

Figure 7.13: Time evolution of generated joint velocities (top) and real joint velocities (bottom)

### Control Implementation

The control system is implemented in LabVIEW by the use of 'MathScript Node' structures. These structures allows one to use MATLAB scripts within the LabVIEW environment and thereby simplifies the transformation from MATLAB. The structural setup of the controller is the same in both programs. The top level VI for the controller is called 'Sliding Mode Controller.vi' and can be seen in Appendix A.

### Trajectory Generation

The LabVIEW program uses the DLS algorithm presented in Section 7.3 to convert operational space trajectories into corresponding joint space trajectories. This time there is no need to simulate the joystick output. Instead, the simple Logitech gamepad shown in Figure 7.14 serves as a master joystick. The three translational velocities of the end-effector are commanded by movement of the the red and green marked joysticks. The three rotational velocities are commanded by moving the same joysticks while pressing the blue marked button. Since the joystick is very sensitive to small movements, a dead zone algorithm is implemented. The algorithm, shown in Figure 7.15, suppresses all readings between $-10$ and 10 by setting these to zero. Outside the dead zone region, a smooth transition will be provided by the functions given in (7.15). The discussed actions are performed by the VIs 'Joystick.vi' and 'InvKin.vi'. These can be found in Appendix A.

$$
\begin{aligned}
y &= x - 10\left[1 - \tanh\left(\frac{x-10}{2}\right)\right], \qquad && x > 10 \\
y &= x + 10\left[1 + \tanh\left(\frac{x+10}{2}\right)\right], \qquad && x < 10
\end{aligned}
\tag{7.15}
$$

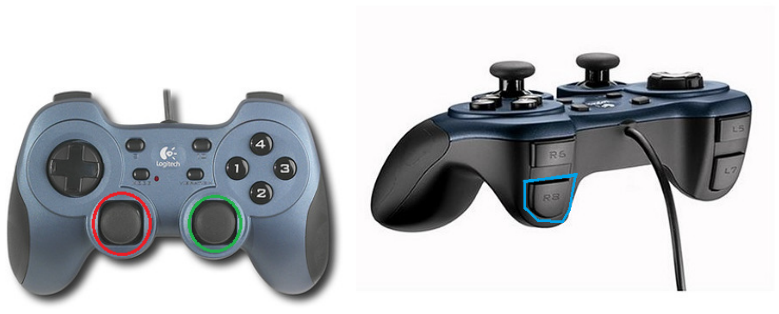

Figure 7.14: Logitech Gamepad

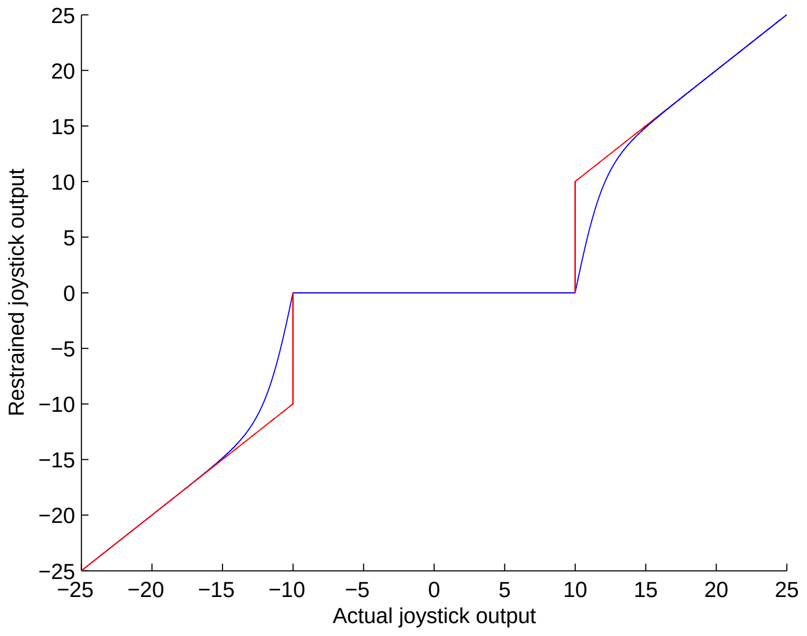

Figure 7.15: Two alternative dead zone solutions.

# Chapter 8

# Discussion

The topic of robot modeling and control is an area of science that has been devoted much time and research. There are a lot of challenges associated with controlling a robot, but there are also a lot of proposed solutions available. This diversity of options makes it hard to make the right decisions and choose the best alternatives. During the making of this thesis, different choices have been made. Even though these choices are results of extensive studies and consultations, they do not necessarily represent the best solutions. This chapter will present a discussion of some of the different choices been made throughout the thesis and the associated consequences.

Also, the lack of experience with both software and robot methodology caused some issues, some of which are solved. These issues will be presented during the following sections.

Finally, an explanation on why no full-scale experiments are performed will be given.

## 8.1 Velocity Measurements

The tracking controller was made under the assumption that the complete state measurements, i.e. both position and velocity measurements of each joint, were available. As the work progressed, it came clear that this was not the case. The Raptor manipulator, as many other manipulators, only provide the user with joint position measurements. This is mostly due to high cost of sensors and that velocity measurements often are contaminated by noise. Although it is possible to make controllers that do not depend on joint velocities, it was decided to use some sort of observer to estimate the velocities. The best practice is to filter the position measurements by a proper high-pass filter with reasonable cutting frequency, e.g. $\xi = \frac{s}{((s/10)+1)^2}$. This will result in a pseudo-velocity signal that serves as a surrogate for velocity measurements. Another alternative is to simply use the difference quotient to calculate the velocity. To reduce the presence of noise, the former velocity calculations may be taken into the equation as

$$v(t) = \kappa \frac{p(t) - p(t-T)}{T} + (1-\kappa)v(t-T) \tag{8.1}$$

where $\kappa < 1$ is a weighting constant and p(t) is the position measurements at time t. None of these solutions are implemented as other issues kept us from completing the full-scale control system.

## 8.2 Lack of Full-Scale Experiments

A full-scale test of the derived control system was not only the final goal of this thesis, but also a great motivational factor. Unfortunately, a full-scale test could not be conducted in the limited time available. The ROV SF 30K had not been used for a long time and most of the equipment, including the Raptor manipulator, needed maintenance. The manipulator was therefore sent abroad to be repaired and controlled. After returning, the manipulator was reinstalled and tested with the existing control system. However, this was very late in the semester and there was no time to log communication data and to create an interface to the manipulator. Hopefully the work done in this thesis can still serve as a basis for future work on the manipulator system.

## 8.3 Modeling and Simulation Issues

During the making of this thesis, the manipulator model presented in the project thesis [8] was proven incorrect. Erroneous assumptions in connection with the D-H convention and the derivations of the Jacobian led to an incorrect response. These issues are now fixed and the model should therefore be correct.

The Raptor manipulator system is both powerful and costly. It is important to verify that the software works as intended before implementing it on the real system. This verification can be done by a real-time HIL-simulator, implemented in LabVIEW by a 'Control & Simulation Loop'. However, this requires the use of a fixed-step ODE solver. Simulations, on the other hand, have shown that the system is best suited for variable-step solvers. If fixed-step solvers are used, the time step of the simulation must be very small. This, combined with the fact that each iteration takes a fair amount of time, makes real-time simulations unlikely. Figure 8.1 shows the length of the time steps during a variable step size simulation of the manipulator. It is clear that the system requires very small time steps in the beginning of the simulation. Whether this is caused by the initial conditions of the simulation or something else is not known. This issue might indicate that the derived model is stiff, even though it is not expected to be.

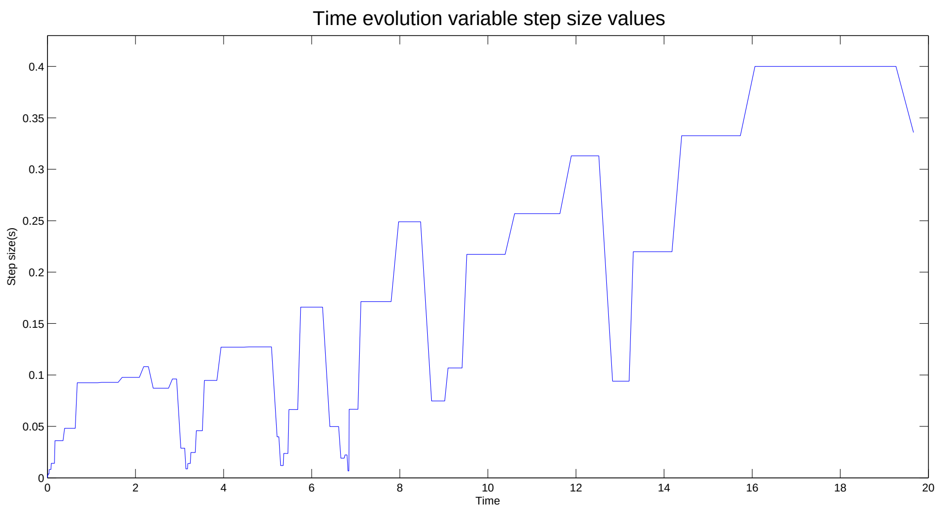

Figure 8.1: Length of time steps during simulation of manipulator model.

# Chapter 9

# Conclusion and Further Work

## 9.1 Conclusion

The main purpose of this thesis have been to investigate and select suitable approaches for dynamic modeling, simulation, and control of the Raptor manipulator. The final goal was to develop a working control system for the full-scale manipulator, thus providing an alternative to the existing, commercial control system. This included the development of an interface between the created and the existing software.

It has been shown that the field of robot modeling and control is wide and comprehensive. In combination with little experience, this makes it hard to choose the most suitable solutions and procedures. Nevertheless, different approaches have been chosen, used, and accounted for. Most of the topics of interest have been thoroughly explained either in this thesis or in the project thesis [8].

The well known topic of inverse kinematics have been devoted much time and attention. Due to the large extent of the subject, the focus was limited to algorithms based on velocity kinematics. Several alternative solutions was introduced, two of which were tested and verified. The DLS method proved superior to the Jacobian inverse method and gave most accurate results when $\lambda$ was small.

The available information about the Raptor manipulator was very limited. Therefore, a simplified model based on the kinematic structure of the manipulator was made. The mass and inertia tensor for each link was calculated by rough geometric estimations. These derivations were inaccurate, but with the limited information available this was the only way to go. The identification technique of CAD modeling was mentioned as an alternative approach, but this would require more extensive geometry information.

An entire chapter was also devoted to introduce the properties of the electronic communication system. A basic understanding of the topic can be an advantage when designing the interface system. Unfortunately, there was no time to log communication data and therefore impossible to create the mentioned interface.

The nonlinear control method of choice was the sliding-mode control principle. Simulations and studies showed that it was a robust method that can handle unknown parameters and uncertainties in the model. The final controller relied on quite a few assumptions, some more reasonable than others. The assumptions of a known friction force was probably the most inaccurate one. However, with the limited information available it was assumed to be adequate.

Simulations of the dynamic model and control system gave satisfactory results. When no control forces were applied, the manipulator model was expected to behave like a multi joint pendulum. This assumption proved to be correct, thus the model was also assumed to be correct. A simulation of the complete system, included controller and IK algorithms, also gave some good results. The control algorithm worked well and the robot was able to follow the generated joint trajectories.

Even though the assignment proved to be too ambitious, it is possible that this thesis can serve as the starting point for future work.

## 9.2 Recommendations for Further Work

Further work on this subject should include more accurate parameter estimations. However, this requires more comprehensive information about the geometry and materials of the manipulator. Some information could be obtained by physically measure the geometry of the manipulator. If the geometric properties are available, the parameters could be estimated by the use of CAD analysis. The resulting model will then be closer to the real manipulator.

Also, more extensive and correct models of the actuator dynamics should be implemented. The actuator dynamics have only been assumed to take the form of an uncertain multiplicative term and an additive term. There are several ways of deriving these models, but these are not taken into considerations in this thesis.

The friction forces are assumed to be known, but in reality, there are no indications of the magnitude of these friction forces. It is therefore recommended to investigate alternative methods for friction estimation. Also, it would be interesting to find a model that incorporates drag forces acting on each link, since the ROV is working under water. These might be found in model tests, but this is expensive. Another alternative is to use CFD programs.

As of today, the full-scale control system is programmed in LabVIEW. However, a fair amount of work still remains before the system can be used on the Raptor manipulator. First, the communication data must be logged and processed to develop the interface between the user computer and the manipulator system. The digital data format is given in the Engineering Support Document [28], but it does not specify other than the number of bytes per sensor. Second, the control system must be tested on a HIL-simulator to avoid damage on equipment or humans.

# Bibliography

[1] Jake J. Abbott, Panadda Marayong, and Allison M. Okamura. Haptic virtual fixtures for robot-assisted manipulation. In *12th International Symposium of Robotics Research (ISRR*, pages 49–64, 2005.

[2] Douglas Brooks. Differential signals, rules to live by. *Printed Circuit Design*, 2001.

[3] Samuel R. Buss and Jin-Su Kim. Selectively damped least squares for inverse kinematics. *Journal of Graphics Tools*, 10(3):37–49, 2005.

[4] Ricardo Campa, César Ramíres, Karla Camarillo, Víctor Santibáñez, and Israel Soto. *Advances in Robot Manipulators*, pages 417 – 442. Ernest Hall, 2010.

[5] Peter I. Corke. *Robotics, Vision & Control: Fundamental Algorithms in Matlab.* Springer, 2011. ISBN 978-3-642-20143-1.

[6] Willian R. Ferrell and Thomas B. Sheridan. Supervisory control of remote manipulation. *IEEE Spectrum*, pages 81–88, 1967.

[7] Thor I. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control.* John Wiley & Sons, Inc, 2011.

[8] Morten Sletteberg Haugen. Project thesis: Modeling and control of rov manipulator, 2011.

[9] Sandra Hirche, Manuel Ferre, Jordi Barrio, Claudio Melchiorri, and Martin Buss. Bilateral control architectures for telerobotics. In *Advances in Telerobotics*, volume 31, pages 163–176. Springer Berlin / Heidelberg, 2007.

[10] National Instruments. Labview. `http://www.ni.com/labview/`, 2012. Visited 10.05.2011.

[11] Hassan K. Khalil. *Nonlinear Systems, Third Edition.* Prentice Hall, 2002.

[12] Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal of Robotics and Automation*, RA-3(1):43–53, 1987.

[13] Bård Kjos, Arne B. Mikalsen, Camilla Tepfers, Claude Davidsen, Einar Hestmann, Geir Maribu, Guttorm Sindre, Nils-Christian Haugen, Per Borgesen, and Per Arne Godejord. *Innføring i Informasjonsteknologi.* Tapir Akademiske Forlag, 2003.

[14] Alain Liégeois. Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE Transactions on Systems, Man, and Cybernetics*, 7:868–871, 1977.

[15] A Cybernet Group Company Maplesoft. Maple 15. `http://www.maplesoft.com/products/maple/`, 2012. Visited 10.05.2011.

[16] Karl Mathia. *Robotics for Electronics Manufactering.* Cambridge University Press, 2010.

[17] MathWorks. Matlab. `http://www.mathworks.se/products/matlab/`, 2012. Visited 10.05.2011.

[18] Michael Meredith and Steve Maddock. Using a half-jacobian for real-time inverse kinematics. Technical report, Department of Computer Science, University of Sheffield.

[19] Uwe Mettin. *Principles for Planning and Analyzing Motions of Underactuated Mechanical Systems and Redundant Manipulators.* PhD thesis, Umeå University, 2009.

[20] Nicolas Mollet. *Remote and Telerobotics.* InTech, 2010.

[21] Yoshihiko Nakamura and Hideo Hanafusa. Inverse kinematic solutions with singularity robustness for robot manipulator control. *Journal of Dynamic Systems, Measurement and Control*, 108:163–171, 1986.

[22] Thomas B. Sheridan. *Supervisory Control of Remote Manipulators, Vehicles and Dynamic Processes: Experiments in Command and Display Aiding.* PhD thesis, Massachusetts Institute of Technology, 1983.

[23] Bruno Siciliano. Kinematic control of redundant robot manipulators: A tutorial. *Journal of Intelligent and Robotic Systems*, 3:201–212, 1990.

[24] Bruno Siciliano and Oussama Khatib. *Handbook of Robotics.* Springer, 2008.

[25] Bruno Siciliano and Lorenzo Sciavicco. *Robotics: Modelling, Planning and Control.* Springer-Verlag, 2000.

[26] Roger Skjetne. *The Maneuvering Problem.* Department of Engineering Cybernetics, NTNU, 2005.

[27] Mark W. Spong, Seth Hutchinson, and M. Vidyasagar. *Robot Dynamics and Control.* John Wiley & Sons, Inc, Hoboken, 2004.

[28] Kraft TeleRobotics. Engineering support document. Technical report.

[29] Kraft TeleRobotics. Robotic science and technology. `http://krafttelerobotics.com/`, 2011. Visited 10.06.2012.

[30] Kraft TeleRobotics. Raptor system manual, 2011.

[31] TIA. The telecommunications industry association. `http://www.tiaonline.org/`, 2012. Visited 12.05.2011.

[32] Tamás Urbancsek. *Modern Control Architecture for Designing Multiagent Telerobot Systems*. PhD thesis, Budapest University of Technology and Economics, 2009.

[33] Vadin Utkin and Hoon Lee. Chattering problem in sliding mode control systems. *Preprints of the 2nd IFAC Conf. on Analysis and Design of Hybrid Systems*, 2006.

[34] Charles W. Wampler. Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. *IEEE Transactions on Systems, Man, and Cybernetics*, 15:93–101, 1986.

[35] D. E. Whitney. Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on Man-Machine Systems*, 10:47–53, 1969.

[36] Lizhong Zheng and Robert Gallager. Principles of digital communication. Technical report, Massachusetts Institute of Technology, 2006.

# Appendix A

# Contents of Attached Folder

**PDF Files:**

- The Raptor Manipulator Data Sheet: A digital copy of the data sheet

**Maple Files:**

- Dynamic Model Calculations.mw: Automated framework for derivation of the dynamic model
- Calculations of potential energy.mw: Derivation of the potential energy
- Calculation of Jacobian matrix.mw: Calculates the Jacobian matrix

**MATLAB and Simulink Files**

- Run_M.m: File that simulates the system and plots all results
- Simulation_M.mdl: Full Simulink model
- init_par_M.m: Initial file for the Simulink model
- plotall.m: Plotting file
- SLcreateC_M.m: Matlab code for the centrifugal/Coriolis vector C(q,$\dot{q}$)$\dot{q}$
- SLcreateD_M.m: Matlab code for the inertia matrix M(q)
- SLcreateDinv_M.m: Matlab code for the inverse of the inertia matrix M(q)
- SLcreateF_M.m: Matlab code for the friction vector F($\dot{q}$)
- SLcreatePHI_M.m: Matlab code for the gravity/buoyancy vector G(q)
- SLcreateY_M.m: Matlab code for the alternative signum function $\psi$

# Appendix B

# Simulink: Joystick simulator

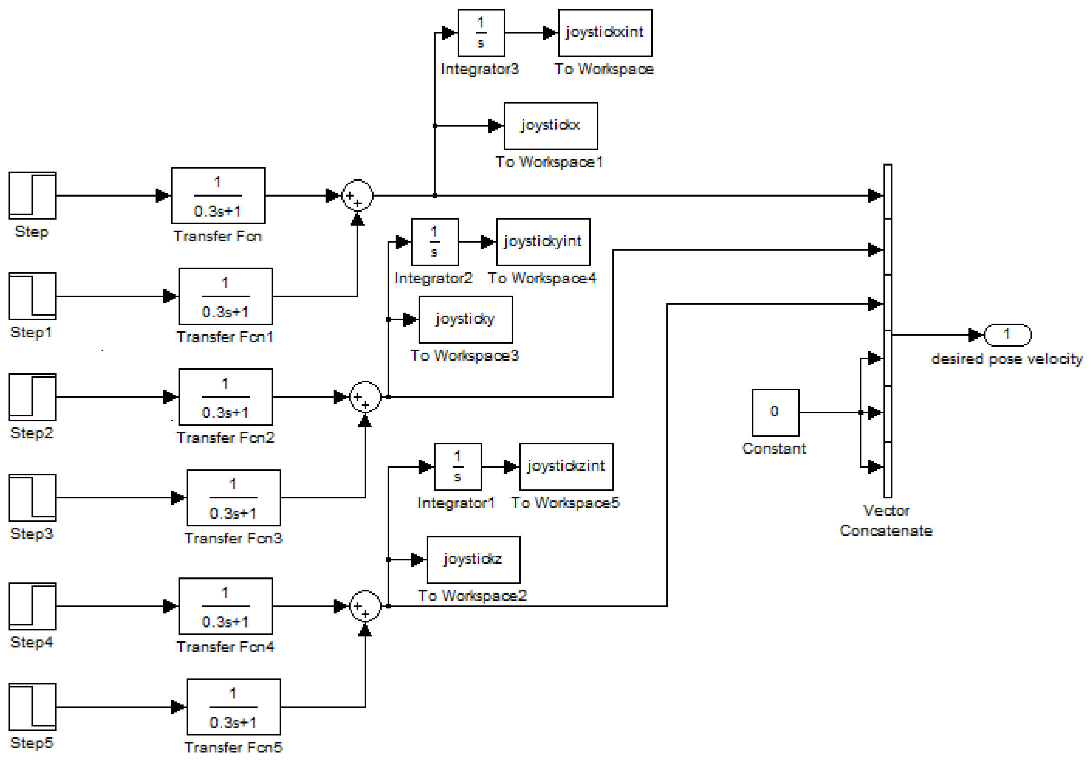

Figure B.1: Simulink diagram of joystick simulator