Force Feedback in the da Vinci Surgical Robot

AALBORG UNIVERSITY
STUDENT REPORT

Master Thesis | Ricard Bordalba Llaberia
Aalborg University | Electronics and IT |
Control & Automation

AALBORG UNIVERSITY
DENMARK

Electronics and IT
Aalborg University
http://www.aau.dk

Title:
Force Feedback in the
da Vinci Surgical Robot

Theme:
Surgical Robotics

Project Period:
September 1st 2015 - June 2nd 2016

Project Group:
1036

Participant(s):
Ricard Bordalba Llaberia

Supervisor(s):
Christoffer Sloth

Copies: 2

Page Numbers: 141

Date of Completion:
June 1, 2016

Abstract:

In the recent years, robotic surgery has become an important type of operation within minimally invasive surgery.
Limited or absent haptic feedback is considered to be among reasons that impede further spread of surgical robots. The lack of tactile feeling and the ongoing discussion among doctors about its necessity has served as the main motivation for this research. The da Vinci Surgical Robot available at AAU is used, for which a dynamical model is derived.
Due to the limitations in operating rooms, sensors are not a straightforward option. Therefore, a method to estimate contact force is developed in this thesis. Robot parameters are first estimated. Then, an extended Kalman filter is designed which is able to estimate external forces. Finally, a haptic device is used to verify estimated forces when robot is in contact with tissue.

Contents

Preface

This report documents the work conducted in the period from September 1st to June 2nd. It is focused on developing a force feedback system for the da Vinci Surgical Robot available in the Surgical Robotics Lab at AAU. The author would like to thank Assistant Engineer Simon Jensen for his help with the custom made AAU da Vinci hardware and Postdoc Karl Damkjær Hansen for his guidance with the operative system used in the da Vinci robot.

Reading Guide

Unless otherwise noted, this report uses the following notations:

- Cites and references to sources are denoted by square brackets containing author's surname and year of publication, directing to the bibliography section.
- Figures, equations and tables are numbered according to chapters and sequence, e.g. second figure of Chapter 2 is denoted as Fig. 2.2. However, equations are enclosed in brackets, e.g. Eq. (3.1).
- Units are given in square brackets, e.g. [m] for meters.
- Symbols and acronyms used in this report are presented in the nomenclature after this preface.
- Appendices are places after the main report and the bibliography, referred to by capital letters.

The attached CD contains relevant data, copies of reference used for the research, MATLAB and Maple scripts, ROS files and a digital copy of this report.

Ricard Bordalba Llaberia
<rborda14@student.aau.dk>

Nomenclature

Acronyms

AAU	Aalborg University
MIS	Minimally Invasive Surgery
FDA	Food and Drug Administration
EMA	European Medicines Agency
DOF	Degree of Freedom
ROS	Robotic Operating System
PI	Proportional Integral
PD	Proportional Derivative
DH	Denavit-Hartenberg convention
CM	Center of Mass
LSE	Least Square Estimation
WLSE	Weighted Least Square Estimation
PSO	Particle Swarm Optimization
MSE	Mean Square Error
EKF	Extended Kalman Filter
sbRIO	single board Reconfigurable Input/ Output
FPGA	Field Programmable Gate Array
IK	Inverse Kinematics

Symbols

Symbol	Description	Unit
${}^{B}_{A}\mathbf{R}$	Rotation matrix from frame {A} to {B}.	[-]
${}^{B}_{A}\mathbf{T}$	Transformation matrix of points from frame {A} to {B}.	[-]
$s\theta$	Sine of θ.	[-]
$c\theta$	Cosine of θ.	[-]
α	Rotation around x axis	[rad]
β	Rotation around y axis	[rad]
θ	Rotation around z axis	[rad]
${}^{A}P$	Vector of position with respect to a frame {A}	[m]
${}^{A}O_{B}$	Origin vector of frame {B} with respect to {A}	[m]
a_i	Link length (DH)	[m]
α_i	Link twist angle (DH)	[rad]
d_i	Joint distance (DH)	[m]
θ_i	Joint angle (DH)	[rad]
$\mathbf{J}$	Robot Jacobian	[-]
${}^{0}v_{ee}$	Linear velocity of the end-effector	[m/s]
${}^{0}\omega_{ee}$	Angular velocity of the end-effector	[rad/s]
$\boldsymbol{J}_v$	Jacobian matrix for the linear velocity	[-]
$\boldsymbol{J}_\omega$	Jacobian matrix for the angular velocity	[-]
τ	$n \times 1$ vector of motor torques	[Nm]
q	$n \times 1$ vector of joint position or generalized coordinates	[rad]
$\dot{q}$	$n \times 1$ vector of joint velocity	[rad/s]
$\ddot{q}$	$n \times 1$ vector of joint acceleration	[rad/s^2]
$\mathbf{M}(q)$	$n \times n$ mass matrix of the manipulator	[kg m^2]
$V(\dot{q}, q)$	$n \times 1$ vector of centrifugal and Coriolis terms	[Nm]
$G(q)$	$n \times 1$ vector of gravity terms	[Nm]
$F(\dot{q})$	$n \times 1$ vector of friction torques	[Nm]
τ_{ext}	$n \times 1$ vector of external torques seen at the joints	[Nm]
$\mathcal{L}$	Lagrangian function of the robot	[J]
k	Kinetic energy of the da Vinci robot.	[J]
u	Potential energy of the da Vinci robot.	[J]
m_i	Mass of the i^{th} link	[kg]
${}^{j}P_{c_i}$	CM of the i^{th} link with respect to frame {j}	[m]
${}^{j}v_{c_i}$	CM velocity of the i^{th} link with respect to frame {j}	[m/s]
${}^{j}\omega_i$	Angular velocity of the i^{th} link with respect to frame {j}	[rad/s]
N	Number of robot links	[-]
$\mathcal{I}$	Inertia Tensor around the inertia frame.	[kg m^2]
$\mathbf{I}$	Inertia Tensor around the CM of the rigid body frame.	[kg m^2]

Symbol	**Description**	**Unit**
I_x	Moment of inertia about x axis	[kg m^2]
I_y	Moment of inertia about y axis	[kg m^2]
I_z	Moment of inertia about z axis	[kg m^2]
$\mathbb{I}$	Identity matrix	[-]
g	3×1 gravity vector	[m/s^2]
K_s	Spring torsion coefficient	[Nm/rad]
l_{c_i}	Distance from the CM of the i^{th} link with respect to frame {i}	[m]
ρ	Mass density of the links	[kg/m^3]
r_i	Radius of the i^{th} link	[m]
h_i	Height of the i^{th} link	[m]
v	Viscous constant of torque friction	[Nms/rad]
c	Coulomb constant of torque friction	[Nm]
η	Motor gear ratio	[-]
$\boldsymbol{\eta}$	Diagonal matrix of gear ratios	[-]
J_m	Motor inertia	[kg m^2]
$\mathbf{J}_m$	Diagonal matrix of motor inertias	[kg m^2]
$\mathbf{K}_p$	Positive diagonal matrix of proportional gains	[-]
$\mathbf{K}_d$	Positive diagonal matrix of derivative gains	[-]
$\mathbf{C}(\dot{q}, q)$	Coriolis and centripetal coupling matrix	[W]
$\mathbf{F}$	Positive diagonal matrix with viscous friction coefficients	[Nms/rad]
$\boldsymbol{\Phi}$	Observation matrix of system	[-]
ω_f	Fundamental frequency of the optimal trajectory	[rad/s]
N_f	Number of Fourier elements	[-]
T_s	Sampling time	[s]
T_f	Period of optimal trajectory	[s]
a	Trajectory parameter	[-]
b	Trajectory parameter	[-]
$\mathbb{F}$	Data matrix build with $\boldsymbol{\Phi}$	[-]
$\mathbf{W}$	Diagonal matrix of weights of the measured torque	[-]
$\theta_{j,i}$	Parameter j of joint i	[-]
M	Number of periods of the repeated trajectory	[-]
K	Number of samples of one trajectory	[-]
F_{ee}	Vector of force and torque at the end effector	[-]
f_{ee}	Force at the end effector	[N]
τ_{ee}	Torque at the end effector	[Nm]
k_H	Human and haptic device stiffness	[N/m]
k_T	Tissue stiffness	[N/m]
b_H	Human and haptic device damper	[Ns/m]
$\tilde{y}$	Innovation or measurement residuals of the EKF	[-]

Chapter 1

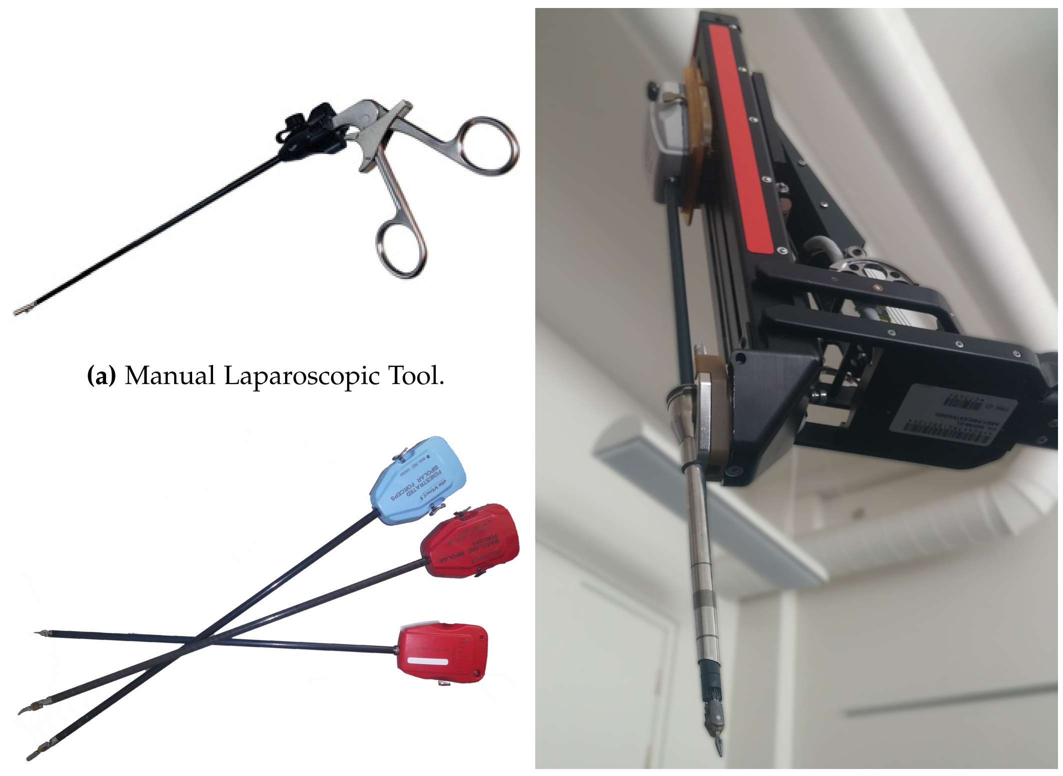
Introduction

The initial chapter of this thesis serves as an introduction to the da Vinci Surgical System and its application to minimally invasive surgery (MIS). First, a brief overview of the da Vinci Surgical System and its origins is given. Then, possible improvements within Surgical Robotics are discussed and the scope of the project is described. Finally, an outline of the thesis is presented.

1.1 Background in Surgical Robotics

Minimally invasive surgery (MIS) has completely changed the effects that surgical procedures causes to patients. Compared to traditional open surgery, MIS has proven to be safer, to reduce recovery time and length of hospital stay, while reducing the patient's pain and scarring [Trejos et al., 2010]. In MIS, small incisions are made to the patient such that the instruments enter the body, while causing the least damage possible to the patient. One type of MIS is the laparoscopic surgery, where thin telescopes with surgical tools attached are inserted into the patient through trocars. The patient's abdomen is inflated with air allowing the surgeon to maneuver the tools guided by visual feedback from a miniature camera (endoscope).

Initially, manual laparoscopic tools were used. It was in the 1980s when the use of master-slave robotic systems for minimally invasive laparoscopic surgery began. Such robots allow the doctor to perform surgery on a patient even though they are not physically in the same location, since camera feedback is provided. This is known as telesurgery. Moreover, it provides a better work environment for the doctor by reducing strain and fatigue. Surgeries that lasts for several hours can worsen the surgeon performance as they experience hand fatigue and tremors, whereas surgical robots are much steadier and smoother.



(a) Manual Laparoscopic Tool.

(b) Da Vinci Laparoscopic Tool.

(c) Tool in trocar.

Figure 1.1: Tools used for both manual and robotic laparoscopic surgery.

Nowadays, the American company Intuitive Surgical is the leading manufacture of surgical robots for minimally invasive laparoscopic surgery. Their robotic surgical system, known as da Vinci Surgical System, was released in 2000, after being approved by the Food and Drug Administration (FDA). Currently, more than 3500 da Vinci units are being used worldwide, where almost 70% of them are in the USA [Surgical, 2016].

1.2 The da Vinci Surgical Robot

Most surgical robots are master-slave systems which can be fully controlled by the surgeon. The da Vinci robot consist of four arms for tool and camera handling. Each of the arms have between 6 and 7 degrees of freedom (DOF), that are positioned inside the patient's body. A camera gives a 3D visual feedback to the master console, while the surgical instrument is controller with a joystick. The surgeon's movement is scaled down in order to improve accuracy. Moreover, the user tremor in the joystick is removed. However, no haptic feedback is provided to the surgeon.

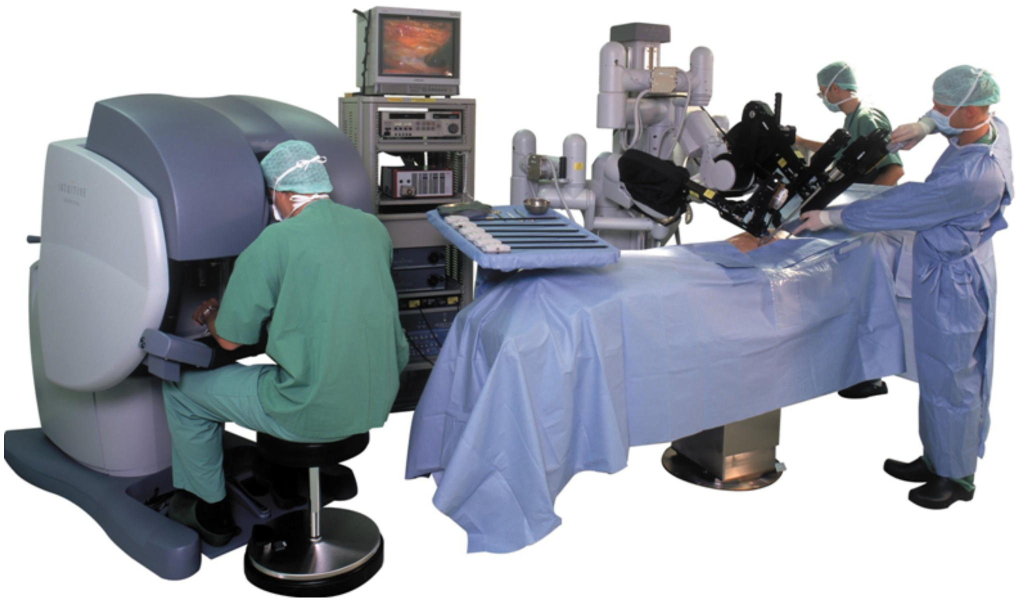


Figure 1.2: Da Vinci surgical System. On the left there is the surgeon master console where the surgeon can control the robot while looking to a 3 dimensional image. On the right, the robotic manipulator is following the trajectory given to operate a patient.

Limited or absent haptic feedback is considered to be among reasons that impede further spread of surgical robots [Enayati et al., 2016]. Once a system is controlled remotely, the user loses all abilities to feel interaction forces and can only rely on visual feedback. The loss of tactile feeling leads to some limitations [Trejos et al., 2010]:

- It is no longer possible to manually palpate tissue in order to evaluate its characteristics.
- Excessive forces might be applied and thus, leading to damage to healthy tissue.
- Insufficient forces might be applied when grabbing or suturing, leading to slippage and loose surgical knots.

Moreover, despite the potential of surgical robotics, research in the area has been relatively slow. Patents owned by Intuitive Surgical Inc. are one of the reasons behind this fact. However, these patents expires by 2016, opening the market for competitors [Wisniewski et al., 2015]. One potential challenger is the Raven open source surgical robot, which, like the da Vinci, also has its origins in the U.S. Army.

At Aalborg University, the Robotic Surgery Group owns a first generation of the da Vinci Surgical System, where both a hardware and software system has been developed in order to create an open source surgical robot in [Wisniewski et al., 2015]. The console has been detached from the robot in order to focus the research on automating surgical robots. The purpose behind it is to eventually create a surgical robot that is semi-autonomous, where responsibility would be shared between the surgeon and a computer [Wisniewski et al., 2015].

Current work at the Robotic Surgery Group is related to safety in robotics surgery. In [Sloth and Wisniewski, 2015], a safety controller was developed for beating-heart surgery, where the tool was kept within a safe distance from the heart by means of barrier certificates. In [Jakobsen and Lykkegaard, 2015], barrier certificates were also used to ensure a safe operation and avoid cutting undesired nerves or veins.

1.3 Improvements within Surgical Robotics

Despite the advantages that surgical robots can bring, there is still a lot of room for improvement. Two main areas are of great interest for further enhancement of robotic surgery: safety operation and haptic feedback.

Safety in robotic surgery

Once surgical tools enter the patient's body, it is probable that they get in contact with nerves, veins or organs that should never be cut. Therefore, it is important to guarantee safe operation, as it has been done at AAU in [Sloth and Wisniewski, 2015] and [Jakobsen and Lykkegaard, 2015].

Haptics feedback

While performing robotic surgery, lack of force feedback is one of the reasons why veins or tissue are damaged unintentionally, as the surgeon can only rely on visual information to estimate the forces applied. However, the use of haptic feedback has been rather underutilized due to practical difficulties. Its use in MIS has been discussed in [Trejos et al., 2010] and [Enayati et al., 2016]. Both stated the need to have force feeling, such that the experience of the user during surgery can be enhanced.

One of the major challenges is to determine how haptic information can be obtained. The use of sensors is quite limited due to the strict regulations when working in operating rooms (e.g. sterilization). For instance, regulations imposed by both the European Medicines Agency (EMA) and the American Food and Drug

Administration (FDA) should be followed [Enayati et al., 2016]. Moreover, the cost of such sensors should be reasonable. For this reason, sensorless force estimation arises as an alternative to solve this issue. This last approach is the one followed in this thesis.

1.4 Project Scope

This section will give an overview of the intended outcomes of this thesis. The lack of force feeling in the da Vinci Surgical System, like the one at AAU, and the ongoing discussion [Trejos et al., 2010][Enayati et al., 2016] about the need of haptic systems in the field of minimally invasive surgery has served as the main motivation for this research. Hence, the overall scope is to **develop a force feedback system for the da Vinci Surgical System** available at AAU. This is accomplished by estimating contact force without the use of force sensors, but only measurements already available from the motors, such as current or joint position. Once the main goal of the thesis is stated, it is important to define preliminary goals:

- **Develop a kinematic and dynamic model**. It implies an understanding of the engineering behind the da Vinci Surgical System.
- **Build a simulation for the robot dynamics** and **design a controller** for it, such that it can be used as a test bench before using the physical robot.
- **Estimation of model parameters** for the da Vinci robot, such that the model can be used in force estimation.
- **Develop a force estimation algorithm** that can estimate external forces without the need of extra sensors.
- **Analysis of the haptic system** in a master-slave configuration of the da Vinci robot. This implies an understanding of the entire configuration of the open source surgical robot at AAU.

1.5 Thesis Outline

This chapter has provided an introduction to surgical robots. The motivation for estimating contact forces and developing a haptic system has also been given. Then, the project scope has been defined. Finally, an outline of the thesis is now provided:

Chapter 2: System Description
The purpose of this chapter is to give an overview of the first generation da Vinci Surgical robot that is used in this thesis, which is available at Aalborg University [Lab, 2012].

Chapter 3: Kinematics
The purpose of this chapter is to develop a kinematic model for the da Vinci Surgical robot. This analysis describes the relation between the different links and joints of the robot, since it is needed for the study of system dynamics.

Chapter 4: Manipulator Dynamics
The purpose of this chapter is to derive a complete model for the da Vinci Surgical robot and to simulate it. At the time of the beginning of this thesis, a dynamic model for the system has not been developed yet.

Chapter 5: Controller for Simulation Model
The purpose of this chapter is to design a controller for the simulation of the da Vinci model and to prove its stability.

Chapter 6: Robot Parameters Estimation
The purpose of this chapter is to identify the robot parameters by means of estimation methods used in industrial robots.

Chapter 7: External Forces Estimation
The purpose of this chapter is to estimate external forces applied to the robot. Once the model is known, force is estimated without using force sensors but only measurements from the motor.

Chapter 8: Haptic System
The purpose of this chapter is to present the haptic setup used in this thesis, such that contact forces can be felt by the user. The stability of the haptic interaction is also discussed.

Chapter 9: Implementation
The purpose of this chapter is to describe the implementation and the contribution of this project to the da Vinci Surgical Robot at AAU [Lab, 2012].

Chapter 2

System Description

The purpose of this chapter is to give an overview of the first generation da Vinci Surgical robot that is used in this thesis, which is available at Aalborg University [Lab, 2012]. The different elements that comprise the system are explained here. However, a detail description of the hardware and software configuration made at AAU is given in Chapter 9.

2.1 Da Vinci Robot at Aalborg University

The da Vinci robot at Aalborg Univeristy is the first generation of this surgical robot which is shown in Fig. 2.1. It consist of two main parts:

- **Surgeon master console:** This part is where the surgeon controls the robot. Two high resolution eyepieces at the console show a 3 dimensional image that gives the surgeon very real vision of the process, compared to the 2 dimensional view that is used in standard laparoscopic surgery.

- **Slave Robotic Manipulator:** It consist of four robotic arms, also called manipulators, that respond to the commands given by the surgeon. Each robotic arm has 6-7 DOF and one of them holds an endoscope.

The slave robotic manipulator has been detached from the surgeon master console in the da Vinci Surgery System which is in the laboratory at AAU. This has been done in order to allow research in semi-autonomous control for surgical robotics.

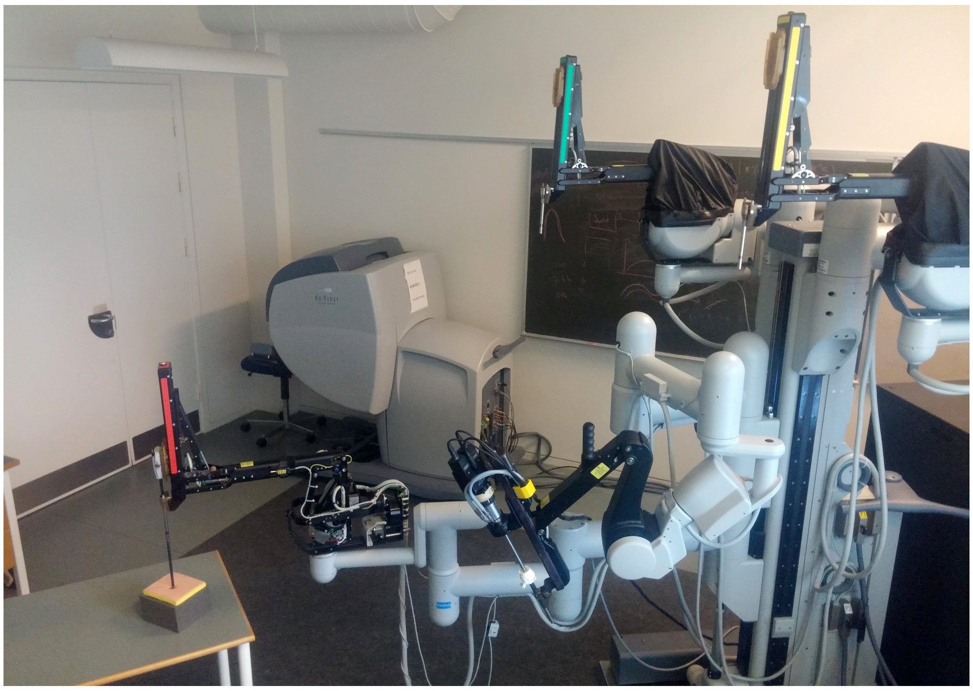


Figure 2.1: 1st generation of the da Vinci Surgical System at Aalborg University.

2.1.1 Slave Robotic Manipulator

The da Vinci Surgical Robot consists of four interactive robotic arms, with different instruments each, controlled from the surgeon console. One of these instruments is an endoscope, a tube with two small cameras and a light, which is used to give 3D image to the console. Each manipulator has three different parts as shown in Fig. 2.2: the arm, the hand and the tool with the end-effector.

- **Arm:** The arm is the first part of the robot, which is furthest from the patient. Its joints are fixed before any procedure and cannot be moved during surgery.
- **Hand:** The hand is the second part of the robot that is attached to the end of the arm. It can be moved during surgery.
- **Tool and end-effector:** The tool or instrument is the last part of the robot. There are several types of instruments that can be used, each one with a specific function in a surgical procedure. Depending on the instrument used, the arm will have either 6 or 7 DOF. The end-effector is the end of the tool and is specially designed to interact with the patient. Its movements should be extremely precise and they try to mimic a human wrist.

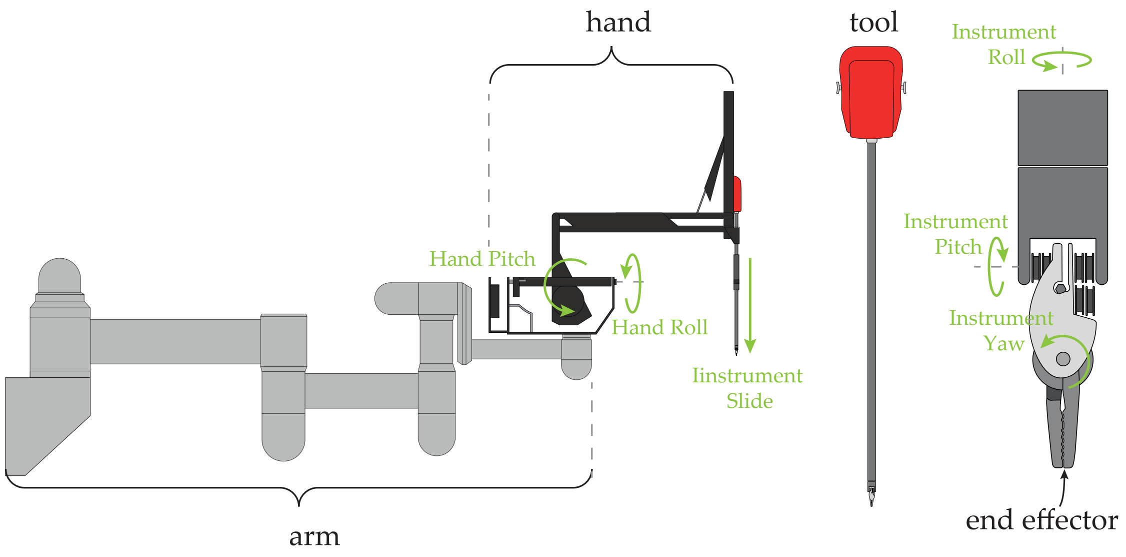


Figure 2.2: Description of the different parts of one arm of the da Vinci Surgical System.

2.1.2 Surgeon Master Console

Generally, the surgeon master console is in the same room than the robotic arms and the patient. However, it can also be placed further away from the operating room. The surgeon has 3D visual feedback obtained from the endoscope cameras, then foot pedals can be used to move this endoscope. Two 7-DOF joysticks provide control of the robotic manipulators, like the one represented in Fig. 2.2. However, no force feedback is given to the joysticks. Furthermore, the console in the da Vinci Surgical Robot at AAU shown in Fig. 2.3 has been detached from the robot.

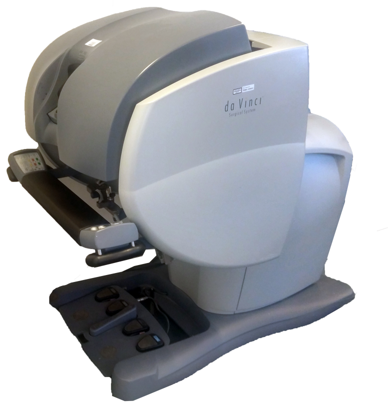


Figure 2.3: Surgeon master console of the da Vinci robot.

Since the main goal of this thesis is to implement force feedback to the system, a haptic device is needed such that a contact feeling can be given to the surgeon. The Geomagic Touch shown in Fig. 2.4 has been acquired as it is widely used in haptic research due to its prize and easy configuration. The Geomagic Touch gives 6 DOF with 3D force feedback. Motors are used to create the 3D forces such that the contact and interaction with objects can be simulated. Its configuration and setup is quite straightforward thanks to its Ethernet connectivity. Moreover, packages for the Robotic Operating System (ROS) are available online. This is important as the ROS platform is used to connect the Geomagic Touch with the da Vinci Surgical Robot.

Figure 2.4: Geomagic Touch acquired for force feedback.

2.1.3 Controllers in the da Vinci Surgical Robot

As mentioned before, the manipulator has been detached from the console in order to be controlled autonomously or by an external joystick. Each joint is moved by Maxon DC motors. Therefore, custom independent controllers are designed for each motor by using a cascade scheme with position, velocity and current PI controllers as shown in Fig.2.5. The cascade scheme is chosen in order to attenuate disturbances such as joint coupling, and to minimize friction effects. In this thesis, the custom controllers have not been modified.

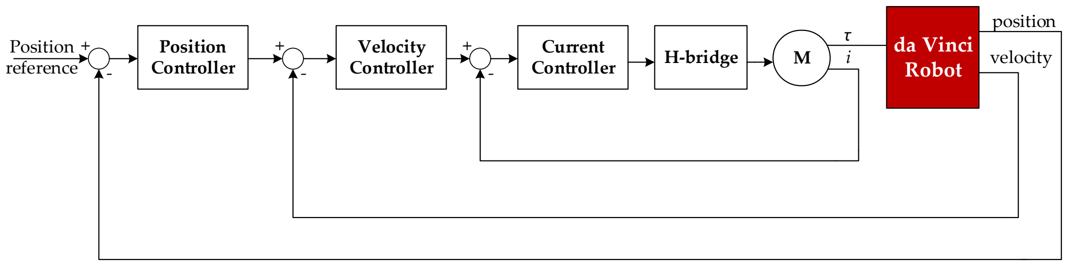


Figure 2.5: Block diagram of the cascade controller designed for the motors of the da Vinci Robot. The three controllers are PI controllers.

Chapter 3

Kinematics

The purpose of this chapter is to develop a kinematic model for the da Vinci Surgical robot which is available at Aalborg University [Lab, 2012]. This analysis describes the relation between the different links of one of the manipulators, as it is needed for the study of the system dynamics in Chapter 4.

3.1 Forward Kinematics

Forward kinematics refers to the equations that compute the position of the robot links by using the joint parameters. In this thesis, the kinematic model of one of the da Vinci manipulators is developed by using the Denavit-Hartenberg notation [Denavit and Hartenberg, 1955]. The hand and tool, which are shown in Fig. 3.1, are studied.

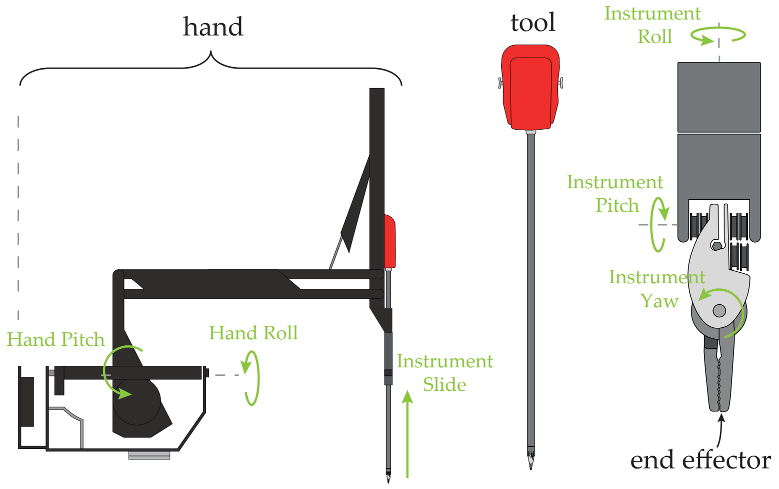


Figure 3.1: Hand, tool and end effector of one manipulator arm with the naming convention used for the da Vinci robot.

3.1.1 Frames Description

To describe the kinematics of links and end-effector of any robot, positions and orientations should be defined with respect to a coordinate system or frame. This frame can be both fixed in the inertial space or fixed to an object (e.g. links or joints). Thus, it is important to define whether position and orientation is relative to one frame or another.

Position is defined as a vector ${}^{A}P = [p_x, p_y, p_z]^T$ with respect to a frame {A}, while orientation is defined using a 3x3 rotation matrix ${}^{A}_{B}\mathbf{R}$ which describes a rotation of frame {B} relative to {A}. This rotation matrix is characterized to be orthogonal, which allows to find the inverse relation by simply using the transpose as in Eq. (3.1):

$$ {}^{A}_{B}\mathbf{R} = {}^{B}_{A}\mathbf{R}^{T}, \quad \text{where} \quad \det(\mathbf{R}) = 1 \tag{3.1} $$

Furthermore, the rotation matrix $\mathbf{R}$ can be build as an arbitrary combination of different rotations around the x, y and z axis of the reference frame used.

$$ \mathbf{R}_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c\alpha & -s\alpha \\ 0 & s\alpha & c\alpha \end{pmatrix} \quad \mathbf{R}_y(\beta) = \begin{pmatrix} c\beta & 0 & s\beta \\ 0 & 1 & 0 \\ -s\beta & 0 & c\beta \end{pmatrix} \quad \mathbf{R}_z(\theta) = \begin{pmatrix} c\theta & -s\theta & 0 \\ s\theta & c\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.2} $$

where $c\alpha$ and $s\alpha$ are shorthands for $\cos(\alpha)$ and $\sin(\alpha)$ respectively, while α, β and θ are the angles of rotation around x, y and z axis respectively. An example of rotation of frame {B} relative to frame {A} around x axis is shown in Fig. 3.2.

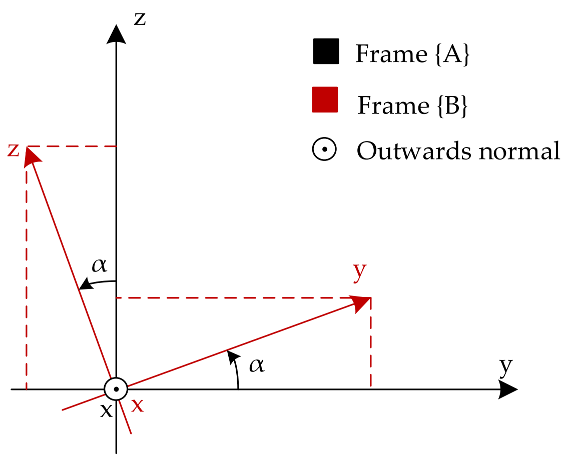


Figure 3.2: Positive rotation of frame {B} relative to frame {A} of α around x axis.

Very often, points in space are described in different frames. Then, Eq. (3.3) is used to obtain a mapping of one point described in one frame ${}^{B}P$ to a description in another frame ${}^{A}P$.

$$ {}^{A}P = {}^{A}_{B}\mathbf{R} \cdot {}^{B}P + {}^{A}O_{B} \tag{3.3} $$

where ${}^{A}O_{B}$ is the origin vector of frame {B} with respect to {A}. In Fig. 3.3, it is shown how frame {B} is described with respect to frame {A} with a translation ${}^{A}O_{B}$ and a rotation ${}^{A}_{B}\mathbf{R}$ movement.

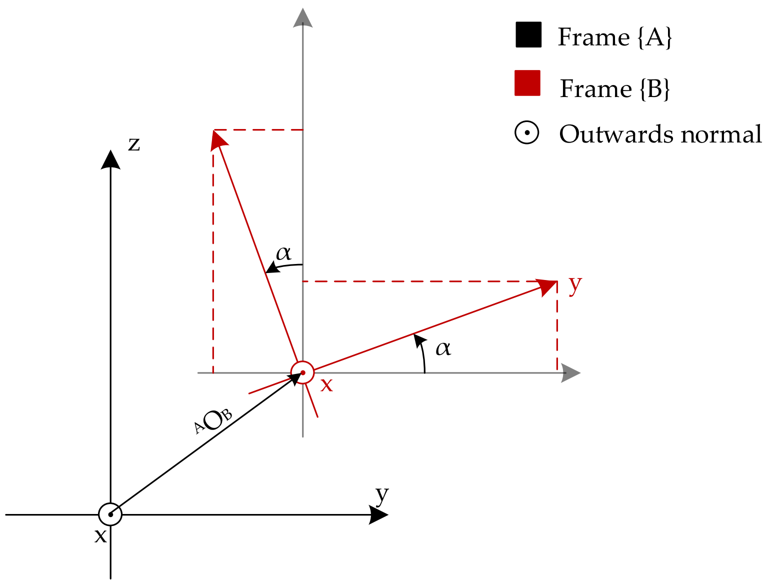


Figure 3.3: Translation ${}^{A}O_{B}$ and rotation of α around x axis of frame $\{B\}$ with respect to frame $\{A\}$.

This relation between two frames can be expressed in a 4x4 homogeneous transformation matrix **T** [Craig, 2009], defined as:

$$ {}^{A}_{B}\mathbf{T} = \left(\begin{array}{c|c} {}^{A}_{B}\mathbf{R} & {}^{A}O_{B} \\ \hline 0_{1x3} & 1 \end{array}\right) \tag{3.4} $$

It is built by using the rotation matrix and the origin vector ${}^{A}O_{B}$. The extra row makes the **T** matrix square and allows the concatenation of different transformations. In order to transform points in frame {B} to frame {A}, as it is done in Eq. (3.3), an extra row with a 1 is needed in the vector P. Then, Eq. (3.3) can be expressed in a compact form using **T** matrix as:

$$ {}^{A}P = {}^{A}_{B}\mathbf{T} \cdot {}^{B}P = \left(\begin{array}{c|c} {}^{A}_{B}\mathbf{R} & {}^{A}O_{B} \\ \hline 0_{1x3} & 1 \end{array}\right) \begin{pmatrix} {}^{B}P \\ 1 \end{pmatrix} \tag{3.5} $$

An important property of the **T** matrix is that it can describe a frame {N} with respect to a frame {0} if intermediate frame descriptions are known:

$$ {}^{0}_{N}\mathbf{T} = {}^{0}_{1}\mathbf{T} \cdot {}^{1}_{2}\mathbf{T} \cdots {}^{N-1}_{N}\mathbf{T} \tag{3.6} $$

For the hand and tool of the da Vinci robot, a definition of frames is needed in order to describe the kinematic chain, where this chain is the sequence of links and joints that goes from the base of the hand to the end effector of the tool. Different approaches can be done to define the frames, but due to its simplicity, the Denavit-Hartenberg (DH) convention is used in this thesis.

3.1.2 Denavit-Hartenberg Convention

The Denavit-Hartenberg convention is commonly used in robotics because it reduces the number of parameters needed to describe the kinematic chain. Four terms per link are necessary: two describing the link and two describing the connection to the previous link. The DH convention states that [Denavit and Hartenberg, 1955]:

- Frame {i} is attached to link i.
- The Z_i axis is in the direction of the motion axis of joint $i+1$.
- The X_i axis is parallel to the common normal between Z_{i-1} and Z_i.
- The Y_i axis is now chosen by the right hand coordinate system.
- a_i or *link length* is the distance from Z_i to Z_{i-1} measured along X_i.
- α_i or *link twist angle* is the angle from Z_{i-1} to Z_i measured about X_i using the right hand coordinate system.
- d_i or *joint distance* is the distance from X_{i-1} to X_i measured along Z_{i-1}.
- θ_i or *joint angle* is the angle from X_{i-1} to X_i measured about Z_{i-1}.

In this convention, the transformation between two consecutive frames {i} and {i-1} is described with a rotation along the Z_i axis plus a rotation along the X_i axis, such that:

$$ {}^{i-1}_{i}\mathbf{T} = \left(\begin{array}{cc} & 0 \\ \mathbf{R}_z(\theta_i) & 0 \\ & d_i \\ \hline 0 & 1 \end{array}\right) \left(\begin{array}{cc} & a_i \\ \mathbf{R}_x(\alpha_i) & 0 \\ & 0 \\ \hline 0 & 1 \end{array}\right) = \left(\begin{array}{cccc} c\theta_i & -c\alpha_i s\theta_i & s\alpha_i s\theta_i & a_i c\theta_i \\ s\theta_i & c\alpha_i c\theta_i & -s\alpha_i c\theta_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \tag{3.7} $$

where $\mathbf{R}_z(\theta)$ and $\mathbf{R}_x(\alpha)$ are obtained from Eq. (3.2).

Now, the DH convention is applied to the definitions of the frames for the da Vinci hand and tool as shown in Fig. 3.4. A DH kinematic chain from previous work at AAU [Jakobsen and Lykkegaard, 2015] is used for the hand and tool of the robot and a new base frame is set at the base link. Notice that the base frame from

Fig. 3.4 is not moving as it is part of the arm fixed position. Likewise, due to the counterweight in link 3, joint 2 is not vertically aligned with joint 3 and therefore, there is small angular offset between frames.

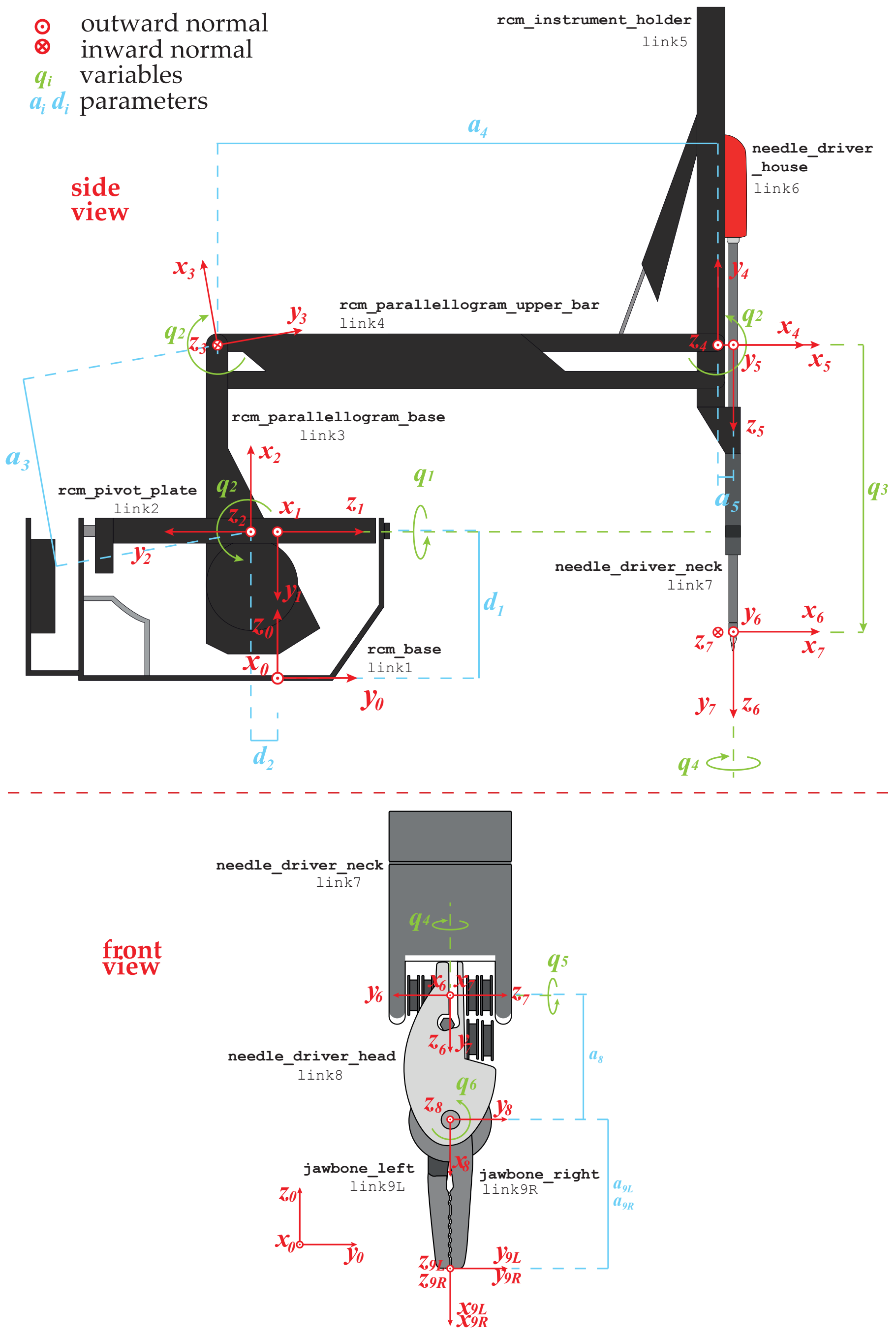


Figure 3.4: Coordinate frames defined using the DH convention.

Links 3, 4 and 5 are built such that they move as a parallelogram, which makes them have the same joint angle q_2. This implies that there are a total of 7 variables that determine the position of the robot. Therefore, one manipulator of the da Vinci robot (hand and tool) has 7 degrees of freedom. In case there is one end-effector instead of two ($9L$ and $9R$), then the system has 6 DOF. The parameters for this frame definition are summarized in Table 3.1.

i	α_i [rad]	a_i [m]	d_i [m]	θ_i [rad]	Name
1	$-\pi/2$	0	0.161	0	base
2	$-\pi/2$	0	-0.035	$-\pi/2 + q_1$	pivot plate
3	π	0.190	0	$0.03 + q_2$	parallellogram base
4	π	0.515	0	$0.03 + \pi/2 + q_2$	parallel. upper bar
5	$\pi/2$	0.040	0	q_2	instrument holder
6	0	0	$0.282 + q_3$	0	needle driver house
7	$\pi/2$	0	0	q_4	needle driver neck
8	$\pi/2$	0.009	0	$\pi/2 + q_5$	needle driver head
9L	0	0.009	0	q_6	jawbone left
9R	0	0.009	0	q_7	jawbone right

Table 3.1: Parameters of the DH convention for the da Vinci robot hand and tool. The variables represented as q can be controlled.

A total of 7 variables define uniquely the configuration of the 7 DOF of the manipulator. In the study of the rigid body dynamics developed in Chapter 4, these parameters are known as generalized coordinates q. Its generalized velocities $\dot{q}$ are the time derivatives of the generalized coordinates. Now, all the transformation matrices are written for later use in the dynamics analysis as a function of these generalized coordinates q. Note that there are two end effector in the da Vinci robot, therefore frames {9R} and {9L} are described with respect to {8}.

$$
\begin{gathered}
{}^{0}_{1}\mathbf{T} = \left(\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & d_1 \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \quad
{}^{1}_{2}\mathbf{T} = \left(\begin{array}{cccc} s(q_1) & 0 & c(q_1) & 0 \\ -c(q_1) & 0 & s(q_1) & 0 \\ 0 & -1 & 0 & d_2 \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \\
{}^{2}_{3}\mathbf{T} = \left(\begin{array}{cccc} c(q_2+0.03) & s(q_2+0.03) & 0 & a_3 \cdot c(q_2+0.03) \\ s(q_2+0.03) & -c(q_2+0.03) & 0 & a_3 \cdot s(q_2+0.03) \\ 0 & 0 & -1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \\
{}^{3}_{4}\mathbf{T} = \left(\begin{array}{cccc} -s(q_2+0.03) & c(q_2+0.03) & 0 & -a_4 \cdot s(q_2+0.03) \\ c(q_2+0.03) & s(q_2+0.03) & 0 & a_4 \cdot c(q_2+0.03) \\ 0 & 0 & -1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \\
{}^{4}_{5}\mathbf{T} = \left(\begin{array}{cccc} c(q_2) & 0 & s(q_2) & a_5 \cdot c(q_2) \\ s(q_2) & 0 & -c(q_2) & a_5 \cdot s(q_2) \\ 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \quad
{}^{5}_{6}\mathbf{T} = \left(\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & q_3+0.282 \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \\
{}^{6}_{7}\mathbf{T} = \left(\begin{array}{cccc} c(q_4) & 0 & s(q_4) & 0 \\ s(q_4) & 0 & -c(q_4) & 0 \\ 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \quad
{}^{7}_{8}\mathbf{T} = \left(\begin{array}{cccc} -s(q_5) & 0 & c(q_5) & -a_8 \cdot s(q_5) \\ c(q_5) & 0 & s(q_5) & a_8 \cdot c(q_5) \\ 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \\
{}^{8}_{9L}\mathbf{T} = \left(\begin{array}{cccc} c(q_6) & -s(q_6) & 0 & a_{9L} \cdot c(q_6) \\ s(q_6) & c(q_6) & 0 & a_{9L} \cdot s(q_6) \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right) \quad
{}^{8}_{9R}\mathbf{T} = \left(\begin{array}{cccc} c(q_7) & -s(q_7) & 0 & a_{9R} \cdot c(q_7) \\ s(q_7) & c(q_7) & 0 & a_{9R} \cdot s(q_7) \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right)
\end{gathered}
\tag{3.8}
$$

In order to verify the transformation matrices defined according to the DH convention, the Robotics Toolbox in MATLAB [Corke, 2011] is used as it can give a graphical representation of the robot. Notice that dynamical parameters can be given to Robotics Toolbox, however it is designed for serial links and the parallelogram can not be modeled. Therefore, only a kinematic model is built, while dynamics are found in Chapter 4. A 3D plot of the robot is shown in Fig. 3.5 given the DH parameters in Table 3.1.

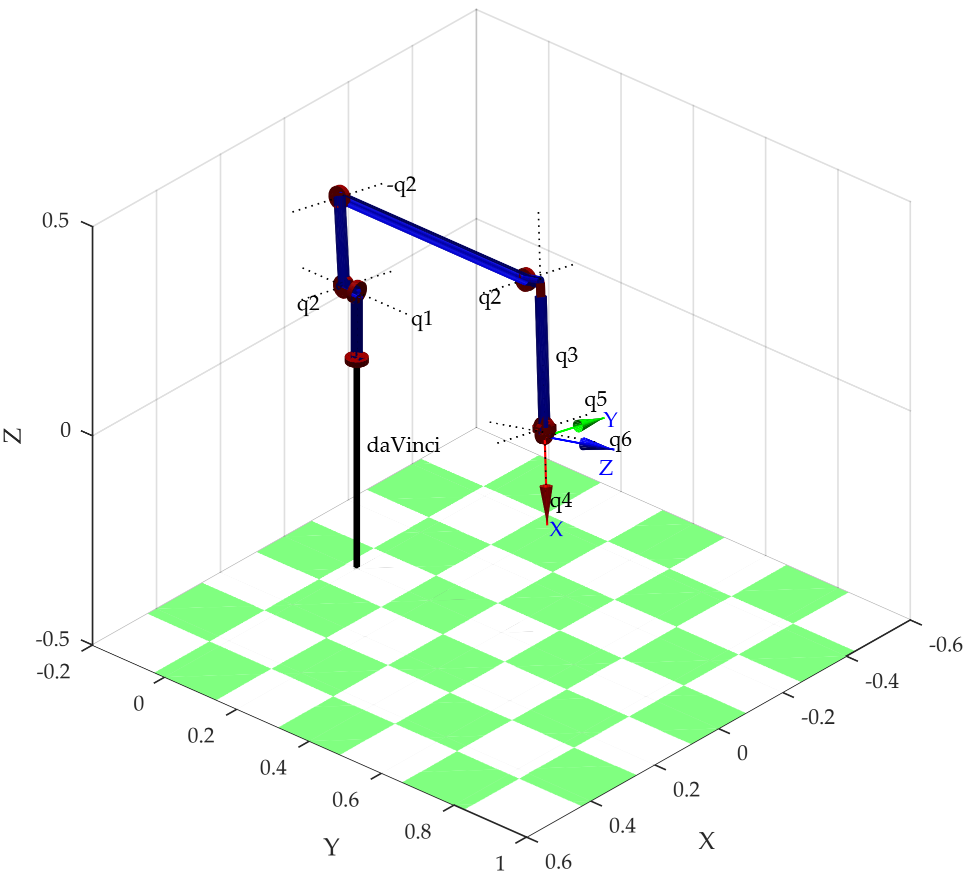


Figure 3.5: 3D representation of the daVinci hand and tool in the Robotics Toolbox (Matlab) according to the frames defined with the DH convention.

3.2 The Manipulator Jacobian

A Jacobian **J** is a matrix of first order partial derivatives of a vector function and is a useful tool widely used in robotics. Basically, it defines a relation or mapping between two different representation of a system. For instance, a robot end-effector is defined both by:

- Its position and orientation with respect to a base frame $\{0\}$, which will be denoted as ${}^{0}x$.
- A set of joint angles, known as generalized coordinates q in this thesis.

Therefore, in order to obtain this relation, the robot Jacobian **J** is defined as [Spong et al., 2004]:

$$\mathbf{J} = \frac{\partial x}{\partial q} \tag{3.9}$$

which is rearranged by adding ∂t to the numerator and denominator in order to relate linear and angular velocity of the end-effector to joint velocities:

$$\mathbf{J} = \frac{\partial x}{\partial t}\frac{\partial t}{\partial q} \quad \rightarrow \quad \frac{\partial x}{\partial t} = \mathbf{J} \cdot \frac{\partial q}{\partial t} \tag{3.10}$$

Now, considering the time derivatives of x and q, Eq. (3.10) is expressed as:

$$\dot{x} = \mathbf{J} \cdot \dot{q} \quad \rightarrow \quad \begin{bmatrix} {}^{0}v_{ee} \\ {}^{0}\omega_{ee} \end{bmatrix} = \begin{bmatrix} \boldsymbol{J_v} \\ \boldsymbol{J_\omega} \end{bmatrix} \cdot \dot{q} \tag{3.11}$$

where

${}^{0}v_{ee}$	is the linear velocity of the end-effector [m/s]
${}^{0}\omega_{ee}$	is the angular velocity of the end-effector [rad/s]
$\dot{q}$	is the $n \times 1$ vector of joint velocities [rad/s] (revolute) and [m/s] (prismatic)
$\boldsymbol{J_v}$	is the Jacobian matrix for the linear velocity
$\boldsymbol{J_\omega}$	is the Jacobian matrix for the angular velocity

Another useful application of the Jacobian is to relate torques at the joints with forces and torques at the end-effector as it is done in Chapter 7. Thus, the Jacobian matrix is now defined in this chapter for its further use in this thesis.

Chapter 4

Manipulator Dynamics

The kinematic model described in Chapter 3 considers the motion of the robot, but not the forces required to cause motion. This chapter presents the process of modeling the manipulator dynamics of the da Vinci robot available at the AAU laboratory. At the time of the beginning of this thesis, a dynamic model for the da Vinci robot has not been developed yet. The chapter will be divided in four main parts:

- First, the Euler-Lagrange formulation is used to derive the general expression of the robot dynamics based on the energy of the links.
- The energy of the robot is needed in the Euler-Lagrange formulation. Therefore, both potential and kinetic energy are defined for each link. To do so, position, linear and angular velocity of all links are defined with respect to the same frame, the base frame.
- Then, a friction model is also defined for all the joints.
- Finally, a simulation of the manipulator dynamics is created in Simulink in order to have a test bench for the da Vinci robot.

4.1 Euler-Lagrange Formulation

There are two approaches to dynamic modeling: one is the Newton-Euler method, which is based on the balance of forces and torques, and the other one is the Euler-Lagrange method, which is based on the energy of the robot links. Nevertheless, both methods should reach the same equations describing the dynamics, which can be written in a generalized form for any manipulator with n degrees of freedom [Craig, 2009].

$$\tau = \mathbf{M}(q)\ddot{q} + V(q,\dot{q}) + G(q) + F(\dot{q}) + \tau_{ext} \quad \text{[Nm]} \tag{4.1}$$

where

τ	is an $n \times 1$ vector of motor torques [Nm]
q	is an $n \times 1$ vector of joint position [rad]
$\dot{q}$	is an $n \times 1$ vector of joint velocity [rad/s]
$\ddot{q}$	is an $n \times 1$ vector of joint acceleration [rad/s^2]
$\mathbf{M}(q)$	is the $n \times n$ mass matrix of the manipulator [kg m^2]
$V(\dot{q}, q)$	is an $n \times 1$ vector of centrifugal and Coriolis terms [Nm]
$G(q)$	is an $n \times 1$ vector of gravity terms [Nm]
$F(\dot{q})$	is an $n \times 1$ vector of friction torques [Nm]
τ_{ext}	is an $n \times 1$ vector of external torques seen at the joints [Nm]

Notice that units are for **revolute** joints. If joint is **prismatic**, q is a linear displacement [m] and τ is not a torque anymore but a force [N].

In this thesis, the Euler-Lagrange formulation is used. It relies on the energy properties of any mechanical system in order to compute the equations of motion. First, the Lagrangian $\mathcal{L}$ is defined as the difference between the kinetic $k(q, \dot{q})$ and potential energy $u(q)$ of the system [Murray et al., 1994]:

$$\mathcal{L}(q, \dot{q}) = k(q, \dot{q}) - u(q) \quad [\mathrm{J}] \tag{4.2}$$

where q is the vector of n generalized coordinates, which describe the configuration of any mechanical system of n DOF, and $\dot{q}$ is the vector of generalized velocities. Note that the DH joint variables are already generalized coordinates as mentioned in Section 3.1.2. Therefore, the hand and tool of the da Vinci robot has 7 joint variables or generalized coordinates to describe the position of the end-effector.

Now, consider Hamilton's principle [Wisniewski, 2015b], which states that the motion of a mechanical system from time a to b is such that the integral of the Lagrangian $\mathcal{L}$:

$$I(t, q, \dot{q}) = \int_a^b \mathcal{L}(t, q, \dot{q}) dt \tag{4.3}$$

has a stationary value. It means that among all curves q from a to b, the motion of the system will occur along the curve that gives a stationary value (an extreme value) to the integral $I(t, q, \dot{q})$, that is:

$$\frac{\partial I(t, q, \dot{q})}{\partial q(t)} = 0 \tag{4.4}$$

Then, as it is described in [Wisniewski, 2015a], the solution to Eq. (4.4) is given by the Euler-Lagrange equation:

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}} - \frac{\partial \mathcal{L}}{\partial q} = 0 \tag{4.5}$$

where this trajectory describes the motion of the mechanical system when only conservative forces are present. The gravitational force, spring force or magnetic force are examples of conservative forces that would appear in Eq. (4.5). However, non conservative forces, such as friction, motor forces or external forces, are not considered in Eq. (4.5). Therefore, in order to include an arbitrary force (not necessary conservative), the Lagrange-d'Alembert Principle, also known as the Virtual Work's Principle, is used. It states that the trajectory of a mechanical system influenced by a force Q (not necessary conservative) is such that the equation of motion for the system is given by [Wisniewski, 2015b]:

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}} - \frac{\partial \mathcal{L}}{\partial q} = Q \tag{4.6}$$

where Q is known as the generalized force or torque vector, and its i^{th} component is a torque [Nm] if joint i is **revolute**, or is a force [N] if joint i is **prismatic**. Likewise, the i^{th} component of q is a linear position [m] if joint is **prismatic** or is an angular position [rad] if joint is **revolute** [Spong et al., 2004]. By using the definition of the Lagrangian from Eq.(4.2), Eq. (4.6) can be written as:

$$\frac{d}{dt}\frac{\partial\left(k(q,\dot{q}) - u(q)\right)}{\partial \dot{q}} - \frac{\partial\left(k(q,\dot{q}) - u(q)\right)}{\partial q} = Q \tag{4.7}$$

where potential energy u does not depend on velocity $\dot{q}$ but only position q. Therefore, the term $\frac{\partial u(q)}{\partial \dot{q}} = 0$ and hence, Eq. (4.7) is reduced to:

$$\frac{d}{dt}\frac{\partial k}{\partial \dot{q}} - \frac{\partial k}{\partial q} + \frac{\partial u}{\partial q} = Q \tag{4.8}$$

As later explained in Sec. 4.4, Eq.(4.8) will lead to an equation of the form of:

$$\mathbf{M}(q)\ddot{q} + V(q,\dot{q}) + G(q) = Q \tag{4.9}$$

where:

- The term $\mathbf{M}(q)\ddot{q}$ involves the second derivative of the generalized coordinates q, the link masses and inertia elements.
- $V(q,\dot{q})$ are quadratic terms involving the first derivative of q. If the terms consist of products of $\dot{q}_i^2$ they are called centrifugal and if they consist of product of $\dot{q}_i\dot{q}_j$, where $j \neq i$, they are called Coriolis terms.
- The term $G(q)$ comes from deriving the potential energy and only depends on q.

Finally, considering that external torques (or forces) and friction torques (or forces) are non-conservative and opposed to the actuator torques (or forces), they can just be included in Q as:

$$Q = \tau - F(\dot{q}) - \tau_{ext} \tag{4.10}$$

where:

τ	is a vector of actuator torques [Nm] or forces [N]
τ_{ext}	is a vector of external torques [Nm] or forces [N]
$F(\dot{q})$	is a vector of velocity dependent friction torques [Nm] or forces [N]

Note that the friction model is developed in Section 4.3. Then, Eq. (4.9) becomes:

$$\tau = \mathbf{M}(q)\ddot{q} + V(q,\dot{q}) + G(q) + F(\dot{q}) + \tau_{ext} \tag{4.11}$$

4.2 Kinetic and Potential Energy

The dynamics are derived assuming that the kinetic and potential energy can be express in terms of generalized coordinates q. Therefore, it is necessary to compute these terms as a function of the joints variables (positions and velocities) of the da Vinci robot. Likewise, it is important to **define the energy with respect to the inertial or base frame**.

In [Spong et al., 2004], the kinetic energy of a rigid body is defined as the sum of both its translational and rotational kinetic energy. If the daVinci links are assumed to be rigid bodies, Eq. (4.12) is used to find the kinetic energy of the manipulator.

$$k_i = \underbrace{\frac{1}{2} m_i \, {}^0v_{c_i}^T \, v_{c_i}}_{\text{Translational}} + \underbrace{\frac{1}{2} \, {}^0\omega_i^T \, \mathcal{I}_i \, {}^0\omega_i}_{\text{Rotational}} \quad [\text{J}] \tag{4.12}$$

where

k_i	is the kinetic energy of the i^{th} link [J]
m_i	is the mass of the i^{th} link [kg]
${}^0v_{c_i}$	is the center of mass velocity of the i^{th} link about the base frame [m/s]
${}^0\omega_i$	is the angular velocity of the i^{th} link with respect to the base frame [rad/s]
$\mathcal{I}_i$	is the 3×3 matrix of inertia tensor expressed in the base frame [rad/s^2]

Now, the total kinetic energy of the manipulator can be computed as the sum of kinetic energy of its links:

$$k = \sum_{i=1}^{N} k_i \quad [\text{J}] \tag{4.13}$$

where N is the number of links. Similarly, the potential energy u of the manipulator is given by the sum of individual potential energy of the links:

$$u = \sum_{i=1}^{N} u_i \quad [\mathrm{J}] \tag{4.14}$$

As the links are assumed to be rigid bodies, then its potential energy is given in [Spong et al., 2004] as:

$$u_i = m_i \, g^T \, {}^0P_{c_i} \quad [\mathrm{J}] \tag{4.15}$$

where

u_i	is the potential energy of the i^{th} link [J]
g	is the 3×1 gravity vector [m/s^2]
m_i	is the mass of the i^{th} link [kg]
${}^0P_{c_i}$	is the 3×1 center of mass of the i^{th} link with respect to frame $\{0\}$ [m]

4.2.1 Position and Velocities of Links

In order to derive the potential and kinetic energy of the links, position and velocity of the Center of Mass (CM) of the i^{th} link is needed. Due to the structure of the DH convention shown in Fig. 4.1, position of the i^{th} CM is easier to determine with respect to frame {i}. Therefore, Eq. (4.16) is used to find it with respect to the inertial (or base) frame {0}, that is:

$${}^0P_{c_i}(q) = {}^0O_i(q) + {}^0_i\mathbf{R}(q)\,{}^iP_{c_i} = {}^0_i\mathbf{T}(q)\,{}^iP_{c_i} \quad [\mathrm{m}] \tag{4.16}$$

where

${}^0P_{c_i}$	is the center of mass of the i^{th} link with respect to base frame [m]
${}^iP_{c_i}$	is the center of mass of the i^{th} link with respect to frame {i} [m]
${}^0O_i$	is the center of the i^{th} frame with respect to base frame [m]
${}^0_i\mathbf{R}$	is the 3×3 rotation matrix from frame {i} to {0}
${}^0_i\mathbf{T}$	is the 4×4 homogeneous matrix from frame {i} to {0}

It is clear that the CM position can not be determined easily due to the different shapes of the links. The CM is placed such that it can be located easily with respect to the frames already defined and therefore, make the calculations simpler. For instance, the CM of link 3 is set close to the counterweight of the link, but such that it is still aligned with the x axis of frame {3} as seen in Fig. 4.1.

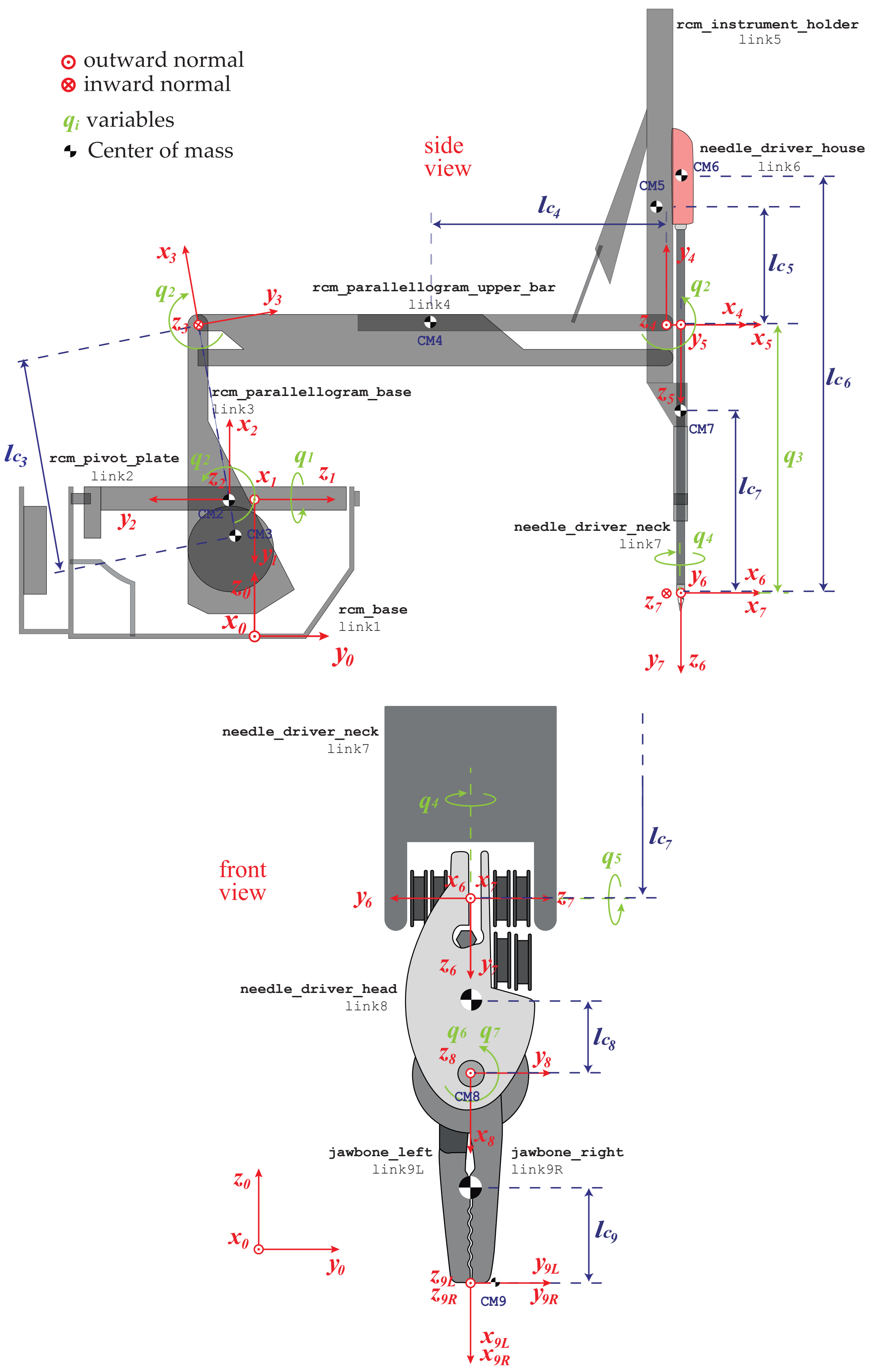


Figure 4.1: Center of mass representation of the i^{th} link altogether with the DH convention. The CM is defined with respect to the frame {i}, as it is attached to link i.

The placement of the center of mass is summarized in Eq. (4.17). Initial guesses of these distances are shown in Table 4.1. They are chosen according to the geometry of the links, but they may not be precise and are used as an initial guess of the parameters.

$$\begin{aligned}
{}^{2}P_{C_2} &= [0 \quad l_{c_2} \quad 0]^T & {}^{3}P_{C_3} &= [-l_{c_3} \quad 0 \quad 0]^T \\
{}^{4}P_{C_4} &= [-l_{c_4} \quad 0 \quad 0]^T & {}^{5}P_{C_5} &= [0 \quad 0 \quad -l_{c_5}]^T \\
{}^{6}P_{C_6} &= [0 \quad -l_{c_6} \quad 0]^T & {}^{7}P_{C_7} &= [0 \quad -l_{c_7} \quad 0]^T \\
{}^{8}P_{C_8} &= [-l_{c_8} \quad 0 \quad 0]^T & {}^{9R}P_{C_{9R}} &= [-l_{c_9} \quad 0 \quad 0]^T \\
{}^{9L}P_{C_{9L}} &= [-l_{c_9} \quad 0 \quad 0]^T
\end{aligned} \tag{4.17}$$

l_{c_2} [m]	l_{c_3} [m]	l_{c_4} [m]	l_{c_5} [m]	l_{c_6} [m]	l_{c_7} [m]	l_{c_8} [m]	l_{c_9} [m]
0	0.34	0.257	0.3	0.4	0.252	0.0045	0.0045

Table 4.1: Distances defined to locate the CM easily using the existing frames.

By differentiating the position expression in Eq. (4.16), velocity is found. Notice that the position of the i^{th} link is constant with respect to frame {i} as seen in Eq. (4.17), but it is not with respect to frame {0}:

$$ {}^{0}v_{c_i} = \frac{d\left({}^{0}P_{c_i}\right)}{dt} = {}^{0}_{i}\dot{\mathbf{R}}(q)\,{}^{i}P_{c_i} + {}^{0}\dot{O}_i(q) \quad [\mathrm{m/s}] \tag{4.18}$$

Likewise, angular velocity of the i^{th} link should be computed with respect to the inertial frame. First, notice that each link is moving with the angular velocity of the previous link plus its own rotation. Moreover, considering that in the DH convention, the angular velocity ${}^{i-1}\omega_i$ of the i^{th} link is measured about Z_{i-1}, the rotation matrix ${}^{0}_{i-1}\mathbf{R}$ is used to express it in the inertial frame and then, it is added to the angular velocity of the previous link as [Spong et al., 2004]:

$$\begin{aligned}
{}^{0}\omega_i &= {}^{0}\omega_{i-1} + {}^{0}_{i-1}\mathbf{R} \cdot {}^{i-1}\omega_i \\
&= {}^{0}\omega_1 + {}^{0}_{1}\mathbf{R} \cdot {}^{1}\omega_2 + {}^{0}_{2}\mathbf{R} \cdot {}^{2}\omega_3 + \cdots + {}^{0}_{i-1}\mathbf{R} \cdot {}^{i-1}\omega_i \quad [\mathrm{rad/s}]
\end{aligned} \tag{4.19}$$

Now, the angular velocity of each link with respect to the previous frame ${}^{i-1}\omega_i$ is related to generalized velocities $\dot{q}$ as follows:

$$\begin{aligned}
{}^{0}\omega_1 &= \begin{bmatrix}0 & 0 & 0\end{bmatrix}^T & {}^{1}\omega_2 &= \begin{bmatrix}0 & 0 & \dot{q}_1\end{bmatrix}^T \\
{}^{2}\omega_3 &= \begin{bmatrix}0 & 0 & \dot{q}_2\end{bmatrix}^T & {}^{3}\omega_4 &= \begin{bmatrix}0 & 0 & \dot{q}_2\end{bmatrix}^T \\
{}^{4}\omega_5 &= \begin{bmatrix}0 & 0 & \dot{q}_2\end{bmatrix}^T & {}^{5}\omega_6 &= \begin{bmatrix}0 & 0 & 0\end{bmatrix}^T \\
{}^{5}\omega_6 &= \begin{bmatrix}0 & 0 & \dot{q}_4\end{bmatrix}^T & {}^{7}\omega_8 &= \begin{bmatrix}0 & 0 & \dot{q}_5\end{bmatrix}^T \\
{}^{8}\omega_{9L} &= \begin{bmatrix}0 & 0 & \dot{q}_6\end{bmatrix}^T & {}^{8}\omega_{9R} &= \begin{bmatrix}0 & 0 & \dot{q}_7\end{bmatrix}^T
\end{aligned} \tag{4.20}$$

Note that link 1 angular velocity is zero as it is the base link, whereas link 6 only has a sliding movement $\dot{q}_3$ with respect to frame {5}.

4.2.2 Inertia Tensor

The inertia tensor $\mathcal{I}$ is expressed in the inertial frame. However, it is not easy to compute it as it depends on the configuration of the robot. Whereas the inertia tensor $\mathbf{I}$ expressed in a frame attached to each link's CM (body frame) is constant. Then, both inertia tensor $\mathcal{I}_i$ and $\mathbf{I}_i$ can be related by using the rotation matrix between the inertial and body frame [Spong et al., 2004]:

$$\mathcal{I}_i = {}^0_i\mathbf{R} \cdot \mathbf{I}_i \cdot {}^0_i\mathbf{R}^T \quad [\mathrm{kg\,m}^2] \tag{4.21}$$

Now, note that frame {i} is attached to link i according to the DH convention. However, this frame is not placed at the CM but at the joint. Therefore, a new frame {i'} is placed at the CM, which is parallel to the frame {i}, as seen in Fig. 4.2. As both frames are parallel, the already known rotation matrix ${}^0_i\mathbf{R}$ can be used in Eq. (4.21) to convert the body inertia tensor $\mathbf{I}_i$ to the inertial frame of reference.

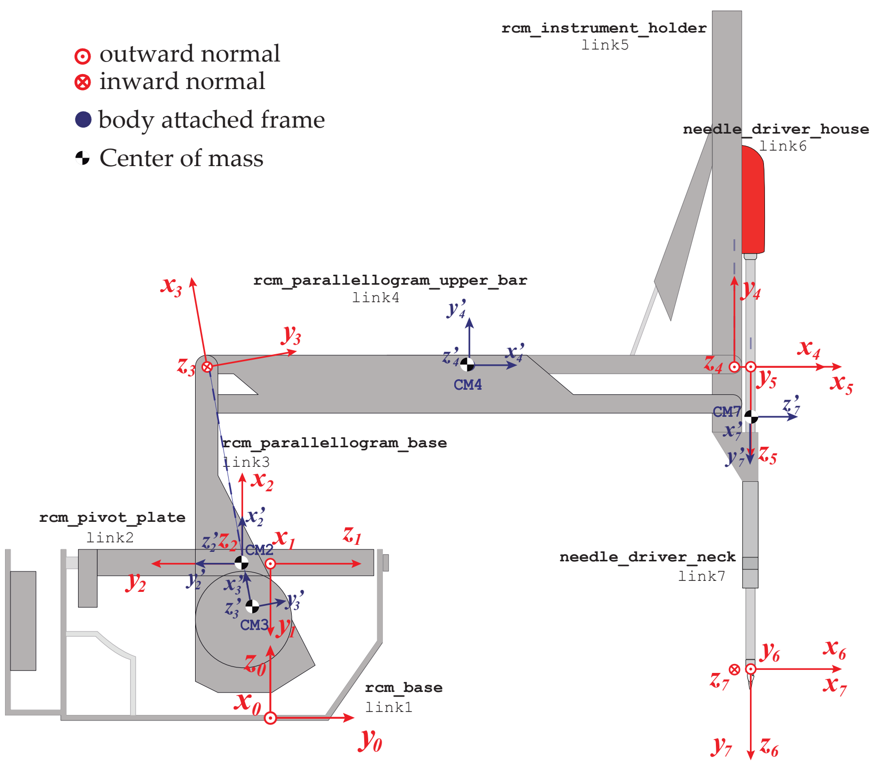


Figure 4.2: A new set of body frames are placed at the center of mass of the i^{th} link parallel to the known frame {i}. An example is shown for link 2, 3, 4 and 7. The same concept is applied to the remaining links.

The constant inertia tensor **I** is a symmetric 3x3 matrix that depends on the mass distribution of the object and its geometry. Many tables exist for common shapes that can ease the process. In order to define it, let $\rho(x, y, z)$ be the mass density of this object, then the inertia tensor **I** is computed as:

$$\mathbf{I} = \begin{pmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{pmatrix} \tag{4.22}$$

where the diagonal coefficients are called the principal moments of inertia about the x, y and z axis and are defined as:

$$\begin{aligned} I_{xx} &= \int_V (y^2 + z^2)\rho(x, y, z)\, dV = \int_X \int_Y \int_Z (y^2 + z^2)\rho(x, y, z)\, dx\, dy\, dz \\ I_{yy} &= \int_V (x^2 + z^2)\rho(x, y, z)\, dV = \int_X \int_Y \int_Z (x^2 + z^2)\rho(x, y, z)\, dx\, dy\, dz \\ I_{zz} &= \int_V (x^2 + y^2)\rho(x, y, z)\, dV = \int_X \int_Y \int_Z (x^2 + y^2)\rho(x, y, z)\, dx\, dy\, dz \end{aligned}$$

where the integrals are computed over the region in space occupied by the rigid body defined by X, Y, Z. The off diagonal elements are called the cross products of inertia. If the mass distribution of the body is symmetric with respect to the body frame, then these elements are 0, otherwise they are computed as:

$$\begin{aligned} I_{xy} = I_{yx} &= -\int_V xy\, \rho(x, y, z)\, dV = -\int_X \int_Y \int_Z xy\, \rho(x, y, z)\, dx\, dy\, dz \\ I_{xz} = I_{zx} &= -\int_V xz\, \rho(x, y, z)\, dV = -\int_X \int_Y \int_Z xz\, \rho(x, y, z)\, dx\, dy\, dz \\ I_{yz} = I_{zy} &= -\int_V yz\, \rho(x, y, z)\, dV = -\int_X \int_Y \int_Z yz\, \rho(x, y, z)\, dx\, dy\, dz \end{aligned}$$

where the integrals are computed over the region in space occupied by the rigid body defined by X, Y, Z. Estimating the inertia matrix is a difficult task. Therefore, for the purpose of having a simple estimation of this matrix, all the links of the da Vinci are assumed to be cylinders of radius r and height h, with uniform mass density and with the CM at the geometrical center of the cylinder. Fig. 4.3 shows an illustration of how this is done with links 3 and 4.

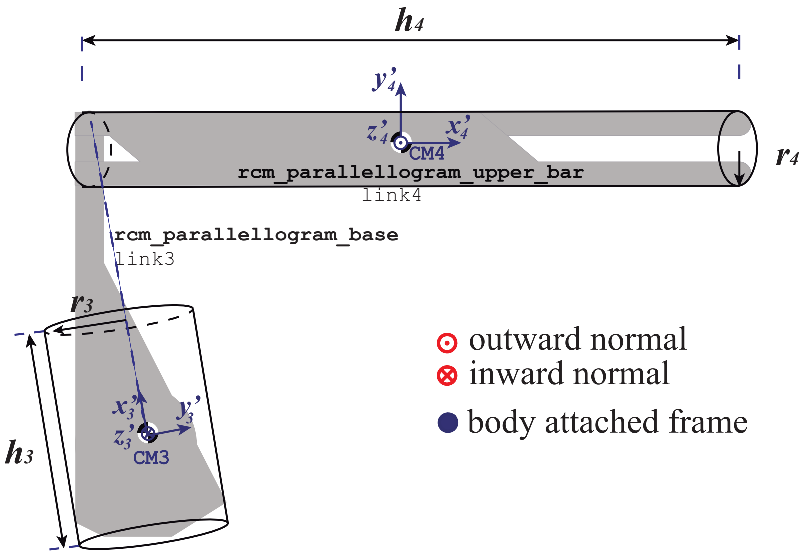


Figure 4.3: Example of link 3 and 4 modeled as cylinders.

Then, the inertia tensor **I** of one cylinder, expressed in the body frame, is proven in [Serway, 1986] to be:

$$\mathbf{I}_i = \begin{pmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{pmatrix} = \begin{pmatrix} \frac{1}{12}mh^2 + \frac{1}{4}mr^2 & 0 & 0 \\ 0 & \frac{1}{12}mh^2 + \frac{1}{4}mr^2 & 0 \\ 0 & 0 & \frac{1}{2}mr^2 \end{pmatrix} \tag{4.23}$$

where the z axis goes along the height of the cylinder, but due to the existing frames it is possible that the axis change with the links (eg. link 3 of Fig. 4.3 has the x axis aligned with the height). From now on, I_{xx}, I_{yy} and I_{zz} will be denoted as I_x, I_y and I_z respectively.

Finally, determining the mass of the links could be done by dismantling the manipulator and weighing all the links, however it is not an easy task. Therefore, intuitive guesses are given in Table 4.2. On the other hand, the height and radius of the cylinders are chosen according to the dimensions of the links.

	link 1	link 2	link 3	link 4	link 5	link 6	link 7	link 8	link 9R/9L
mass [kg]	0	0.5	4	0.7	0.7	0.079	0.079	0.04	0.01
height [m]	0	0.22	0.19	0.515	0.26	0.1	0.4	0.009	0.009
radius [m]	0	0.035	0.1	0.07	0.035	0.07	0.035	0.005	0.005

Table 4.2: Mass and dimension of the cylinder (height and radius) chosen for each link.

Note that Table 4.2 gives an initial guess of the parameters which are used for the model simulation.

4.3 Friction Model

In all mechanical system there is a loss due to friction. For this reason, the friction on the joints of the manipulator should be defined. First of all, the viscous friction is added. It is a very simple model for friction, where the friction torque is proportional to the velocity of the joint [Craig, 2009, Ch. 6]:

$$\tau_f = v \cdot \dot{q} \tag{4.24}$$

where v is a viscous-friction constant of units [Nm/(rad/s)] if joint is **revolute** or [N/(m/s)] if it is **prismatic**. Another simple model of friction that could be added is the Coulomb friction, which is a constant torque that is opposed to the direction of the joint velocity [Craig, 2009, Ch. 6]:

$$\tau_f = c \cdot \text{sign}(\dot{q}) \tag{4.25}$$

where c is the coulomb-friction constant of units [Nm] if joint is **revolute** or [N] if it is **prismatic**. Usually, friction can also show dependency on the joint position q but it is not considered in this model. Therefore, the total friction torque (or force) can now be written as one friction terms which is velocity dependent:

$$F(\dot{q}) = v \cdot \dot{q} + c \cdot \text{sign}(\dot{q}) \tag{4.26}$$

However, other effects such as the breakaway friction or Stribeck friction can also be present in the robot [Dupont, 1990]. A representation of viscous and coulomb friction altogether with other friction models is shown in Fig.4.4:

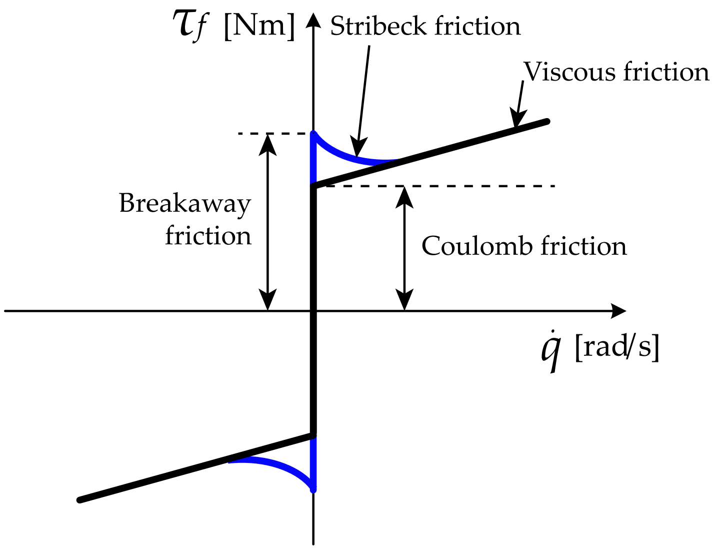


Figure 4.4: Representation of different friction models for a revolute joint. Only Coulomb and viscous friction is used in this thesis.

As it is previously shown in Eq. (4.9), this friction is added to the manipulator dynamics as a vector $F(\dot{q})$ of non-conservative torques (forces if joint is prismatic) altogether with actuator and external torques:

$$\tau = \mathbf{M}(q)\ddot{q} + V(q, \dot{q}) + G(q) + F(\dot{q}) + \tau_{ext} \tag{4.27}$$

The actuators that create the joint torques τ are DC motors. Once the motor dynamics, developed in Appendix A, are included, Eq. (4.27) is written as:

$$\boldsymbol{\eta}\,\tau_m = (\mathbf{J}_m + \mathbf{M}(q))\ddot{q} + V(\dot{q}, q) + F(\dot{q}) + G(q) + \tau_{ext} \tag{4.28}$$

where τ_m is a vector of motor torques and $\boldsymbol{\eta}$ is a diagonal matrix with motor gear ratios η in the diagonal. $\mathbf{J}_m$ is the motor inertia diagonal matrix with diagonal elements $\eta_i^2 J_{m_i}$, where J_{m_i} is the moment of inertia of the motor [kg m^2]. Finally, assuming that the motor inertia is much smaller compared to the mass and inertia terms of the manipulator, $\mathbf{J}_m$ can be neglected and Eq. (4.28) becomes:

$$\boldsymbol{\eta}\,\tau_m = \mathbf{M}(q)\ddot{q} + V(\dot{q}, q) + F(\dot{q}) + G(q) + \tau_{ext} \tag{4.29}$$

4.4 Overview of the Manipulator Dynamics

The Euler Lagrange formalism is used to obtain a dynamical model of the manipulator based on an energy study of the system. To do so, terms such as center of mass position, moments of inertia or angular velocity should be expressed in the inertial frame as explained along this chapter. It implies that the complexity of these terms will increase considerably when the number of joints increases. For instance, the center of mass of the last link will depend on all the generalized coordinates q. Therefore, the dynamics cannot be developed by hand and the software Maple is used to find the symbolic expression [MapleSoft, 2014]. The procedure followed to find the manipulator dynamics is summarized in the following steps:

1. Generalized coordinates q and its first and second time derivative are defined as time dependent variables ($\dot{q}$ and $\ddot{q}$ respectively).

2. Transformation matrices from consecutive frames ${}^{i-1}_{i}\mathbf{T}$ are defined according to the DH convention from Chapter 3. Then, the transformation matrix from frame {i} to the base frame {0} is computed. The rotation matrix ${}^{0}_{i}\mathbf{R}$ is also obtained.
$${}^{0}_{i}\mathbf{T} = {}^{0}_{1}\mathbf{T}\,{}^{1}_{2}\mathbf{T} \cdots {}^{i-1}_{i}\mathbf{T} \tag{4.30}$$

3. The CM position of link i is defined with respect to frame {i}. Then, it is transformed to base frame {0} using transformation matrix ${}^{0}_{i}\mathbf{T}$ found in Eq. (4.30).
$${}^{0}P_{c_i}(q) = {}^{0}_{i}\mathbf{T}(q)\,{}^{i}P_{c_i} \tag{4.31}$$

4. Time derivative of position with respect to base frame $\{0\}$ is performed:

$$^0v_{c_i} = \frac{d\left(^0P_{c_i}\right)}{dt} \tag{4.32}$$

5. Angular velocity of each link is computed. For link i, that is:

$$^0\omega_i = {^0\omega_{i-1}} + {^0_{i-1}\mathbf{R}}\,{^{i-1}\omega_i} \tag{4.33}$$

6. The inertia tensor of each link i with respect to its body frame ($\mathbf{I}_i$) is defined considering links as cylinders. Then, it is computed with respect to the base frame ($\mathcal{I}_i$).

$$\mathcal{I}_i = {^0_i\mathbf{R}}\,\mathbf{I}_i\,{^0_i\mathbf{R}^T} \tag{4.34}$$

7. Both translational and rotational kinetic energy of the N links is calculated, then they are added to find the total kinetic energy.

$$k = \sum_{i=1}^{N}\left(\frac{1}{2}m_i\,{^0v_{c_i}^T}\,{^0v_{c_i}} + \frac{1}{2}\,{^0\omega_i^T}\,\mathcal{I}_i\,{^0\omega_i}\right) \tag{4.35}$$

8. The potential energy of the N links is computed. Then, the total potential energy is found as:

$$u = \sum_{i=1}^{N} m_i\,g^T\,{^0P_{c_i}} \tag{4.36}$$

9. The dynamics are derived using the Euler Lagrange formalism altogether with the total potential and kinetic energy. Notice that even though there are 10 links (including the base), links 3, 4 and 5 move as a parallelogram with the same joint angle q_3. Therefore, there are 7 generalized coordinates q and one expression is computed per q_i:

$$\frac{d}{dt}\frac{\partial k}{\partial \dot{q}_i} - \frac{\partial k}{\partial q_i} + \frac{\partial u}{\partial q_i} = Q_i \tag{4.37}$$

10. From the expression obtained in step 9, the matrix $\mathbf{M}(q)$ is found by taking the elements that involve the second derivative of the generalized coordinates q, the vector $V(q,\dot{q})$ is found by taking the elements that consist of quadratic terms involving the first derivative of q and finally, $G(q)$ are the terms that depend on gravity and the spring force. Then, the system equation is written:

$$Q = \mathbf{M}(q)\ddot{q} + V(q,\dot{q}) + G(q) \tag{4.38}$$

11. Finally, non conservative torques are included to the expression found, that is external torques τ_{ext} and friction torques $F(\dot{q})$ that are opposed to the actuator torque τ. Moreover, actuator torques τ can be expressed as motor torques τ_m. Notice that when the joint is prismatic, these terms are forces and not torques.

$$\tau = \boldsymbol{\eta}\tau_m = \mathbf{M}(q)\ddot{q} + V(q,\dot{q}) + G(q) + F(\dot{q}) + \tau_{ext} \tag{4.39}$$

The Maple code, where the symbolic expression of the dynamics is derived, can be found in Appendix C. The equation obtained is now used for both system identification and model simulation.

4.5 MATLAB Simulation

The dynamical equation for the daVinci robot developed in this chapter is used to build up a simulation model. The model uses estimated values of the physical parameters defined in Tables 4.1 and 4.2.

4.5.1 Simulation Model

The simulation of the daVinci robot is divided into three main parts:

- **Parameter Initialization**: a MATLAB script is used to define the parameters of the model. Therefore, it is possible to modify these values in the model.
- **Simulink Model**: The model is built in Simulink, where the daVinci dynamical equation that is simulated is given by:

  $$\tau = \mathbf{M}(q)\ddot{q} + V(\dot{q},q) + F(\dot{q}) + G(q) + \tau_{ext} \tag{4.40}$$

  In order to use it in Simulink, it is rewritten as follows:

  $$\ddot{q} = \mathbf{M}(q)^{-1}\left(\tau - V(\dot{q},q) - F(\dot{q}) - G(q) - \tau_{ext}\right) \tag{4.41}$$

  Eq. (4.41) is written in an S-Function with $\ddot{q}$ as an output. Finally, a **Second Order Integrator** block in Simulink is used to integrate Eq. (4.41) in order to compute q and $\dot{q}$. The choice of a Second Order Integrator before two First Order Integrators is because it allows to simulate a hard stop easier (i.e. when the robot reaches its physical limits). As explained in [Rouleau, 2014], if two integrators in series are used, then a logic should be built in order to reset/disable/saturate the integrators. On the contrary, the second order integrator already brings this option, such that when q hits the physical limits, then $\dot{q}$ is set to zero and stops integrating without the need of extra logic. Moreover, in [Rouleau, 2014] it was shown how the Second Order Integrator is solving the simulation in 25% less time steps than using First

Order Integrators. This will have a significant impact on the performance as the model for the da Vinci Robot is quite large. The implementation in Simulink is shown in Fig. 4.5.

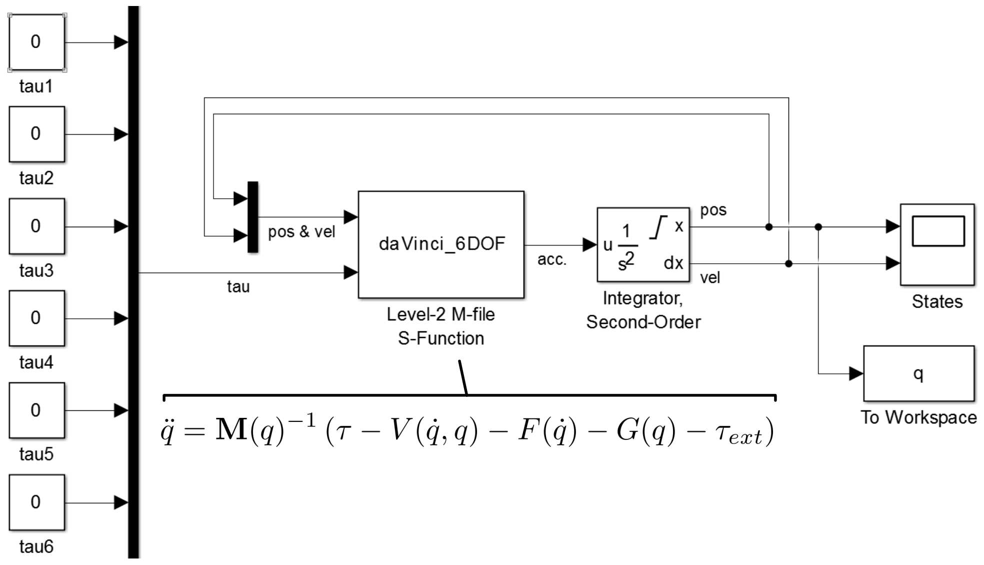

0
tau1
0
tau2
0
tau3
0
tau4
0
tau5
0
tau6
pos & vel
tau
daVinci_6DOF
Level-2 M-file S-Function
acc.
Integrator, Second-Order
pos
vel
States
q
To Workspace
$\ddot{q} = \mathbf{M}(q)^{-1}\left(\tau - V(\dot{q},q) - F(\dot{q}) - G(q) - \tau_{ext}\right)$

Figure 4.5: Open loop da Vinci model with a Second Order Integrator in Simulink.

- **Graphical Representation**: Results from the simulation are shown graphically as in Fig. 4.6. The kinematic model from Chapter 3 is used to plot the simulated results by using Robotics Toolbox in MATLAB [Corke, 2011].

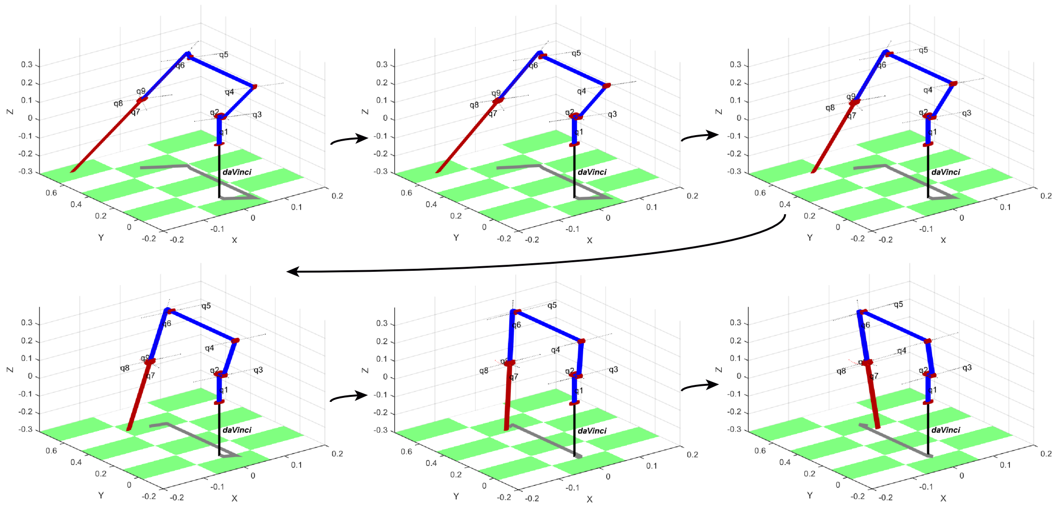


Figure 4.6: Example of graphical representation of the da Vinci simulation results.

4.6 Conclusions

In this chapter, a model for the daVinci robot is found by using the Euler-Lagrange formulation, based on the kinetic and potential energy of the robot. Some assumptions are done in order to derive the model:

- The links are assumed to be rigid bodies.
- The links are considered to be cylinders, as it is easier when computing the link inertia.
- The center of mass of each link is chosen to be aligned with the frame of each link.

To find this model is of great importance, as it is later needed when estimating contact forces at the daVinci robot in Chapter 7. Furthermore, a simulation is built in Simulink, where the nonlinear multivariable model is defined by using initial guesses of the parameters. Therefore, the results obtained in simulation are not exact as in the real da Vinci robot. Nonetheless, the simulation gives the opportunity to have a test bench for the da Vinci robot and work with it without having the physical robot.

Chapter 5

Controller for Simulation Model

The purpose of this chapter is to design a controller for the simulation of the da Vinci model developed in Chapter 4. First, the Lyapunov theory is used to prove the stability of a PD controller and then, it is verified in the simulation. Finally, an integral action is also added to the PD controller.

5.1 PD Control and Stability Analysis

A simple independent PD-control scheme for setpoint tracking is known to work in the general case of a system of the form of Eq. (5.1) [Spong et al., 2004].

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

Its asymptotically stablility can be proven by using the Lyapunov theory explained in [Khalil, 2002] and a short introduction is now given. First, consider the system

$$\dot{x} = f(x) \tag{5.2}$$

where $f : D \rightarrow \mathbb{R}^n$ is continuous on the domain $D \subseteq \mathbb{R}^n$. Suppose that $\bar{x} \in D$ is an equilibrium point of (5.2), that is $f(\bar{x}) = 0$. Then, let $V : D \rightarrow \mathbb{R}$ be a Lyapunov function candidate, which is continuously differentiable, that proves asymptotically stability, that is [Khalil, 2002]:

$$\begin{aligned} V(\bar{x}) = 0 \quad &\text{and} \quad V(x) > 0 \quad \text{in} \quad D - \{\bar{x}\} \\ \dot{V}(\bar{x}) = 0 \quad &\text{and} \quad \dot{V}(x) < 0 \quad \text{in} \quad D - \{\bar{x}\} \end{aligned} \tag{5.3}$$

In order to prove asymptotically stability for the simulation model, the Lyapunov candidate in [Spong et al., 2004] is used and their methodology is followed along this section. First, an independent joint PD-control can be expressed in the following vector form:

$$\tau = \mathbf{K}_p(q_{ref} - q) - \mathbf{K}_{\,d}\dot{q} = \mathbf{K}_p\tilde{q} - \mathbf{K}_d\dot{q} \tag{5.4}$$

where

$\dot{q}$	is the joint velocity [m/s]
q_{ref}	is a constant position reference [m]
$\tilde{q}$	is the joint position error [m]
$\mathbf{K}_p$	is a positive definite diagonal matrix of proportional gains [-]
$\mathbf{K}_d$	is a positive definite diagonal matrix of derivative gains [-]

The system model is defined without Coulomb friction as:

$$\tau = \mathbf{M}(q)\ddot{q} + V(\dot{q}, q) + F(\dot{q}) + G(q) \tag{5.5}$$

where $F(\dot{q})$ consist of viscous friction. The velocity dependent vector $V(\dot{q}, q)$ and the viscous friction $F(\dot{q})$ should be rewritten in a different form, such that the system dynamics becomes [Spong et al., 2004]:

$$\tau = \mathbf{M}(q)\ddot{q} + \mathbf{C}(\dot{q}, q)\dot{q} + \mathbf{F}\dot{q} + G(q) \tag{5.6}$$

where $\mathbf{C}(\dot{q}, q)$ is known as the Coriolis and centripetal coupling matrix and $\mathbf{F}$ is a positive definite diagonal matrix with the viscous friction constant in the diagonal. Furthermore, it is known that the gravity term $G(q)$ will give a stationary error when using a PD controller [Spong et al., 2004]. However, assuming that $G(q)$ is known, it can be removed from the system if it is included in the control law as:

$$\tau = \mathbf{K}_p\tilde{q} - \mathbf{K}_d\dot{q} + G(q) \tag{5.7}$$

which yields to the following closed loop system dynamics:

$$\mathbf{K}_p\tilde{q} - \mathbf{K}_d\dot{q} = \mathbf{M}(q)\ddot{q} + \mathbf{C}(\dot{q}, q)\dot{q} + \mathbf{F}\dot{q} \tag{5.8}$$

whose equilibrium point is:

$$\begin{aligned} \tilde{q} = 0 \quad &\rightarrow \quad q = q_{ref} \\ \dot{q} &= 0 \end{aligned}$$

Now, lets consider the following Lyapunov function candidate in [Spong et al., 2004]:

$$V = \underbrace{\frac{1}{2}\dot{q}^T\mathbf{M}(q)\dot{q}}_{\text{Kinetic Energy}} + \underbrace{\frac{1}{2}\tilde{q}^T\mathbf{K}_p\tilde{q}}_{\text{Controller Term}} \tag{5.9}$$

where the first term is the kinetic energy of the daVinci robot and the second term is related to the proportional part of the feedback controller. Now, considering:

- $\mathbf{M}(q)$ is a symmetric positive definite matrix of mass and inertia terms.

- $\mathbf{K}_p$ is a positive definite diagonal matrix of controller gains.

Then, it can be concluded that V is a positive function except for the equilibrium point $\dot{q} = 0$ and $q = q_{ref}$, where V is zero. If now it can be proven that the Lyapunov function V is decreasing toward zero along any trajectory of the system, then it means that the system always goes towards the equilibrium point. Therefore, if $\dot{V}$ is always negative, then V will always decrease. Then, the time derivative of V is given by:

$$\begin{aligned}\dot{V} &= \underbrace{\frac{1}{2}\dot{q}^T(\mathbf{M}(q) + \mathbf{M}(q)^T)\ddot{q} + \frac{1}{2}\dot{q}^T\dot{\mathbf{M}}(q)\dot{q}}_{\text{Time Derivative of Kinetic Energy}} - \dot{q}^T\mathbf{K}_p\tilde{q} \\ &= \underbrace{\dot{q}^T\mathbf{M}(q)\ddot{q} + \frac{1}{2}\dot{q}^T\dot{\mathbf{M}}(q)\dot{q}}_{\substack{\text{Time Derivative of} \\ \text{Kinetic Energy}}} - \dot{q}^T\mathbf{K}_p\tilde{q}\end{aligned} \tag{5.10}$$

Notice that q_{ref} in $\tilde{q}$ is constant and the terms $\mathbf{M}(q)$ and $\mathbf{M}(q)^T$ are symmetric. Now, the term $\mathbf{M}(q)\ddot{q}$ can be obtained from the system equation in (5.6), then substituting into Eq. (5.10) yields:

$$\begin{aligned}\dot{V} &= \dot{q}^T(\mathbf{K}_p\tilde{q} - \mathbf{K}_d\dot{q} - \mathbf{C}(q,\dot{q})\dot{q} - \mathbf{F}\dot{q}) + \frac{1}{2}\dot{q}^T\dot{\mathbf{M}}(q)\dot{q} - \dot{q}^T\mathbf{K}_p\tilde{q} \\ &= -\dot{q}^T\mathbf{K}_d\dot{q} - \dot{q}^T\mathbf{F}\dot{q} + \frac{1}{2}\dot{q}^T\left(\dot{\mathbf{M}}(q) - 2\mathbf{C}(q,\dot{q})\right)\dot{q}\end{aligned} \tag{5.11}$$

where $\dot{\mathbf{M}}(q) - 2\mathbf{C}(q,\dot{q})$ is a skew symmetric as explained in [Spong et al., 2004], which implies that any vector $\dot{q}$ fulfills:

$$\dot{q}^T\left(\dot{\mathbf{M}}(q) - 2\mathbf{C}(q,\dot{q})\right)\dot{q} = 0 \tag{5.12}$$

Then, by using the skew symmetric property in (5.12), the time derivative of the Lyapunov function becomes:

$$\begin{aligned}\dot{V} &= -\dot{q}^T\mathbf{K}_d\dot{q} - \dot{q}^T\mathbf{F}\dot{q} \\ &= -\dot{q}^T\left(\mathbf{K}_d + \mathbf{F}\right)\dot{q} \leq 0\end{aligned} \tag{5.13}$$

Notice that $\mathbf{F}$ and $\mathbf{K}_d$ are positive definite diagonal matrices of viscous friction coefficients and derivative gains respectively. But Eq. (5.13) only shows that $\dot{V}$ is zero when $\dot{q} = 0$, while q does not necessary have to be q_{ref}. Therefore it should be proven that $\dot{V}$ can only be zero at the equilibrium point $\dot{q} = 0$ and $q = q_{ref}$ and not any other point. To do so, the LaSalle Invariance Principle is used [Khalil, 2002]. If it can be proven that only the equilibrium point can stay identically in $\dot{V}(t) = 0$, then the equilibrium point is asymptomatically stable.

Suppose that the Lyapunov function reaches zero at some time t, then:

$$\dot{V}(t) = 0 \quad \forall t \tag{5.14}$$

Then, from the Lyapunov derivative in Eq. (5.13) it can be seen that $\dot{q}(t) = 0$ and hence $\ddot{q}(t) = 0$. Then, by setting $\ddot{q} = 0$ and $\dot{q} = 0$ in the closed loop system in Eq. (5.8), it results in:

$$\mathbf{K}_p \tilde{q} = 0 \tag{5.15}$$

which implies that only the equilibrium point $\tilde{q} = 0$ ($q = q_{ref}$) and $\dot{q} = 0$ is the solution for the system that stays identically in $\dot{V} = 0$. Therefore, by the LaSalle Invariance Principle it can be stated that the equilibrium point is asymptomatically stable. Moreover, if V is radially unbounded then $\bar{x}$ is globally asymptotically stable, that is:

$$V(x) \to \infty \quad \text{as} \quad ||x|| \to \infty \tag{5.16}$$

which is fulfilled in the Lyapunov candidate of Eq. (5.9). Therefore, the **equilibrium point is globally asymptomatically stable**.

5.2 Controller Implementation and Results

After asymptomatically stability is proven in Sec. 5.1, now it can be verified for the daVinci model. Therefore, once the simulation is built in Simulink, independent PD controllers are added to it and the closed loop model is shown in Fig. 5.1, where the control law with gravity compensation is defined as:

$$\tau = \mathbf{K}_p \tilde{q} - \mathbf{K}_d \dot{q} + G(q) \tag{5.17}$$

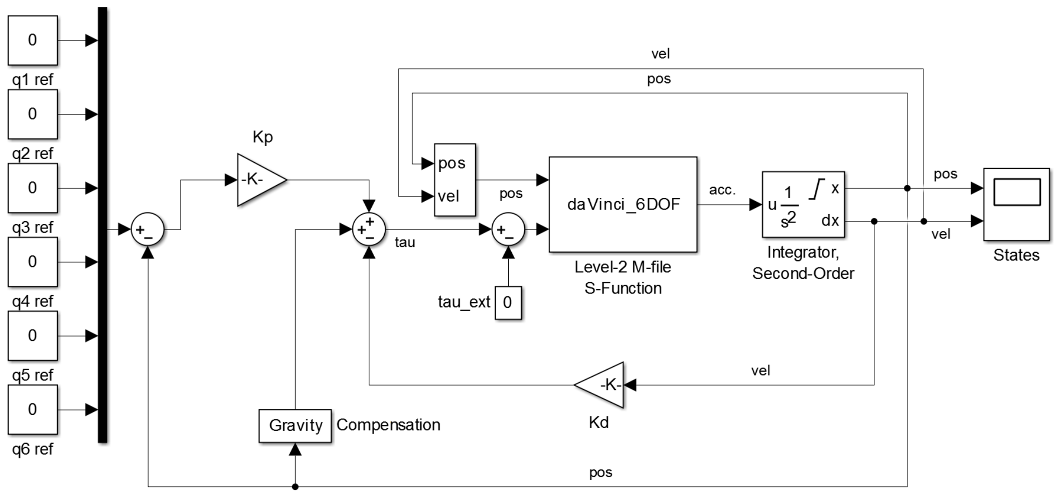


Figure 5.1: Closed loop Simulink implementation with the PD controller.

Notice that the gravity term $G(q)$ is computed at every iteration. Moreover, there are no limitation on the proportional and derivative gains $\mathbf{K}_p$ and $\mathbf{K}_d$ except for being positive definite. Therefore, priority has been given to verify the asymptotically stability, instead of adjusting the gains. The values chosen are shown in Table 5.1.

Joint Name	*Hand Roll*	*Hand Pitch*	*Ins. Slide*	*Ins. Roll*	*Ins. Pitch*	*Ins. Yaw*
K_p	500	500	500	10	10	1
K_d	50	50	40	2	0.2	0.02

Table 5.1: Table of proportional K_p and derivative K_d gains for each joint.

First, non-zero initial conditions are given to each joint position q, while joint velocities $\dot{q}$ are set to zero as shown in Table 5.2. Finally, references for all joints are set to zero.

Joint Name	*Hand Roll*	*Hand Pitch*	*Ins. Slide*	*Ins. Roll*	*Ins. Pitch*	*Ins. Yaw*
q_{init}	$-\pi/8$	$\pi/8$	0.2	$-\pi/4$	$-\pi/4$	$-\pi/4$
$\dot{q}_{init}$	0	0	0	0	0	0
q_{ref}	0	0	0	0	0	0

Table 5.2: Table of initial condition to the simulation.

The system response is shown in Fig 5.2, where both position and velocity goes to zero very fast. Therefore, it can be seen how with gravity compensation, any independent PD controller would be able to control the system for setpoint tracking. Moreover, it is also interesting to see how the Lyapunov function is always decreasing and eventually is zero when system reaches the equilibrium point as seen in Fig. 5.3.

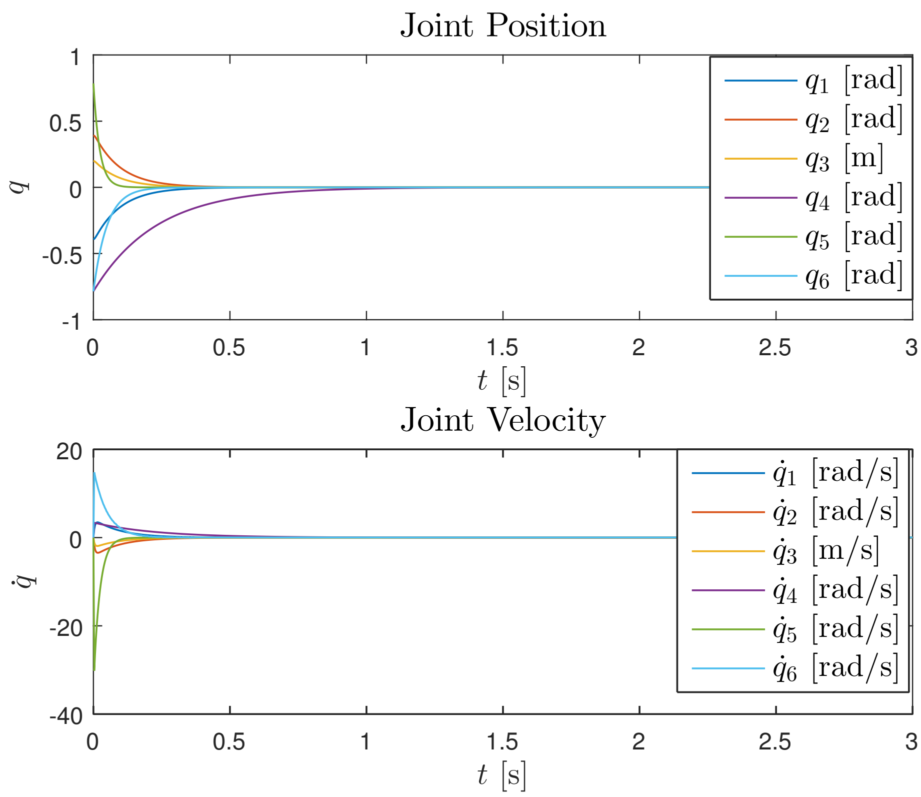


Figure 5.2: Closed loop response with PD controller and gravity compensator.

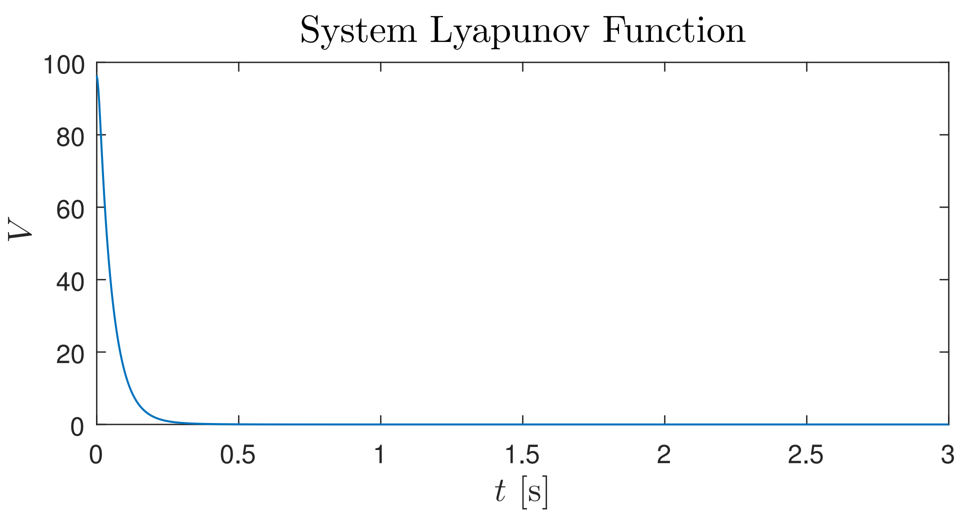


Figure 5.3: The Lyapunov function of the system which is always decreasing until it reaches zero when system is at the equilibrium point.

Another test is carried out to see how the PD controller reaches a non-zero reference. First, zero initial conditions are given to each joint position q and velocities $\dot{q}$. Finally, different references for all joints are given.

Joint Name	*Hand Roll*	*Hand Pitch*	*Ins. Slide*	*Ins. Roll*	*Ins. Pitch*	*Ins. Yaw*
q_{init}	0	0	0	0	0	0
$\dot{q}_{init}$	0	0	0	0	0	0
q_{ref}	$-\pi/8$	$\pi/8$	0.2	$-\pi/4$	$-\pi/4$	$-\pi/4$

Table 5.3: Table of initial condition to the simulation.

As seen before, the controller can follow any reference and all joints reach the setpoint.

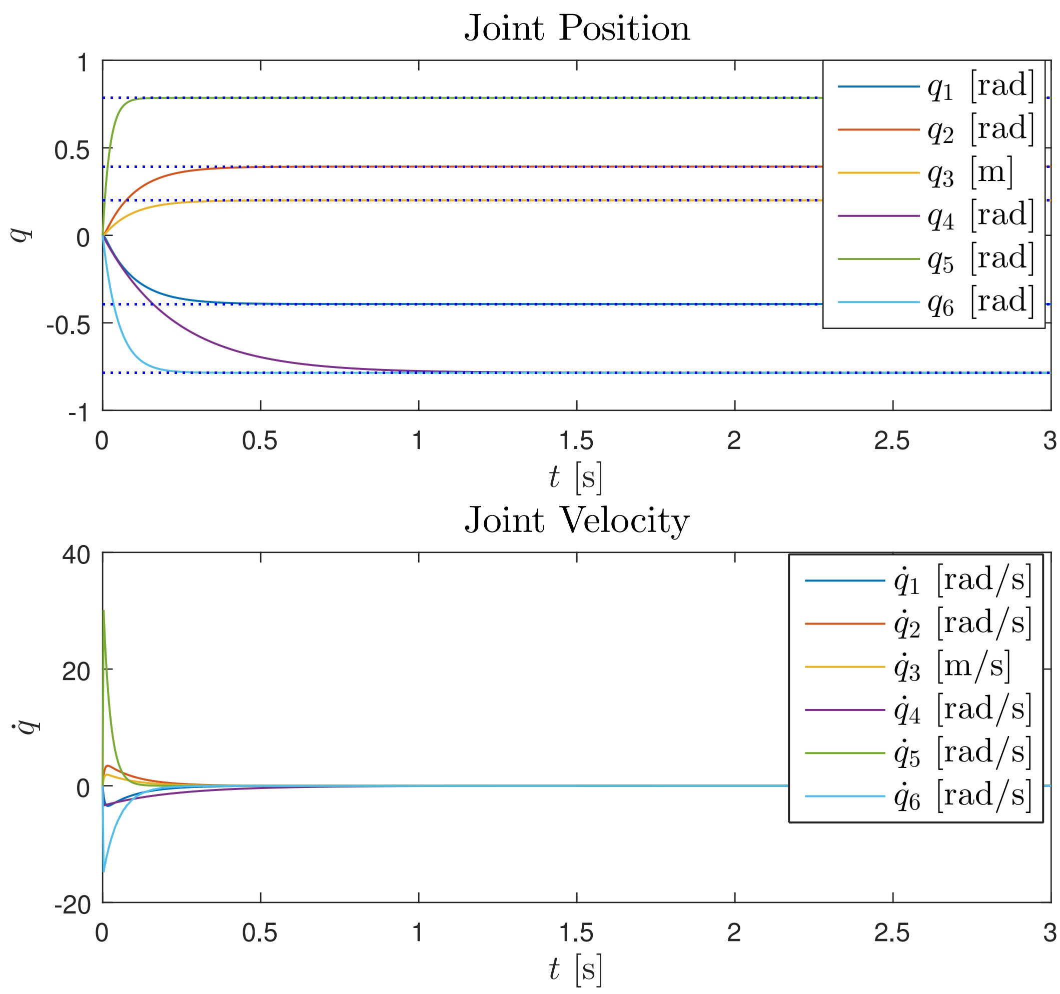


Figure 5.4: Closed loop response with PD controller and gravity compensator.

In practice, it is possible that the gravity term $G(q)$ is not known exactly. In such cases, a PD controller alone will not be able to reach the reference given and a small steady state error will appear. As an alternative to a PD controller with gravity compensator, one could add an integral action to remove this offset. For this reason, PID controllers are included in the simulation as shown in Fig. 5.5 in cases the gravity compensation term is not precise.

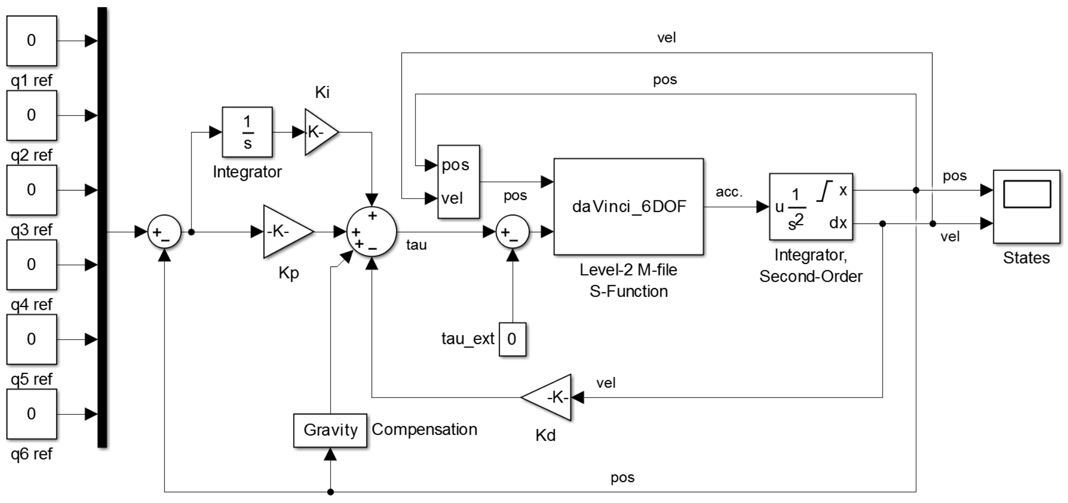


Figure 5.5: Closed loop Simulink implementation of the PID controller.

The final gains chosen are shown in Table 5.4.

Joint Name	Hand Roll	Hand Pitch	Ins. Slide	Ins. Roll	Ins. Pitch	Ins. Yaw
K_p	500	500	500	10	10	1
K_i	50	50	40	2	1	0.1
K_d	50	50	40	2	0.2	0.02

Table 5.4: Table of proportional K_p, derivative K_d and integral K_i gains for each joint.

The same test to Fig. 5.4 is now carried out, where the initial condition summarized in Table 5.3 are used. Moreover, the mass in the gravity compensator is 0.9 times the mass in the da Vinci model in order to see how the integral action correct the steady state error. In fact, it can be seen how the integral action makes the system response faster.

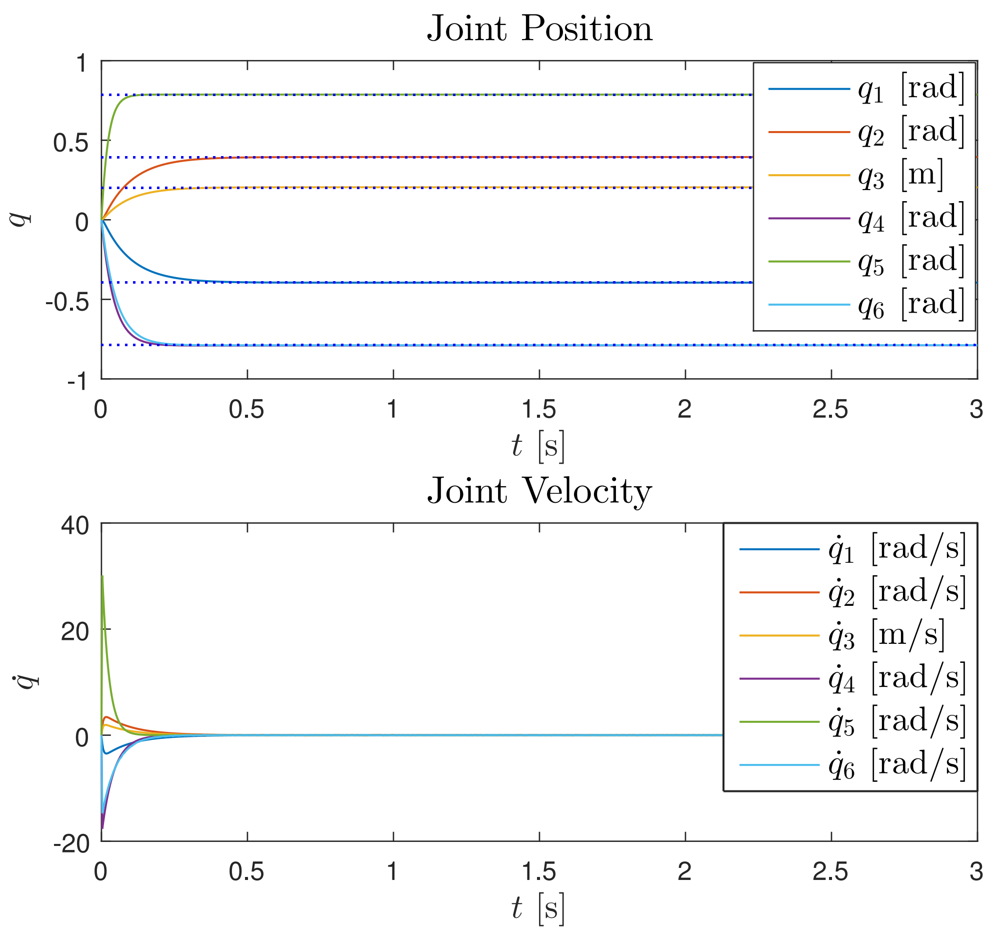


Figure 5.6: Closed loop response with PID controller.

5.3 Conclusions for the Controller

In this chapter, a PD controller is designed for the model found in Chapter 4. Then, asymptotically stability of this PD controller is proved by using Lyapunov theory and finally, it is tested in simulation.

To do so, it was necessary to compensate for gravity. In case the gravity term is not known exactly, in practice it means that there will be a steady state error. As an alternative to the PD control, an integral action is added such that the PID controller is able to cope with the steady state error due to gravity.

Chapter 6

Robot Parameters Estimation

A detailed dynamic model was developed in Chapter 4 using the Euler-Lagrange formulation and described using the physical parameters of the da Vinci robot. Although initial guesses are used for simulation, accurate results are needed in order to use the model for force estimation. Therefore, all parameters such as mass, inertia products, friction constants or center of mass should be found.

Sometimes, CAD data of robot parts is used to obtain these parameters, e.g. the CAD model of the KUKA KR15 industrial manipulator is used to obtain reference values of the inertial parameters in [Swevers et al., 2007]. However, such data is not available for the da Vinci robot. Also, dismantling the different parts and measuring the parameters is not a feasible option. Moreover, friction constants can not be estimated using such methods. Therefore, experimental identification is a more common approach to obtain accurate parameters, as it has been done in [Khosla, 1987], [Swevers et al., 2007] or [Janot et al., 2014]. Their approach is based on rewriting the dynamics differential equation in terms of the parameters, also called parametrization. Then, it uses motion and actuator torque data, which is measured during experiments, to determine the parameters by fitting it to the differential equation previously found. The robot trajectory should be permanently exciting the dynamical system, which is an important fact in order to assure a proper estimation of all the parameters [Swevers et al., 2007]. Moreover, it is repeated periodically in order to reduce the effect of noise in the measurement as explained in Sec. 6.3.

As explained in Chapter 4, the formulation of the robot dynamics can be expressed in the following form:

$$\tau = \mathbf{M}(q)\ddot{q} + V(\dot{q}, q) + G(q) + F(\dot{q}) + \tau_{ext} \quad [\text{Nm}] \tag{6.1}$$

where

τ	is a vector of generalized torques [Nm]
q	is an $n \times 1$ vector of joint position [rad]
$\mathbf{M}(q)$	is a symmetric mass matrix of the manipulator and motors [kg m^2]
$V(\dot{q}, q)$	is an $n \times 1$ vector of centrifugal and Coriolis terms [Nm]
$G(q)$	is an $n \times 1$ vector of gravity terms [Nm]
$F(\dot{q})$	is an $n \times 1$ vector of friction of the motors and joints [Nm]
τ_{ext}	is an $n \times 1$ vector of external torques seen at the joints [Nm]

For parameter estimation, external torques τ_{ext} in Eq. (6.1) are set to zero. Therefore, throughout this section, $\tau_{ext} = 0$. Now, the system dynamics used for estimation become:

$$\tau = \mathbf{M}(q)\ddot{q} + V(\dot{q}, q) + G(q) + F(\dot{q}) \tag{6.2}$$

Then, parametrization of the dynamics consists of linear factorization of Eq. (6.2) into the form:

$$\tau = \mathbf{\Phi}(q, \dot{q}, \ddot{q})\,\theta \tag{6.3}$$

where θ is a vector of parameters, which can be a combination of different physical parameters of the daVinci. $\mathbf{\Phi}$ is called the observation or identification matrix, which is a matrix that depends only on the motion of the robot. An interesting fact about Eq. (6.3) is that it is now linear with respect to the parameters θ and it simplifies the parameter estimation substantially. Analytical methods such as Least Square Estimation (LSE) or Weighted Least Square Estimation (WLSE) can now be used to estimate θ, as it is done in [Swevers et al., 2007], by using data of torque τ, position q, velocity $\dot{q}$ and acceleration $\ddot{q}$.

Often, parametrization of Eq. (6.1) is not an straightforward task. The more joints a robot has, the higher is the complexity of the dynamics equation and therefore, parameterizing is also more difficult. Alternatively, in [Jahandideh and Namvar, 2012a] and [Jahandideh and Namvar, 2012b] it was shown how to use Particle Swarm Optimization (PSO) in order to estimate the physical parameters of the robot without the need to do any parametrization procedure as in Eq. (6.3). While parametrization is skipped, the minimization problem becomes non-convex. For this reason, the difference between convex and non-convex optimization will be discussed in this chapter.

This chapter is based on the previous referenced work and it will be divided in the following parts:

- Description of the model used for system identification.
- Short discussion about convex and non-convex optimization problems and how they can be solved.

- The experiment design part, where an optimal trajectory, which is permanently exciting the dynamical system, is found. Moreover, measured data is processed before the estimation.

- Parameter estimation part, where two methods are compared. First, the methodology in [Swevers et al., 2007] is followed, where the dynamics equation is parametrized and then, the LSE and WLSE is used. The second method avoids the parametrization and uses Particle Swarm Optimization as explained in [Jahandideh and Namvar, 2012a]. Finally, both methods are tested to estimate the daVinci robot parameters.

6.1 Reduced Model

Due to the complexity of the daVinci model found in Chapter 4, it is very difficult to use it in the estimation. Therefore, all the joint are assumed to be independent even though in practice that is not true and there may be coupling between different joints. First, lets consider the system equation:

$$\tau = \mathbf{M}(q)\ddot{q} + V(\dot{q}, q) + G(q) + F(\dot{q}) \tag{6.4}$$

In order to find a reduced model independent for each joint, it is assumed that only one joint is moving while the others are fixed. Then, the model for the i^{th} joint is found by setting $q_j = 0$, $\dot{q}_j = 0$ and $\ddot{q}_j = 0$ in Eq. (6.4), where $j \neq i$. Then, Eq. (6.4) becomes:

$$\tau_i = \mathbf{M}(q_i)\ddot{q}_i + V(\dot{q}_i, q_i) + G(q_i) + F(\dot{q}_i) \tag{6.5}$$

From now on, the system dynamics in (6.4) will be used to refer the dynamics for one joint in order to avoid using the subscript i as in Eq. (6.5). After the simplification, rewriting the dynamics for each joint into the parametrized form of Eq. (6.2) becomes easier.

$$\tau = \mathbf{\Phi}(q, \dot{q}, \ddot{q})\,\theta \tag{6.6}$$

Notice that initially, only the three joints shown in Fig 6.1 will be estimated: *Hand Pitch*, *Hand Roll* and *Instrument Slide*. The rest are considered to be fixed. Therefore, only these three joints will be used to prove the estimation of contact force in the end-effector. To estimate the remaining joints is the first improvement to the project that should be done after the initial design.

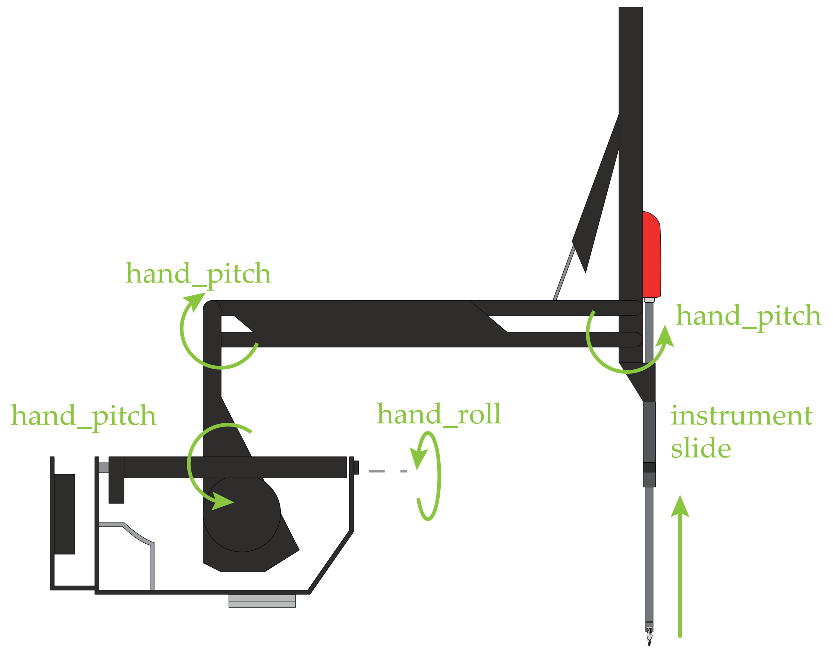


Figure 6.1: Representation of the joints that will be estimated. Notice that the joints *Hand Pitch* move as a parallelogram.

The system equation is developed in Chapter 4. For each of the joints, the independent joint equation is given. Then, it is parametrized. Note that the parameters follows the notation used in Chapter 4. For instance, mass of link 5 is m_5, while v_1 is the viscous friction coefficient for q_1.

Hand Roll (q_1)

$$\begin{aligned}\tau_1 =&(\eta_1^2 J_{m_1} + Iy_2 + Ix_5 + Iy_3 + 0.036\, m_3 + 0.036\, m_4 + 0.036\, m_5 + 0.38\, m_5 lc_5 \\ &- 0.38\, m_3 lc_3 + m_3 lc_3{}^2 + Ix_4 + m_5 lc_5{}^2)\ddot{q}_1 + v_1\, \dot{q}_1 + c_1\, \mathrm{sign}(\dot{q}_1) \\ &+ (-m_5 lc_5 - 0.19\, m_5 - 0.19\, m_4 - 0.19\, m_3 + m_3 lc_3)\; g\; \sin(q_1) \quad [\mathrm{Nm}] \end{aligned} \tag{6.7}$$

which can be parametrized as:

$$\tau_1 = \theta_1\, \ddot{q}_1 + \theta_2 g\, \sin(q_1) + \theta_3\, \dot{q}_1 + \theta_4\, \mathrm{sign}(\dot{q}_1) \quad [\mathrm{Nm}] \tag{6.8}$$

Then, $\mathbf{\Phi_1}$ becomes:

$$\mathbf{\Phi_1}(q_1, \dot{q}_1, \ddot{q}_1) = \begin{pmatrix} \ddot{q}_1 & g\sin(q_1) & \dot{q}_1 & \mathrm{sign}(\dot{q}_1) \end{pmatrix} \tag{6.9}$$

Hand Pitch (q_2)

$$\begin{aligned}\tau_2 =&(\eta_2^2 J_{m_2} + Iz_3 + Iy_5 + Iy_6 + Iy_8 + Iy_9 + 0.04\,(m_3 + m_4 + m_5) - 0.38\, m_3 lc_3 \\ &+ 0.18(m_6 lc_6 + m_7 lc_7 + m_8 lc_8) - 0.22 m_9 lc_9 + 0.38\, m_5 lc_5 + m_3 {lc_3}^2 \\ &+ m_5 {lc_5}^2 + m_6 {lc_6}^2 + m_7 {lc_7}^2 + m_8 {lc_8}^2 + m_9 {lc_9}^2)\, \ddot{q}_2 + v_2\, \dot{q}_2 + c_2\, \text{sign}(\dot{q}_2) \\ &+ (-0.19\,(m_4 - m_3 - m_5) + 0.09\,(m_6 + m_7) + 0.10(m_8 + m_9) + m_3 lc_3 \\ &- m_5 lc_5 - m_6 lc_6 - m_7 lc_7 - m_8 lc_8 - m_9 lc_9) g\, \sin(q_2) + \\ &(-0.01\,(m_4 + m_3) + 0.03\,(m_5 + m_6 + m_7 + m_8 + m_9 + \; m_3 lc_3)) g\, \cos(q_2) \quad [\text{Nm}]\end{aligned} \tag{6.10}$$

which can be parametrized as:

$$\tau_2 = \theta_1\, \ddot{q}_2 + \theta_2 g\, \sin(q_2) + \theta_3 g\, \cos(q_2) + \theta_4\, \dot{q}_2 + \theta_5\, \text{sign}(\dot{q}_2) \quad [\text{Nm}] \tag{6.11}$$

Then, $\mathbf{\Phi_2}$ becomes:

$$\mathbf{\Phi_2}(q_2, \dot{q}_2, \ddot{q}_2) = \begin{pmatrix} \ddot{q}_2 & g \sin(q_2) & g \cos(q_2) & \dot{q}_2 & \text{sign}(\dot{q}_2) \end{pmatrix} \tag{6.12}$$

Instrument Slide (q_3)

$$\tau_3 = \left(\eta_3^2 J_{m_3} + m_6 + m_7 + m_8 + m_9\right) \ddot{q}_3 - (m_6 + m_7 + m_8 + m_9) g + v_3 \dot{q}_3 + c_3 \text{sign}(\dot{q}_3) \quad [\text{N}]$$

which can be parametrized as:

$$\tau_3 = \theta_1\, (\ddot{q}_3 - g) + \theta_2\, \ddot{q}_3 + \theta_3\, \dot{q}_3 + \theta_4\, \text{sign}(\dot{q}_3) \quad [\text{N}] \tag{6.13}$$

Then, $\mathbf{\Phi_3}$ becomes:

$$\mathbf{\Phi_3}(q_3, \dot{q}_3, \ddot{q}_3) = \begin{pmatrix} \ddot{q}_3 - g & \ddot{q}_3 & \dot{q}_3 & \text{sign}(\dot{q}_3) \end{pmatrix} \tag{6.14}$$

6.2 Convex and non-Convex Optimization

Throughout this chapter, optimization problems are formulated for both trajectory optimization and parameter estimation. For this reason, a short introduction to convex and non-convex optimization is given. First, a minimization problem is generally written in the form of:

$$\begin{aligned} \text{Minimize :} \quad & f(x) \\ \text{Subject to :} \quad & g(x) \leq 0 \end{aligned}$$

where $f(x)$ is the function to minimize and $g(x)$ are constraints to the minimization problem, which can also be expressed as $x \in \mathcal{X}$, where $\mathcal{X}$ is the constraint set.

Then, the problem is to find an optimal x^*, where $x^* \in \mathcal{X}$, that gives the minimum value in $f(x)$ among all the possible x that satisfy the constraints, that is:

$$f(x^*) \leq f(x) \quad \text{for all} \quad x \in \mathcal{X}$$

Once the minimization problem is formulated, it is important to distinguish if the problem is convex or not, as some algorithm are more efficient than others when the problem is convex.

Convex Optimization

A minimization problem is convex if both the objective function $f(x)$ and the set $\mathcal{X}$ are convex. Once the minimization problem is convex, it can be solved efficiently. That is because in a convex optimization problem, the optimal solution x^* is the global optimal solution. Therefore, it is important to define both what a convex function and a convex set are:

- A set $\mathcal{X}$ is convex if the line segment between any two points in $\mathcal{X}$ lies in $\mathcal{X}$, i.e., if for any $x_1, x_2 \in \mathcal{X}$ and any θ with $0 \leq \theta \leq 1$, there is [Boyd and Vandenberghe, 2004]:
$$\theta x_1 + (1-\theta)x_2 \in \mathcal{X} \tag{6.15}$$
An example of convex and non-convex sets are shown in Fig. 6.2.

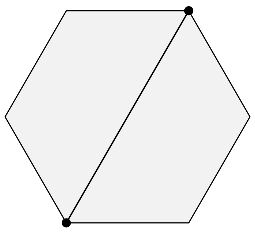
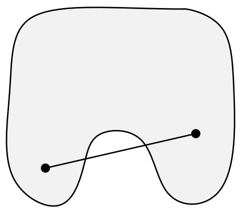



Figure 6.2: Example of simple convex and non-convex sets in $\mathbb{R}^2$. The hexagon set in the left is convex. Instead, the set in the right is not convex, since part of the line between the two points does not belong to the set.

- A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if the domain of f (Dom(f)) is a convex set, and if the line segment between any two points on the graph lies above the graph [Boyd and Vandenberghe, 2004], that is:
$$\forall x_1, x_2 \in \text{Dom}(f), \forall\theta \in [0,1] \qquad f(\theta x_1 + (1-\theta)x_2) \leq \theta f(x_1) + (1-\theta)f(x_2)$$
It can be graphically represented in Fig. 6.3:

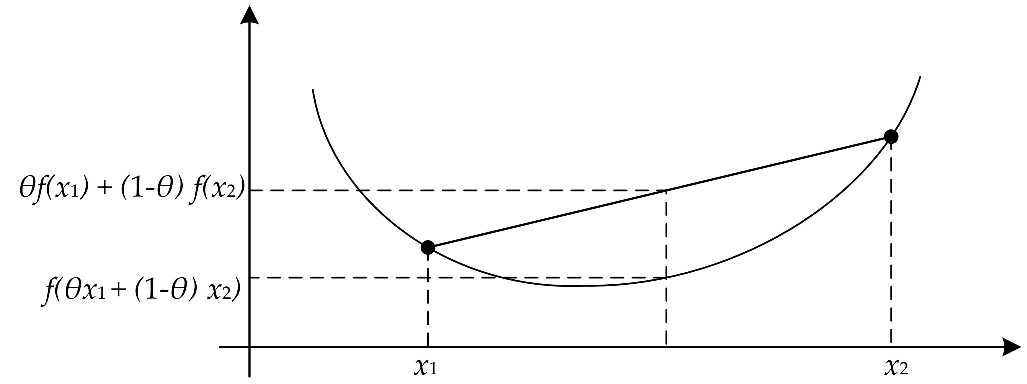


Figure 6.3: Convex function representation, where the line segment between two point x_1 and x_2 of the function f always lies above f.

A Linear Least Square Problem is a special case of convex optimization with no constraints which is used in Sec. 6.4 to estimate robot parameters.

Non-Convex Optimization

If the minimization problem is not known to be convex (i.e. non-convex), there may be many local optimal solutions and to find a global minimum is a more challenging task. An example of local and global minimums is shown in Fig. 6.4.

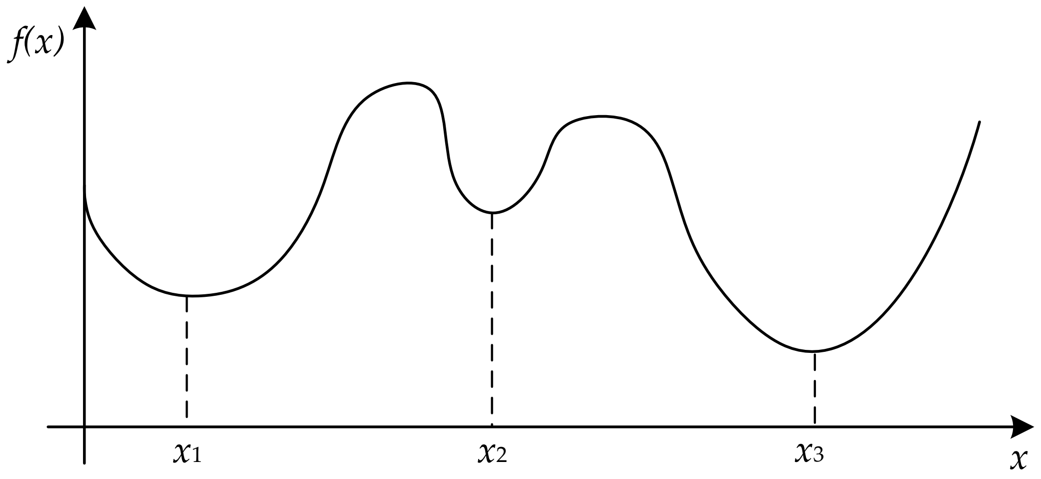


Figure 6.4: Representation of local and global minimums in a non-convex function. While x_1 and x_2 are local minimums, x_3 is the global minimum of $f(x)$ that it is desired to find.

Then, heuristic optimization algorithms such as Particle Swarm Optimization

(PSO) [Kennedy and Eberhart, 1995] can be used to find the global optimal solution. Nonetheless, a global minimum can not be guaranteed in a non-convex problem, as the same algorithm, with different initial conditions, may yield different results.

6.3 Experiment Design

In order to identify the parameters efficiently, it is important to have a proper experiment design. First of all, it is necessary that the system is sufficiently excited in order to have an accurate estimation of all the parameters. Therefore, this exciting trajectory is found by solving an optimization problem.

6.3.1 Trajectory Generation

An optimal trajectory for the system can be any waveform that minimize an optimization problem. But if this trajectory is periodic and bandlimited, the data can be processed and the noise effect reduced. For this reason, it is very common to parametrize the trajectory of the joints as a finite Fourier series, which was introduced by [Swevers et al., 2007] and widely used afterwards. It has the following form:

$$q(t) = a_0 + \sum_{k=1}^{N_f} \left(\frac{a_k}{k\omega_f} \sin(k\omega_f t) - \frac{b_k}{k\omega_f} \cos(k\omega_f t) \right) \tag{6.16}$$

$$\dot{q}(t) = \sum_{k=1}^{N_f} \left(a_k \cos(k\omega_f t) + b_k \sin(k\omega_f t) \right) \tag{6.17}$$

$$\ddot{q}(t) = \sum_{k=1}^{N_f} \left(-a_k k\omega_f \sin(k\omega_f t) + b_k k\omega_f \cos(k\omega_f t) \right) \tag{6.18}$$

where ω_f is the fundamental frequency of the trajectory, a and b are parameters to be found that optimize the trajectory, while N_f will determine the number of Fourier elements. The Fourier series is periodic with period $T_f = 2\pi/\omega_f$ and is chosen to be a multiple of the sampling period T_s. Then, the range of frequencies of each trajectory will go from ω_f to $N_f\omega_f$.

It is important to note that both the number of Fourier elements N_f and the fundamental frequency ω_f are set beforehand as it is done in [Swevers et al., 2007], and not included in the optimization problem. This implies that if a low fundamental frequency ω_f is selected, then the robot can cover all the amplitude of the trajectory easily. However, high frequencies mean high acceleration, which is also important to properly estimate moments and products of inertia of each link.

Therefore, there is a trade-off between choosing high and low fundamental frequencies. Furthermore, constraints in position, velocity and acceleration of each link are considered for trajectory optimization.

6.3.2 Trajectory Optimization

Now that a periodic trajectory is defined in Eq. (6.16), convenient values for the parameters a_k and b_k are selected. Usually, they are found by solving non-linear optimization problems with physical constraints on the robot motion. The most common method is the d-optimality criterion, which minimizes the minus logarithm of the determinant of the covariance matrix of the model parameter estimates $\mathbb{F}$ defined in [Swevers et al., 1997] as:

$$\mathbb{F} = \begin{bmatrix} \mathbf{\Phi}(q(t_1), \dot{q}(t_1), \ddot{q}(t_1)) \\ \vdots \\ \mathbf{\Phi}(q(t_K), \dot{q}(t_K), \ddot{q}(t_K)) \end{bmatrix} \tag{6.19}$$

where $\mathbf{\Phi}$ is the observation matrix defined in Eq. (6.3), and $\mathbb{F}$ is built by evaluating $\mathbf{\Phi}(q(t), \dot{q}(t), \ddot{q}(t))$ along K samples of position, velocity and acceleration of the Fourier series trajectory defined in Eq. (6.16). Therefore, the optimization criteria for a properly excited trajectory is given by:

$$\begin{aligned} \text{Minimize :} \quad & -\log\left(|\mathbb{F}^T\mathbb{F}|\right) \\ \text{Subject to :} \quad & g(q, \dot{q}, \ddot{q}) \leq 0 \end{aligned} \tag{6.20}$$

where $g(q, \dot{q}, \ddot{q}) \leq 0$ are constraints on the motion of the robot, which are defined as:

$$q \in [\underline{q} \quad \overline{q}] \qquad \dot{q} \in [\underline{\dot{q}} \quad \overline{\dot{q}}] \qquad \ddot{q} \in [\underline{\ddot{q}} \quad \overline{\ddot{q}}]$$

where the lower bar express a minimum value and the upper bar express a maximum value. The logarithm of the determinant is known to be a convex function [Boyd and Vandenberghe, 2004]. However, $\mathbb{F}$ is not necessary convex and therefore, convexity is not assured in Eq. (6.20). For instance, as the Coulomb friction is defined as sign($\dot{q}$) in matrix $\mathbf{\Phi}$, then $\mathbf{\Phi}$ can not be considered convex. Nonetheless, even though a global minimum can not be guaranteed, the trajectory can be optimized.

This optimization problem can be solved with respect to the trajectory parameters using any gradient descent algorithm, and the function `fmincon` is used in MATLAB to find the optimal trajectory for each joint. The steps followed are now summarized:

- Choose a trajectory period T_f multiple of the sample period T_s. It is important that it is fast enough to excite the system, but also that the robot is able to follow the given trajectory. Notice that ω_f may differ from one joint to another if its dynamics are considered faster.

- The Fourier series is defined with $N_f = 2$ as:

$$q(t) = a_0 + \sum_{k=1}^{2} \left(\frac{a_k}{k\omega_f} \sin(k\omega_f t) - \frac{b_k}{k\omega_f} \cos(k\omega_f t) \right) \tag{6.21}$$

$$\dot{q}(t) = \sum_{k=1}^{2} \left(a_k \cos(k\omega_f t) + b_k \sin(k\omega_f t) \right) \tag{6.22}$$

$$\ddot{q}(t) = \sum_{k=1}^{2} \left(-a_k k\omega_f \sin(k\omega_f t) + b_k k\omega_f \cos(k\omega_f t) \right) \tag{6.23}$$

which gives a total of five parameters $(a_0, a_1, a_2, b_1, b_2)$ to find in the minimization problem defined in Eq. (6.20).

- Constraints for q, $\dot{q}$ and $\ddot{q}$ are defined for one period T_f of the trajectory in the form:

$$Ax \leq b \tag{6.24}$$

where $x = a_0, a_1, a_2, b_1, b_2$ and q, $\dot{q}$ and $\ddot{q}$ are written as Ax. For instance, the constraint $q(t) \leq \bar{q}$ is defined for one period of the trajectory (t is sampled from 0 to T_f) as:

$$\begin{pmatrix} 1 & \frac{1}{\omega_f}\sin(\omega_f 0) & -\frac{1}{\omega_f}\cos(\omega_f 0) & \frac{1}{2\omega_f}\sin(2\omega_f 0) & -\frac{1}{2\omega_f}\cos(2\omega_f 0) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \frac{1}{\omega_f}\sin(\omega_f T_f) & -\frac{1}{\omega_f}\cos(\omega_f T_f) & \frac{1}{2\omega_f}\sin(2\omega_f T_f) & -\frac{1}{2\omega_f}\cos(2\omega_f T_f) \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ b_1 \\ a_2 \\ b_2 \end{pmatrix} \leq \begin{pmatrix} \bar{q} \\ \vdots \\ \bar{q} \end{pmatrix} \tag{6.25}$$

- $\mathbb{F}$ is built with $\mathbf{\Phi}(q(t), \dot{q}(t), \ddot{q}(t))$ as in Eq. (6.19), where $q(t)$, $\dot{q}(t)$ and $\ddot{q}(t)$ are samples for one period of the trajectory (from 0 to T_f). See Sec. 6.1 for the definition of $\mathbf{\Phi}(q(t), \dot{q}(t), \ddot{q}(t))$ of each joint. For instance, for joint *Hand Roll*, $\mathbb{F}$ becomes:

$$\mathbb{F} = \begin{pmatrix} \mathbf{\Phi}(q(0), \dot{q}(0), \ddot{q}(0)) \\ \vdots \\ \mathbf{\Phi}(q(T_f), \dot{q}(T_f), \ddot{q}(T_f)) \end{pmatrix} = \begin{pmatrix} \ddot{q}_1(0) & g\sin(q_1(0)) & \dot{q}_1(0) & \text{sign}(\dot{q}_1(0)) \\ \vdots & \vdots & \vdots & \vdots \\ \ddot{q}_1(T_f) & g\sin(q_1(T_f)) & \dot{q}_1(T_f) & \text{sign}(\dot{q}_1(T_f)) \end{pmatrix} \tag{6.26}$$

Notice that $q(t)$, $\dot{q}(t)$ and $\ddot{q}(t)$ depend on the parameters $(a_0, a_1, a_2, b_1, b_2)$ that should be found.

- Then, position, velocity and acceleration constraints are included in the function to minimize:

$$\begin{aligned} \text{Minimize :} \quad \text{Minimize :} \quad & -\log\left(|\mathbb{F}^T\mathbb{F}|\right) \\ \text{Subject to :} \quad & Ax \leq b \end{aligned} \tag{6.27}$$

- `fmincon` is now used to minimize Eq. (6.27). In Fig. 6.5, trajectories for the three joints are shown:

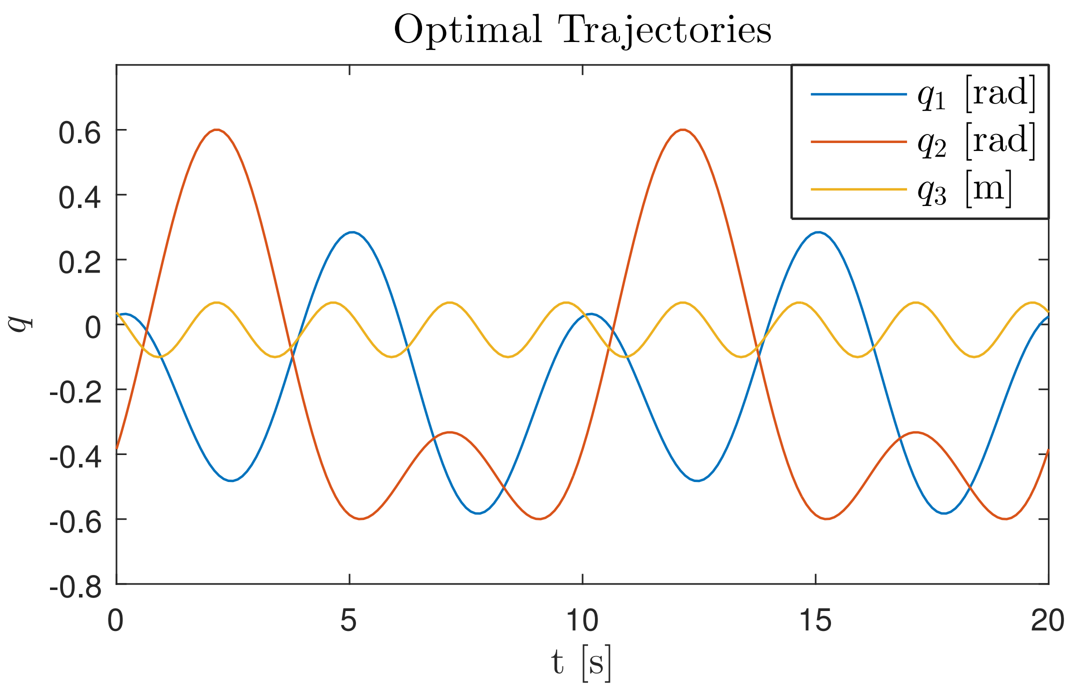


Figure 6.5: 20 seconds of permanent exciting trajectories for the three joints.

The trajectory parameters for the three joints in Fig. 6.5 are shown in Table 6.1:

Joint Name	a_0	a_1	b_1	a_2	b_2	T_f [s]	$f(x^*)$
Hand Roll	-0.1848	0.0260	0.0813	0.0609	-0.4261	10	-8.5972
Hand Pitch	-0.1912	0.2861	-0.0639	0.1741	0.3701	10	-12.7527
Instrument Slide	-0.0164	0.0001	0.0002	-0.1632	-0.1350	5	-8.6718

Table 6.1: Table of the trajectory parameters for each joint found after solving the minimization problem.

The constraints of each joint are shown in Table 6.2 and are chosen after testing the limitations in position, velocity and acceleration in the daVinci robot.

Joint Name	$\underline{q}$	$\overline{q}$	$\underline{\dot{q}}$	$\overline{\dot{q}}$	$\underline{\ddot{q}}$	$\overline{\ddot{q}}$
Hand Roll	-0.6	0.6	-1.2	1.2	-1.2	1.2
Hand Pitch	-0.6	0.6	-1.2	1.2	-1.2	1.2
Instrument Slide	-0.0672	0.1097	-2	2	-3	3

Table 6.2: Table of constraints in the trajectory for each joint. Units for the position, velocity and acceleration of *Hand Roll* and *Hand Pitch* are [rad], [rad/s] and [rad/s^2] respectively, while units for *Instrument Slide* are [m], [m/s] and [m/s^2].

6.3.3 Signal Processing

The robot trajectory is now periodically repeated. It is important to have several periods of the signal as it will be useful to reduce measurement noise. This section is inspired by [Swevers et al., 2007].

Motor position θ_m is measured with encoders in the motor. Then, using the gear ratio η, the joint position q can be found as:

$$q(t) = \frac{1}{\eta}\theta_m(t) \tag{6.28}$$

On the other hand, the torque data is obtained using current measurements in the motor, as it is known that they are proportional and related with the motor torque constant K_m. Also, the motor torque is higher at the joint due to the gear:

$$\tau(t) = \eta K_m \, i(t) \tag{6.29}$$

The motor torque constant of all the motors is known and shown in Table 6.3. Likewise, the gear ratio of the motors is also known in advance from the motor specifications.

	Motor 1	Motor 2	Motor 3	Motor 4	Motor 5	Motor 6	Motor 7
η [-]	200	200	1340	7.5	12.4	12	12
K_m [Nm/A]	0.0438	0.0438	0.0438	0.0438	0.0438	0.0438	0.0438

Table 6.3: Motor parameters.

Notice that Motor 3 corresponds to the prismatic joint, therefore, the gear ratio relates motor angular position to linear position of the joint.

Data Averaging and Noise Estimation

Measurement noise is present in all sensors and actually, current measurements are always very noisy. This noise may cause error in the estimation, however it can be reduced. First of all, since the experiment data is periodic, data averaging can improve the signal-to-noise as shown in Fig. 6.6. Therefore, the mean is computed from the repeated trajectory.

$$\bar{x}(k) = \frac{1}{M} \sum_{m=1}^{M} x_m(k) \tag{6.30}$$

where

x	corresponds to either measured torque τ or joint position q
$\bar{x}(k)$	is the average of measured data at sample k
$x_m(k)$	is the k^{th} sample within the m^{th} period
M	denotes the number of periods of the repeated trajectory
K	is the number of samples of one trajectory

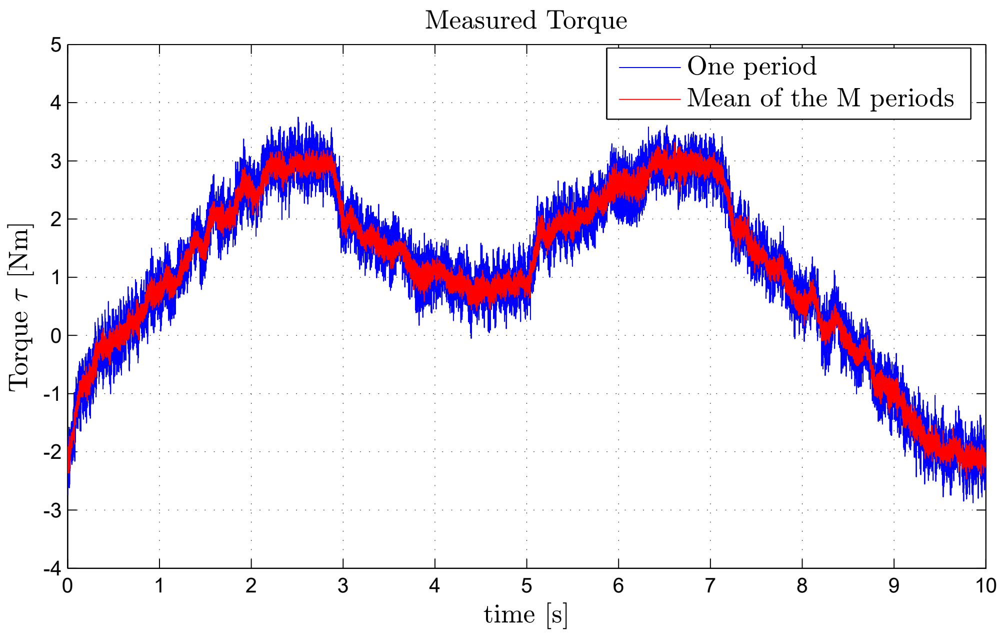


Figure 6.6: The average of the M periods of the trajectory (red) is compared with one period (blue).

Also, the variance of the mean measured data can be computed, as it can give information to whether a mean sample is better than another or not. For instance, a sample with low variance is considered more relevant in the estimation than a sample with high variance. By using the the M periods with K samples, it can be computed as:

$$\sigma_{\bar{x}}^2(k) = \frac{1}{(M-1)} \sum_{m=1}^{M} (x_m(k) - \bar{x}(k))^2 \tag{6.31}$$

Notice that Eq. (6.31) is not the variance of the noise but the variance of the mean $\bar{x}$ at every sample k, which is computed using the M periods of the signal. It is used in Sec. 6.4 to weight the mean samples in the estimation. The variance of the noise can be computed as:

$$\sigma_x^2 = \frac{1}{(MK-1)} \sum_{k=1}^{K} \sum_{m=1}^{M} \left(x_m(k) - \bar{x}(k)\right)^2 \tag{6.32}$$

It is expected that after data averaging, the measurement noise of the torque is reduced. This noise is given by:

$$e_\tau(k) = \bar{\tau}(k) - \tau(k) \tag{6.33}$$

In order to see the characteristics of this noise, the data histogram is checked in Fig. 6.7. It appears to be a normal distribution with zero mean.

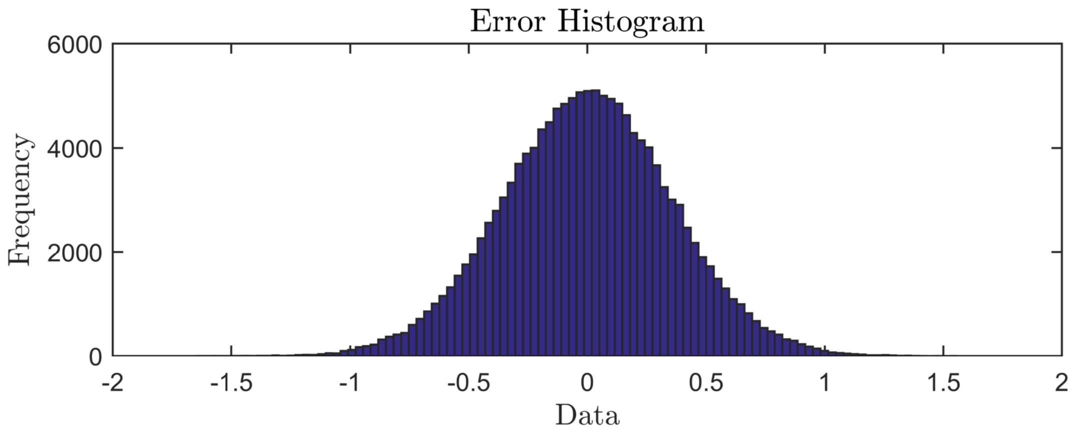


Figure 6.7: The histogram shows the distribution of the torque noise removed after data averaging.

Joint Velocity and Acceleration

Velocity and acceleration are not directly measured, but they can always be computed by numerical differentiation of position q. However, if noise is present in the position signal, it can affect the accuracy of this calculation as velocity and acceleration data becomes very noisy. Fortunately, as the designed trajectory is a periodic signal with known frequency spectrum, it can be differentiated precisely using the frequency domain as explained in [Swevers et al., 2007]. Once the robot follows the trajectory given, it will be very easy to distinguish between signal and noise in the frequency spectrum of the measured data and filter it. The analytical differentiation of position is explained as follows:

- The sampling frequency F_s of the joint position measurement should be at least twice the bandwidth of the periodic signal ($2F_f$) in order to avoid aliasing.

- Once the trajectory is measured, the averaged joint position $\bar{q}(t)$ computed using Eq. (6.30) is converted to the frequency domain using the Fast Fourier Transform (`fft` in MATLAB). The spectrum $Q(f)$ of one trajectory is shown in Fig. 6.8.
- Signal is filtered in the frequency domain, a rectangular window $H(f)$ is multiplied per $Q(f)$ in the frequency domain, such that frequencies that does not belong to the periodic signal are removed. From the trajectory defined in Sec. 6.3.1, it is known that each trajectory consist of frequencies at zero, ω_f and $2\omega_f$.

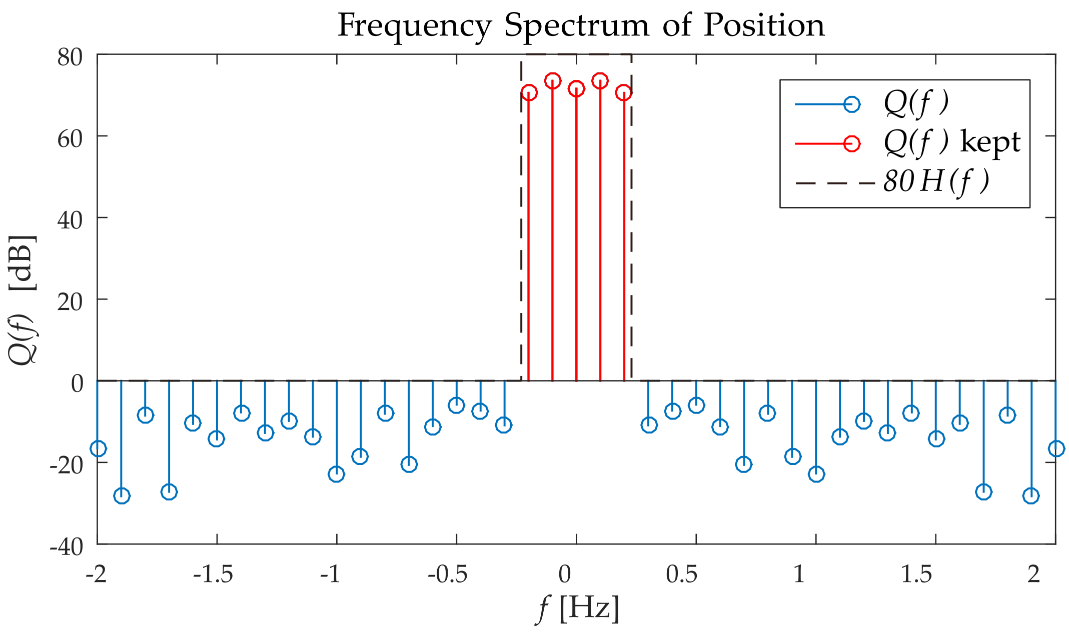


Figure 6.8: Frequency spectrum of $Q(f)$, which is a Fourier series of frequencies 0.1 and 0.2 Hz and a constant value (frequency zero). A rectangular windows is used to remove the rest of the spectrum considered noise.

- Now the signal can be differentiated by multiplying $Q(f)$ per the frequency response of a single and double differentiator:

$$\mathcal{F}\left[\frac{d^n\bar{q}(t)}{dt^n}\right] = (j\omega)^n Q(f) \tag{6.34}$$

where $\omega = 2\pi f$ and n determines the degree of the derivative ($n = 1$ determines the first derivative to obtain velocity and $n = 2$ determines the second derivative to obtain acceleration).

- Finally, the spectrum is transformed back to the time domain using the Inverse Fast Fourier Transform (`ifft` in MATLAB). The resulted free of noise signals are the first and second derivative of the position, which are velocity and acceleration respectively. Compared to numerical differentiation (e.g. $\dot{q}(k) = (q(k) - q(k-1))/T_s$), the signal is smoother as the noise has been removed as seen in Fig. 6.9. The effect of the noise is more appreciated in the acceleration.

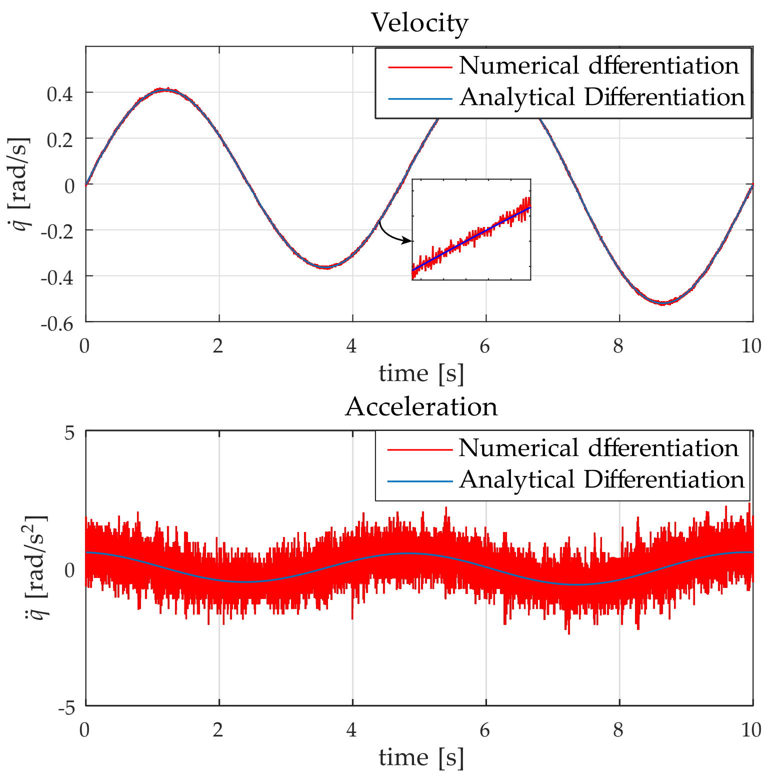


Figure 6.9: First and second derivative of position for *Hand Roll*. Differentiation in the frequency domain (analytical differentiation) is compared with the numerical differentiation given.

Noise in q has been removed after data averaging of the position and filtering the undesired frequencies in the frequency domain. Ideally, the noise should have zero mean with normal distribution. Therefore, first the noise is shown to have zero mean and normal distribution in Fig. 6.10. Moreover, noise should be uncorrelated but it is possible that it is slightly correlated as seen in Fig. 6.11. It means that a small part of the robot trajectory may have been removed during filtering. However, it is considered that the filtering has removed only measurement noise. Therefore, the computed q, $\dot{q}$ and $\ddot{q}$ is free of noise.

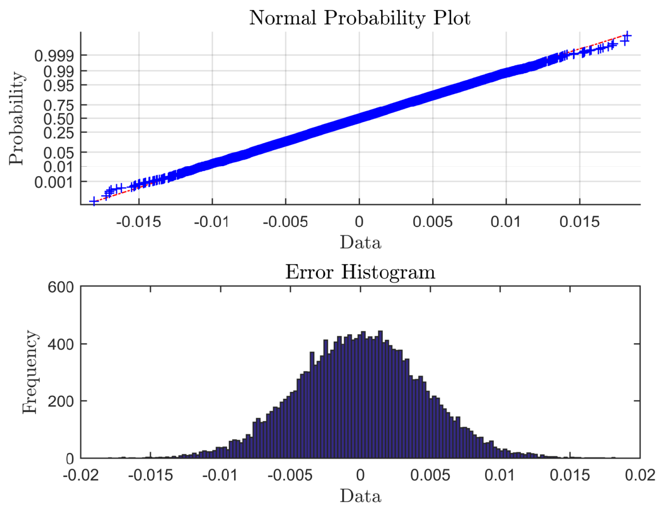


Figure 6.10: The normal probability plot of the noise (upper graph) identifies departures from normality, which is represented as a deviation from the red straight line. The histogram (lower graph) also represents how the distribution is normal.

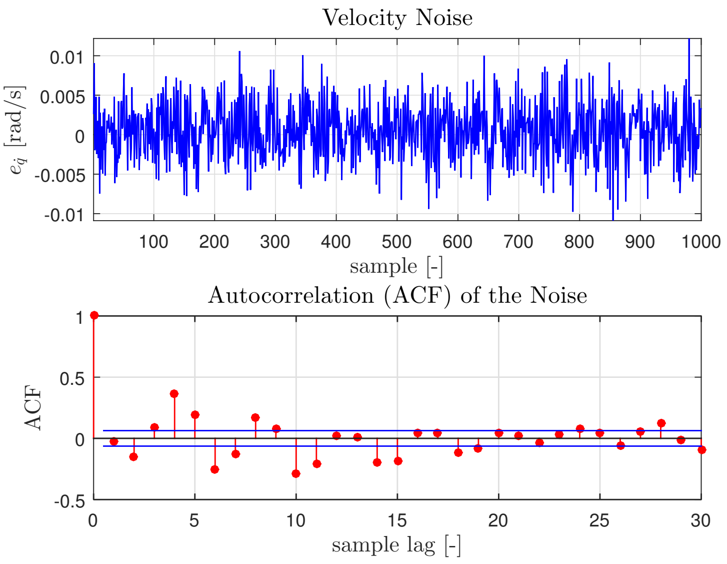


Figure 6.11: The difference between the analytical and the numerical differentiation is plotted for 1000 samples (upper graph) and its correlation is shown (lower graph).

6.4 Linear Least Squares Estimation

At the beginning of this chapter, it was assumed that all the joints are independent. Therefore, each joint is excited and estimated separately. For now, only the *Hand Pitch*, *Hand Roll* and *Instrument Slide* are used. Then, consider the robot dynamics in Eq.(6.35), where external forces are considered zero:

$$\tau = \mathbf{M}(q)\ddot{q} + V(\dot{q}, q) + G(q) + F(\dot{q}) \tag{6.35}$$

As mentioned before, Eq. (6.35) can be written linearly in terms of the parameters θ to be estimated [Swevers et al., 2007]:

$$\tau = \mathbf{\Phi}(q, \dot{q}, \ddot{q})\,\theta \tag{6.36}$$

See Sec. 6.1 for the parametrized equations. Once the system dynamics can be expressed as a linear equation with respect to the parameters as in Eq. (6.36), the mean measurements τ, q, $\dot{q}$ and $\ddot{q}$ computed in Sec. 6.3.3 can be used to write the following system of linear equations:

$$\tau = \mathbb{F}\,\theta \tag{6.37}$$

where

$$\mathbb{F} = \begin{bmatrix} \mathbf{\Phi}(q(t_1), \dot{q}(t_1), \ddot{q}(t_1)) \\ \vdots \\ \mathbf{\Phi}(q(t_K), \dot{q}(t_K), \ddot{q}(t_K)) \end{bmatrix} \tag{6.38}$$

is built using the identification matrix $\mathbf{\Phi}$ evaluated at every time sample,

$$\tau = \begin{bmatrix} \tau(t_1) \\ \vdots \\ \tau(t_K) \end{bmatrix} \tag{6.39}$$

is built using the torque data at every time sample, and

$$\theta = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_p \end{bmatrix} \tag{6.40}$$

is the vector of independent parameters or combination of parameters. Considering that the number of samples K (i.e. the number of equations) is higher than the number of parameters p to estimate, then the system in Eq. (6.37) is overdetermined (i.e. more number of equations than unknown parameters to find). A common method to get unbiased estimates of the parameters is the Linear Least Square Estimation (LLSE), which estimate the vector θ that minimizes the square of the estimated torque error with respect to the estimated parameters θ, given data of τ, q, $\dot{q}$ and $\ddot{q}$. LLSE is a widely known optimization problem, which is convex and unconstrained [Boyd and Vandenberghe, 2004] and can be easily solved. First, lets formulate the optimization problem as:

$$\text{Minimize:} \quad f(\theta) = \|\tau - \mathbb{F}\theta\|^2 \tag{6.41}$$

One way to take into account the noise present in the torque τ measurement is to weight the corresponding error using a diagonal weighting matrix **W**, which is used to discriminate between accurate and inaccurate data as it is explained in [Swevers et al., 2007]. Then, the problem to solve can be expressed as Weighted Least Square Estimation (WLSE), where the function to be minimized becomes:

$$f(\theta) = ||\mathbf{W}^{1/2}\,(\tau - \mathbb{F}\theta)\,||^2 \tag{6.42}$$

where

$\mathbf{W}$	is an $K \times K$ diagonal matrix of weights of the measured torque
τ	is an $K \times 1$ vector of the mean measured torque [Nm]
$\mathbb{F}$	is an $p \times K$ system matrix
θ	is an $p \times 1$ vector of unknown parameters

Note that the above dimensions are for the case of one joint. The weighted square error should now be minimized, which can be expressed as:

$$\begin{aligned} f(\theta) &= ||\mathbf{W}^{1/2}\,(\tau - \mathbb{F}\theta)||^2 \\ &= (\tau - \mathbb{F}\theta)^T\mathbf{W}(\tau - F\theta) \\ &= \tau^T\mathbf{W}\tau - \theta^T\mathbb{F}^T\mathbf{W}\tau - \tau^T\mathbf{W}\mathbb{F}\theta + \theta^T\mathbb{F}^T\mathbf{W}\mathbb{F}\theta \end{aligned} \tag{6.43}$$

where $\theta^T\mathbb{F}^T\mathbf{W}\tau$ and $\tau^T\mathbf{W}\mathbb{F}\theta$ have dimensions 1×1, so they are scalars, hence $\theta^T\mathbb{F}^T\mathbf{W}\tau = \tau^T\mathbf{W}\mathbb{F}\theta$. Then, the weighted square error to minimize becomes:

$$f(\theta) = \tau^T\mathbf{W}\tau - 2\theta^T\mathbb{F}^T\mathbf{W}\tau + \theta^T\mathbb{F}^T\mathbf{W}\mathbb{F}\theta \tag{6.44}$$

By differentiating (6.44) with respect to θ and equating to zero, the minimum of the convex function is found:

$$-2\mathbb{F}^T\mathbf{W}\tau + 2(\mathbb{F}^T\mathbf{W}\mathbb{F}\theta) = 0 \tag{6.45}$$

Once Eq. (6.45) is rearranged, the estimated $\hat{\theta}$ that minimizes the weighted square error is expressed as:

$$\hat{\theta}_{WLSE} = (\mathbb{F}^T\mathbf{W}\mathbb{F})^{-1}\mathbb{F}^T\mathbf{W}\tau \tag{6.46}$$

Note that if $\mathbf{W} = \mathbb{I}$, where $\mathbb{I}$ is the identity matrix, then the WLSE problem becomes the LSE. In order to find the weighting matrix **W**, the variance of the mean torque, defined in Eq. (6.31), is used:

$$\sigma^2_{\bar{\tau}}(k) = \frac{1}{(M-1)}\sum_{m=1}^{M}\left(\tau_m(k) - \bar{\tau}(k)\right)^2 \tag{6.47}$$

Notice that the data used in the WLSE is the mean of a repeated periodic trajectory, and the different periods are used to find the above variance in the torque. Then, a covariance diagonal matrix $\mathbf{\Sigma}$ is built using the variance found in Eq. (6.47). The higher is the variance, the worse is the sample. Therefore, its inverse $\mathbf{\Sigma}^{-1}$ is used as a weight for the WLSE.

$$\mathbf{W} = \mathbf{\Sigma}^{-1} = \begin{bmatrix} \frac{1}{\sigma_\tau^2(1)} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{1}{\sigma_\tau^2(K)} \end{bmatrix} \tag{6.48}$$

Finally, Eq. (6.46) becomes:

$$\hat{\theta}_{WLSE} = (\mathbb{F}^T \mathbf{\Sigma}^{-1} \mathbb{F})^{-1} \mathbb{F}^T \mathbf{\Sigma}^{-1} \tau \tag{6.49}$$

If data is not weighted, then the WLSE becomes the LSE:

$$\hat{\theta}_{LSE} = (\mathbb{F}^T \mathbb{F})^{-1} \mathbb{F}^T \tau \tag{6.50}$$

Both methods are later compared, altogehter with PSO estimation, in Sec. 6.6.

6.5 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is an optimization algorithm which was inspired by natural phenomena, such as the behavior of bird swarm in the nature. It is a technique to solve both convex and non-convex optimization problems which was originally introduced in [Kennedy and Eberhart, 1995]. PSO has been used ever since in different optimization applications until it found way into system identification problems. For instance, in [Xia et al., 2012] it is used to solve a non-convex optimal power flow problem in power systems. Jahandideh and Namvar introduced the use of PSO in parameter estimation of robot dynamics in [Jahandideh and Namvar, 2012a] and [Jahandideh and Namvar, 2012b]. The interesting part of their approach is that the parametrization of the robot dynamics is not needed, as it sometimes can be a difficult task. This is the reason why PSO is chosen as an alternative to compare with WLSE/LSE in this project.

Likewise, it is important to remark that for convex functions, where the minimum is global, other tools can be used to obtain the optimal solution more efficiently. Therefore, notice that PSO is used to solve a non-convex problem in order to find the parameters of the system.

The main goal of PSO is to optimize a function $f : \mathcal{R}^n \rightarrow \mathcal{R}$ given some constraints to the physical parameters (e.g. mass is positive):

$$\begin{aligned} \text{Minimize :} \quad & f(x) \\ \text{Subject to :} \quad & x \in \mathcal{X} \end{aligned}$$

where $\mathcal{X} \subseteq \mathcal{R}^n$. Then, the algorithm of PSO uses a swarm of k particles or individuals that search for an optimal solution for $f(x)$ in an n-dimensional space ($x \in \mathcal{R}^n$). An overview of the algorithm is given in Fig. 6.12.

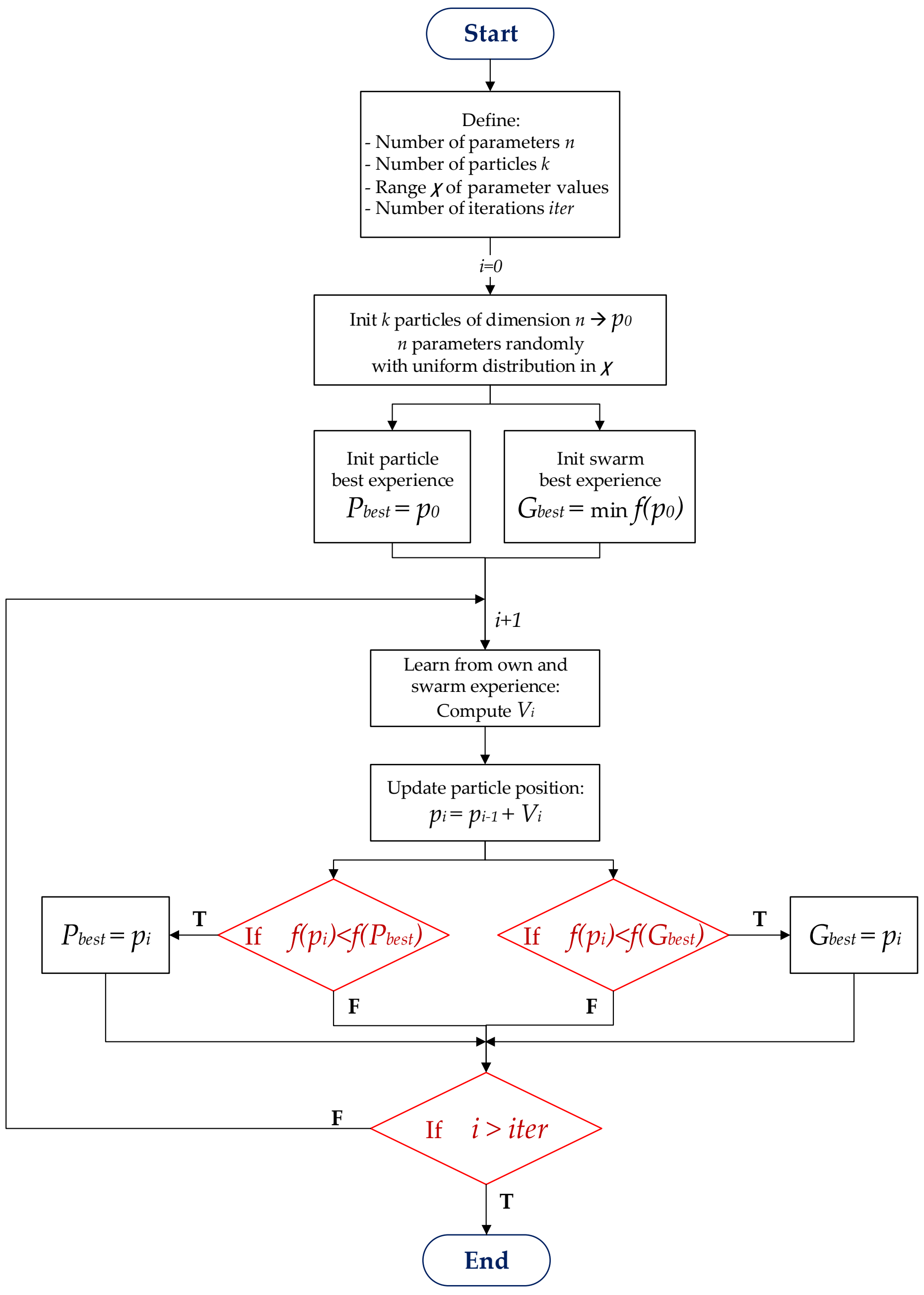

Start
Define:
- Number of parameters n
- Number of particles k
- Range χ of parameter values
- Number of iterations $iter$
i=0
Init k particles of dimension $n \rightarrow p_0$
n parameters randomly
with uniform distribution in χ
Init particle best experience $P_{best} = p_0$
Init swarm best experience $G_{best} = \min f(p_0)$
i+1
Learn from own and swarm experience: Compute V_i
Update particle position: $p_i = p_{i-1} + V_i$
$P_{best} = p_i$
T
If $f(p_i)<f(P_{best})$
If $f(p_i)<f(G_{best})$
T
$G_{best} = p_i$
F
F
F
If $i > iter$
T
End

Figure 6.12: Overview of the PSO algorithm.

The algorithm is described as follows:

- Each particle represents a potential solution of x to the optimization problem. Each particle is written in an $n \times 1$ vector p, where n is the dimension of x.
- Each particle of the swarm is initialized randomly with uniform distribution within $\mathcal{X}$.
- Each particle calculates the function value $f(p_i)$ of its current position p_i, where i stands for the i^{th} iteration.
- Each particle knows its own best experience P_{best} and the global best experience of all the swarm G_{best}. At every iteration i, the best experience P_{best} is compared with the current position p_i and it is updated if necessary. The same comparison is made with G_{best} and p_i as:

$$\begin{aligned} P_{best} &= p_i \quad \text{if} \quad f(p_i) < f(P_{best}) \\ G_{best} &= p_i \quad \text{if} \quad f(p_i) < f(G_{best}) \end{aligned}$$

- Each particle moves according to its experience (P_{best}) and the swarm experience (G_{best}). The movement of one particle at the i^{th} iteration is divided in three parts:

$$V_i = \underbrace{w_i V_{i-1}}_{\text{Previous direction}} + \underbrace{C_1 r_1 \left(P_{best} - p_{i-1}\right)}_{\text{Cognitive part}} + \underbrace{C_2 r_2 \left(G_{best} - p_{i-1}\right)}_{\text{Social part}} \qquad (6.51)$$

where

V_i	is an $n \times 1$ vector of the direction of one particle
p_i	is an $n \times 1$ particle at iteration i
P_{best}	is $n \times 1$ vector which stores the best experience of the particle
G_{best}	is $n \times 1$ vector which stores the best experience of all k particles
r_1 r_2	are uniformly distributed random number in the range $[0, 1]$
C_1	is the cognitive learning rate (learning from own experience)
C_2	is the social learning rate (learning from group experience)
w_i	is the inertia weight which defines how fast the particles move

This direction V_i can be seen as a gradient which leads the particle to the function minimum. It is initialized to $V_0 = 0$. Then, the new position of the particle is defined using the previous position and the gradient found:

$$p_i = p_{i-1} + V_i \qquad (6.52)$$

- This process is iterated until some conditions are fulfilled. Generally, the condition to stop is either a predefined number of iterations, or a given number of iteration without any significant change in G_{best}. Then, the global best experience G_{best} at the last iteration is considered to be the optimal solution.

The PSO algorithm is implemented in MATLAB using the Particle Swarm Optimization Toolbox from [Birge, 2006]. An important advantage about PSO compared to other methods for non-convex problems is that the use of several particles will make it easier to reach the global minimum as shown in Fig. 6.13.

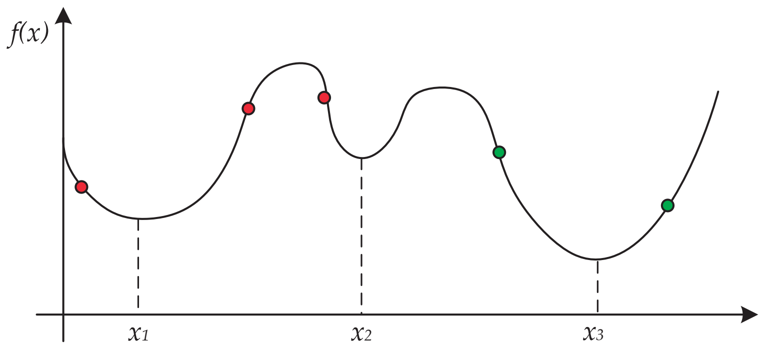


Figure 6.13: Illustration of PSO optimization. The use of multiple particles will help to reach a global minimum as each particle is initialized in a different point. The red particles are closer to local minimums like x_1 or x_2, while the green particles are closer to the global minimum x_3. The swarm learning algorithm will eventually lead all particles to x_3.

6.5.1 PSO in Parameter Estimation

As an alternative to WLSE, in [Jahandideh and Namvar, 2012a] they suggested that PSO could also be used to estimate robot parameters **without parametrization** of the system dynamics. The concept of PSO is that it minimizes or maximizes a function. Therefore, an error function is defined in order to be minimized. First, the robot dynamics equation is used to estimate the torque, which depends on the measured data and the set of potential estimates of the parameters:

$$\hat{\tau} = \mathbf{M}(q)\ddot{q} + V(\dot{q}, q) + G(q) + F(\dot{q}) \tag{6.53}$$

Then, the error can be defined as the difference of the measured and predicted torque:

$$e(t) = \tau(t) - \hat{\tau}(t) \tag{6.54}$$

Now, the error for all K samples is written in a vector E as:

$$E = \begin{bmatrix} e(t_1) \\ \vdots \\ e(t_K) \end{bmatrix} \tag{6.55}$$

Finally, the function to be minimized can be defined as the 2-norm of the error vector:

$$f = \|E\|_2 \tag{6.56}$$

Even though the 2-norm is known to be a convex function [Boyd and Vandenberghe, 2004], $e(t)$ is non-convex. That is because $\hat{\tau}$ is not parametrized and some parameters appear as quadratic terms or multiplying other terms in $\hat{\tau}$ (e.g. $m\,l_c$ and $m\,l_c^2$ in Sec. 6.1). Therefore, the **problem is non-convex**. Nevertheless, notice that if $\hat{\tau} = \mathbf{\Phi}\theta$, the problem would become convex. However, then parametrization will be needed again and here it is desired to avoid it.

Now, PSO can be used to minimize Eq. (6.56). As it is using a swarm of particles, it is more possible to reach the global minimum and not a local minimum. At least, it would perform better compared to other algorithms, such as gradient descent, that uses just one initial point in order to find a local minimum. A total of 24 particles are used to find the physical parameters of each joint. The algorithm is iterated a total of 3000 times unless it stops when the best global experience does not change for 150 iterations. An example of how the global best experience evolves during the algorithm is shown in Fig. 6.14.

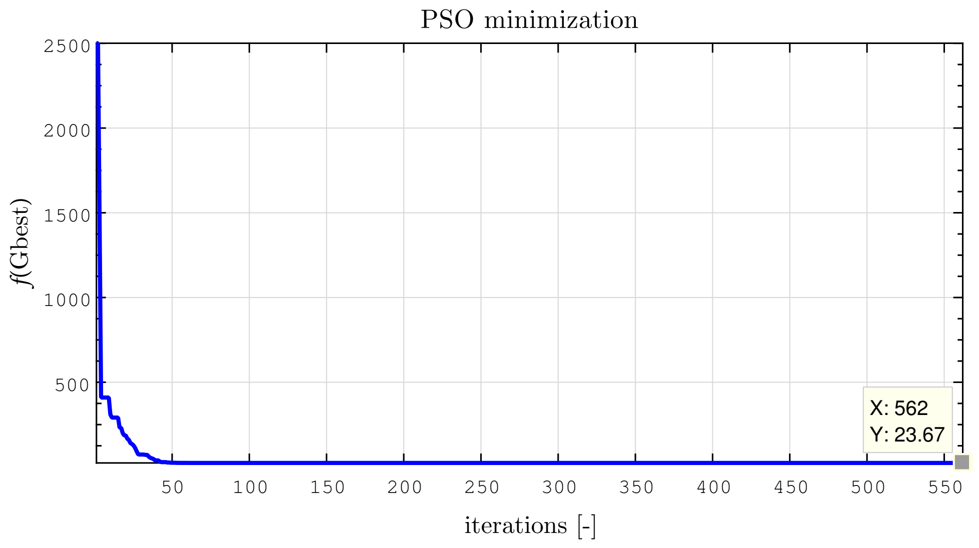


Figure 6.14: Minimization of the function using PSO.

Results of the estimation and are shown and discussed in Sec. 6.6.

6.6 Parameter Estimation Results

In this chapter, estimation of the joints is done one at a time. Both WLSE/LSE and PSO are used to estimate the parameters and they are compared with the initial guesses done in Chapter 4 by computing the Mean Square Error (MSE). The equations used for the estimation were developed in Maple and given in Sec. 6.1.

Hand Roll (q_1)

Its dynamical equation is found as:

$$\begin{aligned}\tau_1 =& (\eta_1^2 J_{m_1} + Iy_2 + Ix_5 + Iy_3 + 0.036\, m_3 + 0.036\, m_4 + 0.036\, m_5 + 0.38\, m_5 lc_5 \\ &- 0.38\, m_3 lc_3 + m_3 {lc_3}^2 + Ix_4 + m_5 {lc_5}^2)\ddot{q}_1 + v_1\, \dot{q}_1 + c_1\, \mathrm{sign}(\dot{q}_1) \\ &+ (-m_5 lc_5 - 0.19\, m_5 - 0.19\, m_4 - 0.19\, m_3 + m_3 lc_3)\; g\, \sin(q_1) \quad [\mathrm{Nm}] \qquad (6.57)\end{aligned}$$

which has a total of 12 parameters for PSO. Eq. (6.57) can be expressed in 4 independent parameters for WLSE/LSE:

$$\tau_1 = \theta_{1,1}\, \ddot{q}_1 + \theta_{2,1} g\, \sin(q_1) + \theta_{3,1}\, \dot{q}_1 + \theta_{4,1}\, \mathrm{sign}(\dot{q}_1) \quad [\mathrm{Nm}] \qquad (6.58)$$

In Table 6.4 the initial guesses are compared with the estimations done using WLSE/LSE and PSO:

Parameters	Initial Guess	WLSE	LSE	PSO
$\theta_{1,1}$	0.3843	0.4521	0.4528	0.4528
$\theta_{2,1}$	0.194	0.0110	0.0103	0.0103
$\theta_{3,1}$	1	0.3707	0.3706	0.3705
$\theta_{4,1}$	1	0.2203	0.21968	0.2197
MSE	1.3899	0.0199	0.0197	0.0197

Table 6.4: Table of the estimated parameters for *Hand Roll*.

There is no difference between weighted and not weighted LSE so they are considered the same. Also, PSO gives practically the same results. Note that WLSE/LSE uses Eq. (6.58) while PSO uses Eq. (6.57). A comparison is now made between the predicted torque with the two methods and the torque prediction error is also computed.

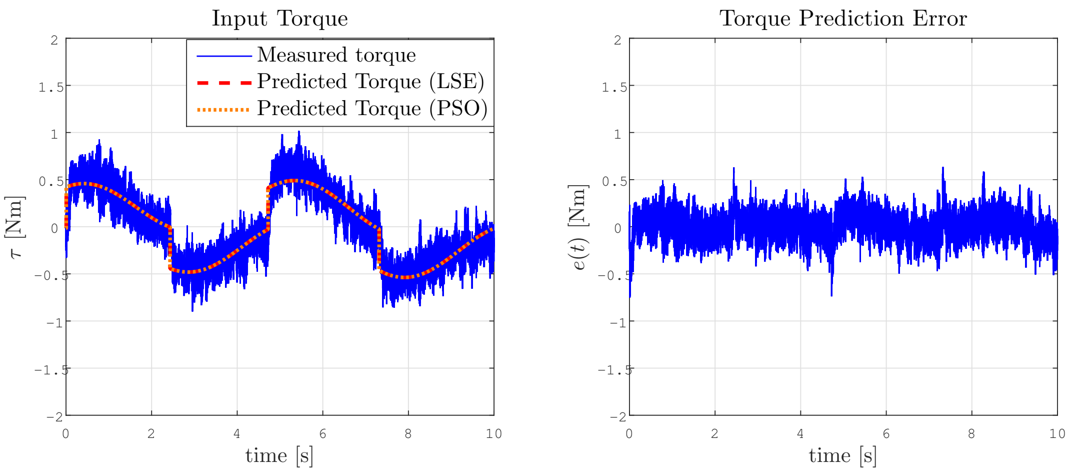


Figure 6.15: Comparison of the estimation with LSE and PSO (left) for *Hand Roll* and the torque prediction error with LSE (right).

Ideally, if the model was exact, the torque prediction error would be the noise of the torque measurement, which is considered to have a normal distribution with zero mean. Moreover, the torque estimation error should be uncorrelated.

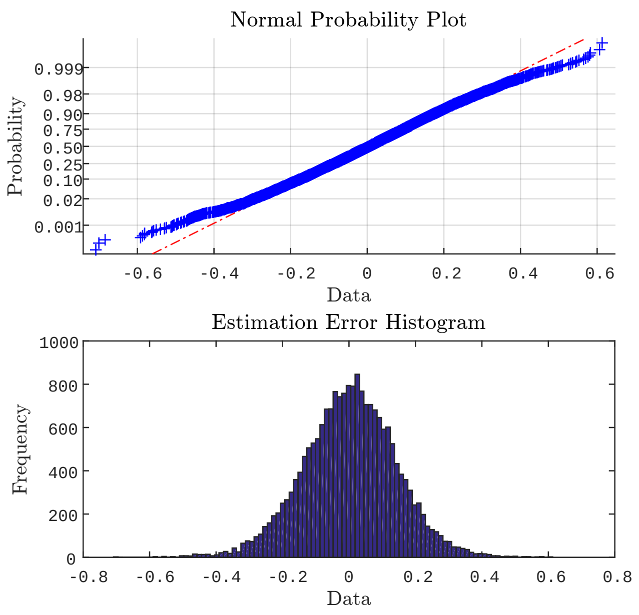


Figure 6.16: The normal probability plot of torque prediction error (upper graph) shows that the data follows a normal distribution. The histogram (lower graph) shows the distribution centered at zero.

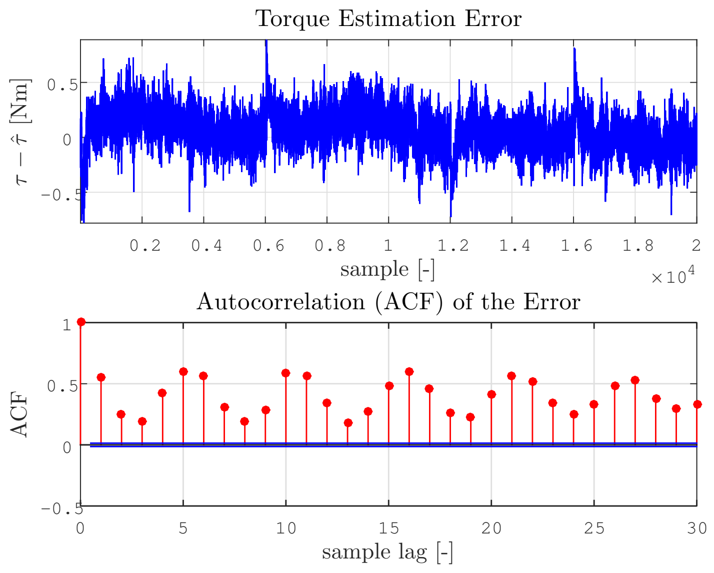


Figure 6.17: A high correlation appears in the error, but it is expected as there will always be model uncertainties when dealing with the real world.

Error appears to be quite correlated. This is because the model is not exact as it is difficult to model the real world. One example is the Coulomb friction, which does not fit exactly the model. Other possible causes are discussed in Sec. 6.7. However, the error appear to have a normal distribution with zero menan as seen in Fig. 6.16. Since the estimation fits quite well the measured data, it is considered valid for the force estimation in Chapter 7.

Hand Pitch (q_2)

Its dynamical equation is found as:

$$\begin{aligned}\tau_2 =&(\eta_2^2 J_{m_2} + Iz_3 + Iy_5 + Iy_6 + Iy_8 + Iy_9 + 0.04\,(m_3 + m_4 + m_5) - 0.38\,m_3 lc_3\\ &+ 0.18(m_6 lc_6 + m_7 lc_7 + m_8 lc_8) - 0.22 m_9 lc_9 + 0.38\,m_5 lc_5 + m_3 {lc_3}^2\\ &+ m_5 {lc_5}^2 + m_6 {lc_6}^2 + m_7 {lc_7}^2 + m_8 {lc_8}^2 + m_9 {lc_9}^2)\,\ddot{q}_2 + v_2\,\dot{q}_2 + c_2\,\mathrm{sign}(\dot{q}_2)\\ &+ (-0.19\,(m_4 - m_3 - m_5) + 0.09\,(m_6 + m_7) + 0.10(m_8 + m_9) + m_3 lc_3\\ &- m_5 lc_5 - m_6 lc_6 - m_7 lc_7 - m_8 lc_8 - m_9 lc_9) g\,\sin(q_2) +\\ &(-0.01\,(m_4 + m_3) + 0.03\,(m_5 + m_6 + m_7 + m_8 + m_9 + m_3 lc_3)) g\,\cos(q_2) \quad [\mathrm{Nm}]\end{aligned} \tag{6.59}$$

which has a total of 21 parameters for PSO. They can be expressed in 5 independent parameters for WLSE/LSE:

$$\tau_2 = \theta_{1,2}\,\ddot{q}_2 + \theta_{2,2} g\,\sin(q_2) + \theta_{3,2} g\,\cos(q_2) + \theta_{4,2}\,\dot{q}_2 + \theta_{5,2}\,\mathrm{sign}(\dot{q}_2) \quad [\mathrm{Nm}] \tag{6.60}$$

In Table 6.5 the initial guesses are compared with the estimations done using WLSE/LSE and PSO:

Parameters	Initial Guess	WLSE	LSE	PSO
$\theta_{1,2}$	0.2961	0.2381	0.2384	0.2406
$\theta_{2,2}$	1.9475	0.1350	0.1343	0.1345
$\theta_{3,2}$	0.115	0.064	0.0705	0.0704
$\theta_{4,2}$	1	0.9110	0.9064	0.9061
$\theta_{5,2}$	0.1	0.4151	0.4155	0.4156
MSE	2.9145	0.0461	0.0460	0.0460

Table 6.5: Table of the estimated parameters for *Hand Pitch*.

Again, results are very similar for all methods, while initial guesses are not. A comparison is also made between the predicted torque from LSE and PSO with the measured torque. The torque prediction error is also shown.

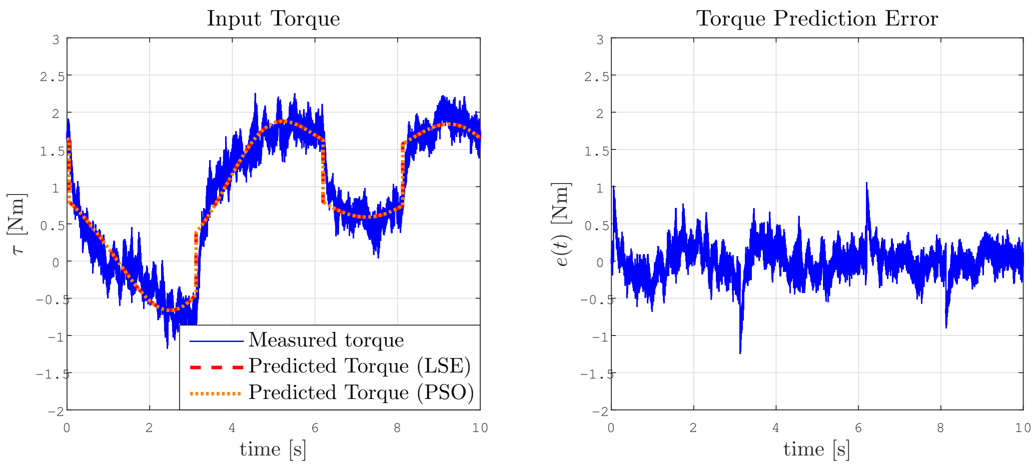


Figure 6.18: Comparison of the estimation with LSE and PSO (left) for *Hand Pitch* and the torque prediction error with LSE (right).

Ideally, the torque prediction error should have normal distribution with zero mean and be uncorrelated. It is clear that due to model uncertainties, it is not uncorrelated and the autocorrelation is not shown. However, the normal plot and the histogram in Fig. 6.19 show that the error mean is zero with a higher variance than a normal distribution. Nevertheless, the estimation fits well the measured data and therefore, it is considered a valid model.

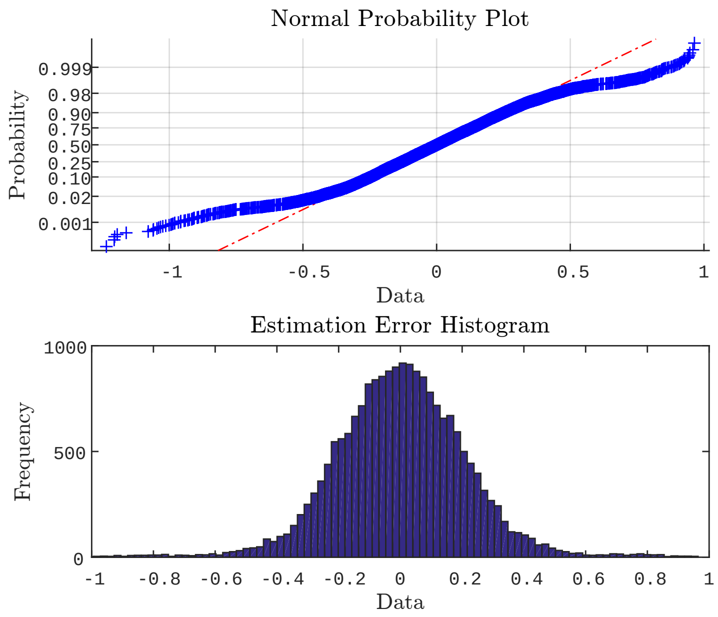


Figure 6.19: The normal probability plot of torque prediction error (upper graph) show that the distribution has with higher variance than expected. The histogram (lower graph) shows that it is a distribution with zero mean.

Instrument Slide (q_3)

Its dynamical equation is found as:

$$\tau_3 = \left(\eta_3^2 J_{m_3} + m_6 + m_7 + m_8 + m_9\right) \ddot{q}_3 - (m_6 + m_7 + m_8 + m_9)g + v_3\dot{q}_3 + c_3\text{sign}(\dot{q}_3) \quad [\text{N}]$$

which has a total of 7 parameters to be used in PSO, and can be expressed in 4 independent parameters for WLSE/LSE as:

$$\tau_3 = \theta_{1,3}(\ddot{q}_3 - g) + \theta_{2,3}\ddot{q}_3 + \theta_{3,3}\,\dot{q}_3 + \theta_{4,3}\,\text{sign}(\dot{q}_3) \quad [\text{N}] \tag{6.61}$$

In Table 6.6 the initial guesses are compared with the estimations done using WLSE/LSE and PSO.

Parameters	Initial Guess	WLSE	LSE	PSO
$\theta_{1,3}$	0.141	2.4610	2.5626	2.485
$\theta_{2,3}$	10^{-6}	-0.0750	-0.0712	0.0003
$\theta_{3,3}$	1	3.2723	3.2723	3.1939
$\theta_{4,3}$	3	3.9842	3.9840	3.9975
MSE	8.8435	1.8100	1.8101	2.2986

Table 6.6: Table of the estimated parameters for *Instrument Slide*.

Notice that the results for this joint are not as good as with the other two. Moreover, the value $\theta_{2,3}$ obtained from WLSE and LSE is not physically possible as the motor inertia J_{m_3} should be positive, while in PSO it was constrained. However, the model found with LSE is the one that gives the lowest MSE and therefore, it is chosen for the force estimation. The test was conducted at high speed, however the joint is never gonna be moved fast and the dynamics will not be so present at smaller velocities. Then, only friction will be seen at the joint. Moreover, autocorrelation or histogram plot of the error will not be shown for this joint, as the error is far from ideal and no further conclusions can be obtained from them.

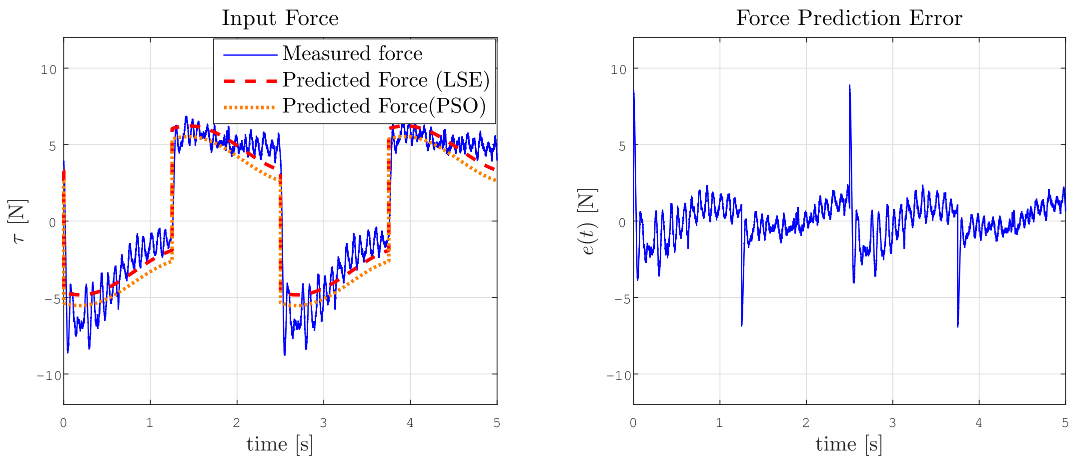


Figure 6.20: Comparison of the estimation with LSE and PSO (left) for *Instrument Slide* and the force prediction error with LSE (right).

6.7 Error Propagation

Once the estimation results are obtained, it can be seen that the model error is not totally ideal. This may be caused by several reason which are considered here:

- Filtering the position may remove part of the real trajectory, even though it is practically the same and it helps to obtain free of noise velocity and acceleration.
- As the fundamental frequency ω_f and the number of Fourier elements N_f of the optimal trajectory are chosen beforehand in Sec. 6.3.1, it might be that the trajectory found is not sufficiently exciting the system. Then, when joints are moved at higher velocities, the model may not be precise enough.
- The friction in the joints does not follow exactly the theoretical model. Actually, this friction is not symmetric as it behaves differently in both direction. The position is increased slowly in the *Instrument Slide* joint such that the

velocity and acceleration terms can be neglected. Also, it is moved horizontally such that the gravity term is zero. This way, the motor force generated is equal to the friction forces. As seen in Fig. 6.21, hysteresis occurs in the friction. Moreover, Coulomb friction does not behave as a sign function even though it is approximated as such. So it is important to note that the model used does not fit exactly the real world behavior.

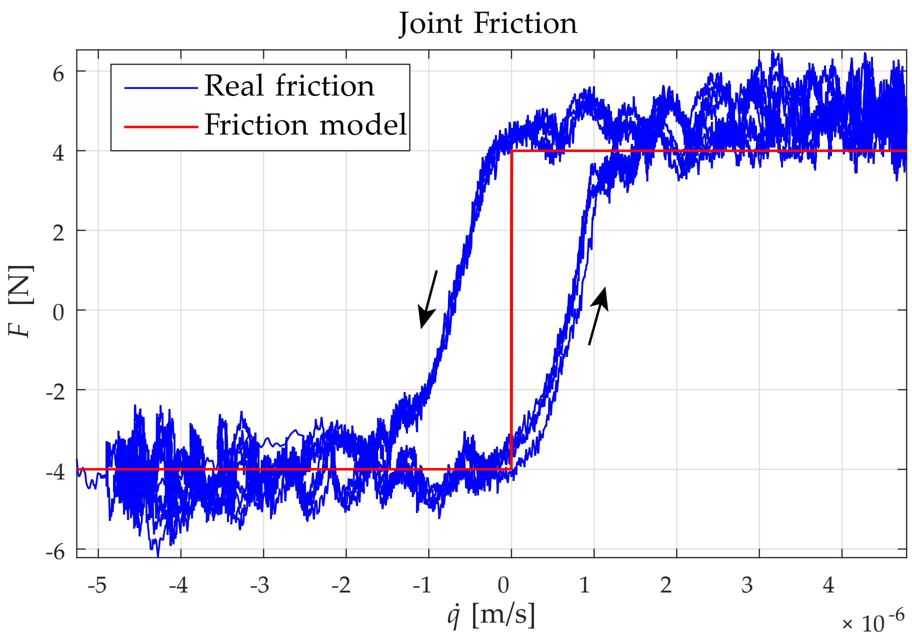


Figure 6.21: Friction of the *Instrument Slide* joint compared with the model used.

- The controllers in the da Vinci robot are used to follow the optimal trajectory. Therefore, the estimation is made in closed loop and the fastest dynamics that appear in all joints may be caused by the controller.
- The model for each joint is considered independent to the others. However, joints with high inertias like *Hand Roll* or *Hand Pitch* may be affected by each other. Joints like *Instrument Slide* are not so affected by dynamics but mostly friction. Therefore, the independent model may be good for some joints but not all. One could use an alternative estimation method where the controller is included in the model [Hof and Schrama, 1994].

Despite the model is not ideal, it is never easy to obtain accurate results when dealing with the real world. Therefore, the estimation is considered good to be used in the force estimation in Chapter 7.

6.8 Conclusions for Estimation Results

In this chapter, an experiment was performed in order to identify the robot parameters. A permanent exciting trajectory, which was periodic, was designed. The method proved to remove the noise present in the position measurement and it allowed to compute the velocity and the acceleration free of noise, while noise in the torque measurements was reduced.

The identification was simplified by moving one joint at a time and fixing the others, which reduced the total number of parameters that could be estimated. It is possible that joints *Hand Roll* and *Hand Pitch* are dependent on other joints, while *Instrument Slide* is mostly affected by friction. Therefore, the independent joint model may be good for certain joints but not all. Two approaches were used to estimate the parameters by using the measured data from the experiment:

- First, the Weighted and non weighted Least Square Estimation are derived. Their results show that there is no significant improvement when weighting the data, so there is no need to use WLSE.
- On the other hand, the algorithm Particle Swarm Optimization also gave similar results to LSE while skipping parametrization of the dynamics. Moreover, constraints can be included in PSO while in LSE cannot. Therefore, it is a good alternative method when parametrization is not an easy task. However, due to the physical constraints, results for the joint *Instrument Slide* were worst.

The prediction error was used to verify the accuracy of the estimation. The model was good except for velocities close to zero due to the limited accuracy at low velocity of the Coulomb model used. Furthermore, the results for the *Instrument Slide* joint are worse compared to the other two. However, the models found are considered to be good enough for force estimation in Chapter 7.

Chapter 7

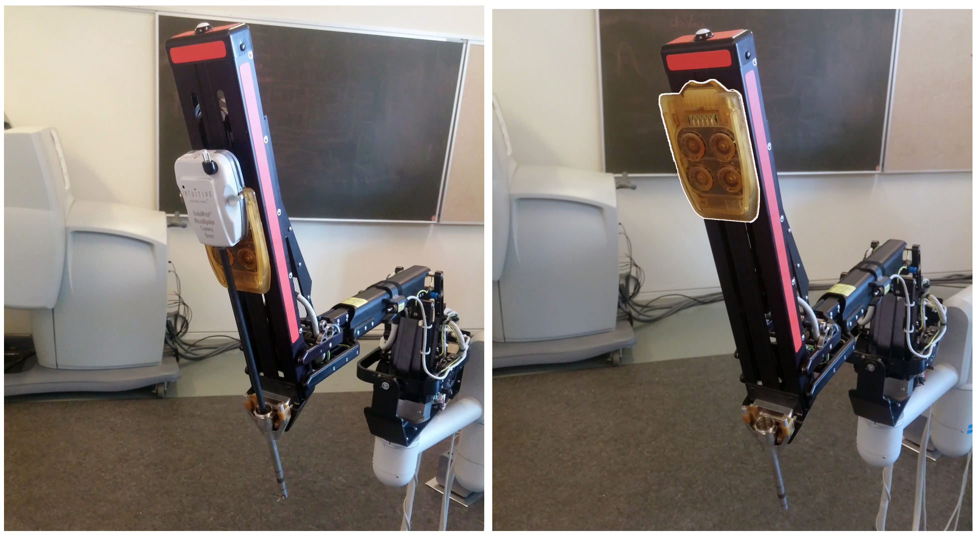
External Forces Estimation

The purpose of this chapter is to estimate external forces using the models found in Chapter 6. Once the external forces are estimated, they can then be used for force feedback in a haptic device in order to create a contact feeling to the surgeon.

In the da Vinci surgical system, currently the surgeon does not have feeling of applied force during operation and therefore, it may worsen his experience. A survey was carried out in [Trejos et al., 2010] where it was stated the need of force sensing in minimally invasive surgery. The first consideration would be to add force sensors at the end effector such that they can measure the force. In practice, it is not so easy to add anything to the end effector as it will prevent a normal use during surgery. Moreover, in case a sensor could be placed on the tool, two problems would arise:

- All sensors that are in contact with the patient should be able to withstand a sterilization procedure [Trejos et al., 2010]. This implies that the choice of the force sensor is limited.
- In the da Vinci robot, each tool is limited to a specific number of uses. As the sensors should be placed at the tool, their lifetime is also short. Then, the cost would increase considerably.

One could place torque sensors where the instrument is attached, such that the they are not inside the patient during surgery and also, such that they should not be removed when an instrument is changed as shown in Fig. 7.1.



(a) Instrument to be removed. **(b)** Instrument holder (outlined).

Figure 7.1: Force sensors could be placed at the instrument holder as it is not removed when instrument is changed.

Nevertheless, to avoid adding any sensor, the alternative is to estimate these contact forces using available measurements from the motors (i.e. current and position measurement), which is commonly used in haptic feedback as it is desired in this project. Furthermore, it is important to have a known model, including friction dynamics, in order to obtain accurate results.

A very common approach is to use a disturbance observer, which compares the difference between the outputs of the nominal model (i.e. without external forces) and the measured system output, as it is done in [Eom et al., 1998]. An extended Kalman filter is also used in [Jung et al., 2006] and [Lee and Ahn, 2010] to estimate external forces, which are considered disturbances. A different approach has been done in [Stolt, 2015] and [Wahrburg et al., 2014], where a convex optimization problem is solved, but still using measurements of joint angles and motor current. In [Stolt, 2015], a probabilistic disturbance model was developed for the friction and then, external forces were estimated by solving a convex optimization problem. This method is focused on estimating the force in industrial robots when velocities are very small, as dynamics from acceleration are very low and neglected. All methods agree that modeling the friction is an important part in force estimation. From the referenced work about force estimation, only [Lee and Ahn, 2010] has actually been used in telesurgery applications, while the other examples are related to industrial robots.

This chapter is based on the previous referenced work and is divided as follows:

- First, the model used for force estimation is described.
- An extended Kalman filter is used to estimate torques at three joint by using the models in Chapter 6, while the remaining joints are considered to be fixed.
- Then, a relation between forces applied at the end-effector and the estimated torques at the joints is given, where the Jacobian of the da Vinci robot is used.
- Finally, results from the extended Kalman filter for each joint are shown and discussed.

7.1 Extended Kalman Filter for Force Estimation

If a model of the system is known, then τ_{ext} can be estimated using this model. First, lets consider the system from Chapter 4:

$$\tau = \mathbf{M}(q)\ddot{q} + V(\dot{q}, q) + F(\dot{q}) + G(q) + \tau_{ext} \quad \text{[Nm]} \tag{7.1}$$

where

τ	is a vector of motor torques [Nm]
q	is an $n \times 1$ vector of joint position [rad]
$\mathbf{M}(q)$	is a symmetric mass matrix of manipulator and motor [kg m^2]
$V(\dot{q}, q)$	is an $n \times 1$ vector of centrifugal and Coriolis terms [Nm]
$G(q)$	is an $n \times 1$ vector of gravity terms [Nm]
$F(\dot{q})$	is an $n \times 1$ vector of friction of motors and joints [Nm]
τ_{ext}	is an $n \times 1$ vector of external torques seen at the joints [Nm]

Notice that units are for **revolute** joints. If joint is **prismatic**, units of q are [m], gear ratio is [1/m] and instead of torques, there are forces [N].

Intuitively, one could say that by using the model in Eq. (7.1) and the parameter estimation made in Chapter 6, the external joint torques could be estimated simply using:

$$\hat{\tau}_{ext} = \tau - \mathbf{M}(q)\ddot{q} - V(\dot{q}, q) - F(\dot{q}) - G(q) \tag{7.2}$$

Unfortunately, this estimation will include an error e, which consist of both unmodeled dynamics and sensor noise from τ:

$$\hat{\tau}_{ext} = \tau_{ext} + e \tag{7.3}$$

In order to show it, a small torque is applied to the Hand Pitch joint and the external torque is estimated in Fig. 7.2 by using Eq. (7.2). Even though it appears that external torques can be estimated, there is a lot of noise present.

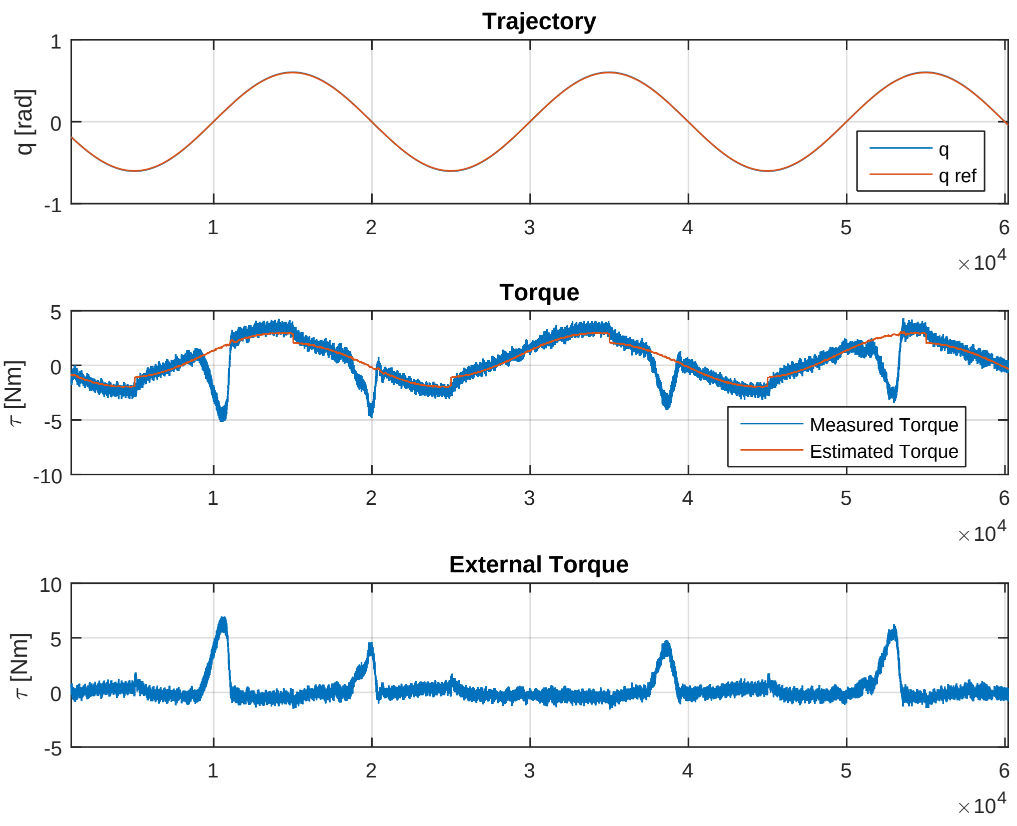


Figure 7.2: The *Hand Pitch* joint is following a given trajectory while external torques are applied to this joint. The motor reacts to them to keep following the trajectory and a noisy torque is computed from Eq. (7.2).

While the error due to unmodeled dynamics can only be removed by improving the model, which is not an easy task, the measurement noise can be removed if a filter is applied. In these cases, it is very common to use a Kalman filter as it is done in [Jung et al., 2006] and [Lee and Ahn, 2010]. For instance, in [Lee and Ahn, 2010] an adaptive Kalman Filter (AKF) is used for force estimation in a telesurgery application, where an adaptive rule for the measurement covariance matrix $\mathbf{R}$ is suggested. Then, external forces are considered as a disturbance and estimated. In this project, a non linear model is found, therefore, the discrete-time Extended Kalman Filter (EKF) for each joint is used instead.

7.1.1 Extended Kalman Filter Algorithm

First, lets define the nonlinear dynamic model and measurement model in discrete for the general case of the EKF, where the system and measurement noise are considered to be white and uncorrelated [Grewal and Andrews, 2008]:

$$\begin{aligned} x_{k+1} &= f_d(x_k, u_k) + w_k \qquad w_k \sim \mathcal{N}(0, \mathbf{Q}_k) \\ y_k &= h_d(x_k, u_k) + v_k \qquad v_k \sim \mathcal{N}(0, \mathbf{R}_k) \\ &E(w\, v^T) = 0 \end{aligned} \tag{7.4}$$

where

x_k	is the system state vector at time k
y_k	is the observation vector of the system at time k
$f_d(x_k, u_k)$	is the non linear discrete system equation
$h_d(x_k, u_k)$	is the non linear discrete measurement equation
$\mathcal{N}(\mu, \mathbf{Q})$	denotes a Gaussian distribution with mean μ and covariance $\mathbf{Q}$
w_k	is the system noise with zero mean and covariance $\mathbf{Q}_k$
$\mathbf{Q}_k$	is the system noise covariance matrix
v_k	is the measurement noise with zero mean and covariance $\mathbf{R}_k$
$\mathbf{R}_k$	is the measurement noise covariance matrix

The EKF uses the derivation of the KF and then approximates and linearizes the non linear terms only when necessary. Therefore, f_d and h_d should be linearized with respect to x:

$$\mathbf{F}_k = \left.\frac{\partial f_d(x,u)}{\partial x^T}\right|_{\hat{x}_{k|k}, u_k} \qquad \mathbf{H}_k = \left.\frac{\partial h_d(x,u)}{\partial x^T}\right|_{\hat{x}_{k|k-1}, u_k} \tag{7.5}$$

where $\hat{x}_{k|k}$ is the estimation of x_k given data at time k and $\hat{x}_{k|k-1}$ is the estimation of x_k given data at time $k-1$. This notation is now used for the recursive algorithm of the EKF, which is divided in three parts and described in [Grewal and Andrews, 2008]:

- **Initialization** of the algorithm:

$$\begin{aligned} \hat{x}_{0|-1} &= 0 \\ \mathbf{P}_{0|-1} &= \mathbf{Q}_0 \end{aligned}$$

- **Measurement update** after receiving y_k and u_k, where the previous state prediction is compared with the actual measurement and then corrected:

$$\begin{aligned} \hat{y}_{k|k-1} &= h_d(\hat{x}_{k|k-1}, u_k) \\ \tilde{y}_{k|k-1} &= y_k - \hat{y}_{k|k-1} \\ \mathbf{K}_k &= \mathbf{P}_{k|k-1}\mathbf{H}_k^T \left(\mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k\right)^{-1} \\ \hat{x}_{k|k} &= \hat{x}_{k|k-1} + \mathbf{K}_k\, \tilde{y}_{k|k-1} \\ \mathbf{P}_{k|k} &= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k|k-1}(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)^T + \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^T \end{aligned} \tag{7.6}$$

- **Time update** from k to $k+1$, where the next state and covariance matrix $\mathbf{P}$ are predicted:

$$\begin{aligned} \hat{x}_{k+1|k} &= f_d(\hat{x}_{k|k}, u_k) \\ \mathbf{P}_{k+1|k} &= \mathbf{F}_k \mathbf{P}_{k|k} \mathbf{F}_k^T + \mathbf{Q}_k \end{aligned} \tag{7.7}$$

  where

$\hat{y}_{k\|k-1}$	is the predicted measurement at k from estimation of $\hat{x}_k$ made at $k-1$
$\tilde{y}_{k\|k-1}$	is the innovation or measurement residual at time k
$\mathbf{K}_k$	is the Kalman gain
$\hat{x}_{k\|k}$	is the corrected predicted state using the Kalman gain
$\mathbf{P}_{k\|k-1}$	is the a priori covariance matrix of the states estimate at time k
$\mathbf{P}_{k\|k}$	is the a posteriori covariance matrix of the states estimate at time k
$\hat{x}_{k+1\|k}$	is the corrected predicted state using the Kalman gain

The predicted state $\hat{x}_{k|k}$ is the corrected state of the system with the Kalman gain and therefore, it is the desired information from the Kalman.

7.1.2 System Model

A model is needed for the estimation of external torques τ_{ext} in the joints. For any robot manipulator such as the da Vinci robot, the system model was given as:

$$\tau = \mathbf{M}(q)\ddot{q} + V(\dot{q}, q) + F(\dot{q}) + G(q) + \tau_{ext} \tag{7.8}$$

Note that the independent joint dynamics defined and estimated in Chapter 6 are used in Eq. (7.8). Considering $\dot{q} = dq$, then it can be rewritten as First-Order Ordinary Differential Equations:

$$\begin{cases} \dot{q} = dq \\ \dot{dq} = \mathbf{M}(q)^{-1}\left(\tau - V(dq, q) - F(dq) - G(q) - \tau_{ext}\right) \end{cases} \tag{7.9}$$

Notice that τ_{ext} is considered a disturbance and in order to estimate it using the EKF, it should be represented as a state. As the external torques are unknown, one common assumption is to consider that this disturbance is slowly varying and practically constant, which can be represented as:

$$\dot{\tau}_{ext} = 0 \tag{7.10}$$

Which gives the following system equation:

$$\begin{cases} \dot{q} = dq \\ \dot{dq} = \mathbf{M}(q)^{-1}\left(\tau - V(dq, q) - F(dq) - G(q) - \tau_{ext}\right) \\ \dot{\tau}_{ext} = 0 \end{cases} \tag{7.11}$$

where q, dq and τ_{ext} are the states of the system and τ is the input. Moreover, in the da Vinci, position and velocity measurement are available, so the measurement equation becomes:

$$y = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}}_{\mathbf{H}} \begin{bmatrix} q \\ dq \\ \tau_{ext} \end{bmatrix} \tag{7.12}$$

Both Eq. (7.11) and (7.12) can be expressed as the following continuous system:

$$\begin{aligned} \dot{x} &= f(x, u) \\ y &= \mathbf{H}\, x \end{aligned} \tag{7.13}$$

where the output equation is linear but not the system equation. As the EKF algorithm is implemented in discrete, the dynamical model should also be discrete. For this reason, Eq. (7.13) is discretized by using the Forward Euler method as:

$$\begin{aligned} x_{k+1} &= x_k + T_s\, f(x_k, u_k) \\ y_k &= \mathbf{H}\, x_k \end{aligned} \tag{7.14}$$

where T_s is the sampling time. Therefore, the discrete model can be written as:

$$\begin{cases} q_{k+1} = q_{k+1} + T_s\, dq_k \\ dq_{k+1} = dq_k + T_s\, \mathbf{M}(q_k)^{-1} \left(\tau_k - V(dq_k, q_k) - F(dq_k) - G(q_k) - \tau_{ext_k}\right) \\ \tau_{ext_{k+1}} = \tau_{ext_k} \end{cases} \tag{7.15}$$

Finally, the non linear discrete system in Eq. (7.14) should be linearized at $\hat{x}_{k|k}$ as it is needed to compute the a priori covariance matrix $\mathbf{P}_{k+1|k}$ in the time update step in (7.7), that is:

$$\mathbf{F}_k = \mathbf{I} + T_s \left. \frac{\partial f(x, u)}{\partial x^T} \right|_{\hat{x}_{k|k}, u_k} \tag{7.16}$$

while the measurement equation in Eq. (7.14) is already linear. Notice that the Coulomb model used is $F_c = C\,\text{sign}(\dot{q})$, which is not differentiable at $\dot{q} = 0$. For this reason, it is assumed that the system will never be exactly at $\dot{q} = 0$ and then, the differentiation is zero.

7.2 Joint Torques to End-Effector Forces

Now that τ_{ext} is estimated with the EKF, the Jacobian $\mathbf{J}$ can be used to find also the forces/torques applied at the end-effector. Therefore, the end-effector force/ torque vector F_{ee} is related with the joint torque τ_{ext} using the robot Jacobian matrix [Linderoth et al., 2013]:

$$\tau_{ext} = \mathbf{J}^T F_{ee} \quad [\text{Nm}] \tag{7.17}$$

where the robot Jacobian $\mathbf{J}$ is defined in Chapter 3 during the kinematics study of the hand and tool. Both the end effector force/torque vector F_{ee} and the Jacobian $\mathbf{J}$ can be partitioned as follows:

$$\tau_{ext} = \mathbf{J}^T F_{ee} = \begin{bmatrix} \boldsymbol{J_v}^T & \boldsymbol{J_\omega}^T \end{bmatrix} \begin{bmatrix} f_{ee} \\ \tau_{ee} \end{bmatrix} \tag{7.18}$$

where f_{ee} is a 3x1 vector of external forces at the end-effector [N] and τ_{ee} is a 3x1 vector of external moments applied at the end-effector [Nm]. It is now necessary to compute the translational $\boldsymbol{J_v}$ and the rotational $\boldsymbol{J_\omega}$ part of the Jacobian, which map joint velocities with end-effector linear and angular velocity respectively:

$$\begin{aligned} {}^0v_n &= \boldsymbol{J_v}\dot{q} \quad [\text{m/s}] \\ {}^0\omega_n &= \boldsymbol{J_\omega}\dot{q} \quad [\text{rad/s}] \end{aligned} \tag{7.19}$$

7.2.1 Translational Jacobian

To find $\boldsymbol{J_v}$, the position of the last link n, the end-effector, should be known with respect to frame $\{0\}$. Lets consider the transformation matrix from the end-effector frame to the base frame, which can be computed by concatenating intermediate frames as explained in Chapter 3:

$${}^0_n\mathbf{T}(q) = {}^0_1\mathbf{T} \cdot {}^1_2\mathbf{T} \cdots {}^{n-1}_{n}\mathbf{T} = \begin{pmatrix} {}^0_n\mathbf{R}(q) & {}^0O_n(q) \\ 0_{1x3} & 1 \end{pmatrix} \tag{7.20}$$

where

${}^0O_n(q)$	is the origin of the end-effector frame with respect to the base frame
${}^0_n\mathbf{R}(q)$	is the rotation from the end-effector frame with respect to base frame

Now, the term $\boldsymbol{J_v}$ can be computed with the partial derivative of the position of the end-effector ${}^0O_n(q)$ with respect to the joint positions [Spong et al., 2004]:

$$\boldsymbol{J_v} = \frac{\partial\, {}^0O_n(q)}{\partial q} \tag{7.21}$$

7.2.2 Rotational Jacobian

To find $\boldsymbol{J_\omega}$ becomes easier if the angular velocity of the end effector is computed, which is defined as the sum of the previous link's angular velocity with respect to frame $\{0\}$:

$${}^0\omega_n = {}^0\omega_1 + {}^0_1\mathbf{R} \cdot {}^1\omega_2 + {}^0_2\mathbf{R} \cdot {}^2\omega_3 + \cdots + {}^0_{n-1}\mathbf{R} \cdot {}^{n-1}\omega_n \quad [\text{rad/s}] \tag{7.22}$$

where ${}^{i-1}\omega_i$ is the rotation of joint i measured along z axis of joint $i-1$, which is the generalized velocity $\dot{q}\hat{k}$ if joint is **revolute** and $0\hat{k}$ if joint is **prismatic**, and $\hat{k} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$. By looking to the generalized coordinates q defined in Chapter 3, the term ${}^{i-1}\omega_i$ can be determined and the angular velocity of the end-effector becomes:

$$ {}^{0}\omega_n = {}^{0}_{1}\mathbf{R} \cdot \dot{q}_1\hat{k} + {}^{0}_{2}\mathbf{R} \cdot \dot{q}_2\hat{k} + {}^{0}_{3}\mathbf{R} \cdot \dot{q}_2\hat{k} + {}^{0}_{4}\mathbf{R} \cdot \dot{q}_2\hat{k} + {}^{0}_{6}\mathbf{R} \cdot \dot{q}_4\hat{k} + {}^{0}_{7}\mathbf{R} \cdot \dot{q}_5\hat{k} + {}^{0}_{8}\mathbf{R} \cdot \dot{q}_6\hat{k} \quad [\text{rad/s}] \tag{7.23} $$

Notice that the angular Jacobian from Eq. (7.2) relates end-effector angular velocity with joint velocity. Therefore, the contribution of the joint velocity $\dot{q}_i$ (six in total) to the end-effector angular velocity ${}^{0}\omega_n$ can be used to find $\mathbf{J}_{\omega}$, that is:

$$ J_\omega = \begin{bmatrix} {}^{0}_{1}\mathbf{R}\,\hat{k} & \left({}^{0}_{2}\mathbf{R}\,\hat{k} + {}^{0}_{3}\mathbf{R}\,\hat{k} + {}^{0}_{4}\mathbf{R}\,\hat{k} \right) & 0_{3x1} & {}^{0}_{6}\mathbf{R}\,\hat{k} & {}^{0}_{7}\mathbf{R}\,\hat{k} & {}^{0}_{8}\mathbf{R}\,\hat{k} \end{bmatrix} \tag{7.24} $$

7.2.3 Reduced Jacobian

The calculation of the Jacobian is done in Maple. For simplicity, only the joints *Hand Roll, Hand Pitch* and *Instrument Slide*, shown in Fig. 7.3, are used in the model, while the rest are fixed. As mentioned in Chapter 6, to use the remaining joints is the first improvement to the project that should be done after the initial design. Moreover, by assuming that only forces are applied at the end-effector, then Eq. (7.18) becomes:

$$ \tau_{ext} = J_v{}^T f_{ee} \tag{7.25} $$

where $J_v{}^T$ is a 3×3 matrix, as only three joints are used in this model. As $J_v{}^T$ is square, it can now be inverted in order to obtain forces at the end-effector f_{ee}:

$$ f_{ee} = \left(J_v{}^T \right)^{-1} \tau_{ext} \tag{7.26} $$

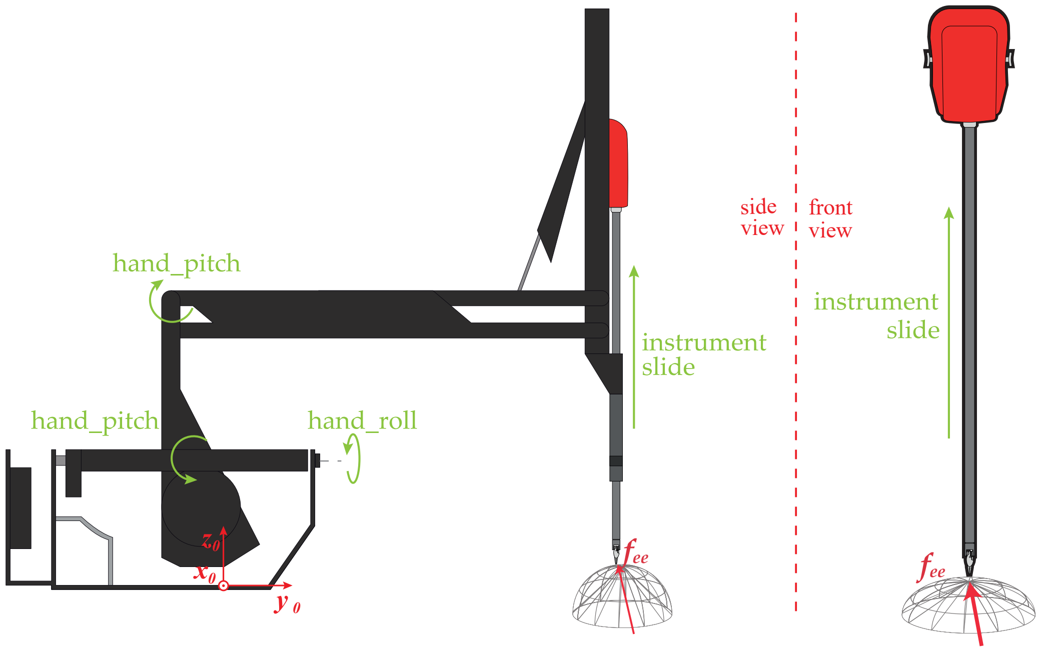


Figure 7.3: Joint movements that are used to estimate external forces f_{ee} at the end-effector. The outlined instrument will move only with a slide movement and the end-effector is assumed to be fixed and not moving.

7.3 Force Estimation Results in Simulation

The extended Kalman filter is first tested in the simulation. All measurements are sampled at a rate of 100 Hz. Zero-mean Gaussian noise is added to the measurement according to the standard deviation shown in Table 7.1. The two measurements have different characteristics, since velocity is computed from position measurement and its noise variance is higher.

Measured Variable	Symbol	Standard deviation
Joint Position	q	0.01 rad
Joint Velocity	dq	0.008 rad/s

Table 7.1: Table of standard deviation of the noise added to simulation measurements.

Only one joint is tested in simulation, the *Hand Pitch*, in order to prove that external forces can be estimated by the EKF designed. The experiment is shown in Fig. 7.4 and summarized as follows:

- A sinusoidal trajectory is given to each joint.

- External torques are applied periodically to the joint. A triangle wave generator is used to create them.
- The controller reacts very fast when external forces are applied in order to keep following the reference.

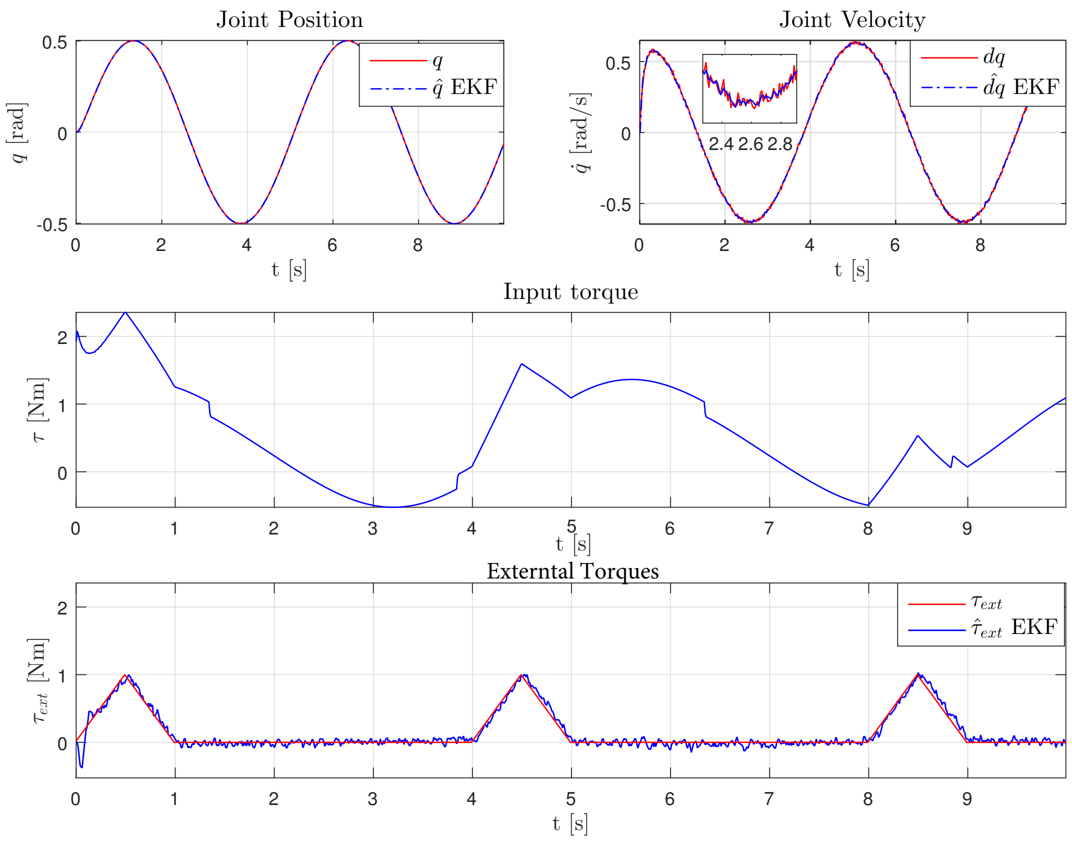


Figure 7.4: The upper plots shows the measurements of the system. The middle plot shows the input torque, which reacts to external torques. The lower plot compares the external torques applied with the estimation form the EKF.

Results show that the controller designed in Chapter 5 reacts very fast to external forces applied. Moreover, the EKF is able to capture the external torques as it is desired. The system and measurement noise covariance matrices are tuned in order to obtain uncorrelated measurement residuals (innovation) as seen in Fig. 7.5. Note that they could be tuned further in order to reduce the noise in the estimation of external torques, however simulation is only used to prove that external torques can be estimated with the EKF designed.

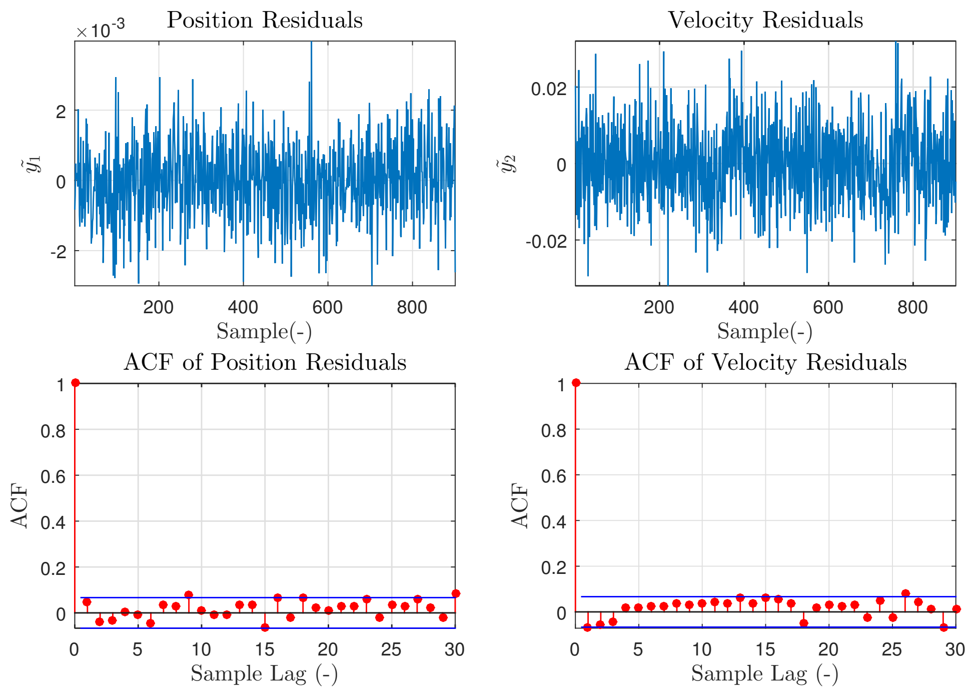


Figure 7.5: Residuals for the joint *Hand Pitch* (simulation) and its autocorrelation.

The residuals for both measurement appear to be uncorrelated as expected. In simulation, the system is ideal and the model used in the EKF is exact.

7.4 Force Estimation Results in da Vinci Robot

The model used for the EKF is based on the independent joint dynamics described and estimated in Chapter 6. Each joint is now tested separately in the da Vinci robot, where:

- A sinusoidal trajectory, different to the one used for estimation, is given to each joint.
- Forces/torques τ_{ext} are manually applied to the joint in order to see how the EKF is able to capture them. Its characteristics are unknown but the time when they are applied is known and thus, it is noted in the plots.
- The input force/torque measured is compared with the input force/torque that is obtained from the model, which is given as:

$$\hat{\tau} = \mathbf{M}(q)\ddot{q} + V(\dot{q}, q) + F(\dot{q}) + G(q) \tag{7.27}$$

- The estimated external force/torque from the EKF is compared with the one obtained from Eq. (7.2), which is:

$$\hat{\tau}_{ext} = \tau - \mathbf{M}(q)\ddot{q} - V(\dot{q}, q) - F(\dot{q}) - G(q) = \tau - \hat{\tau} \tag{7.28}$$

Hand Roll

Results for this joint show high peaks in input torque when external torques are applied. The EKF is actually able to capture these changes in the input very fast, while it remains close to zero when no torque is manually applied to the joint.

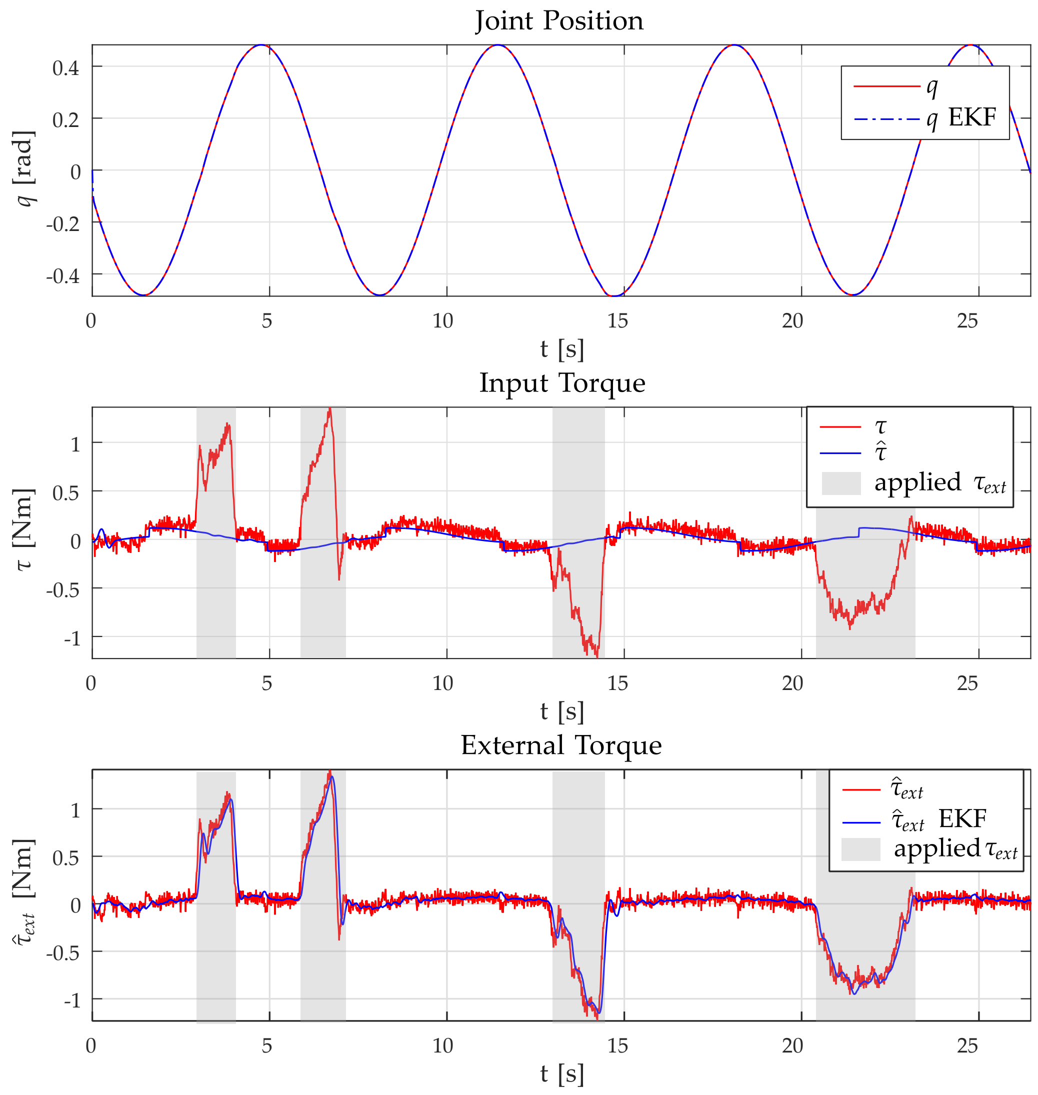


Figure 7.6: The upper graph shows the trajectory followed by the *Hand Roll*. In the middle plot, the input torque measured is compared with the torque from the model. Finally, the lower graph shows the external torque estimated.

Hand Pitch

Results for this joint also show that the EKF is able to capture external torques successfully, while it remains close to zero when no torque is applied.

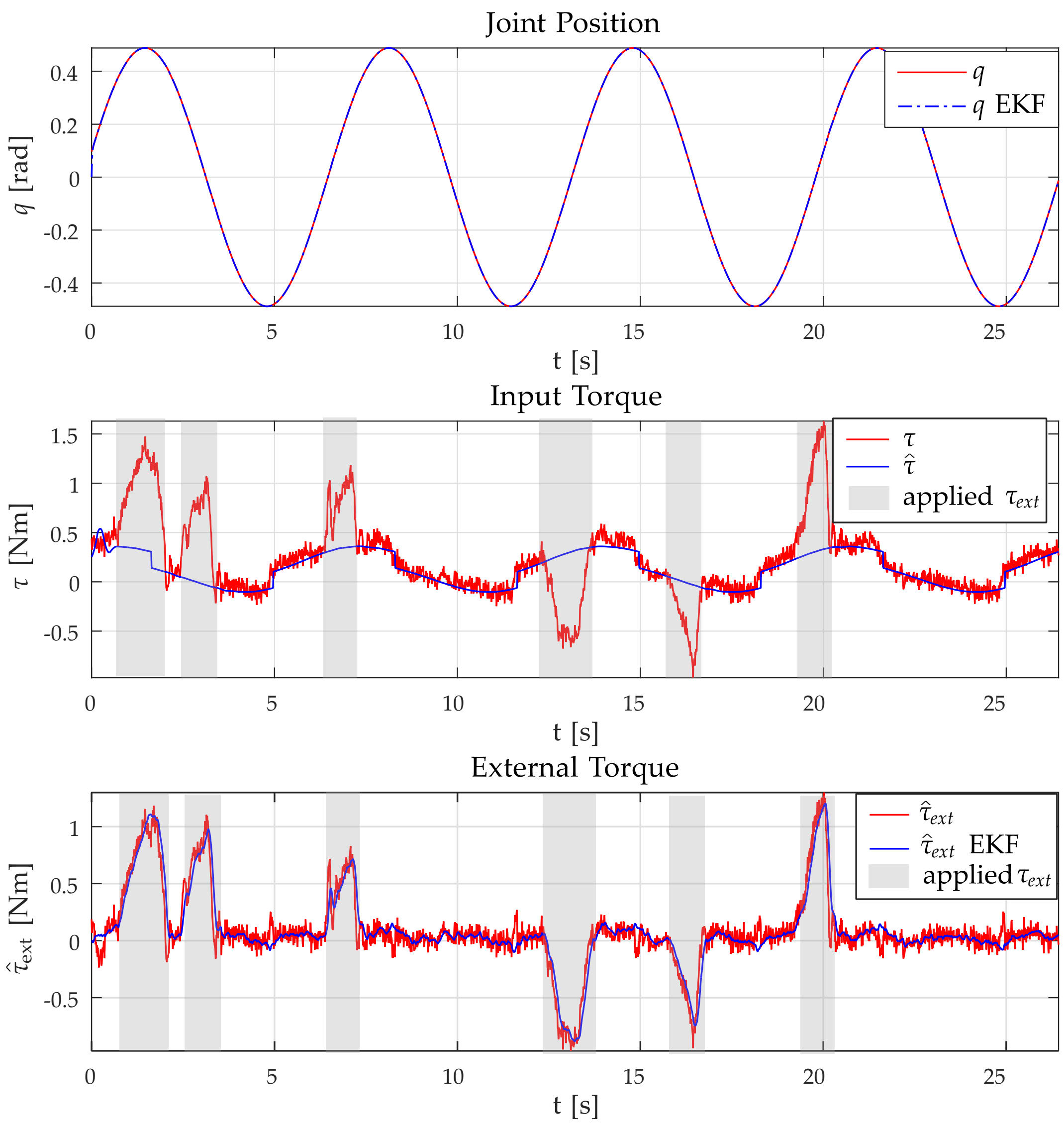


Figure 7.7: The upper graph shows the trajectory followed by the *Hand Pitch*. In the middle plot, the input torque measured is compared with the torque from the model. Finally, the lower graph shows the external torque estimated.

Instrument Slide

The model for this joint found in Chapter 6 is known to be inaccurate. However, the force applied is quite big compared to the force needed to move the joint.

Therefore, the EKF can capture external forces and remain close to zero when they are not applied. Moreover, the Coulomb model also create errors in the external torque $\hat{\tau}$ from Eq. (7.28) when velocity changes sign. However, the EKF is able to smooth this error.

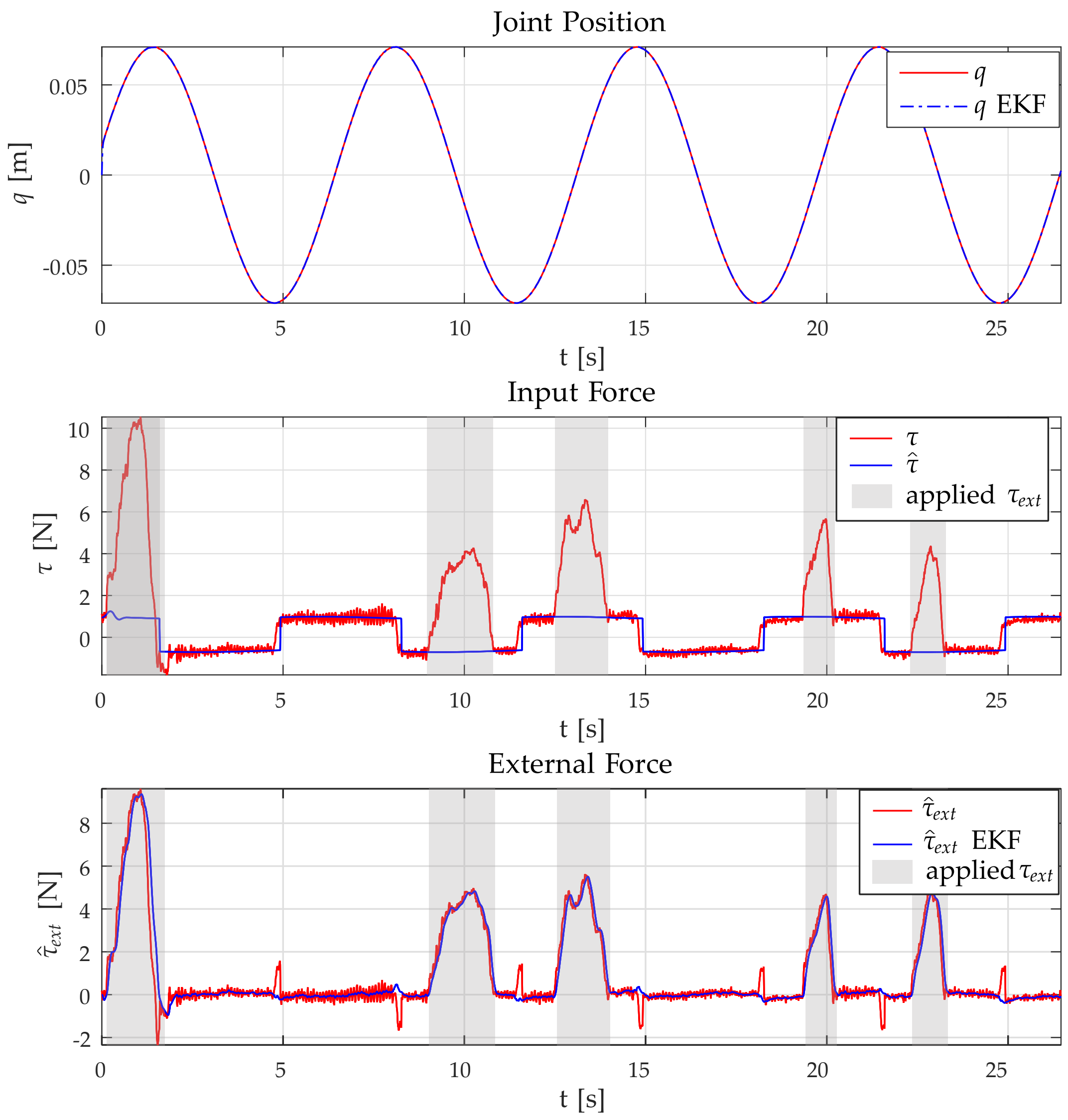


Figure 7.8: The upper graph shows the trajectory followed by the *Instrument Slide*. In the middle plot, the input torque measured is compared with the torque from the model. Finally, the lower graph shows the external torque estimated.

Correlation of the Error

The measurement residuals from the EKF should be white noise as the measurement noise that was assumed in the model. However, the model considered external forces as a constant disturbance and they are not. Moreover, the friction model (Coulomb) is also known to be bad when velocities are close to zero. Therefore, it is expected that residuals are correlated when velocity is close to zero and when non constant external forces are applied. Therefore, the autocorrelation of the error is now checked when none of the conditions mentioned above are fulfilled in order to verify the rest of the model. The system and measurement noise covariance matrices from the EKF are tuned in order to obtain the most uncorrelated residuals possible.

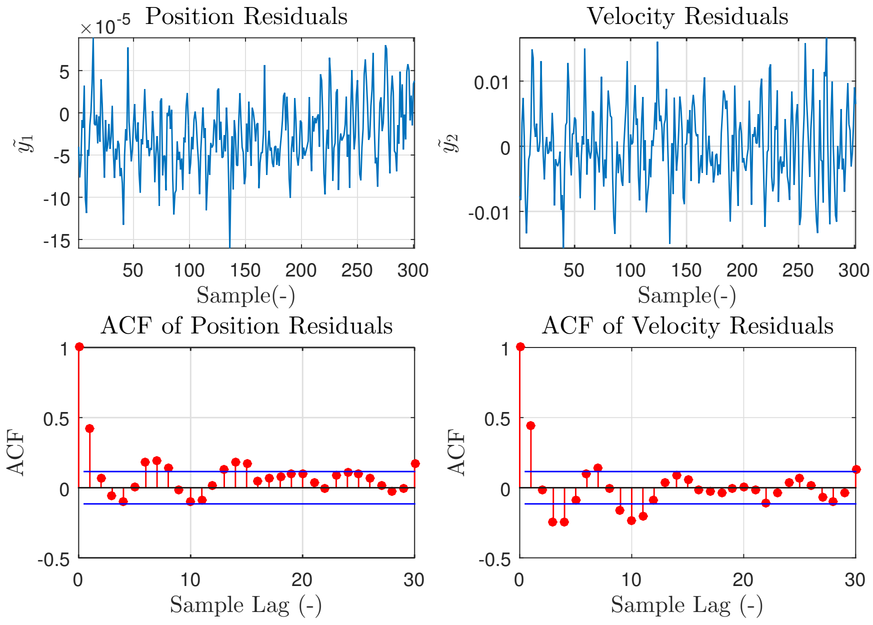


Figure 7.9: Position and velocity residuals for the joint *Hand Roll* and its autocorrelation.

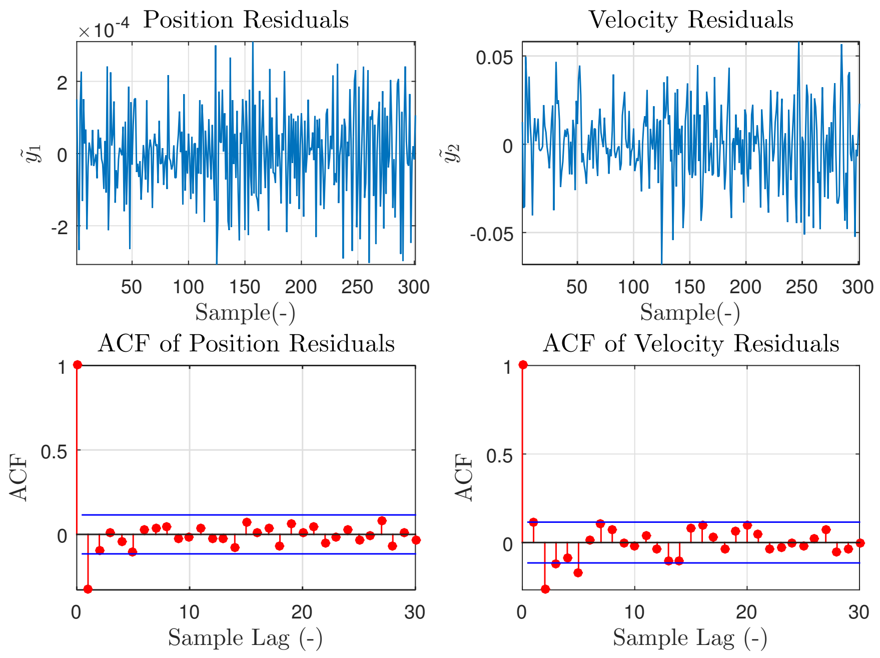


Figure 7.10: Position and velocity residuals for the joint *Hand Pitch* and its autocorrelation.

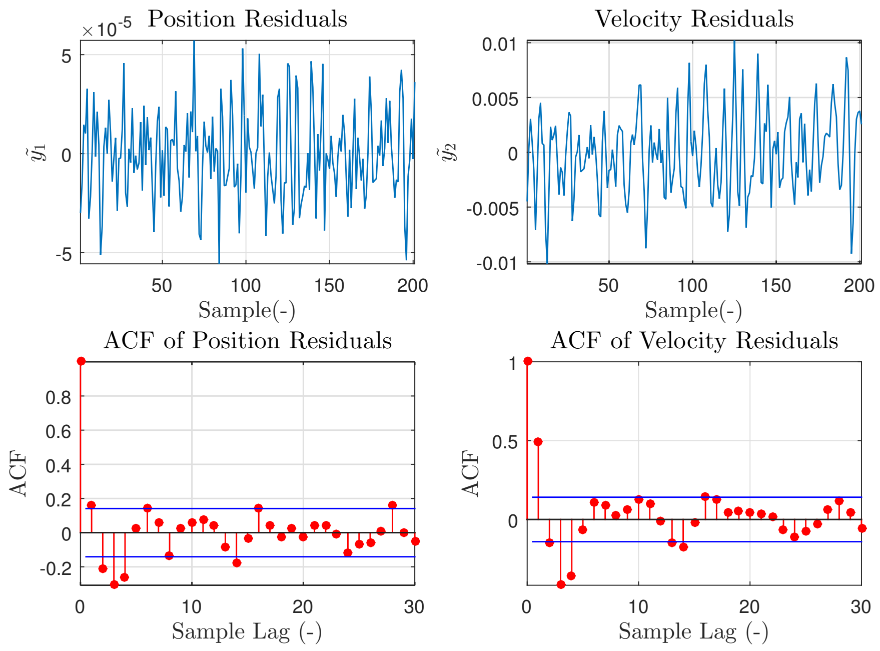


Figure 7.11: Position and velocity residuals for the joint *Instrument Slide* and its autocorrelation.

Residuals for the three joints appear to be slightly correlated. However, it is known that the model is not ideal and a small correlation can be expected. Moreover, results from *Instrument Slide* joint show more correlation than the other two joints, that is acceptable as the fit of the model found in Chapter 6 is the worst of the three joints.

7.5 Magnitude of the Force

Initially, the measurement of input torque or force τ is obtained through current measurements. The gear η and motor constant K_m of each motor taken from the motor specifications in Table 6.3 are used to compute it as:

$$\tau = \eta K_m \, i \tag{7.29}$$

where i is the motor current. However, the motor constant has not been measured and it may not be accurate. Therefore, the estimated force may be scaled. One way to check it is to attach a known mass m to the *Instrument Slide* joint while it is aligned vertically. This way, the external force applied to the joint are known. The estimated external torque is compared with the known weight as seen in Fig. (7.12).

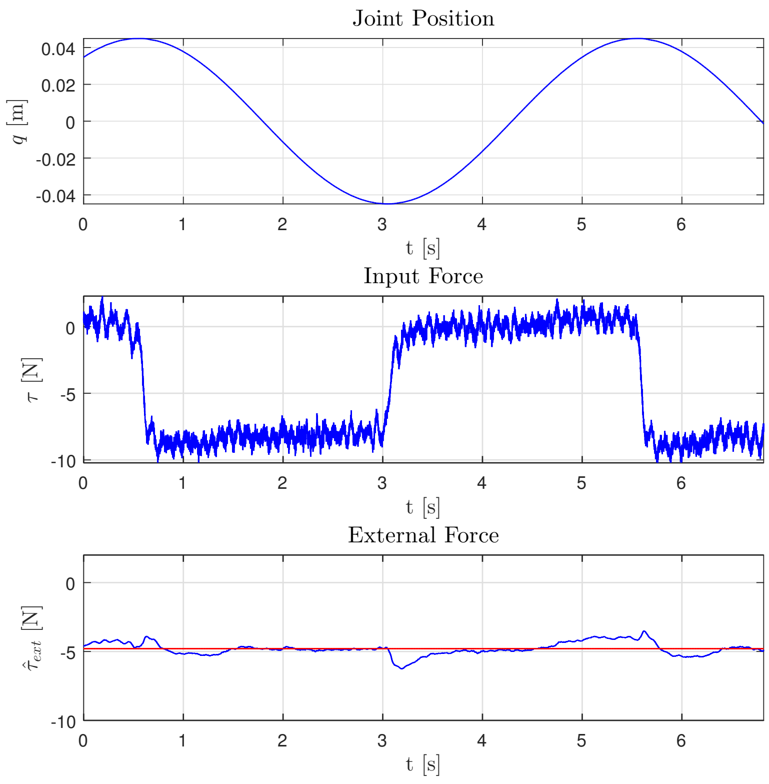


Figure 7.12: Trajectory of *Instrument Slide* joint (upper graph), where a constant weight is attached to the joint. A constant force from the EKF is estimated (lower graph). Its mean (red) is the weight estimated by the EKF.

First, a trajectory has been given to the slide movement such that a constant weight should be estimated, for a mass $m = 0.097$ kg, that is:

$$\tau_{ext} = m\,g = 0.9516 \quad [\mathrm{N}] \tag{7.30}$$

From the EKF, a weight is estimated in Fig. 7.12 with a mean of $\hat{\tau}_{ext} = 4.7945$ N. Then, the scaling factor can be found as:

$$\text{scale} = \frac{4.7945}{0.9516} = 5.0384$$

Note that the force/torque estimated in each joint is already scaled in the EKF results in Sec. 7.4.

7.6 Conclusions for the Extended Kalman Filter

This chapter verifies that external forces/torques in the joints can actually be estimated by using sensor measurements in the motor without the need of expensive force/torque sensors. The disturbance model for external forces/torques has proved to work both in simulation and the real robot.

It should be noted that independent joint models are used in the EKF. However, there may be a coupling between them as mentioned in Chapter 6. But if future work focus on improving the model, the same EKF can still be used to estimate external forces without the need of extra sensors.

Chapter 8

Haptic System

Once external forces are estimated in Chapter 7, they are used in a haptic device such that contact forces can be felt by the user. The purpose of this chapter is to present the setup used in this thesis and show the results obtained in the da Vinci robot. Moreover, stability of the human arm with the haptic device interaction is discussed.

First, the system is modeled as a switched linear system in order to describe when the system is in contact or not with the tissue. Then, stability is proven by solving an LMI problem and finding a Common Quadratic Lyapunov Function for the hybrid system. An LMI problem is also solved in [Dang et al., 2012] in order to find the critical system parameters in which the haptic system becomes unstable. However, their model is not a switched system. Another approach was used in [Gil et al., 2007] and [Hulin et al., 2008] to find a stability boundaries for haptic rendering by using the Routh-Hurwitz criterion.

Finally, experimental results of the haptic implementation for the da Vinci robot are shown in one dimension. Tissue is used to represent the interaction of the robot with the patient. Furthermore, possible future applications for the estimated joint torques/forces are discussed.

8.1 Setup Description

The haptic interface is used to link the human arm and the da Vinci robot in both directions, such that the robot follows the human's commands and the human feels the interaction forces in the robot. The setup used in the laboratory is shown in Fig. 8.1.

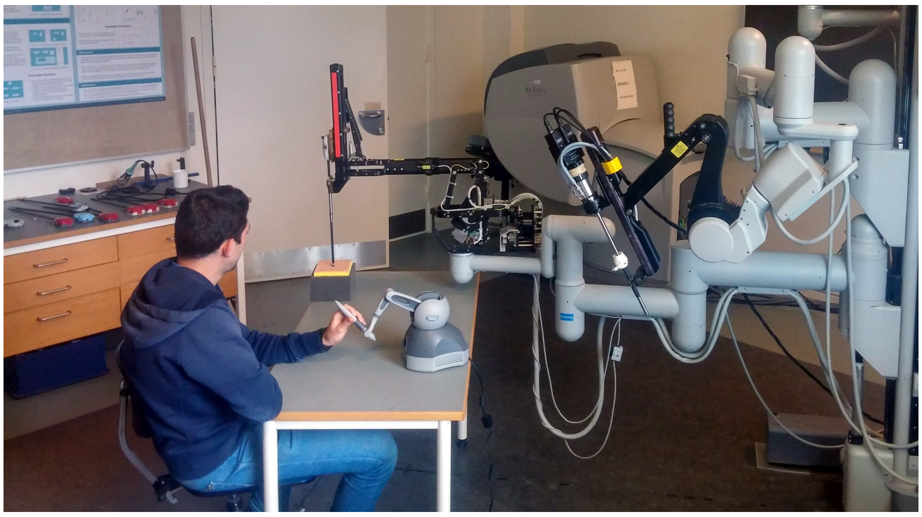


Figure 8.1: Overview of the setup in the laboratory.

The setup in Fig. 8.1 is described as follows:

- The human operator holds the haptic device. The Geomagic Touch shown in Fig. 8.2 is used in this thesis. It has 6 DOF and it is used to command position to the da Vinci robot. Note that it is not designed specifically for the da Vinci robot, since its joint movements are not directly related with da Vinci joints. Therefore, inverse kinematics are computed to relate joystick end-effector position with the robot end-effector position.
- The Geomagic Touch has three motors that are used to create 3D force feedback. Then, estimated force in the robot end-effector is used for force-feedback in the haptic device.
- Both the da Vinci Surgical Robot and the haptic device have been connected by using ROS as explained in Chapter 9.

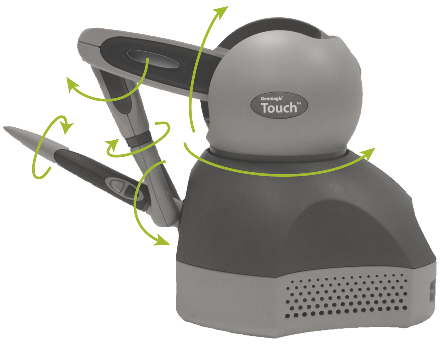


Figure 8.2: Representation of the 6DOF of the Geomagic Touch.

8.2 System Model

If the joystick is just used to command the da Vinci robot without force feedback in the haptic device, the haptic system is stable as no force is introduced that may lead to instability. However, it is possible that once the external forces are created in the haptic device, the closed loop system in Fig. 8.3 may become unstable. Therefore, the closed loop system is modeled in order to discuss stability.

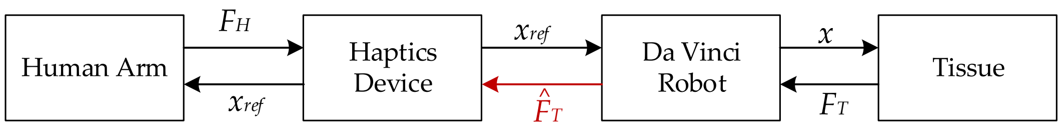


Figure 8.3: Overview of the closed loop setup used in this project. The red arrow represents how the system is closed when estimated force is sent to the Haptic Device.

A model for the one dimension case of the haptic system is developed. The different parts represented in the diagram of Fig. 8.3 are considered separately and are finally combined.

Human Arm and Haptic Device

A human arm is holding the haptic device, shown in Fig. 8.2, while applying a force F_H to it. A very common approach to model both human arm and haptic device is to consider that the connection between them is ideal, which means that both move as one rigid body. Then, they can be modeled as a mass m with a spring K_H and a damper b_H as it is done in [Dang et al., 2012] and [Hulin et al., 2008]. A model for the haptic device used has also been developed in [Cavusoglu and Feygin, 2001].

Da Vinci Robot and Controller

The human force moves the joystick. Then, the joystick position x_{ref} is sent to the robot controller and the da Vinci robot moves following the joystick commands. As the controller in the da Vinci robot runs at high rate, it can be assumed that the commanded position is the same than the da Vinci robot position $x_{ref}(t) = x(t)$. Therefore, the da Vinci Robot block in Fig. 8.3 is considered as an identity block with gain 1 when no scaling from haptic device to robot is made.

System Delay

When a filter such as the EKF in Chapter 7 is used, a delay may appear. Moreover, the network used to connect both the haptic device and the robot may also create some communication delay. However, only a simple model is desired. Therefore, delay is considered to be very small and thus, neglected.

Tissue

Finally, once the surgical robot is in contact with tissue, a reaction force F_T is estimated and $\hat{F}_T$ is sent back to the haptic device so a force can be created and the loop is closed. Due to the increasing use of surgical simulation, several models for tissue have been developed. As described in [Ayache and Delingette, 2003], the main existing models of soft tissue are:

- **Surface or Volumetric Tissue Model:** The tissue is represented by deformable surfaces or volumes. Surface models are computationally are known to be faster than volumetric ones. However, volumetric models give a more realistic simulation of cutting or suturing operations, as these operations change the geometry of the tissue model [Ayache and Delingette, 2003]. The finite element models are the most common representation of this type of models, where a finite number of basic shapes are used, such as triangles, quadrilaters or hexagons.

- **Spring and Particles Model:** This model consist of a set of points that are linked by spring and/or dampers. For instance, both in [Chen et al., 2007] and [Pezzementi et al., 2008] a mesh of springs is used as shown in Fig. 8.4, where in the second the spring constants were derived from a learning algorithm.

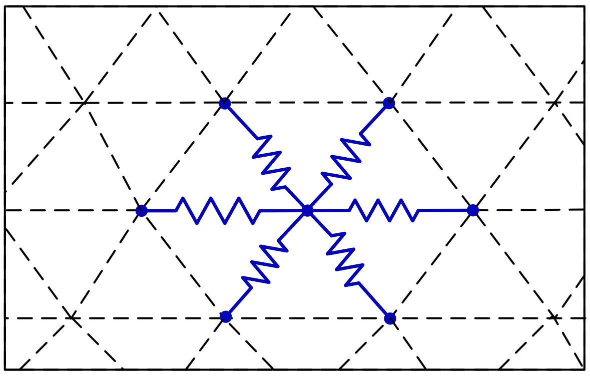


Figure 8.4: Representation of a tissue surface modeled with masses connected to its six neighbors through springs.

- **Heuristic Model:** The use of more practical models are the alternative to physical models. In [Pezzementi et al., 2008], they mention the use of an heuristic model to simulate an isotropic elastic membrane.

A detailed model is not a priority in this thesis and a simple 1D model is used. Therefore, a basic static model is chosen, where tissue is considered a spring. Then, the force of tissue F_T is given by:

$$F_T = -k_T\, x \tag{8.1}$$

where k_T is the tissue stiffness [N/m] and x is the deformation from the rest position [m]. In order to verify this model, the robot end-effector is moved with constant low velocity towards tissue as shown in Fig. 8.5.

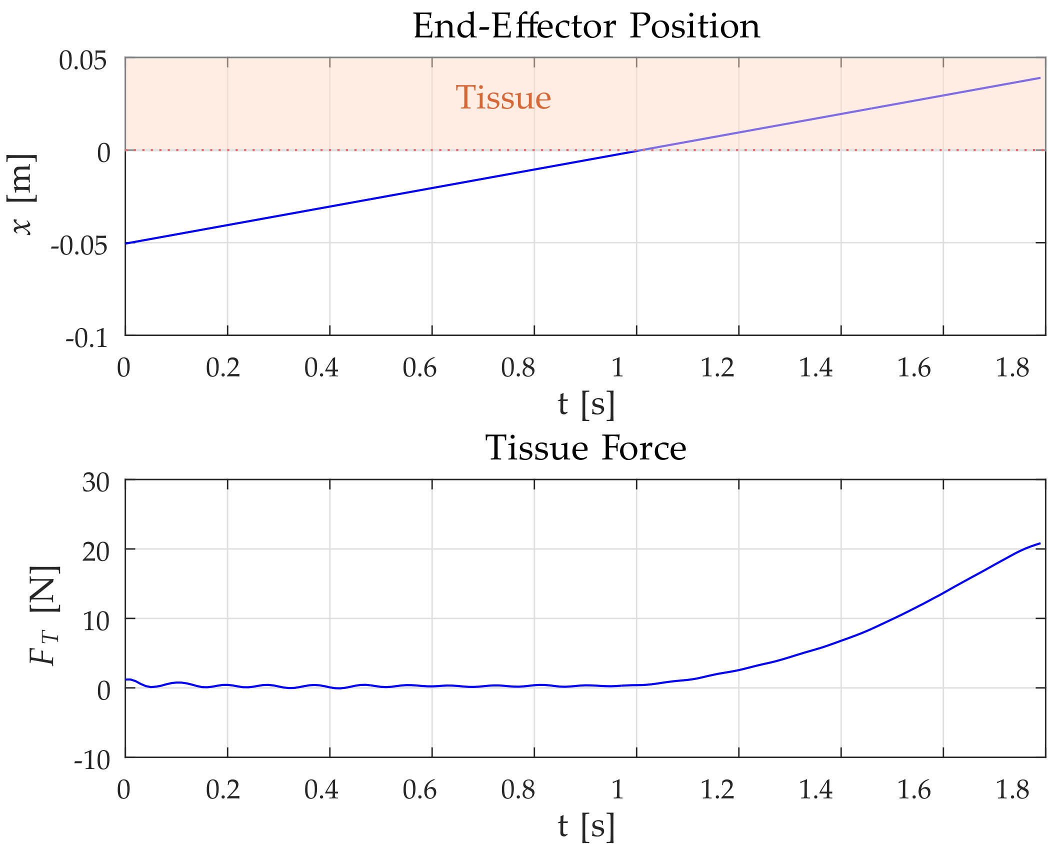


Figure 8.5: The robot end-effector is moved with constant velocity towards tissue (upper plot). Then the EKF is used to estimate contact force (lower plot).

Force is estimated with the EKF in Chapter 7 and a linear fit is done in Fig. 8.6 in order to obtain a guess for the tissue stiffness of $K_T = 457.24$ [N/m].

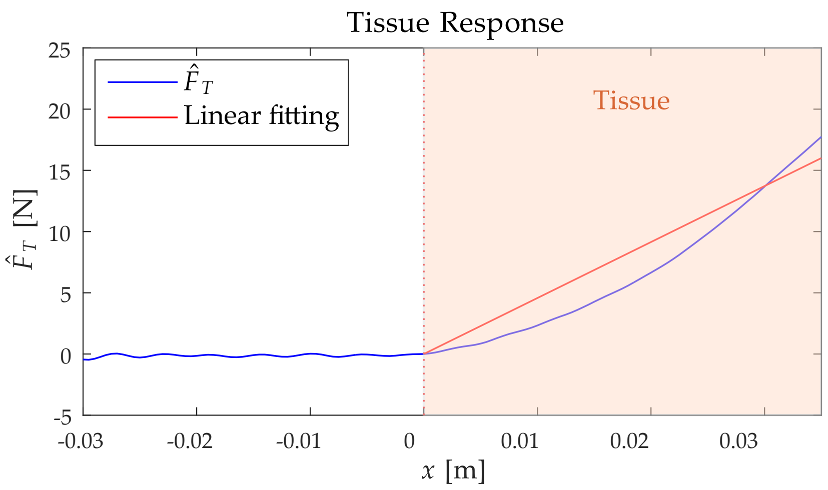


Figure 8.6: Tissue force response when end-effector is in contact (shadowed region).

From Fig. 8.6 it can be seen how the fit is not perfect since tissue has its own dynamics. It is expected as a simple static force model is used in order to discuss stability. However, to obtain a perfect tissue model is not a priority.

8.2.1 Hybrid System

Once all the elements of the closed loop system in Fig 8.3 are defined, they can be combined in order to obtain the model shown in Fig. 8.7:

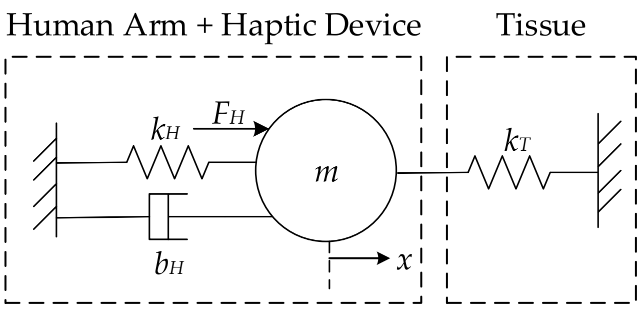


Figure 8.7: Physical model of the addressed system when robot is in contact with tissue.

The representation in Fig. 8.7 describes when **robot is in contact with tissue**. The system dynamics are expressed as:

$$m\ddot{x} = F_H - k_H\, x - b_H\, \dot{x} - k_T\, x \tag{8.2}$$

which can be rewritten as first order differential equations:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{1}{m}\left(F_H - k_H\,x_1 - k_T\,x_1 - b_H\,x_2\right) \end{aligned} \tag{8.3}$$

and then expressed in the following linear state space form:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ -\dfrac{(K_H + K_T)}{m} & -\dfrac{b_H}{m} \end{bmatrix}}_{\mathbf{A}_1} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ \dfrac{1}{m} \end{bmatrix}}_{\mathbf{B}} F_H \tag{8.4}$$

However, Eq. (8.3) only defines the system dynamics when robot is in contact with tissue, that is when $x_1 \geq 0$. In order to describe when **the robot is away from the tissue**, that is when $x_1 < 0$, the system is expressed without tissue stiffness as:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \frac{1}{m}\left(F_H - k_H\,x_1 - b_H\,x_2\right) \end{aligned} \tag{8.5}$$

which can be expressed in a linear state space form:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ -\dfrac{K_H}{m} & -\dfrac{b_H}{m} \end{bmatrix}}_{\mathbf{A}_2} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ \dfrac{1}{m} \end{bmatrix}}_{\mathbf{B}} F_H \tag{8.6}$$

Therefore, the haptic system addressed in this thesis is defined by the two subsystems of Eq. (8.4) and (8.6) with a state dependent switching signal. This is known as a switched system or hybrid system. By considering no human forces, that is $F_H = 0$, the switched system can be expressed as:

$$\dot{x} = \mathbf{A}x \begin{cases} \mathbf{A}_1\,x & \text{if} \quad x_1 \geq 0 \\ \mathbf{A}_2\,x & \text{if} \quad x_1 < 0 \end{cases} \tag{8.7}$$

8.3 Stability for Haptic Interaction

A necessary requisite for any haptic application is that stability should be preserved. In order to prove it, the switched system defined in Eq. (8.7) should be stable. Many studies discuss stability regions where the haptic system is stable, either by applying the Routh-Hurwitz criterion as in [Gil et al., 2007] [Hulin et al., 2008] or by using Lyapunov stability theory as in [Dang et al., 2012]. All consider a delay in the system. However, none of them consider a switched system where the model changes when the robot is not in contact, as it is done in this thesis.

It is possible that each linear system is stable. However, it may be that together the hybrid system is not as it is shown in Fig. 8.8. Therefore, the stability of both linear system should be studied together.

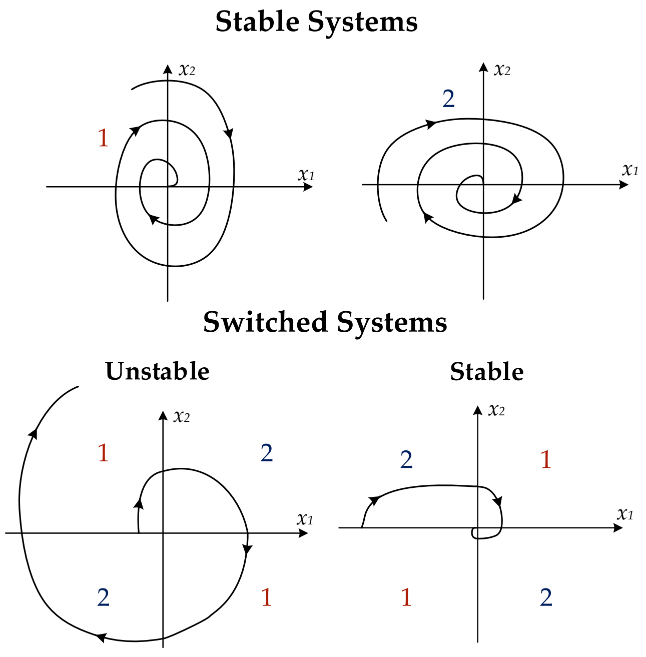


Figure 8.8: Example of two stable subsystems (top figures). When they describe a switched system, they can become unstable (bottom left) or remain stable (bottom right) under certain switching condition.

For the stability analysis, first it should be proven that the switched system is stable when there is no restriction on the switching signal. This is known as the stability analysis under arbitrary switching. It states that if there exists a Common Quadratic Lyapunov Function (CQLF) for all the subsystems, then the switched system stability is guaranteed under arbitrary switching. Moreover, it is known that the condition for the existence of a CQLF can be expressed as Linear Matrix Inequalities (LMIs) [Lin and Antsaklis, 2009]. First, a quadratic Lyapunov function is defined as:

$$V = x^T \mathbf{P}\, x \tag{8.8}$$

where $\mathbf{P} \in \mathbb{R}^{n\times n}$ is a symmetric positive definite matrix. Its derivative becomes:

$$\dot{V} = x^T(\mathbf{A}_i^T\mathbf{P} + \mathbf{P}\mathbf{A}_i)x \quad \forall\, i \in \{1,2\} \tag{8.9}$$

where $(\mathbf{A}_i^T\mathbf{P} + \mathbf{P}\mathbf{A}_i)$ is negative definite for all subsystems and $i \subseteq \mathbb{Z}$. Both Eq. (8.8) and (8.9) can now be expressed as a LMI problem. That is, a CQLF exists if the following LMI constraints are fulfilled:

$$\begin{gathered} \mathbf{P} > 0 \\ \mathbf{A}_i^T\mathbf{P} + \mathbf{P}\mathbf{A}_i < 0 \quad \forall i \in \{1,2\} \end{gathered} \tag{8.10}$$

For studying the stability of the hybrid system defined in Eq. (8.7), both mass m and damper coefficient b_H are fixed to 1 [kg] and 1 [Ns/m] respectively. Then, for every possible value of the human stiffness K_H, the tissue stiffness K_T is increased until the LMI problem can not be solved and thus, a CQLF under arbitrary switching does not exist. However, it may be possible that under certain switching conditions, the hybrid system is still stable. YALMIP [Löfberg, 2004] in MATLAB is used to solve each LMI problem.

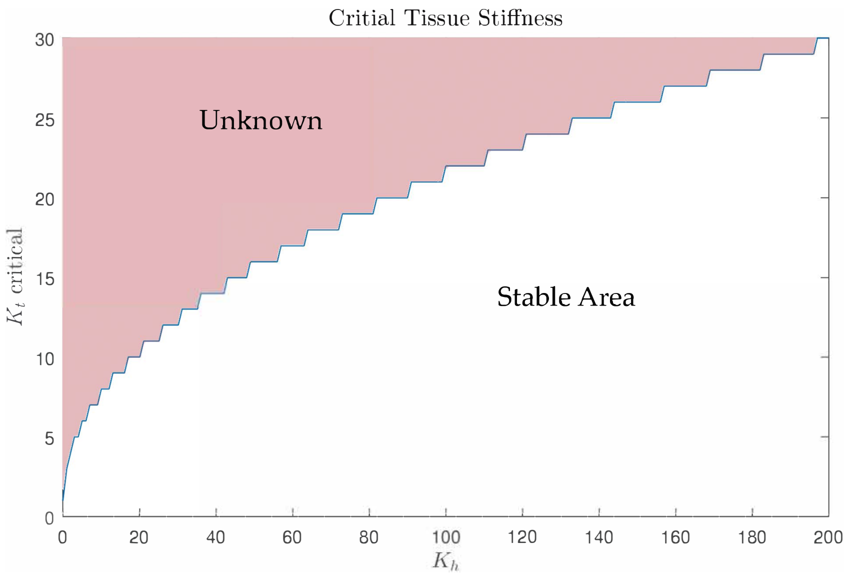


Figure 8.9: Plot shows the region of $K_T - K_H$ where a CQLF has been found that proves stability under arbitrary switching. However, A CQLF has not been found in the red region and stability is not verified.

A simulation of the hybrid system in the known stable area is shown in Fig. 8.10. Besides stability, another interesting factor that could be consider is the settling time. Not only the system has to be stable, but it should reach the equilibrium point within an acceptable time. Nonetheless, performance analysis is not in the scope of the project and no further study is done.

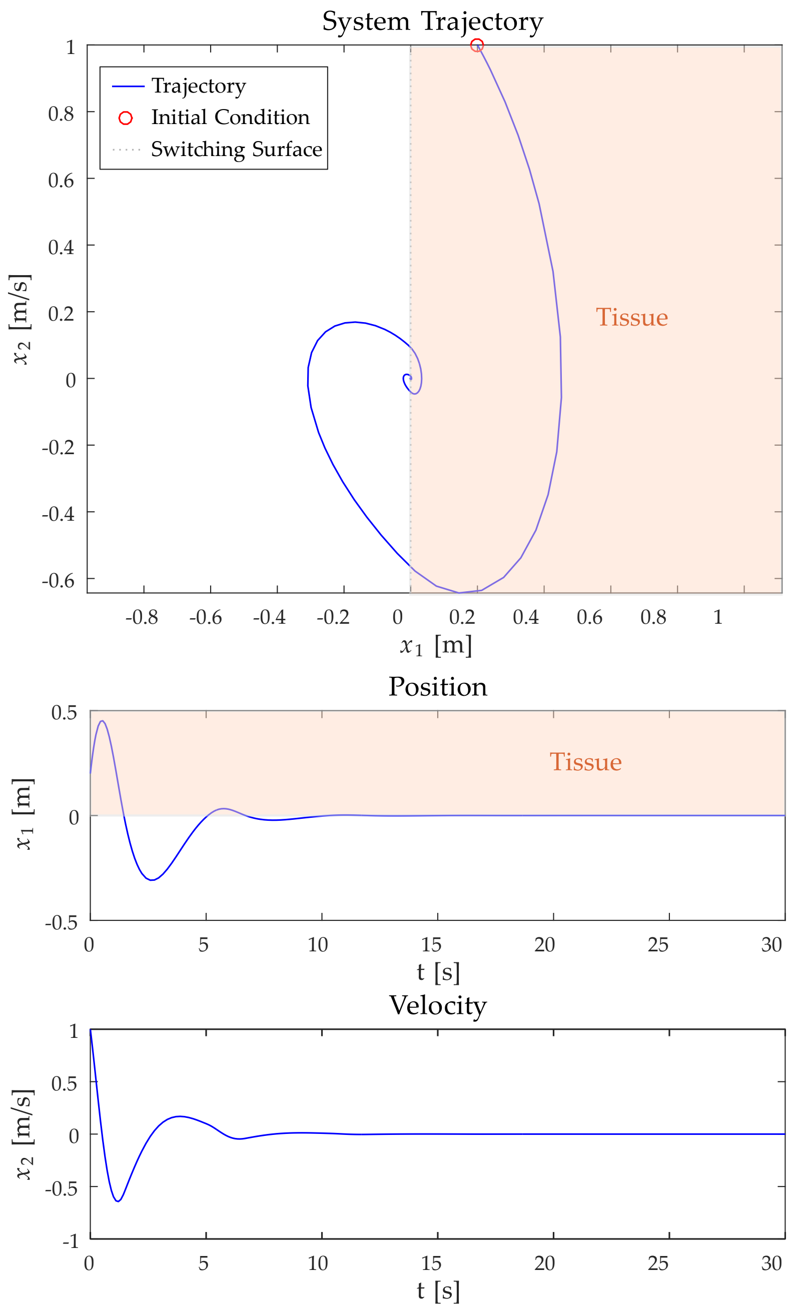


Figure 8.10: Simulation of the system where its trajectory is going towards the origin (upper plot). Position response and velocity response are showed in the middle and lower plot respectively.

8.4 Haptic Implementation in the da Vinci Robot

As mentioned before, in this thesis the Geomagic Touch is used to move the robot. A three dimensional force is estimated in the end-effector and created in the haptic device. Its joints are not directly couple with the da Vinci joints. This implies that inverse kinematics (IK) are needed to map the end-effector position from 3D cartesian space (received from the joystick) to the 6D joint space. The KDL inverse kinematics solver in ROS is used for such purpose. Results for the force feedback in one dimension are shown.

A test is carried out where tissue is placed on a flat surface as shown in Fig. 8.11, such that the da Vinci end-effector can reach it. Then, both *Hand Roll* and *Hand Pitch* are fixed while the *Instrument Slide* is following the z axis movement of the haptic device.

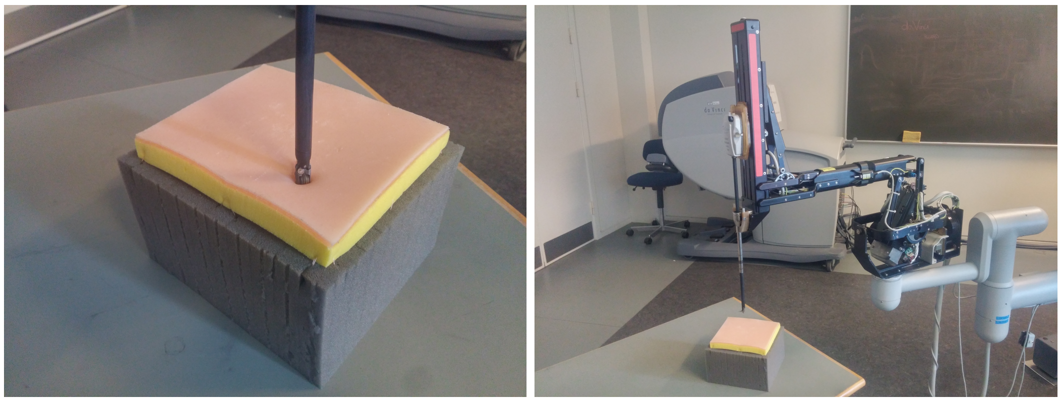


Figure 8.11: Tissue used to test the vertical movement of the da Vinci robot which is commanded by the haptic device.

Estimated external forces are only expected in the z axis when end-effector is in contact with tissue, that is when $z > 0$. Moreover, higher forces are expected when end-effector is further inside the tissue. The test intends to demonstrate the following:

- Initially, the robot end-effector is moved in and out of the tissue very fast between 0 to 8s. This is done in order to see if the EKF is able to react to fast changes in the force.
- From 8 to 15s, the joystick is moved very fast while robot is away from tissue. This is done in order to see if the EKF is still precise at higher speed. If so, the estimated force should be zero.
- From 15s to the end, the robot is moved at lower speed while it enters the

tissue. This way, it can be seen how estimated external forces are higher when end-effector is deeper inside the tissue.

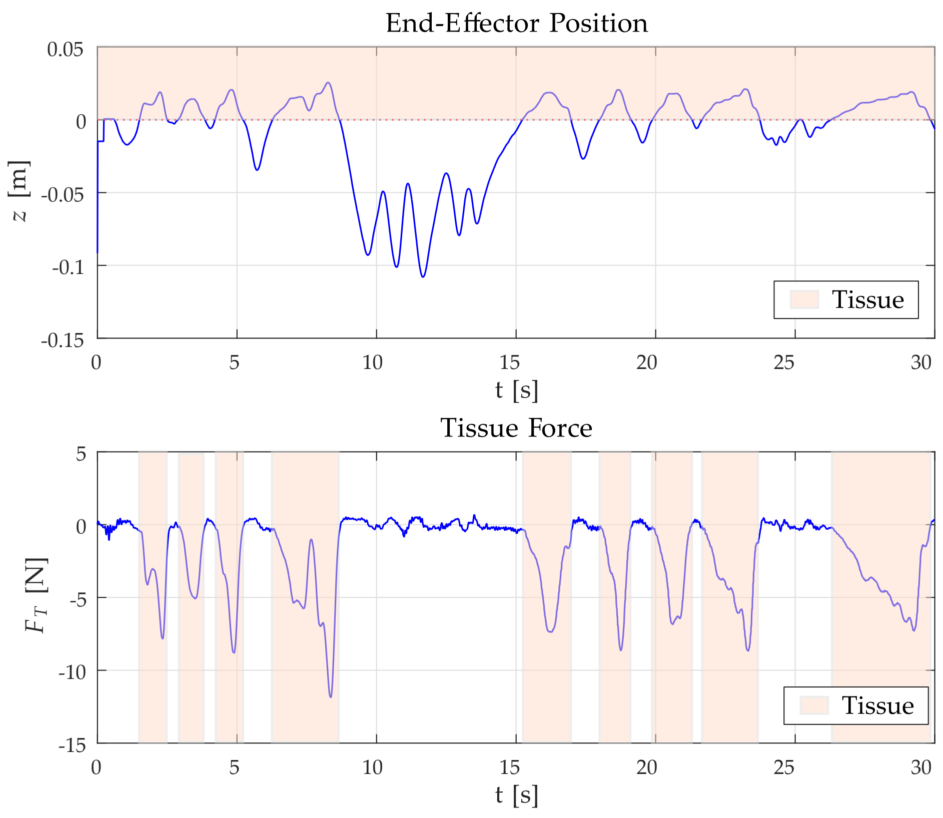


Figure 8.12: In the upper plot, the robot end-effector is moved vertically up and down while interacting with tissue (shaded area). In the lower plot, the EKF estimates force when it is in contact with the tissue (shaded area).

Results show that the EKF algorithm from Chapter 7 estimates contact forces only when robot is in contact with tissue. Likewise, it can be seen how estimated force is higher when end-effector is deeper inside the tissue as it is expected. Moreover, although the friction model was not very accurate, the estimation has hardly been affected by it. When no forces are applied, small oscillations around zero appear in the estimated force. A threshold could actually be applied to remove them.

After commenting the results, it can be stated that a force feeling for the da Vinci Surgical Robot has been implemented successfully. Even though only the one dimension case is shown, it demonstrates how tissue can be felt just by using sensor measurements from the motor.

8.5 Discussion for the Haptic System

Haptic feedback has only been shown in the one dimensional case. Results are promising as tissue contact is estimated by the EKF without the use of extra sensors. Even though it is only shown in one dimension, it is considered sufficient to establish a "proof of concept" framework.

Once it is proven that external forces can be estimated in the joints, another interesting application is to use them to create grabbing or pinching feeling. However, the haptic device has 6 DOF (7 are needed for grabbing) and it only has 3D force feedback (joint force feedback is needed). Ideally, the joystick joints should be directly coupled with the da Vinci joints as the original joystick is (then no IK solver is needed). Then, force feedback should be implemented for each joint in order to create the desired pinching or grabbing feeling. However, the original da Vinci joystick cannot be modified and an equivalent joystick which allows force feedback is needed. This is out of the scope of the project and it can be considered in future work.

Chapter 9

Implementation

The purpose of this chapter is to describe the implementation and the contribution of this project to the da Vinci Surgical Robot at AAU [Lab, 2012]. First, a short description to the actual system setup is given. Modifications have been made to the software, both in low level (microcontroller) and high level (ROS). The Extended Kalman filter runs in the microcontroller, while the haptic device is interfaced using ROS.

9.1 System Setup

As mentioned in Chapter. 2, a cascade scheme with position, velocity and current controllers is used in the da Vinci robot as shown in Fig. 9.1. While position controller is implemented in a microcontroller, velocity and current controllers are implemented in the motor driver in order to ensure sufficient controller speed.

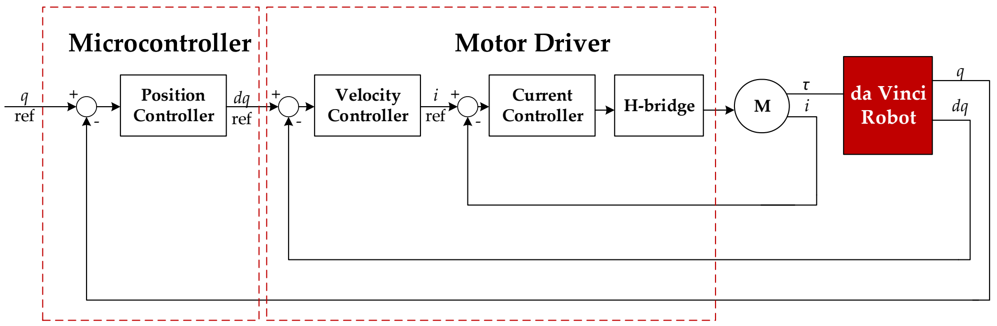


Figure 9.1: Block diagram of the cascade controller designed for the motors of the da Vinci Robot. The three controllers are PI controllers. Position controller is implemented in the microcontroller, while velocity and current controllers are implemented in motor drivers.

The actual hardware configuration of the AAU Surgical Robot is presented in the diagram of Fig. 9.2 and is described as follows:

- **Client PC:** It is used for processes that are not possible to run in real-time and require more computing power, such as path planning or user interaction. The computer runs the Robotic Operating System (ROS), an open source software framework for robots, and is connected to the embedded computer via a TCP/IP network, with a JSON based serialization protocol and a message rate of 100 Hz [Wisniewski et al., 2015]. It receives sensor measurements (position, velocity and motor current).

- **Embedded Computers:** There are two single board Reconfigurable Input/ Output (sbRIO) from National Instruments which are responsible for getting position, velocity and current measurements from encoders and potentiometers in the MAXON motors. This is done through a **Field Programmable Gate Array** (FPGA) based hardware, which is configured to interface the sensors. Then, sensor measurements are sent both to the client PC and motor drivers. The reason why two sbRIO are used is the lack of input/output ports in one. Position controllers are implemented here and the control reference is received from the client PC to drive the motors. Moreover, they also ensure security constraints, such as stopping the motors when joint limits are reached or limiting the control values to protect the motor [Wisniewski et al., 2015].

- **LabView:** It is used to program both the processor and FPGA using the same toolchain. It also provides a graphical interface in order to manage the data, such as control signals or sensors measurements. Moreover, it allows to record and save this data.

- **ESCON Motor Driver:** One driver per motor is used and tuned for speed control with inner current loop, which runs in real-time. Therefore, both controllers run at very high speed.

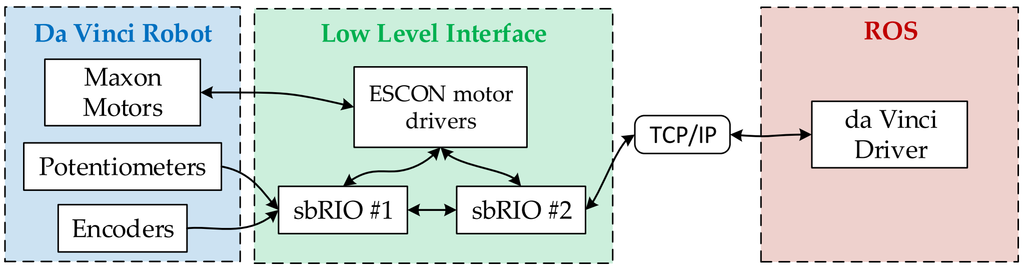


Figure 9.2: Overview of the custom configuration made for the da Vinci Robot located at AAU.

In this thesis, the original system setup has been extended in order to add the haptic device and implement the force feedback to the surgeon. The modifications made are shown in the diagram of Fig. 9.2 and are explained as follows:

- An **Extended Kalman Filter** is developed in Chapter 7. It is preferred that estimation of contact force runs at higher speed than the haptic device, therefore the EKF is implemented in the microcontroller sbRIO. An m code for the EKF is given to LabView, where it is converted to microcontroller code.
- A **ROS Driver** is used to connect the haptic device Geomagic Touch with ROS, which is connected to the PC via TCP/IP network. This driver is available online as explained in Sec. 9.2.1.
- The **Haptic Interface** is created in ROS in order communicate the haptic system described in Chapter 8 with the da Vinci Robot. First, reference position from the joystick is sent to robot through the TCP/IP communication. Then, estimated contact force is received in ROS. Finally, it is sent to the haptic device in order to create the force feedback to the user.

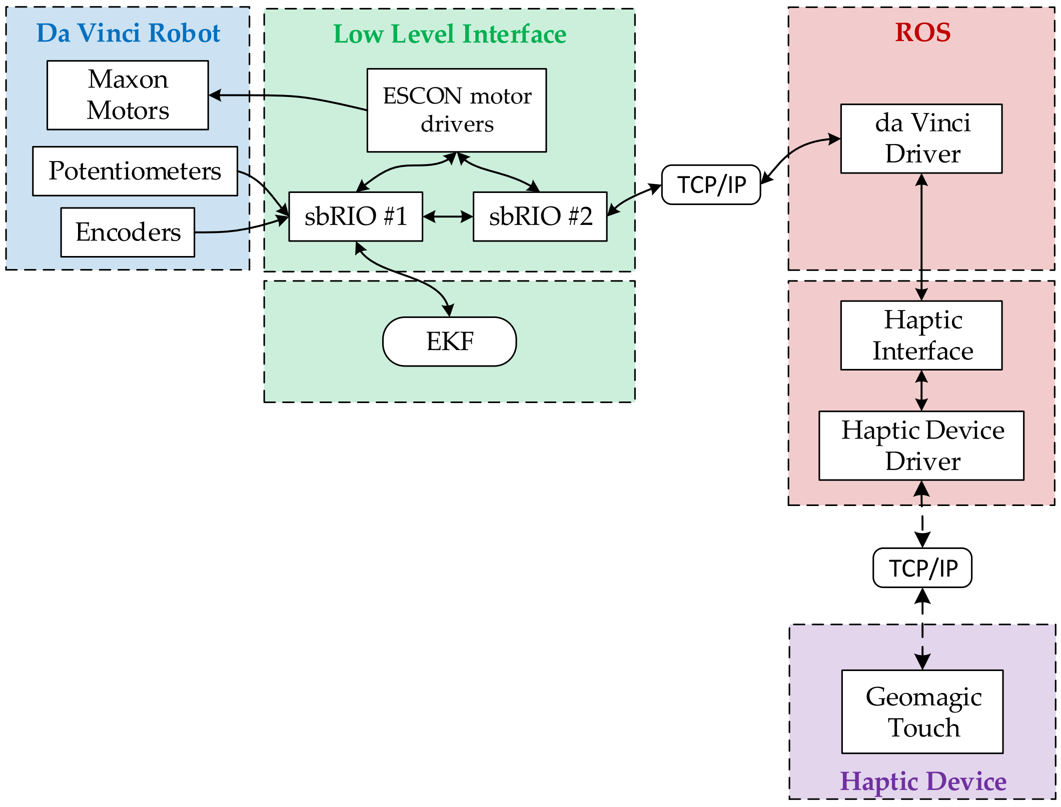


Figure 9.3: Overview of the configuration of the da Vinci Robot located at AAU with the modifications made in this thesis.

9.2 ROS Environment

ROS is an open-source operating system for robots. It provides the same functionalities than any standard operating system, such as hardware abstraction, low-level device control, message-passing between processes, and package management. ROS is represented in a graph architecture as shown in Fig. 9.4, where processes that can be distributed across machines (also known as nodes) are loosely coupled using the ROS communication infrastructure. The ROS environment is currently only developed for Ubuntu [ROS, 2016].

Before elaborating further on the ROS structure of this thesis, it is important to have an overview of the general terms used in the ROS environment. Basically, the main concepts that should be known are:

- **Node:** is a process that performs some computation.
- **Topic:** is the communication between two or more ROS node.
- **Message:** is an information that is sent to a topic.
- **Publish:** Nodes can publish to topics in order to send messages.
- **Subscribe:** Nodes can subscribe to topics in order to read messages.
- **Service:** is a communication used to interact between nodes. It consist of a request message that awaits for a reply message.
- **Package:** is simply an organized set of ROS elements. It might contain ROS nodes, libraries, datasets or configuration files.

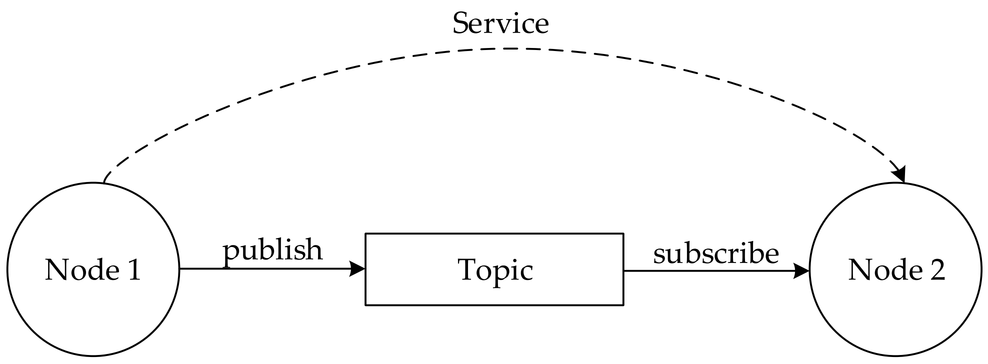


Figure 9.4: Graph representation of a ROS network. A service sends a request from Node 1 which awaits for a reply from Node 2. Moreover, Node 1 publishes a message to a topic and Node 2 is subscribed to it in order to read it.

9.2.1 ROS Structure of this Thesis

The ROS environment created in this thesis to interface with the da Vinci Surgical Robot contains several packages that are grouped in a workspace. The packages are described as follows:

- **davinci_description**: The URDF model of the da Vinci robot, where the Unified Robot Description Format (URDF) is an XML format for representing a robot model. It basically describes the kinematic chain of the robot.
- **davinci_driver**: The interface between the physical Davinci Robot and ROS. It requires the `davinci_despription` package. It subscribes to joint references, which are sent to the robot, and it publish sensor measurements and estimated force, which are received from the robot and microcontroller.
- **omni_description**: URDF model for the Geomagic Touch, previously known as the Sensable Phantom Omni.
- **phantom_omni**: The interface between the Geomagic Touch and ROS. It publishes joint states, which are received from the haptic device, and subscribes to 3 dimensional force data in order to sent it to the haptic device. This package has been modified in order to publish the position in Cartesian space of the joystick end-effector.
- **haptic_interface**: The node created in this thesis to communicate the Geomagic Touch with the da Vinci robot. Its structure is explained in Sec. 9.3. First, it subscribes to joystick end-effector position and estimated force in the robot. Then, it publish estimated force to the haptic device and commands joints position to the robot (after computing IK).

All the the code related to the da Vinci robot is located at the github site of the Robotic Surgery Group from Aalborg University:
`https://github.com/AalborgUniversity-RoboticSurgeryGroup`

The code for the Geomagic Touch is located at the github site of Dane Powell:
`https://github.com/danepowell`

However, the whole ROS workspace used can be found in Appendix C. An overview of the environment created in ROS is given in Fig. 9.5.

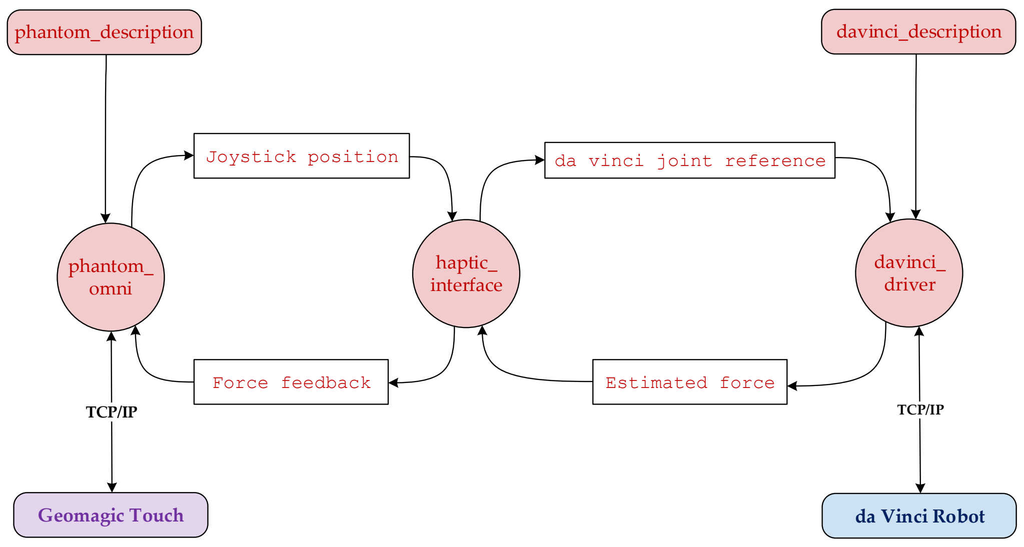


Figure 9.5: Overview of the interface created in ROS (red) in order to communicate the da Vinci robot (blue) with the Haptic Device (purple).

9.3 User Interface

A user interface is created in order to use the ROS node created in this thesis. First, the ROS structure should be properly set up as explained in App B. After running the haptic node in ROS, a graphical user interface appears with multiple modes as shown below:

```
*****************************************************************
The following modes are avaiable:
-----------------------------------------------------------------
press ’a’ to move daVinci with Haptics Device (force feedback)
press ’b’ to move daVinci with Haptics Device (no force feedback)
press ’c’ to give joint setpoints to da Vinci (FK)
press ’d’ to give 3d force setpoint to joystick
press ’e’ to move Slide joint with Haptic Device
*****************************************************************
```

- The options *a* and *b* consist of controlling the da Vinci robot with the joystick, with and without force feedback. Position in the haptic device is directly related to position of the robot end-effector. Then, inverse kinematics are computed for the da Vinci Surgical Robot in order to find corresponding joints position q to the end-effector position. The algorithm to run this mode is shown in Fig. 9.6.

- Modes c and d are used to give position and force references to the da Vinci robot and haptic device respectively.
- Mode e is similar to a and b but the haptic device is only moving the *Instrument Slide* joint of the da Vinci robot. This way, inverse kinematics are not needed, as the joystick z axis is directly the reference for the slide movement. This is done in order to avoid computing inverse kinematics.

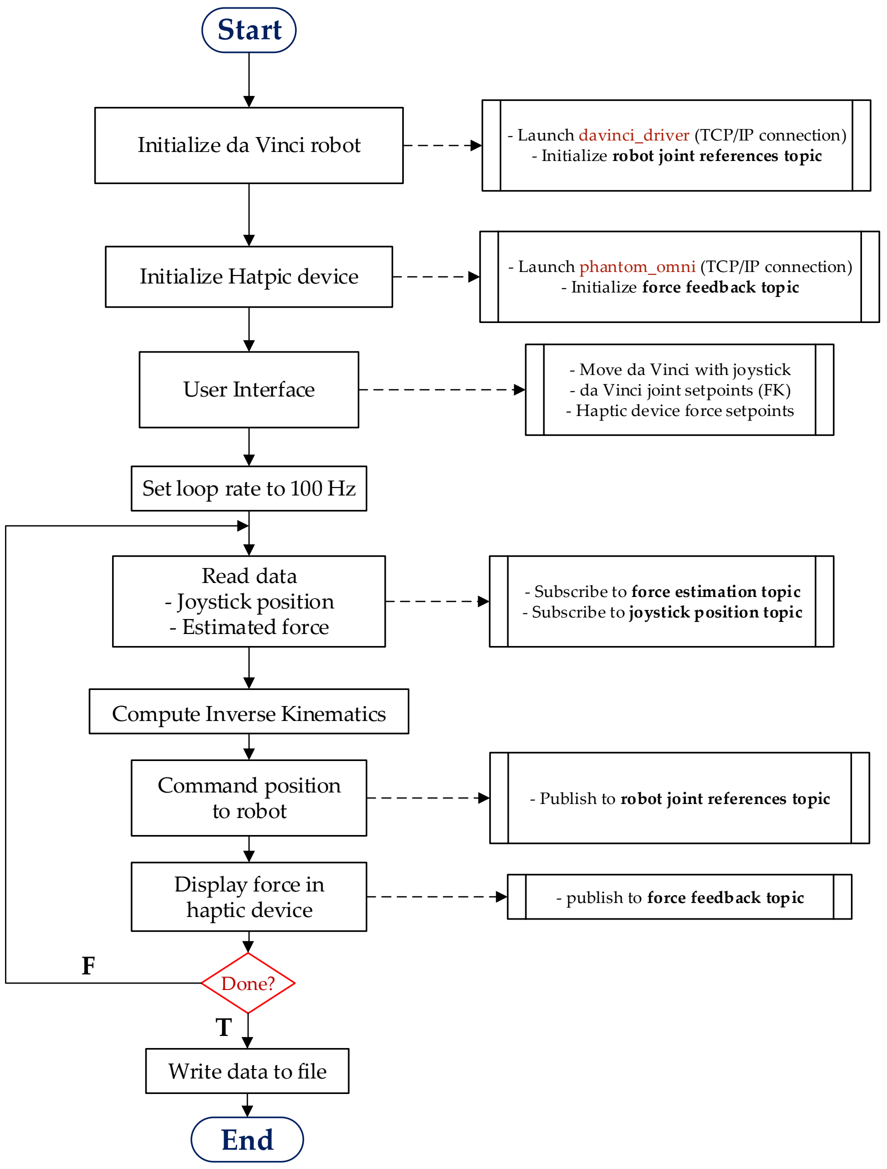

Start
Initialize da Vinci robot
- Launch davinci_driver (TCP/IP connection)
- Initialize robot joint references topic
Initialize Hatpic device
- Launch phantom_omni (TCP/IP connection)
- Initialize force feedback topic
User Interface
- Move da Vinci with joystick
- da Vinci joint setpoints (FK)
- Haptic device force setpoints
Set loop rate to 100 Hz
Read data
- Joystick position
- Estimated force
- Subscribe to force estimation topic
- Subscribe to joystick position topic
Compute Inverse Kinematics
Command position to robot
- Publish to robot joint references topic
Display force in haptic device
- publish to force feedback topic
F
Done?
T
Write data to file
End

Figure 9.6: Algorithm of the ROS node. The position of the haptic device is sent to the robot while the estimated force in the robot is sent to the haptic device.

Chapter 10

Conclusion and Discussion

This chapter finalize the report by summing up the work and findings of this project. It also concludes the results obtained throughout this thesis. Lastly, possible improvements are discussed and recommendations for future work are given.

Conclusion

The aim of this project is to develop a haptic interface for the da Vinci Surgical Robot in order to enhance the capabilities of the surgeon, as analysed in Chapter 1. Difficulties arise when sensors are used to obtain haptic information, as they are limited due to strict regulations in operating rooms. Therefore, sensorless force estimation becomes a real alternative to the use of sensors.

Estimating contact forces can not be done without a detailed model of the system. Therefore, a kinematic and dynamic model of the da Vinci Surgical Robot is developed in Chapter 3 and 4. Then, a simulation is built for the nonlinear multivariable system in Simulink and a controller is designed for it in Chapter 5. Simulation has been used as a test bench5.

The problem of obtaining the model parameters has been simplified by assuming independent joint models. In Chapter 6 it is demonstrated how parameters can be estimated if a proper exciting trajectory for the system is found. Two different estimation methods (constrained and unconstrained) are compared with similar accuracy. Moreover, results show that friction (mostly Coulomb) appears to behave slightly different than the theoretical model. However, the models found for three joints are considered to be valid for estimating contact forces.

An extended Kalman filter is then presented in Chapter 7, where external torques and forces are modeled as disturbances in order to estimate them. Results for three joints show that forces at the joints can actually be estimated by using available position and current measurement from the motor.

Lastly, estimated forces at the joints are used to compute forces applied at the robot end-effector. Then, they are used for force feedback in a haptic device in Chapter 8. Results are shown in one dimension as it is considered sufficient to prove the concept. Forces are estimated when the robot is in contact with tissue as it is desired. All the haptic interaction has been implemented in the ROS environment as explained in Chapter 9.

Based on the results obtained in this project, it is concluded that force feeling can be achieved by estimating contact forces only from motor sensor measurements. Thus, the prime focus of this thesis is achieved. A haptic device connected to the da Vinci Surgical Robot at AAU is used to test it. Moreover, the preliminary goals set at the beginning of the thesis are also fulfilled. Despite the project is considered successful, there are still opportunities for further enhancements.

Improvements and Future Work

The developed force estimation proved to capture contact forces in the da Vinci Surgical robot. However, there is still room for improvement:

- Even though a nonlinear multivariable model of the da Vinci robot is found in Chapter 4, a nonlinear independent model for each joint is used. Therefore, future work can focus on estimating parameters for the multivariable model.
- Only three joints are used to estimate external forces although the da Vinci robot has a total of 6 or 7 joints (depending on the configuration). Therefore, estimate the remaining joints is considered the next step to the actual work.
- The friction model used (mostly Coulomb) does not fit exactly the theoretical model. Therefore, it may be interesting to research further in this matter.
- A joystick with 3D force feedback is used to test the estimated contact forces in the robot. However, it does not include force feedback in all the joints. This implies that only contact forces at the end-effector can be tested. For instance, grabbing or pinching feeling can not be implemented with this device. Therefore, it may be interesting to use a haptic device with force feedback for all joint for such purpose.

This project is hereby finished.

Bibliography

[Lab, 2012] (2012). Surgical Robotics Lab AAU. `http://www.es.aau.dk/sections-labs/Automation-and-Control/Laboratory+facilities/Surgical+Robotics+Lab/`. Accessed: 2015.

[ROS, 2016] (2016). ROS Documentation. `http://http://wiki.ros.org/`. Accessed: 2016.

[3DSystems, 2016a] 3DSystems (2016a). Geomagic Touch Device Drivers: Help Guide.

[3DSystems, 2016b] 3DSystems (2016b). Geomagic Touch Device Guide. `http://dl.geomagic.com/binaries/support/downloads/Sensable/3DS/Geomagic-Touch_Device_Guide.pdf`. Accessed: 2016.

[Ayache and Delingette, 2003] Ayache, N. and Delingette, H. (2003). *Surgery Simulation and Soft Tissue Modeling: International Symposium, IS4TM 2003. Juan-Les-Pins, France, June 12-13, 2003, Proceedings*. Lecture Notes in Computer Science. Springer Berlin Heidelberg.

[Birge, 2006] Birge, B. (2006). Particle swarm optimization toolbox in matlab. `http://www.mathworks.com/matlabcentral/fileexchange/7506-particle-swarm-optimization-toolbox`.

[Boyd and Vandenberghe, 2004] Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Berichte über verteilte messysteme. Cambridge University Press.

[Cavusoglu and Feygin, 2001] Cavusoglu, M. C. and Feygin, D. (2001). Kinematics and dynamics of phantom model 1.5, haptic interface. Technical Report UCB/ERL M01/15, EECS Department, University of California, Berkeley.

[Chen et al., 2007] Chen, F., Gu, L., Huang, P., Zhang, J., and Xu, J. (2007). Soft tissue modeling using nonlinear mass spring and simplified medial representation. In *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 5083–5086.

[Corke, 2011] Corke, P. I. (2011). *Robotics, Vision & Control: Fundamental Algorithms in*. Springer.

[Craig, 2009] Craig, J. J. (2009). *Introduction to Robotics: Mechanics and Control*. Pearson, 3rd edition.

[Dang et al., 2012] Dang, Q. V., Vermeiren, L., Dequidt, A., and Dambrine, M. (2012). Analyzing stability of haptic interface using linear matrix inequality approach. In *Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on*, pages 1129–1134.

[Denavit and Hartenberg, 1955] Denavit, J. and Hartenberg, R. S. (1955). *A kinematic notation for lower-pair mechanisms based on matrices*. Journal of Applied Mechanics.

[Dupont, 1990] Dupont, P. E. (1990). Friction modeling in dynamic robot simulation. In *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pages 1370–1376 vol.2.

[Enayati et al., 2016] Enayati, N., Momi, E. D., and Ferrigno, G. (2016). Haptics in robot-assisted surgery: Challenges and benefits. *IEEE Reviews in Biomedical Engineering*, PP(99):1–1.

[Eom et al., 1998] Eom, K. S., Suh, I. H., Chung, W. K., and Oh, S. R. (1998). Disturbance observer based force control of robot manipulator without force sensor. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 4, pages 3012–3017 vol.4.

[Gil et al., 2007] Gil, J. J., Sanchez, E., Hulin, T., Preusche, C., and Hirzinger, G. (2007). Stability boundary for haptic rendering: Influence of damping and delay. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 124–129.

[Grewal and Andrews, 2008] Grewal, M. and Andrews, A. (2008). *Kalman Filtering: Theory and Practice Using MATLAB*. Wiley.

[Hof and Schrama, 1994] Hof, P. M. V. D. and Schrama, R. J. P. (1994). Identification and control — Closed-loop issues.

[Hulin et al., 2006] Hulin, T., Preusche, C., and Hirzinger, G. (2006). Stability boundary for haptic rendering: Influence of physical damping. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1570–1575.

[Hulin et al., 2008] Hulin, T., Preusche, C., and Hirzinger, G. (2008). Stability boundary for haptic rendering: Influence of human operator. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3483–3488.

[Jahandideh and Namvar, 2012a] Jahandideh, H. and Namvar, M. (2012a). Use of pso in parameter estimation of robot dynamics; part one: No need for parameterization. pages 1–6.

[Jahandideh and Namvar, 2012b] Jahandideh, H. and Namvar, M. (2012b). Use of pso in parameter estimation of robot dynamics; part two: Robustness. pages 1–6.

[Jakobsen and Lykkegaard, 2015] Jakobsen, B. L. and Lykkegaard, C. K. (2015). *Safety in Automated Surgery with the da Vinci Robot*. AAU.

[Janot et al., 2014] Janot, A., Vandanjon, P. O., and Gautier, M. (2014). A generic instrumental variable approach for industrial robot identification. *IEEE Transactions on Control Systems Technology*, 22(1):132–145.

[Jung et al., 2006] Jung, J., Lee, J., and Huh, K. (2006). Robust contact force estimation for robot manipulators in three-dimensional space. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 220(9):1317–1327.

[Kennedy and Eberhart, 1995] Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. 4:1942–1948 vol.4.

[Khalil, 2002] Khalil, H. (2002). *Nonlinear Systems*. Pearson Education. Prentice Hall.

[Khosla, 1987] Khosla, P. K. (1987). *An algorithm to estimate manipulator dynamics parameters*. Carnegie-Mellon University, Robotics Institute.

[Lee and Ahn, 2010] Lee, S. C. and Ahn, H. S. (2010). Sensorless torque estimation using adaptive kalman filter and disturbance estimator. In *Mechatronics and Embedded Systems and Applications (MESA), 2010 IEEE/ASME International Conference on*, pages 87–92.

[Lin and Antsaklis, 2009] Lin, H. and Antsaklis, P. J. (2009). Stability and stabilizability of switched linear systems: A survey of recent results. *IEEE Transactions on Automatic Control*, 54(2):308–322.

[Linderoth et al., 2013] Linderoth, M., Stolt, A., Robertsson, A., and Johansson, R. (2013). Robotic force estimation using motor torques and modeling of low velocity friction disturbances. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 3550–3556.

[Löfberg, 2004] Löfberg, J. (2004). YALMIP : A toolbox for modeling and optimization in MATLAB. In *CCA/ISIC/CACSD*.

[MapleSoft, 2014] MapleSoft (2014). *MAPLE User's Manual*. MapleSoft.

[Murray et al., 1994] Murray, R. M., Li, Z., and Sastry, S. S. (1994). *A Mathematical Introduction to Robotic Manipulation*. CRC Press.

[Pezzementi et al., 2008] Pezzementi, Z., Ursu, D., Misra, S., and Okamura, A. M. (2008). Modeling realistic tool-tissue interactions with haptic feedback: A learning-based method. In *2008 Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 209–215.

[Rouleau, 2014] Rouleau, G. (2014). How to Model a Hard Stop in Simulink. `http://blogs.mathworks.com/simulink/2014/01/22/how-to-model-a-hard-stop-in-simulink/`.

[Serway, 1986] Serway, R. (1986). *Physics for scientists & engineers*. Number v. 2 in Physics for Scientists & Engineers. Saunders College Pub.

[Shi and Eberhart, 1998] Shi, Y. and Eberhart, R. (1998). A modified particle swarm optimizer. pages 69–73.

[Sloth and Wisniewski, 2015] Sloth, C. and Wisniewski, R. (2015). *Recent Advances in Mechanism Design for Robotics: Proceedings of the 3rd IFToMM Symposium on Mechanism Design for Robotics*, chapter Towards Safe Robotic Surgical Systems, pages 165–175. Springer International Publishing, Cham.

[Spong et al., 2004] Spong, M. W., Hutchinson, S., and Vidyasagar, M. (2004). *Robot Dynamics and Control*. 2nd edition.

[Stolt, 2015] Stolt, A. (2015). *On Robotic Assembly using Contact Force Control and Estimation*. Department of Automatic Control, Lund University.

[Surgical, 2016] Surgical, I. (2016). da Vinci Products FAQ. `http://phx.corporate-ir.net/phoenix.zhtml?c=122359&p=irol-faq`. Accessed: 2016.

[Swevers et al., 1997] Swevers, J., Ganseman, C., Tukel, D., De Schutter, J., and Van Brussel, H. (1997). Optimal robot excitation and identification. *Robotics and Automation, IEEE Transactions on*, 13(5):730–740.

[Swevers et al., 2007] Swevers, J., Verdonck, W., and De Schutter, J. (2007). Dynamic model identification for industrial robots. *Control Systems, IEEE*, 27(5):58–71.

[Trejos et al., 2010] Trejos, A. L., Patel, R. V., and Naish, M. D. (2010). Force sensing and its application in minimally invasive surgery and therapy: a survey. 224:1435–1454.

[Wahrburg et al., 2014] Wahrburg, A., Zeiss, S., Matthias, B., and Ding, H. (2014). Contact force estimation for robotic assembly using motor torques. In *Automation Science and Engineering (CASE), 2014 IEEE International Conference on*, pages 1252–1257.

[Walter, 1999] Walter, L. (1999). Hooke's law, simple harmonic oscillator. mit course 8.01: Classical mechanics, lecture 10.

[Wisniewski, 2015a] Wisniewski, R. (2015a). *Mechanical Systems II: Introduction to Calculus of Variations*. Aalborg University.

[Wisniewski, 2015b] Wisniewski, R. (2015b). *Mechanical Systems II: Lagrange Mechanics*. Aalborg University.

[Wisniewski et al., 2015] Wisniewski, R., Sloth, C., Jensen, S., and Hansen., K. D. (2015). Instrumentation of the da vinci robotic surgical system.

[Xia et al., 2012] Xia, S., Bai, X., Guo, Z., and Xu, Y. (2012). Improved particle swarm optimization for non-convex optimal power flow. In *Power and Energy Engineering Conference (APPEEC), 2012 Asia-Pacific*, pages 1–5.

Appendix A

Actuator Dynamics

The manipulator dynamics in Chapter 4 did not include the dynamics of the motors that generate the torques to move the joints. In this appendix, the motor dynamics are described and related to the manipulator dynamics in order to see how relevant they are to the model.

Joints DC Motors Dynamics

This section provides a detailed model of the DC motors at the joints of the da Vinci robot. Notice that this section is inspired by [Spong et al., 2004, Ch. 10]. In Chapter 4, the manipulator dynamics were expressed in a nonlinear equation of the following form:

$$\tau = \mathbf{M}(q)\ddot{q} + V(q,\dot{q}) + G(q) + F(\dot{q}) + \tau_{ext} \tag{A.1}$$

where the generalized forces or torques τ are generated by DC motors. Therefore, it is important to understand how this torque is generated. A DC motor consists of a fixed stator with permanent magnets and a movable rotor (also called armature), which rotates due to a generated magnetic flux. This rotation produces a torque proportional to the current. At the same time, when a motor is rotating, a voltage opposed to the current flow is created.

Electrical Equation

The electrical circuit of a DC motor is shown in Fig.A.1:

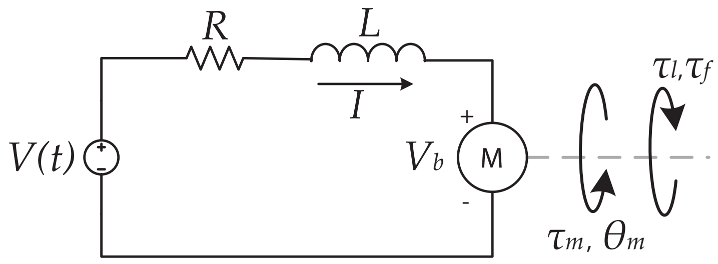


Figure A.1: Representation of the electrical circuit of a DC motor with the torque τ_m generated when current is flowing. Load torque τ_l and friction torques τ_f are also present in a motor and opposed to the motor torque.

where:

V(t)	is the armature voltage [V]
L	is the armature impedance [H]
R	is the armature resistance [Ω]
V_b	is the back induced voltage opposed to the current flow [V]
i	is the armature current [A]
θ_m	is the rotor angular position [rad]
τ_m	is the generated motor torque [Nm]
τ_l	is the load torque of the manipulator opposed to the motor torque [Nm]
τ_f	is the friction torque in the motor [Nm]

Once the circuit in Fig.A.1 is defined, its electrical equation is given by:

$$V(t) - V_b(t) = L\,\frac{di(t)}{dt} + R\,i(t) \tag{A.2}$$

where the back induced voltage is proportional to the angular velocity of the motor:

$$V_b(t) = K_b\,\omega_m(t) = K_b\,\dot{\theta}_m(t) \tag{A.3}$$

with K_b the back induced voltage constant [Vs/rad], while the motor torque generated is proportional to the current:

$$\tau_m(t) = K_m\,i(t) \tag{A.4}$$

with K_m the torque constant [Nm/A]. Finally, the electrical equations can be combined into one:

$$V(t) - K_b\,\dot{\theta}_m(t) = L\,\frac{di(t)}{dt} + R\,i(t) \tag{A.5}$$

Mechanical Equation

Newton's second law for rotation is used to relate the torques with angular acceleration such that:

$$\tau_m(t) - \tau_f(t) - \tau_l(t) = J_m\, \ddot{\theta}_m(t) \tag{A.6}$$

where:

τ_m	is the torque generated by the motor [Nm]
J_m	is the motor inertia [Kg m^2]
τ_f	is the friction torque [Nm]
τ_l	is the load torque [Nm]

Often, in robotic applications, the speed required by the load (manipulator) is rather smaller than the motor speed. For such cases, gears are placed between the motor and the load in order to reduce η times the angular velocity of the load. At the same time, the gear ratio η causes an increase in the torque τ seen at the load (or a load torque reduction seen at the motor)[Craig, 2009, Ch. 9]:

$$\tau = \quad \eta\, \tau_m \tag{A.7}$$

Furthermore, in any motor there is loss due to friction. It is modelled as a torque opposed to the motor torque that is dependent on the motor angular velocity $\dot{\theta}_m(t)$, which is divided into viscous and Coulomb friction:

$$\tau_f(t) = v_m\, \dot{\theta}_m(t) + c_m\, \mathrm{sign}(\dot{\theta}_m(t)) \tag{A.8}$$

where v_m is the friction-viscous constant of the motor [Nm s/rad] and c_m is the Coulomb friction constant of the motor [Nm]. Commonly, friction is also affected by the joint position in many manipulator joints, but this dependency is not considered in this thesis. Due to the gear, the load torque from the manipulator is η times smaller in the motor dynamics, that is:

$$J_m\, \ddot{\theta}_m(t) + v_m\, \dot{\theta}_m(t) + c_m\, \mathrm{sign}(\dot{\theta}_m(t)) = \tau_m(t) - \tau_l(t)/\eta \tag{A.9}$$

Now, angular position of the motor θ_m and joint position q can be related using the gear ratio as:

$$\theta_{m_i} = \eta_i\, q_i \tag{A.10}$$

where η_i is the gear ratio of joint i, with units [-] if joint is **revolute**, and [1/m] if joint is **prismatic**. Then Eq. (A.9) can be written in terms of generalized coordinates q_i of the manipulator. For joint i and motor i, that is:

$$\tau_l = \eta_i \tau_{m_i} - \eta_i^2 J_{m_i}\, \ddot{q}_i - \eta_i^2 v_{m_i}\, \dot{q}_i - \eta_i c_{m_i}\, \mathrm{sign}(\dot{q}_i) \tag{A.11}$$

Note that when the motor is seen from the manipulator side, both motor inertia and motor viscous friction appear to be increased by a factor η^2, while the motor torque and motor Coulomb friction are increased by a factor of η. The motor dynamics in Eq. (A.11) are now merged with the manipulator dynamics in Eq. (A.1) considering the load torque from the manipulator is the generalized torque [Spong et al., 2004], that is $\tau = \tau_l$. Once they are written in matrix form, both manipulator and motor dynamics become:

$$\boldsymbol{\eta}\, \tau_m = \left(\mathbf{J}_m + \mathbf{M}(q)\right) \ddot{q} + V(\dot{q}, q) + F(\dot{q}) + G(q) + \tau_{ext} \tag{A.12}$$

where $\mathbf{J}_m$ is the motor inertia diagonal matrix with diagonal elements $\eta_i^2 J_{m_i}$. Vectors $V(\dot{q}, q)$ and $G(q)$ are the manipulator dynamics terms. $F(\dot{q})$ now expresses the vector of combined Coulomb and viscous friction of both the motors and joints. However, note that $F(\dot{q})$ consist mostly of motor friction, as motor friction is increased when seen from the manipulator side as shown in Eq. (A.11). Finally, the input τ_m is a vector of motor torques τ_{m_i} and $\boldsymbol{\eta}$ is a diagonal matrix with gear ratios η_i in the diagonal. Finally, both manipulator inertia and motor inertia can be combined in one term $\mathbf{M}(q)$:

$$\boldsymbol{\eta}\, \tau_m = \mathbf{M}(q)\ddot{q} + V(\dot{q}, q) + F(\dot{q}) + G(q) + \tau_{ext} \tag{A.13}$$

Summary

The daVinci dynamics can be divided into three parts:

- The electrical dynamics of the DC motors, which can be considered to be very fas compared to the mechanical dynamics:

$$V - K_b\, \dot{\theta}_m = L\, \frac{di}{dt} + R\, i \tag{A.14}$$

- The relation of the motor current with the motor torque τ_m:

$$\tau_m = K_m i \tag{A.15}$$

- The mechanical dynamics of the motor and the manipulator:

$$\boldsymbol{\eta}\, \tau_m = \mathbf{M}(q)\ddot{q} + V(\dot{q}, q) + F(\dot{q}) + G(q) + \tau_{ext} \tag{A.16}$$

In this project, only the mechanical dynamics are used for force estimation, while the electrical dynamics are derived in order to understand how a DC motor works.

Appendix B

Setup Guide in ROS

This appendix serves as a guide to set up the ROS environment used in this thesis. Files and folder mentioned throughout this section can be obtained from Appendix C. It is assumed that Ubuntu is used as operating system in the computer, as ROS is currently only developed for Ubuntu. The guide is divided in two main parts: haptic device configuration and ROS configuration.

Geomagic Touch Configuration in Ubuntu

- Download the OpenHaptics SDK and the Geomagic Touch device driver from:
  `https://3dsystems.teamplatform.com/pages/102863?t=fptvcy2zbkcc`
- Install them. It is recommended to follow the ReadMe guides provided in the download site.
- Connect the Geomagic Touch device as shown below:

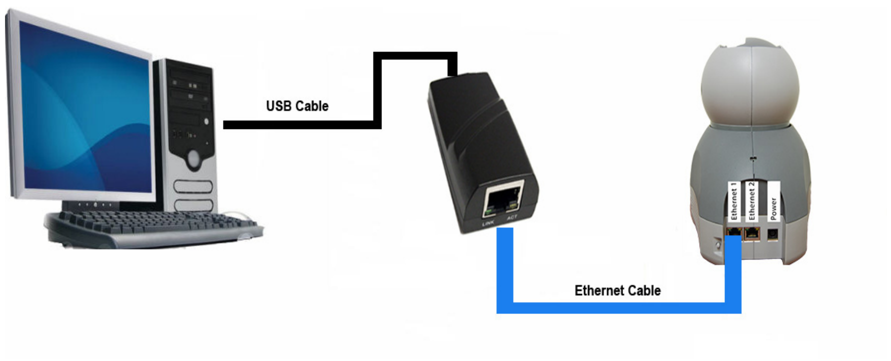


Figure B.1: Connection of the device to the computer. Source: [3DSystems, 2016b].

- Plug the Geomagic Touch device to the power supply. Ensure that the Geomagic Touch Status Indicator Light is lit (it will either be blinking yellow or solid orange if working properly). If it is not, check all of the connections.

- Click on the Network icon to see the list of the connections. If you have multiple wired connections, note the connection name that is using USB-LAN connection.

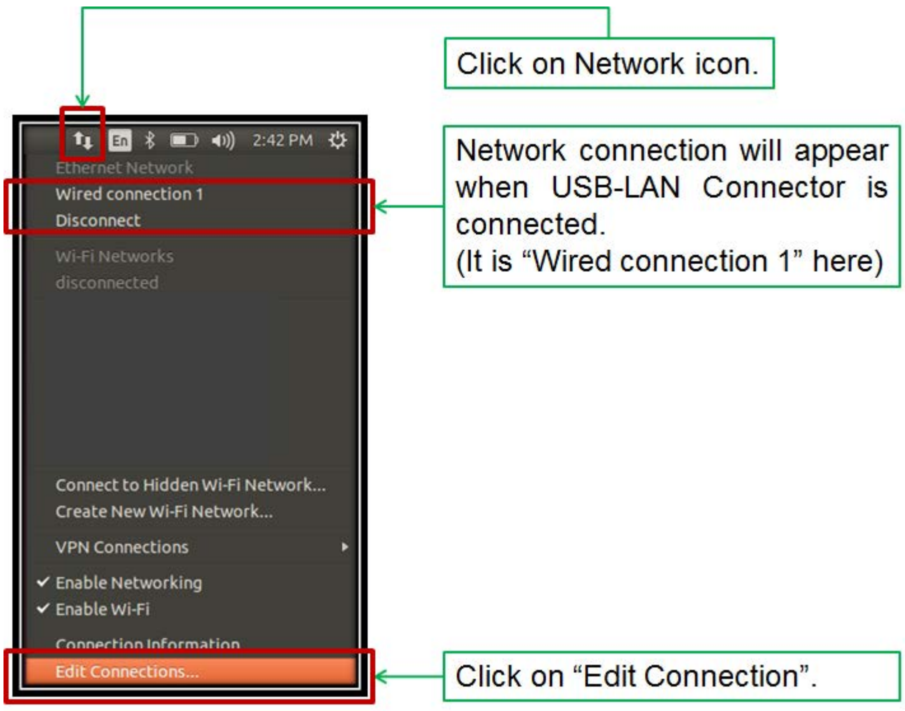

Click on Network icon.
Network connection will appear when USB-LAN Connector is connected.
(It is “Wired connection 1” here)
Click on “Edit Connection”.
2:42 PM
Ethernet Network
Wired connection 1
Disconnect
Wi-Fi Networks
disconnected
Connect to Hidden Wi-Fi Network...
Create New Wi-Fi Network...
VPN Connections
✓ Enable Networking
✓ Enable Wi-Fi
Connection Information
Edit Connections...

Figure B.2: Network configuration. Source: [3DSystems, 2016a].

- Click on *Edit Connection...*. Then, select the network connection name and click on *Edit*.

- After clicking *Edit*, a new window, *Editing <network connection name>* window will appear. Navigate to *IPv4 Settings* tab, and click on *Method* dropdown list. Select *Link-Local Only* option from the dropdown list and click on *Save*.

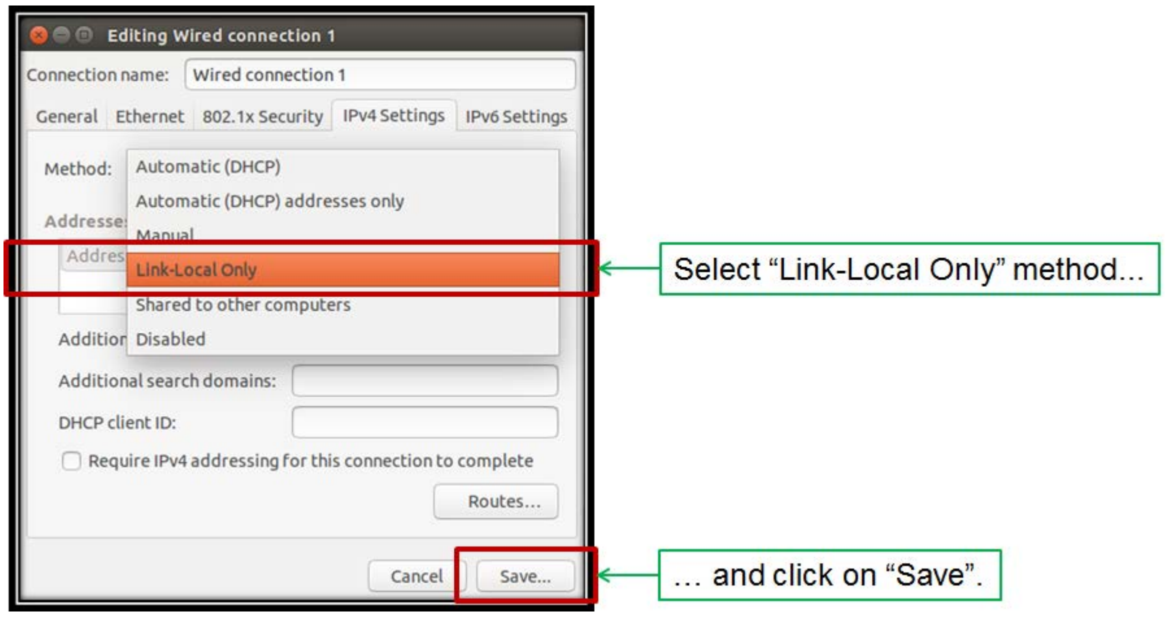
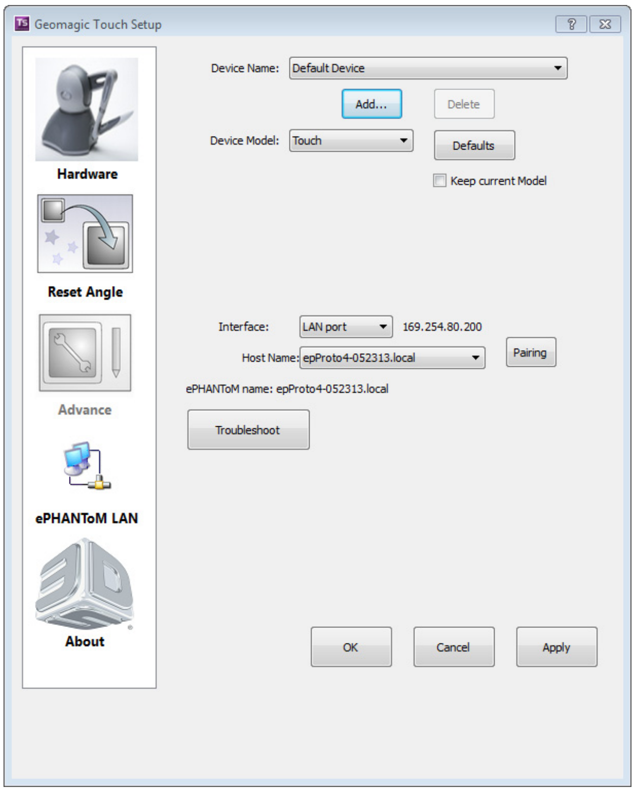

Editing Wired connection 1
Connection name: Wired connection 1
General
Ethernet
802.1x Security
IPv4 Settings
IPv6 Settings
Method:
Automatic (DHCP)
Automatic (DHCP) addresses only
Manual
Link-Local Only
Shared to other computers
Disabled
Additional search domains:
DHCP client ID:
Require IPv4 addressing for this connection to complete
Routes...
Cancel
Save...
Select “Link-Local Only” method...
... and click on “Save”.

Figure B.3: Select Link-Local. Source: [3DSystems, 2016a].

- Go to the installation folder of the Geomagic Touch in Ubunutu and open the Geomagic Touch Setup, by default it is in:
  `/opt/geomagic_touch_device_driver/Geomagic_Touch_Setup`

- Make sure that the correct Geomagic Device Model (Touch) is selected on the Hardware tab. Set *Device Name* to *Geomagic Touch*. If you choose another name, be sure it is also changed in the `phantom_omni` ROS node, otherwise the device will not be detected in ROS.

- After you have identified your device you will need to lock or *Pair* it with your host PC. To do so, click the *Pairing* button on the setup window and click on the *Pair* button on the back of the Geomagic Touch device.



Device Name: Default Device
Device Model: Touch
Interface: LAN port 169.254.80.200
Host Name: epProto4-052313.local
ePHANToM name: epProto4-052313.local

Figure B.4: The Geomagic Touch Setup Window. Source: [3DSystems, 2016b].

- To test the configured haptic devices, go to:
  `/opt/geomagic_touch_device_driver/`
  `Geomagic_Touch_Geomagic_Touch_Diagnostic`
- The Geomagic Touch device is now properly configured to be used in ROS.

ROS Configuration

- Install ROS. The follow URL is recommended:
  `http://wiki.ros.org/ROS/Installation`
- Create a ROS workspace in the terminal:
  ```
  $ mdkir -p ~/daVinci_ws/src
  ```
- Navigate to the source directory (`src`) and type the following to initialize the workspace:
  ```
  $ cd ~/daVinci_ws/src
  $ catkin_init_workspace
  ```

- Copy the following packages to the source directory (`src`) of the workspace:
  - `davinci_description`
  - `davinci_driver`
  - `omni_description`
  - `phantom_omni`
  - `haptic_interface`
- Build packages in the catkin workspace:
  ```
  $ cd ~/daVinci_ws
  $ catkin_make
  ```
- Add workspace to your ROS environment. This should be done every time a new terminal is open:
  ```
  source ~/daVinci_ws/devel/setup.bash
  ```
- Establish TCP/IP connection between ROS and the RIO board by launching the driver:
  ```
  $ roslaunch davinci_driver davinci_driver.launch
  ```
- Establish TCP/IP connection between ROS and the Geomagic Touch haptic device. Open a new terminal and type:
  ```
  $ roslaunch phantom_omni omni.launch
  ```
- Run the haptic interface node created in this thesis. Open a new terminal and type:
  ```
  $ rosrun haptic_interface haptic_interface
  ```
- Now, the user interface described in Chapter 9 appears:
  ```
  *****************************************************************
  The following modes are avaiable:
  -----------------------------------------------------------------
  press 'a' to move daVinci with Haptics Device (force feedback)
  press 'b' to move daVinci with Haptics Device (no force feedback)
  press 'c' to give joint setpoints to da Vinci (FK)
  press 'd' to give 3d force setpoint to joystick
  press 'e' to move Slide joint with Haptic Device
  *****************************************************************
  ```

Appendix C

Attached CD

The attached CD contains a digital copy of this thesis, altogether with all the code developed. They are divided as follows:

- MATLAB scripts:
  - System Identification scripts.
  - Simulink dynamic model for the da Vinci Surgical Robot.
  - Extended Kalman filter scripts.
- Maple code where the robot dynamics are derived using the Euler-Lagrange formulation.
- ROS Workspace used in this thesis.