Acad Year

(15/16)

Project No.

(C014)

Force Control Without Force Sensor for a Robot Manipulator

Force Control Without Force Sensor for a Robot Manipulator

Raymond Djajalaksana

SCHOOL OF MECHANICAL AND AEROSPACE ENGINEERING

NANYANG TECHNOLOGICAL UNIVERSITY

Year(2015/2016)

NANYANG TECHNOLOGICAL UNIVERSITY

C014

FORCE CONTROL WITHOUT FORCE SENSOR

FOR A ROBOT MANIPULATOR

Submitted in Partial Fulfillment of the Requirements

for the Degree of Bachelor of Mechanical Engineering

of the Nanyang Technological University

by

RAYMOND DJAJALAKSANA

SCHOOL OF MECHANICAL ENGINEERING

2015/2016

Abstract

In robotic assembly task, the robotic arm must be able to cooperate with a lot of uncertainites in the dynamic environment. Thus, the ability to know about contact force is crucial in assembly robot.

This project seeks to develop a model to estimate and control the contact force of manipulator arm without force sensor, using only motor currents or torques.

The report starts from literature review from previous researches about contact force estimation, followed by mathematical models to calculate the contact force from motor torques.

The next section will be about the necessary equipments and setups to run the experiments. Thereafter, results and model identification are presented, where the developed algorithm to estimate contact force is available in the following chapter. Verification of the result is also shown after the algorithm chapter.

The next chapter talks about the force control of the robot using motor torques, where one particular small assembly task is demonstrated.

The last chapter touches on the conclusion and future works needed from this project.

Acknowledgments

I would like to express my sincere gratitude and appreciation to the following people for their generous help and guidance, which made this final year project become feasible, enjoyable, and fruitful.

Assistant Professor Pham Quang Cuong, the project supervisor, for being generous with his advices and guidance.

Dr Fransisco Suarez, the research fellow, for his continuous help and assistance during the whole completion of the project.

All members of CRI group and Mechatronics friends, for their invaluable support regarding project matters.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 General Introduction

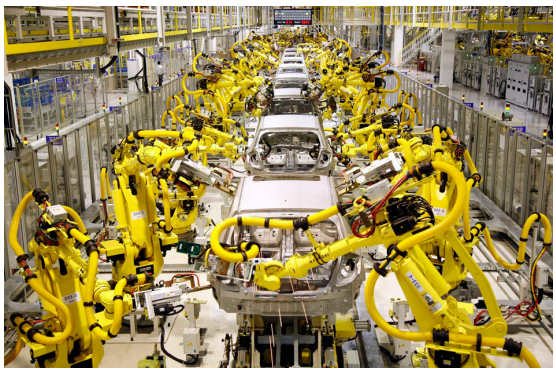
Nowadays, robotics play a very crucial role in industrial area by greatly increasing the industrial productivity. It helps factory workers in many ways:

First, it helps the workers in performing monotonous and tedious task. As humans are bad with dealing the same task over and over again, it will put high stress in workers physical and mental condition. In contrast, a machine or robot can perform the same task in a loop when programmed to do so. And normally it will achieve the same result which is also one advantage compared to workers result.

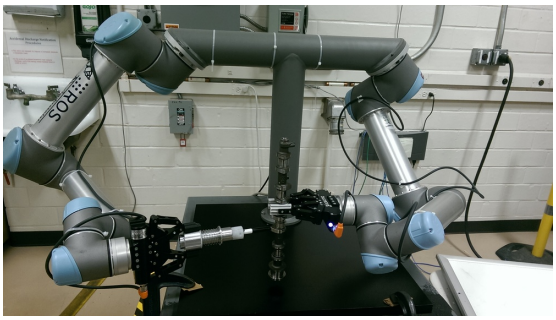
Second, it can replace human in doing some dangerous tasks. For instance, welding operation is quite dangerous as the activity deals with high temperature. By having the robot to perform the welding, it greatly reduce the risk for the workers to get burned or accident in the process of welding.

Third, robots are very good to deal with precision and accuracy while humans are not. This makes some operation can only be done by the robots and not manually. One example is making the electronics components such as micro chip, small transistors, etc.

However, these achievements are done because of a highly structured environment such as heavy industry (for example car assembly) where every parameters are known and fixed. In contrast, robotics performance in light industry are still poor. For instances, assembly of small and fragile parts in electronics, food, and other industries. This is because robotics are still bad in dealing with dynamics and unstructured environment where uncertainities are common.



(a) Robotics in heavy industries



(b) Manipulators for assembly task

Figure 1.1: Robotics in some applications

In light of this, researches and developments in this topic are still intense until now. Many works have attempted to create a framework for fine assembly procedures. For instance, a recent paper by (Suarez-Ruiz & Pham, 2015) has introduced the complete framework for fine assembly task.

1.2 Motivation

A typical six degrees of freedom robotic arm has many applications. By having a six degrees of freedom, it can moves freely in all translations and orientations. This robustness is one advantage that five or less degrees of freedom do not have. Thus, it is obvious to have many applications for such manipulator arm. Depending on the attached end-effector it has many purposes. For example, by attaching a gripper on the end-effector, it can perform assembly task or pick and place task. On the other hand, if a drill or a torch is attached, it can be used

for drilling or welding in industries. While the arm can also be used for assembly operation, its performance has still to be improved.

Particularly for robotic assembly tasks, to perform a versatile assembly the robotic arm must be able to cooperate with a lot of uncertainites in the dynamic environment. One example, the manipulator has to be able to know when the contact with an object is happening and then maintaining the stable contact throughout the operation. This ability requires knowledge of contact force for the robot. Thus, estimation of contact forces is very important since it will help the robot to determine and control the contact with objects in dynamics environment. While this can be done using accurate force/torque sensor, the sensor is normally expensive and requires mechanical integration with the robotic arm (Wahrburg, Morara, Cesari, Matthias, & Ding, 2015). Hence in some arms it might not be possible to attach this sensor. However, most of the robotic arms will have feedback of their motor currents and maybe torques. And obviously, this motor currents will drive the manipulator to do the works and so contact force should be able to be determined from this parameter without relying on additional sensor. This idea becomes one of the motivation of this project.

1.3 Objective

This project aims to estimate and control the contact force of an assembly robot based on the arm motor currents/torques. Understanding of the mathematical model of the robots dynamic, friction, and control theory are considered as important knowledges to work with this project.

The project will be focusing on a certain Denso arm. Thus, the developed systems will be built specifically for this arm. Additionally, some problems that are discussed in this project will be only addressed for this Denso arm and might not be available for other arms.

1.4 Scope

The scope of this project is divided into five parts. First, the project will start from understanding of the general models of robot dynamic and friction. Thus, literature reviews and readings are included in this step. The next step is to perform some experiments to get all necessary data to develop the model. This includes setup preparations, running the experiments, and collections of the data. The third step will be processing all the results and develop the system to estimate the contact force. Which after that, validation of the built model to the real data will be the next step. Lastly, the system will be used and tested to do force control in one small assembly task which is a pin insertion task.

Chapter 2

Literature Review

Many researches have been done in order to estimate the contact force. The main idea to estimate the contact force is to directly apply dynamic equations of a robot, knowing the value of joint torques. However different approaches have also been explored in the past few decades. Early approaches use observers for force estimation like in (Ohishi, Miyazaki, Fujita, & Ogino, 1991). Another approach in (Stolt, Linderoth, Robertsson, & Johansson, 2012) involves detune the low-level joint position control loop to estimate contact force. Furthermore, recent approaches by using Bayesian approach and generalized momentum with Kalman filter are studied in (Wahrburg, Zeiss, Matthias, & Ding, 2014) and (Wahrburg et al., 2015) respectively. Additionally, studies of comparison between two different approaches are done in (Damme et al., 2011). The study compares the result from filtered dynamic equation of external force with generalized momentum method. More details of these works will be explained in the next subsection.

2.1 Related Works

The purpose of this project is very similiar to (Wahrburg et al., 2014) in which the paper aims to estimate the contact force by using motor torques. However, the focus of the authors is on the new extensive method to further improved the

estimation value. First, they introduce the problem description which start from the basic dynamic of robotic system. And then the paper begins explaining the solution to estimate the contact force by using Bayesian approach. It depends on tuning the covariance matrices in their Bayesian method to estimate the error estimation and hence compensate it to get the more accurate contact force. Apart from there, the paper also consider the friction discussion since it will affect the result quite severe. Based on the previous paper, they concludes that by using a simple static friction model which is a coulomb and viscous friction model, they can get a reasonable calculation of the friction. To calculate the friction model, the robot was moved in a free motion without any load since in this case the friction will play a major role. The next part is where the authors introduce steps that needs to be done to use the new proposed method and the results of their experiments. They summarized that their method has been verified with the experimental data.

Another research which most of the authors are the same in (Wahrburg et al., 2014) propose a different approach in estimating the contact force of robotic manipulator (Wahrburg et al., 2015). This time, they use generalized momentum to simplified the basic robotic system equation. And from there, Kalman filter was used to have better estimation of the cartesian contact force. Unfortunately, the method presented in this paper could not be used now as the dynamic parameters such as mass, inertia moment, etc. are needed for generalized momentum while for now, the identification of those parameters have not been done for the manipulator arm that is going to be used. Hence, this method will not be applicable as for now, until dynamic parameters of the arm has been identified.

Different study by (Stolt et al., 2012) shows that a force estimation can be performed by playing with the low-level joint control loop. The basic idea is that joint control error will act like a spring and hence, force can be estimated based on this control error. The idea was verified in a small part of assembly task. This

idea while it seems to be promising, it is not really applicable since tuning the gain at joint controller is out of the scope of this project.

One interesting paper by (Damme et al., 2011) compared about two different ways to estimate contact force at the end-effector of the robot. The methods they disccussed are: recursive least-square algorithm with filtered dynamic model approach and generalized momentum based disturbance observer. It is notable that in this paper, they also use the simple static friction model like what (Wahrburg et al., 2014) used. After running the simulation and experiments, it was found that both of approaches give a similiar results despite a different origin. It was suggested that observer-based algorithm is better if fast response is needed, while least-squares based algorithm is better for system with noise and time lag tolerance.

To sum up, there are many approaches to estimate the contact force of end-effector manipulators that have been studied. While each have their own advantages and disadvantages, all of the methods are sourced from the basic dynamic system of the robot.

Chapter 3

Mathematical Model

3.1 Arm Dynamic Equation

The basic dynamic equation of a six degrees of freedom (6-DOF) robotic arm is described by:

$$\Sigma\tau_{joint} = \tau_{mot} + \tau_{ext} = M(q)\ddot{q} + C(\dot{q}, q)\dot{q} + G(q) + \tau_{friction} \tag{3.1}$$

Where τ_{mot} is the motor torque, τ_{ext} is the external torques applied to the respective joint where in this project is due to end-effector force / torque, and $\tau_{friction}$ is the torque because of joint friction. $M(q)$ defines the inertia matrix, $C(\dot{q}, q)$ defines the coriolis matrix, and $G(q)$ is the joint torque resulting from gravity. The combination of these three terms can be called as dynamic torque (τ_{dyn}). In other words, for each of joint, the summation of external joint torques ($\tau_{mot} + \tau_{ext}$) are equal to the torques needed to overcome the friction and to perform the dynamic motion. Initially the motor torque will be calculated from motor currents by using the relation:

$$\tau_{mot} = K_m I_{mot} \tag{3.2}$$

However, there is a problem regarding motor currents where the solution is to get the motor torque value directly from the arm (see chapter 5.1). And since the unit is in percentage (%) and not in SI unit, calibration is needed to change it to SI. Thus, it can be written as:

$$\tau_{mot} = K_{denso}\tau_{denso} \tag{3.3}$$

Where K_{denso} will be called denso gain for ease of reference.

On the other hand, transformation of the end-effector wrench to the external joint torques need a Jacobian matrix, which the equation is:

$$\tau_{ext} = J^T F_{ext} \tag{3.4}$$

Where $J^T \in R^{6\times6}$ is the transpose of jacobian matrix and $F_{ext} = [F\ \tau] \in R^{6\times1}$ is the contact force. It is also important to consider friction since it can cause large error in manipulators (Kermani, Wong, Patel, Moallem, & Ostojic, 2004). However, the modeling of friction is not easy as there are a lot of phenomenas in the friction itself. Thus, it is good to choose the simplest model that can have a reasonable result. The next section below will explain the friction model in more detail.

3.2 Friction Model

In general, friction models can be divided into two categories: static and dynamic friction. A model that depends only on current velocity is called static friction, whereas the friction related to non-stationary velocities is called dynamic model.

In static friction, including coulomb and viscous friction, stiction phase and stribeck effect can be also captured. However, dynamic friction can capture

more phenomenas such as: small displacements occurring during stiction phase, hysteretic effect, and variations in break-away force. In short, dynamic model explains the friction in microscopic level and hence can capture more phenomenas.

During the early stage of the project, both type of model will be considered, whereby in the late stage only the static model will be used as dynamic model has some difficulties.

3.2.1 Static Friction Model

Viscous and Coulomb is the simplest form of static friction model. The basic mathematical form is as below:

$$F = F_c sign\left(\dot{x}\right) + \beta\dot{x} \tag{3.5}$$

Where F is the friction force, $\dot{x}$ is relative velocity to contact surfaces, β is the viscous friction coefficient, and F_c is coulomb friction. The model however does not include the stiction effect. The stiction effect can be easily added into the model, that is: a movement can be created only when the applied external force is greater than friction force Fs (Bona & Indri, 2005). By adding this effect and adjusting the equation for robotic arm, the equation will be:

$$\tau_{friction} = K_c sign\left(\dot{q}\right) + K_v\dot{q} \tag{3.6}$$

Because of stiction and coulomb effect, the friction model will introduce discontinuous profile when the direction of velocity changes. The friction force diagram for this model can be seen in figure below where it shows a discontinuity at velocity equals to zero.

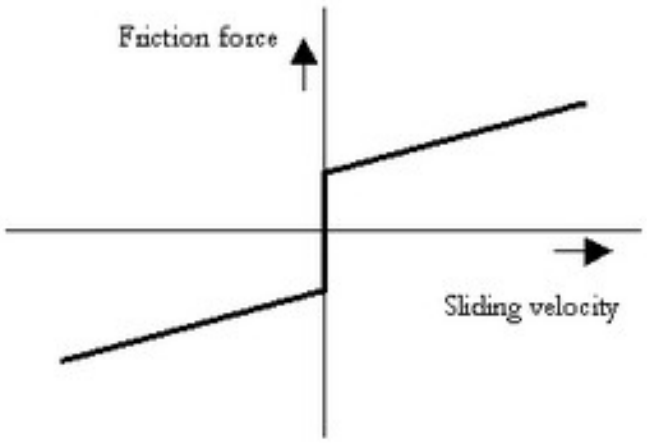


Figure 3.1: Static friction profile

3.2.2 Dynamic Friction Model

The well-known Dahl model is the simplest form to describe the dynamic friction behaviour. With Dahl model, hysteretic effects can be explained since it introduces the lag in the changes of friction force as velocity changes. However, Dahl model does not include some other effects, such as stiction and Stribeck effect. A refined form of Dahl model that includes these effects is called LuGre friction model (Bona & Indri, 2005).

Dahl friction plays with an internal state variable z, and then defines the friction force as:

$$\dot{z} = \dot{q} - \frac{|\dot{q}|}{F_c}\sigma z \tag{3.7}$$

$$\tau_{friction} = \sigma z \tag{3.8}$$

Where σ is the bristle stiffness parameter. Because of this internal state z, Dahl model can explain the hysteretic effect in friction. Be aware that for this equation, it requires a correct initial value of z. Figure below shows the diagram of Dahl model.

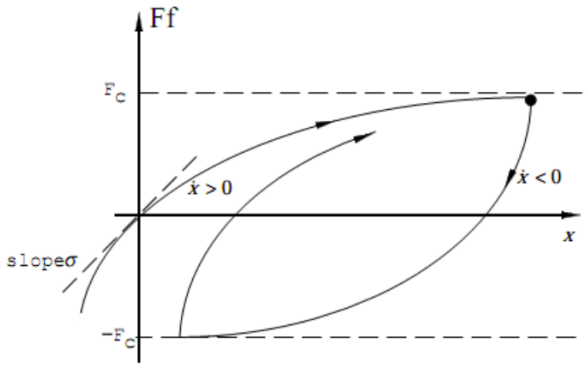


Figure 3.2: Dahl model of dynamic friction

Unlike Coulomb and viscous friction model, Dahl model does not have discontinuous profile in the equation. This becomes one advantage Dahl model has over the static friction. However, since the model is more complex, it imposes some difficulties such as: estimation of initial state is not easy, the result is sensitive to the parameter (σ), and the necessity of high precision data.

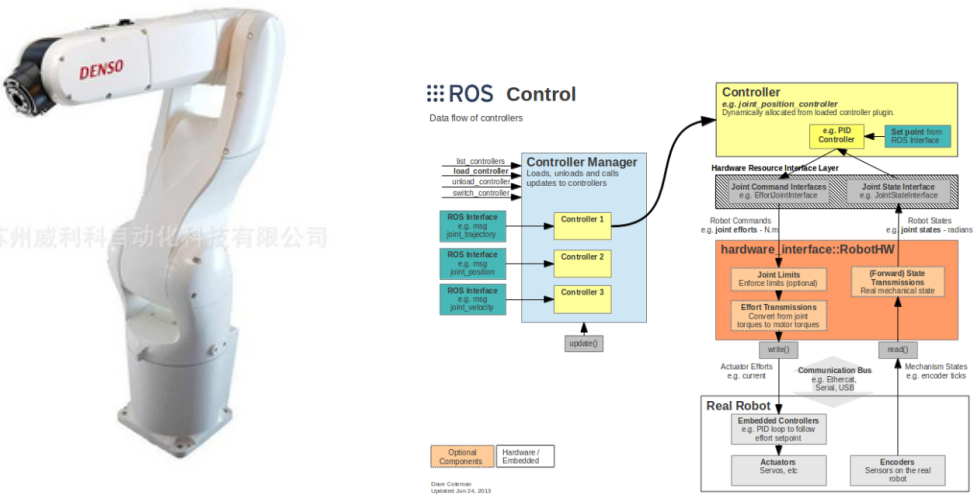
Chapter 4

Equipments

In this project, necessary equipments, software, and robotic arm that are going to be used are: 1 × Denso VS060A3-AV6 arm, 1 × end-effector handle, 1 × ATI Gamma F/T Sensor (SI-32-2.5 calibration), 1 × Robotiq gripper, Ubuntu 12.04 LTS, OpenRave, and Denso ROS.

4.1 Denso VS060

Denso VS060A3-AV6 is a six DOF robotic arm from Denso company. It has absolute encoder for all of its joint position. An RC-8 controller is also provided for interfacing with this arm. The computer is connected to Denso arm via LAN cable. The architecture of the hardware interface for this arm can be seen in figure below. By using Denso ROS as the ROS, the robot can be moved by sending a command position or a list of positions. Thus, controlling the robot from velocity or acceleration domain are not necessary as it has been taken care by the program and controller.



(a) Denso VS060 arm (b) Hardware interface of Denso arm

Figure 4.1: Denso VS060A3-AV6

4.2 ATI Gamma F/T Sensor

ATI Gamma F/T sensor SI-32-2.5 is used for the experiment. It has been calibrated with the following sensing range: $f = [32, 32, 100]N$ and $\tau = [2.5, 2.5, 2.5]Nm$. This F/T Sensor is attached into the end-effector of Denso arm to measure the contact force and torque. After this sensor, another handle is attached on top of the F/T sensor. Hence, with the addition of contact force and torque, F/T sensor will also read the weight and inertial force from the handle.

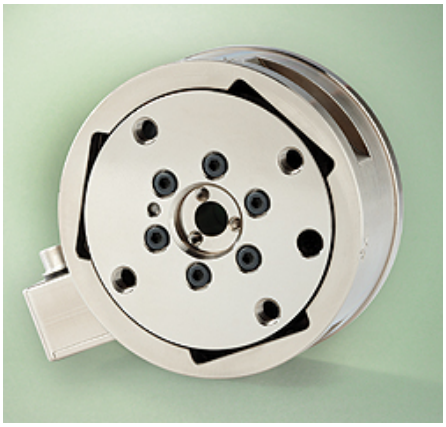


Figure 4.2: ATI Gamma F/T Sensor

4.3 Robotiq Gripper

This two-finger gripper from Robotiq will be used for this project to perform one particular assembly task, which is pin insertion to the hole. The gripper has an opening up to 85 mm and a grip force from 5 to 220N.

Figure 4.3: Robotiq gripper

4.4 End-effector handle

This is the handle of the end-effector arm. It has a round sphere surface. The arm will make a contact with environments through this handle. This handle is primarily used for model parameters identification.

Figure 4.4: End-effector handle

4.5 Ubuntu 12.04 LTS

The OS of the working computer is Linux Ubuntu, with version of 12.04. It is not the lattest version and it should stay in 12.04 version for operating the arm. Some important packages are also installed, they are OpenRave and Denso ROS. Python and C++ are used to run the Denso arm.

4.6 OpenRave

OpenRAVE is a package that provides an environment for testing, developing, and deploying motion planning algorithms in robotics applications. It focuses on the simulation analysis of motion planning. It is to be used together with Denso ROS to control the Denso arm with its analysis guidance. The OpenRave is oftenly used in Python script.

Figure 4.5: OpenRave environment

4.7 Denso ROS

Denso ROS is a robot operating system that works on the Denso arm. With Denso ROS, the manipulator arm can be controlled from computer. The packages

are written in C++ and Python while the script to run the robot is usually written in Python codes.

Chapter 5

Methodology

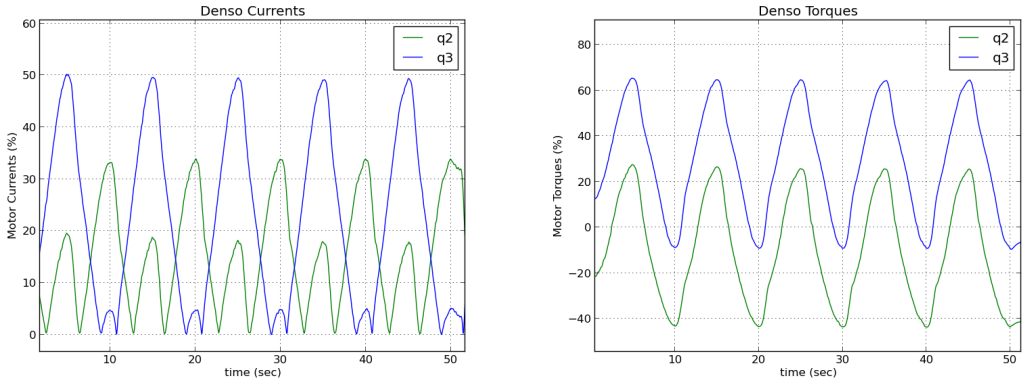
5.1 Motor Currents Reading

Before continue with more experiments and identification, there is one problem that needs to be solved first: the arm only gives absolute value of the motor currents (Fig. 5.1a). In the figures, q2 refers to the second joint, and q3 is the third joint. The profile of the motor currents in the figure are supposed to be sinusoidal profile with 5 peaks. Instead, the Denso arm gives about 10 peaks. Thus there is a need to identify the sign of the motor currents of each joints. Three methods were tried but only one was successful.

The first attempt was to give the motor current sign based on the sign of τ_{dyn} during non-contact condition($\tau_{dyn} = M(q)\ddot{q} + C(\dot{q}, q)\dot{q} + G(q)$). However, this poses a problem when the τ_{dyn} goes near zero as it will not be clear enough to identify the sign. The second method was to match the derivative of motor currents with derivative τ_{dyn} during pre-contact motion (e.g.: if τ_{dyn} is increasing, the motor currents has to be increasing too, and vice versa). Unfortunately, this principle is working only if motor currents are very smooth and steady which is a very difficult condition.

After the first two attempts, it was found out that we can actually extract

motor torques value from Denso and this time it includes the sign of it. The data is then plotted against motor currents with the sign of motor torques to see the relation. From Fig. 5.2 we can see that there is a good linearity between these two variables, and so we can now actually use motor torques instead of motor currents. By doing this, not only we do not need to take care of the currents sign problem anymore, we also simplify our problem since what we are interested in is the motor torques, not motor currents. However, since the value is not in the SI unit calibration is needed to adjust the value into the SI unit.



(a) Raw data of motor currents (b) Raw data of motor torques

Figure 5.1: Reading of motor currents and torques

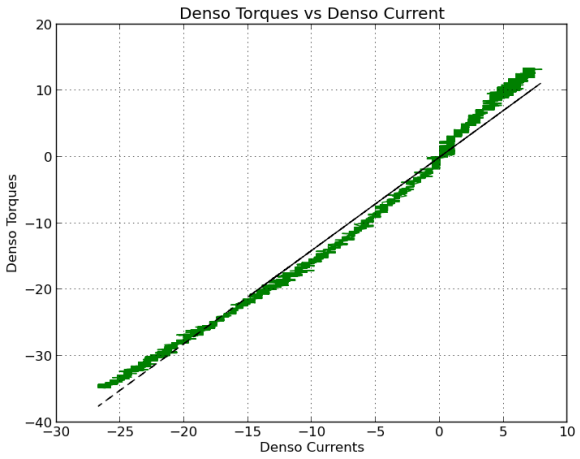
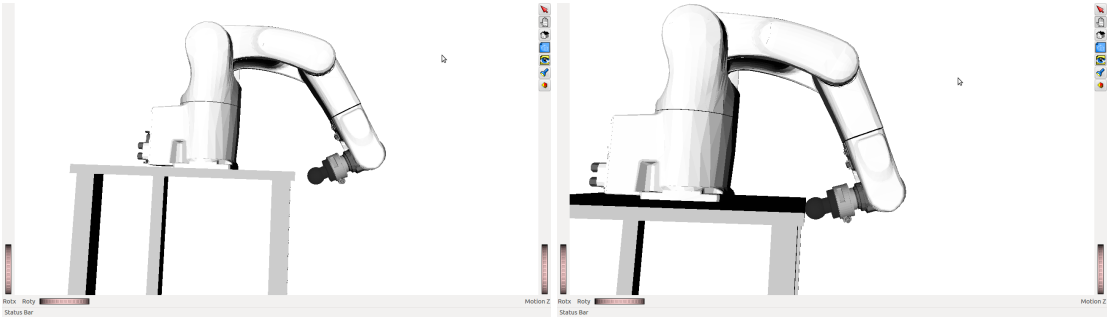


Figure 5.2: Motor torques vs motor currents with sign

5.2 One-stage Experiment

In the early process of the project, one-stage experiment was executed. The experiment can be described like this: The robot is moved into pre-contact position near any fixed object. The arm then moves slowly until contact is detected using F/T sensor. From there, a sinusoidal motion of the end-effector is performed to push the fixed object. This will make some joints to forcely push the robot. Figures below illustrate the simulation of the setup.



(a) Robot is in pre-contact position

(b) robot push the ground in a sinusoidal motion.

Figure 5.3: One-stage experiment simulation

The purpose of this experiment is to capture all of the phenomenas (friction and contact force) in one experiment and so, identification of all parameters can be done for each joint in one experiment. Furthermore, depending on the setup one experiment can test for more than one joint at the same time, thus cutting the time to do the experiment.

While it is feasible to do so, identification of the parameters turned out to be difficult. The reasons are: 1) while all effects of the phenomena can be captured, it is difficult to distinguish one with each other, hence making it difficult in identifying each parameter and 2) performing the experiment for two or more joints at the same time is not recommended as it can be seen as inconsistent experiment (e.g: one experiment that captures for one joint might give different results when captures for two joints). Hence, the setup of experiment was then

changed to be two-stage experiment.

5.3 Two-Stage Experiment

As the name suggest, for each joint there are two experiments which are required to be performed to identify the two important parameters : K_{denso} and $\tau_{friction}$ (see chapter 3.1). The purpose of setting different type of experiment is to eliminate the effect of other parameters, thus the resulted data is to be affected only because of one parameter at one experiment. That way the identification of each parameter tends to be easier. The first stage of experiment is the high torque collection data to calibrate the motor torque (K_{denso}) while the second stage of experiment is the high velocity collection data which is performed to identify the $\tau_{friciton}$ characteristics. The data is captured in broad range of value to capture as many phenomena as possible.

5.3.1 High Torque Collection Data

The experiment is meant to collect the torque measurement from wide range to calibrate the motor torque. Firstly, the robot is moved to a position such that the interested joint is prone to receive a high external torque from contact force. While the robot is fixed, the force is introduced at the end-effector of the arm such that the interested joint will experience a high torque. Motor torques are recorded from Denso arm and contact force/torque are recorded through ATI F/T sensor. The experiment is repeated in a different position for each joints. See figure below for illustration.

Figure 5.4: High torque collection data experiment for second joint

5.3.2 High Velocity Collection Data

The setups for this experiment are based on (Prete, Mansard, Ponce, Stasse, & Nori, 2015) and (Vuong & Jr., 2009). To capture the friction phenomena, one joint of interest will be moved in a sinusoidal motion. However, the robot must be positioned accordingly such that when this joint is moving, the gravity effect of the links is either zero/unaffected or constant, thus making the interested joint moves purely because of inertia of the linkages and joint friction. Experiments where the joint torque will not be affected by gravity can be done for all joints except second and third joint. The four joints are then moved with the equation of motion:

$$q = (a_0 + at)\, sin(\omega t) \tag{5.1}$$

On the other hand, setups for the second and third joints will be slightly different as gravity will always affect these two joints. Thus, other setups are to position the robot such that $\frac{dG}{dq}$ is minimized for the respective joint. In this way, the changes in the motor torques due to the gravity will be minimized and hence, it is expected to give a similiar results with the other four joints. However,

this means that the joint cannot be moved too far from the position and so the previous equation of motion cannot be used for the setup as it tends to move further from the stable position. Instead, another equation of motion will be executed for these two joints:

$$q = (a_0)\, sin(\omega_t t) \tag{5.2}$$

$$\omega_t = \omega_0 + Bt \tag{5.3}$$

The figures below illustrate the joint movements during the high velocity experiments. During the experiments data of denso torques, joint positions, and F/T sensor are recorded.

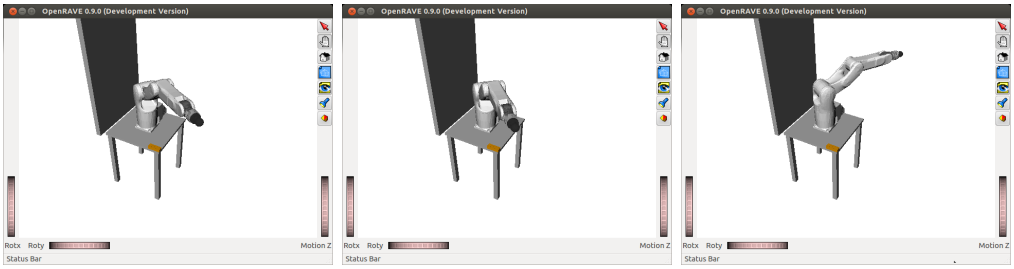


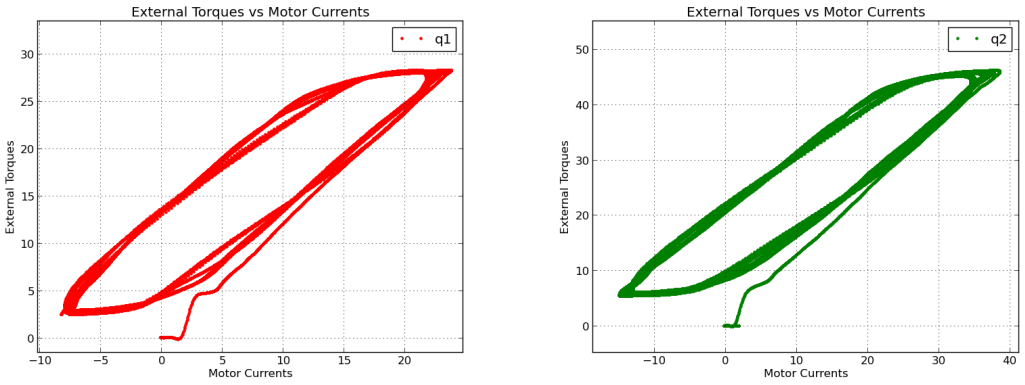
Figure 5.5: fig: High velocitiy collection data experiment for first joint

Chapter 6

Results and Model Identification

6.1 Identification for One-stage Experiment

The sample result of experiments can be seen in Fig. 6.1. The x-axis is the motor torques while y-axis is the external joint torques that are transformed from F/T sensor. The reference value (zero-value) of both axis is taken from pre-contact position.



(a) First joint (b) Second joint

Figure 6.1: Sample results of one-stage experiment (first two joints)

Firstly, the value for identification is referenced with pre-contact condition. From computation it is known that there is only a few changes in τ_{dyn} ($\tau_{dyn} = M(q)\ddot{q} + C(\dot{q}, q)\dot{q} + G(q)$) during contact motion and so it can be ommitted.

Thus, the dynamic equation can be simplified into just:

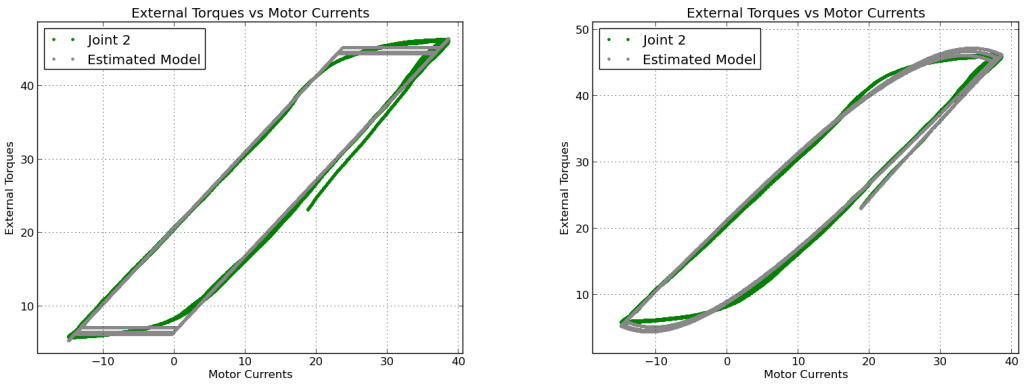
$$J^T \Delta F_{ext} = K_{denso} \Delta \tau_{denso} - \tau_{friction} + C \tag{6.1}$$

Here a constant is introduced where it is actually a friction during precontact movement, $\tau_{friction,precontact}$. For friction, we will identify it using static friction and Dahl friction. As for Dahl friction, some adjustments to the equation need to be done to match with the optimization. The modified Dahl equation that is going to be used is :

$$\dot{z} = p_2 \dot{\tau_{mot}} - |\dot{\tau_{mot}}| \, p_3 z \tag{6.2}$$

$$\tau_{friction} = p_1 z \tag{6.3}$$

To identify all the unknown parameters, the equation is optimized to match with the fitted data. The number of parameters to be optimized will depend on the model of friction that we choose. This optimization is done by using nelder-mead method. One of the result of optimization is shown in Fig. 6.2. It shows the result for both friction model that is going to be used.



(a) Using viscous and coulomb model (b) Using Dahl model

Figure 6.2: Example of optimization result of one joint (e.g.:second joint)

From Fig. 6.2, it is quite clear that the results will depend on the friction

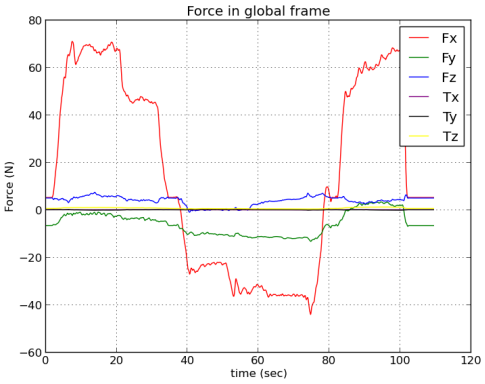
model that we choose. Using Dahl model, the error value to the fitted data is less than using coulomb + viscous friction. However, Dahl model imposes one problem. The model requires initial state of z. During optimization, this value is unknown and hence it was optimized together with other parameters. However, this initial state will always change for every different position of the arm. Thus, it makes the model more difficult to implement since we need to know the initial state. This will be more clear in the next chapter of validation (see chapter 8).

While the identification can be done, it is difficult to distinguish the effect of each parameter, especially between viscous effect and denso gain. Hence, the experiment was then rescheduled using two-stage experiment (see section 5.3).

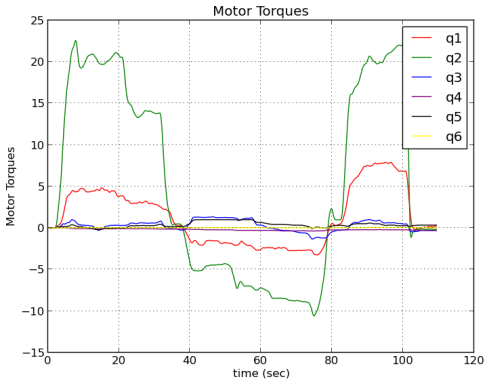
6.2 Preliminary Results for Two-stage Experiment

Two figures below are some samples of the collected data during the experiment. The whole data are available in the appendix. The data has been filtered to eliminate the noise reading available from the sensor or motor. It uses low pass filter for smoothing the result. Low pass filter for F/T sensor have the order of 3 with cutoff frequency of 2 Hz. As for the motor torque, the low pass filter is set to have the order of 5 and cutoff frequency of 1 Hz unless special adjustment is required.

Fig. 6.3 represents the result of high torque experiment for one of the joint and Fig. 6.4 represents the data collected during the free motion experiment. The results from Fig. 6.3 will be further processed for calibrating the motor torque while results in Fig. 6.4 are used for friction identification.

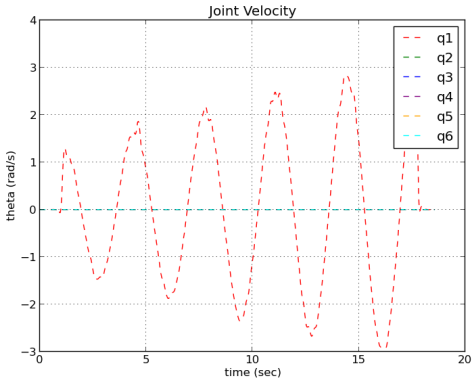


(a) Filtered data of external wrench detected from F/T sensor

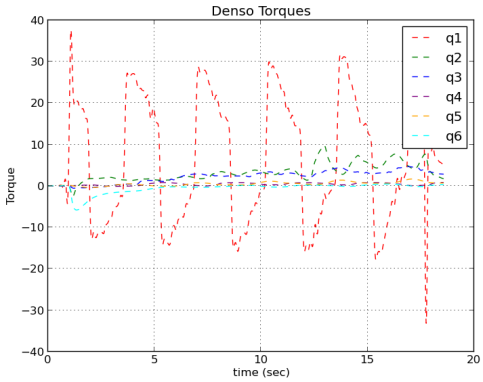


(b) Filtered data of denso torques

Figure 6.3: Sample data of the second joint high torque experiment



(a) Filtered data of joint velocity



(b) Filtered data of denso torques

Figure 6.4: Sample data of the first joint high velocity experiment

6.3 Denso Gain Identification

Based on the setup that has been mentioned in subsection 5.3.1, it is known that during the experiment the joints are stationary and hence $\dot{q} = 0$, $\ddot{q} = 0$. Also since it is not moving, there is no friction effect for the joint. Hence by using relation in (3.3) and (3.4), equation (3.1) can be simplified to:

$$-J^T F_{ext} = K_{denso}\tau_{denso} - G(q) \tag{6.4}$$

This makes the calibration of K_{denso} becomes more easier to identify as it is a simple linear problem. To get the value of K_{denso}, the model is optimized from the data (τ_{denso}, F_{ext}) that has been gathered. The optimization is still done by using nelder-mead method.

The diagram in Fig. 6.5 shows the plot of τ_{denso} against $-J^T F_{ext}$. The green line represents the experimental data while the black line is the model with optimized parameter K_{denso}. As it can be seen, the data is not perfectly linear as what it is supposed to. The reason might be because of the deadzone in motor controller: where errors below some value will be counted as zero. To adjust for this effect, piecewise-linear model can be used to fit the data. Although it will give better results, there are some limitations to the model that make the implementation to be difficult (Prete et al., 2015). Hence, simple linear model is still chosen despite the unability to capture the deadzone.

Since the deadzone was not taken into account, the result tends to underestimate the external torques. While it might be undesirable to accurately estimate the contact force, it is better than overestimating the external torques. It is reported from (Prete et al., 2015) that the controller will become more unstable when the system overestimate the force. This is also another reason not to recalculate the denso gain.

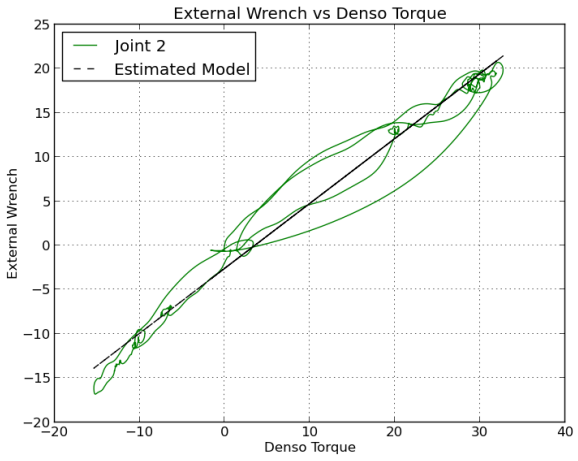


Figure 6.5: τ_{denso} vs $-J^T F_{ext}$ in high torque experiment for second joint

The results of the identification can be seen in the table below:

Joint	$K_{denso}(N.m/\%)$
1	0.9726975092370144
2	0.7374652918414186
3	0.5011872413412258
4	0.17752933989182662
5	0.22525237806401718
6	0.08119127531050022

Table 6.1: Identification of denso gain parameter

6.4 Friction Identification

From section 5.3.2 it is known the torques that were exerted during the experiment are coming from friction and inertia torques only. And by choosing the static friction model, the complex equation of 3.1 can be reduced to be a simple dynamic equation of the rigid body:

$$\tau_{motor} = K_{denso}\tau_{denso} = K_c sign\left(\dot{q}\right) + K_v\dot{q} + I\ddot{q} \tag{6.5}$$

Where I here is the inertia due to the arm links to the joint. Since all the joints are stationary except for the joint of interest, the inertia should remain constant. This equation is then fitted with the real data to get the three unknown parameters for each joint. As the optimization now is more complex compared to denso gain identification, the results have to be bounded to make sure that the optimization will not give a wrong answer. Since all of the parameters cannot be negative, the boundary will be: $[K_c, K_v, I] \geq 0$. the optimization is now done by using Sequential Least SQuares Programming (SLSQP) method provided by SciPy Python. This is because SLSQP can give a boundary condition to the

parameters which nelder-mead could not.

Figure below shows the result of optimization to the real plot for the first joint. It can be seen that the plot of the motor torques data are similiar with the static friction diagram(Fig. 3.1) although there are some differences. These differences are most likely due to the inertia that affects the motor torques value. For the second and third joint where inertia is involved, the differences can be seen clearer(Fig. A.4). In contrast, the inertia involved for fourth and sixth joint is very small such that the plot is now look the same as static friction.

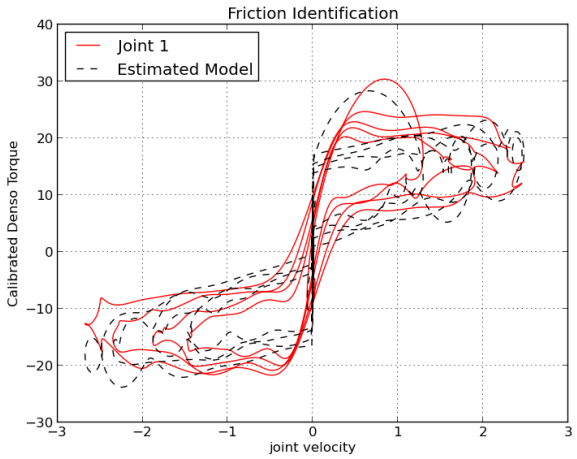


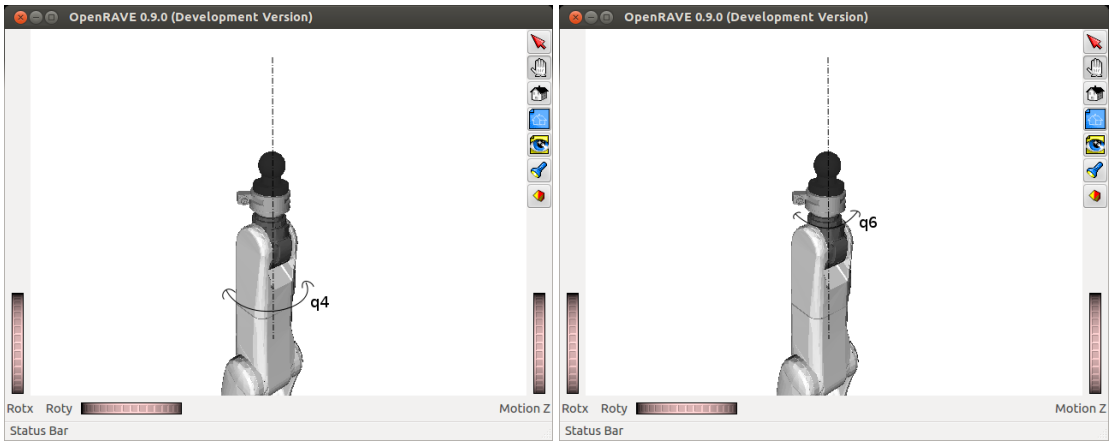
Figure 6.6: Results for friction identification

The identification results of all parameters for all joints are compiled on the table below:

Joint	$K_c(N.m)$	$K_v(N.s^2)$	$I(N.s^2)$
1	8.760254598800614	3.5593393087764476	1.2298914843787225
2	6.5844162226978975	16.631011445572256	1.6028523428327148
3	3.6559073575564893	7.240485152284423	0.4106659387975339
4	3.656967856159363	2.896319866560603	$1.204029526208893e^{-18}$
5	2.5006379799006013	3.798780079420939	0.12806276310671558
6	1.336290767286993	0.6059130581961736	0.0

Table 6.2: Identification of friction torque

It is interesting to see that the results for fourth and sixth joints give a very small inertia value. This might be acceptable when considering the setup of the experiments for these two joints. Fig. 6.7 show the setup for the fourth and sixth joints during the experiment. Based from this, the inertia will follow inertia of a cylindrical rod with small radius with the centerline as the rotation axis. This configuration will give a smaller value compared to other configurations. Because of this, the optimization will heavily emphasise friction more than the inertia.



(a) Fourth joint (b) Sixth joint

Figure 6.7: Setup of fourth and sixth joints during high velocity experiment

Chapter 7

Force Estimation Algorithm

7.1 Estimation of Contact Force using Motor Torques

After all necessary parameters have been identified, contact force can now be estimated using denso torques. The formula to measure the contact force is:

$$F_{contact} = \left(J^T\right)^{-1} \left(K_{denso}\tau_{denso} - \left(\tau_{dyn} + K_c sign\left(\dot{q}\right) + K_v\dot{q}\right)\right) \tag{7.1}$$

$$\tau_{dyn} = M\left(q\right)\ddot{q} + C\left(\dot{q}, q\right)\dot{q} + G\left(q\right) \tag{7.2}$$

Where $F_{contact}$ is the force that the robot gives to the object and τ_{dyn} is the torque due to the dynamic motion. Since dynamic model identification has not been performed, the calculation of τ_{dyn} could not be done manually and hence it relies on the result given from the OpenRave simulation. But before continuing further, the simulation result must be verified with the experimental data first to guarantee the truthness of the value.

7.2 Comparison of Denso Torque and OpenRave Simulation

To compare the value of calibrated denso torque with the OpenRave , a simple setup experiment can be done. The setup can be described like this: for one joint, the robot is positioned such that the motor will exert some torques due to the weight of the links only. Because it is not moving and no external force is introduced, the dynamic equation is reduced to be :

$$K_{denso}\tau_{denso} = G(q) \tag{7.3}$$

The value of $G(q)$ is computed using OpenRave. The value should be comparable to the left side of equation.

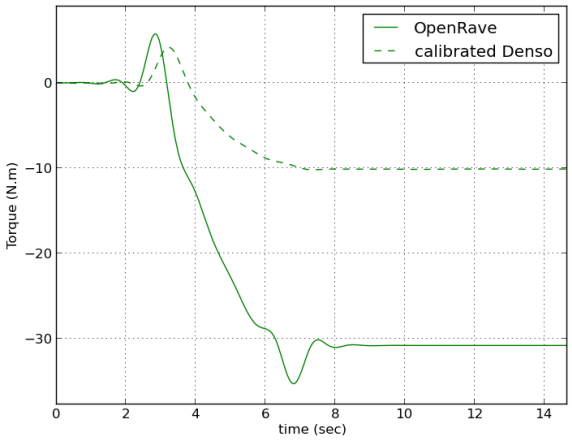


Figure 7.1: Joint Torque Verification for Second joint

However, the figure in Fig. 7.1 shows a different result. The continuous line represents the torque computed from OpenRave, (e.g.:$G(q)$) and the strip line represents the motor torque after calibration (e.g.: $K_{denso}\tau_{denso}$). It is very clear that values are not similiar. The value given in OpenRave is much higher than calibrated denso torque. There are two possibilites that might be the reason of this differences, which are: 1) The gain value (K_{denso}) is incorrect due to some

problems that might arise in F/T sensor (i.e.: F/T sensor not calibrated) or 2) The OpenRave gives incorrect computation. It is quite difficult to investigate the second reason as until now the real dynamic parameters of the Denso arm have not been identified, thus torque computation of dynamic motion could not be done for now except using OpenRave. On the other hand, the ATI F/T sensor should have been calibrated before the experiments, thus the first possibility is unlikey to happen.

Hence, from this comparison it can be found that there are some pieces that are still missing. Since the OpenRave result has not been investigated whether it is truly correct or not, it is decided not to use the OpenRave to compute the dynamic torque. However, it makes a hole in the model estimation since the dynamic torque is required. To solve this problem, quasi-static assumption is used in the model to fill the hole in the model.

7.3 Quasistatic Based Model

To solve the problem regarding τ_{dyn}, the model is assumed to be quasistatic condition. It means that the changes of τ_{dyn} between pre-contact and contact is negligible and hence, it can be assumed to be constant. Thus by carefully selecting the pre-contact as a reference point, this term will dissappear in the equation (7.1). Thus, the modified model equation becomes:

$$F_{contact} = \left(J^T\right)^{-1} \left(K_{denso}\tau'_{denso} - (\tau_{fric} - \tau_{fric,ref})\right) \tag{7.4}$$

$$\tau_{fric} = K_c sign\left(\dot{q}\right) + K_v\dot{q} \tag{7.5}$$

Here τ'_{denso} refers to the denso torque value after being referenced with the value during pre-contact position ($\tau'_{denso} = \tau_{denso} - \tau_{denso,init}$) and $\tau_{fric,ref}$ is the friction at the pre-contact point. Since it relies on the assumption that τ_{dyn} is

zero, there are two main limitations to this model. They are:

1) The pre-contact position cannot be too far from the contact point.

2) The robot should not move and accelerate too fast since τ_{dyn} is very sensitive to these parameter.

Hence it is best to choose the reference point as close as possible to the contact area and move the robot slowly from there.

7.4 Algorithm Flows

The following algorithm is created based on the quasistatic model that has been explained in the previous section. The steps of the algorithm are:

1) Move the robot to the pre-contact position. It is preferred to choose the point as close as possible to the contact area.

2) Move the robot slowly to the contact position.

3) Quickly make the initialization to the denso torque and friction. For each joint, take the current value of the denso torque as a zero-based value ($\tau_{denso,init}$) and calculate the friction from the joint velocity and store the value in the $\tau_{fric,ref}$. There are two points that can be choosen as reference points, they are: A. The point where the robot is not moving yet (step 1) or B. The point right after the robot starts to move (step 2). The only difference about these twp points is about the presence of friction. However, since the friction has been identified, ideally these two points should give the same results.

4) Estimate the contact force based on denso torques and joint velocities. The denso torque is then subtracted by $\tau_{denso,init}$ to get τ'_{denso} while joint velocity is used to estimate the friction part. Use the equation (7.4) to get the contact force estimation.

These steps are the general steps required to compute the contact force. Some adjustments can be added accordingly depending on the usage of the algo-

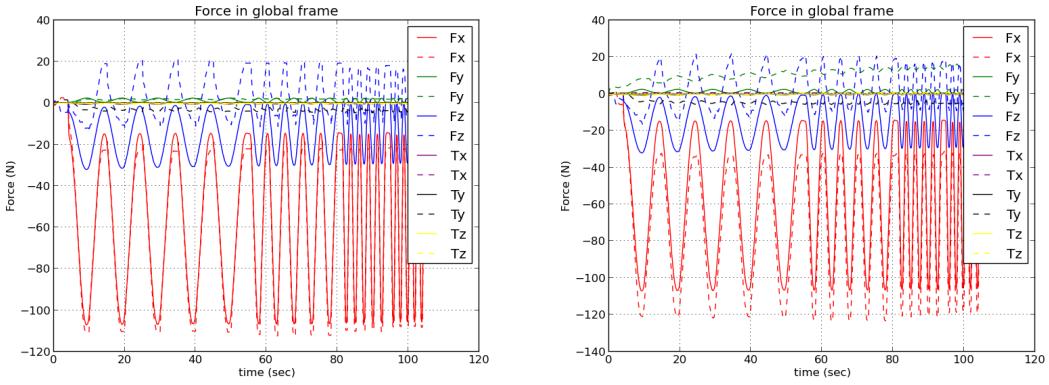
rithm (i.e.: whether it is used for offline computation or real time computation). The snippet code for the developed algorithm is available in the appendix (see Fig. A.5). The codes are developed in Python language.

Chapter 8

Model Validations

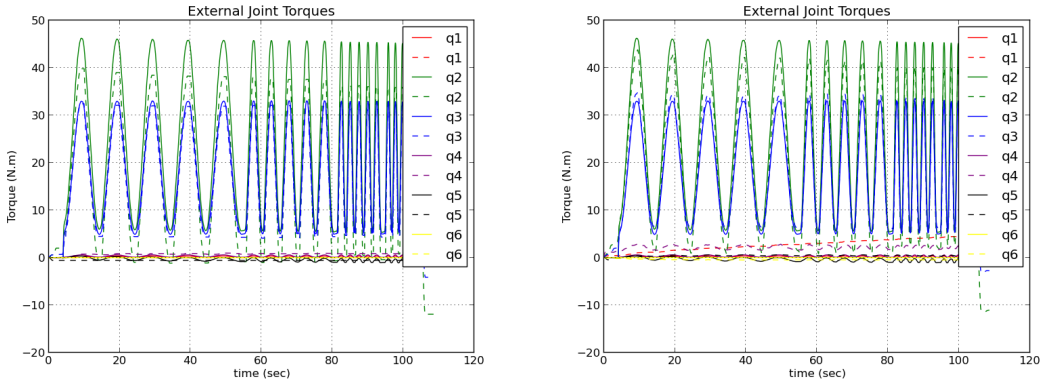
8.1 Validation of One-Stage Experiment Results

After identifying the paramaters for all the required equation, developed algorithm model now was built to estimate the contact force. Hence, there are two versions of algortithm, one that use Dahl and another one that use coulomb and viscous. The estimated force is then compared to real force from F/T sensor for validation. The results are shown in Fig. 8.1.



(a) Using static model (b) Using Dahl model

Figure 8.1: Validation result of estimated force. (- - : estimated output, – : real output)



(a) Using static model

(b) Using Dahl model

Figure 8.2: External joint torques estimation of Fig. 8.1 (- - : estimated output, – : real output)

In Fig. 8.2b it is quite clear that Dahl algorithm gives a good estimation for second and third joint. However, it is bad when estimating the first joint. This is more likely due to two reasons. First, the initial state of z might be incorrect. For every arm position, it is supposed to have specific state of z, however as we lack of knowledge of this value, it was only calculated using some basic assumption. The second reason is due to stability of motor currents. Since change of internal state is a function of rate of motor currents, unstable motor currents will drift the value of z. However, seeing that the value of first joint seems to be drifted over the time, hence it is more likely that the second reason is the main problem. Due to these problems, it is then decided to leave the Dahl model for the rest of the progress.

On the other hand, the results using coulomb and viscous can be seen in Fig. 8.1a and Fig. 8.2a. The estimated force in x-axis is quite satisfactory and this is the main force that acting on the robot. However, the force estimation of other axis is not as good as the first one. This is because of the error estimation of external joint torques that leads to the force. The comparison of external joint torques estimation can be seen in Fig. 8.2. The root mean square error values of estimated contact force and torque using this model are presented in Table 8.1.

On some aspects, especially for torques it has large errors. This is because there is no contact torques introduced, hence the value from force sensor is always near 0.

	RMSE	max-min	RMSE / (max-min)(%)
Force x	5.231976	107.774842	4.854543
Force y	0.942644	2.391211	39.421205
Force z	18.686503	32.631091	57.265945
Torque x	0.116993	0.048434	241.550320
Torque y	3.542738	0.879940	402.611434
Torque z	0.350672	0.192694	181.983440

Table 8.1: Root mean square value of estimated contact force using static friction

While it gives a reasonable results for both model, this is partially because the old data was being used for validation, hence it is quite obvious that it will give a good estimation. Validation with new data will be required to really verify the developed model. However, since the methodologies were changed in the middle, these results are not analysed further.

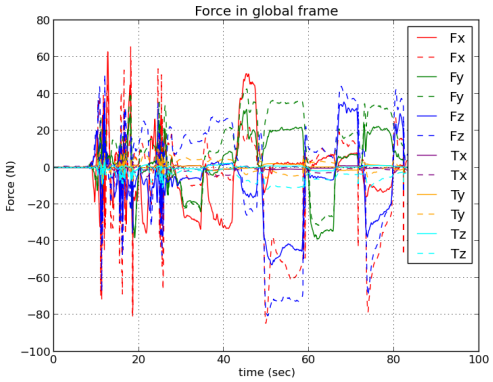
8.2 Validation of Two-Stage Experiment Results

New setups and data that differ from identification data were performed and collected. The algorithm in section 7.4 was then used to estimate the contact force in offline mode. The results were then compared with the real contact force measured from F/T sensor.

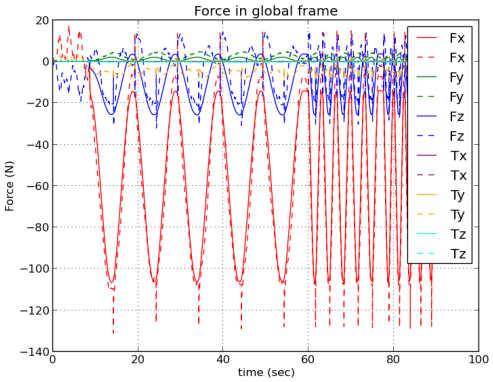
Three types of contact was performed for the validation. First is static contact force, where the robot was fixed and external force was then introduced to the robot. Second is where the robot was pushing an object in a continuous sinusoidal motion. And the last contact type is where the robot pushed an object with a

step function.

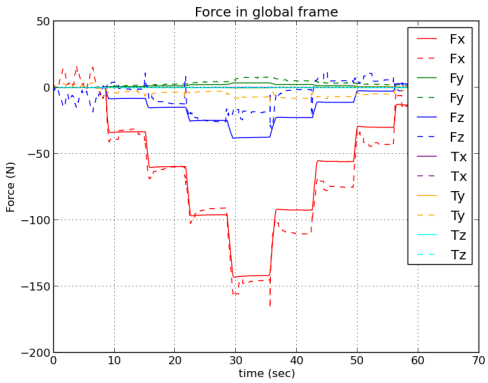
The results could be seen in Fig. 8.3 for all types of contact. The continuous lines are the measurement from F/T sensor while the strip lines represent the predicted value using the developed algorithm. Results for individual axis can be examined in the appendix (see Fig. A.6 - Fig. A.8). The root-mean-square from these data are also given in the next table.



(a) Static contact force



(b) Sinusoidal contact force



(c) Step function contact force

Figure 8.3: Verification of developed model for various contact type(- - : estimated output, – : real output)

Root Mean Square of Various Contact Type			
	static (%)	sinusoidal (%)	step (%)
Force x	46.201581	7.035855	6.812475
Force y	45.474764	79.921285	61.432838
Force z	26.483134	33.682928	29.129328
Torque x	146.662725	247.210383	136.615923
Torque y	186.067329	683.422297	532.088055
Torque z	408.659309	90.338212	51.192202

Table 8.2: Root mean square value for all types of contact

From the root mean square value, the estimation gives horrible results for some of the axes. There are possible explanations for this: 1) The range of external force/torque introduced is small 2) Poor denso gain identification due to the effect of the deadzone and 3) Poor friction identification. For example, torque z in static contact force has a large error that possibly due to the large deadzone of the sixth joint (see Fig. A.3f). On the other hand, torque x and y have a bad estimation during the sinusoidal and step contact force since for this experiment only external force x and z are introduced, thus making the range value is small. In contrast, the model can give a much better estimation for axis with large range of external torque/force. For instance, force x during sinusoidal and step motion has a good estimation of the force.

Hence, it can be concluded that the model will work better when contact force/torque is large and it will perform poorly for small or no contact force.

Chapter 9

Contact Force Control

Since estimation algorithm has been created, force control of the arm for assembly tasks can be performed now. Instead of using F/T sensor to control the contact force, motor torques are used to control the contact force through the model. However, readings of F/T sensor are still captured to know the performance of the estimated model.

9.1 Controller

There are two controllers that are used during this project to control the robot, they are force controller and position controller. Both controllers are PD controller with adjustable gain of P and D elements.

9.1.1 Force Controller

force controller is a controller that use force as the input and feedback to drive the arm position. The figure below describes the block diagram of the force controller used. First a reference force is set, which after that the arm will move because of the difference between desired force and feedback force. The feedback force used will be calculated from the model prepared using motor torques and

joint velocity. Hence, the algorithm will be used in this force controller architecture.

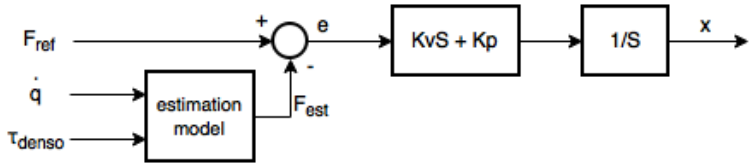


Figure 9.1: Force controller using motor torques and joint velocities

9.1.2 Position Controller

Position controller is a controller that use position as the input and feedback to drive the arm position. It is more straightforward since it is position to position relation. It does not require force element in this controller and hence the algorithm created is not used in this function. See the figure below for the simple block diagram.

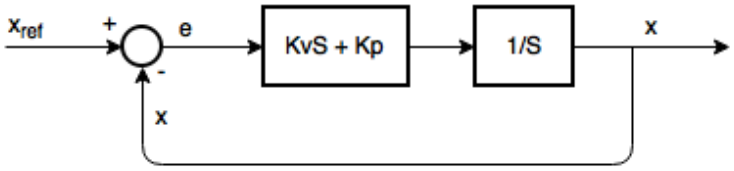


Figure 9.2: Position controller

9.2 Pin Insertion Task

One assembly task that will be performed is the pin insertion task. Basically the robot has to search the hole in an object and then insert the pin into the hole. The general steps to perform this operation are:

1) Contact detection using position controller. The arm end-effector will move in z-axis based on position control. Force in z-axis (F_z) will be checked to determine when a contact with the plane is made. A contact is said to be detected

once F_z has exceed the threshold value, which in this task is set to be 20N.

2) Hole detection using hybrid controller. After the robot has made a contact with the plane, the gripper holding the pin will move in the contact plane (x-y plane) using position control principle to search for the hole. The movement follows a spiral equation, where:

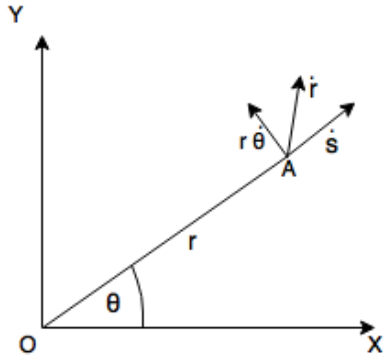


Figure 9.3: Spiral movement vector

$$\vec{\dot{r}} = \dot{s}\left(\left(cos\left(\omega t\right) - \omega t * sin\left(\omega t\right)\right) i + \left(sin\left(\omega t\right) + \omega t * cos\left(\omega t\right)\right) j\right) \tag{9.1}$$

While it searches for the hole, the robot will also have to maintain the contact force F_z with the plane, hence the manipulator will also need to move in z-axis. This movement is controlled by force controller. Thus, the gripper movement is controlled by both position and force controller and so it is called as hybrid controller. The contact force is maintained to be at 15N. To determine the hole existance, force in x and y-axis (F_x, F_y) are used as the indicator. It is set such that if the resultant force of (F_x, F_y) is greater than 60N for more than half second then the hole is detected.

3) Pin insertion using hybrid controller. When the hole has been detected, the final step is for the manipulator to insert the pin into the hole. To do this force controller is used to position the arm in x-y plane so that the pin will stay inside the hole while position controller is used to insert the pin in z-axis. The

force controller in x-y plane is set to stay at 15N for each axis. The pin is said to be successfully inserted if F_z is bigger than 80N.

All these steps will be executed using motor torques as the replacement of the F/T sensor. However since data from F/T sensor are also recorded, the force control performance using motor torques can be compared with the F/T sensor.

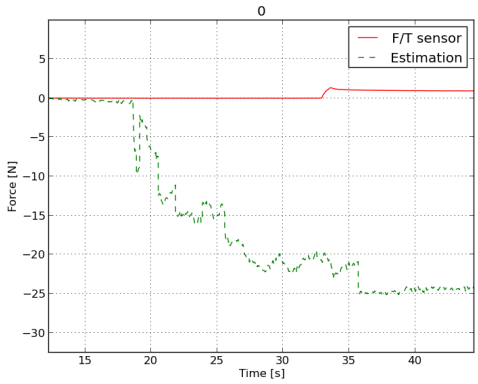
After several tuning to the PD gain in the controller, the manipulator was able to successfully insert the pin. The performance during the operation will be discussed in the next subsections.

9.2.1 Contact Detection

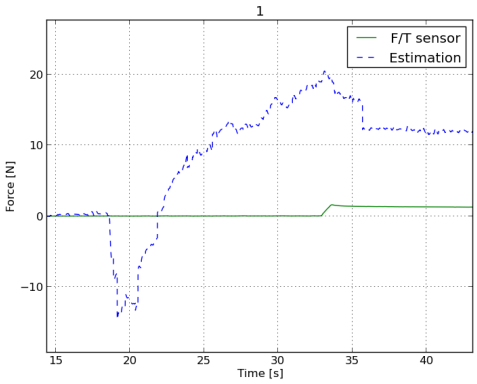
As what has been described in section 7.4, the robot needs to move into the precontact position as close as possible to the plane of interest. In this case, the reference point is taken when the robot is not moving yet. This is to check whether it friction parameters that has been identified is good enough to compensate for the friction. After referencing to this point, the arm moves slowly approaching to the contact area. The force is then estimated using motor torques and joint velocities.

The result of using the motor torques to estimate the end-effector force can be examined in Fig. 9.4. Results from ATI F/T sensor are also attached in the figure for comparison. The results show that the force estimation in x and y are not really reliable. This may indicate that the friction model identified might be off from the true friction such that it still produce some offset for the force estimation. Other reasons for this might be because the results of neglecting some important parameters such as torques for dynamic motion and deadzone problem during the identification.

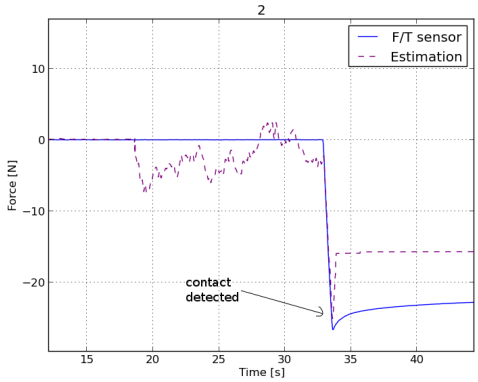
In contrast, force estimation in z direction is much better than the other two axis. While the average of F_z is not zero, it is still below the threshold. There is also a clear difference in F_z value where it easily exceed the threshold when



(a) Force -x



(b) Force -y



(c) Force -z

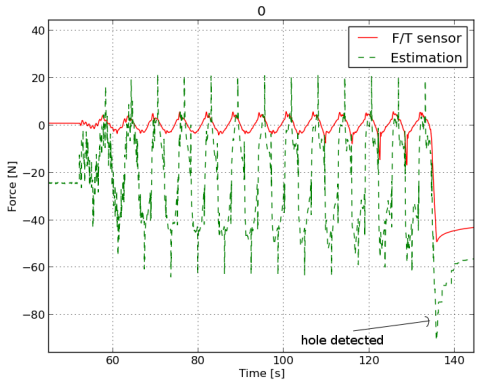
Figure 9.4: Force estimation in contact detection. (- - : estimated output, – : F/T sensor output)

a contact is happening. Furthermore, The value is comparable with the F/T sensor data when there is a contact (error = 32.56%). Hence, it is seen that using the estimation algorithm, contact detection can still be performed without F/T sensor.

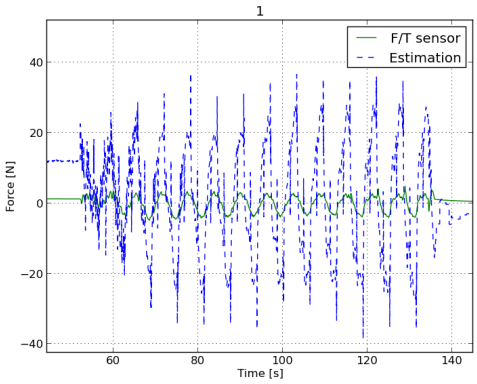
9.2.2 Force Control in Hyrbrid Controller for Hole Detection

The force data for all axis during this process are shown in Fig. 9.5. The continuous line represent the results from F/T sensor while discontinuous line is the computed force without F/T sensor.

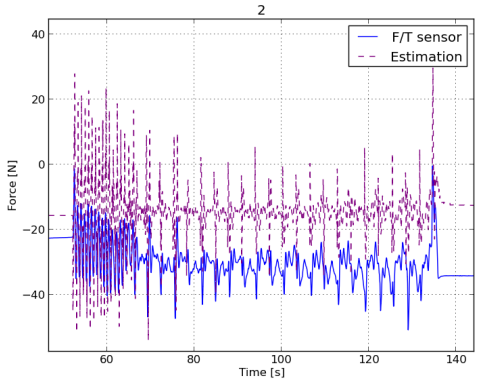
During the early stage of the searching, the force control in z-axis is unstable.



(a) Force -x



(b) Force -y



(c) Force -z

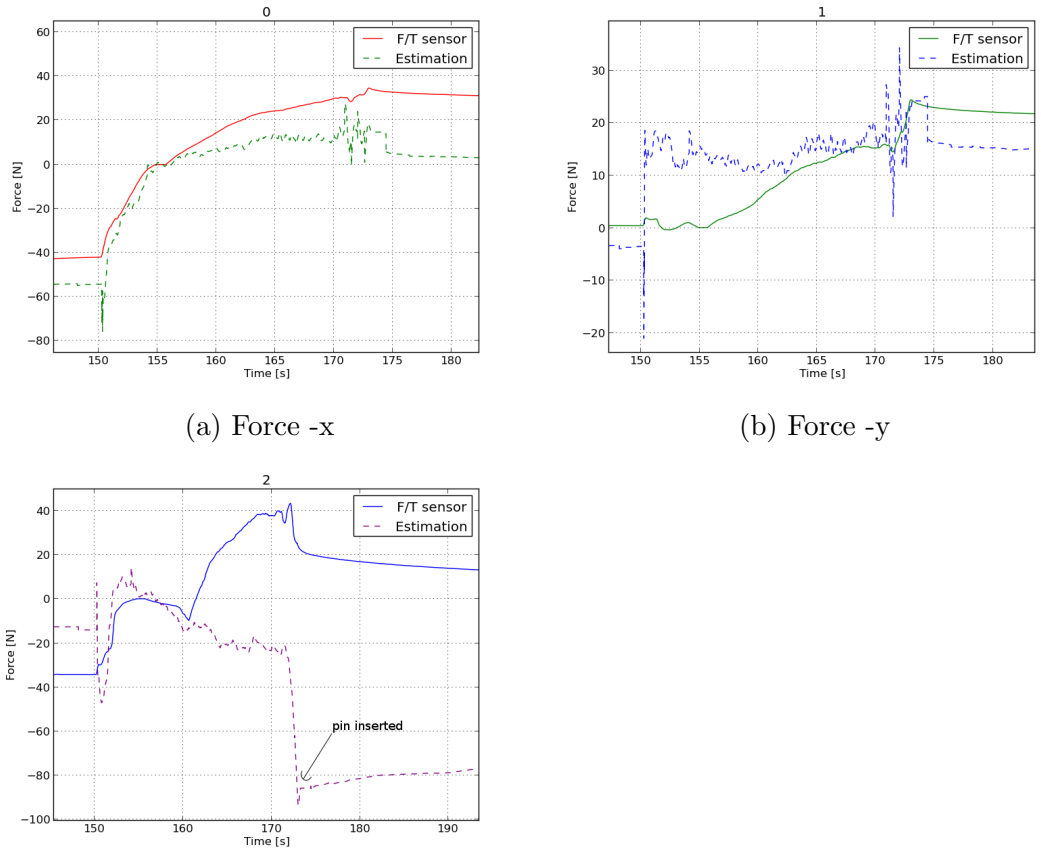
Figure 9.5: Force for all axis during hole detection. (- - : estimated output, – : F/T sensor output)

This is due to the fast change in the joint velocities during the early spiral movement where the circle line is near zero. However after several seconds the force become more stable and oscillate in around reference value which is 15 N.

Comparison with the F/T sensor shows that the result from the model tend to underestimate the force and hence, leading to the actual force greater than computed. The F/T sensor have an average value of -31.15 N (52.45 % error) during the force control. While this performance is not wanted, this is better than overestimating the force. Since in this case it will be guaranted that the actual contact will have to happen first before the robot know about the contact. Thus, in this case the contact with the surface plane is still occuring during the hole searching.

9.2.3 Force Control in Hyrbrid Controller for Pin Insertion

The force data for all axis during this final step are shown in Fig. 9.6. The continuous line represent the results from F/T sensor while discontinuous line is the computed force without F/T sensor.



(a) Force -x

(b) Force -y

(c) Force -z

Figure 9.6: Force for all axis during pin insertion. (- - : estimated output, – : F/T sensor output)

In this process two identical force controller is set to control each x and y position. Apart from the first few seconds where F_y tend to be overestimated, the force controllers give similiar performances with the previous force controller, that is the algorithm tend to underestimate the contact force between pin and the object. Different behaviour is also found in the F_z when the pin has been

inserted. A positive force is observed from F/T sensor while the model estimated the force to be below -80N.

In overall, while the estimated force is still not accurate according to F/T sensor, it gives the same pattern with the F/T sensor with underestimated value. Moreover it has been shown that it can be used as the force controller to control the contact force due to the underestimate behaviour of the model.

Chapter 10

Conclusion and Future Works

10.1 Conclusion

This project studied about contact force control using only motor currents / torques. Hence force estimation was the most crucial part in this project. This project specifically studied about the Denso VS-060 arm. Previous works had been reviewed about many advance approaches that had been researched to estimate the contact forces without force sensor. Due to some limitations, not all of the methods could be applied for this project.

A basic dynamic system was used as a fundamental model to estimate the contact force. A friction model was needed in the model and two basic friction were chosen: viscous and Coulomb model (static friction) and Dahl model (dynamic friction). To be able to estimate contact force without force / torque sensor, identification of unknown parameters of the system was needed. In general, two parameters were to be identified, they are: friction and gain parameter from denso motor currents / torques to real value of joint torque from motor.

Before continue with experiments, problem regarding motor currents were present and hence it had to be solved first. The problem was that the motor currents would always give positive value, thus a way to determine the sign of

the currents was needed. A solution to this was to use motor torques as it also had the sign value along with the number. Hence, there is no need to use motor currents anymore as motor torques are now available.

Thereafter, experiments were performed. The model identification then had been done and verified in the early stage. However, as the experiment was not structurized and the results were difficult to be identified, it was then retried using more structured and easier setup which was two-stage experiment. All necessary parameters were then recalculated.

Once all of the parameters had been identified, the algorithm to estimate the contact force was created. The model would estimate the force based on only motor torques and joint velocities. The computation also assumed quasistatic model for simplicity.

The next step was to validate the developed model estimation with the F/T sensor using new data. Based on data, it was believed that the results were more reasonable once contact forces had been introduced.

Finally, a force control based on the algorithm was constructed, Then a small assembly task which is pin insertion was performed using this force controller to control the manipulator. After several tuning of PD gain the manipulator was able to successfuly insert the pin into the hole.

10.2 Future Works

The future works for this project may include: troubleshoot of the current problem of the OpenRave or perform identification of robot inertial parameters and improve the model to have a better estimation of contact force.

References

Bona, B., & Indri, M. (2005). Friction Compensation in Robotics: an Overview. In *Proc. of 44th ieee conference on decision and control, and the european control conference 2005* (p. 4360-4367).

Damme, M. V., Beyl, P., Vanderborght, B., Grosu, V., Ham, R. V., Vanderniepen, I., et al. (2011). Estimating Robot End-Effector Force from Noisy Actuator Torque Measurements. In *Proc. of ieee international conference on robotics and automation* (p. 1108-1113).

Kermani, M., Wong, M., Patel, R., Moallem, M., & Ostojic, M. (2004). Friction compensation in low and high-reversal-velocity manipulators. In *Proc. icra '04, 2004 ieee int. conf. on robotics and automation* (p. 4320-4325).

Ohishi, K., Miyazaki, M., Fujita, M., & Ogino, Y. (1991). H observer based force control without force sensor . In *Proc. of industrial electronics, control and instrumentation* (p. 1049-1054).

Prete, A. D., Mansard, N., Ponce, O. E. R., Stasse, O., & Nori, F. (2015). Implementing Torque Control with High-Ratio Gear Boxes and without Joint-Torque Sensors.

Stolt, A., Linderoth, M., Robertsson, A., & Johansson, R. (2012). Force Controlled Robotic Assembly without a Force Sensor. In *Proc. of ieee international conference on robotics and automation* (p. 1538-1543).

Suarez-Ruiz, F., & Pham, Q.-C. (2015). A Framework for Fine Robotic Assembly.

Vuong, N. D., & Jr., M. H. A. (2009). Dynamic Model Identification for Industrial Robots. *Acta Polytechnica Hungarica*, *6*(5), 51-68.

Wahrburg, A., Morara, E., Cesari, G., Matthias, B., & Ding, H. (2015). Cartesian Contact Force Estimation for Robotic Manipulators using Kalman Filters and Generalized Momentum. In *Proc. of ieee international conference on automation science and engineering* (p. 1230-1235).

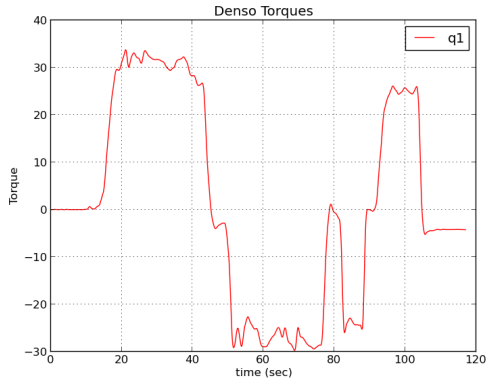
Wahrburg, A., Zeiss, S., Matthias, B., & Ding, H. (2014). Contact Force Estimation for Robotic Assembly using Motor Torques. In *Proc. of ieee international conference on automation science and engineering* (p. 1252-1257).

Appendix

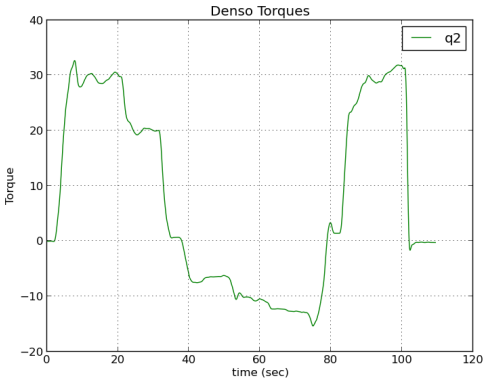
Two-stage Experiment Results

High torque Collection Data

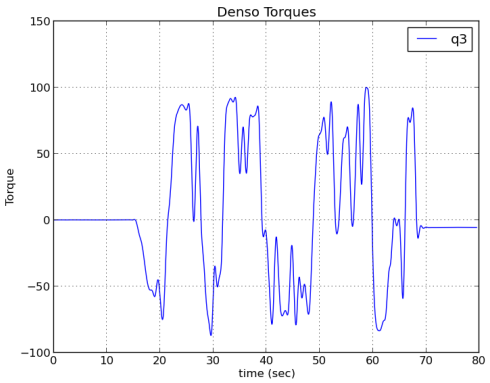
Figure A.1: Denso torque during high torque experiment



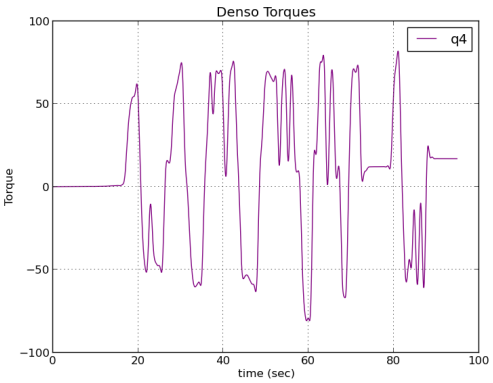
(a) First Joint



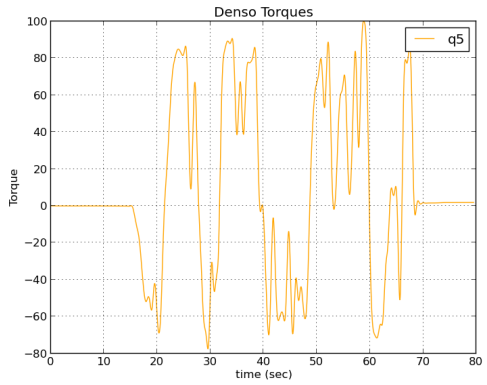
(b) Second Joint



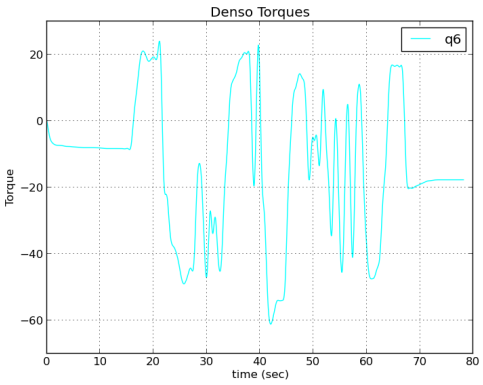
(c) Third Joint



(d) Fourth Joint

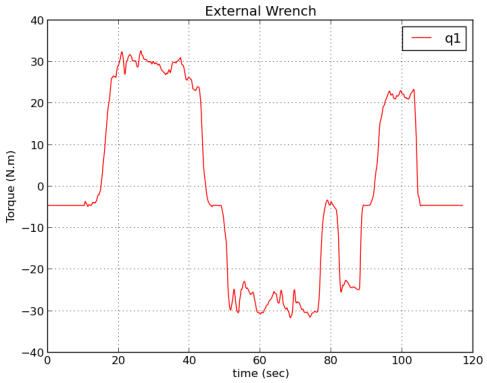


(e) Fifth Joint

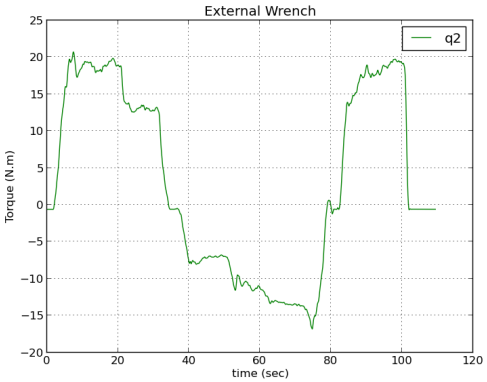


(f) Sixth Joint

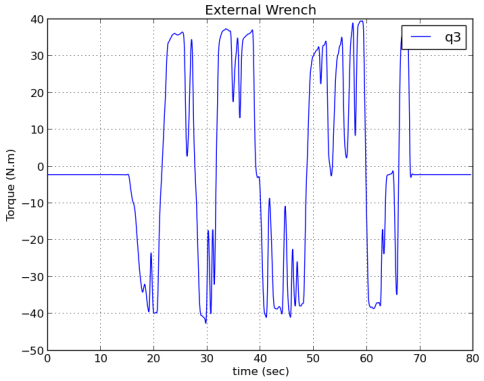
Figure A.2: External torques during high torque experiment



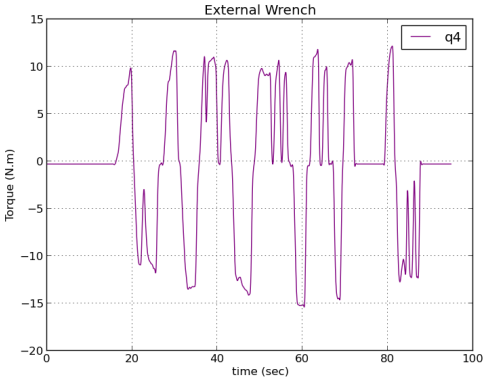
(a) First Joint



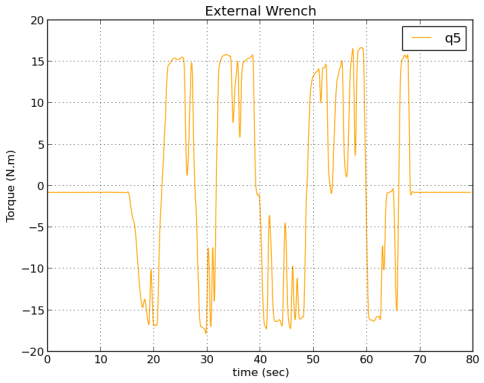
(b) Second Joint



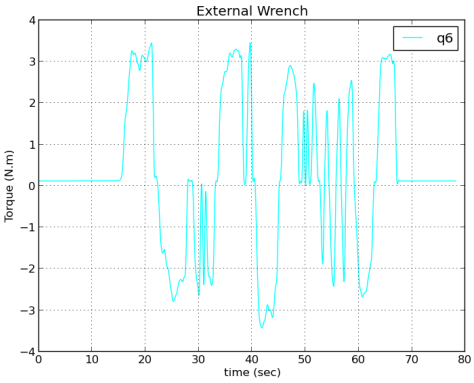
(c) Third Joint



(d) Fourth Joint

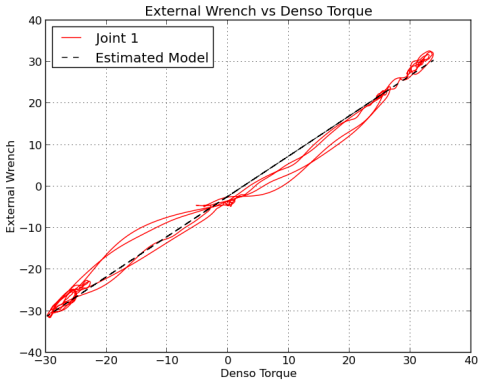


(e) Fifth Joint

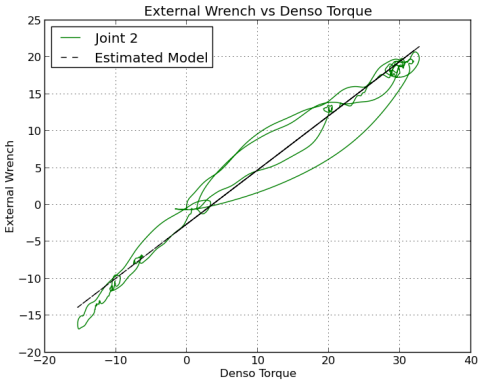


(f) Sixth Joint

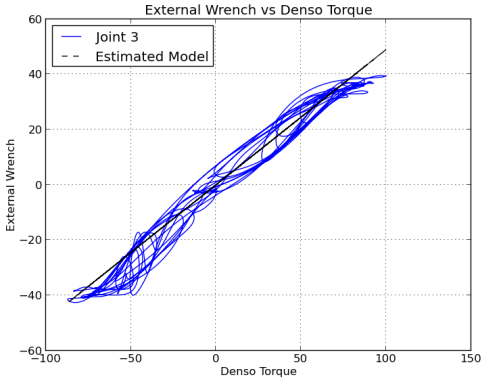
Figure A.3: Denso Gain Identification



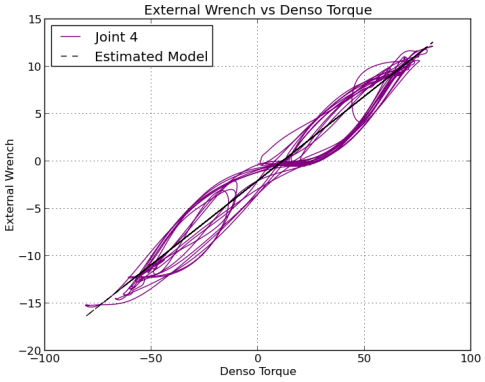
(a) First Joint



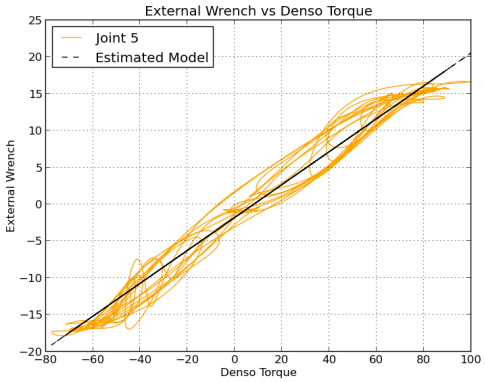
(b) Second Joint



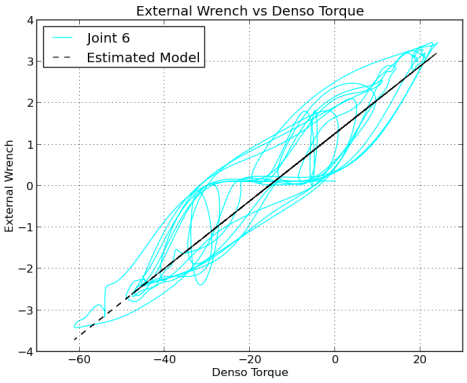
(c) Third Joint



(d) Fourth Joint



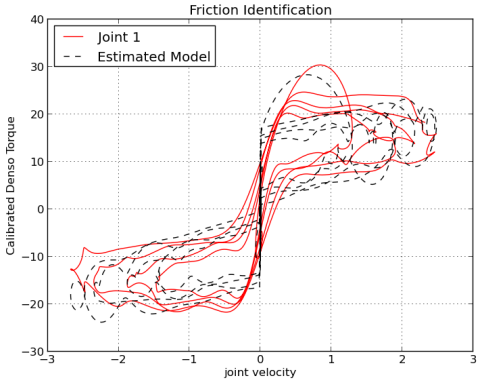
(e) Fifth Joint



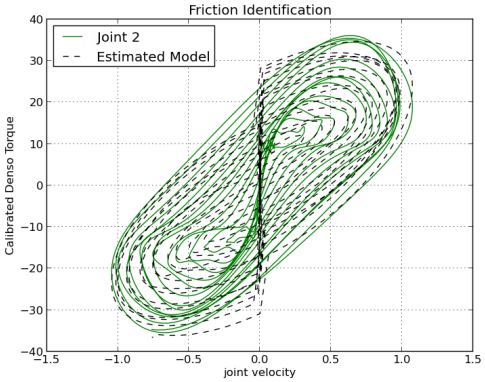
(f) Sixth Joint

High Velocity Collection Data

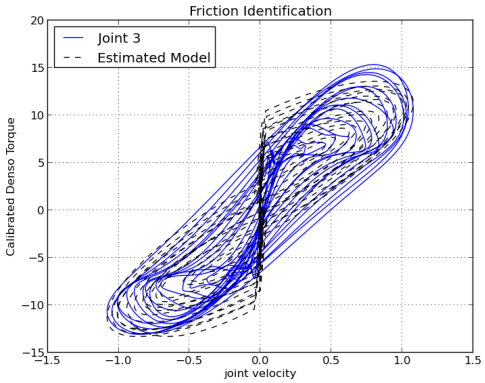
Figure A.4: Friction Identification



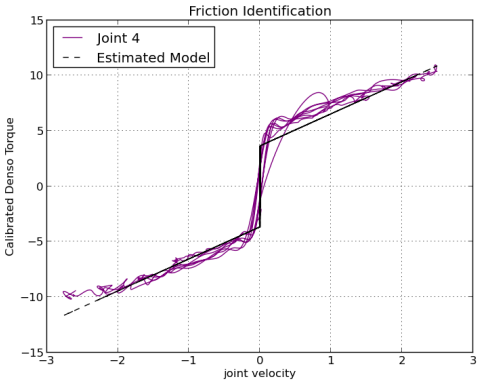
(a) First Joint



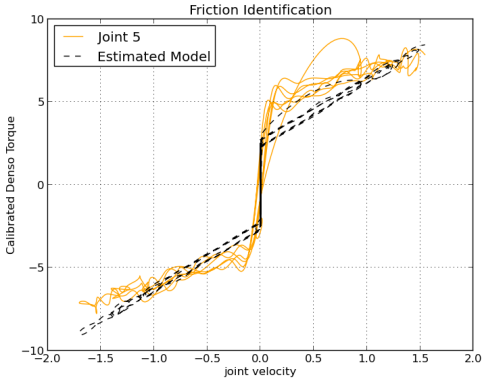
(b) Second Joint



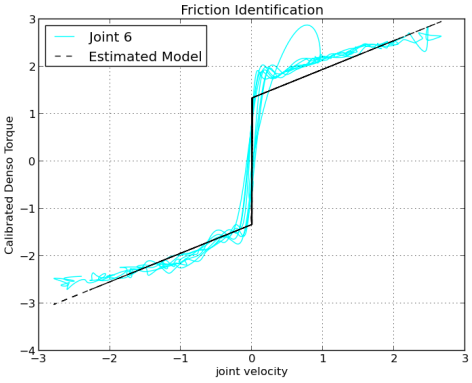
(c) Third Joint



(d) Fourth Joint



(e) Fifth Joint



(f) Sixth Joint

Contact Force Estimation Algorithm

Figure A.5: Contact Force Estimation Algorithm

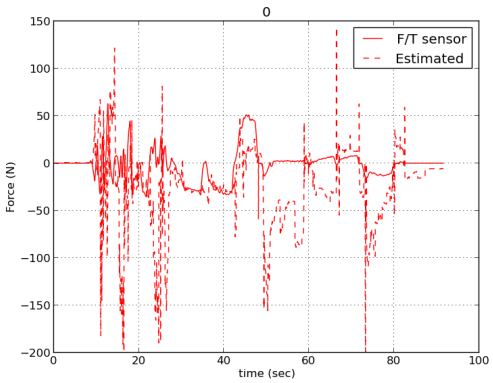
```
class ForceEstimation(object):
  def __init__(self , rave, config, js_rate = 250.0):
    # Initialization
    self.init_precon = True
    self.rave = rave
    self.p = np.array(config)
    self.js_rate = js_rate
    self.J = np.zeros((6,6))
    self.friction = np.zeros(6)
    self.friction_ref = np.zeros(6)
    self.qd_thres = 0.000007
    self.mot_tor_buf = np.zeros(6)
    self.qd_sign_buf = np.zeros(6)

  def EstimateEndEffectorForces(self, q, mot_tor):
    self.qd = np.diff(q, n=1, axis=0) * self.js_rate
    self.est_tor = np.zeros((len(self.qd),6))
    self.est_force = np.zeros((len(self.qd),6))
    if(self.init_precon):
      self.InitializePreContact(self.qd[0,:], mot_tor[0,:])
      self.init_precon = False

    for i,vel in enumerate(self.qd):
      self.UpdateOpenRaveDynamics(q[i,:] , vel)
      [self.est_tor[i,:] , self.est_force[i,:]] = self.EstimateForce(vel, mot_tor[i,:])
    return (self.est_tor , self.est_force)

  def EstimateForce(self, qd , mot_tor):
    est_tor = self.EstimateExternalTorques(qd , mot_tor)
    est_force = np.linalg.solve(self.J.T, np.array(est_tor))
    return (est_tor, est_force)
```

(a) snippet code

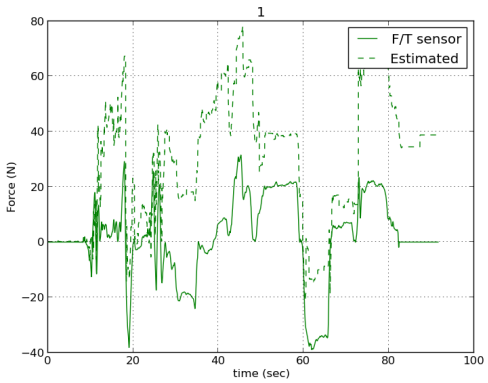
```
  def EstimateExternalTorques(self, qd, mot_tor):
    est_tor = np.zeros(6)
    for joint_num in range(0,6):
      if(abs(qd[joint_num])<self.qd_thres):
        #~ self.friction[joint_num] = 0.
        qd_sign = 0
      else:
        #~ self.friction[joint_num] = self.p[joint_num,1]*np.sign(qd[joint_num]) + self.p[joint_num,2]*qd[joint_num]
        qd_sign = qd[joint_num]

      if (np.sign(self.qd_sign_buf[joint_num])==np.sign(qd_sign)):
        self.friction[joint_num] = self.p[joint_num,1]*np.sign(qd_sign) + self.p[joint_num,2]*qd_sign
        est_tor[joint_num] = self.p[joint_num,0] * (mot_tor[joint_num] - self.tor_ref[joint_num]) -(self.friction[joint_num]-self.friction_ref[joint_num])
        self.mot_tor_buf[joint_num] = mot_tor[joint_num]
        self.qd_sign_buf[joint_num] = qd_sign
      elif(abs(self.p[joint_num,0]*(self.mot_tor_buf[joint_num] - mot_tor[joint_num]))< (0.9*self.p[joint_num,1])):
        self.friction[joint_num] = self.p[joint_num,1]*np.sign(self.qd_sign_buf[joint_num]) + self.p[joint_num,2]*self.qd_sign_buf[joint_num]
        est_tor[joint_num] = self.p[joint_num,0] * (self.mot_tor_buf[joint_num] - self.tor_ref[joint_num]) -(self.friction[joint_num]-self.friction_ref[joint_num])
      else:
        self.friction[joint_num] = self.p[joint_num,1]*np.sign(qd_sign) + self.p[joint_num,2]*qd_sign
        est_tor[joint_num] = self.p[joint_num,0] * (mot_tor[joint_num] - self.tor_ref[joint_num]) -(self.friction[joint_num]-self.friction_ref[joint_num])
        self.mot_tor_buf[joint_num] = mot_tor[joint_num]
        self.qd_sign_buf[joint_num] = qd_sign
    #~ est_tor = np.multiply( self.p.T[0] , (mot_tor - self.tor_ref)) -(self.friction-self.friction_ref)
    return est_tor

  def InitializePreContact(self, qd, mot_tor):
    self.tor_ref = np.array(mot_tor)
    for joint_num in range(0,6):
      if(abs(qd[joint_num])<self.qd_thres):
        self.friction_ref[joint_num] = 0.
        self.qd_sign_buf[joint_num] = 0
      else:
        self.friction_ref[joint_num] = self.p[joint_num,1]*np.sign(qd[joint_num]) + self.p[joint_num,2]*qd[joint_num]
        self.qd_sign_buf[joint_num] = qd[joint_num]
      self.mot_tor_buf[joint_num] = self.tor_ref[joint_num]
    return

  def UpdateOpenRaveDynamics(self, q , qd):
    self.rave.SetDOFValues(q)
    self.J[:3,:] = self.rave.manipulator.CalculateJacobian()
    self.J[3:,:] = self.rave.manipulator.CalculateAngularVelocityJacobian()
```

(b) snippet code (continued)

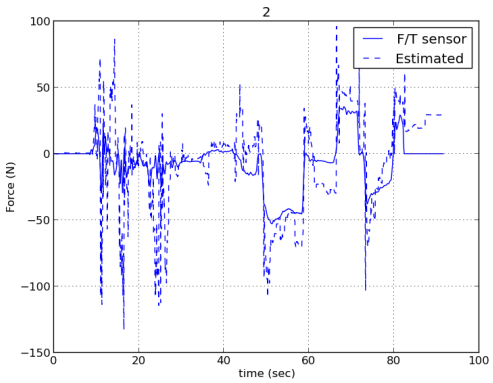
Figure A.6: Static contact force



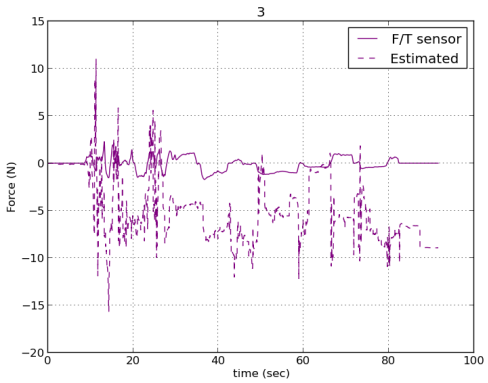
(a) Force x



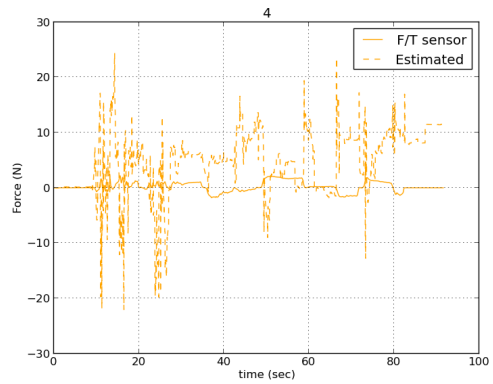
(b) Force y



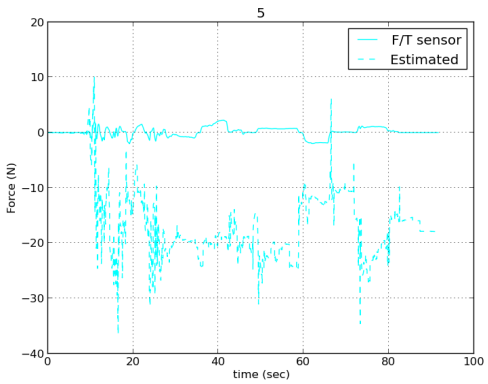
(c) Force z



(d) Torque x

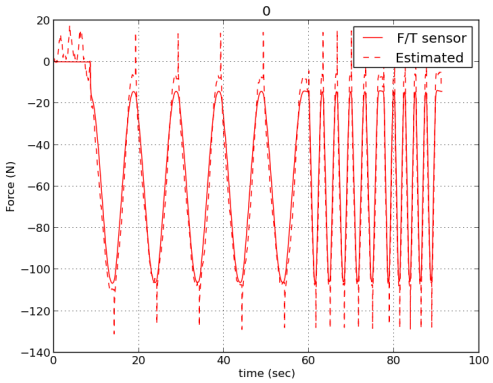


(e) Torque y

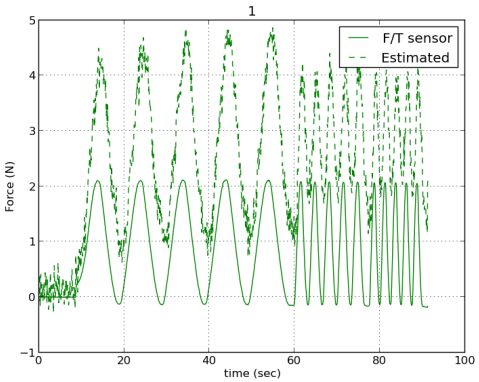


(f) Torque z

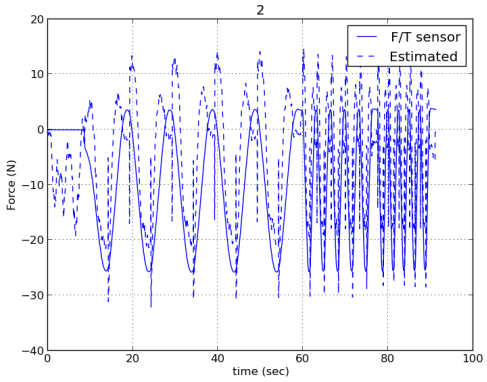
Figure A.7: Sinusoidal contact force



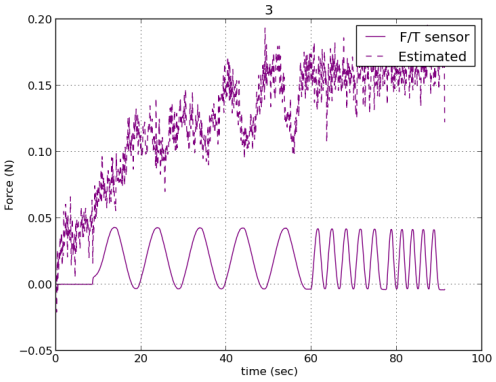
(a) Force x



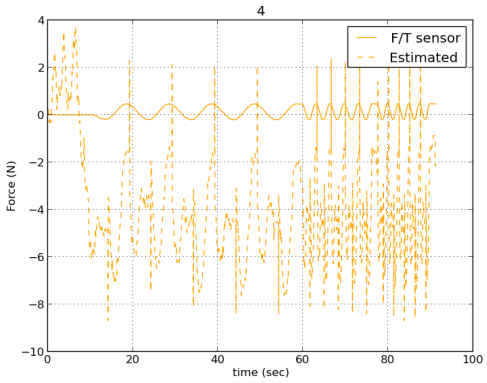
(b) Force y



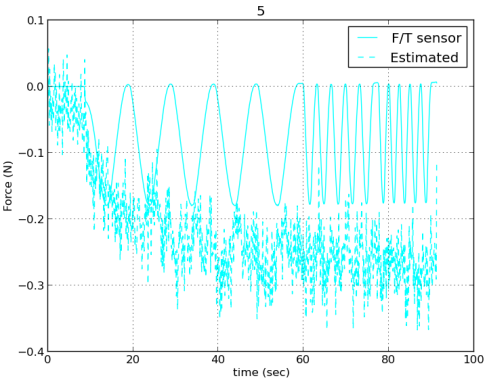
(c) Force z



(d) Torque x

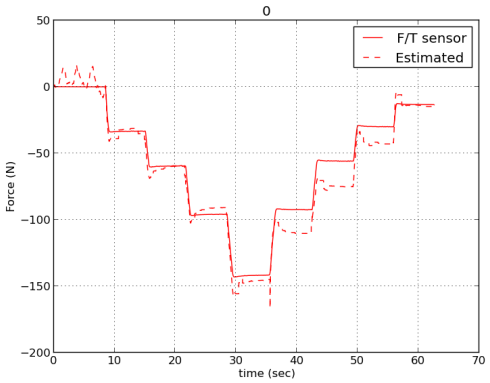


(e) Torque y

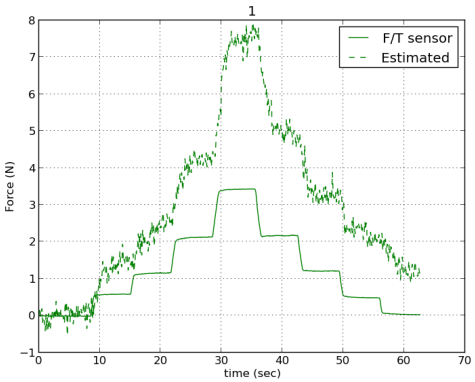


(f) Torque z

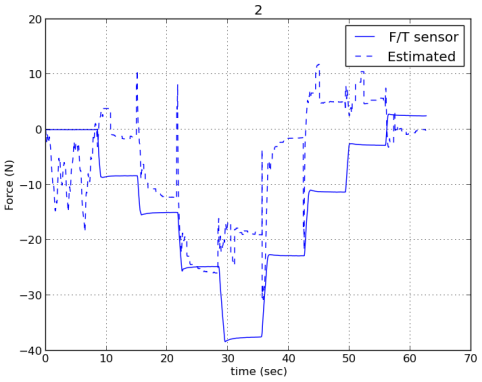
Figure A.8: Step function contact force



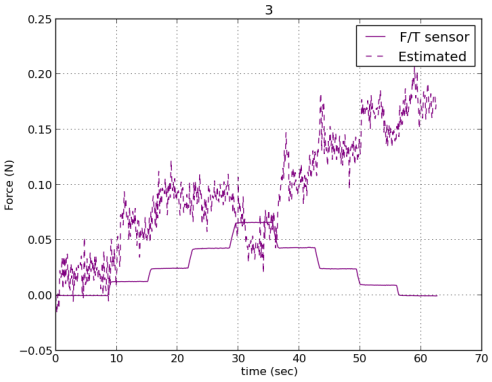
(a) Force x



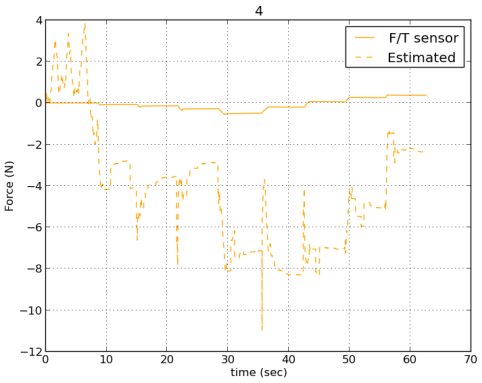
(b) Force y



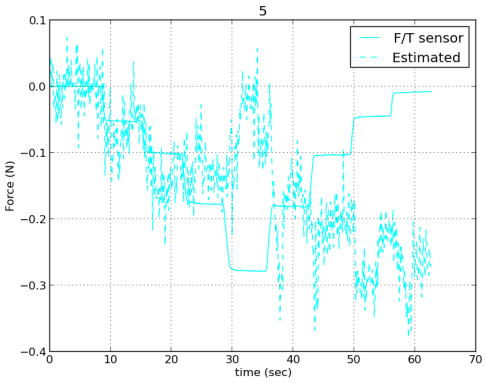
(c) Force z



(d) Torque x



(e) Torque y



(f) Torque z