ISSN 0280–5316
ISRN LUTFD2/TFRT--5768--SE

Modeling and Grey-box Identification of a Robot Manipulator

Luis Rodriguez Blanco

Department of Automatic Control
Lund Institute of Technology
June 2006

Department of Automatic Control
Lund University
Box 118
SE-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
June 2006

Document Number
ISRN LUTFD2/TFRT--5768--SE

Author(s)
Luis Rodriguez Blanco

Supervisor
Rolf Johansson and Anders Robertsson
at Automatic Control in Lund, Sweden

Sponsoring organisation

Title and subtitle
Modeling and Grey-box Identification of a Robot Manipulator
(Modellering och grey-box-identifiering av robotmanipulator)
(Modelado e identificacion en "caja gris" de un robot manipulador)

Abstract

Modeling and parameter identification of the main three links of an industrial robot of the type ABB IRB 2000 is considered. This Master Thesis has been done at the Department of Automatic and Control, Lund Institute of Technology working with a real robot located in the Robotics Laboratory. The model has been built using the simulation software called Dymola (Dynamic Modeling Laboratory), and the parameter identification has been performed with some tools included in the same software.

The procedures requested in order to carry out the modeling and the parameter calibration are explained in detail, and finally a short description about the control design in Dymola is done.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
English

Number of pages
38

Recipient's notes

Security classification

Contents

1. Introduction

The purpose of this work is developing a grey box model of a three link robot manipulator. Next step will be to improve its behaviour by means of design a new controller.

An *ABB IRB-2000* will be used, which is located in the Robotics Laboratory of the Department of Automatic Control of Lund Institute of Technology. As it was said before, only the main three links will be taken into account, considering the third link length as the sum of the upper arm and the wrist stretched ones.

In order to do that, the *Dymola* simulation software[1] [1] will be used. This software is based on *Modelica* language [2], and permits to build models easily using Drag and Drop, and simulate them as well as other many possibilities that will be seen later.

After a short description of the robot; its physical characteristics and its operation modes, it will see how to work with a set of tools included in the used software in order to be able to estimate the parameters of the grey box model.

The modeling and the experiments performed to do that will be explained with more detail, and after that it will see how to improve the robot behaviour by means of a new controller.

Finally, some conclusions and optional ways of continuing this work will be exposed.

[1] Dymola Version 6 was used

2. Robot description and control

2.1 The IRB-2000

The robot is formed by a base attached to the floor, where the engine unit and the gear box of the first joint are included. This degree of freedom lets the first link move with respect a vertical axis.

Above that, it has the first link, where the engine units and gear boxes of the second and third link are.

The second joint is directly actuated by the gear box, and allows the second joint to move with respect to an horizontal axis. That is, the robot moves forward or backward.

However, the third link is actuated by mean of a *gravity compensation arm*, which is connected between the gear box in the second link, and the third link. This is the reason because of when the second joint is moving, and not the third, this link stays horizontal with the same torque applied, since the third engine is stopped. It is just hanging the gravity effect of link three.

2.2 The Control

Because of security reasons, it is not possible to do the parameter estimation in an usual way, it means, in open loop. Therefore, we will implement the real controller in the model, and the estimation will be done in close loop. The controller consists of two *cascade PID's*, as we can see in [3].

Actually the control is working only with the *proportional effect* in the position loop, and the *proportional* and *integrative effects* in the velocity loop, as it could be read in the *C code*[1].

The Excitation Handler

In order to move one or more joints of the robot, an interface in *rbmatlab* was developed[4]. Thanks to it, it is possible to create an input signal easily, and use it as a position, velocity o torque reference. After that, the interface receive the position signals from de sensors in the robot.

The basic steps to move the robot with the *Excitation Handler* are:

1. Execute "irb2000boot" in a terminal.
2. Wait until the control program starts.*Rbmatlab* can be executed at the same time.
3. Execute "Exc_handler" in *rbmatlab*.
4. Create the trajectory in either *rbmatlab* or *Exc_handler* and put it in the channel which it is gone to be used as a reference.

[1] See Appendix A

5. It is suitable to "plot" the data and ensure that any error exist.
6. Select the signals to be recorded.
7. If it is the first time in the session that the robot is going to be moved, it is necessary to "open" the connection in the menu.
8. Select "define" and wait for "Define done" message in the control menu.
9. Make sure that all the safety switches connected to the doors are off(if not, there is an opened door), the red emergency buttons are released, and switch on the Run-mode button on the cabinet.
10. While pressing the yellow handler, and keeping in its middle position, execute "Start".
11. Once the robot is stopped, execute "Receive" to get the data from the robot.

After that, the received data can be plotted, or saved to be used later. If the save option is chosen, a new *mat* file will be created. When this file is open in *Matlab* four new variables are created; the input data is stored in "EXC_INP" and the output in "EXC_RESP".

The order in which the data, corresponding to the variables selected in the "signal selection" menu in the *Exc_handler*, is saved in the variable, is from the first link to the last one and: "uv, p, u, v" for the "EXC_INP". Only the position output is saved in "EXC_RESP". It means that, for instance, if the signals wanted to save are satured torque, position and speed for links one, two and three, the first three columns in the "EXC_INP" variable will be "uv1", "uv2" and "uv3". Then "p1","p2" and "p3" and finally "v1", "v2" and "v3".

3. Modeling and Parameter Estimation

3.1 Introduction

In this chapter we will see how the model is made using the *Dymola Multibody Library*, and what are the available tools in the same software to calibrate de model parameters and try to get the same response in the robot and in the model, when the same input is used.

The modeling and identification task were not done separately but an initial model was build and tested, and starting with it, new elements were added and other were removed taking care the calibration results, in order to do it more accurate.

3.2 Parameter Estimation Tools

Dymola includes a library called *Design Library*, that is composed by some different functions to perform the parameter estimation. These are: *calibrate, validate, checkCalibrationSensitivity, sweepParameter and sweepTwoParameter.* A manual about how to use them with some examples can be found in [5]. We will summarize briefly the basic options that will be used in the next section.

All the functions included in this library have a similar way of use. Because of that, the *calibrate* function will be taken as a reference, and the differences with the other functions will be explain separately.

Once the desired function has been found in the Dymola tree browser, the "Call Function..." option has to be selected in the "right click menu". Then the main window of the function pops.

If the model that is going to be used has not been compiled before, it will be necessary to select it here (see fig. 3.1).

Next, the file containing the data from the robot has to be specified in the cases menu (see fig. 3.2), as well as the simulation time, and if a calibration

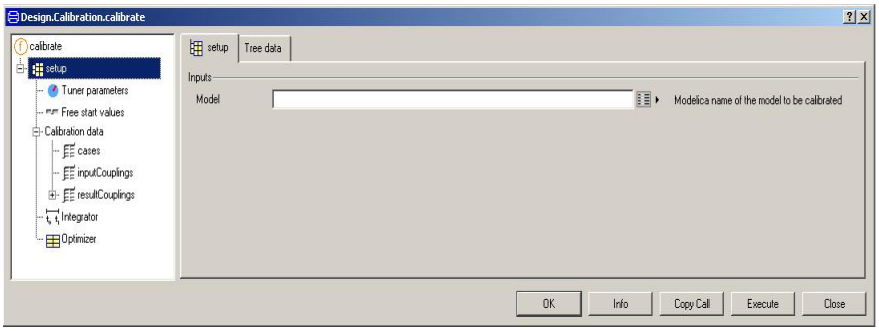

Figure 3.1 Model selection

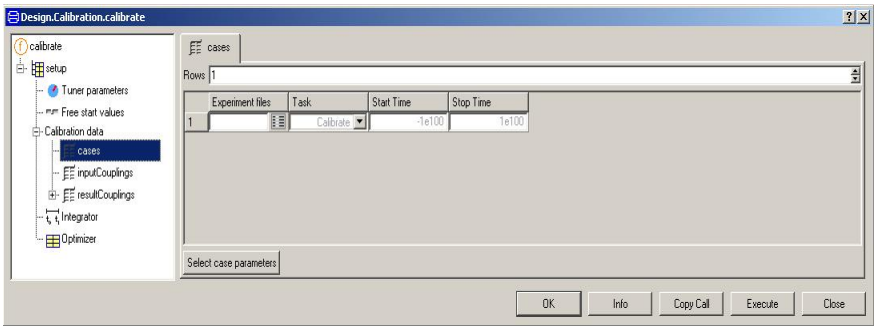

Figure 3.2 Data file selection

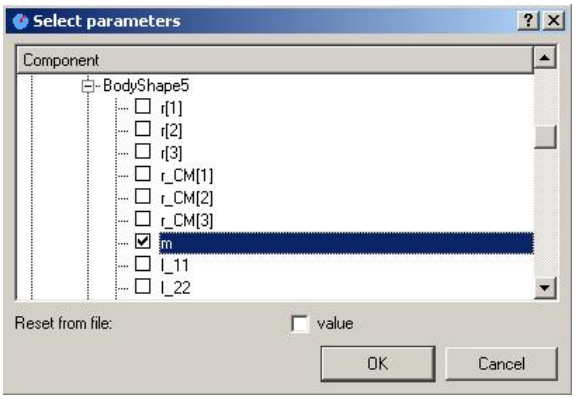

Figure 3.3 Calibration parameters selection

or validation task is going to be executed. This file can be a *mat* or a *cvs* file. The main advantage of *cvs* file is that it can contain text to know what data is put in each column. However it needs to be processed and converted to that kind of file in *Matlab*. A script was developed to do this easily depending on what sets of data was needed. The code can be seen in the B Appendix. In addition to that, *Microsoft Excel* was used to name the columns.

In this menu is also possible to specify the initial conditions to be used in the model, if they are different respect to the ones in the experiment. Unfortunately, it does not work as well as it could be desired. One of the biggest problems is that an initial position has to be specified in degrees, while the rest of the time *Dymola* works in radians. If the fact that the robot works in motor-radians is added, the time that is lost with that simple, but tedious operations does not worth so that this option was slightly used. Moreover, the parameter tuning is being done in close loop so that there would be a lot of variables that need to be initialized to start the simulation with the robot in a particular position.

The calibration parameters are selected in the parameters menu (see fig. 3.3). In the "validate" function it is not necessary to select any parameter, although it is in the "sweepParameter", "sweepTwoParameters" and "checkCalibrationSensitivity".

After the selection is done, the parameters appears in fig. 3.4 where it is possible to assign the initial value which one the calibration will starts, and the limits between the value can be modified. This is important, because if some

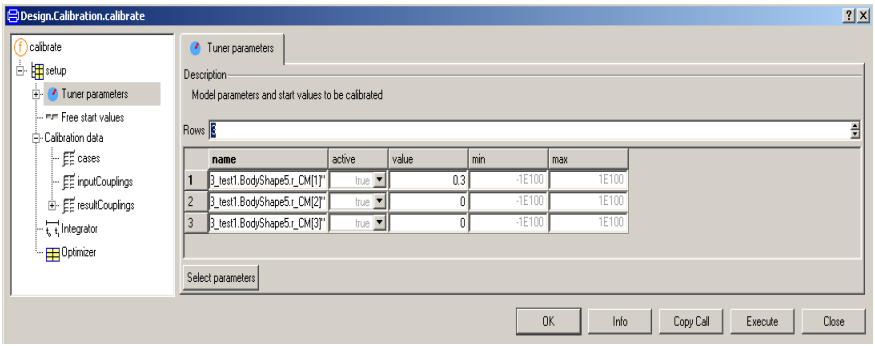

Figure 3.4 Calibration parameters selection

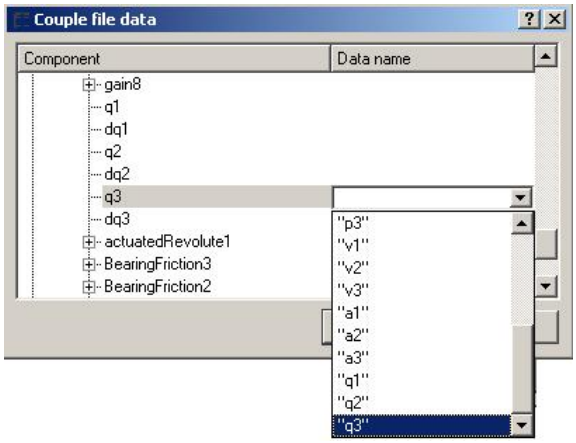

Figure 3.5 Couple data selection

parameters are being tuned at the same time, it is possible that the program assigns wrong values because of they are coupled.

The sections *inputCouplings* and *resultCouplings* are related to the inputs and outputs of the model. It is possible to use the data contained in the *csv* file as an input, when in the model the inputs have been defined as *connectors*. Other possible inputs could be any of the *Sources* blocks, or *Tables*. To use an external input, it is necessary to go to *inputCoupling* and select *Couple file data* and then assign the inputs with the data stored in the file (see fig. 3.5).

In the *resultCouplings* menu is where the calibration criterion is defined. That is, here is where the desired output and the model output are specified. The operation mode is similar to the previous one, except that here it is possible to use different signals to calibrate the model, and a weight column allows the user to give them different weights (see fig. 3.6).

Using the checkCalibrationSensitivity only the function output differs on what has been said before. This function analyses the sensitivities and dependencies between the parameters selected, since it is possible that two or more of them have not a significative influence in the output, or only a linear combination of these parameters can influence the criterion, so that they could not be estimated individually.

Another chance to observe how a parameter influence in some output, the *sweepParameter* and *sweepTwoParameters* functions can be used. These function simulates a model giving different values to one or two parameters,

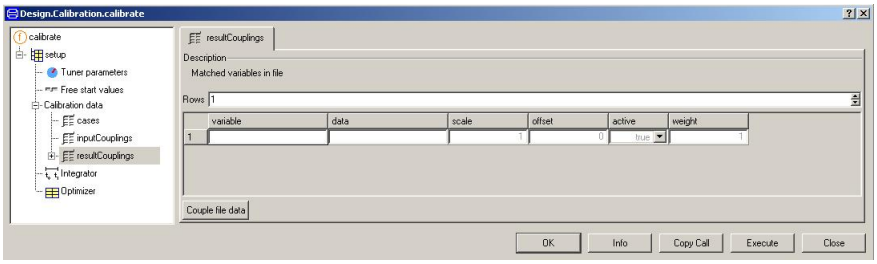

Figure 3.6 resultCoupling menu

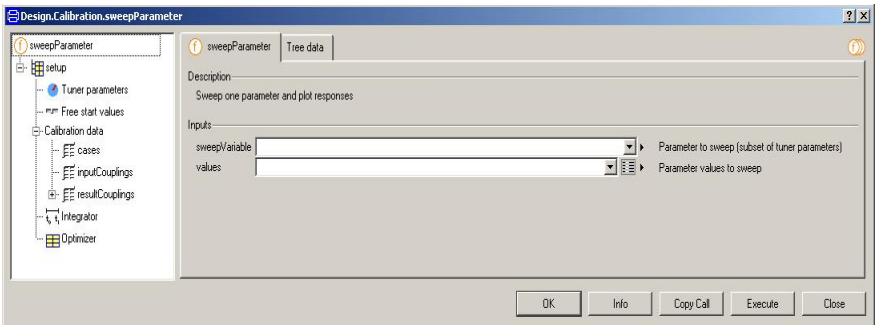

Figure 3.7 sweepParameter menu

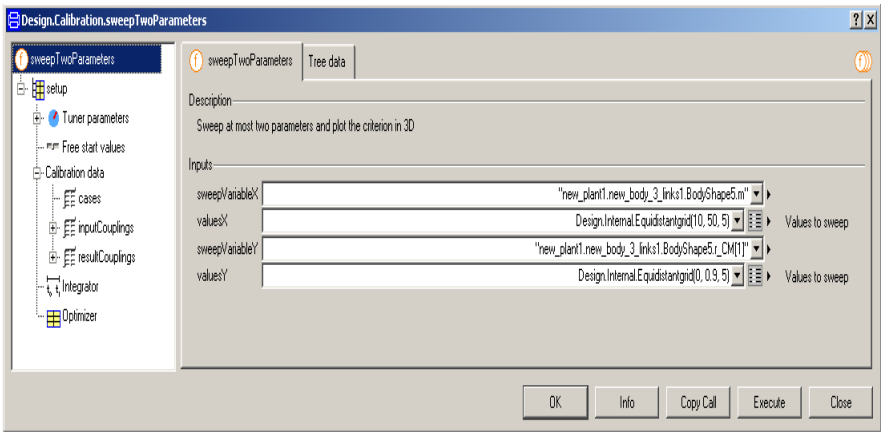

Figure 3.8 sweepTwoParameters menu

and plotting the results in a two or three dimension graph respectively. The "modus operandi" is the same, but it is necessary to define which parameter or parameters will be swept, and which values will be given (see figs. 3.7 and 3.8).

3.3 The Robot Model and Parameter Estimation

The first model of the robot body was built as a simple set of *Body Shapes*[1] connected by means of *Revolute Joints*[2], using the data referred to the arm lengths from [6] (see fig. 3.9). The element between the joint three and the last *Body Shape* was supposed without mass.

In the first model the *gravity compensation arm* was not included, the *actuated revolute joint* for the third link was put between the second and the third link. The position and speed signals were measured in radians with sensors[3].

Next step was the design of the engines and gear boxes model. This class, named *Gears and Engines* connect the controller and the body. Only the *Torque* and the *Gear Box*[4] classes were included here, although in next works, flexibilities or gear loss could be introduced (see fig. 3.10).

At first, the torque signal got from the real robot data was used as the input in order to perform the parameter estimation in open loop, but because of the noise, and the instability of the system, any good result was obtained. Then, the controller was added, and a close loop identification was carried out.

The controller in fig. 3.11 was modeled using the *Continuous* and *Non-*

[1]Included in Modelica.Mechanics.Multibody.Parts
[2]Included in Modelica.Mechanics.Multibody.Joints
[3]Included in Modelica.Mechanics.Rotational.Sensors
[4]Included in Modelica.Mechanics.Rotational

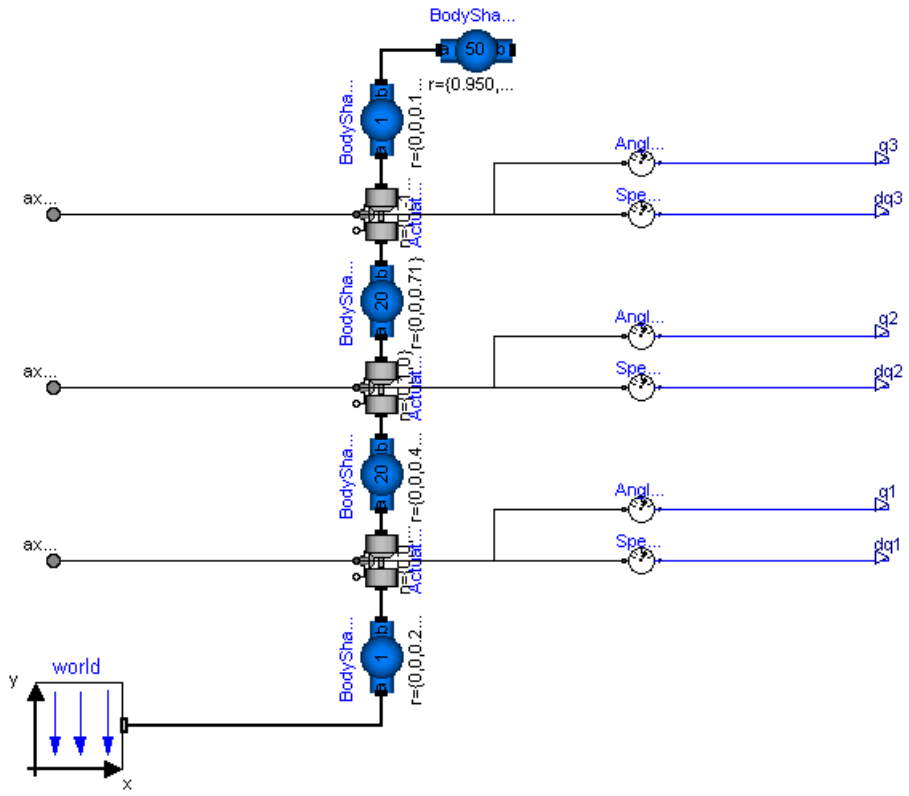

Figure 3.9 First model of the robot body

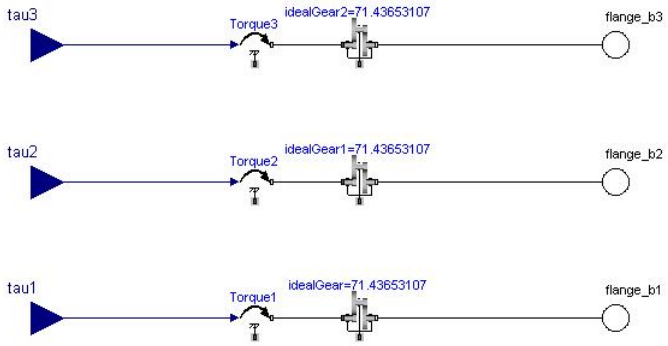

Figure 3.10 Engines and gear boxes model

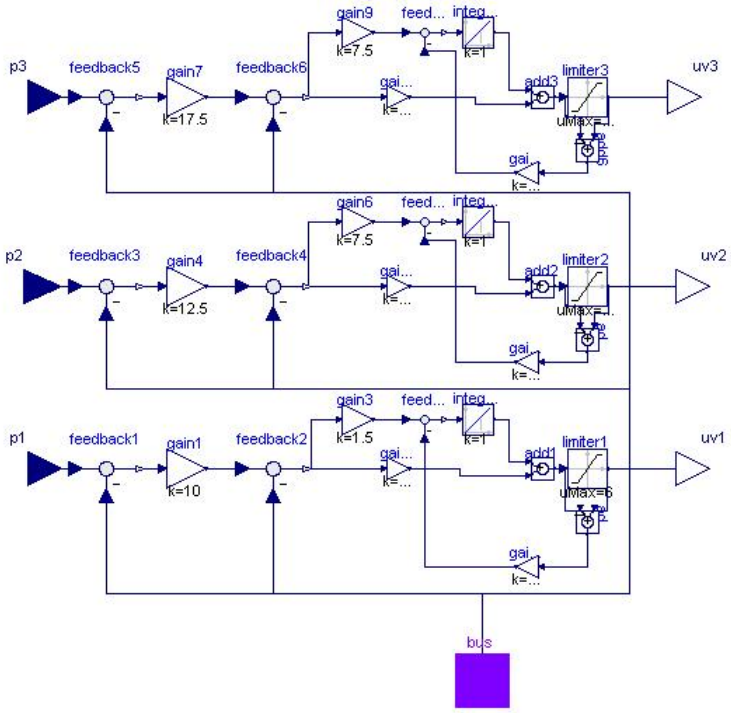

Figure 3.11 Controller

linear Blocks[5] following the scheme presented in [3].

Then, instead of using the torque signal as reference, the position signal was used as the main criterion with the torque one as a secundary reference, and therefore, different robot trajectories would be design to perform the calibration. This choice has been done this way just because, as the identification is carried out in close loop, and the system is supposed to be stable, different values for the robot parameters could be found that make it follows the desired trajectories with very different torque values, and the model would not correspond with the real robot.

[5]Included in Modelica.Blocks

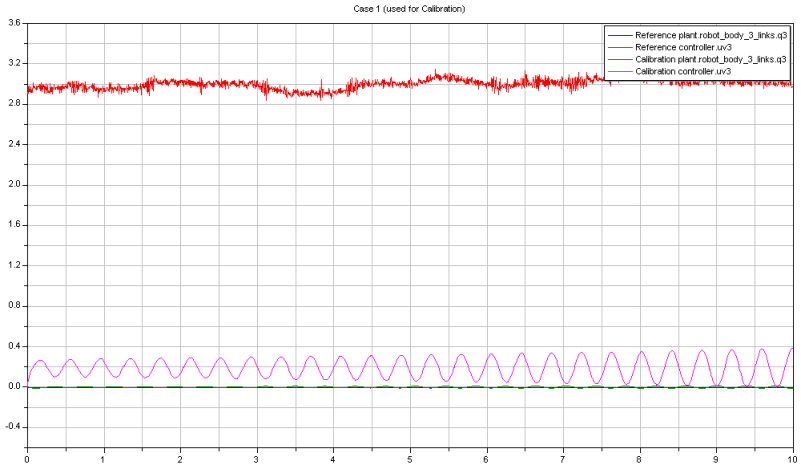

Figure 3.12 Mass estimation for the third link

Some authors suggest several ways to design optimal trajectories, to collect the data and use it to estimate parameters. Some of them [7] get the excitation signal as the result of an optimization problem, in which the dynamic equations and different constraints are involved. Others [8, 9] use multi-sine signals or triangle signals as reference speed. With that trajectories it is possible to estimate several parameter at the same time.

In our case, the trajectories were chosen according with the general dynamic equation of a robot [10], and simple experiments were performed in order to calibrate just few parameters at the same time, or even only one. The more parameter to be tuned at the same time, the more time it takes to the program to finish the estimation, and sometimes the computer hangs.

According to the equation 3.1, of the motion for an open-chain manipulator, there are terms depending on the position, speed, or acceleration valules:

$$M(\theta)\ddot{\theta} + C(\theta,\dot{\theta})\dot{\theta} + N(\theta,\dot{\theta}) = \tau \tag{3.1}$$

Where M is the manipulator inertia matrix, C is the Coriolis matrix and N includes gravity and friction terms. Since the gear ratios are known and quite hight, we can consider the links uncoupled, and we can discard the Coriolis terms.

According with the above-mentioned, we used different static configurations of the robot to get the terms related with the gravity forces. The robot was moved at different velocities to get the bearing friction values, and finally, sinusoidal signals with different amplitude and frequency were applied in each link to fit the inertia tensor values while all the links were moving at the same time.

The first experiment tried to get the mass values by means of keeping the robot without moving. The mass of the third link was the first to be tuned; as it was related in the previous section, the calibration function was used. The initial position of the centers of masses was the middle of the arm. The robot was stopped in its home position, and then the data were stored. Using that, and selecting the mass as the parameter to be calibrated,the position and torque signals as calibration criterion, the funcion was executed. The result of that is shown in fig. 3.12.

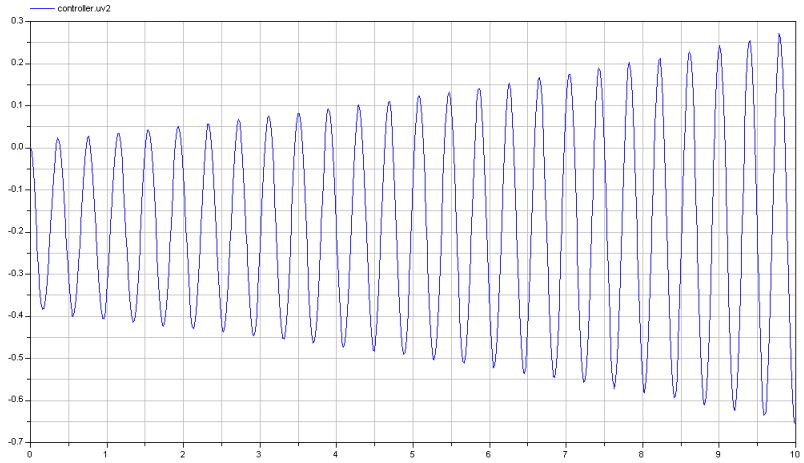

Figure 3.13 Joint 2 torque in the model

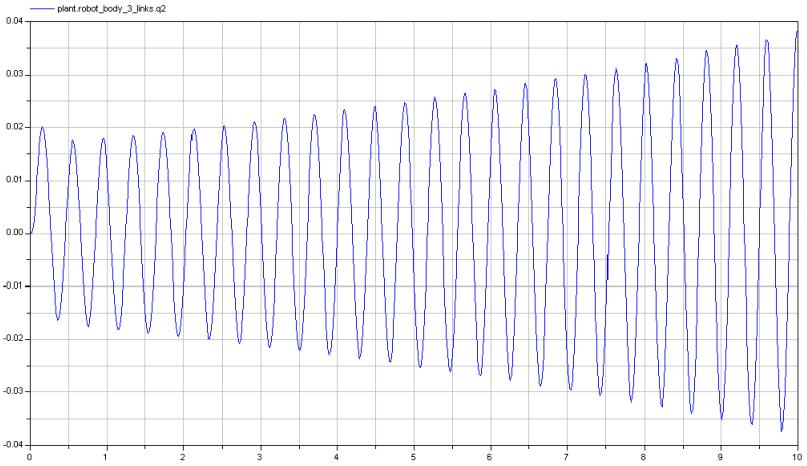

Figure 3.14 Joint 2 position in the model

The calibration returns a value for the mass equal to 2.18 kg. which is impossible. Also, as it is seen in the graph, the torque reference is quite bigger than the torque applied in the model. Checking the values of the torque and position in the second joint (fig. 3.13 and 3.14), it can be appreciated than the system becames almost unstable, even with small values of mass like 1 kg. in the second link. Moreover, it is seen in fig. 3.15 that the torque in the joint two should be constant and near zero.

Even moving the third link with a considerable speed, it is possible to prove that the torque in joint two is very small (see fig. 3.16), and therefore, it looks like the necessary torque applied in the respective joint to move the links two and three were almost independent, what it is not show in the model.

After some similar experiments, keeping the second link in different positions and moving the third, that idea was proved. The torque in the second joint is mainly used to hang the second link, and the conclusion reached was that what was able to supply that was the *gravity compensation arm*. The importance of this part at the back of the robot was noticed and it was included although the complexity of the model increased as well. The first consequence of this, was that the simulation times became longer. Notice that it is nec-

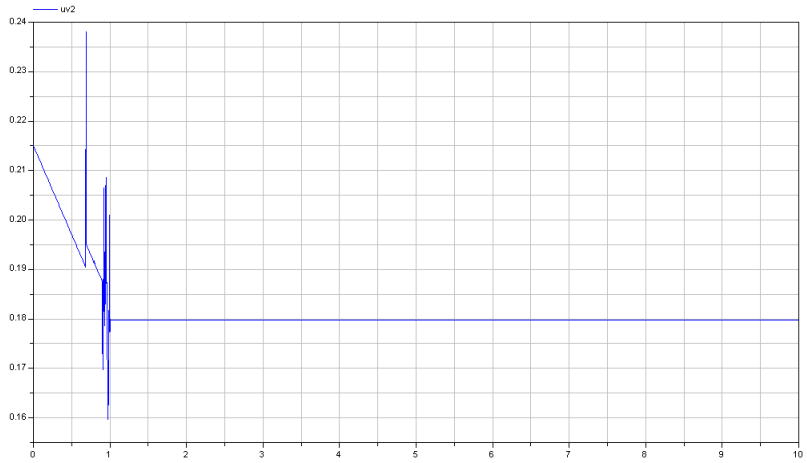

Figure 3.15 Joint 2 torque in the real robot

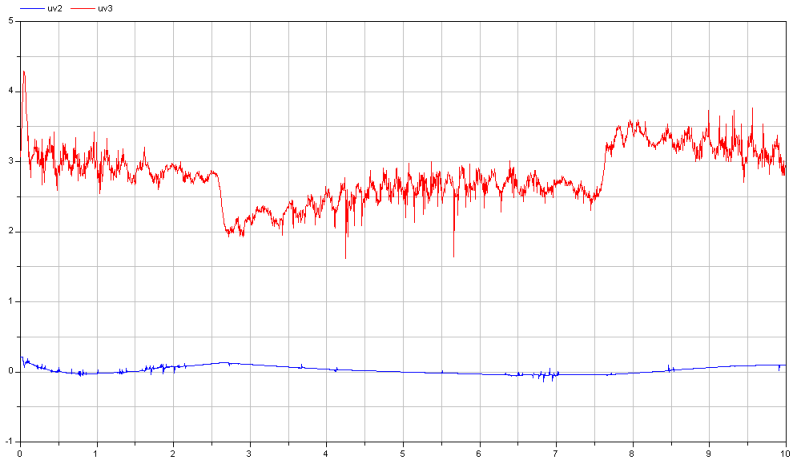

Figure 3.16 Torques in joint 2 and 3 moving the third link

essary to define one *revolution joint* as a planar cut joint, in the planar loop made up by the four joints. If not, a simulation error will be displayed. This option is included in one of the tabs in the joint properties menu.

Refering to equation 3.1, instead of include a spring and a damper to simulate de frictions, a *Bearing Friction* class was used in each one of the revolution joints. It consists of a table with speed values in the horizontal axis, and the resistive torque values on the vertical. The rest of the values are lineary interpolated.

To complete the new model, the conectors for the torque signals and the sensors were included. Because of the real sensors output is measured in motor-radians, a *gain block* was included to convert the model sensor output to the appropiate value, and then work with the model in the same way as in the real robot is done. After that, a new body model was created, as can be seen in fig. 3.17.

Another consequence of adding the new part in the model was a greater stability of the system, making possible to perform some simulations with reasonable random values, because of that the calibration task was taken up

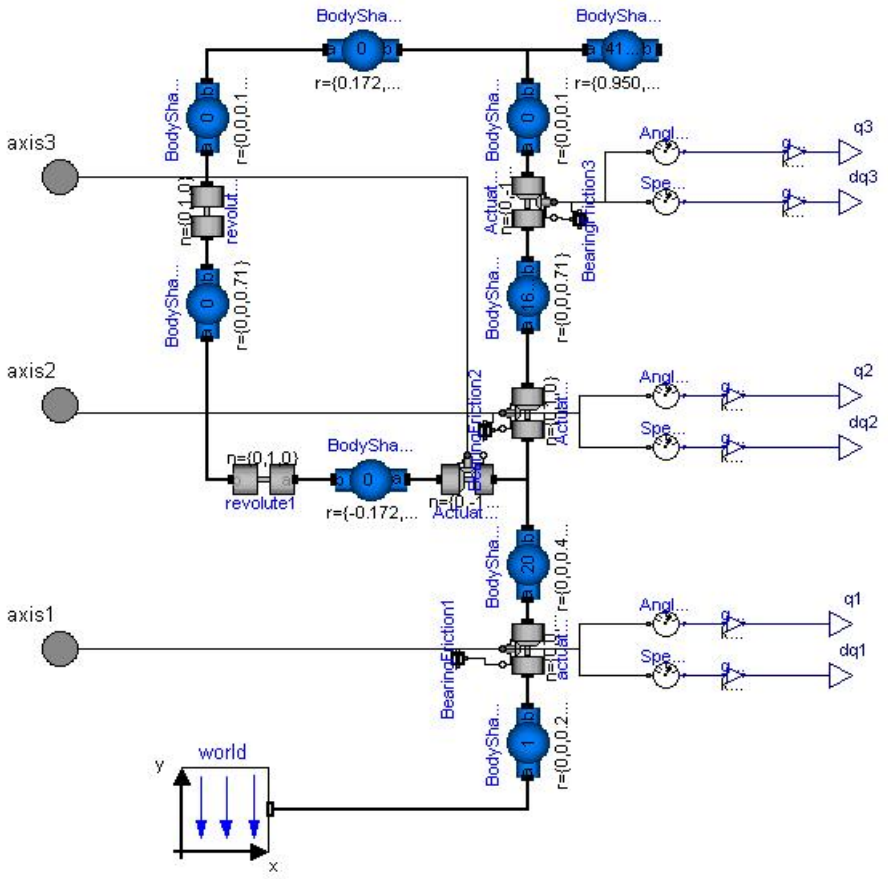

Figure 3.17 Body model

again.

In order to see the influence that the values of the masses and centers of gravity of the second and third link has in the torque signal, the *sweepParameter* function was employed. Four different experiment were done; selecting as variable parameter the masses and the position of the center of mass for each link and the saturated torques as the criterion(see figs. 3.18 to 3.21).

By means of these tests, it was proved that both of the masses and the possition of the center of mass in the second link had a significative influence on the torque applied to joint two, being the center of mass of the third link indepent of it. That and the third link mass, as it is obvious, had influence on the third joint torque.

It is also possible to execute the *sweepTwoParameters* function with the mass and the center of mass as tuner parameters, and the torque on second joint as criterion to chech that the center of mass has not any influence on it(see fig. 3.22.)

With this, the next step was to perform a calibration of the three parameters related to the second joint torque, using the saturated torque signal from the second engine of the robot as criterion, and once these parameters were tuned, calibrate the possition of the center of mass in the third link in a similiar way. The trajectory used to do that, was trapezium in the second link reference with which the robot reach a fixed possition, and stays there for a while, and after that, it comes back to the home position. The third link stays

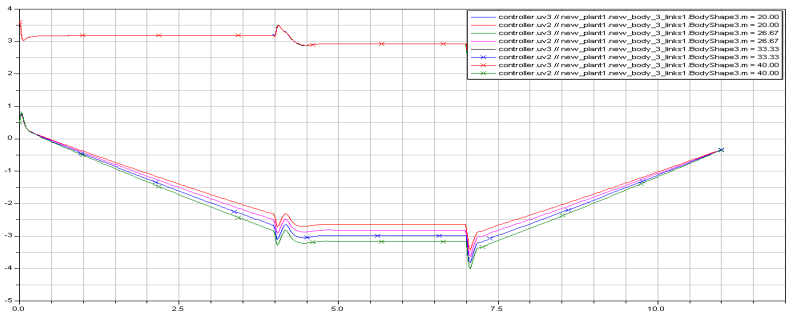

Figure 3.18 Influence of the second link mass in the uv2 and uv3 torques

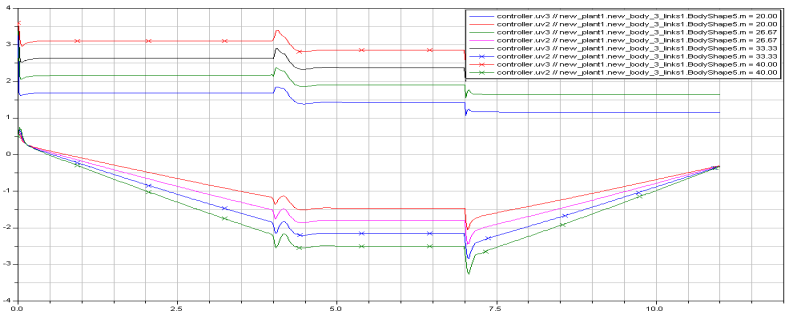

Figure 3.19 Influence of the third link mass in the uv2 and uv3 torques

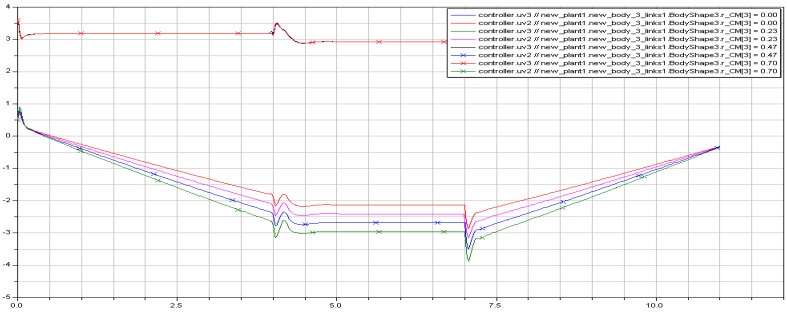

Figure 3.20 Influence of the center of mass of the second link in the uv2 and uv3 torques

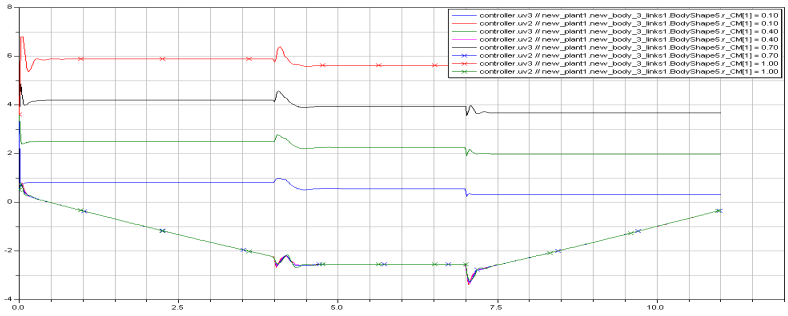

Figure 3.21 Influence of the center of mass of the third link in the uv2 and uv3 torques

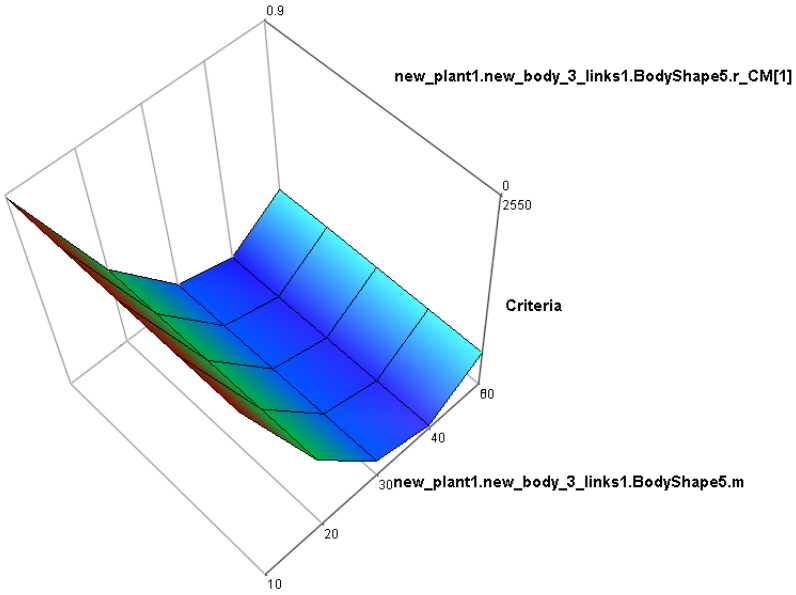

Figure 3.22 *sweepTwoParameters* output

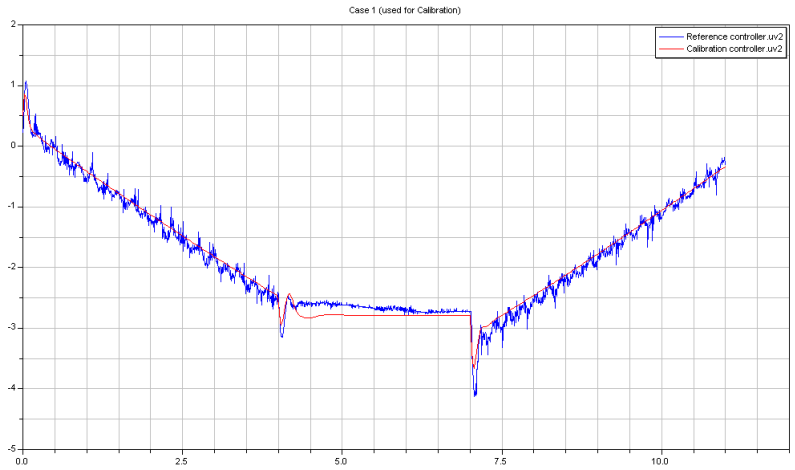

Figure 3.23 Calibration result for the second joint torque

all the time horizontal. The results are shown in figs. 3.23 and 3.24, and despite the noise coming from the torque sensors, it is noticeable that the torque signal from the model is similar to the real values, regarding that the model is being built, and there are some other influences(inertias, frictions) that are not included on it yet.

Perhaps it would be easier to move the robot to that position, and keep it there. Then, record only the data corresponding to the interval in which the robot is stopped, but to implement that in the model is too hard because of the initial conditions definition, so that the other way was chosen thinking about performing an initial calibration to get the first values, and if the validation result was not quite good, try to perform the calibration in other way.

With regard to the parameters of the first link, it is of supposing that as

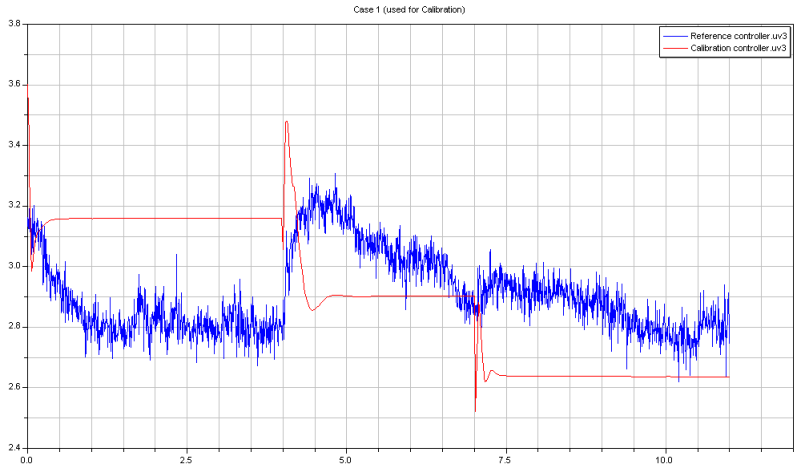

Figure 3.24 Calibration result for the third joint torque

the rotational axis is vertical, the inertia parameters[6] will be more significant than the mass. This can be observed by means of: executing the checkCalibrationSensitivity function with the inertia tensor parameters as tuners, and the torque and position signals as criterions, which concludes that:

```
 Sensitivity Checking.

The calibration criteria are insensitive for small variations
around the nominal values in the following parameters:

new_plant1.new_body_3_links1.BodyShape2.I_11
new_plant1.new_body_3_links1.BodyShape2.I_22
new_plant1.new_body_3_links1.BodyShape2.I_21
new_plant1.new_body_3_links1.BodyShape2.I_31
new_plant1.new_body_3_links1.BodyShape2.I_32
```

Therefore only the "I_33" parameter will be used. Next, using the *sweepTwoParameters* function, choosing the mass and the inertia element as tuners, the same criterion, and an sinusoidal input trajectory(acceleration non equal to zero), we can see how these parameters influence on it(see fig. 3.25), and prove that the mass has not any influence on the torque. Because of the inertia parameters are related to the acceleration, we tuned all the other parameters for the links leaving the inertia tensor to be calibrated with a simultaneous movement of the three links at the end.

As it was said before, the bearing friction model is composed of Coulomb friction and viscous friction together [11]. Some experiments at different speeds were carried out, in order to fill the table of the *Bearing Friction* class, always taking care about the torque saturation limits so that not to put in a risk the robot.

Once more, the *calibrate* function was executed with an triangular input position reference(constant velocity in absolute values), the corresponding parameter in the *Bearing Friction* table as tuner, and the output position and

[6]The inertia tensor of each *Body Shape* is composed of: $I_{11}, I_{22}, I_{33}, I_{21}, I_{31}$ and I_{32}

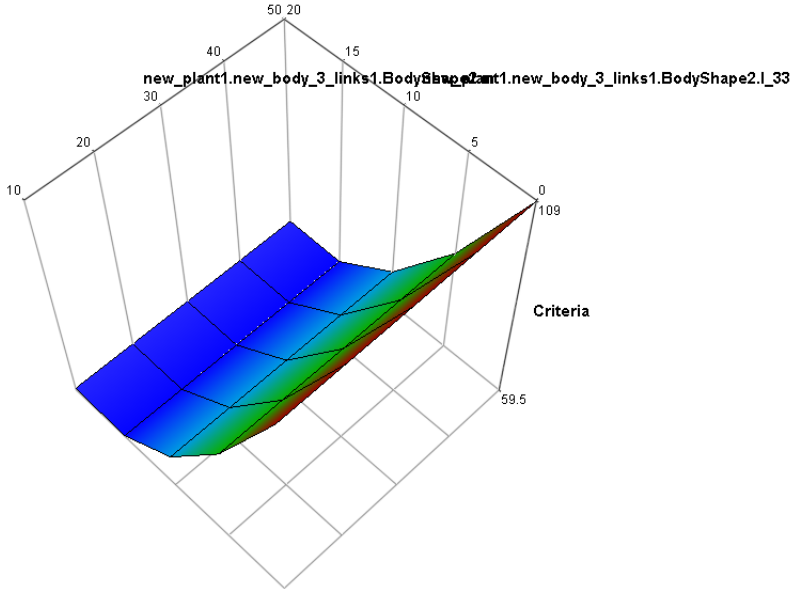

Figure 3.25 The mass of the link 1 has not any influence on the torque

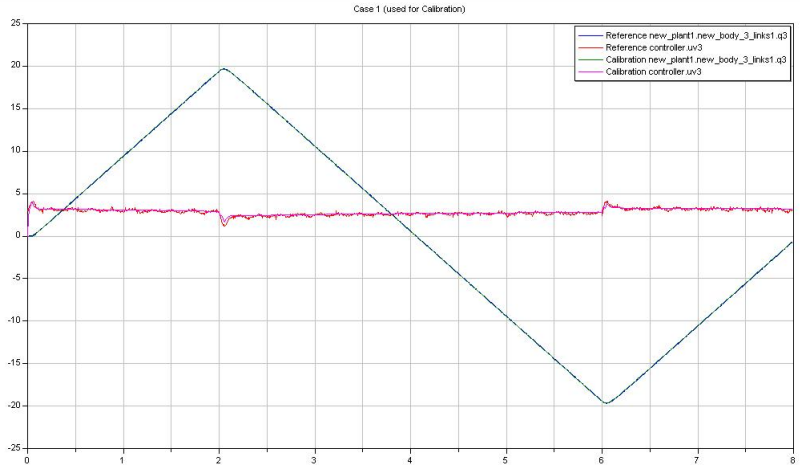

Figure 3.26 Bearing Friction Calibration

torques signals as criterions, giving two times more weight to the position(see fig. 3.26). This kind of experiment was repeated with different speeds in each link in order to get the friction curves shown in figs. 3.27 to 3.29.

Now, there are only the inertia tensor parameters left to be estimated. Since they are directly linked with the acceleration values, three different sinusoidal trajectories were applied to all the joints to move all the links at the same time. As it was saif before, the inertia tensor of each *Body Shape* is made by six elements so that we selected all of them as tuner parameters, the output position and torque signal as criterions and then, checked if all of them are significative in the model. In order to do that, the *checkCalibrationSensitivity* function was used with the three links individually, before calibrate the parameters. The output of this function was, for link two and three respectively[7]:

[7]This function was executed previously with link 1 in pag. 20

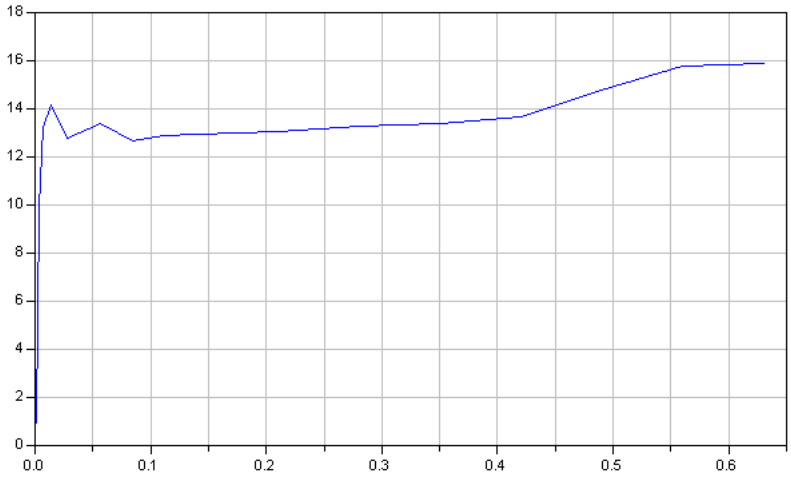

Figure 3.27 Bearing Friction Curve in Joint 1

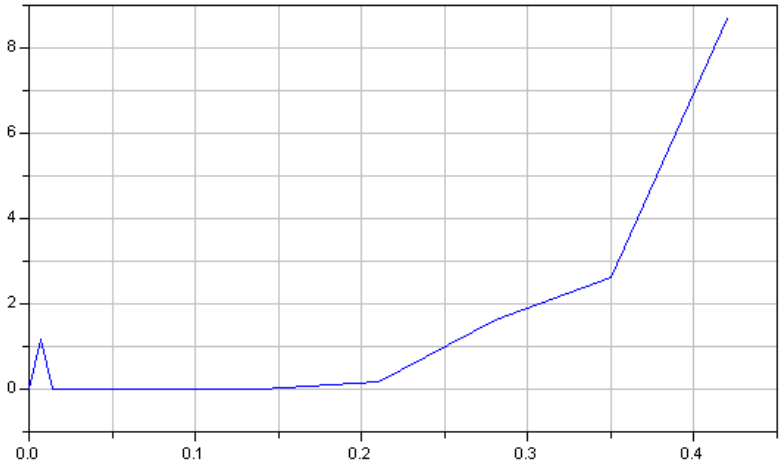

Figure 3.28 Bearing Friction Curve in Joint 2

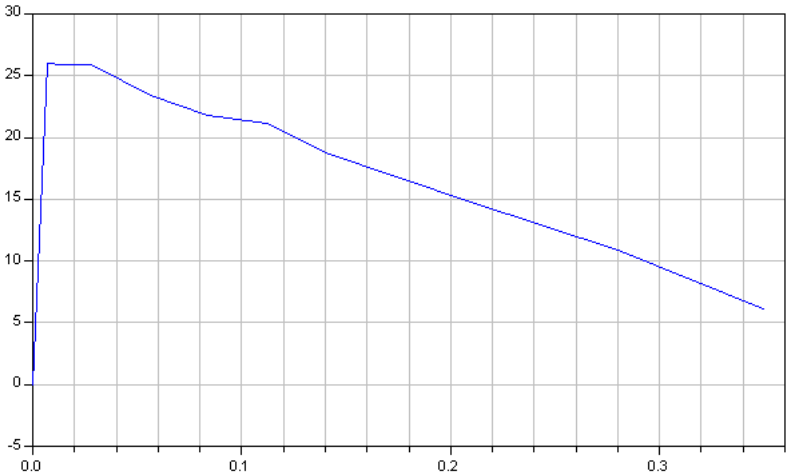

Figure 3.29 Bearing Friction Curve in Joint 3

```
Sensitivity Checking.

The calibration criteria are insensitive for small variations
around the nominal values in the following parameters:

new_plant1.new_body_3_links1.BodyShape3.I_21
new_plant1.new_body_3_links1.BodyShape3.I_31
new_plant1.new_body_3_links1.BodyShape3.I_32

in the following linear parameter combinations:

-new_plant1.new_body_3_links1.BodyShape3.I_32-
  0.5550*new_plant1.new_body_3_links1.BodyShape3.I_11-
  0.3026*new_plant1.new_body_3_links1.BodyShape3.I_31

-new_plant1.new_body_3_links1.BodyShape3.I_21+
  0.0065*new_plant1.new_body_3_links1.BodyShape3.I_11-
  1.0400*new_plant1.new_body_3_links1.BodyShape3.I_31

-new_plant1.new_body_3_links1.BodyShape3.I_22-
  0.6795*new_plant1.new_body_3_links1.BodyShape3.I_11-
  0.8898*new_plant1.new_body_3_links1.BodyShape3.I_31

-new_plant1.new_body_3_links1.BodyShape3.I_33+
  3.2853*new_plant1.new_body_3_links1.BodyShape3.I_11-
  20.6474*new_plant1.new_body_3_links1.BodyShape3.I_31



Sensitivity Checking.

The calibration criteria are insensitive for small variations
around the nominal values in the following parameters:

new_plant1.new_body_3_links1.BodyShape5.I_11
new_plant1.new_body_3_links1.BodyShape5.I_22
new_plant1.new_body_3_links1.BodyShape5.I_21
new_plant1.new_body_3_links1.BodyShape5.I_31
new_plant1.new_body_3_links1.BodyShape5.I_32

in the following linear parameter combinations:

-new_plant1.new_body_3_links1.BodyShape5.I_31-
  1.1031*new_plant1.new_body_3_links1.BodyShape5.I_32-
  1.0848*new_plant1.new_body_3_links1.BodyShape5.I_22

-new_plant1.new_body_3_links1.BodyShape5.I_33-
  0.4947*new_plant1.new_body_3_links1.BodyShape5.I_32-
  0.3504*new_plant1.new_body_3_links1.BodyShape5.I_22

-new_plant1.new_body_3_links1.BodyShape5.I_21+
  0.0138*new_plant1.new_body_3_links1.BodyShape5.I_32+
```

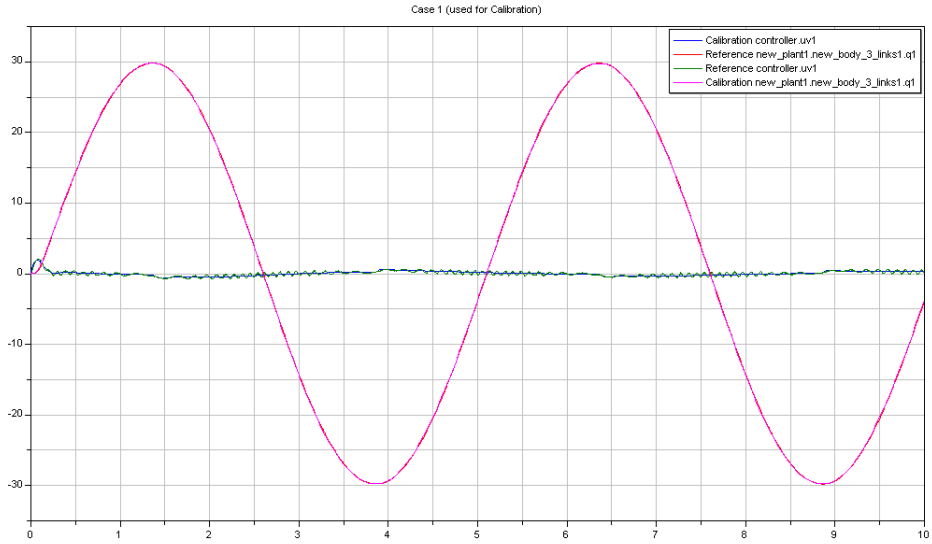

Figure 3.30 Inertia tensor calibration for link 1

```
  0.0194*new_plant1.new_body_3_links1.BodyShape5.I_22

-new_plant1.new_body_3_links1.BodyShape5.I_11+
  0.5737*new_plant1.new_body_3_links1.BodyShape5.I_32+
  0.4146*new_plant1.new_body_3_links1.BodyShape5.I_22
```

In this case this function gave us besides the parameters that had not any influence on the criterion, the linear combinations that some of the tuners formed. Since some of them are the ones that do not affect the criterion, they were removed as calibration parameters, and the function was executed again. Then the new output was:

```
Sensitivity Checking.

The calibration criteria are sensitive for small variations
around the nominal values in all tuner parameters and in all
their linear combinations
```

Therefore, those parameters were used as tuners, with the same inputs and the position output and torque signals from the robot as criterion and the *calibrate* function was executed to get the last parameters for the links. See figs. 3.30 to 3.33 to see final results of the calibration task in the three links.

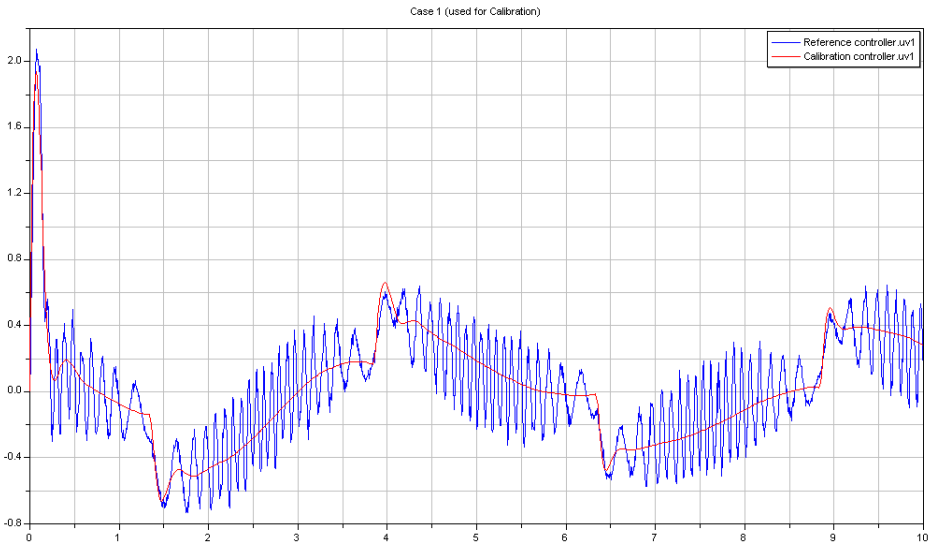

Figure 3.31 Detail of the torque in link 1

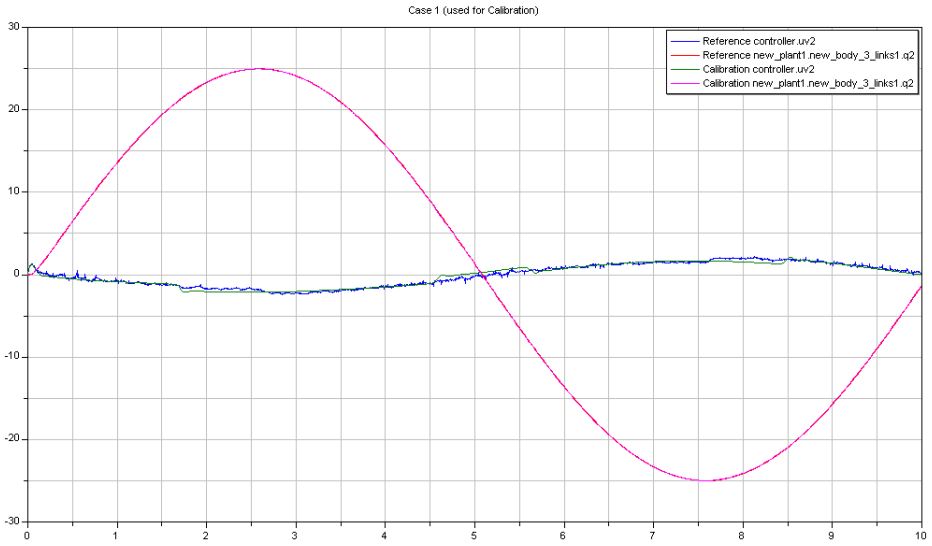

Figure 3.32 Inertia tensor calibration for link 2

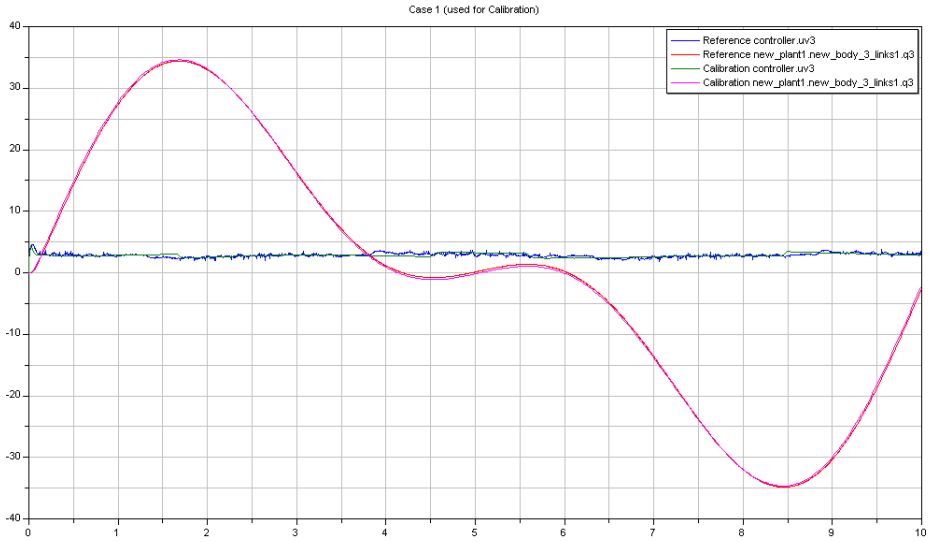

Figure 3.33 Inertia tensor calibration for link 3

4. Control Design

The goal of this chapter is to show how a controller can be designed using some tools included in the *Dymola Design* library[1], as *Overshoot, Settling Time* or *Rise Time* blocks. An exhaustive description of their operation mode can be found on [12].

Basically, these blocks are conected to the signal that is wanted to be optimized, and the specifications are set in their own setup windows. In this case, the criterion are imposed by the blocks used, and setting a weight for each one. After that, a similar process to the calibration one is followed, executing the *Optimization* function, and selecting as tuners parameters the controller constants.

The initial implementation of the controller was kept, and two additional feedforward loops were added, one for the speed, and one using the inverse model of the one developed before. The idea is with the desired trajectories and the inverse model, generate the suitable torque to be applied in the real robot(in this case, the model), and therefore, avoid waiting for the error in the position loop, which is slower, and introduce a delay.

The implementation of the new controller is shown in fig. 4.1. On the top it is possible to see the how the inverse model is inserted; it has been flipped horizontal, and the *Two Inputs* and *Two Outputs* blocks[2] have been used to be able to connect two inputs and two outputs respectively.

In figs. 4.2 to 4.4, are displayed the difference between the reference position and the real output from the sensors in the real robot in the laboratory and in

[1]This library was include since Dymola 6

[2]Included in Modelica.Blocks.Math

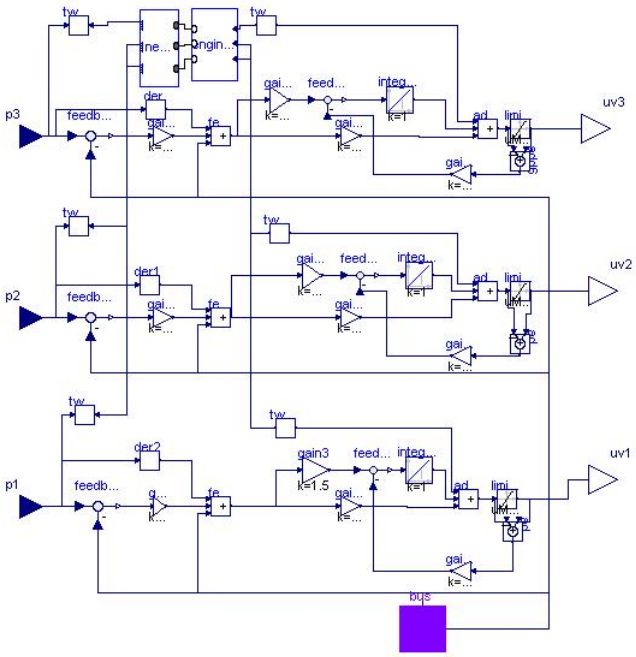

Figure 4.1 New Feedforward Controller

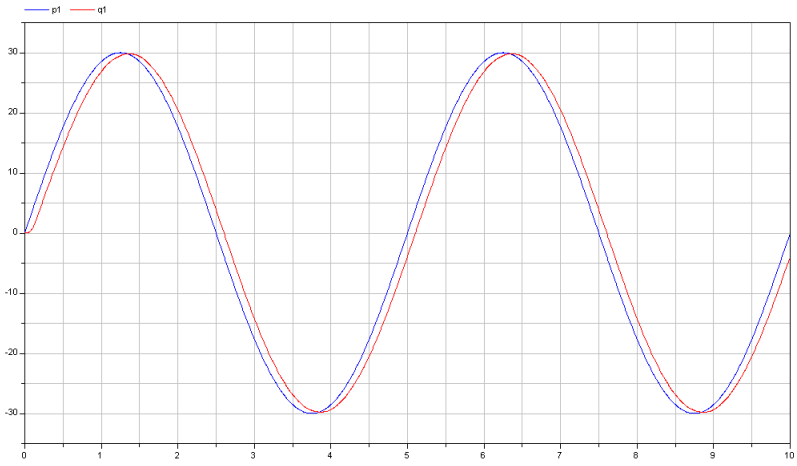

Figure 4.2 Real Input and Output for link 1

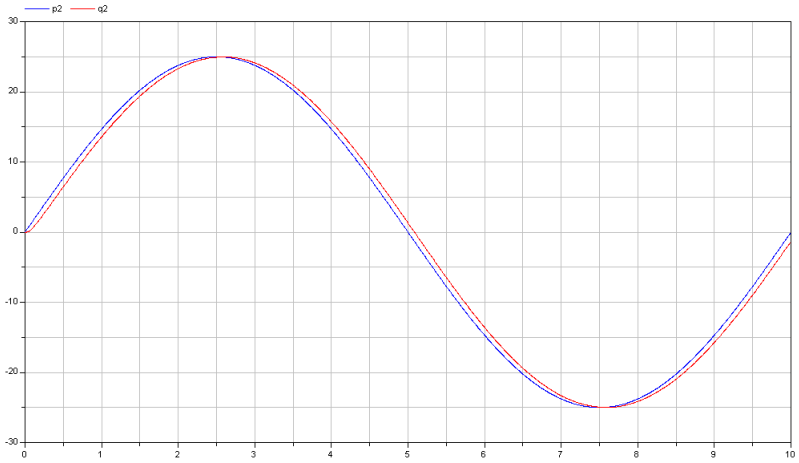

Figure 4.3 Real Input and Output for the link 2

figs. 4.5 to 4.7 the same signals are plotted when the new controller is working in the model. Since the previous controller had a good design, it was not necessary to change any value of its constants to get an good result, altough it is obvious that is an ideal one since it is not affected by noise and the inverse model is being used with the direct model. In fig. 4.8 it is possible to appreciate an initial transient error, that vanish quickly thanks to the feedforward loops. Only the error in one link has been displayed, altough similar errors and a similar behaviour appeared in the other links.

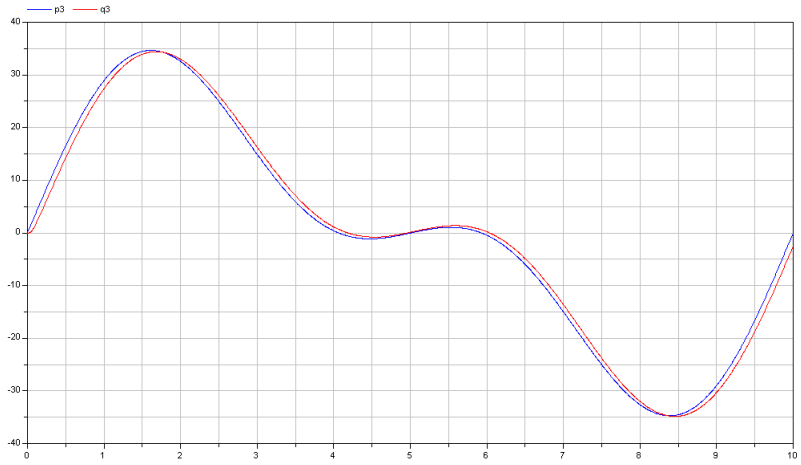

Figure 4.4 Real Input and Output for the link 3

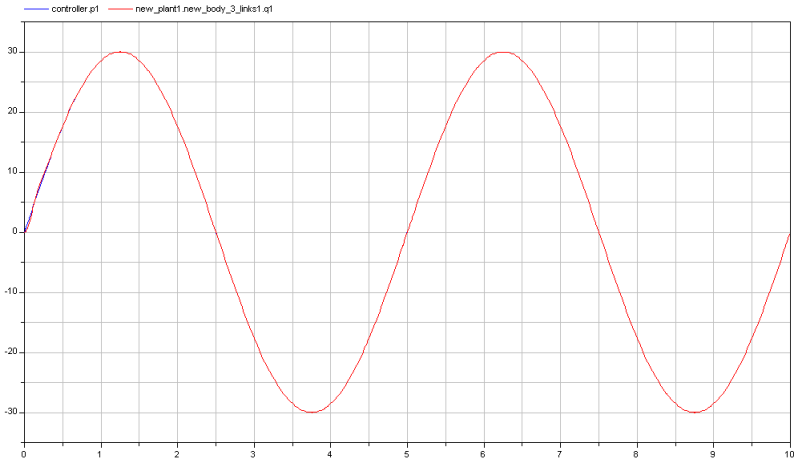

Figure 4.5 Theoretical Response for the Model with the Feedforward Controller in link 1

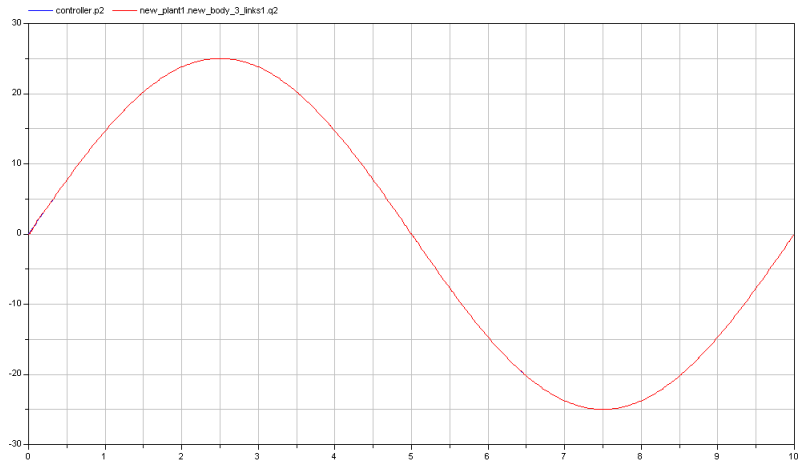

Figure 4.6 Theoretical Response for the Model with the Feedforward Controller in link 2

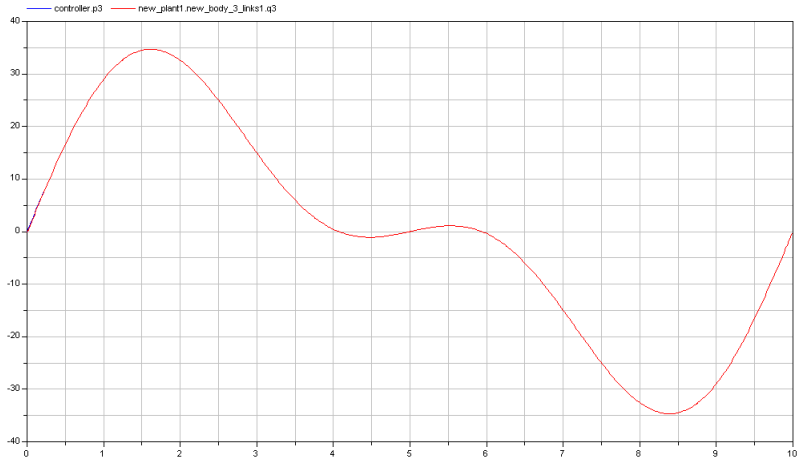

Figure 4.7 Theoretical Response for the Model with the Feedforward Controller in link 3

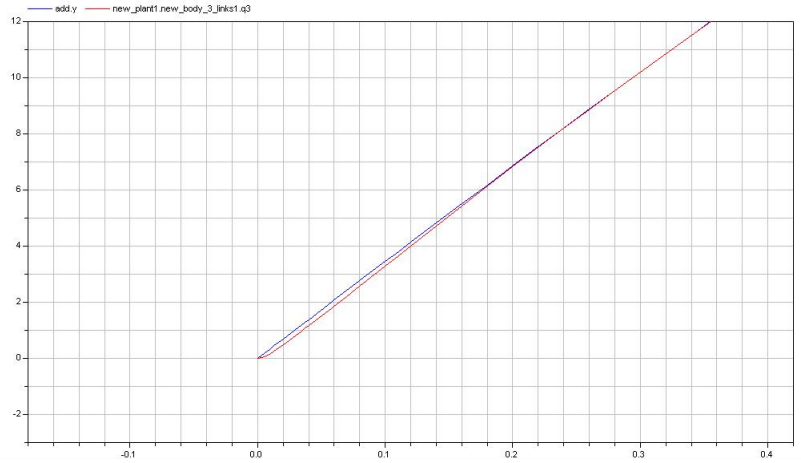

Figure 4.8 Initial error in the model with the Feedforward Controller

5. Conclusions and Future Work

During this paper, a different way of modeling and parameter estimation task has been introduced.

Dymola is a very powerful software to perform simulations, and thanks to the new features introduced in the new version (*Dymola 6*), it is also possible to work in indentification and optimization task in an confortable enviroment on an easy way.

Although it is not compulsory to have any previous knowledge about *Modelica* language, sometimes it is good to find out a problem, which is quite difficult when you are working with a big model and something is wrong.

Personally, I have not found a lot of problems using this sofware, except the usual ones that appear when someone starts to work with a new program. Perhaps the help guide could be better.

Regarding to the main goal of this Master Thesis, a good model of a three links robot has been developed, taking care of the space restriction in the laboratory to move the robot in other possibilities, and the risk of forcing it moving at higher speeds or frequencies.

But at least the main steps about how to build a grey-box model, and calibrate its parameters with Dymola have been presented, and in a future work, another things could be added.

For instance, it could be possible to increase the degrees of freedom, check the effect of flexibilities in the position tracking when the robot is subjected to very quick direction changes, or the influence of different tools with different center of mass.

With regard to the control design, given the possibilities, it was not necessary to introduce a lot of changes, and was really easy to check the improvement of the robot behavior, at least theoretically, with the use of the feedforward and the inverse model. But of course, using the design tools described before, the control design becomes a easy task, with a basic knowledge in control theory and avoiding the hard mathematical calculations.

A. Controller values

```
Two cascaded PID-controllers (Pos and Vel)

# {joint1 joint2 .... joint6}

IrbKinematics_JointType RegParDefault_KPosDefault = {
  10.0, 12.5, 17.5, 10.0, 15.0, 10.0
};

IrbKinematics_JointType RegParDefault_KVelDefault = {
  /* From DSP era: 0.3, 0.75, 0.75, 0.03, 0.05, 0.03 */
  0.3/2, 0.75/2, 0.75/2, 0.03/2, 0.05/2, 0.03/2
};
IrbKinematics_JointType RegParDefault_TiVelDefault = {
  0.1, 0.05, 0.05, 0.02, 0.01, 0.01
};
IrbKinematics_JointType RegParDefault_TiPosDefault = {
  0.0, 0.0, 0.0, 0.0, 0.0, 0.0
};
IrbKinematics_JointType RegParDefault_TdDefault = {
  0.0, 0.0, 0.0, 0.0, 0.0, 0.0
};

// filter gains for derivative part  sTd/(sTd/N+1)
IrbKinematics_JointType RegParDefault_NDefault = {
  10.01, 10.02, 10.03, 10.04, 10.05, 10.06
};

IrbKinematics_JointType RegParDefault_TauMaxDefault = {
  6.0, 10.5, 6.8, 1.5, 1.8, 0.9
  /* 6.3, 10.9, 7.0, 1.54, 1.93, 1.0 */
  /* 3.0, 6.0, 4.0, 0.7, 1.0, 0.6 */
};


// some problems with offset at start...
IrbKinematics_JointType RegParDefault_ImaxVelDefault = {
  3.5, 6.0, 4.0, 0.8, 1.2, 0.6
  /* Disable I-parts where possible due to friction/noise problems: */
  /* 0.0, 6.0, 4.0, 0.0, 0.0, 0.0 */
};
```

B. Matlab Code For Data Processing

```
function parameters=trtparamtrs(exc_inp,exc_resp,sim_time,file_name)
%Given the outputs and inputs froma Exec_handler, the function trtparamtrs
%mofify them in order to create the correct *csv file to be used in
%Dymola.
%
%parameters=trtparamtrs(exc_inp,exc_resp,sim_time,file_name)
%
%exc_inp and exc_resp are the output files of Exec_handler, sim_time is the
%simulation time, and file_name is a string which is used to create de csv
%file in the Dymola work folder.
%
%Index to be added in Excel.
%for three links or:
%time,uv1,uv2,uv3,u1,u2,u3,p1,p2,p3,q1,q2,q3
%  1   2   3   4   5  6  7  8  9 10 11 12 13


disp(' ');
disp(' ');
disp('WARNING!');
disp(' ');
disp('Make sure that the input sim_time and the sample time values are right');
disp(' ');
num_data=length(exc_inp);
sample_time=0.005
time=[0:sample_time:sim_time]';
limit=num_data-2; %Due to the diff function

uv1=exc_inp(1,:)';
uv2=exc_inp(2,:)';
uv3=exc_inp(3,:)';
uv=[uv1 uv2 uv3];


u1=exc_inp(7,:)';
u2=exc_inp(8,:)';
u3=exc_inp(9,:)';
u=[u1 u2 u3];


p1=exc_inp(4,:)';  %Used with the model in motor_rad.
p2=exc_inp(5,:)';
p3=exc_inp(6,:)' + p2;
p=[p1 p2 p3];


v1=(1/sample_time).*diff(p1);
v2=(1/sample_time).*diff(p2);
v3=(1/sample_time).*diff(p3);
```

```
v=[v1 v2 v3];


q1=exc_resp(1,:)'; %Used with the model in motor_rad.
q2=exc_resp(2,:)';
q3=exc_resp(3,:)' + q2;
q=[q1 q2 q3];


% dq1=(1/sample_time).*diff(q1); Not used because of the noise.
% dq2=(1/sample_time).*diff(q2);
% dq3=(1/sample_time).*diff(q3);
% dq=[dq1 dq2 dq3];



parameters=[time(1:limit,:) uv(1:limit,:) u(1:limit,:) p(1:limit,:) q(1:limit,:)];


if isa(file_name,'char')
    cd C:\Documents and Settings\'Luis\'Mis documentos\'Dymola\
    csvwrite(file_name,parameters);
%     csvwrite(['tau1_' file_name],parameters(:,[1 2]));
%     csvwrite(['tau2_' file_name],parameters(:,[1 3]));
%     csvwrite(['tau3_' file_name],parameters(:,[1 4]));
    cd C:\MATLAB7\work
end
```

C. Calibration Results

FIRST LINK DATA	
mass	irrelevant
center of mass	irrelevant
I_{11}	0 kg.$\cdot m^2$
I_{22}	0 kg.$\cdot m^2$
I_{33}	16.69833 kg.$\cdot m^2$
I_{21}	0 kg.$\cdot m^2$
I_{31}	0 kg.$\cdot m^2$
I_{32}	0 kg.$\cdot m^2$

SECOND LINK DATA	
mass	15.8036 kg.
center of mass	{0,0,0.71}
I_{11}	0.00026 kg.$\cdot m^2$
I_{22}	10.84756 kg.$\cdot m^2$
I_{33}	0.013 kg.$\cdot m^2$
I_{21}	0 kg.$\cdot m^2$
I_{31}	0 kg.$\cdot m^2$
I_{32}	0 kg.$\cdot m^2$

THIRD LINK DATA	
mass	37.8274 kg.
center of mass	{0.55936,0,0}
I_{11}	0 kg.$\cdot m^2$
I_{22}	0 kg.$\cdot m^2$
I_{33}	0.00328 kg.$\cdot m^2$
I_{21}	0 kg.$\cdot m^2$
I_{31}	0 kg.$\cdot m^2$
I_{32}	0 kg.$\cdot m^2$

BEARING FRICTION VALUES IN JOINT 1	
Speed(rad/seg.)	$\tau(N \cdot m)$
0.001	0.952422
0.004	9.90996
0.007	13.226781
0.014	14.147392
0.028	12.788362
0.056	13.415834
0.084	12.684469
0.112	12.878753
0.14	12.951634
0.21	13.033332
0.28	13.271714
0.35	13.395182
0.42	13.689285
0.49	14.801288
0.56	15.766499
0.63	15.86827

BEARING FRICTION VALUES IN JOINT 2	
Speed(rad/seg.)	$\tau(N \cdot m)$
0.007	1.177682
0.014	0
0.028	0
0.056	0
0.084	0
0.112	0
0.14	0
0.21	0
0.28	1.617848
0.35	2.629839
0.42	8.696505

BEARING FRICTION VALUES IN JOINT 3	
Speed(rad/seg.)	$\tau(N \cdot m)$
0.007	25.996664
0.014	25.914174
0.028	25.898918
0.056	23.455017
0.084	21.736233
0.112	21.153204
0.14	18.785745
0.21	14.792344
0.28	10.864357
0.35	6.1032148

D. Bibliography

[1] `http://www.dynasim.se`.

[2] `http://www.modelica.org`.

[3] B. Hendriks, "Implementation of industrial robot control," Master's thesis, Lund Institute of Technology, March 1996.

[4] "A short guide to the robot system." www.control.lth.se/ andersro/RobotLab.html.

[5] "Model calibration manual." PDF manual included in Dymola Help.

[6] A. Robotics, "Product manual irb 2000," 1991.

[7] M. Gautier and W. Khalil, "Exciting trayectories for the identification of base inertial parameters of robots," tech. rep., Laboratoire d'Automatique de Nantes, 1991.

[8] M. N. Måns Östring, Svante Gunnarsson tech. rep., Department of Electrical Engineering, Linköpings universitet, 2003.

[9] S. G. Erik Wernholt, "Nonlinear grey-box identification of industrial robots containig flexibilities," tech. rep., Department of Electrical Engineering, Linköpings universitet.

[10] S. S. S. Richard M. Murray, Zexiang Li, *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994. Equation (4.24) pag.171.

[11] W. K. . E. Dombre, *Modeling, Identification & Control of Robots*. Hermes Penton Science, 2002.

[12] "Model optimization manual." PDF manual included in Dymola Help.