REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
الجمهورية الجزائرية الديمقراطية الشعبية

MINISTRY OF HIGHER EDUCATION
AND SCIENTIFIC RESEARCH

HIGHER SCHOOL IN APPLIED SCIENCES
--T L E M C E N--

وزارة التعليم العالي والبحث العلمي

المدرسة العليا في العلوم التطبيقية
-تلمسان-

Mémoire de fin d'étude

Pour l'obtention du diplôme d'Ingénieur

Filière : Automatique
Spécialité : Automatique

Présenté par : AISSANI Omar

Thème

Modeling and Control of a 4 DOF Robot Manipulator

Soutenu publiquement, le 14/07/2021, devant le jury composé de :

M. CHERKI Brahim	Professeur	ESSA. Tlemcen	Président
M. ARICHI Fayssal	Maître de conférences B		Directeur de mémoire
M. MOKHTARI Rida	Maître de conférences A		Co- Directeur de mémoire
M. ABDELLAOUI Ghouthi	Maître de conférences B		Examinateur 1
M. BENSALAH Choukri	Maître de conférences B		Examinateur 2

Année universitaire : 2020/2021

Acknowledgments

My sincere thanks go to Professor CHERKI Brahim for accepting to be the president of the examiners. I am genuinely impressed by his knowledge and humbleness. One must be lucky to be one of his students. I would also like to thank Dr. ABDELLAOUI Ghouthi and Dr.BENSALAH Choukri for the valuable inputs to be included in this thesis and generously spent precious time in giving the suggestions and comments for this thesis.

I would like to thank my supervisors Dr.ARICHI Fayssal, as well as DR.MOKHTARI Rida, for their patience, guidance and reviews.

Most importantly, I am grateful for my parents's love and unconditional support. I have no words to acknowledge the sacrifices you made, Thank God for giving me you.

My dear sisters Ouassila and Hidayet. You have always had a way of pulling me out of a miserable mood and putting a smile on my face. I am so proud of you both. I wish you all the best.

Abstract

In this thesis we were interested in studying a 4 degrees of freedom robotic arm, that is SCARA, which stands for Selective Compliance Articulated Robot Arm, We started by exploring the forward and inverse kinematics models of a SCARA robot, we then computed its velocity kinematics, then we modeled some robot dynamics using Euler-Lagrange formalism. We analyzed PID controllers and Computed Torque Controller for a single joint robot and finally we have used SIMULINK Simscape Multibody Toolbox to simulate the robot.

Résumé

Dans ce mémoire, nous étions intéréssé par l'étude d'un bras manipulateur à 4 degré de liberté de type SCARA. Nous avons étudié le modèle géométrique direct ainsi que le modèle géométrique inverse. Nous avons aussi obtenu le modèle cinétique du robot. Et puis nous avons essayé de modéliser quelques robots manipulateur en utilisant le formalism d'Euler-Lagrange. Nous avons analysé le régulateur PID et la méthode du couple calculé qui est plus connue sous le nom Anglais Computed Torque controller CTC. Finalement nous avons modélisé le robot SCARA en utilisant le toolbox de SIMULINK, Simscape Multibody.

Key Words

SCARA, Robotic Manipulator, Computed Torque Control, CTC, PID, Simscape Multibody, Jacobian, Forward Kinematics, Inverse Kinematics, Forward Dynamics, Inverse Dynamics

Table of Contents

List of Figures

Chapter 1

Introduction

It is just amazing how humans can do activities like moving around and manipulating objects, cooking, practicing sports, driving a car, etc. Without any considerable efforts. Although these tasks seem seamless to us and most of us take them for granted, they are in fact very complicated problems. Let's analyze what it takes to pour water in a cup. We should first recognize the bottle. we should then know where it is in space. Then, we move our hands to the position of the bottle, we try to grip it and lift it off. Move the bottle around whilst trying to not spill the water. When reaching the cup we should reorient the bottle such that the water can be poured, we should know when to stop or otherwise we will overflow the cup, and there is more to it.
Robotics is the science that treats these kind of problems, some of them are: Kinematics and dynamics, trajectory planning, path planning, vision and sensing, etc.

1.1 Etymology

The word robotics was derived from the word robot, which was introduced to the public by Czech writer Karel Čapek in his play R.U.R. (Rossum's Universal Robots), which was published in 1920. The word robot comes from the Slavic word robota, which means slave/servant. The play begins in a factory that makes artificial people called robots, creatures who can be mistaken for humans

1.2 Robotic Manipulators

Robotic manipulators are a special kind of robots that specialize mainly in doing repeatable tasks that involve manipulating object in 3D space, following precise paths, etc. It is defined as a set of links connected to each other by a joint driven by a motor, at the end of this chain there is an end effector e.g. vacuum gripper, spray paint, etc.
There exist different types of joints as depicted in 1.1, such as: revolute joints, prismatic joints, universal joints, ball Joints, etc.
The two main joints used in robotics are:

- Revolute joint (R), it has only one degree of freedom, that is rotation around a axis
- Prismatic joint (P), characterized by one degree of freedom, that is sliding in the direction of a given axes

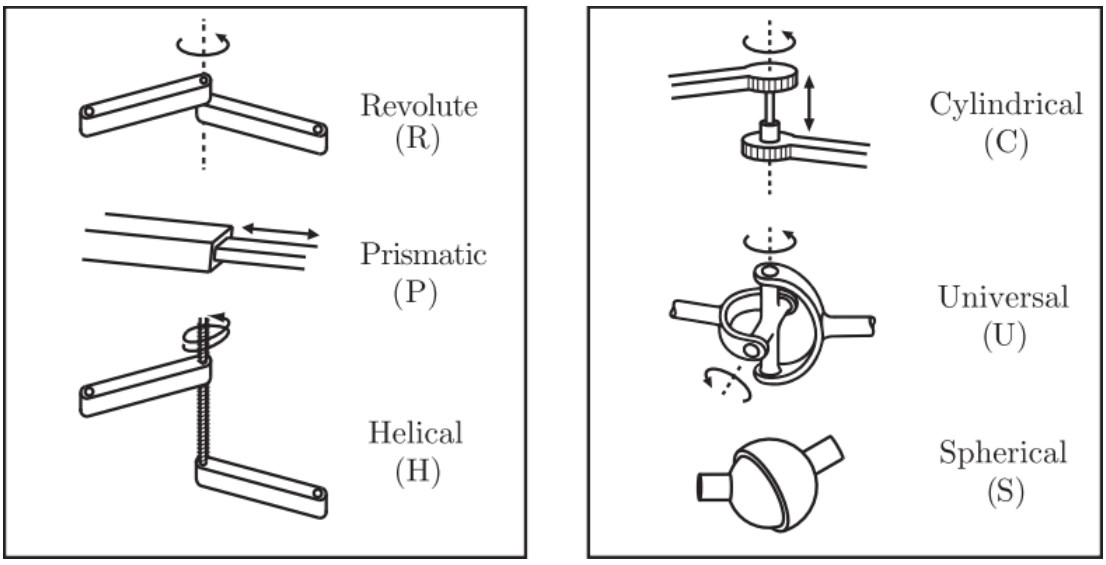

Figure 1.1: Typical robot joint

1.3 Types of Robot Manipulators

There are different types of manipulators robots, each of them is suitable to do a particular task, some of these robots are:

Gantry Robots

These robots have linear joints and are mounted overhead. They are also called Cartesian and rectilinear robots. see figure 1.2

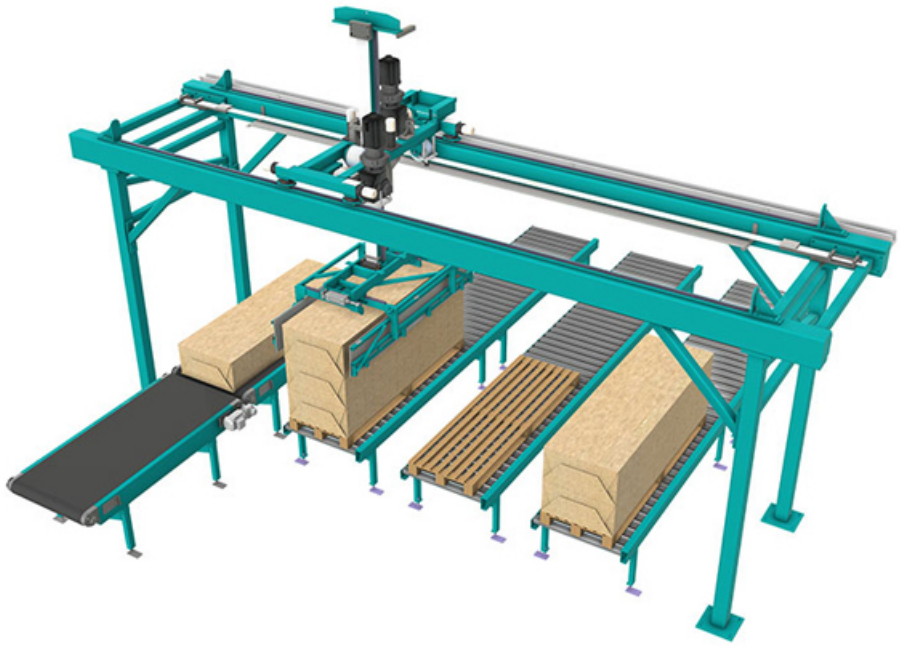

Figure 1.2: Example of Gantry Robot

SCARA Robots

SCARA stands for *Selective Compliance Assembly Robot Arm*, it was invented by Prof. Hiroshi Makino of Yamanashi University, Japan, in 1972. It is one of the most used robots in industry as they are suitable for pick and place applications. Figure 1.3 and 1.4 show two different

configuration of SCARA robots.

Figure 1.3: PRRR SCARA Robot

Figure 1.4: RRPR SCARA Robot

Articulated Robot
Articulated robots are the most used robotic arms, since they are versatile and can provide decent speeds and a wide range of possible orientations. These robots are widely used in automotive industry to paint vehicles, assembly, etc. Figure 1.5 shows an articulated robot.

Figure 1.5: An Example of Articulated Robot

Cylindrical robots
A cylindrical robot is shown in figure 1.6

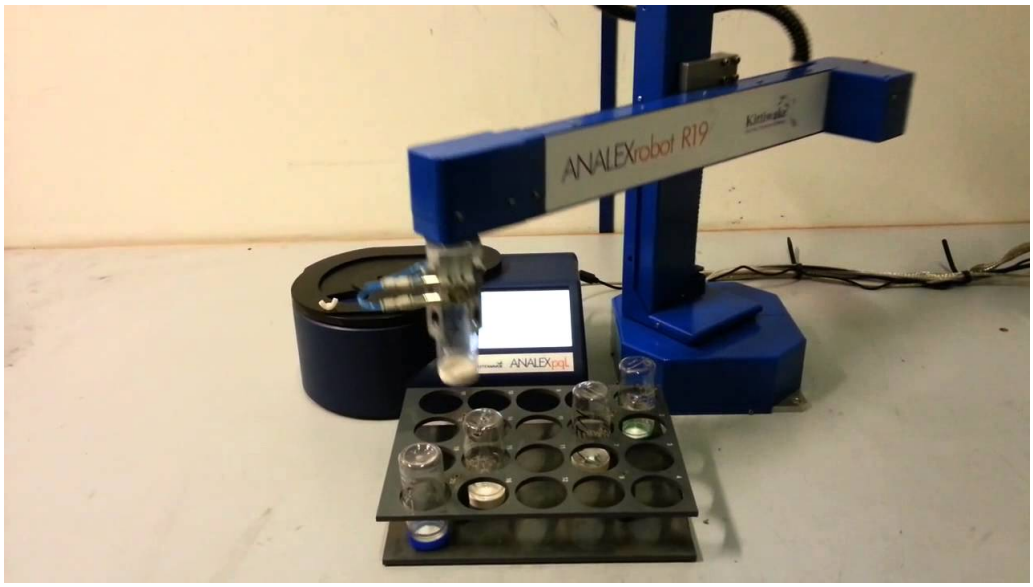

Figure 1.6: Cylindrical Robot

Delta Robot

Delta robots 1.7 are one of the fastest robots since the motors are not located at each joint, which makes the links lighter, hence it can reach fast speeds. These robots are used as pick and place machines and as 3D printers.

Figure 1.7: Delta Robot

Stewart-Gough Platform
This robot is a parallel robot composed of 6 linear actuators. It is widely used as a flight simulator. A Stewart-Gough Platform is depicted in figure 1.8

Figure 1.8: Stewart Platform

1.4 History of Robotic Arms in Manufacturing

It is widely understood that the first programmable robotic arm was designed by George Devol in 1954. Collaborating with Joseph Engelberger, Devol established the first robot company, Unimation in 1956, in the USA. Then in 1962 General Motors implemented the Unimate robotic arm in its assembly line for the production of cars. A few years later, a mechanical engineer at Stanford University, Victor Scheinman was developing a robotic arm that was one of the first to be completely controlled by a computer in 1969. This industrial robot, known as the *Stanford Arm* was the first six axes robotic arm and influenced a number of commercial robots that followed. A Japanese company, Nachi, developed their first hydraulic industrial robotic arm in 1969 and after this a German firm, Kuka, pioneered the first commercial six axes robotic arm, called Famulus, in 1973. Predominantly, these robots were utilised for spot welding tasks in manufacturing plants but as technology developed, the range of tasks that robotic arms could perform also expanded. The advances in technology includes the increasing variety in end-of-arm tooling that has become available. This means that Robotic arms can perform a wide range of tasks beyond welding depending on the tools that are attached to the end of their arms. Current innovations in end of arm tools include; 3D Printing tool heads, heating devices to mould and bend materials, and suction devices to fold sheet metal.

Chapter 2

Kinematics

2.1 Forward Kinematics

Forward Kinematics also known as *Direct Kinematics*, refers to the use of the kinematics equations of a robot to compute the position and orientation of the end-effector from specified values for the joint parameters as shown in figure 2.1:

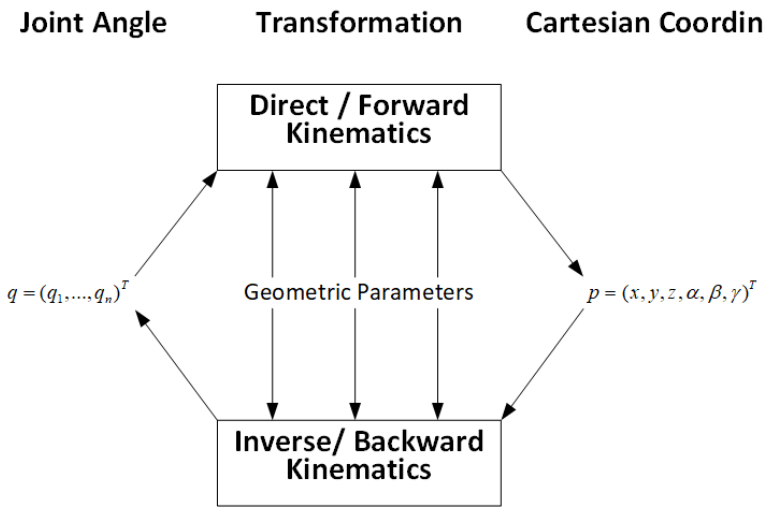

Figure 2.1: Forward Kinematics Diagram

2.1.1 SCARA Manipulator

We can compute the forward kinematics equation of a SCARA manipulator with two main approaches, first approach is straight forward and purely geometrical while the second one is systematic and makes use of the **Denavit-Hartenberg** convention.

Geometrical Approach

The SCARA Manipulators can be decomposed into two simpler parts, the Z translation and a planar movement that contains the 3 revolute joints as depicted in figure 2.2

Using simple vector operations:

$$\overrightarrow{OC} = \overrightarrow{OA} + \overrightarrow{AB} + \overrightarrow{BC} \tag{2.1}$$

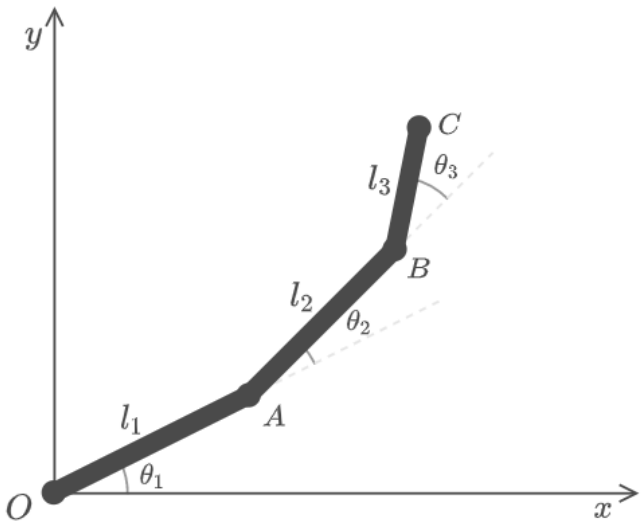

Figure 2.2: Forward Kinematics Diagram

Writing the vectors as a function of $\theta_1,\ \theta_2,\ \theta_3$:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} l_1 \cos(\theta_1) \\ l_1 \sin(\theta_1) \end{pmatrix} + \begin{pmatrix} l_2 \cos(\theta_1 + \theta_2) \\ l_2 \sin(\theta_1 + \theta_2) \end{pmatrix} + \begin{pmatrix} l_3 \cos(\theta_1 + \theta_2 + \theta_3) \\ l_3 \sin(\theta_1 + \theta_2 + \theta_3) \end{pmatrix} \tag{2.2}$$

We finally get:

$$\begin{cases} x = l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) + l_3 \cos(\theta_1 + \theta_2 + \theta_3) \\ y = l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2) + l_3 \sin(\theta_1 + \theta_2 + \theta_3) \end{cases} \tag{2.3}$$

Since all the rotations are about z axis, we can sum all the angles to get the orientation of the end-effector, like so:

$$\varphi = \theta_1 + \theta_2 + \theta_3 \tag{2.4}$$

The z coordinates of end-effector is nothing but joint variable d_4:

$$z = d_4 \tag{2.5}$$

Denavit-Hartenberg Convention

The kinematic diagram of a PRRR SCARA manipulator is shown in figure 2.3:
At first, we have to assign the frames according to the **Denavit-Hartenberg** Rules:

> **Rule 1**: z_n axis is the axis of rotation for a revolute joint and the axes of translation for a prismatic joint.
>
> **Rule 2**: x_n axis must be perpendicular to both z_n and z_{n-1} .
>
> **Rule 3**: y_n axis is determined from the x_n axis and z_n axis by using the right-hand rule.
>
> **Rule 4**: x_n must intersect with the z_{n-1} axis.

One possible solution to this problem is shown in figure 2.4
After Assigning the frames, we should get the **Denavit-Hartenberg** parameters, that are:

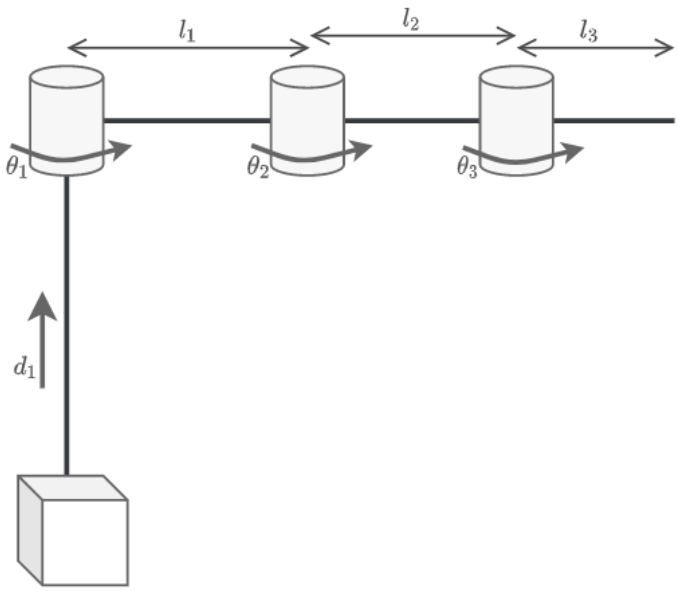

Figure 2.3: Kinematic Diagram of a PRRR SCARA

- θ is the rotation around z_{n-1} that is needed to get x_{n-1} to match x_n.
- α is the rotation around x_n that is required to get z_{n-1} to match z_n.
- r is the distance between the frames $n-1$ and n along x_n.
- d is the distance between the frames $n-1$ and n along z_{n-1}.

The **DH** parameters of PRRR SCARA manipulator of figure 2.3 are shown in this table

	θ	α	r	d
1	0	0	0	d_4
2	θ_1	0	l_1	0
3	θ_2	0	l_2	0
4	θ_3	0	l_3	0

We can now get the homogeneous transformation matrices using the formula shown bellow:

$$H_n^{n-1} = \begin{bmatrix} \cos(\theta_n) & -\sin(\theta_n)\cos(\alpha_n) & \sin(\theta_n)\sin(\alpha_n) & r_n\cos(\theta_n) \\ \sin(\theta_n) & \cos(\theta_n)\cos(\alpha_n) & -\cos(\theta_n)\sin(\alpha_n) & r_n\sin(\theta_n) \\ 0 & \sin(\alpha_n) & \cos(\alpha_n) & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

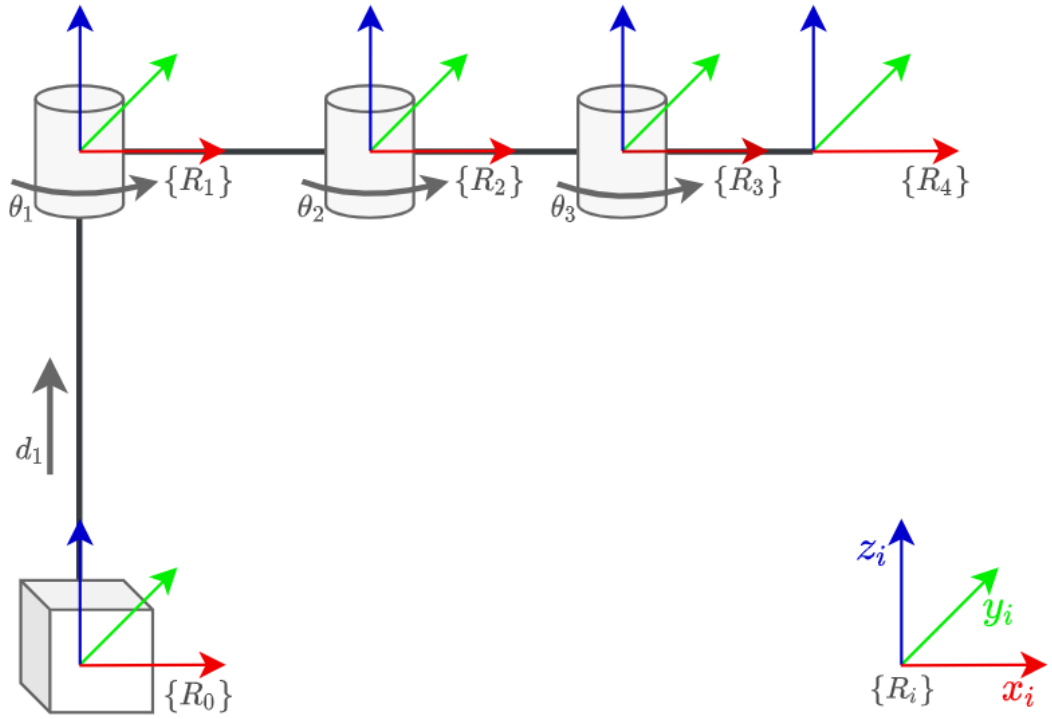

Figure 2.4: Kinematic Diagram of a PRRR SCARA

The homogeneous matrix H_1^0 is given by:

$$H_1^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

It represents translational movement without rotations, that's why the rotational part is an identity matrix.
The homogeneous matrices H_2^1, H_3^2 and H_4^3 are given by:

$$H_2^1 = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & l_1\cos(\theta_1) \\ \sin(\theta_1) & \cos(\theta_1) & 0 & l_1\sin(\theta_1) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

$$H_3^2 = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & l_2\cos(\theta_2) \\ \sin(\theta_2) & \cos(\theta_2) & 0 & l_2\sin(\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.9}$$

$$H_4^3 = \begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 & l_3\cos(\theta_3) \\ \sin(\theta_3) & \cos(\theta_3) & 0 & l_3\sin(\theta_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.10}$$

As we can see, all these homogeneous transformation matrices have both translation movement (along x and y) and rotation movement, all the rotation matrices are elementary rotation

matrices and since we have chosen to not change the orientation of **DH** frames, they are all rotations around the z axis.
The homogeneous transformation that describes the position and orientation of the end-effector with respect to the reference frame is obtained by post-multiplying the **HT** Matrices,

$$H_4^0 = H_1^0 H_2^1 H_3^2 H_4^3 \tag{2.11}$$

After computing this product and by using trigonometric identities to simplify the matrix, we get:

$$H_4^0 = \left[\begin{array}{ccc|c} \cos(\theta_{123}) & -\sin(\theta_{123}) & 0 & l_1\cos(\theta_1)+l_2\cos(\theta_{12})+l_3\cos(\theta_{123}) \\ \sin(\theta_{123}) & \cos(\theta_{123}) & 0 & l_1\sin(\theta_1)+l_2\sin(\theta_{12})+l_3\sin(\theta_{123}) \\ 0 & 0 & 1 & d_4 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{2.12}$$

The rotation matrix of the obtained homogeneous transformation is an elementary rotation around z axis, the total angle φ is the sum of all joint rotations.
The Following MATLAB function computes the homogeneous transformation matrix based on DH parameters:

```
function H = DH(Theta, Alpha, r, d)
    % Homogeneous Transformation from DH Parameters
    cT = cos(Theta);
    sT = sin(Theta);
    cA = cos(Alpha);
    sA = sin(Alpha);

    H = [cT -sT*cA sT*sA r*cT; sT cT*cA -cT*sA r*sT; 0 sA cA d; 0
        0 0 1];
end
```

This MATLAB script computes the forward kinematics equations for a PRRR SCARA manipulator using symbolic variables:

```
syms q1 q2 q3 q4 real; % Joint Variables
syms L1 L2 L3; % Link Lengths
assume([L1 L2 L3] > 0);
assumeAlso([L1 L2 L3], 'real');

H0_1 = HT(0, 0, 0, q4);
H1_2 = HT(q1, 0, L1, 0);
H2_3 = HT(q2, 0, L2, 0);
H3_4 = HT(q3, 0, L3, 0);

H0_2 = simplify(H0_1 * H1_2);
H0_3 = simplify(H0_2 * H2_3);
H0_4 = simplify(H0_3 * H3_4)
```

The output of this script is shown in figure 2.5 and it corresponds to the results we have obtained in 2.12.

2.2 Inverse Kinematics

Inverse Kinematics is the problem of getting the joint variables that takes the end-effector to the desired position and orientation, i.e. getting q_i such as $q = f^{-1}(p)$. There is no systematic way

```
H0_4 =

[ cos(q1 + q2 + q3), -sin(q1 + q2 + q3), 0, L2*cos(q1 + q2) + L1*cos(q1) + L3*cos(q1 + q2 + q3)]
[ sin(q1 + q2 + q3),  cos(q1 + q2 + q3), 0, L2*sin(q1 + q2) + L1*sin(q1) + L3*sin(q1 + q2 + q3)]
[                 0,                  0, 1,                                                   q4]
[                 0,                  0, 0,                                                    1]
```

Figure 2.5: Forward Kinematics Obtained Using MATLAB

of obtaining a solution to this problem, for simple robot configurations the inverse kinematics can be determined by exploring the geometry of the robot, and using trigonometric identities. However, for complex robots there are numerical algorithms that allow us to get the inverse kinematics, such as, Newton-Raphson algorithm.
As opposed to forward kinematics, the computation of inverse kinematics is quite complex for serial manipulators, for the following reasons:

- The inverse kinematics equations are in general highly nonlinear, and thus it is not always possible to find an analytical solution.
- There might be multiple solutions, (we will see an example in the next section).
- Infinite solutions may exist, that is true for kinematically redundant manipulators.
- The solution obtained are non admissible, as a result of manipulator structure.

The solution for this problem is crucial. Because in order to program a robot to do a task in operational space, we will provide only coordinates in Cartesian space, the arm however only understands the joint variables. So the controller of the arm must contain an inverse kinematics solver built-in to it.

2.2.1 Two Links Planar Robot

We will start by a relatively simple robot structure, solving this one will become handy and useful to obtain the inverse kinematics of other robot manipulators structures.
The two links planar robot structure is shown in figure 2.6.
Using Pythagorean theorem we have:

$$r^2 = x^2 + y^2 \tag{2.13}$$

Recalling al-Kashi's theorem also known as the cosine rule, in the triangle shown in figure 2.7

$$c^2 = a^2 + b^2 - 2ab\cos(\gamma) \tag{2.14}$$

By using 2.14 we have:

$$\begin{aligned} r^2 &= l_1^2 + l_2^2 - 2l_1l_2\cos(\alpha) \\ \cos(\alpha) &= \frac{l_1^2 + l_2^2 - r^2}{2l_1l_2} \\ \cos(\alpha) &= \frac{l_1^2 + l_2^2 - x^2 - y^2}{2l_1l_2} \end{aligned} \tag{2.15}$$

It's easy to see that $\alpha = \pi - \theta_2$, and since $\cos(\pi - x) = -\cos(x)$ we get:

$$cos(\theta_2) = \frac{x^2 + y^2 - l_1^2 - l_2^2}{2l_1l_2} \tag{2.16}$$

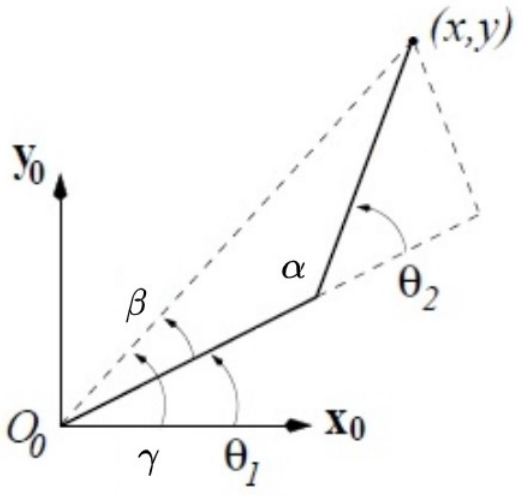

Figure 2.6: Two links Planar Robot

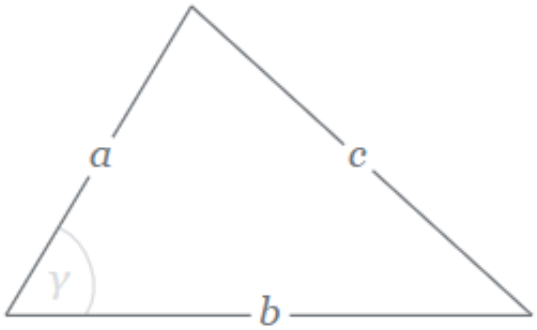

Figure 2.7: Triangle to Visualize the Cosine Rule

This result is particularly interesting, we know that the cosine is a pair function i.e. $\cos(x) = \cos(-x)$. in other words, if $y = \cos(x)$ then $x = \arccos(y)$ or $x = -\arccos(y)$. This means that we have two possible solution for the inverse kinematics of this robot manipulator configuration. The first solution is called elbow down pose, and the other solution is called elbow up pose, we can see this geometrically as depicted in figure 2.8,
We will continue this analysis by choosing explicitly the elbow up pose, that is

$$\theta_2 = \arccos\left(\frac{x^2 + y^2 - l_1^2 - l_2^2}{2l_1 l_2}\right) \tag{2.17}$$

We should now get the joint variable θ_1. We notice that:

$$\theta_1 = \gamma - \beta \tag{2.18}$$

The angle γ is a function of x and y,

$$\gamma = \arctan2(y,\ x) \tag{2.19}$$

In order to get θ_1 we should first get the expression of β. From figure 2.8.A we have:

$$\tan(\beta) = \frac{l_2 \sin(\theta_2)}{l_1 + l_2 \cos(\theta_2)} \tag{2.20}$$

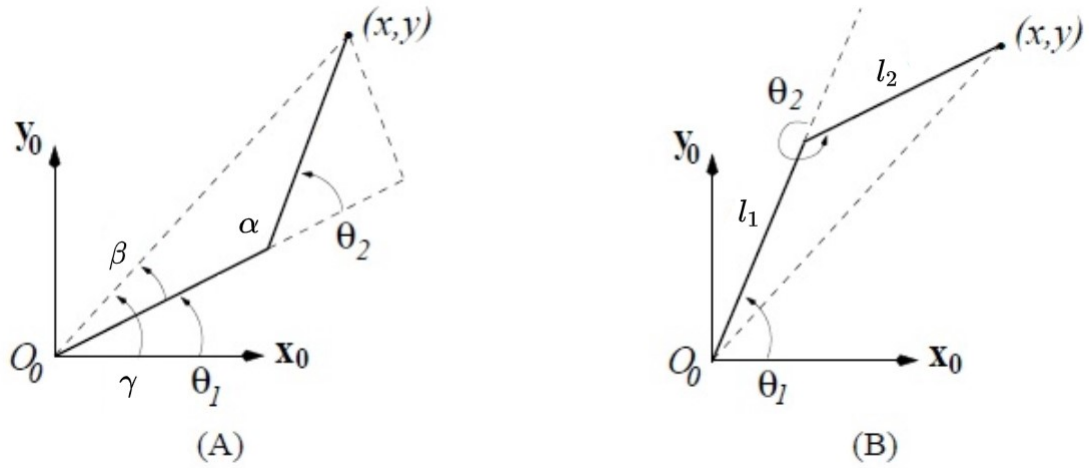

Figure 2.8: Elbow up vs Elbow Down

Thus, the expression of the joint variable θ_1 is

$$\theta_1 = \arctan2(y, x) - \arctan2(l_2 \sin(\theta_2),\ l_1 + l_2 \cos(\theta_2)) \tag{2.21}$$

We finally have:

$$\begin{cases} \theta_2 = \arccos\left(\dfrac{x^2 + y^2 - l_1^2 - l_2^2}{2 l_1 l_2}\right) \\ \theta_1 = \arctan2(y, x) - \arctan2(l_2 \sin(\theta_2),\ l_1 + l_2 \cos(\theta_2)) \end{cases} \tag{2.22}$$

Now we should study the second solution, Elbow Up pose, θ_2 is then given by:

$$\theta_2 = -\arccos\left(\frac{x^2 + y^2 - l_1^2 - l_2^2}{2 l_1 l_2}\right) \tag{2.23}$$

We notice from figure 2.8.B that

$$\theta_1 = \delta \ + \ \beta \tag{2.24}$$

the equation (2.20) still holds, thus joint angle θ_1 is,

$$\theta_1 = \arctan2(y, x) + \arctan2(l_2 \sin(\theta_2),\ l_1 + l_2 \cos(\theta_2)) \tag{2.25}$$

The inverse kinematics equation of this pose is:

$$\begin{cases} \theta_2 = -\arccos\left(\dfrac{x^2 + y^2 - l_1^2 - l_2^2}{2 l_1 l_2}\right) \\ \theta_1 = \arctan2(y, x) + \arctan2(l_2 \sin(\theta_2),\ l_1 + l_2 \cos(\theta_2)) \end{cases} \tag{2.26}$$

We notice that the first and second solutions are quite similar, we wish writing them in a single more general form, by using the $\pm, \mp$ notation. basically when $\pm$ is $+$ then, $\mp$ is $-$ and vice versa. So we can safely write:

$$\begin{cases} \theta_2 = \pm\arccos\left(\dfrac{x^2 + y^2 - l_1^2 - l_2^2}{2 l_1 l_2}\right) \\ \theta_1 = \arctan2(y, x) \pm \arctan2(l_2 \sin(\theta_2),\ l_1 + l_2 \cos(\theta_2)) \end{cases} \tag{2.27}$$

2.2.2 Three Links Planar Robot

The joint angle θ_3 can be expressed as a function of the orientation of the end-effector φ and the joint angles θ_1 and θ_2:

$$\varphi = \theta_1 + \theta_2 + \theta_3 \iff \theta_3 = \varphi - \theta_1 - \theta_2 \tag{2.28}$$

Let $\bar{x}$ and $\bar{y}$ be :

$$\left\langle \begin{matrix} \bar{x} = x - l_3\ \cos(\varphi) \\ \bar{y} = y - l_3\ \sin(\varphi) \end{matrix} \right\rangle \tag{2.29}$$

This variable change reduces the problem into a smaller one, that is computing the inverse kinematics of two link planar robot with x and y being $\bar{x}$ and $\bar{y}$ respectively.
The inverse kinematics of elbow down pose is given by

$$\theta_2 = \arccos\left(\frac{\bar{x}^2 + \bar{y}^2 - l_1^2 - l_2^2}{2l_1 l_2}\right) \tag{2.30}$$

$$\theta_1 = \arctan2(\bar{y},\ \bar{x}) - arctan2\left(l_2 \sin(\theta_2),\ l_1 + l_2\cos(\theta_2)\right) \tag{2.31}$$

$$\theta_3 = \varphi - \theta_1 - \theta_2 \tag{2.32}$$

The inverse kinematics of elbow up pose is,

$$\theta_2 = -\arccos\left(\frac{\bar{x}^2 + \bar{y}^2 - l_1^2 - l_2^2}{2l_1 l_2}\right) \tag{2.33}$$

$$\theta_1 = \arctan2(\bar{y},\ \bar{x}) + \arctan2\left(l_2 \sin(\theta_2),\ l_1 + l_2\cos(\theta_2)\right) \tag{2.34}$$

$$\theta_3 = \varphi - \theta_1 - \theta_2 \tag{2.35}$$

We can write the solution in a compact form using $\pm$ and $\mp$ notations,

$$\begin{cases} \theta_2 = \pm \arccos\left(\frac{\bar{x}^2 + \bar{y}^2 - l_1^2 - l_2^2}{2l_1 l_2}\right) \\ \theta_1 = \arctan 2(\bar{y},\ \bar{x}) \mp \arctan 2\left(l_2 \sin(\theta_2),\ l_1 + l_2\cos(\theta_2)\right) \\ \theta_3 = \varphi - \theta_1 - \theta_2 \end{cases} \tag{2.36}$$

Chapter 3

Velocity Kinematics

Now that we know the position and orientation of the end effector, we should figure out a way to get the operational velocities based on the joint velocities.

3.1 Geometrical Jacobian

The goal of differential kinematics is to find the relationship between the joint velocities and the end-effector linear and angular velocities. In other words, it is desired to express the end-effector linear velocity $\dot{p}$ and angular velocity ω as a function of the joint velocities $\dot{q}$ by means of the following relations:

$$\dot{p} = J_p(q)\,\dot{q} \tag{3.1}$$

$$\omega = J_o(q)\,\dot{q} \tag{3.2}$$

In compact form 3.1 and 3.2 can be written as:

$$\xi = \begin{bmatrix} \dot{p} \\ \omega \end{bmatrix} = J(q)\,\dot{q} \tag{3.3}$$

The J matrix is called the geometric Jacobian it is a 6 by n matrix, where 6 is the number of possible velocities $(\dot{x}, \dot{y}, \dot{z}, \omega_x, \omega_y, \omega_z)^T$ and n is the number of joints in the robot.
The geometrical Jacobian computation depends on the joint type.
If the joint is prismatic we have:

$$J_i = \begin{bmatrix} R_{i-1}^0 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.4}$$

For a revolute joint, the Jacobian is computed as:

$$J_i = \begin{bmatrix} R_{i-1}^0 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \times \left(d_n^0 - d_{i-1}^0\right) \\ R_{i-1}^0 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \end{bmatrix} \tag{3.5}$$

with d_i^0 being the translational part of the homogeneous matrix H_i^0.

In order to get the geometrical Jacobian for SCARA robot, we need to compute the homogeneous matrices from frame 0 to frame i, $i = \overline{1,4}$:

$$H_0^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

$$H_1^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

$$H_2^0 = H_1^0 H_2^1 = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & l_1 c\theta_1 \\ s\theta_1 & c\theta_1 & 0 & l_1 s\theta_1 \\ 0 & 0 & 1 & d4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

$$H_3^0 = H_2^0 H_3^2 = \begin{bmatrix} c\theta_{12} & -s\theta_{12} & 0 & l_1 c\theta_1 + l_2 c\theta_{12} \\ s\theta_{12} & c\theta_{12} & 0 & l_1 s\theta_1 + l_2 s\theta_{12} \\ 0 & 0 & 1 & d4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.9}$$

Replacing each column of the Jacobian with its component would result in:

$$\xi = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \hline \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} = \left[\begin{array}{c|c|c|c} R_0^0 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} & R_1^0 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \times (d_4^0 - d_1^0) & R_2^0 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \times (d_4^0 - d_2^0) & R_3^0 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \times (d_4^0 - d_3^0) \\ \hline 0 \\ 0 \\ 1 & R_1^0 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} & R_2^0 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} & R_3^0 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \end{array}\right] \begin{pmatrix} \dot{d}_4 \\ \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{pmatrix} \tag{3.10}$$

Evaluating this matrix gives us the final expression of the geometrical Jacobian:

$$J = \left[\begin{array}{c|c|c|c} 0 & -l_1 s\theta_1 - l_2 s\theta_{12} - l_3 s\theta_{123} & l_2 s\theta_{12} - l_3 s\theta_{123} & -l_3 s\theta_{123} \\ 0 & l_1 c\theta_1 + l_2 c\theta_{12} + l_3 c\theta_{123} & l_2 c\theta_{12} + l_3 c\theta_{123} & l_3 c\theta_{123} \\ 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{array}\right] \tag{3.11}$$

As we can see the dimension of this Jacobian is 4×6 matrix, that's because we only have 4 joints, We notice that the $4th$ and $5th$ rows are null, and that's evident as we have no control over rotations around x and y axis. Since the Jacobian matrix is not square, it doesn't admit an inverse. for that we remove the $4th$ and $5th$ rows as they are useless to us anyways. Now we get a square Jacobian matrix

$$J_{4\times4} = \begin{bmatrix} 0 & -l_1 s\theta_1 - l_2 s\theta_{12} - l_3 s\theta_{123} & l_2 s\theta_{12} - l_3 s\theta_{123} & -l_3 s\theta_{123} \\ 0 & l_1 c\theta_1 + l_2 c\theta_{12} + l_3 c\theta_{123} & l_2 c\theta_{12} + l_3 c\theta_{123} & l_3 c\theta_{123} \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \tag{3.12}$$

From now on, whenever we write J, we refer to $J_{4\times4}$.
To ease the computation, we have created a MATLAB script, that can compute the geometrical Jacobian of any given robot manipulator's configuration.
Jacobian column MATLAB function:

```
function J = Ji(H0_i, H0_n, jointType)
     % Get Jacobian Column i
     d = H0_n(1:3, 4) - H0_i(1:3, 4);  d_n^0 - d_i-1^0
     Ri = H0_i(1:3, 1:3);  R_i-1^0
     R = Ri(:, 3);

     if jointType == 'P' || jointType == 'p'
          J = [R; [0; 0; 0]];
     elseif jointType == 'R' || jointType == 'r'
          J = [cross(R, d); R];
     end
 end
```

MATLAB script to get SCARA Jacobian

```
%% Compute Homogeneous Transformations
HomogeneousTransformations

%% Jacobian Columns
J1 = Ji(H0_0, H0_4, 'P');
J2 = Ji(H0_1, H0_4, 'R');
J3 = Ji(H0_2, H0_4, 'R');
J4 = Ji(H0_3, H0_4, 'R');
J_full = [J1 J2 J3 J4];  6x4 Jacobian
J = [J_full(1:3, :); J_full(6, :)];  4x4 Jacobian
invJ = simplify(inv(J));  J^-1

%% Symbolic Computation
syms dx dy dz wx wy wz real
syms dq1 dq2 dq3 dq4 real

xi_full = [dx; dy; dz; wx; wy; wz];
xi = [dx; dy; dz; wz];
dq = [dq4; dq1; dq2; dq3];

%% Jacobian
xi_full = J_full*dq;
xi = J*dq

%% Inverse Jacobian
dq = invJ*xi
```

3.2 Analytical Jacobian

There exist another type of Jacobian, that is the analytical Jacobian, it is computed via differentiation of the direct kinematics function with respect to the joint variables.
The translational velocity of the end-effector is the time derivative of vector p, that is:

$$\dot{p} = \frac{\partial p}{\partial q}\dot{q} = J_p(q)\dot{q} \tag{3.13}$$

The above sections have shown the way to compute the end-effector velocity in terms of the velocity of the end-effector frame. The Jacobian is computed by following a geometric technique in which the contributions of each joint velocity to the components of end-effector linear and angular velocity are determined.
If the end-effector position and orientation are specified in terms of a minimal number of parameters in the operational space, it is natural to ask whether it is possible to compute the Jacobian via differentiation of the direct kinematics function with respect to the joint variables. To this purpose, below an analytical technique is presented to compute the Jacobian, and the existing relationship between the two Jacobians is found.
The translational velocity of the end-effector frame can be expressed as the time derivative of vector p, representing the origin of the end-effector frame with respect to the base frame, i.e.

$$\dot{p} = \frac{\partial p}{\partial q}\dot{q} = J_p(q)\dot{q} \tag{3.14}$$

For what concerns the rotational velocity of the end-effector frame, the minimal representation of orientation in terms of three variables ϕ can be considered. Its time derivative $\dot{\phi}$ in general differs from the angular velocity vector defined above. In any case, once the function $\phi(q)$ is known, it is formally correct to consider the Jacobian obtained as:

$$\dot{\phi} = \frac{\partial \phi}{\partial q}\dot{q} = J_\phi(q)\dot{q} \tag{3.15}$$

Applying this to the PRRR SCARA robot would give

$$\begin{cases} \dot{x} = -\dot{\theta}_1 \left(l_1 s\theta_1 + l_2 s\theta_{12} + l_3 s\theta_{123}\right) - \dot{\theta}_2 \left(l_2 s\theta_{12} + l_3 s\theta_{123}\right) - \dot{\theta}_3 \left(l_3 s\theta_{123}\right) \\ \dot{y} = \dot{\theta}_1 \left(l_1 c\theta_1 + l_2 c\theta_{12} + l_3 c\theta_{123}\right) + \dot{\theta}_2 \left(l_2 c\theta_{12} + l_3 c\theta_{123}\right) + \dot{\theta}_3 \left(l_3 c\theta_{123}\right) \\ \dot{z} = \dot{d}_4 \\ \dot{\phi} = \dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_3 \end{cases} \tag{3.16}$$

3.3 Inverse Jacobian

In the previous sections we wanted to get the end effector velocities as a function of joint velocities, in this section we will be interested in getting the joint velocities that correspond to the operational velocities. we can do so by pre-multiplying (3.3) by the inverse of the Jacobian matrix

$$J^{-1}\xi = J^{-1}J\dot{q} \tag{3.17}$$

We then get,

$$\dot{q} = J^{-1}\xi \tag{3.18}$$

$$\begin{pmatrix} \dot{d}_4 \\ \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{pmatrix} = J^{-1} \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_z \end{pmatrix} \tag{3.19}$$

The computation of the inverse Jacobian matrix was done in MATLAB, using the previous

script, The expression of the inverse Jacobian matrix is given by:

$$J^{-1} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ \frac{c\theta_{12}}{l_1 s\theta_2} & \frac{s\theta_{12}}{l_1 s\theta_2} & 0 & \frac{l_3 s\theta_3}{l_1 s\theta_2} \\ \frac{-l_2 c\theta_{12} - l_1 c\theta_1}{l_1 l_2 s\theta_2} & \frac{-l_2 s\theta_{12} - l_1 s\theta_1}{l_1 l_2 s\theta_2} & 0 & -l_3 \frac{l_1 s\theta_{23} + l_2 s\theta_3}{l_1 l_2 s\theta_2} \\ \frac{c\theta_1}{l_2 s\theta_2} & \frac{s\theta_1}{l_2 s\theta_2} & 0 & \frac{l_3 s\theta_{23} + l_2 s\theta_2}{l_2 s\theta_2} \end{bmatrix} \tag{3.20}$$

Thus,

$$\begin{pmatrix} \dot{d}_4 \\ \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{pmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ \frac{c\theta_{12}}{l_1 s\theta_2} & \frac{s\theta_{12}}{l_1 s\theta_2} & 0 & \frac{l_3 s\theta_3}{l_1 s\theta_2} \\ \frac{-l_2 c\theta_{12} - l_1 c\theta_1}{l_1 l_2 s\theta_2} & \frac{-l_2 s\theta_{12} - l_1 s\theta_1}{l_1 l_2 s\theta_2} & 0 & -l_3 \frac{l_1 s\theta_{23} + l_2 s\theta_3}{l_1 l_2 s\theta_2} \\ \frac{c\theta_1}{l_2 s\theta_2} & \frac{s\theta_1}{l_2 s\theta_2} & 0 & \frac{l_3 s\theta_{23} + l_2 s\theta_2}{l_2 s\theta_2} \end{bmatrix} \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \omega_z \end{pmatrix} \tag{3.21}$$

Chapter 4

Dynamics

Before, we were only studying the kinematics of the robot, that is both operational and joint positions and velocities without taking into account what have caused the movement. In this chapter we will be interested in the dynamics of robot manipulators, i.e. the torques, forces, accelerations, etc. This is a crucial part since we will be using the dynamics to control the arm, since naturally it's unstable.
There are different ways of obtaining the dynamics model, the two main ways are Lagrangian dynamics and Newton-Euler formulation.
We will be using Lagrangian dynamics since it is conceptually elegant and quite effective for robots with simple structures. But we will see after, that the calculation can quickly be cumbersome.

4.1 Lagrangian Formulation

The first step in the Lagrangian formulation of dynamics is to choose a set of independent coordinates $q \in \mathbb{R}^n$ that fully describes the system's configuration. The coordinates q are called generalized coordinates. Once generalized coordinates have been chosen, these then define the generalized forces $\mathcal{T} \in \mathbb{R}^n$. A Lagrangian function $\mathcal{L}(q, \dot{q})$ is then defined as the overall system's kinetic energy $T(q, \dot{q})$ minus the potential energy $U(q)$.

$$\mathcal{L}(q, \dot{q}) = T(q, \dot{q}) - U(q) \tag{4.1}$$

The equations of motion are obtained using Euler-Lagrange equation:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}}\right) - \frac{\partial \mathcal{L}}{\partial q} = \mathcal{T} - \frac{\partial \mathcal{D}}{\partial \dot{q}} \tag{4.2}$$

Where $\mathcal{D}$ is the Rayleigh dissipation function, it is defined as:

$$\mathcal{D} = \sum_{i=1}^{n} \frac{1}{2} \beta_i \, \dot{\theta}_i^{\,2} \tag{4.3}$$

4.2 Single Joint Robot

Consider a single joint robot shown in figure 4.1, where l_g is the distance between the pivot point and the center of mass (represented by the checkered circle), ϕ is the angle between the link and the center of mass and finally θ is the joint position. This robot has a mass m and a moment of inertia about the center of mass I.

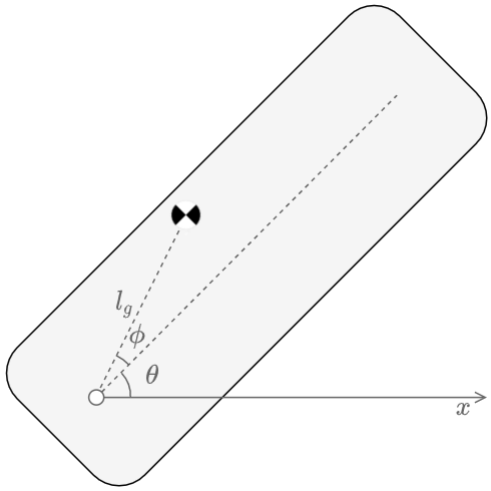

Figure 4.1: Single Joint Robot

The position of center of mass of the link is given by:

$$\begin{cases} x = l_g \cos(\theta + \phi) \\ y = l_g \sin(\theta + \phi) \end{cases} \tag{4.4}$$

Differentiating the position with respect to time would result in the linear velocities, that is:

$$\begin{cases} \dot{x} = -l_g \dot{\theta} \sin(\theta + \phi) \\ \dot{y} = l_g \dot{\theta} \cos(\theta + \phi) \end{cases} \tag{4.5}$$

The Kinetic energy of the link is composed by the kinetic energy of translational movement and the one of rotational movement, like so

$$T = \frac{1}{2} m v^2 + \frac{1}{2} I \dot{\theta}^2 \tag{4.6}$$

$$T = \frac{1}{2} m \left(\dot{x}^2 + \dot{y}^2 \right) + \frac{1}{2} I \dot{\theta}^2 \tag{4.7}$$

$$T = \frac{1}{2} m \left(l_g^2 \, \dot{\theta}^2 \sin^2(\theta + \phi) + l_g^2 \, \dot{\theta}^2 \cos^2(\theta + \phi) \right) + \frac{1}{2} I \dot{\theta}^2 \tag{4.8}$$

We then get

$$T = \frac{1}{2} m l_g^2 \, \dot{\theta}^2 + \frac{1}{2} I \dot{\theta}^2 \tag{4.9}$$

$$T = \frac{1}{2} M \dot{\theta}^2 \tag{4.10}$$

Where

$$\left\langle M = m \, l_g^2 + I \right\rangle \tag{4.11}$$

The potential energy of this link is,

$$U = mgy = mg \, l_g \sin(\theta + \phi) \tag{4.12}$$

The Rayleigh dissipation function of this robot is the following:

$$D = \frac{1}{2}\beta\,\dot{\theta}^2 \tag{4.13}$$

We compute the Lagrangian to use it to get the equation of motion

$$\mathcal{L} = T - U \tag{4.14}$$

$$\mathcal{L} = \frac{1}{2}M\dot{\theta}^2 - mgl_g\sin(\theta+\phi) \tag{4.15}$$

Using Euler-Lagrange equation we have

$$\frac{d}{dt}\left(\frac{\partial\mathcal{L}}{\partial\dot{\theta}}\right) - \frac{\partial\mathcal{L}}{\partial\theta} = \mathcal{T} - \frac{\partial\mathcal{D}}{\partial\dot{\theta}} \tag{4.16}$$

After computation we get the inverse dynamics equation

$$\mathcal{T} = M\,\ddot{\theta} + \beta\,\dot{\theta} + mg\,l_g\cos(\theta+\phi) \tag{4.17}$$

With some algebraic manipulation we get the forward dynamics equation

$$\ddot{\theta} = \frac{1}{ml_g^2 + I_{zz}}\Big(\mathcal{T} - \beta\,\dot{\theta} - mg\,l_g\cos(\theta+\phi)\Big) \tag{4.18}$$

Another interesting representation is the state space form, that is

$$\begin{cases} \dot{x} = f(x) + g(x)u \\ y = h(x) \end{cases} \tag{4.19}$$

For that we set the following variables:

$$\left\langle \begin{array}{l} x_1 = \theta \\ x_2 = \dot{x_1} \\ u = \mathcal{T} \end{array} \right\rangle \Longrightarrow \left\langle \begin{array}{l} \theta = x_1 \\ \dot{\theta} = x_2 \\ \ddot{\theta} = \dot{x_2} \end{array} \right\rangle \tag{4.20}$$

Finally, the state space model is given by:

$$\begin{cases} \dot{x_1} = x_2 \\ \dot{x_2} = \dfrac{1}{ml_g^2 + I_{zz}}\left(-mg\,l_g\cos(x_1+\phi) - \beta\,x_2 + u\right) \end{cases}$$

In order to simulate the dynamics of this robot we have created a SIMULINK model shown in figure 4.2

4.3 Two Links Planar Robot

The position of the center of mass of links 1 and 2 are given by:

$$\begin{cases} x_1 = l_{g_1}\mathrm{c}(\theta_1+\phi_1) \\ y_1 = l_{g_1}\mathrm{s}(\theta_1+\phi_1) \end{cases} \qquad \begin{cases} x_2 = l_1\mathrm{c}(\theta_1) + l_{g_2}\mathrm{c}(\theta_1+\theta_2+\phi_2) \\ y_2 = l_1\mathrm{s}(\theta_1) + l_{g_2}\mathrm{s}(\theta_1+\theta_2+\phi_2) \end{cases} \tag{4.21}$$

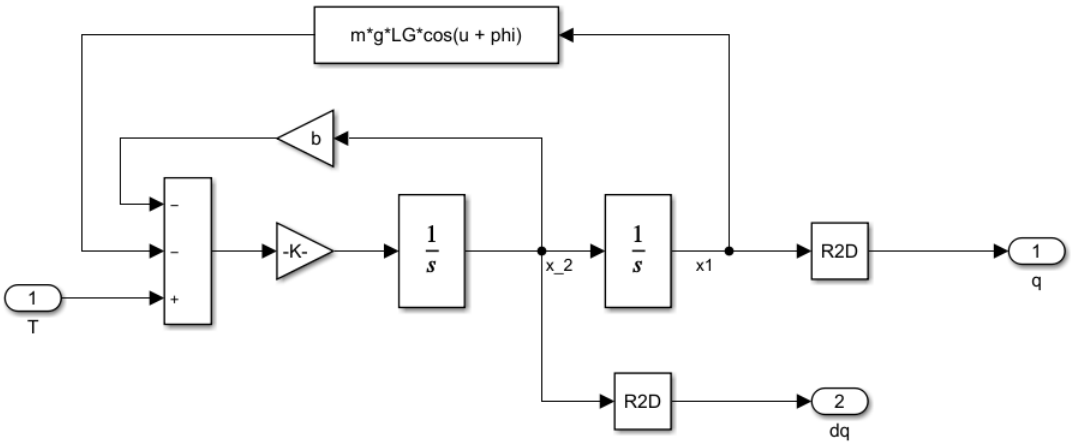

Figure 4.2: Implementation of Single Joint Robot in SIMULINK

We get the velocities of link 1 and link 2 by deriving each position with respect to time

$$\begin{cases} \dot{x_1} = -l_{g_1}\dot{\theta_1}\mathrm{s}(\theta_1+\phi_1) \\ \dot{y_1} = l_{g_1}\dot{\theta_1}\mathrm{c}(\theta_1+\phi_1) \end{cases} \qquad \begin{cases} \dot{x_2} = -l_1\dot{\theta_1}\mathrm{s}(\theta_1) - l_{g_2}(\dot{\theta_1}+\dot{\theta_2})\ \mathrm{s}(\theta_1+\theta_2+\phi_2) \\ \dot{y_2} = l_1\dot{\theta_1}\mathrm{c}(\theta_1) + l_{g_2}(\dot{\theta_1}+\dot{\theta_2})\ \mathrm{c}(\theta_1+\theta_2+\phi_2) \end{cases} \tag{4.22}$$

We will need the square of velocities to compute the kinetic energy:

$$v_1^2 = \dot{x_1}^2 + \dot{y_1}^2 = l_{g_1}^2\dot{\theta_1}^2 \tag{4.23}$$

$$v_2^2 = \dot{x_2}^2 + \dot{y_2}^2 = \left(l_1^2 + l_{g_2}^2 + 2l_1l_{g_2}\mathrm{c}(\theta_2+\phi_2)\right)\dot{\theta_1}^2 + l_{g_1}^2\dot{\theta_2}^2 + 2\left(l_{g_2}^2 + l_1l_{g_2}\mathrm{c}(\theta_2+\phi_2)\right)\dot{\theta_1}\dot{\theta_2} \tag{4.24}$$

We choose the joint coordinates θ_1 and θ_2 as the generalized coordinates. The generalized forces $\mathcal{T}_1$ and $\mathcal{T}_2$ then correspond to joint torques.

Kinetic Energy

The kinetic energy of each link is the sum of the kinetic energy of translational kinetic energy and rotational kinetic energy:

$$T_1 = \frac{1}{2}m_1v_1^2 + \frac{1}{2}I_1\dot{\theta_1}^2 \tag{4.25}$$

$$T_2 = \frac{1}{2}m_2v_2^2 + \frac{1}{2}I_2\left(\dot{\theta_1}^2 + \dot{\theta_1}^2\right) \tag{4.26}$$

T_1 and T_2 are then given by:

$$T_1 = \frac{1}{2}\left(ml_{g_1}^2 + I_1\right)\dot{\theta_1}^2 \tag{4.27}$$

$$\begin{aligned} T_2 = {} & \frac{1}{2}\left(I_2 + m_2l_1^2 + m_2l_{g_2}^2 + 2m_2l_1l_{g_2}\mathrm{c}(\theta_2+\phi_2)\right)\dot{\theta_1}^2 + \frac{1}{2}\left(I_2 + m_2l_{g_2}^2\right)\dot{\theta_2}^2 \\ & + \left(m_2l_{g_2}^2 + m_2l_1l_{g_2}\mathrm{c}(\theta_2+\phi_2)\right)\dot{\theta_1}\dot{\theta_2} \end{aligned} \tag{4.28}$$

The total kinetic energy is the sum of the kinetic energies of each link:

$$T = T_1 + T_2 \tag{4.29}$$

$$\begin{aligned} T = {} & \frac{1}{2}\left(I_1 + I_2 + m_2l_1^2 + m_1l_{g_1}^2 + m_2l_{g_2}^2 + 2m_2l_1l_{g_2}\mathrm{c}(\theta_2+\phi_2)\right)\dot{\theta_1}^2 + \frac{1}{2}\left(I_2 + m_2l_{g_2}^2\right)\dot{\theta_2}^2 + \\ & \left(m_2l_{g_2}^2 m_2l_1l_{g_2}\mathrm{c}(\theta_2+\phi_2)\right)\dot{\theta_1}\dot{\theta_2} \end{aligned} \tag{4.30}$$

Potential Energy

The potential Energies in the other hand is relatively straight forward to get:

$$U_1 = m_1 g y_1 = m_1 g \, l_{g_1} \sin(\theta_1 + \phi_1) \tag{4.31}$$

$$U_2 = m_2 g y_2 = m_2 g \left(l_1 \sin(\theta_1) + l_{g_2} \sin(\theta_1 + \theta_2 + \phi_2) \right) \tag{4.32}$$

Similarly to kinetic energy, the total potential energy is the sum of each link potential energy

$$U = U_1 + U_2 \tag{4.33}$$

$$U = m_1 g \, l_{g_1} \sin(\theta_1 + \phi_1) + m_2 g \left(l_1 \sin(\theta_1) + l_{g_2} \sin(\theta_1 + \theta_2 + \phi_2) \right) \tag{4.34}$$

This is the potential energy of a two links planar robot in a vertical plan, which is the general form, since we can get the expression of potential energy of a two link planar robot in a horizontal plan just by setting the gravity acceleration g to be null!

Rayleigh Dissipation Function

Here we assume to have only viscous friction at each joint the Rayleigh dissipation function is:

$$\mathcal{D} = \sum_{i=1}^{2} \frac{1}{2} \beta_i \, \dot{\theta_i}^2 \tag{4.35}$$

$$\mathcal{D} = \frac{1}{2} \left(\beta_1 \dot{\theta_1}^2 + \beta_2 \dot{\theta_2}^2 \right) \tag{4.36}$$

Where β_1 and β_2 are the friction coefficients of joint one and two respectively.

Lagrangian

The Lagrangian is given by:

$$\mathcal{L} = T - U \tag{4.37}$$

$$\begin{aligned} \mathcal{L} = & \frac{1}{2} \left(I_1 + I_2 + m_2 l_1^2 + m_1 l_{g_1}^2 + m_2 l_{g_2}^2 + 2 m_2 l_1 l_{g_2} \cos(\theta_2 + \phi_2) \right) \dot{\theta_1}^2 + \frac{1}{2} \left(I_2 + m_2 l_{g_2}^2 \right) \dot{\theta_2}^2 \\ & + \left(m_2 l_{g_2}^2 + m_2 l_1 l_{g_2} \cos(\theta_2 + \phi_2) \right) \dot{\theta_1} \dot{\theta_2} - m_1 g \, l_{g_1} \sin(\theta_1 + \phi_1) \\ & - m_2 g \left(l_1 \sin(\theta_1) + l_{g_2} \sin(\theta_{12} + \phi_2) \right) \end{aligned} \tag{4.38}$$

Euler-Lagrange

In order to get the equations of motion we should compute the Euler-Lagrange equation

$$\frac{d}{dt} \left(\frac{\partial \mathcal{L}}{\partial \dot{\theta}_i} \right) - \frac{\partial \mathcal{L}}{\partial \theta_i} = \mathcal{T}_i - \frac{\partial \mathcal{D}}{\partial \dot{\theta}_i} \; ; \quad i = 1, 2 \tag{4.39}$$

$$\begin{cases} \dfrac{\partial \mathcal{L}}{\partial \dot{\theta}_1} = \left(I_1 + I_2 + m_2 l_1^2 + m_1 l_{g_1}^2 + m_2 l_{g_1}^2 + 2 \, m_2 l_1 l_{g_1} \mathrm{c}(\theta_2 + \phi_2) \right) \dot{\theta}_1 + \left(m_2 l_{g_1}^2 + m_2 l_1 l_{g_2} \mathrm{c}(\theta_2 + \phi_2) \right) \dot{\theta}_2 \\ \dfrac{\partial \mathcal{L}}{\partial \dot{\theta}_2} = \left(m_2 l_{g_2}^2 l_1 l_{g_2} \cos(\theta_2 + \phi_2) \right) \dot{\theta}_1 + \left(I_2 + l_{g_2}^2 \right) \dot{\theta}_2 \end{cases} \tag{4.40}$$

$$\begin{cases} \dfrac{d}{dt} \left(\dfrac{\partial \mathcal{L}}{\partial \dot{\theta}_1} \right) = & \left(I_1 + I_2 + m_2 l_1^2 + m_1 l_{g_1}^2 + m_2 l_{g_2}^2 + 2 m_2 l_1 l_{g_2} \cos(\theta_2 + \phi_2) \right) \ddot{\theta}_1 \\ & + \left(m_2 l_{g_2}^2 + m_2 l_1 l_{g_2} \cos(\theta_2 + \phi_2) \right) \ddot{\theta}_2 \\ & - \left(m_2 l_1 l_{g_2} \sin(\theta_2 + \phi_2) \right) \dot{\theta_2}^2 - \left(2 m_2 l_1 l_{g_2} \sin(\theta_2 + \phi_2) \right) \dot{\theta}_1 \dot{\theta}_2 \\ \dfrac{d}{dt} \left(\dfrac{\partial \mathcal{L}}{\partial \dot{\theta}_2} \right) = & \left(m_2 l_{g_2}^2 + m_2 l_1 l_{g_2} \cos(\theta_2 + \phi_2) \right) \ddot{\theta}_1 + \left(I_2 + m_2 l_{g_2}^2 \right) \ddot{\theta}_2 - \left(l_1 l_{g_2} \sin(\phi_2 + \theta_2) \right) \dot{\theta}_1 \dot{\theta}_2 \end{cases} \tag{4.41}$$

$$\begin{cases} \dfrac{\partial \mathcal{D}}{\partial \dot{\theta}_1} = \beta_1 \dot{\theta}_1 \\ \dfrac{\partial \mathcal{D}}{\partial \dot{\theta}_2} = \beta_2 \dot{\theta}_2 \end{cases} \tag{4.42}$$

Finally we get the inverse dynamics equations, that is the generalized torques:

$$\begin{cases} \mathcal{T}_1 = & \left(I_1 + I_2 + m_2 l_1^2 + m_1 l_{g_1}^2 + m_2 l_{g_2}^2 + 2 m_2 l_1 l_{g_2} \cos(\theta_2 + \phi_2)\right) \ddot{\theta}_1 + \left(m_2 l_{g_2}^2 + 2 m_2 l_1 l_{g_2} \cos(\theta_2 + \phi_2)\right) \ddot{\theta}_2 \\ & - \left(m_2 l_1 l_{g_2} \sin(\theta_2 + \phi_2)\right) \dot{\theta_2}^2 - \left(2 m_2 l_1 l_{g_2} \cos(\theta_2 + \phi_2)\right) \dot{\theta}_1 \dot{\theta}_2 \\ & + \beta_1 \dot{\theta}_1 + m_2 g\, l_{g_2} \cos(\theta_{12} + \phi_2) + m_1 g\, l_{g_1} \cos(\theta_1 + \phi_1) + m_2 g\, l_1 \cos(\theta_1) \\ \\ \mathcal{T}_2 = & \left(m_2 l_{g_2}^2 + m_2 l_1 l_{g_2} \cos(\theta_2 + \phi_2)\right) \ddot{\theta}_1 + \left(I_2 + m_2 l_{g_2}^2\right) \ddot{\theta}_2 + \left(m_2 l_1 l_{g_2} \sin(\theta_2 + \phi_2)\right) \dot{\theta_1}^2 \\ & + \beta_2 \dot{\theta}_2 + m_2 g l_{g_2} \cos(\theta_{12} + \phi_2) \end{cases} \tag{4.43}$$

As we can see, it is quite complicated so we can do some assumptions to further simplify it, we assume that the masses are concentrated in the end of the links, this is translated by the following equalities:

$$I_1 = I_2 = 0 \quad l_{g_1} = l_1 \quad l_{g_2} = l_2 \quad \phi_1 = \phi_2 = 0 \tag{4.44}$$

Substituting (4.44) in (4.43) would give us a simpler model:

$$\begin{cases} \mathcal{T}_1 = & \left(m_1 l_1^2 + m_2 (l_1^2 + l_2^2 + 2 l_1 l_2 \cos(\theta_2))\right) \ddot{\theta}_1 + m_2 \left(l_2^2 + l_1 l_2 \cos(\theta_2)\right) \ddot{\theta}_2 - m_2 l_1 l_2 \sin(\theta_2) \left(2\, \dot{\theta}_1 \dot{\theta}_2 + \dot{\theta_2}^2\right) \\ & + m_2 g\, l_2 \cos(\theta_{12}) + (m_1 + m_2) g l_1 \cos(\theta_1) \\ \mathcal{T}_2 = & m_2 \left(l_2^2 + l_1 l_2 \cos(\theta_2)\right) \ddot{\theta}_1 + m_2 l_2^2 \ddot{\theta}_2 + m_2 g\, l_2 \cos(\theta_{12}) + m_2 l_1 l_2 \dot{\theta_1}^2 \sin(\theta_2) \end{cases} \tag{4.45}$$

The general form of inverse dynamics is given by:

$$\mathcal{T} = M(\theta)\ddot{\theta} + C(\theta, \dot{\theta}) + G(\theta) \tag{4.46}$$

The mass matrix $M_{[2\times 2]}(\theta)$

$$M(\theta) = \begin{bmatrix} m_1 l_1^2 + m_2 (l_1^2 + l_2^2 + 2 l_1 l_2 \cos(\theta_2)) & m_2 \left(l_2^2 + l_1 l_2 \cos(\theta_2)\right) \\ m_2 \left(l_2^2 + l_1 l_2 \cos(\theta_2)\right) & m_2 l_2^2 \end{bmatrix} \tag{4.47}$$

$C(\theta, \dot{\theta})$: Velocity product term

$$C(\theta, \dot{\theta}) = \begin{bmatrix} -m_2 l_1 l_2 \sin(\theta_2) \left(2\, \dot{\theta}_1 \dot{\theta}_2 + \dot{\theta_2}^2\right) \\ m_2 l_1 l_2 \dot{\theta_1}^2 \sin(\theta_2) \end{bmatrix} \tag{4.48}$$

$G(\theta)$: Gravity Term

$$G(\theta) = \begin{bmatrix} m_2 g\, l_2 \cos(\theta_1 + \theta_2) + (m_1 + m_2) g l_1 \cos(\theta_1) \\ m_2 g\, l_2 \cos(\theta_1 + \theta_2) \end{bmatrix} \tag{4.49}$$

These reveal that the equations of motion are linear in $\ddot{\theta}$, quadratic in $\dot{\theta}$, and trigonometric in θ. This is true in general for serial chains containing revolute joints, not just for the $2R$ robot. Forward Dynamics:

$$\ddot{\theta} = M^{-1}(\theta) \left(\mathcal{T} - C(\theta, \dot{\theta}) - G(\theta)\right) \tag{4.50}$$

Since the mass matrix is symmetric positive-definite, we can compute its inverse.

$$M^{-1}(\theta) = \begin{bmatrix} \frac{1}{m_1 l_1^2 + m_2 (l1^2 - l_1^2 \cos^2(\theta_2))} & \frac{-l_2 - l_1 \cos(\theta_2)}{l_1^2 l_2 (m_1 + m_2 - m_2 \cos^2(\theta_2))} \\ \frac{-l_2 - l_1 \cos(\theta_2)}{l_1^2 l_2 (m_1 + m_2 - m_2 \cos^2(\theta_2))} & \frac{(m_1 + m_2) l_1^2 + m_2 l_2^2 + 2\, m_2 l_1 l_2 \cos(\theta_2)}{m_2 l_1^2 l_2^2 (m_1 + m_2 - m_2 \cos^2(\theta_2))} \end{bmatrix} \tag{4.51}$$

After getting the inverse off mass matrix we can get the forward dynamics using the equation (4.50)
The dynamics of a multi joint robot gets complicated more and more. it even gets impractical. Fortunately we have other ways of simulating the dynamics of robots, one of them is using Simscape Multibody toolbox in SIMULINK.

4.4 Three Links Robot

We will use the previous results obtained in the previous section to get the dynamics of the three links planar robot.
in addition to the positions (x_1, y_1) and (x_2, y_2) we need the position of (x_3, y_3), that is:

$$\begin{cases} x_3 = l_1 \mathrm{c}(\theta_1) + l_2 \mathrm{c}(\theta_{12}) + l_{g_3} \mathrm{c}(\theta_{123} + \phi_3) \\ y_3 = l_1 \mathrm{s}(\theta_1) + l_2 \mathrm{s}(\theta_{12}) + l_{g_3} \mathrm{s}(\theta_{123} + \phi_3) \end{cases} \tag{4.52}$$

Compute the derivatives of x_3 and y_3 results in:

$$\begin{cases} \dot{x_3} = -l_2 \mathrm{s}(\theta_{12}) \left(\dot{\theta_1} + \dot{\theta_2}\right) - l_{g_3} \mathrm{s}(\theta_{123} + \phi_3) \left(\dot{\theta_1} + \dot{\theta_2} + \dot{\theta_3}\right) - l_1 \mathrm{s}(\theta_1) \dot{\theta_1} \\ \dot{y_3} = l_2 \mathrm{c}(\theta_{12}) \left(\dot{\theta_1} + \dot{\theta_2}\right) + l_{g_3} \mathrm{c}(\theta_{123} + \phi_3) \left(\dot{\theta_1} + \dot{\theta_2} + \dot{\theta_3}\right) + l_1 \mathrm{c}(\theta_1) \dot{\theta_1} \end{cases} \tag{4.53}$$

We first compute the kinetic energy of the third link, i.e.

$$T_3 = \frac{1}{2} I_3 \left(\dot{\theta_1}^2 + \dot{\theta_3}^2 + \dot{\theta_3}^2\right) + \frac{1}{2} m_3 \left(\dot{x_3}^2 + \dot{y_3}^2\right) \tag{4.54}$$

Since we have already computed T_1 and T_2 in the previous section, we can use 4.30 to get the total kinetic energy of the this robot ,

$$T = T_1 + T_2 + T_3 \tag{4.55}$$

In a similar fashion, we should only compute the potential energy of the third link and add it up to the previously calculated ones.

$$U_3 = m_3 g y_3 U = U_1 + U_2 + U_3 \tag{4.56}$$

After computing the Lagrangian and Euler-Lagrange we get the inverse dynamics of the three links planar robot. recalling the matrix form of inverse dynamics.

$$\mathcal{T} = M(\theta)\ddot{\theta} + C(\theta, \dot{\theta}) + G(\theta) \tag{4.57}$$

Since the equations are so long, we will adopt the following notations:

$$m_{12\ldots n} = m_a + m_b + \cdots + m_n \tag{4.58}$$

$$l_{12\ldots n} = l_a \times l_b \times \cdots \times l_n \tag{4.59}$$

The mass matrix $M(\theta)$ is symmetric, we will only write the upper side:

$$M(\theta) = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ * & M_{22} & M_{23} \\ * & * & M_{33} \end{bmatrix} \tag{4.60}$$

such that,

$$\begin{aligned} M_{11} &= m_{123}l_1^2 + m_{23}l_2^2 + m_3l_3^2 + 2m_3l_{13}\cos(\theta_{23}) + 2m_{23}l_{12}\cos(\theta_2) + 2m_3l_{23}cos(\theta_3) \\ M_{12} &= m_{23}l_2^2 + m_3l_3^2 + m_3l_{13}\cos(\theta_{23}) + m_{23}l_{12}\cos(\theta_2) + 2m_3l_{23}\cos(\theta_3) \\ M_{13} &= m_3l_3^2 + m_3l_{13}\cos(\theta_{23}) + m_3l_{23}\cos(\theta_3) \\ M_{22} &= m_{23}l_2^2 + m_3l_3^2 + 2m_3l_{23}\cos(\theta_3) \\ M_{23} &= m_3l_3^2 + m_3l_{23}\cos(\theta_3) \\ M_{33} &= m_3l_3^2 \end{aligned} \tag{4.61}$$

The Velocity product term $C(\theta, \dot{\theta})$:

$$C(\theta, \dot{\theta}) = \begin{bmatrix} C_1 \\ C_2 \\ C_3 \end{bmatrix} \tag{4.62}$$

Such that,

$$\begin{aligned} C_1 = &- m_{23}l_{12}\sin(\theta_2)\dot{\theta_2}^2 - m_3l_{23}\sin(\theta_3)\dot{\theta_3}^2 - 2m_{23}l_{12}\sin(\theta_2)\dot{\theta_1}\dot{\theta_2} - 2m_3l_{23}\sin(\theta_3)\dot{\theta_2}\dot{\theta_3} \\ &- 2m_3l_{23}\sin(\theta_3)\dot{\theta_1}\dot{\theta_3} - 2m_3l_{13}\sin(\theta_{23})(\dot{\theta_2} + \dot{\theta_3})(\dot{\theta_1} + \dot{\theta_2} + \dot{\theta_3}) \end{aligned} \tag{4.63}$$

$$C_2 = \big(m_{23}l_{12}\sin(\theta_2) + m_3l_{13}\sin(\theta_{23})\big)\dot{\theta_1}^2 - m_3l_{23}\sin(\theta_3)\dot{\theta_3}^2 - 2m_3l_{23}\sin(\theta_3)\dot{\theta_3}(\dot{\theta_1} + \dot{\theta_2}) \tag{4.64}$$

$$C_3 = \big(m_3l_{23}\sin(\theta_3) + m_3l_{13}\sin(\theta_{23})\big)\dot{\theta_1}^2 + m_3l_{23}\sin(\theta_3)\dot{\theta_2}^2 + 2m_3l_{23}\sin(\theta_3)\dot{\theta_1}\dot{\theta_2} \tag{4.65}$$

$$\tag{4.66}$$

The Gravity Term $G(\theta)$:

$$G(\theta) = \begin{bmatrix} m_{123}g\,l_1\cos(\theta_1) + m_{23}g\,l_2\cos(\theta_{12}) + m_3g\,l_3\cos(\theta_{123}) \\ m_{23}g\,l_2\cos(\theta_{12}) + m_3g\,l_3\cos(\theta_{123}) \\ m_3g\,l_3\cos(\theta_{123}) \end{bmatrix} \tag{4.67}$$

Chapter 5

Robot Control

5.1 Introduction

Most robots are driven by actuators that apply a force or torque to each joint. Hence, precisely controlling a robot requires an understanding of the relationship between the joint forces and torques and the motion of the robot; this is the domain of dynamics. Even for simple robots, however, the dynamic equations are complex and dependent on a precise knowledge of the mass and inertia of each link, which may not be readily available. Even if it were, the dynamic equations would still not reflect physical phenomena such as friction, elasticity, backlash, and hysteresis. Most practical control schemes compensate for these uncertainties by using feedback control. After examining the performance limits of feedback control without a dynamic model of the robot, we study motion control algorithms, such as computed torque control.

5.1.1 Control Objectives

- - Motion Control, the robot should move along a specific trajectory. examples of this are: painting, spot welding, laser cutting, 3D Printing, etc. In these tasks, the trajectory is so important.
- Force Control, the robot should apply a desired force to an object or environment. as picking up an egg, if the force is the grab force is too powerful, it could break the egg, in the other hand, if the grab force is too weak it might drop it off.
- Hybrid Motion-Force Control, as writing in a board, we need to control the trajectory of end effector in the plan of the board, and the force into the board, it should be neither too powerful to not break the board, nor too weak that it looses contact with the board and then we won't have the trace of the pen.
- Impedance Control, when the robot is used to render a virtual environment, the user grabs the end effector and moves it around to explore the objects in a virtual world, this control objective can be used to develop virtual reality games, movies or it can be used to simulate hardware that is unavailable or expensive and requires special care.

5.2 Independent Joint Control

Independent Joint Control is the easiest controller since it assumes that there is no coupling between the links, as if they were totally independent, and then it uses some kind of usual feedback controllers (PID for instance) in each link to command the Joint positions. The block diagram of a PID controller is shown in figure 5.1:

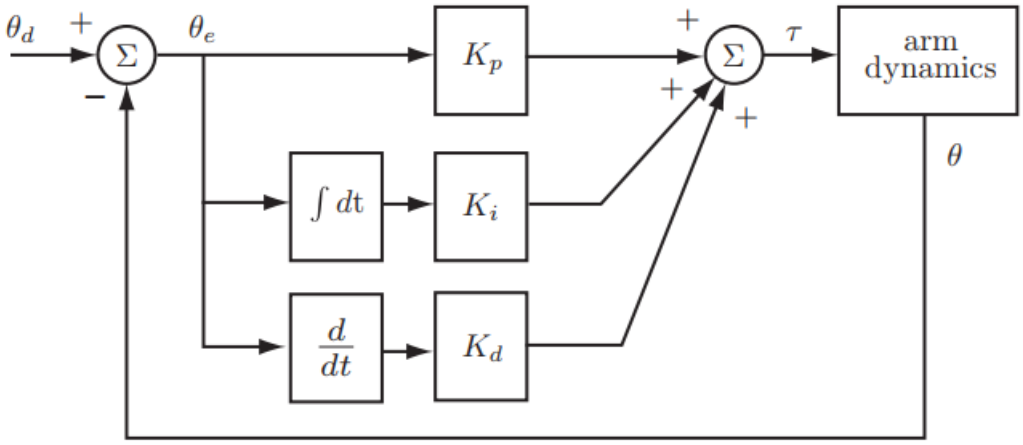

Figure 5.1: Block Diagram of PID Controller

Let's consider the case of the simplest robot, the single joint robot. The results can easily be generalized to **N DOF** robots.

5.2.1 PD Controller with $g = 0$

First, assume that the robot is an horizontal plan, this can easily be translated by putting $g = 0$. Hence the dynamic model of such a robot becomes:

$$\mathcal{T} = M\,\ddot{\theta} + \beta\,\dot{\theta} \tag{5.1}$$

Where $M = ml_g^2 + I_{zz}$.
We will be interested in **Set Point Control** which means that the desired joint position is constant, as a result, we get:

$$\begin{aligned} \dot{\theta}_d &= 0 \\ \ddot{\theta}_d &= 0 \end{aligned} \tag{5.2}$$

We define the error variables to be:

$$\left\langle \begin{aligned} \theta_e &= \theta_d - \theta \\ \dot{\theta}_e &= \dot{\theta}_d - \dot{\theta} \\ \ddot{\theta}_e &= \ddot{\theta}_d - \ddot{\theta} \end{aligned} \right\rangle \tag{5.3}$$

We design a **PD** controller to this arm

$$\mathcal{T} = K_p\theta_e + K_d\dot{\theta}_e \tag{5.4}$$

If we equate 5.1 and 5.4

$$K_p\theta_e + K_d\dot{\theta}_e = M\ddot{\theta} + \beta\dot{\theta} \tag{5.5}$$

Since $\dot{\theta}_d = 0$ and $\ddot{\theta}_d = 0$:

$$\begin{aligned} \dot{\theta}_e &= -\dot{\theta} \\ \ddot{\theta}_e &= -\ddot{\theta} \end{aligned} \tag{5.6}$$

We use this result in 5.5, we will then have:

$$K_p\theta_e - K_d\dot{\theta} = M\ddot{\theta} + \beta\dot{\theta} - M\ddot{\theta} - (K_d + \beta)\dot{\theta} + K_p\theta_e = 0 \tag{5.7}$$

Using 5.6, we get the following second order differential equation:

$$\ddot{\theta}_e + \frac{(K_d + \beta)}{M}\dot{\theta}_e + \frac{K_p}{M}\theta_e = 0 \tag{5.8}$$

The canonical form of a second order differential equation is:

$$\ddot{y} + 2\,\xi\,\omega_n\dot{y} + \omega_n^2 y = 0 \tag{5.9}$$

Where ξ is the damping ration and ω_n is the natural frequency.

$$\xi = \frac{\beta + K_d}{2\sqrt{K_p M}} \quad \omega_n = \frac{K_p}{M} \tag{5.10}$$

In order for the system to be stable K_p and K_d should satisfy the following conditions:

$$K_p > 0 \quad K_d > -\beta \tag{5.11}$$

Usually we set the gains such that we get an critically damped response, this is the case when $\xi = 1$, hence

$$K_d = 2\sqrt{k_p M} - \beta \tag{5.12}$$

Although it seems that we have the freedom to choose K_p as large as we want, We should not forget about the practical constraints, such as actuators maximum velocities, joints limits, unmodeled dynamics, etc.

Simulation

These are the parameters of the link used in simulation:

$$m = 0.25\,\text{kg} \quad l_g = 0.125\,\text{m} \quad I = 0.00135\,\text{kg.m}^2 \quad \phi = 0^\circ \quad g = 0\,\text{m/s}^2 \quad \beta = 0.01\,\text{N.s/m} \tag{5.13}$$

SIMULINK Model:

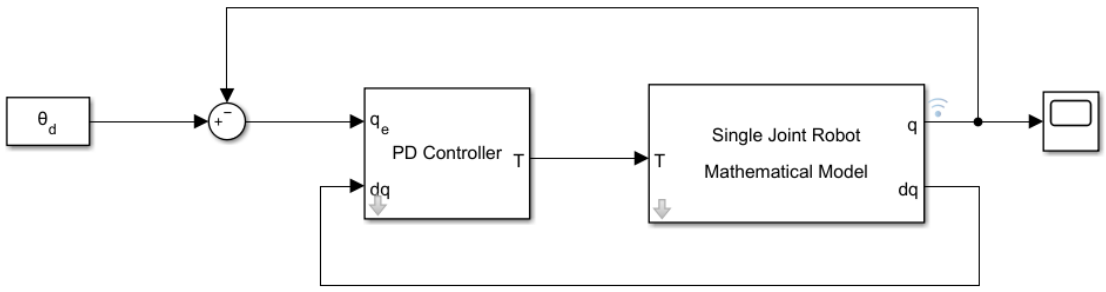

Figure 5.2: SIMULINK Model

The PD Controller is shown in figure 5.3:

We set the input to be a step function of 60° amplitude and we simulate the model with 2 different controller gains, the results are shown in figure:

As we can see, both controllers give a zero steady state error, but we prefer the one with no overshoot since overshoot can cause damage to the robot, or may cause harm to the users.

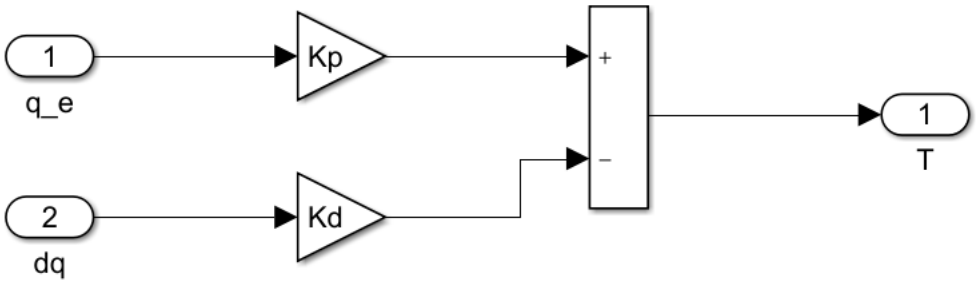

Figure 5.3: PD Controller Implementation in SIMULINK

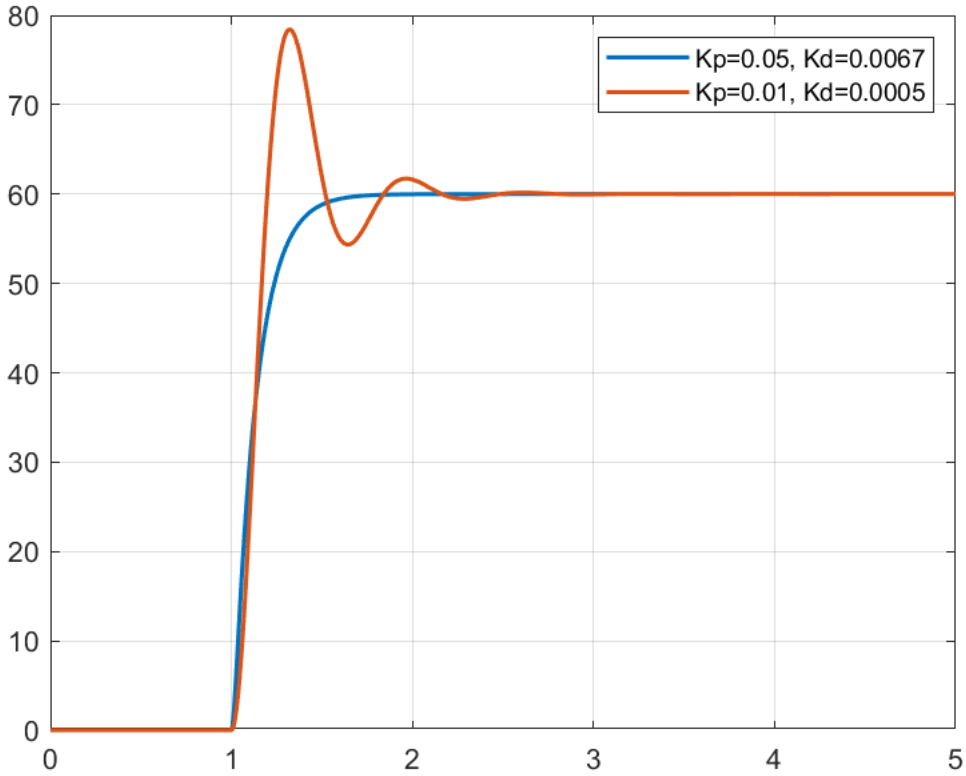

Figure 5.4: step response

5.2.2 PD Controller with $g \neq 0$

in the first section we have discussed the case of a single joint robot in a horizontal plan, we now are interested in studying the effect of gravity on the arm, so we set g to be 9.81m.s^2, the dynamic model of such an arm is:

$$\mathcal{T} = M\,\ddot{\theta} + \beta\,\dot{\theta} + mg\,l_g\cos(\theta + \phi) \tag{5.14}$$

Substituting the PD control law into the dynamics, we get

$$K_p\theta_e - K_d\dot{\theta} = M\ddot{\theta} + \beta\dot{\theta} + mgl_g cos(\theta - \phi) - M\ddot{\theta} - (K_d + \beta)\dot{\theta} + K_p\theta_e = mgl_g cos(\theta - \phi) \tag{5.15}$$

This can be written as:

$$\ddot{\theta_e} + \frac{K_d + \beta}{M}\dot{\theta_e} + \frac{K_p}{M}\theta_e = \frac{mgl_g}{M}cos(\theta - \phi) \tag{5.16}$$

In steady state, i.e when $\ddot{\theta_e} = \dot{\theta_e} = 0$ we get:

$$\theta_e = \frac{mgl_g}{K_p} cos(\theta - \phi) \tag{5.17}$$

We notice that the steady state error is not null anymore when $\theta_d \neq \pm 90 + \phi$. We can decrease this error by increasing the gain K_p but the error will never be zero. Hence, Proportional Derivative controller should not be used if we need good precision (which is almost always the case).

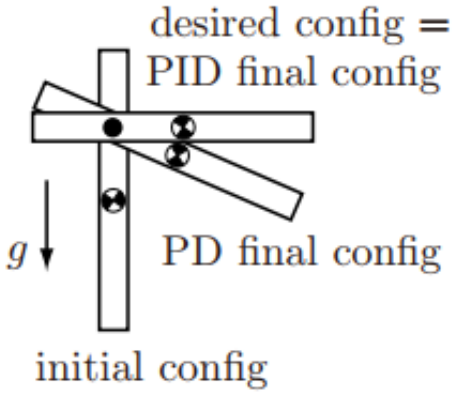

Figure 5.5: Position Error Due to the Gravity

Simulation In the same Simulink model we change the value of g and set it to be equal to $9.81\,\mathrm{m/s^2}$ and let everything else unchanged.
We get the following step response:

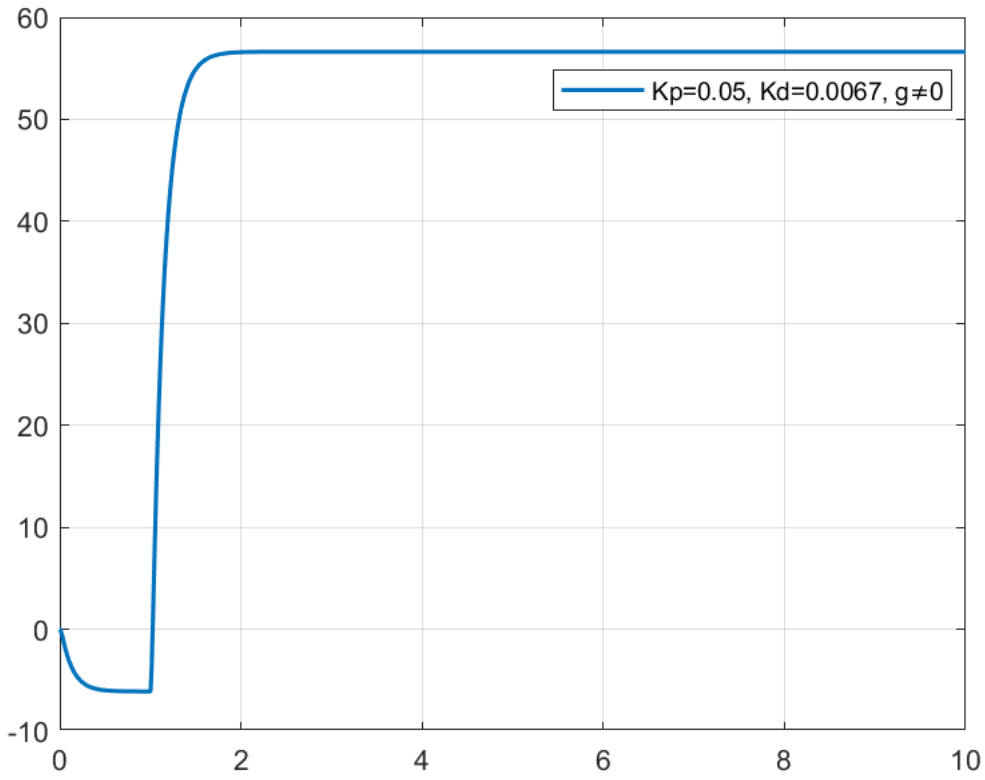

Figure 5.6: Step Response PD Controller with $g \neq 0$

It's clear from figure 5.6 that the steady state error is not null.

5.2.3 PID Controller

In order to compensate this steady state error we consider adding the Integral action to our controller. we end up having a PID controller,

$$\mathcal{T} = K_p\theta_e + K_i \int \theta_e dt + K_d\dot{\theta_e} \tag{5.18}$$

Equating the joint dynamics and the control torque would give us:

$$K_p\theta_e + K_i \int \theta_e dt + K_d\dot{\theta_e} = M\,\ddot{\theta} + \beta\,\dot{\theta} - mg\,l_g\cos(\theta+\phi) \tag{5.19}$$

Since $\cos(\theta+\phi)$ is constant in steady state, we can safely replace it with a constant, we set:

$$\mathcal{T}_{dist} = mg\,l_g\cos(\theta+\phi) \tag{5.20}$$

We then get:

$$M\ddot{\theta_e} + (\beta + K_d)\dot{\theta_e} + K_p\theta_e + K_i \int \theta_e dt = \mathcal{T}_{dist} \tag{5.21}$$

Differentiating the two sides would results in a third order differential equation:

$$M{\theta_e}^{(3)} + (\beta + K_d){\theta_e}^{(2)} + K_p{\theta_e}^{(1)} + K_i\theta_e = 0 \tag{5.22}$$

The characteristic equation of this differential equation is given by:

$$\Delta(s) = s^3 + \frac{\beta + K_d}{M}s^2 + \frac{Kp}{M}s + \frac{K_i}{M} = 0 \tag{5.23}$$

Using Routh-Hurwitz Stability Criterion to get the constraints on K_p, K_d and K_i that assure (at least) the stability of the system:

$$\begin{array}{c|cc} s^3 & 1 & \frac{K_p}{M} \\ s^2 & \frac{\beta + K_d}{M} & \frac{K_i}{M} \\ s^1 & \frac{K_p\beta + K_dK_p - MK_i}{M(\beta + K_d)} & 0 \\ s^0 & \frac{K_i}{M} & 0 \end{array} \tag{5.24}$$

In order for this equation to be stable i.e. have all the roots in the right half plan, all the elements of the first column in the table must have the same sign (positive in this case) and none of them must be null, we get the the following constraints:

$$0 < Kp \qquad -\beta < K_d \qquad 0 < Ki < \tfrac{K_p(\beta+K_d)}{M} \tag{5.25}$$

As opposed to K_p and K_d, the integral gain K_i have both upper and lower bands. and it's written as a function of K_p and K_d. When tuning the PID gains, we should first set K_p and K_d then compute K_i that assures stability
Now that we have the characteristic equation of the system, we can use pole placement to get the appropriate PID gains.

$$\begin{gathered} \Delta(s) = (s - s_1)(s - s_2)(s - s_3) \\ \Delta(s) = s^3 - (s_1 + s_2 + s_3)s^2 + (s_1s_2 + s_2s_3 + s_3s_1)s - s_1s_2s_3 \end{gathered} \tag{5.26}$$

Identifying 5.23 and 5.26 would give us the following equalities:

$$\begin{cases} K_p = M(s_1 s_2 + s_2 s_3 + s_3 s_1) \\ K_i = -M s_1 s_2 s_3 \\ K_d = -M(s_1 + s_2 + s_3) - \beta \end{cases} \tag{5.27}$$

- We set $K_i = 0$ and pick K_p and K_d to give a critical dumping response. Based on (5.12)
- Setting $K_i = \varepsilon$ a small positive gain creates a 3rd pole s_3 close to the origin.
- When we increase the gain K_i the first two poles move away from each other, and the third pole approaches the first one.
- Increasing K_i sufficiently, the first and third poles will be collocated.
- Increasing K_i even more, would introduce imaginary parts to the first and third poles s.t. $s_1 = \overline{s_3}$, making the response oscillatory.
- When $\frac{K_p(\beta+K_d)}{M} < K_i$ The first and third poles will be located in the right half plan, which makes the system instable.

- We set $K_i = 0$ and pick K_p and K_d to give a critical dumping response. Based on (5.12)
- Setting $K_i = \varepsilon$ a small positive gain creates a 3rd pole s_3 close to the origin. - When we increase the gain K_i the first two poles move away from each other, and the third pole approaches the first one. - Increasing K_i sufficiently, the first and third poles will be collocated. - Increasing K_i even more, would introduce imaginary parts to the first and third poles s.t. $s_1 = \overline{s_3}$, making the response oscillatory. - When $\frac{K_p(\beta+K_d)}{M} < K_i$ The first and third poles will be located in the right half plan, which makes the system instable.
This Analysis shows that K_i improves steady states error, but worsen transient response and may even cause instabilities. That's why we tend to choose K_i as small as possible. otherwise we would use another control scheme that is *Computed Torque* which is introduced in the next section.

Simulation

We Keep the same parameters that are used in the previous simulation, and we modify the controller by adding an integrator, the newly created block is shown in this figure:

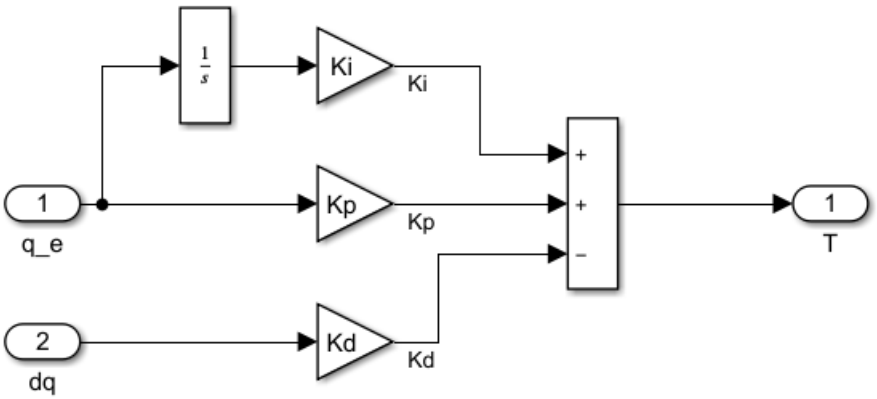

Figure 5.7: PID Controller Implementation in SIMULINK

We run the simulation with various PID gains, and examine their response in this plot:
It's clear that the steady state error is null for both controllers and the response is underdamped.

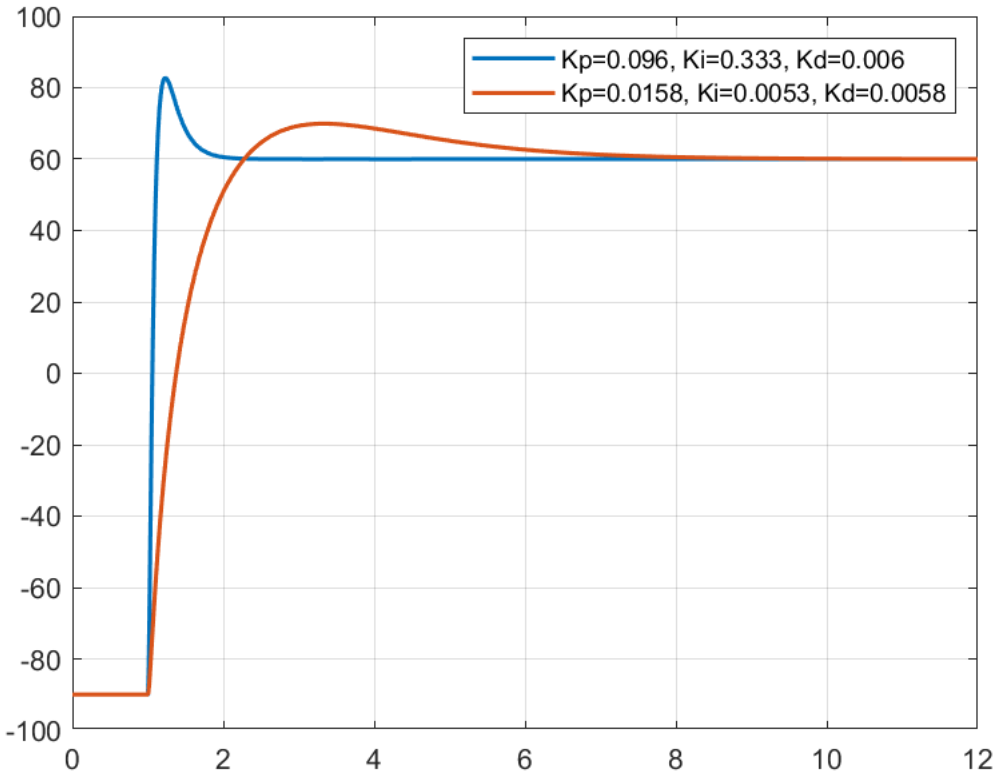

Figure 5.8: PID Controller Implementation in SIMULINK

We see that the torques due to the proportional and derivative terms both go to zero, while the integral term reaches a non zero steady state, that is the torque needed to resist the effect of gravity, even when the error is null.

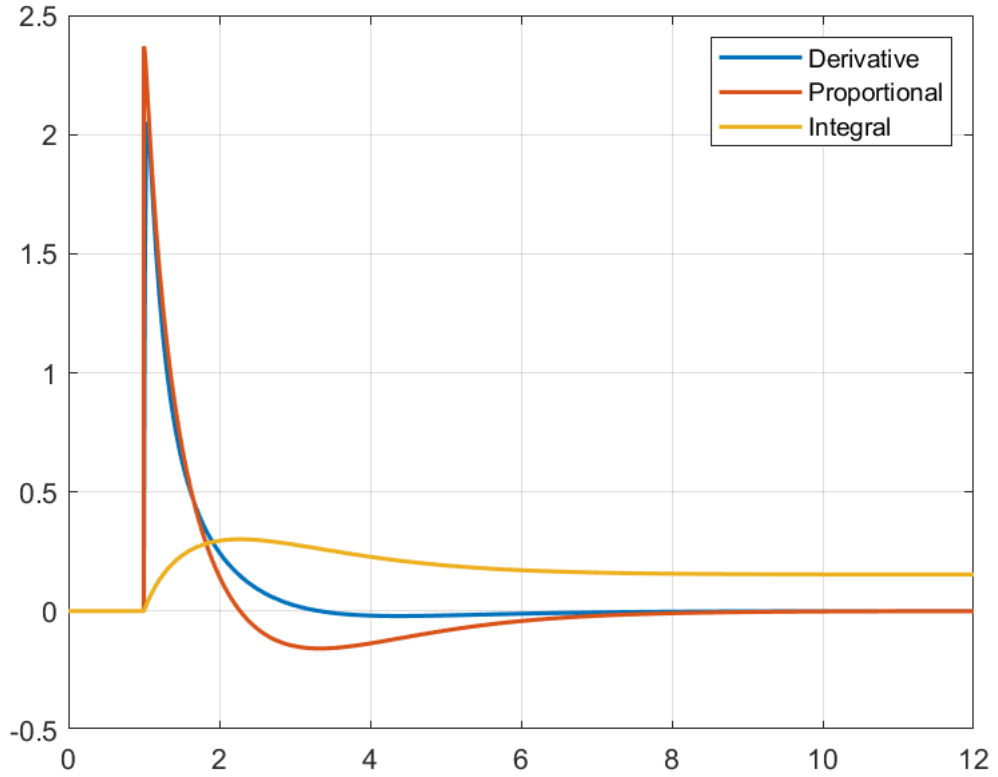

Figure 5.9: PID Controller Implementation in SIMULINK

5.3 Computed Torque Control - CTC

We have seen in Chapter 3 that the robot dynamics equations are coupled, time varying and highly non linear systems. In the previous section we were applying linear controllers, They gave good results to the set point control problem. This time we will be interested in studying a non linear controller that is Computed Torque Control.
Computed Torque is a well known control scheme it uses the robot dynamics model in order to control the robot. This controller is given by the equation:

$$\mathcal{T} = \tilde{M}(\theta)\big(\ddot{\theta}_d + K_v\dot{\theta}_e + K_p\theta_e\big) + \tilde{C}(\theta,\dot{\theta}) + \tilde{G}(\theta) \tag{5.28}$$

Where K_v and K_p are symmetric positive definite design matrices. $\tilde{M}$, $\tilde{C}$, $\tilde{G}$ are the *estimated* mass matrix, the velocity product term and the gravity term respectively. if the model is perfect then $\tilde{M} = M$, $\tilde{C} = C$, $\tilde{G} = G$.
Although 5.28 is so similar to a PD controller, the CTC is a non linear controller since the position and velocity gains are not constant (i.e. time variant) and they depend explicitly on the error θ_e, re-writing the equation 5.28 using the fact that $\theta = \theta_d - \theta$ would ease this observation.

$$\mathcal{T} = \tilde{M}(\theta_d - \theta_e)K_v\dot{\theta}_e + \tilde{M}(\theta_d - \theta_e)K_p\theta_e + \tilde{M}(\theta_d - \theta_e)\ddot{\theta}_d + \tilde{C}(\theta,\dot{\theta}) + \tilde{G}(\theta) \tag{5.29}$$

The block diagram of a typical Computed Torque Controller is given in figure

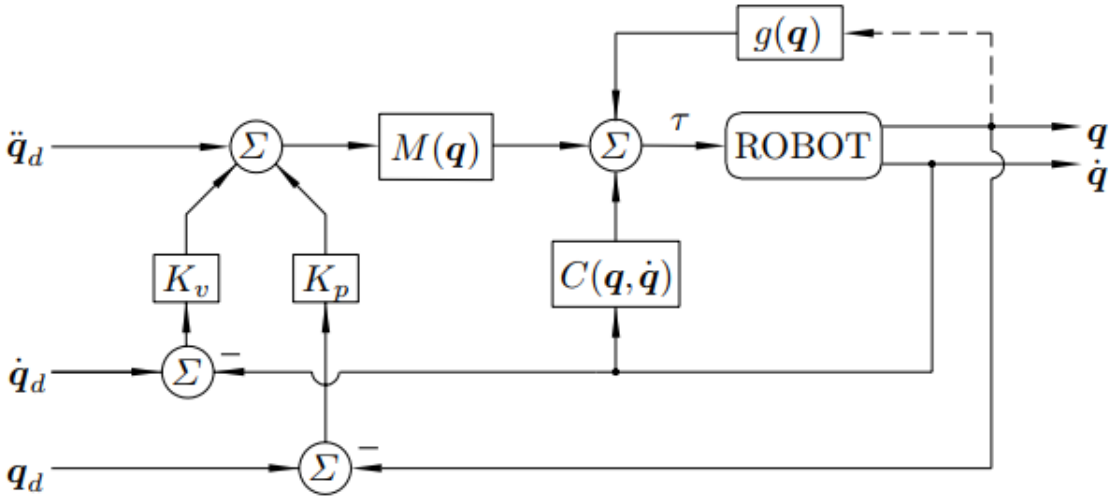

Figure 5.10: Block Diagram of Computed Torque Controller

If we equate the controller torque, and the torque of the dynamic model and by supposing $\tilde{M} = M$, $\tilde{C} = C$, $\tilde{G} = G$. we get

$$M(\theta)\big(\ddot{\theta}_d + K_v\dot{\theta}_e + K_p\theta_e\big) = M(\theta)\ddot{\theta} \tag{5.30}$$

We know that the mass matrix is positive definite, thus we can compute its inverse.
By Pre-multiplying the two sides of 5.30 we get,

$$\ddot{\theta}_e + K_v\dot{\theta}_e + K_p\theta_e = 0 \tag{5.31}$$

The equation 5.31 is N Second-Order Linear Ordinary Differential Equations. We chose the elements of K_v and K_p to give critical responses. that is $\xi = 1$.

$$K_v = 2\sqrt{K_p} \tag{5.32}$$

Simulation
We have developed a computed torque controller for the single joint robot, it is shown in figure 5.11. We have set the gains $Kp = 10$ and $K_v = 2\sqrt{10}$.

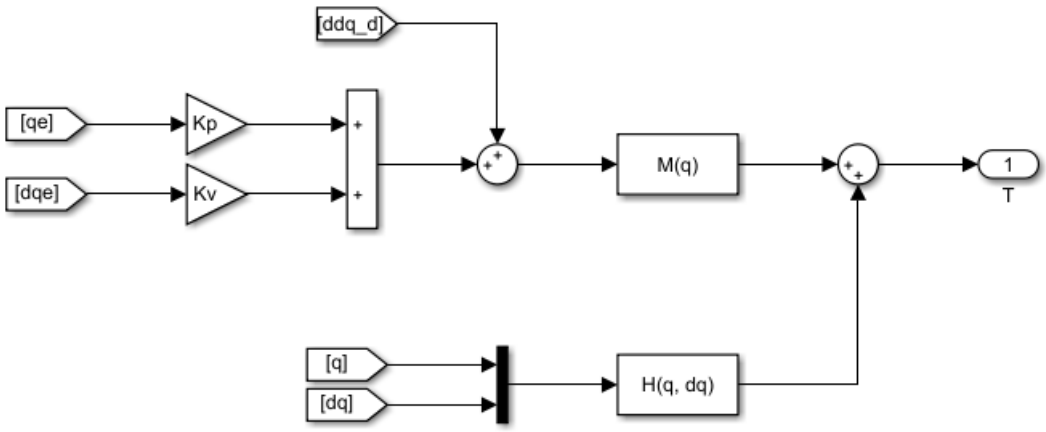

Figure 5.11: Computed Torque Controller Implementation in SIMULINK

The overall SIMULINK model is depicted in figure 5.12

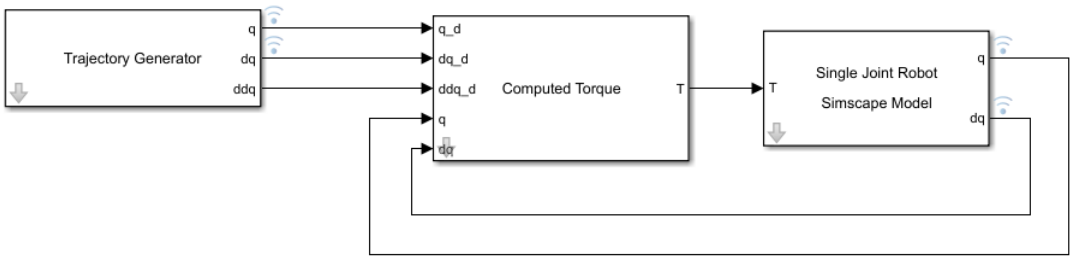

Figure 5.12: Control of a Single Joint Robot using CTC in SIMULINK

The signals provided by the trajectory generator block are shown in figure 5.13
We can see from 5.14 that this controller has a good tracking capability.
We now are interested in seeing the effect of having rough estimates of the dynamic model on the trajectory tracking. We set the controllers parameters to be:

$$m = 1\,kg \quad l_g = 0.1m \quad I = 0.009\,kg.m^2 \quad \beta = 0.1N.m/s \tag{5.33}$$

After simulation, we can see from 5.15 that the reference tracking is not good anymore, this shows how important is having good estimates of the robot. In short the computed torque control is a powerful controller, although we should use it only f we dispose of a good dynamics model. If not, other much simpler controllers can have a superior performance than the CTC.

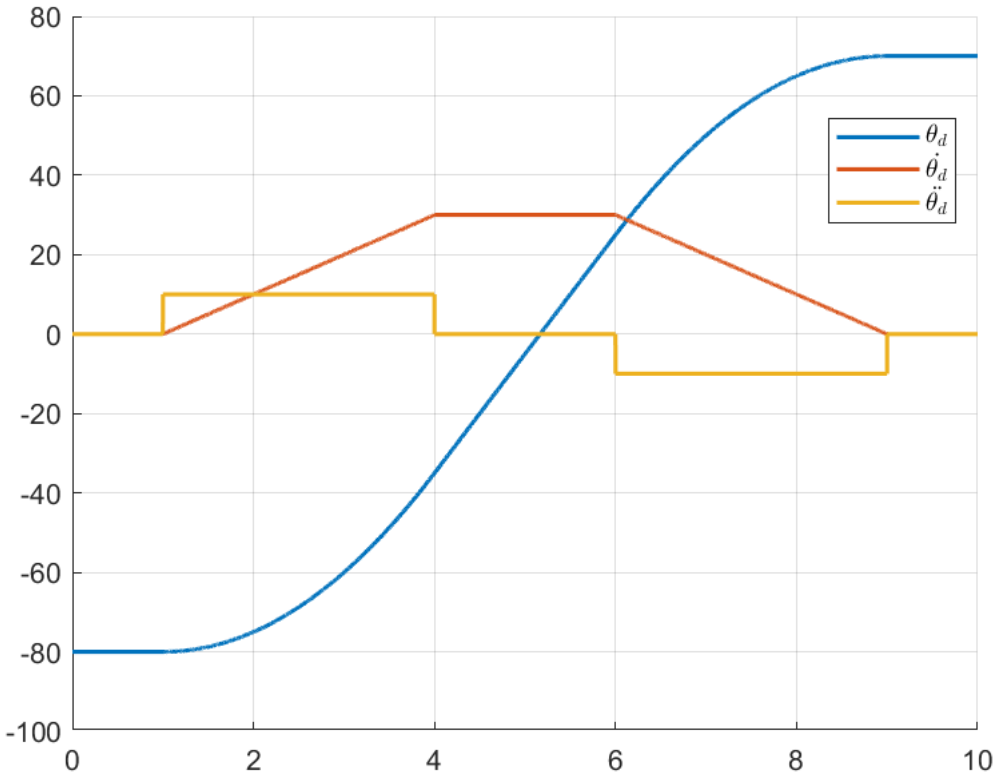

Figure 5.13: Trajectory Generator Output Signals

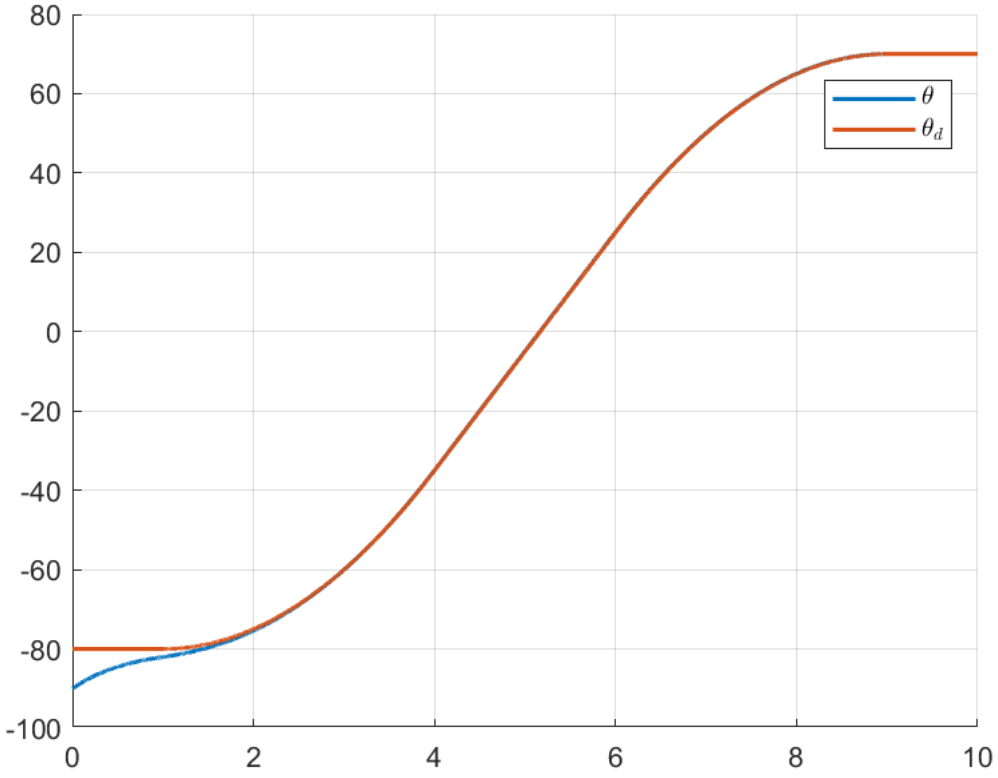

Figure 5.14: The Response of CTC to a Varying Trajectory

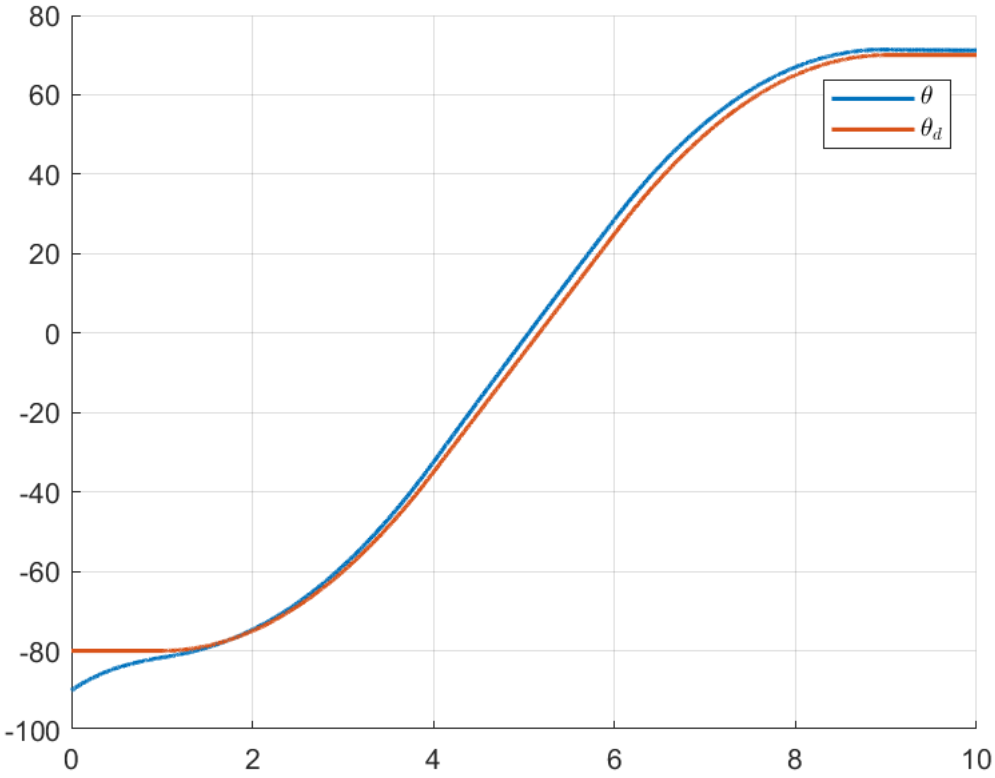

Figure 5.15: The Response of CTC to a Varying Trajectory

Chapter 6

Simulation and Results

In order to simulate the dynamics of a PRRR SCARA robot we have created a SIMULINK model shown in figure 6.1. This was done using Simscape Multibody toolbox. The model contains one prismatic joint and 3 revolute joints, the links were modeled using a CAD software called Autodesk Fusion 360, the CAD files are depicted in figure 6.5

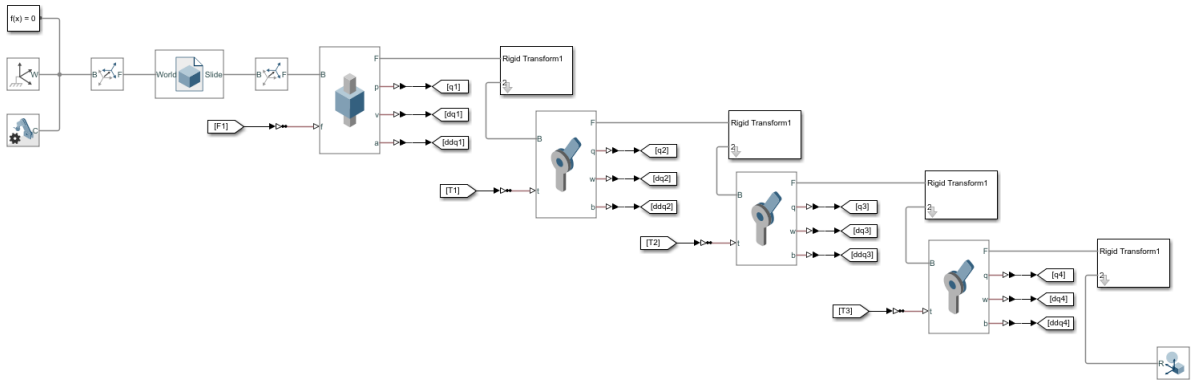

Figure 6.1: The Simscape model of a PRRR SCARA Robot

The Complete Simulink model is shown in figure 6.2

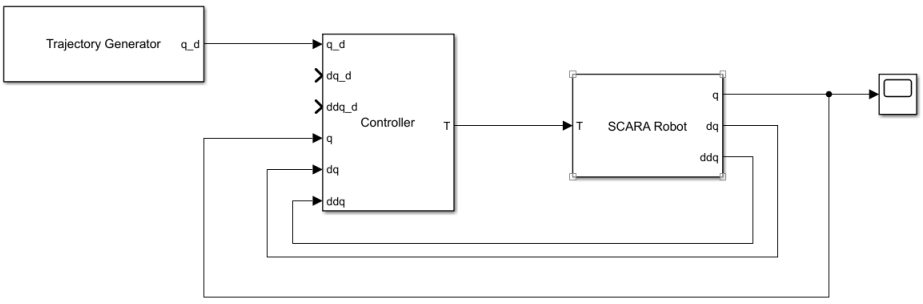

Figure 6.2: The SIMULINK Model

The Trajectory generator shown in figure 6.3 is the block that generates joints trajectories out of Cartesian trajectories, it does this using the inverse kinematics block.

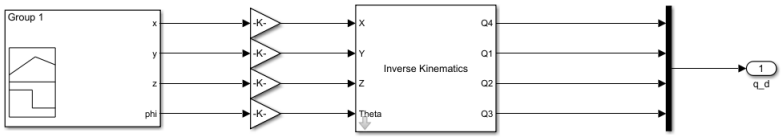

Figure 6.3: The Trajectory Generator Block

The controller block is nothing but 4 simple PD controllers, as shown in figure 6.4. Tuning the PD gains is a tedious task since the model is highly non linear and the dynamics are coupled, we managed to find some gains that work by trial and error. the PID gains we have chosen are:

$$\begin{aligned} K_{p_1} &= 100 & K_{d_1} &= 50 \\ K_{p_2} &= 50 & K_{d_2} &= 0.1 \\ K_{p_3} &= 0.5 & K_{d_3} &= 0.2 \\ K_{p_4} &= 0.05 & K_{d_4} &= 0.01 \end{aligned} \tag{6.1}$$

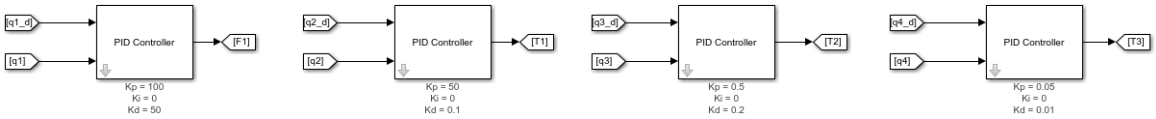

Figure 6.4: PID Controllers

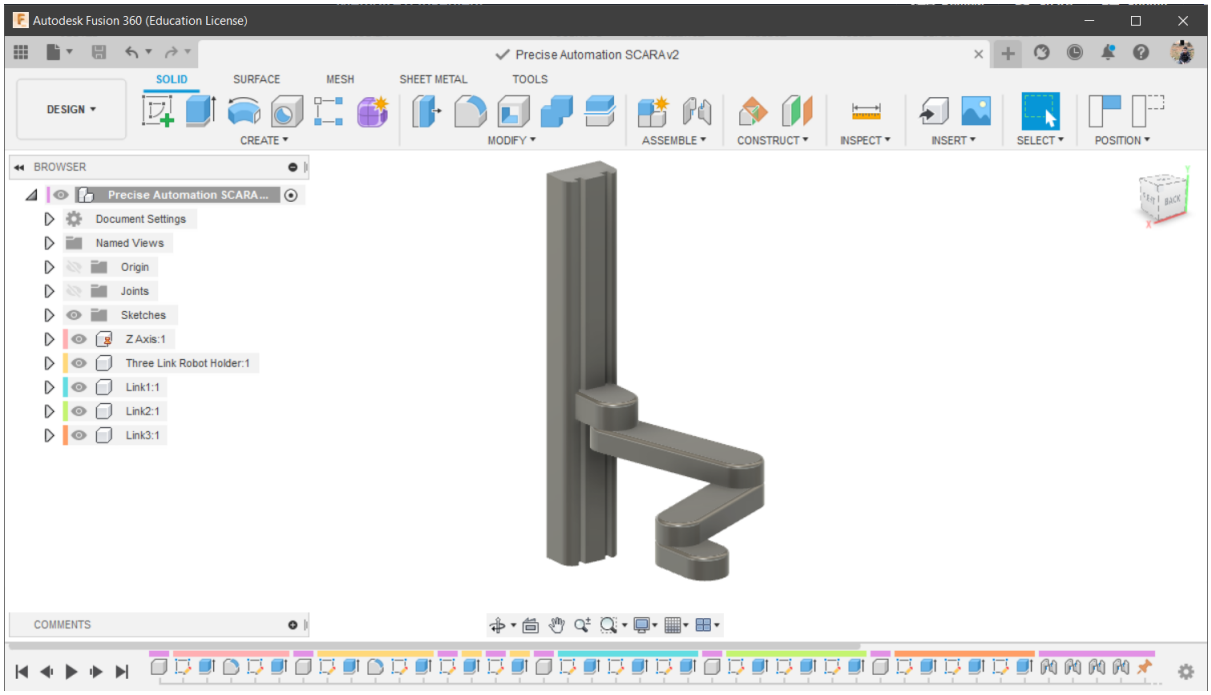

Figure 6.5: The CAD file of SCARA Robot

After Simulation, the mechanics explorer gets opened allowing us to visualize the movement in 3D space, a frame of this is shown in figure 6.6

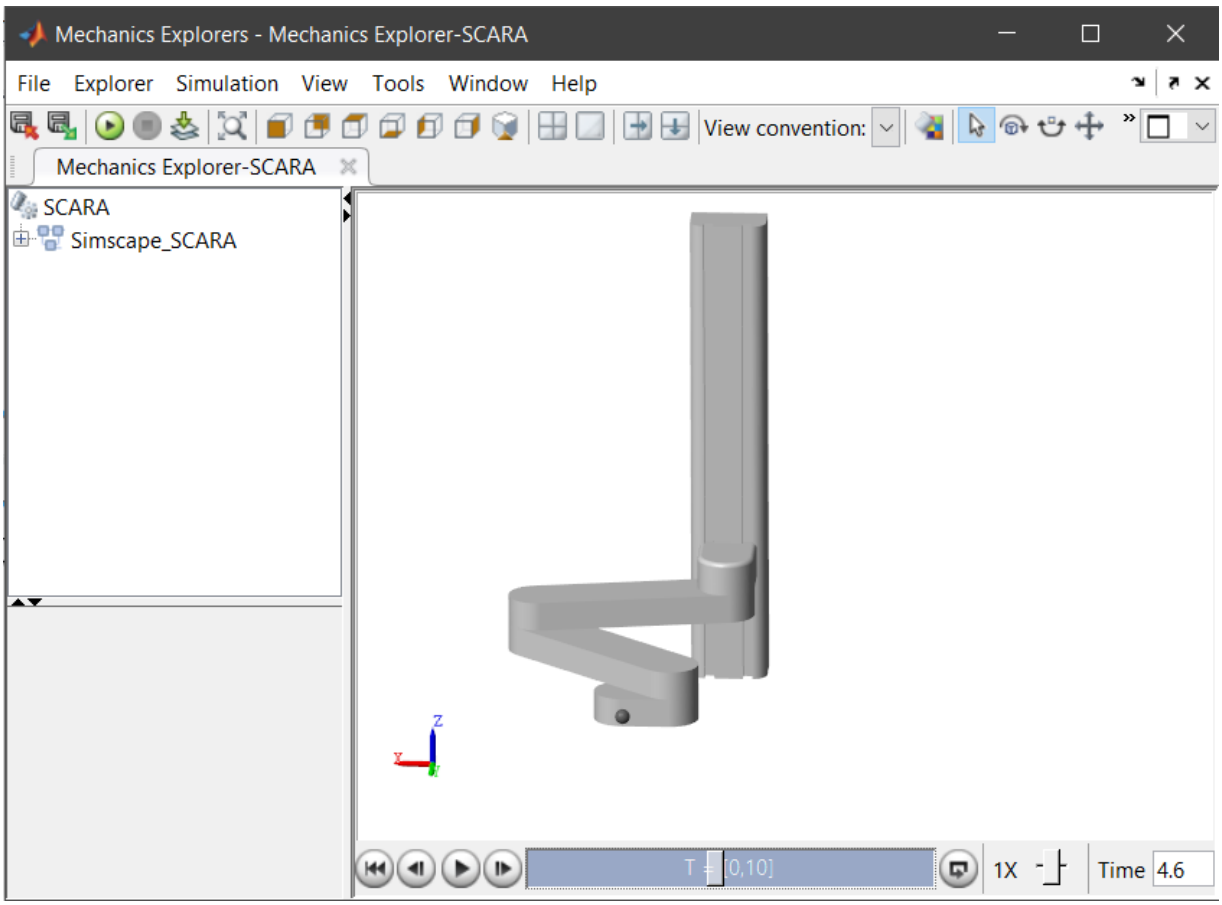

Figure 6.6: Mechanics Explorer Showing the SCARA Robot

The results are shown in figure 6.7 and 6.8
We notice that the PD controller is suitable to control the SCARA robot since it consist of a set of links in horizontal plan. Although the movement is quite jerky.

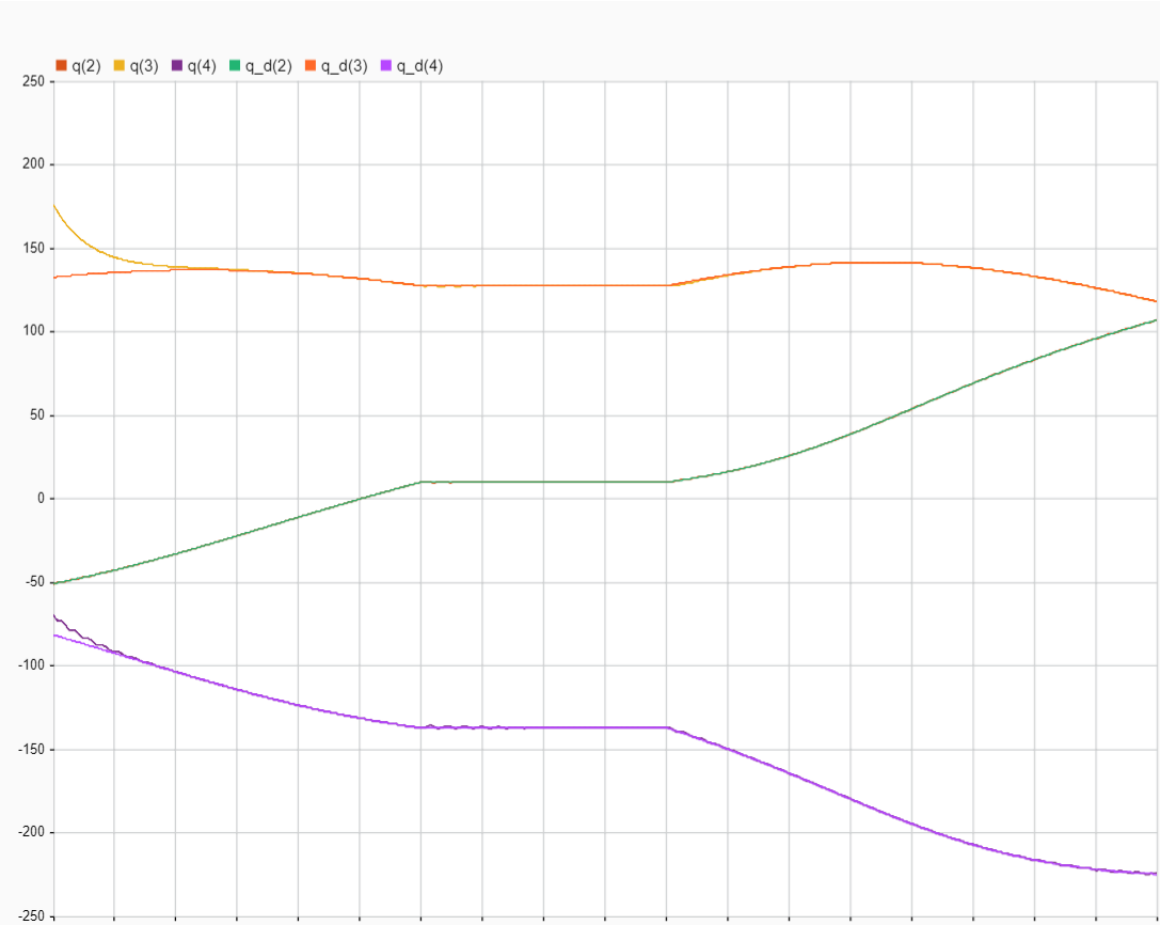

Figure 6.7: Position Tracking of Revolute Joints

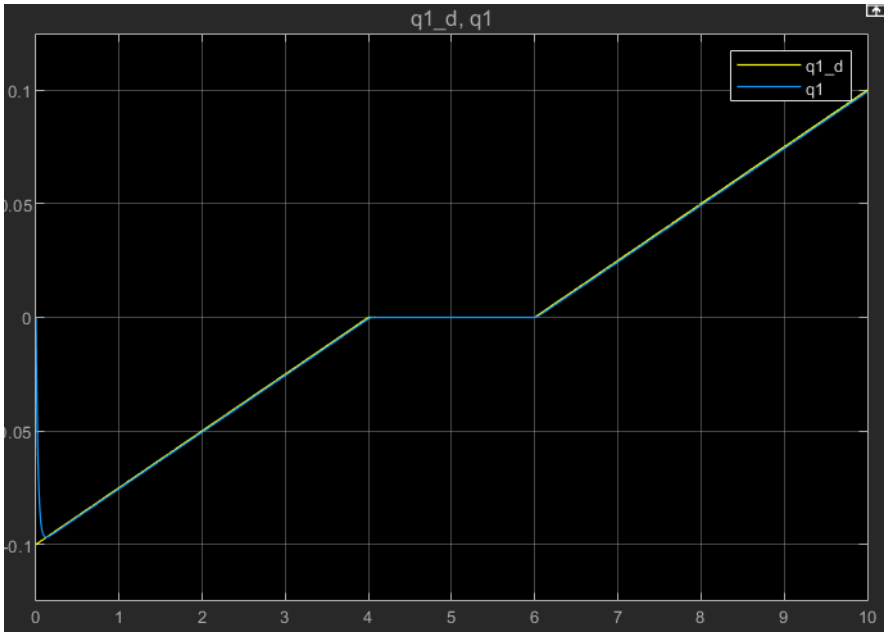

Figure 6.8: Position Tracking of Prismatic Joints

Conclusion

In this thesis we have modeled a SCARA robot using Simscape Multibody, that is a toolbox in SIMULINK, because we have seen that the inverse dynamics models needed to simulate a robot gets complicated when adding more degrees of freedom, We have used PD controllers to control the arm and it turned out to be a good solution although the PID tuning process is not as easy as it was when we had only one degree of freedom. Some future possible work would be to try another controller scheme, like a fuzzy logic controller. Modeling the arm using the more suitable formalism, that is Newton-Euler formalism since it is recursive and computationally optimized.

Bibliography

[1] Kevin M. Lynch and Frank C. Park. *Modern Robotics: Mechanics, Planning, and Control.* Cambridge University Press, 2017.

[2] Sciavicco, Lorenzo, Siciliano, Bruno. *Modelling and Control of Robot Manipulators*

[3] Etienne Dombre and Wisama Khalil. *Modeling, Performance Analysis and Control of Robot Manipulators*

[4] John J. Craig *Introduction to Robotics Mechanics and Control*

[5] Richard M. Murray, Zexiang Li, S. Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*

[6] R. Kelly, V. Santibáñez and A. Loría *Control of Robot Manipulators in Joint Space*

[7] Peter Corke *Robotics, Vision and Control Fundamental Algorithms in MATLAB*