The Islamic University of Gaza
Deanship of Post Graduated Studies
Faculty of Engineering
Electrical Engineering Department

الجامعة الإسـلامية بغزة
عمادة الدراســات العليا
كُليـة الهندســـــــة
قسم الهندسة الكهربائية

Modeling and High Precision Motion Control of 3 DOF Parallel Delta Robot Manipulator

By

Eng. Hamdallah A. H. Alashqar

354/2007

Advisor

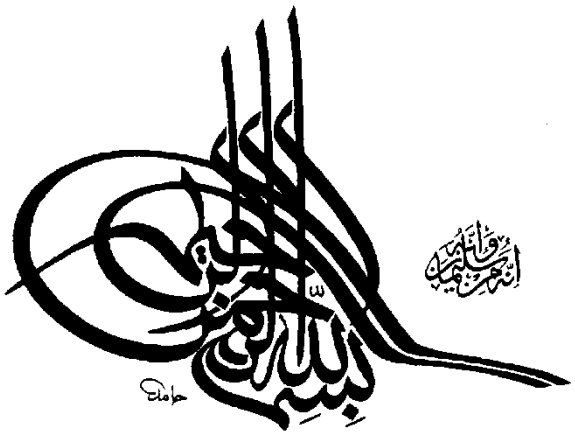
Prof. Dr. Mohammed T. Hussein

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master in Electrical Engineering

Gaza-Palestine

1434-2013

Thesis approval



(وَقُل رَّبِّ زِدْنِي عِلْمًا)

[طه: 114]

Dedication

To the teacher of the world, leader of the nation and mercy of Allah to mankind, Prophet Muhammad peace be upon him

To my lovely parents who are honor by this moment

To my Wife and my sweet daughter Lian

Acknowledgements

First and forever, all praise and thanks for Allah, who gave me the strength, and patience to carry out this work in this good manner. I would like to deeply thank my supervisor Prof. Dr. Mohammed Hussein for his assistance, guidance, support, patience, and encouragement. I would like to deeply thank my discussion committee members Dr. Hatem Elaydi and Dr. Iyad Abu-hadrous for their assistance and encouragement. I would like to thank the Islamic University of Gaza providing me a good academic environment. In addition, I would like to acknowledge the Academic staff of Electrical Engineering, who supports me to carry out this work. Finally, great thanks go to my beloved family for their endless praying and continuous support.

ABSTRACT

This Master thesis describes the CAD- Modeling of the Parallel DELTA robot, designed by Autodesk Inventor® software program. DELTA Robot is a Multi-Input Multi Output Nonlinear System (MIMO), so, PID controller and Model Predictive Controller (MPC) are implemented to improve the performance of Robot .but due to the variations in the dynamic models of each system, it is nearly impossible to conclusively determine the most appropriate controller to design. Therefore, this thesis compares the simulation results of two controllers, namely the PID and MPC respectively; on a 3 DOF Parallel DELTA robot in order to determine which controller would yield the best control performance.

By comparing the simulation results for the joint angles error and the end effector trajectory error plots for the PID and MPC controllers, MPC controller gave the best results than PID controller. Then, a great contribution added at the response of DELTA robot. Because of Robot arms are highly geared; this reason let the robot to be more robust. MPC controller held the Potential to be the most likely candidate controllers to implement on the physical structure of the 3-DOF Parallel DELTA robot. But PID controller is easier in software implementation inside embedded systems as microcontrollers.

ملخص:

التمثيل الرياضي و التحكم عالي الدقة بالروبوت المتوازي "دلتا" ثلاثي الأبعاد

اقترحت في هذه الرسالة التمثيل الرياضي للروبوت عن طريق التصميم ثلاثي الأبعاد باستخدام برنامج (Autodesk Inventor).الروبوت عبارة عن نظام متعدد المداخل ومتعدد المخارج غير خطي , لذلك فهو بحاجة الي نظام تحكم (PID) أو (MBC) لتحسين مستوي الدقة والأداء للروبوت. كل نظام له نموذج ديناميكي خاص به , لذلك يصعب تحديد نوع المتحكم لكل نظام بدقة . لذلك هذه الرسالة تقارن نتائج تطبيق نوعين من المتحكمات (PID) و (MBC) علي الروبوت المتوازي دلتا ثلاثي الأبعاد واختيار المتحكم الذي يضمن للروبوت الأداء الأفضل.

بعد تمثيل ومقارنة النتائج المتوقعة من كلا نظامي التحكم (PID) ونظام التحكم (MBC) , اثبتت النتائج الإفتراضية بأن كلا المتحكمين أضافا تحسيناً كبيراً علي حركة الروبوت لكن المتحكم (MBC) أعطي نتائج أفضل من المتحكم (PID) . لكن بالرغم من هذا فإن نظام التحكم (PID) أفضل في التمثيل والتنفيذ عمليا علي الروبوت دلتا وذلك لسهولة برمجته علي انضمة التحكم الدقيق (المايكروكنترولر).

Table of Contents:

List of Figures

List of Tables

List of Abbreviations

TCP	Tool Center Point
DOF	Degree of Freedom
{R}	Reference frame at the base plate.
PID	Proportional Integral Derivative Control
MPC	Model Predictive Control
$\mathbf{J_x}$	Jacobian matrix in Cartesian space
$\mathbf{J_\theta}$	Jacobian matrix in joint space
PC	Personal Computer

Chapter 1

Introduction

1.1 Background

There are essentially two types of robot manipulators: serial and parallel. Serial manipulators consist of a number of links connected in series to one another to form a kinematic chain. Each joint of the kinematic chain is usually actuated. This type of structure is known as an open chained mechanism. Parallel manipulators, on the other hand, consist of a number of kinematic chains connected in parallel to one another. The kinematic chains work in unison to move a common point. This common point usually consists of a manipulator that performs a certain task. For the purpose of the three degrees of freedom (3 DOF) parallel DELTA robot system described in this thesis, the common point will also be referred to as the end effector. Since the kinematic chains are eventually connected to a common point, a parallel manipulator is considered a closed chained mechanism. The actuators in parallel manipulators are usually located at the base or close to the base of the system, which is in stark contrast to serial manipulators which have actuators at every joint. The advantages of this type of configuration include the fact that it could achieve a higher load capacity due to the decrease in the mass of the overall system, it can produce high accelerations at the end effector and it has a high mechanical stiffness to weight ratio [1].
The disadvantages of this type of configuration include the fact that the dynamic model is quite complex in nature and there are many instances of singularities that must be mapped out and avoided in order to maintain control of the system. Parallel robots come in a wide variety of designs and applications ranging from the Stewart platform or Hexapod Parallel Robot shown in Fig. 1.1.a, which is used in aircraft motion simulators to the Delta robot, which is used in packaging plants. This endows the fact that there cannot be a conclusive result as to which controller best suits the functionality of all parallel robots. Therefore, it is logical to experiment with various control techniques to observe upon which controller would garner the most satisfactory results based on a specific mechanical system.
This thesis presents the reader with the simulation results obtained from the implementation of PID control and Model Predictive Control (MPC) on 3 DOF

Parallel DELTA robots. The parameters of the dynamic model of this system are derived in detail followed by the derivation of the inverse kinematics of the mechanical model. The non-singular region is then defined based on the results obtained in the inverse kinematics. It is important to map out the non-singular region since it is the only location in which the parallel robot is able to operate under stable conditions. If the parallel robot were to enter a singular region, it would render the controller ineffective and cause the entire system to become unstable. It is impossible to adequately design any controllers for the parallel robot without a clear understanding of the dynamic model and the inverse kinematics of the mechanical model.

In recent years the number of studies and applications of parallel robots have increased. One of the most popular applications is in industry packaging. The above is due to their ease of construction, the lightness of their structure and the high accelerations obtained by these devices.

Unlike the serial-type robot manipulators, which only have an open-loop kinematic chain, parallel configuration allows for a distribution of payload among their two, or more closed-loop, kinematic chains. To illustrate this point consider Fig.1.1.a shows a parallel-architecture robot, used for object loading and unloading. Fig.1.2 shows a SCARA-type serial-architecture robot. By comparing the images it is easy to appreciate the difference between the two types of architecture. In the case of the serial manipulator greater robustness is required, as each link carries not only the weight of the successive links but also the motors and payload. This creates a cantilever effect in each link and, as a result, a greater deformation overall. In contrast, in the parallel architecture the actuators are fixed to the base of the manipulator so that the weight of the motors is not supported by the kinematic chains. In addition, the payload is distributed among the kinematic chains that con-form the manipulator. This results in thinner and lighter kinematic chains, which in turn results in an increased payload capacity of the manipulator, relative to its total mass.

(A) (B)

Figure 1.1: (a) Hexapod Parallel Robot, (b) CODIAN Robotics Parallel robot

Figure1.2: SCARA-type serial-architecture robot

A disadvantage of parallel robots is their typically low cost effectiveness, based on complex kinematics and rather expensive control units, as well as the poor workspace to robot-dimension ratio [1].

On the other hand, the advantages of parallel robots stated before indicate that their capabilities can be optimally oriented if their specifications are task-adapted to the desired application. To facilitate flexibility and to enlarge the field of application, it is reasonable to use a reconfigurable robot design. This will also help to overcome the typical challenges of parallel robots, such as high costs and undersized workspaces.

1.2 Motivation

Modeling and Digital control of Dynamic system was my target of my thesis; Autodesk Inventor Software has a power capability in modeling of mechanical systems. No need to extract the dynamic equations of robot. Simulink tool of Matlab can simulate the body of the robot as built in Autodesk Inventor software program.

This Master Thesis treats the modeling of the Parallel DELTA robot actuated with Servo DC motors and drive units. Also the kinematics for a Delta-3 robot is implemented to be able to see where the traveling plate has for position for different arm angle configurations.

1.3 Objectives and Methodology

Objectives of this thesis can be summarized as follows:

Design and Building three legs Delta Parallel Robot: therefor, Delta Robot was designed via Autodesk Inventor software program.
Forward and inverse kinematics analysis: Both forward and inverse kinematic algorithms have been developed, which are essential for the motion planning and control of a parallel robot.
Workspace analysis: It is necessary to ensure that delta parallel robot has a reasonable workspace volume. Workspace analysis is also required in the design of the parallel robot. Hence, a workspace visualization scheme has been developed for the modular parallel robot system. PID and MPC

controllers are used to improve the end effector path tracking for the DELTA robot.

1.4 Problem Statement

The optimal control problem can be stated as: find a closed loop optimal controller that minimizes error between the measured phase and actual phase wanted to track specified path. Optical encoder is attached in the end of each Servo motor shaft, measuring the actual phase of the link and from that; we can calculate radius speed and acceleration.
Controlling of Delta Parallel robot wants true modeling for its dynamics, so, by using Autodesk Inventor program, we can model the robot easily and test its motion in Simulink Matlab tool.

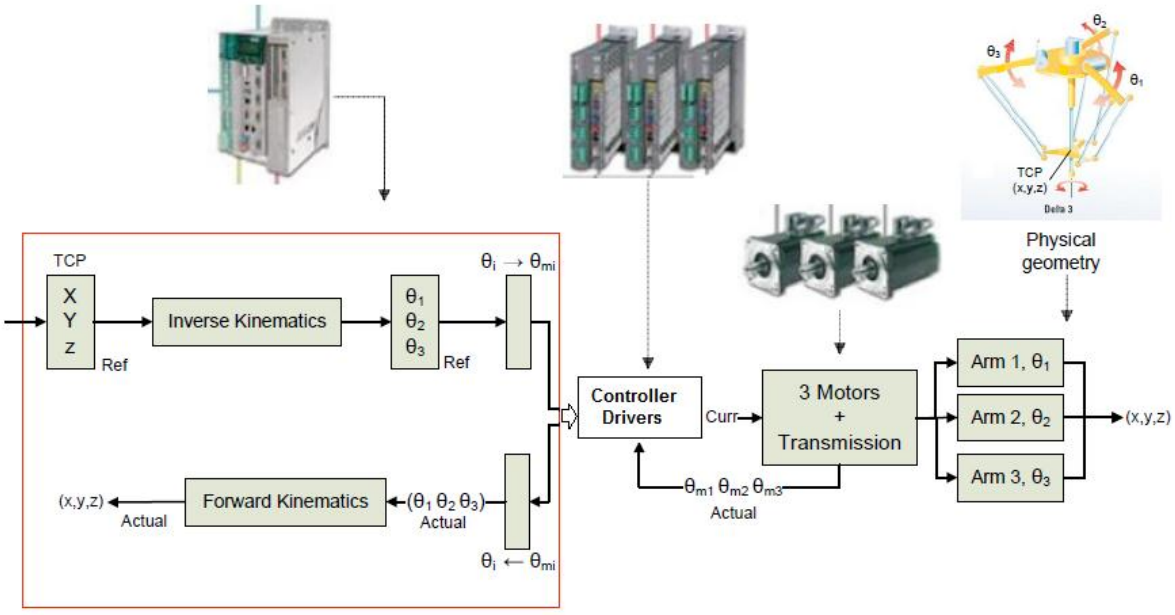


Figure 1.3: DELTA Robot Control System.

The three Drivers control one motor each to actuate the three arms at the Delta-3 robot. From Fig.1.3, Controller unit calculates inverse kinematics of reference position x,y,z of moving platform in delta robot , actuators drive servo motors under the effect of PID controller leading the end effector to the target point in minimum time and no error as possible .

1.5 Thesis Contribution

In this thesis, a new mechanical model of Delta parallel Robot was introduced, and a digital Controller system based on microcontroller chips are interfaced directly to PC computer via serial communication. Simulink Matlab tool can communicate by hardware Controller unit, which compute the kinematics of delta robot in place of Microcontroller. This model opens a new road to master students to use other control systems and contribute the motion precision of the moving platform.

1.6 Literature Review

Modeling and control of a Closed Chain Parallel DELTA Robot is very difficult especially, when using traditional methods in modeling.

- YangminLi, Qingsong Xu [3] proposed the simplified dynamic equations derived via the virtual work principle on 3-TRC translational parallel kinematic machine.

- André Olsson [4] describes the virtual work principle mathematical modeling of a Delta-3 robot actuated by motors and drive units. Experiments with comparison between the Simulink model and the real robot are done.

- Angelo Liadis[5] proposed Lagrangian principle for modeling 2 DOF parallel robot, and introduced eight controllers , fuzzy and non-fuzzy controllers. Experiments with comparison between the Simulink model and the real robot are done.

- Mohsen, Mahdi, Mersad [9] describes the Dynamics modeling and trajectory tracking control of a new structure of spatial parallel robots from Delta robots family. This paper compared implementation of computed torque (C-T) method using adaptive Neuro-fuzzy controller and conventional PD controller.

- Yangmin Li and Qingsong Xu [12] performed inverse dynamic modeling based upon the principle of virtual work for medical Delta Robot. The dynamic control uses computed torque method.

1.7 Thesis Overview

The purpose of this thesis is to determine the most appropriate controller to implement on a 3 DOF DELTA parallel robot apparatus. Chapter 2 will discuss and derive the equations for the modelling of the parallel robot using the dynamic equations of the constrained system and the inverse kinematics of the mechanical structure. Chapter 3 will consist of the derivations of PID and Model Predictive Controllers (MPC). Chapter 4 describes the path planning which the robot must follow to travel from point to another point in Cartesian space. Chapter 5 will compare and analyze the simulation results of each controller utilizing MATLAB. The plots of the joint angles and end effector trajectory along with their respective errors and torque will be compared between all the controllers and a generalized conclusion of these simulation results will be garnered. Chapter 6 will entail the overall recommendation of the candidate controller which best suits the needs of the parallel robot system. A description of the improvements or additions that can be executed in future research endeavors will be investigated to conclude this thesis.

Chapter 2

Kinematics and Dynamics

As mentioned in the previous chapter, I introduced the advantages and disadvantages of parallel robot and compared their performance with serial type robots. In this chapter, I will study the kinematics of 3 DOF Parallel DELTA robot.

2.1 DELTA Type Parallel Robot

The well-known Delta robot structure was proposed by R. Clavel in [2]. Fig. 2.1 shows the main components of this robot, which consists of three or four closed-loop kinematic chains. The robot has three degrees of freedom.

The parallelograms ensure the constant orientation between the fixed and the mobile platform, allowing only translation movements of the latter. The end effector of the manipulator is located on the mobile platform [3].
Parallel Robot can move products in a three dimensional Cartesian coordinate system.

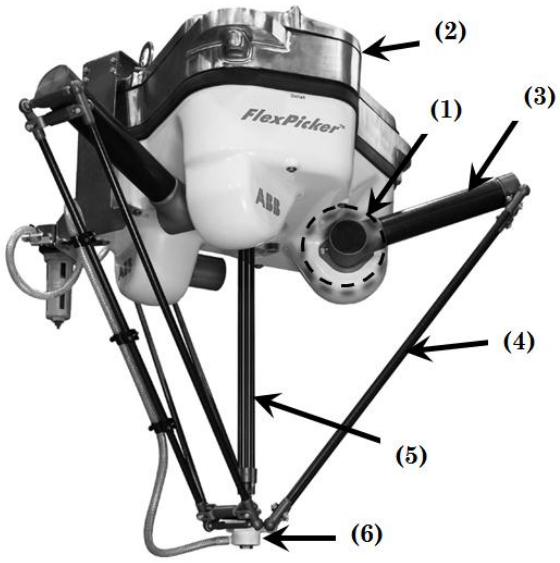


Figure 2.1: Parallel DELTA Robot Components

The combination of the constrained motion of the three arms connecting the traveling plate to the base plate ensues in a resulting 3 translator degrees of freedom (DOF). As an option, with a rotating axis at the Tool Center Point (TCP), four DOF are possible.

The Robot consists of, consider Fig.2.1:

1) Three Actuators.
2) Base plate.
3) Upper robot arm.
4) Lower robot arm (Forearm).
5) Rotation arm (optional, 4-DOF).
6) Travelling plate, TCP.

The upper robot arms are mounted direct to the actuators to guarantee high stability. And the Three actuators are rigidly mounted on the base plate with 120° in between. Each of the three Lower robot arms consists of two parallel bars, which connects the upper arm with the travelling Plate via ball joints. Lower frictional forces result from this. The wear reduces respectively as a result. To measure each motor shaft angle a Quadrature Optical encoder is used. A fourth bar, rotational axes, is available for the robot mechanics as an option. The actuator for this axis is then mounted on the upper side of the robot base plate. The bar is connected directly to the tool and ensures for an additional rotation motion [4].

2.2 Inverse Kinematics

The purpose of determining the inverse kinematics of this parallel robot is to accurately model the angle produced at each joint at a specific location of the e effector. This is advantageous for two main reasons; the first being that it is relatively simple to define any reasonable trajectory for the end effector to traverse and secondly, it can track different trajectories in a non-singular region [5].

The constrained three degrees of freedom system shown in Fig. 2.3 will also be applicable in this section. It should be noted that the parameters of the overall system are known, which include: the range of the desired angles for θ_1, θ_2 and θ_3 respectively, the overall length of each upper link L_a and the overall length of each lower link L_b. the desired location of the end effector in

the x and y axis respectively and the horizon distance between the two motor shafts (c).

The problem of the Inverse kinematics solution is to find the actuators states θ_1, θ_2, θ_3, known the end-effector position ($\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$). To find the inverse kinematics solution let us refer to Fig. 2.3 Also, let consider the origin of the reference system fixed on the platform and the axes such as depicted in Fig. 2.3 Note that the parallelogram has been considered as a single link ($\boldsymbol{l_b}$).

`

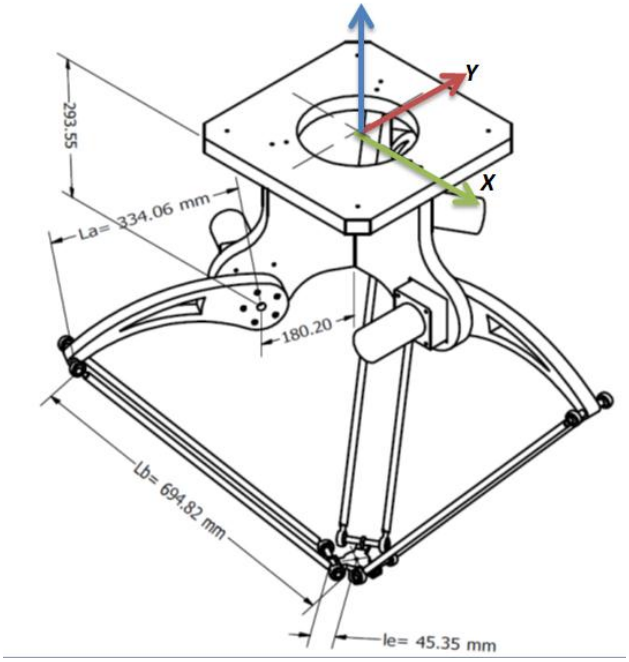


Figure 2.2: Delta Parallel Robot side View Designed on Autodesk Inventor.

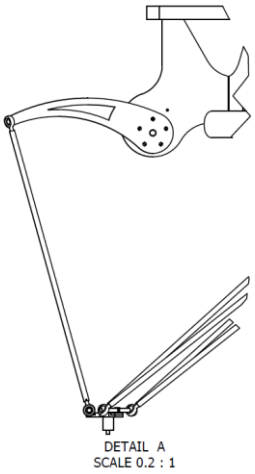


Figure 2.3: one link side view of DELTA Parallel Robot.

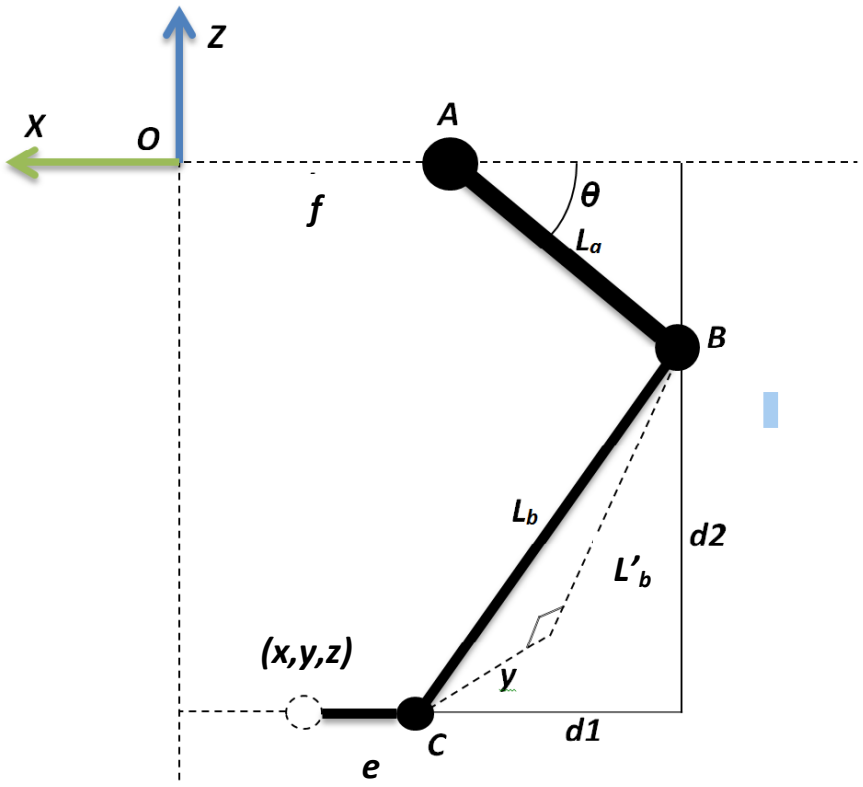


Figure 2.4: First kinematics chain, XZ plane projection.

The analysis begins considering each kinematics chain separately, for the first kinematics chain; shown in Fig.2.3, we make a projection to the X-Z plane, which yields a vector closed loop as shown in Fig.2.4.

From Fig. 2.4, we have:

$$d_1^2 + d_2^2 = l_b^{'2} \tag{2.1}$$

Where,

$$l_b^{'2} = l_b^2 - y^2 \tag{2.2}$$

In addition we have that:

$$OA + l_a \cos(\theta_1) = -x + DC + d_1$$

Solving for d_1, yields:

$$d_1 = T + l_a \cos(\theta_1) \tag{2.3}$$

Where,

$$T = OA + x - DC$$

Also, from the geometry of Fig. 2.5 we have,

$$d_2 = z + l_a \sin(\theta_1) \tag{2.4}$$

Substituting (2.2), (2.3) and (2.4) in (2.1) and simplifying, we obtain:

$$2Tl_a \cos(\theta_1) + 2zl_a \sin(\theta_1) = K \tag{2.5}$$

With,

$$\mathrm{K} = l_a^2 - l_b^2 - x_0^2 - z^2 - T^2$$

Substituting in (2.5) the trigonometric identities,

$$\cos(\theta_1) = \frac{1-t^2}{1+t^2} \text{ , } \sin(\theta_1) = \frac{2t}{1+t^2} \text{ , where, } t = \tan\left(\frac{\theta_1}{2}\right)$$

We obtain:

$$\mathrm{e}_1 t^2 + e_2 t + e_3 = 0 \tag{2.6}$$

Where:

$$e_1 = 2Tl_a + K$$
$$e_2 = -4zl_a$$
$$e_3 = -2Tl_a + K$$

Solving (2.6) for t yields,

$$\theta_1 = 2\tan^{-1}\frac{-e_2 \pm \sqrt{e_2^2 - 4e_1e_3}}{2e_1} \tag{2.7}$$

From the previous equations, we can conclude that,

$$\theta_1 = f(x, y, z) \tag{2.8}$$

Following the same procedure, the others two kinematics chains configurations can be solved. We can take advantage of the symmetry of Delta Robot and consider the fact that each kinematics chain is rotated 120 degree relative to each other. We could take the base the first kinematics chain and multiply it by the rotation matrix (120^O for θ_2 and 240^O for θ_3) and then apply the process used to solve the first kinematics chain. Once followed the procedure described previously, the values of θ_2 and θ_3 can be found. In general, there are a total of eight possible robot postures corresponding to a given end-effector location [6].

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} cos(a) & -sin(a) & 0 \\ sin(a) & cos(a) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2.9}$$

Where,

$$\begin{aligned} x' &= \cos(\alpha).x - \sin(\alpha).y \\ y' &= \sin(\alpha).x + \cos(\alpha).\mathrm{y} \\ z' &= z \end{aligned} \tag{2.10}$$

Where α is the angle of rotation about z axis, From Eq. (2.10) yields:

$$\theta_{2,3} = f(x', y', z') \tag{2.11}$$

Hence, there are generally two solutions of θ_1 and therefore two configuration of the kinematics chain Fig. 2.5 corresponding to each end-effector location. When Eq. 2.7 yields a double root, the two links of the kinematics chain are in a fully stretched-out or folded-back configuration named singular

configuration. When Eq. 2.7 yields no real solution, the specified end-effector location is not reachable. Despite of the two possible solutions, only the negative root have to be taken because the positive one could cause interference between the elements of the robot as depicted in Fig 2.5.

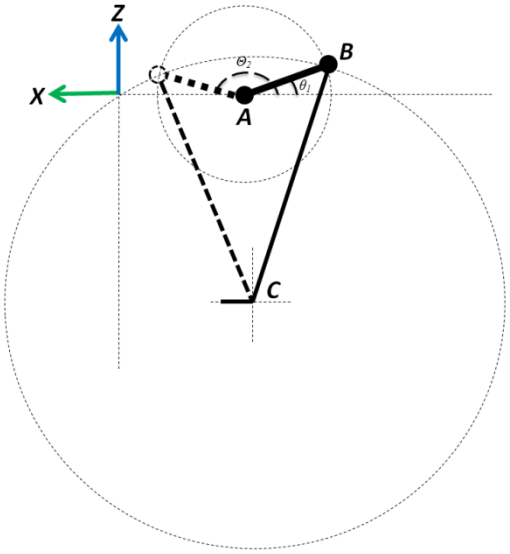


Figure 2.5: Two possible configurations of the kinematics chain due to θ_1.

The inverse kinematics solution is tested for special cases by examining Eq. 2.7: If, $e_2^2 - 4e_1e_3 > 0$ then the circle swept by vector AB intersects the sphere swept by vector BC in two locations. If, $e_2^2 - 4e_1e_3 = 0$, then the circle and sphere are tangent, and the manipulator is in a singular position. If, $e_2^2 - 4e_1e_3 < 0$, then the circle and the sphere do not intersects and there are no real solutions. If e1=e2 =e3=0, then the circle lies on the sphere, and there are infinite number of solutions [1].

2.3 Forward Kinematics

The forward kinematics also called the direct kinematics of a parallel manipulator determines the *(x, y, and z)* position of the travel plate in base-frame, given the configuration of each angle θ*i* of the actuated revolute joints, see Fig.2.6.

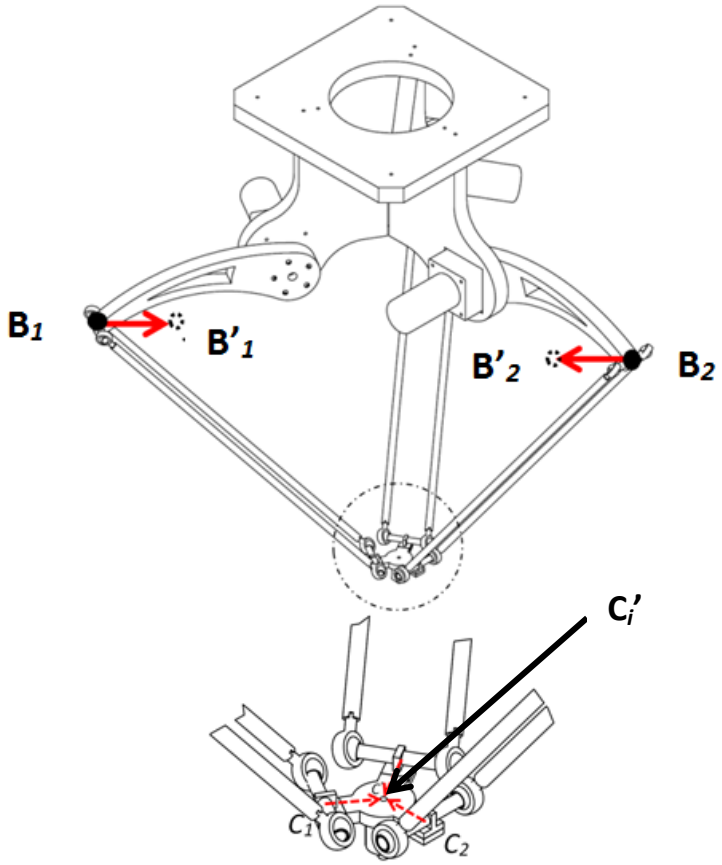


Figure 2.6: Configuration chosen for direct kinematics analysis

Consider three spheres each with the center at the elbow $\mathbf{B}_i$ of each robot arm chain, and with the forearms lengths $\mathbf{l}_b$ as radius. The forward kinematic model for a Parallel Delta Robot can then be calculated with help of the intersection between these three spheres. When visualizing these three spheres they will intersect at two places.

One intersection point where z is positive and one intersection point where z coordinate is negative. Based on the base frame {R} where z -axis is positive

upwards the TCP will be the intersection point when z is negative. Fig. 2.7 shows the intersection between three spheres. Where two spheres intersects in a circle and then the third sphere intersects this circle at two places.

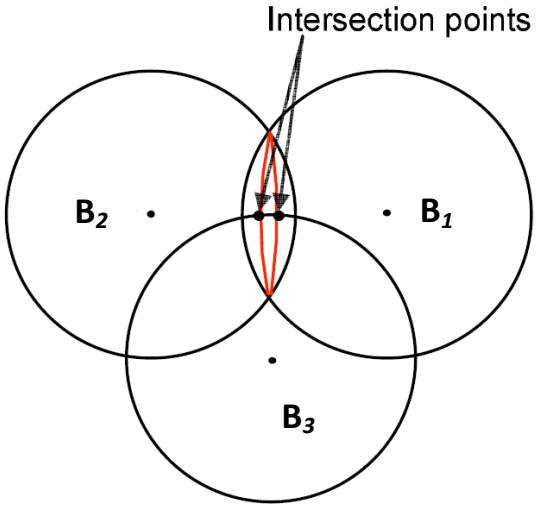


Figure 2.7: two spheres intersect in a circle and a third sphere intersect the circle at two places

Based on the model assumptions made, the vector B*i* that describes the elbow coordinates for each of the three arms as

$$B_i = [\, f + l_a \cos(\theta_i) \quad 0 \quad l_a \sin(\theta_i)]^T \tag{2.12}$$

To calculate the direct kinematics we move the center of the spheres to inside from points B_i to the points B_i' for i=1, 2 and 3 respectively. After this transition the three spheres will intersect in the TCP point.

$$B_i^{'} = [(f - e) + l_a \cos(\theta_i) \quad 0 \quad l_a \sin(\theta_i]^T \tag{2.13}$$

Where $\boldsymbol{e} = B_1B_1^{'} = B_2B_2^{'} = B_3B_3^{'}$ is the length of shifted distance, clearly described in Fig.2.6.

To achieve a matrix that describes all of the three points in the base frame {**O**} one has to multiply B_i with the rotational matrix $\mathbf{R}_z^0$:

$$\mathbf{R}_z^0 = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.14}$$

The result is the matrix $\mathbf{B}^{'}$,

$$\mathbf{B}^{'} = \mathbf{R}_z^0 \ \mathbf{B}_i^{'} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix} [(f-e)+l_a\cos(\theta_i) \quad 0 \quad l_a\sin(\theta_i]^T$$

$$\mathbf{B}^{'} = \begin{bmatrix} \cos(\alpha)B_{i,x}^{'} \\ \sin(\alpha)B_{i,x}^{'} \\ B_{i,\mathrm{z}}^{'} \end{bmatrix} = \begin{bmatrix} \cos(\alpha)[(f-e)+l_a\cos(\theta_i)] \\ \sin(\alpha)[(f-e)+l_a\cos(\theta_i)] \\ l_a\sin(\theta_i) \end{bmatrix} = \begin{bmatrix} s_{i,x} \\ s_{i,\mathrm{y}} \\ s_{i,\mathrm{z}} \end{bmatrix} \tag{2.15}$$

Then can three spheres be created with the forearms lengths l_b as radius, and their centers in B_i respectively. The general equation for a sphere is

$$(x-s_{i,x})^2+(y-s_{i,\mathrm{y}})^2+(z-s_{i,\mathrm{z}})^2=r^2 \tag{2.16}$$

This gives the three equations for three links i =1,2 and 3 respectively. For link (1) the upper arm is parallel to x- axis and perpendicular to y-axis, so the rotation angle α=0, but the other two links have a rotation angles α = 120 for link (2) and α = -120 for link (3).

$$\begin{aligned}
&(x-\cos(\alpha_1)[(f-e)+l_a\cos(\theta_1)])^2+(y-\sin(\alpha_1)[(f-e)+l_a\cos(\theta_1)])^2+(z-l_a\sin(\theta_1))^2=l_b{}^2\\
&(x-\cos(\alpha_2)[(f-e)+l_a\cos(\theta_2)])^2+(y-\sin(\alpha_2)[(f-e)+l_a\cos(\theta_2)])^2+(z-l_a\sin(\theta_2))^2=l_b{}^2\\
&(x-\cos(\alpha_3)[(f-e)+l_a\cos(\theta_3)])^2+(y-\sin(\alpha_3)[(f-e)+l_a\cos(\theta_3)])^2+(z-l_a\sin(\theta_3))^2=l_b{}^2
\end{aligned} \tag{2.17}$$

After substitution the values α_1=0, α_2 = 120, and α_3=-120 in Eq. 13, we get the three sphere equations,

$$
\begin{aligned}
&(x-[(f-e)+l_a\cos(\theta_1)])^2+(y)^2+(z-l_a\sin(\theta_1))^2=l_b{}^2\\
&(x+\frac{1}{2}[(f-e)+l_a\cos(\theta_2)])^2+(y-\frac{\sqrt{3}}{2}[(f-e)+l_a\cos(\theta_2)])^2+(z-l_a\sin(\theta_2))^2=l_b{}^2\\
&(x+\frac{1}{2}[(f-e)+l_a\cos(\theta_3)])^2+(y+\frac{\sqrt{3}}{2}[(f-e)+l_a\cos(\theta_3)])^2+(z-l_a\sin(\theta_3))^2=l_b{}^2
\end{aligned}
\qquad (2.18)
$$

Rearrange Eq. 2.18 we obtain,

$$
\begin{aligned}
&(x+k_{11})^2+(y+k_{12})^2+(z+k_{13})^2=l_b{}^2\\
&(x+k_{21})^2+(y+k_{22})^2+(z+k_{23})^2=l_b{}^2\\
&(x+k_{31})^2+(y+k_{32})^2+(z+k_{33})^2=l_b{}^2
\end{aligned}
\qquad (2.19)
$$

Where,

$$
\begin{aligned}
k_{11}&=(f-e)+l_a\cos(\theta_1)\\
k_{12}&=0\\
k_{13}&=-l_a\sin(\theta_1)\\
k_{21}&=\frac{1}{2}[(f-e)+l_a\cos(\theta_2)]\\
k_{22}&=-\frac{\sqrt{3}}{2}[(f-e)+l_a\cos(\theta_2)]\\
k_{23}&=-l_a\sin(\theta_2)\\
k_{31}&=\frac{1}{2}[(f-e)+l_a\cos(\theta_3)]\\
k_{32}&=\frac{\sqrt{3}}{2}[(f-e)+l_a\cos(\theta_3)]\\
k_{33}&=-l_a\sin(\theta_3)
\end{aligned}
$$

After expanding Eq. 2.19 we obtain,

$$x^2 + y^2 + z^2 + 2k_{i1}x + 2k_{i2}\,\mathrm{y} + 2k_{i3}\,\mathrm{z} = l_b{}^2 - (k_{i1}^2 + k_{i2}^2 + k_{i3}^2)\ ,\ \mathrm{i}=1,2,3 \qquad (2.20)$$

Subtract Eq.2.20 with i=2 from Eq. 2.20 with i=1, we obtain,

$$2(k_{11} - k_{21})x + 2(k_{12} - k_{22})\,\mathrm{y} + 2(k_{13} - k_{23})\,\mathrm{z} = (k_{21}^2 + k_{22}^2 + k_{23}^2) - (k_{11}^2 + k_{12}^2 + k_{13}^2) \qquad (2.21)$$

Subtract Eq.2.20 with i=3 from Eq. 2.20 with i=1, we obtain,

$$2(k_{11} - k_{31})x + 2(k_{12} - k_{32})\,\mathrm{y} + 2(k_{13} - k_{33})\,\mathrm{z} = (k_{31}^2 + k_{32}^2 + k_{33}^2) - (k_{11}^2 + k_{12}^2 + k_{13}^2) \qquad (2.22)$$

Simplifying Eq.2.21 and Eq.2.22 we obtain,

$$\begin{aligned} a_1 x + b_1 y + c_1 z &= d_1 \\ a_2 x + b_2 y + c_2 z &= d_2 \end{aligned} \qquad (2.23)$$

Where,

$$\begin{aligned} a_1 &= 2(k_{11} - k_{21}) \\ b_1 &= 2(k_{12} - k_{22}) \\ c_1 &= 2(k_{13} - k_{23}) \\ a_2 &= 2(k_{11} - k_{31}) \\ b_2 &= 2(k_{12} - k_{32}) \\ c_2 &= 2(k_{13} - k_{33}) \end{aligned}$$

And,

$$\begin{aligned} d_1 &= (k_{21}^2 + k_{22}^2 + k_{23}^2) - (k_{11}^2 + k_{12}^2 + k_{13}^2) \\ d_2 &= (k_{31}^2 + k_{32}^2 + k_{33}^2) - (k_{11}^2 + k_{12}^2 + k_{13}^2) \end{aligned}$$

Arranging Eq.2.23 in a matrix form, we obtain,

$$\begin{bmatrix} a_1 & b_1 \\ a_2 & b_2 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}=\begin{bmatrix} d_1 - c_1 z \\ d_2 - c_2 z \end{bmatrix} \tag{2.24}$$

Define $\Delta = a_1 b_2 - a_2 b_1$, then for case $\Delta \neq 0$,

$$\begin{aligned} \Delta x &= (d_1 - c_1 z) b_2 - (d_2 - c_2 z) b_1 \\ &= (b_2 d_1 - b_1 d_2) + (b_1 c_2 - b_2 c_1)\,\mathrm{z} \\ \Delta y &= (d_2 - c_2 z) a_2 - (d_1 - c_1 z) a_1 \\ &= (a_1 d_2 - a_2 d_1) + (a_2 c_1 - a_1 c_2)\,\mathrm{z} \end{aligned}$$

$$\begin{aligned} x &= \frac{\Delta x}{\Delta} = \frac{b_2 d_1 - b_1 d_2}{\Delta} + \frac{b_1 c_2 - b_2 c_1}{\Delta} z \\ y &= \frac{\Delta y}{\Delta} = \frac{a_1 d_2 - a_2 d_1}{\Delta} + \frac{a_2 c_1 - a_1 c_2}{\Delta} z \end{aligned} \tag{2.25}$$

Consider,

$$f_1 = \frac{b_2 d_1 - b_1 d_2}{\Delta}\ ,\ f_2 = \frac{b_1 c_2 - b_2 c_1}{\Delta}$$
$$f_x = \frac{b_1 c_2 - b_2 c_1}{\Delta}\ ,\ f_y = \frac{a_2 c_1 - a_1 c_2}{\Delta}$$

Then,

$$\begin{aligned} x &= f_1 + f_x z \\ y &= f_2 + f_y z \end{aligned} \tag{2.26}$$

Substituting Eq. 2.26 in Eq.2.19 for i=3; we obtain,

$$(1 + f_x^2 + f_y^2)\,\mathrm{z}^2 + 2([f_x f_1 + f_x k_{31}] + [f_y f_2 + f_y k_{32}] + k_{33}) z + f_{11}^2 + f_{22}^2 + k_{33}^2 - l_b^2 = 0 \tag{2.27}$$

Let,

$$A = (1 + f_x^2 + f_y^2)$$
$$B = 2([f_x f_1 + f_x k_{31}] + [f_y f_2 + f_y k_{32}] + k_{33})$$
$$C = f_{11}^2 + f_{22}^2 + k_{33}^2 - l_b^2$$

Where,

$$f_{11} = f_1 + k_{31}$$
$$f_{22} = f_2 + k_{32}$$

The solution of Equation $Az^2 + Bz + C = 0$ is well known as

$$z = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A} \tag{2.28}$$

From Eq.2.28, we can Evaluates Eq.2.26.

Mathematically neither forward nor inverse kinematics gives single solution. Forward kinematics usually has two solutions, because the passive joint angles formed between upper arm and lower arm are not determined by kinematic equations. Then the solution that is within the robots work area must be chosen. With the base frame {O} in this case, it will lead to the solution with negative z coordinate.

The output solution has Four cases are possible:

1) Generic solution. The two solutions are realized at the intersection of a circle and a sphere.
2) Singular solution. Once sphere is tangent to the circle of intersection of the other two spheres, hence there is only one solution possible.
3) Singular solution. The center of any two spheres coincides, resulting in an infinite number of solutions. This is an unlikely configuration for most practical embodiments of the manipulator, except for the situation when $\theta_1 = \theta_2 = \theta_3 = \pi/2$.
4) No solution. The three spheres do not intersect at a common point.

2.4 Velocity Kinematics

The most relevant loop should be picked up for the intended Jacobian analysis. Let $\vec{\theta}$ be the vector made up of actuated joint variables and $\vec{\mathbf{P}}$ is the position vector of the moving platform. Then

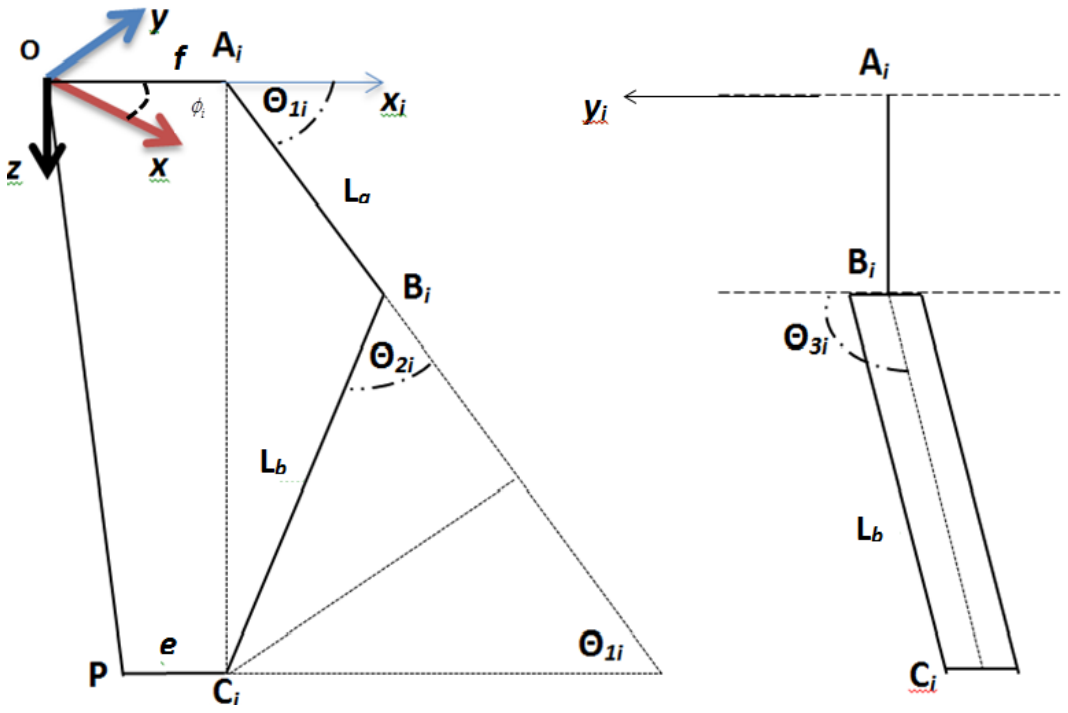


Figure 2.8: (a) Projection of link i *on* $x_i z_i$ plane, (b) end on view

$$\vec{\boldsymbol{\theta}} = \theta_{1i} = \begin{bmatrix} \theta_{11} \\ \theta_{12} \\ \theta_{13} \end{bmatrix}, \vec{\mathbf{P}} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2.29}$$

The Jacobian matrix will be derived by differentiating the appropriate loop closure equation and rearranging the result in the following form

$$\mathbf{J_\theta}\begin{bmatrix}\dot{\theta}_{11}\\ \dot{\theta}_{11}\\ \dot{\theta}_{11}\end{bmatrix} = \mathbf{J_P}\begin{bmatrix}\dot{P}_x = v_x\\ \dot{P}_y = v_y\\ \dot{P}_z = v_z\end{bmatrix} \tag{2.30}$$

where v_x, v_y, and v_z are the x, y, and z components of the velocity of the point ***P*** on the moving platform in the xyz frame. In order to arrive at the above form of the equation, we look at the loop $\boldsymbol{OA_iB_iC_iP}$. The corresponding closure equation in the $\mathbf{x_iy_iz_i}$ frame is

$$\overrightarrow{\mathbf{OP}} + \overrightarrow{\mathbf{PC_i}} = \overrightarrow{\mathbf{OA_i}} + \overrightarrow{\mathbf{A_iB_i}} + \overrightarrow{\mathbf{B_iC_i}} \tag{2.31}$$

In the matrix form we can write it as

$$\begin{bmatrix}P_x\cos\phi_i - P_y\cos\phi_i\\ P_x\sin\phi_i + P_y\cos\phi_i\\ P_z\end{bmatrix} = \begin{bmatrix}f\\0\\0\end{bmatrix} - \begin{bmatrix}e\\0\\0\end{bmatrix} + l_a\begin{bmatrix}\cos\theta_{1i}\\0\\\sin\theta_{1i}\end{bmatrix} + l_b\begin{bmatrix}\sin\theta_{3i}\cos(\theta_{2i}+\theta_{1i})\\ \sin\theta_{3i}\\ \sin\theta_{3i}\cos(\theta_{2i}+\theta_{1i})\end{bmatrix} \tag{2.32}$$

Time differentiation of this equation leads to the desired Jacobian equation. The loop closure equation Eq.2.31 can be re-written as

$$(\vec{\mathbf{P}} + \vec{\mathbf{e}}) = \vec{\mathbf{f}} + \overrightarrow{\mathbf{a_i}} + \overrightarrow{\mathbf{b_i}} \tag{2.33}$$

Where $\overrightarrow{a_i}$ and $\overrightarrow{b_i}$ represents vectors $\overrightarrow{A_iB_i}$ and $\overrightarrow{B_iC_i}$ respectively.

Differentiating Eq.2.33 with respect to time and using the fact that $\vec{f}$ is a vector characterizing the fixed platform, and $\vec{e}$ is a vector characterizing the moving platform

$$\dot{\vec{\mathbf{P}}} = \vec{\mathbf{v}} = \dot{\vec{\mathbf{a}}}_\mathbf{i} + \dot{\vec{\mathbf{b}}}_\mathbf{i} \tag{2.34}$$

The linear velocities on the right hand side of Eq.2.34 can be readily converted into the angular velocities by using the well-known identities.

Thus

$$\vec{\mathbf{v}} = \overrightarrow{\mathbf{w}_{ai}} \times \vec{\mathbf{a}}_i + \overrightarrow{\mathbf{w}_{bi}} \times \vec{\mathbf{b}}_i \tag{2.35}$$

w_{ai} and w_{bi} is the angular velocity of the link i. To eliminate w_{bi}, it is necessary to dot-multiply both sides of Eq. 2.35 and $\mathbf{b}_i$. Therefore

$$\mathbf{b}_i . \vec{\mathbf{v}} = \overrightarrow{\mathbf{w}_{ai}} . (\vec{\mathbf{a}}_i \times \vec{\mathbf{b}}_i) \tag{2.36}$$

Rewriting the vectors of Eq.2.36 in the $x_i y_i z_i$ coordinate frame leads to

$$a_i = l_a \begin{bmatrix} \cos\theta_{1i} \\ 0 \\ \sin\theta_{1i} \end{bmatrix}, \; b_i = l_b \begin{bmatrix} \sin\theta_{3i}\cos(\theta_{2i}+\theta_{1i}) \\ \sin\theta_{3i} \\ \sin\theta_{3i}\cos(\theta_{2i}+\theta_{1i}) \end{bmatrix}$$

$$w_i = \begin{bmatrix} 0 \\ -\dot{\theta}_{1i} \\ 0 \end{bmatrix}, \quad v = \begin{bmatrix} v_x\cos\phi_i - v_y\cos\phi_i \\ v_x\sin\phi_i + v_y\cos\phi_i \\ v_z \end{bmatrix}$$

Substituting the values of a_i, b_i,v_i and $\boldsymbol{v}$ in Eq.2.36 leads to

$$j_{ix}v_x + j_{iy}v_y + j_{iz}v_z = l_a \sin\theta_{2i}\sin\theta_{3i}\dot{\theta}_{1i} \tag{2.37}$$

Where,

$$j_{ix} = \cos(\theta_{1i}+\theta_{2i})\sin\theta_{3i}\cos\phi_i - \cos\theta_{3i}\sin\phi_i$$
$$j_{iy} = \cos(\theta_{1i}+\theta_{2i})\sin\theta_{3i}\sin\phi_i + \cos\theta_{3i}\cos\phi_i$$
$$j_{iz} = \sin(\theta_{1i}+\theta_{2i})\sin\theta_{3i}$$

Expanding Eq.2.37 for i = 1, 2 and 3 yields three scalar equations which can be assembled into a matrix form as

$$\mathbf{j}_x \mathbf{v} = \mathbf{j}_q \dot{\mathbf{q}} \tag{2.38}$$

Where,

$$\mathbf{j_x} = \begin{bmatrix} j_{1x} & j_{1y} & j_{1z} \\ j_{2x} & j_{2y} & j_{2z} \\ j_{3x} & j_{3y} & j_{3z} \end{bmatrix}$$

$$\mathbf{j_q} = l_a \begin{bmatrix} \sin\theta_{21}\sin\theta_{31} & 0 & 0 \\ 0 & \sin\theta_{22}\sin\theta_{32} & 0 \\ 0 & 0 & \sin\theta_{23}\sin\theta_{33} \end{bmatrix}$$

$$\dot{\mathbf{q}} = \begin{bmatrix} \dot{\theta}_{11} & \dot{\theta}_{12} & \dot{\theta}_{31} \end{bmatrix}^T$$

After algebraic manipulations, it is possible to write

$$\mathbf{v = J\dot{q}} \tag{2.39}$$

Where,

$$\mathbf{J = j_x^{-1} j_q} = \begin{bmatrix} \frac{\partial x}{\partial \theta_1} & \frac{\partial y}{\partial \theta_2} & \frac{\partial z}{\partial \theta_3} \\ \frac{\partial x}{\partial \theta_1} & \frac{\partial y}{\partial \theta_2} & \frac{\partial z}{\partial \theta_3} \\ \frac{\partial x}{\partial \theta_1} & \frac{\partial y}{\partial \theta_2} & \frac{\partial z}{\partial \theta_3} \end{bmatrix} \tag{2.40}$$

2.5 Forward and Inverse Singularity analysis

From Eq.2.38 it can be observed that singularity occurs:

1. when det(**Jq**) = 0. This means that either $\theta_{2i} = 0$ or , $\theta_{3i} = 0$ or π for i=1,2,3.
2. when det(**Jx**) = 0. This means that $\theta_{1i} + \theta_{2i} = 0$ or π or $\theta_{3i} = 0$ or π for i=1,2,3.
3. when det(**Jq**)=0 and det(**Jx**) =0. This situation occurs when $\theta_{3i} = 0$ or π for i=1,2 and 3.

In summary, singularity of the parallel manipulator occurs:

1. When all three pairs of the follower rods are parallel. Therefore, the moving platform has three degrees of freedom and moves along a spherical surface and rotates about the axis perpendicular to the moving platform

2. When two pairs of the follower rods are parallel. The moving platform has one degree of freedom; i.e. the moving platform moves in one direction only.

3. When two pairs of the follower rods are in the same plane or two parallel planes. The moving platform has one degree of freedom; i.e. the moving platform rotates about the horizontal axis only.

2.6 Dynamic Equations

Dynamics is the science of motion. It describes why and how a motion occurs when forces and moments are applied on massive bodies. The motion can be considered as evolution of the position, orientation, and their time derivatives. In robotics, the dynamic equation of motion for manipulators is utilized to set up the fundamental equations for control. The links and arms in a robotic system are modeled as rigid bodies.

Therefore, the dynamic properties of the rigid body take a central place in robot dynamics. Since the arms of a robot may rotate or translate with

respect to each other, translational and rotational equations of motion must be developed and described in body-attached coordinate frames B1, B2, B3 … or in the global reference frame G.

There are basically two problems in robot dynamics.

Problem1. We want the links of a robot to move in a specified manner. What forces and moments are required to achieve the motion?
The first Problem is called direct dynamics and is easier to solve when the equations of motion are in hand because it needs differentiating of kinematics equations. The first problem includes robots statics because the specified motion can be the rest of a robot. In this condition, the problem reduces finding forces such that no motion takes place when they act. However, there are many meaningful problems of the first type that involve robot motion rather than rest. An important example is that of finding the required forces that must act on a robot such that its end-effector moves on a given path and with a prescribed time history from the start configuration to the final configuration.

Problem2. The applied forces and moments on a robot are completely specified. How will the robot move?
The second problem is called inverse dynamics and is more difficult to solve since it needs integration of equations of motion. However, the variety of the applied problems of the second type is interesting. Problem 2 is essentially a prediction since we wish to find the robot motion for all future times when the initial state of each link is given.

In this section, we will perform the inverse dynamic modeling of the parallel manipulator based upon the principle of virtual work. The inverse dynamics problem is to find the actuator torques and/or forces required to generate a desired trajectory of the manipulator.[theory of applied robotics boo]

It is often convenient to express the dynamic equations of a manipulator in a single equation that hides some of the details, but shows some of the structure of the equations. The state-space equation When the Newton—Euler equations are evaluated symbolically for any manipulator, they yield a dynamic equation that can be written in the form

$$\boldsymbol{\tau} = \mathbf{M}(\theta)\ddot{\theta} + \mathbf{V}(\theta, \dot{\theta})\dot{\theta} + \mathbf{G}(\theta) \tag{2.41}$$

where $\mathbf{M}(\theta)$ is n x n mass matrix of the manipulator, $\mathbf{V}(\theta,\dot{\theta})$ is a n x 1 vector of centrifugal and Coriolis terms, and $\mathbf{G}(\theta)$ is an n x 1 vector of gravity terms. We use the term state-space equation because the term $V(\theta,\theta')$ has both position and velocity dependence. Each element of $\mathbf{M}(\theta)$ and $\mathbf{G}(\theta)$ is a complex function that depends on θ, the position of all the joints of the manipulator. Each element of $\mathbf{V}(\theta,\dot{\theta})$ is a complex function of both θ and $\dot{\theta}$. We may separate the various types of terms appearing in the dynamic equations and form the mass matrix of the manipulator, the centrifugal and Coriolis vector, and the gravity vector [1].

2.6.1 Virtual Work Dynamics

In this section, we will perform the inverse dynamic modeling of the parallel manipulator based upon the principle of virtual work. The inverse dynamics problem is to find the actuator torques and/or forces required to generate a desired trajectory of the manipulator [9].

Without losing generality of model, we can simplify the dynamic problem by the following hypotheses:

The connecting rods of lower links can be built with light materials such as the aluminum alloy, so

- The lower links rotational inertias are neglected.
- the mass of each lower links, is divided evenly and concentrated at
- The two endpoints of the parallelogram.

Also it is supposed that:

- The friction forces in joints are neglected.
- No external forces suffered.

We consider that $\tau=[\tau_1,\tau_2,\tau_3]$ and $\delta\theta=[\delta\theta_1,\delta\theta_2,\delta\theta_3]$ are the vector of actuator torques and vector of corresponding virtual angular displacements. Furthermore, $\delta p=[\delta x,\delta y,\delta z]$ represents the virtual linear displacements vector of the mobile platform. We can derive the following equations by applying the virtual work principle.

$$\tau^T \delta\theta + M_{Ga}^T \delta\theta + F_{Gp}^T \delta p - M_a^T \delta\theta - F_p^T \delta p = 0 \tag{2.42}$$

Where,

$$M_{Ga} = (\frac{1}{2} m_a + m_b).\mathrm{g}\, l_a.\mathrm{I}.[\cos(\theta_1) \quad \cos(\theta_2) \quad \cos(\theta_3)]^T \tag{2.43}$$

is the upper links gravity torques vector ma and $\mathbf{m_b}$ are mass of upper link and each connecting rod of lower link, respectively. Here ***g*** denotes the gravity acceleration, and ***I*** represent the 3x3 identity matrix.

$$F_{Gp} = [0 \quad 0 \quad -(m_{tcp} + 3m_b)\,\mathrm{g})]^T \tag{2.44}$$

Denotes the mobile platform gravity force vector, and $\boldsymbol{m_{tcp}}$ is mass of the mobile platform.

$$M_a = I_a \ddot{\theta} = I_a[\ddot{\theta}_1 \quad \ddot{\theta}_2 \quad \ddot{\theta}_3]^T \tag{2.45}$$

Where,

$$I_a = (\frac{1}{3} m_a l_a{}^2 + m_b l_a{}^2).\mathrm{I}$$

Represents the upper links inertia torques vector and denotes the upper links inertial matrix with respect to the fixed frame ***O{x, y, z},*** and,

$$F_P = M_P \ddot{P} = (m_{tcp} + 3m_b).\mathrm{I}.[\ddot{\mathrm{x}} \quad \ddot{\mathrm{y}} \quad \ddot{\mathrm{z}}]^T \tag{2.46}$$

Denote the mobile platform inertial forces vector. Eq.2.39 in section 2.4 can be rewritten to,

$$\dot{\mathbf{P}} = \mathbf{J}\dot{\boldsymbol{\theta}} \tag{2.47}$$

Consequently,

$$\delta P = J\delta\theta \tag{2.48}$$

Substituting Eq. 2.48 into Eq. 2.42 results,

$$(\tau^T + M_{Ga}^T + F_{Gp}^T J - M_a^T - F_p^T J)\delta\theta = 0 \tag{2.49}$$

Eq. 2.49 holds for any virtual displacements $\delta\theta$, so we have

$$\tau = M_a + J^T F_p - M_{Ga} - J^T F_{Gp} \tag{2.50}$$

Substitute Eqs.2.44 and 2.45 into Eq. 2.50, allows the generation of

$$\tau = I_a\ddot{\theta} + J^T M_p\ddot{P} - M_{Ga} - J^T F_{Gp} \tag{2.51}$$

Differentiating Eq. 2.47 with respect to time, yields

$$\ddot{P} = J\ddot{\theta} + \dot{J}\dot{\theta} \tag{2.52}$$

Substituting Eq. 2.52 into Eq. 2.51, we can derive that

$$\tau = M(\theta)\ddot{\theta} + V(\theta,\dot{\theta}) + G(\theta)$$

The previous equation described in Eq. 2.41 represents the dynamic model of parallel manipulator in joint space. Here, $\theta \in R^3$ is the controlled variables, and

$$M(\theta) = I_a + J^T M_p J \tag{2.53}$$

Denotes a symmetric positive definite inertial matrix, that $M(\theta) \in R^{3x3}$.

$$V(\theta,\dot{\theta}) = J^T M_p \dot{J} \tag{2.54}$$

Where $V(\theta,\dot{\theta}) \in R^{3x3}$ is the centrifugal and Coriolis forces matrix, and

$$G(\theta) = -\mathrm{M}_{Ga} - J^T F_{Gp} \tag{2.55}$$

Represents the vector of gravity forces, and $G(\theta) \in R^3$.

2.6.2 Non-Rigid Body Effects

It is important to realize that the dynamic equations we have derived do not encompass all the effects acting on a manipulator. They include only those forces which arise from rigid body mechanics. The most important source of forces that are not included is friction. All mechanisms are, of course, affected by frictional forces. In present-day manipulators, in which significant gearing is typical, the forces due to friction can actually be quite large - perhaps equaling 25% of the torque required to move the manipulator in typical situations. In order to make dynamic equations reflect the reality of the physical device, it is important to model (at least approximately) these forces of friction. A very simple model for friction is viscous friction, in which the torque due to friction is proportional to the velocity of joint motion. Thus, we have

$$\tau_{friction} = v\dot{\theta} \tag{2.56}$$

where ***v*** is a viscous-friction constant. Another possible simple model for friction, Coulomb friction, is sometimes used. Coulomb friction is constant except for a sign dependence on the joint velocity and is given by

$$\tau_{friction} = c\,\mathrm{sgn}(\dot{\theta}) \tag{2.57}$$

where ***c*** is a Coulomb-friction constant. The value of ***c*** is often taken at one value when $\dot{\theta} = 0$ the static coefficient, but at a lower value, the dynamic

coefficient, when $\dot{\theta} \neq 0$, whether a joint of a particular manipulator exhibits viscous or Coulomb friction is a complicated issue of lubrication and other effects. A reasonable model is to include both, because both effects are likely:

$$\tau_{friction} = v\dot{\theta} + c\,\mathrm{sgn}(\dot{\theta}) \tag{2.58}$$

It turns out that, in many manipulator joints, friction also displays a dependence on the joint position. A major cause of this effect might be gears that are not perfectly round-their eccentricity would cause friction to change according to joint position. So a fairly complex friction model would have the form

$$\tau_{friction} = f(\theta, \dot{\theta}) \tag{2.59}$$

These friction models are then added to the other dynamic terms derived from the rigid-body model, yielding the more complete model

$$\tau = M(\theta)\ddot{\theta} + V(\theta, \dot{\theta})\dot{\theta} + G(\theta) + F(\theta, \dot{\theta}) \tag{2.60}$$

There are also other effects, which are neglected in this model. For example, the assumption of rigid body links means that we have failed to include bending effects (which give rise to resonances) in our equations of motion. However, these effects are extremely difficult to model and are beyond the scope of this thesis [1].

2.7 Actuator Dynamics

The leg system is basically composed of dc motor, precision revolute bearing and coupling elements. Dc motor model is given below

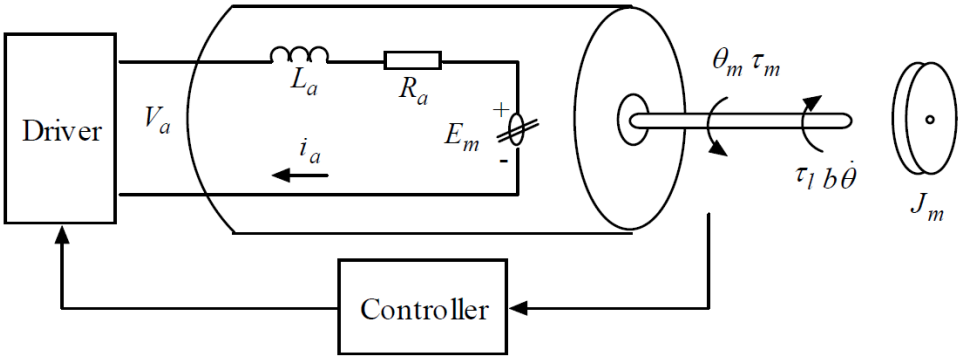


Figure 2.9: DC motor model

The symbols represent the following variables here θ_m is the motor position (radian), τ_m is the produced torque by the motor (Nm), τ_1 is the load torque, v_a is the armature voltage (V), L_a is the armature inductance (H), R_a is the armature resistance (Ω), E_m is the reverse EMF (V), I_a is the armature current (A), K_b is the reverse EMF constant, K_m is the torque constant [10].

$$
\begin{aligned}
L_a \frac{di_a}{dt} + R_a i_a &= V_a - E_m \\
E_m &= K_b \frac{d\theta_m}{dt} \\
\tau_m &= K_m i_a \\
\tau_m - \tau_1 &= j_m \frac{d^2\theta_m}{dt^2}
\end{aligned}
\tag{2.53}
$$

On the assumption of a rigid transmission and with no backlash the relationship between the input forces (velocities) and the output forces (velocities) are purely proportional. This gives,

$$\theta_m = K_r \theta_l \tag{2.54}$$

Where, constant K_r is a parameter which describes the gear reduction ratio. τ_l is the load torque at the robot axis and τ_m is the torque produced by the actuator at the shaft axis. In view of Eq. 2.54 one can write

$$\tau_m = \frac{\tau_l}{K_r} \tag{2.55}$$

To simulate the motion of a manipulator, we must make use of a model of the dynamics such as the one we have just developed. Given the dynamics written in closed form as in (2.52), simulation requires solving the dynamic equation for acceleration:

$$\ddot{\theta} = M^{-1}(\theta)[\tau - V(\theta,\dot{\theta}) - G(\theta) - F(\theta,\dot{\theta})] \tag{2.56}$$

We can then apply any of several known numerical integration techniques to integrate the acceleration to compute future positions and velocities. Given initial conditions on the motion of the manipulator, usually in the form

$$\begin{aligned} \theta(0) &= \theta_\circ \\ \dot{\theta}(0) &= 0 \end{aligned} \tag{2.57}$$

Chapter 3

Controller Design

3.1 Controller Techniques

Using inverse kinematics, we can calculate the joint kinematics for a desired geometric path of the end-effector of a robot. Substitution of the joint kinematics in equations of motion provides the actuator commands. Applying the commands will move the end-effector of the robot on the desired path ideally. However, because of perturbations and non-modeled phenomena, the robot will not follow the desired path. The techniques that minimize or remove the difference are called the control techniques [11].

3.2 Open and Closed-Loop Control

A robot is a mechanism with an actuator at each joint i to apply a force or torque to derive the link (i). The robot is instrumented with position, velocity, and possibly acceleration sensors to measure the joint variables' kinematics. The measured values are usually kinematics information of the frame B_i , attached to the link i. Relative to the frame B_{i-1} or B_0. To cause each joint of the robot to follow a desired motion, we must provide the required torque command. Assume that the desired path of joint variables, $q_d = q(t)$ are given as functions of time. Then, the required torques that causes the robot to follow the desired motion is calculated by the equations of motion and is equal to

$$\mathbf{Q}_c = \mathbf{D}(q_d)\ddot{q}_d + \mathbf{H}(q_d, \dot{q}_d) + \mathbf{G}(\dot{q}_d) \tag{3.1}$$

Where the subscripts *d and c* stands for *desired* and *controlled*, respectively. in an ideal world, the variables can be measured exactly and the robot can

perfectly work based on the equations of motion (3.1). Then, the actuators' control command Q_c can cause the desired path q_d to happen. This is an open-loop control algorithm, that the control commands are calculated based on a known desired path and the equations of motion. Then, the control commands are fed to the system to generate the desired path. Therefore, in an open-loop control algorithm, we expect the robot to follow the designed path, however, there is no mechanism to compensate any possible error.

Now assume that we are watching the robot during its motion by measuring the joints' kinematics. At any instant there can be a difference between the actual joint variables and the desired values. The difference is called error and is measured by

$$\begin{aligned} e &= q - q_d \\ \dot{e} &= \dot{q} - \dot{q}_d \end{aligned} \tag{3.2}$$

Let's define a control law and calculate a new control command vector by

$$\mathbf{Q} = \mathbf{Q}_c + K_d\dot{\mathbf{e}} + K_p\mathbf{e} \tag{3.3}$$

where k_P and kd are constant control gains. The control law compares the actual joint variables $(q, \dot{q})$ with the desired values $(q_d, \dot{q}_d)$, and generates a command proportionally. Applying the new control command changes the dynamic equations of the robot to produce the actual joint variables q.

$$\mathbf{Q}_c + K_d\dot{\mathbf{e}} + K_p\mathbf{e} = \mathbf{D}(q_d)\ddot{q}_d + \mathbf{H}(q_d, \dot{q}_d) + \mathbf{G}(q_d) \tag{3.4}$$

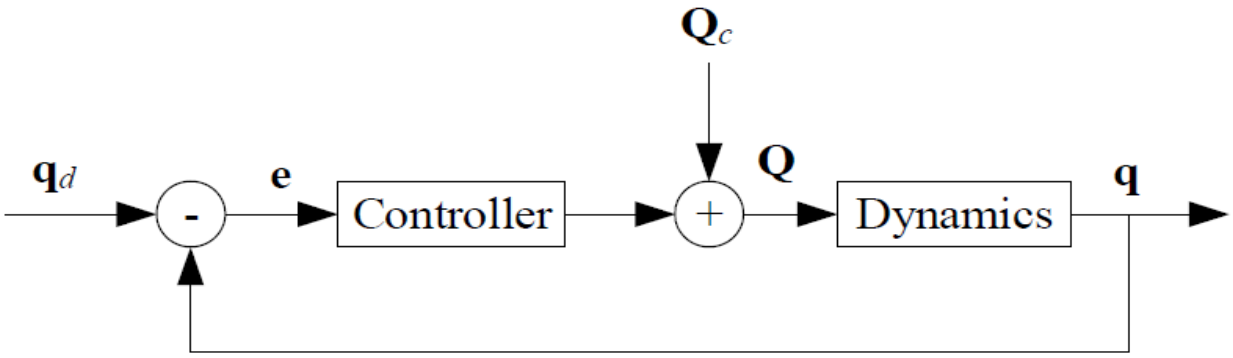


Figure 3.1 Illustration of feedback control algorithm

Fig. 3.1 illustrates the idea of this control method in a block diagram. This is *a closed-loop control algorithm*, in which the control commands are calculated based on the difference between actual and desired variables. Reading the actual variables and comparing with the desired values is called *feedback*, and because of that, the closed-loop control algorithm is also called *a feedback control algorithm.*

The controller provides a signal proportional to the error and its time rate. This signal is added to the predicted command *Qc* to compensate the error.

The principle of feedback control can be expressed as: *Increase the control command when the actual variable is smaller than the desired value and decrease the control command when the actual variable is larger than the desired value.*

3.2.1 Robot Control Algorithms

Robots are nonlinear dynamical systems, and there is no general method for designing a nonlinear controller to be suitable for every robot in every mission. However, there are a variety of alternative and complementary methods, each best applicable to particular class of robots in a particular mission. The most important control methods are as follows:

- *Feedback Linearization or Computed Torque Control Technique.*

In feedback linearization technique, we define a control law to obtain a linear differential equation for error command, and then use the linear control

design techniques. The feedback linearization technique can be applied to robots successfully; however, it does not guarantee robustness according to parameter uncertainty or disturbances. This technique is a model-based control method, because the control law is designed based on a nominal model of the robot.

- *Linear Control Technique*

The simplest technique for controlling robots is to design a linear controller based on the linearization of the equations of motion about an operating point. The linearization technique locally determines the stability of the robot. Proportional, integral, and derivative, or any combination of them, are the most practical linear control techniques.

- *Adaptive Control Technique*

Adaptive control is a technique for controlling uncertain or time-varying robots. Adaptive control technique is more effective for low DOF robots.

- *Robust and Adaptive Control Technique.*

In the robust control method, the controller is designed based on the nominal model plus some uncertainty. Uncertainty can be in any parameter, such as the load carrying by the end-effector. For example, we develop a control technique to be effective for loads in a range of 1 - 10 kg.

- *Gain-Scheduling Control Technique.*

Gain-scheduling is a technique that tries to apply the linear control techniques to the nonlinear dynamics of robots. In gain-scheduling, we select a number of control points to cover the range of robot operation. Then at each control point, we make a linear time-varying approximation to the robot dynamics and design a linear controller. The parameters of the controller are then interpolated or scheduled between control points.

3.3 Computed Torque Control

Dynamics of a robot can be expressed in the form

$$\mathbf{Q} = \mathbf{D}(q)\ddot{q} + \mathbf{H}(q,\dot{q}) + \mathbf{G}(q) + \tau_d \tag{3.5}$$

Where q is the vector of joint variables, and Q is the torques applied at joints, And is τ_d a disturbance .Assume a desired path in joint space is given by a twice differentiable function $q = q_d(t) \in C^2$. Hence, the desired time history of joints' position, velocity, and acceleration are known [12].

We can re-write Eq. 3.5 to:

$$\mathbf{Q} = \mathbf{D}(q)\ddot{q} + \mathbf{N}(q,\dot{q}) + \tau_d \tag{3.6}$$

If this equation includes motor actuator dynamics, then Q is an input voltage.
Define an output or tracking error as:

$$e = q_d - q \tag{3.7}$$

And so,

$$\begin{aligned} \dot{e} &= \dot{q}_d - \dot{q} \\ \ddot{e} &= \ddot{q}_d - \ddot{q} \end{aligned} \tag{3.8}$$

Solving now for $\ddot{q}$ in Eq.3.6 and substituting into Eq. 3.7 yields,

$$\ddot{e} = \ddot{q}_d - D^{-1}(\mathrm{N} + \tau_d - \tau) \tag{3.9}$$

And the disturbance function

$$w = D^{-1}\tau_d \tag{3.10}$$

we may define a state *x(t)* by

$$x = \begin{bmatrix} e \\ \dot{e} \end{bmatrix} \tag{3.11}$$

Write the tracking error dynamics as,

$$\begin{bmatrix} \dot{e} \\ \ddot{e} \end{bmatrix} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} \begin{bmatrix} e \\ \dot{e} \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} u + \begin{bmatrix} 0 \\ I \end{bmatrix} w \tag{3.12}$$

It is driven by the control input *u(t)* and the disturbance *w(t).* Note that this derivation is a special case of the general feedback linearization procedure. The feedback linearizing transformation may be inverted to yield

$$\tau = D(\ddot{q}_d - \mathrm{u}) + \mathrm{N} \tag{3.13}$$

We call this the computed-torque control law. Substituting Eq. 3.13 into Eq.3.5 yields

$$\mathbf{D}(q)\ddot{q} + \mathbf{N}(q,\dot{q}) + \tau_d = \mathbf{D}(\ddot{q}_d - \mathrm{u}) + \mathrm{N} \tag{3.14}$$

or

$$\ddot{e} = u + D^{-1}\tau_d \tag{3.15}$$

3.4 PID Outer-Loop Designs

One way to select the auxiliary control signal u(t) is as the proportional-plus derivative (PD) feedback,

$$u = -k_v \dot{e} - k_b e - k_i \varepsilon \tag{3.16}$$

Then the overall robot arm input becomes

$$\tau = \mathbf{D}(q)(\ddot{q} + k_v\dot{e} + k_b e + k_i\varepsilon) + \mathbf{N}(q,\dot{q}) \tag{3.17}$$

The closed loop error dynamics

$$\ddot{e} + k_v\dot{e} + k_b e + k_i\varepsilon = w \tag{3.18}$$

The next diagram, represent the PD – Computed Torque controller

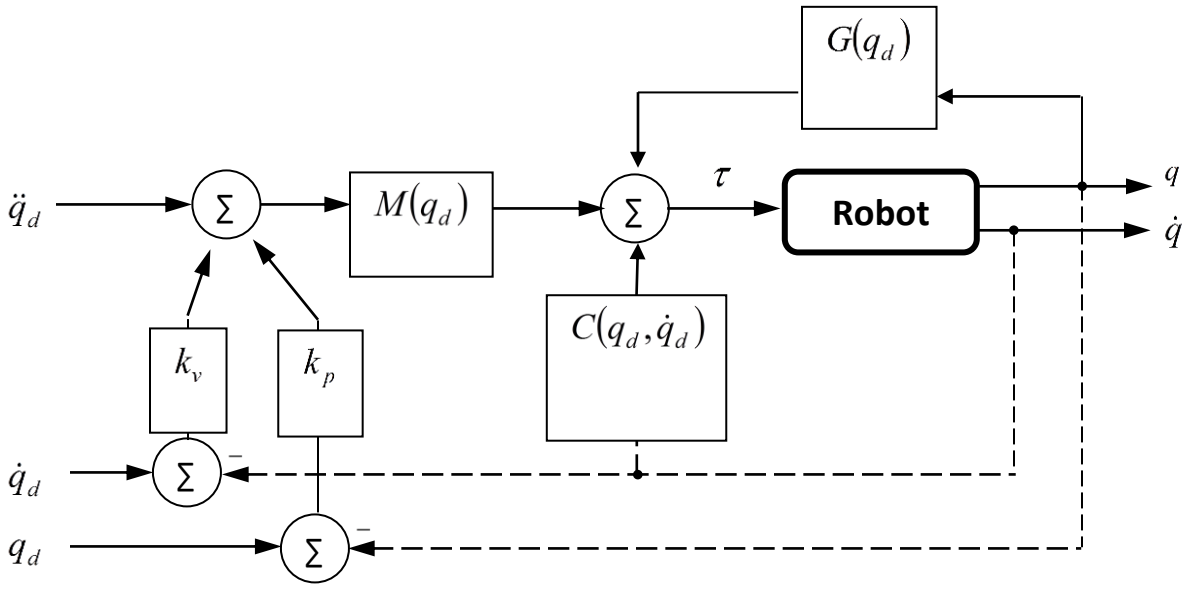


Figure 3.2. Block diagram of computed torque control.

3.5 PD-Plus-Gravity Controller

A useful controller in the computed-torque family is the PD-plus-gravity controller that results when D=I, N=G(q)-q_d, with $G(q)$ the gravity term of the manipulator dynamics. Then, selecting PD feedback for u(t) yields,

$$\tau_c = k_v\dot{e} + k_b e + G(q) \tag{3.19}$$

When the arm is at rest, the only nonzero terms in the dynamics Eq.3.5 are the gravity G (q), the disturbance τ_d , and possibly the control torque τ .
The PD-gravity controller τ_c , includes G (q), so that we should expect good Performance for set-point tracking, that is, when a constant q_d is given so that q_d= 0.

3.6 Optimal PD Controller Design

The goal of implementing any type of controller is to observe the output response it would generate based on the inputted conditions. In order to achieve this, it is necessary to solve for the control input (u) of the system. Each controller has a different method pertaining to how this equation is obtained, but the initial steps to reach this point are all similar.

The end effector of the three degrees of freedom parallel robot will follow a predefined trajectory; hence for tracking control it is appropriate to set the error and change in error as:

$$x_1 = \theta_{1d} - \theta_1,$$
$$x_2 = \theta_{2d} - \theta_2,$$
$$x_3 = \theta_{3d} - \theta_3,$$
$$x_4 = \dot{\theta}_{1d} - \dot{\theta}_1,$$
$$x_5 = \dot{\theta}_{2d} - \dot{\theta}_2,$$
$$x_6 = \dot{\theta}_{3d} - \dot{\theta}_3,$$

Where: θ_{1d} , θ_{2d} and θ_{3d} are the desired angles; θ_1, θ_2 and θ_3 are the actual angles; $\dot{\theta}_{1d}$, $\dot{\theta}_{2d}$ and $\dot{\theta}_{3d}$ are the desired angular velocities, $\dot{\theta}_1, \dot{\theta}_2$ and $\dot{\theta}_3$ are the actual angular velocities; $\ddot{\theta}_{1d}, \ddot{\theta}_{2d}$ and $\ddot{\theta}_{3d}$ the desired angular accelerations. The following system is in lower triangular form, which can be produced by differentiating x_1, x_2, x_3, x_4, x_5 *and* x_6.

$$\dot{x}_1 = x_4$$

$$\dot{x}_2 = x_5$$
$$\dot{x}_3 = x_6 \tag{3.20}$$

$$\begin{bmatrix} \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{bmatrix} = \begin{bmatrix} \ddot{\theta}_{1d} \\ \ddot{\theta}_{2d} \\ \ddot{\theta}_{3d} \end{bmatrix} + D^{-1}(\theta)\left(-u + C(\theta,\dot{\theta}) \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{bmatrix} + G(\theta) \right) \tag{3.21}$$

One aspect that constantly appears when implementing the appropriate controller is the feed forward term u_d. This term represents the desired control input required in the overall system operation. In theory, the actual and desired control input should be identical, but due to system disturbances and the force of gravity, this is known not to be the case. By adding u_d into the specified controller, improved control performance can be achieved. It is defined as:

$$u_d = D(\theta_d)\ddot{\theta}_d + C(\theta_d, \dot{\theta}_d)\dot{\theta}_d + G(\theta_d) \tag{3.22}$$

That is, a Lyapunov function is necessary in order to achieve the desired results. Let this function candidate be:

$$V = 0.5\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} kp_1 & 0 & 0 \\ 0 & kp_2 & 0 \\ 0 & 0 & kp_3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + 0.5\begin{bmatrix} x_4 & x_5 & x_6 \end{bmatrix} D(\theta) \begin{bmatrix} x_4 \\ x_5 \\ x_6 \end{bmatrix} \tag{3.23}$$

It should be noted that K_{P1}, K_{P2} and K_{P3} represent the proportional gains of motors one , two and three respectively. The derivative of V with respect to time becomes:

$$\dot{V} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} kp_1 & 0 & 0 \\ 0 & kp_2 & 0 \\ 0 & 0 & kp_3 \end{bmatrix} \begin{bmatrix} x_4 \\ x_5 \\ x_6 \end{bmatrix} + 0.5 \begin{bmatrix} x_4 & x_5 & x_6 \end{bmatrix} \dot{D}(\theta) \begin{bmatrix} x_4 \\ x_5 \\ x_6 \end{bmatrix}$$
$$+ \begin{bmatrix} x_4 & x_5 & x_6 \end{bmatrix} D(\theta) \begin{bmatrix} \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{bmatrix} \tag{3.24}$$

As previously stated, $D(\theta) - 2C(\theta, \dot{\theta})$ is skew symmetric; hence:

$$0.5 \begin{bmatrix} x_4 & x_5 & x_6 \end{bmatrix} \dot{D}(\theta) \begin{bmatrix} x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} x_4 & x_5 & x_6 \end{bmatrix} C(\theta, \dot{\theta}) \begin{bmatrix} x_4 \\ x_5 \\ x_6 \end{bmatrix} \tag{3.25}$$

Therefore, by substituting equations (3.3) and (3.25) into equation (3.24), it is possible to achieve:

$$\dot{V} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} kp_1 & 0 & 0 \\ 0 & kp_2 & 0 \\ 0 & 0 & kp_3 \end{bmatrix} \begin{bmatrix} x_4 \\ x_5 \\ x_6 \end{bmatrix} + \begin{bmatrix} x_4 & x_5 & x_6 \end{bmatrix} \left(D(\theta) \begin{bmatrix} \ddot{\theta}_{1d} \\ \ddot{\theta}_{2d} \\ \ddot{\theta}_{3d} \end{bmatrix} + C(\theta, \dot{\theta})) \begin{bmatrix} \dot{\theta}_{1d} \\ \dot{\theta}_{2d} \\ \dot{\theta}_{3d} \end{bmatrix} + G(\theta) - \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \right) \tag{3.26}$$

With all the appropriate data defined, it is now possible to determine the equation for the controller. The control effort must satisfy the condition of convergence and it must ensure that the output response is stable. The following controller was chosen to accomplish these requirements:

$$u=\begin{bmatrix}u_1\\u_2\\u_3\end{bmatrix}=\begin{bmatrix}kp_1&0&0\\0&kp_2&0\\0&0&kp_3\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}+\begin{bmatrix}kd_1&0&0\\0&kd_2&0\\0&0&kd_3\end{bmatrix}\begin{bmatrix}x_4&x_5&x_6\end{bmatrix}+\left(D(\theta)\begin{bmatrix}\ddot{\theta}_{1d}\\\ddot{\theta}_{2d}\\\ddot{\theta}_{3d}\end{bmatrix}+C(\theta,\dot{\theta})\begin{bmatrix}\dot{\theta}_{1d}\\\dot{\theta}_{2d}\\\dot{\theta}_{3d}\end{bmatrix}+G(\theta)\right) \tag{3.27}$$

It should be noted that K_{d1} , K_{d2} and K_{d3} represent the derivative gains of motors one, two and three, respectively.

3.7 Model Predictive Control (MPC)

Model Predictive Control (MPC) is an optimal control strategy based on numerical optimization. Future control inputs and future plant responses are predicted using a system model and optimized at regular intervals with respect to a performance index. From its origins as a computational technique for improving control performance in applications within the process and petrochemical industries, predictive control has become arguably the most widespread advanced control methodology currently in use in industry. MPC now has a sound theoretical basis and its stability, optimality, and robustness properties are well understood.

The basic structure of MPC to implement is shown in Fig.3.3. A model is used to predict the future plant outputs, based on past and current values and on proposed future control actions.

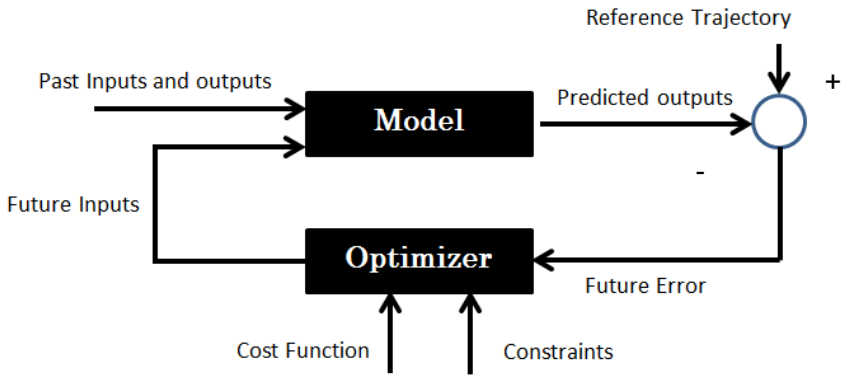


Figure 3.3: Basic Structure of MPC Controller.

These actions are calculated by optimizer taking into account the cost function as well as con- strains. The optimizer is another fundamental part of

the strategy as it provides the control action. The goal is to apply the linear model predictive control to the input-output linearized system to account for the constraints. Since the linear model predictive is more naturally formulated in discrete time, the linear subsystem is discretized with a sampling period T to yield,

$$\begin{aligned} \xi(\mathrm{k}+1) &= \mathrm{A}_d\, \xi(\mathrm{k}) + \mathrm{B}_d\, V(\mathrm{k}) \\ \mathrm{y}(\mathrm{k}) &= \mathrm{H}_d\, \xi(\mathrm{k}) \end{aligned} \tag{3.28}$$

The model consists of:

- A model of the plant to be controlled, whose inputs are the manipulated variables, the measured disturbances, and the unmeasured disturbances
- A model generating the unmeasured disturbances

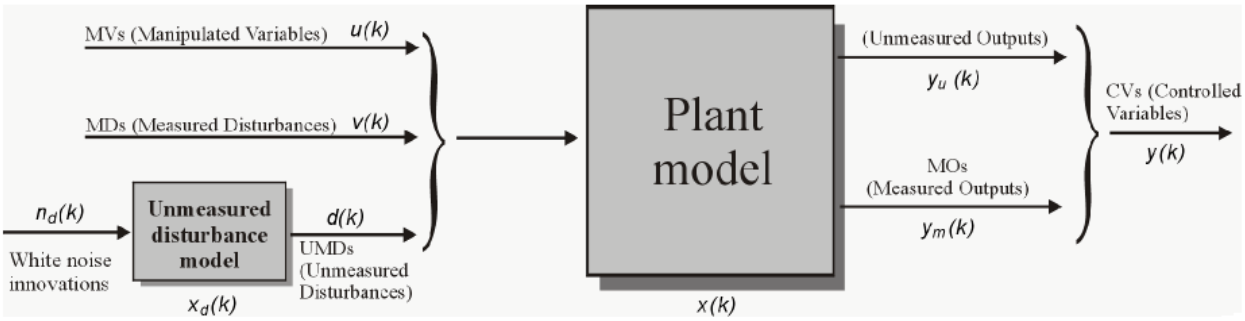


Figure 3.4: Model Used For Optimization

The model of the plant is a linear time-invariant system described by the equations

$$\begin{aligned} \mathrm{x}(\mathrm{k}+1) &= \mathrm{A}\mathrm{x}(\mathrm{k}) + \mathrm{B}_u\mathrm{u}(\mathrm{k}) + \mathrm{B}_v\, v(\mathrm{k}) + \mathrm{B}_d\, d(\mathrm{k}) \\ \mathrm{y}(\mathrm{k}) &= \mathrm{C}_x\, x(\mathrm{k}) + \mathrm{D}_v\, v(\mathrm{k}) + \mathrm{D}_d\, d(\mathrm{k}) \end{aligned} \tag{3.29}$$

The unmeasured disturbance d(k) is modeled as the output of the linear time invariant system:

$$\begin{aligned} x_d(k+1) &= \bar{A}x_d(k) + \bar{B}n_d(k) \\ d(k) &= \bar{C}x_d(k) + \bar{D}n_d(k) \end{aligned} \tag{3.30}$$

The system described by the above equations is driven by the random Gaussian noise $n_d(k)$, having zero mean and unit covariance matrix.

Chapter 4

Path Planning

4.1 Introduction

In this chapter, we concern ourselves with methods of computing a trajectory that describes the desired motion of a manipulator in multidimensional space. Here, trajectory refers to a time history of position, velocity, and acceleration for each degree of freedom.

This problem includes the human-interface problem of how we wish to specify a trajectory or path through space. In order to make the description of manipulator motion easy for a human user of a robot system, the user should not be required to write down complicated functions of space and time to specify the task. Rather, we must allow the capability of specifying trajectories with simple descriptions of the desired motion, and let the system figure out the details. For example, the user might want to be able to specify nothing more than the desired goal position and orientation of the end-effector and leave it to the system to decide on the exact shape of the path to get there, the duration, the velocity profile, and other details.

We also are concerned with how trajectories are represented in the computer after they have been planned. Finally, there is the problem of actually computing the trajectory from the internal representation—or generating the trajectory.

Generation occurs at run time; in the most general case, position, velocity, and acceleration are computed. These trajectories are computed on digital computers, so the trajectory points are computed at a certain rate, called the path-update rate. In typical manipulator systems, this rate lies between 60 and 2000 Hz [1].

4.2 Cubic polynomials

Consider the problem of moving the tool from its initial position to a goal position in a certain amount of time. Inverse kinematics allows the set of joint angles that correspond to the goal position and orientation to be calculated

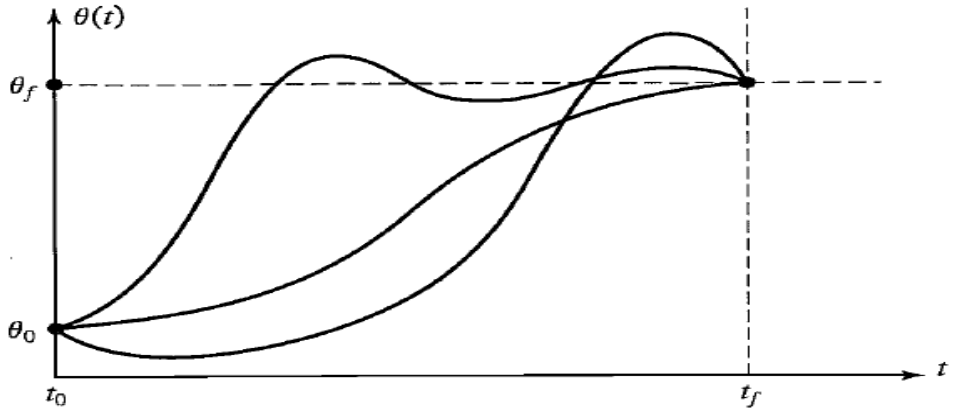


Figure 4.1: Several possible path shapes for a single joint.

The initial position of the manipulator is also known in the form of a set of joint angles. What is required is a function for each joint whose value at t0 is the initial position of the joint and whose value at t_f is the desired goal position of that joint. As shown in Fig. 4.1, there are many smooth functions, $\theta(t)$, that might be used to interpolate the joint value. Several possible path shapes for a single joint. In making a single smooth motion, at least four constraints on $\theta(t)$ are evident. Two constraints on the function's value come from the selection of initial and final values:

$$\theta(0) = \theta_{\circ}$$
$$\theta(t_f) = \theta_f \qquad (4.1)$$

An additional two constraints are that the function be continuous in velocity, which in this case means that the initial and final velocities are zero:

$$\dot{\theta}(0) = 0$$
$$\dot{\theta}(t_f) = 0 \tag{4.2}$$

These four constraints can be satisfied by a polynomial of at least third degree. These constraints uniquely specify a particular cubic. A cubic has the form

$$\theta(t) = a_\circ + a_1 t + a_2 t^2 + a_3 t^3 \tag{4.3}$$

So, the joint velocity and acceleration along this path are clearly

$$\dot{\theta}(t) = a_1 + 2a_2 t + 3a_3 t^2$$
$$\ddot{\theta}(t) = 2a_2 + 6a_3 t \tag{4.4}$$

Solving these equations for the a_i we obtain

$$a_0 = \theta_0$$
$$a_1 = 0$$
$$a_2 = \frac{3}{t_f^2}(\theta_f - \theta_0) \tag{4.5}$$
$$a_3 = \frac{3}{t_f^3}(\theta$$

Using Eq. 4.5, we can calculate the cubic polynomial that connects any initial joint angle position with any desired final position. This solution is for the case when the joint starts and finishes at zero velocity.

4.2.1 Cubic polynomials for a path with via points

So far, we have considered motions described by a desired duration and a final goal point. In general, we wish to allow paths to be specified that include intermediate via points. If the manipulator is to come to rest at each via point, then we can use the cubic solution of Eq. 4.3.

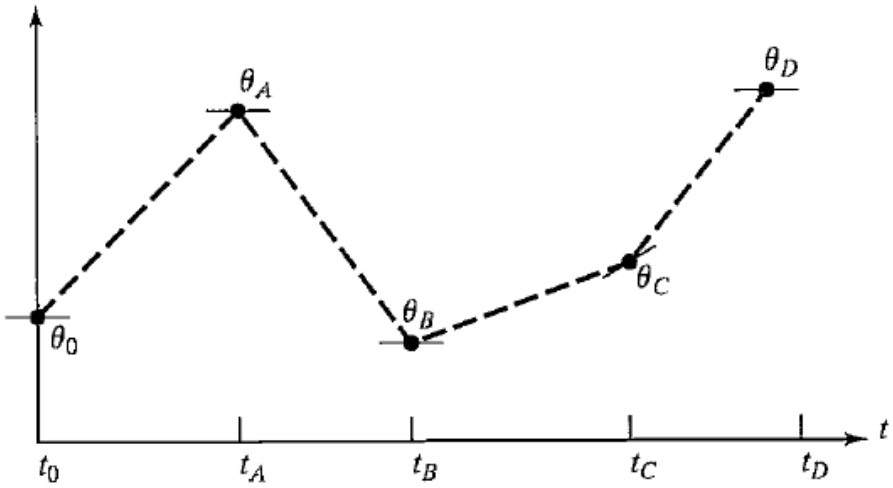


Figure 4.2: Via points with desired velocities at the points indicated by tangents.

If desired velocities of the joints at the via points are known, then we can construct cubic polynomials as before; now, however, the velocity constraints at each end are not zero, but rather, some known velocity

$$\dot{\theta}(0) = \dot{\theta}_\circ$$
$$\dot{\theta}(t_f) = \dot{\theta}_f \tag{4.6}$$

Solving these equations for a_i then we obtain

$$a_0 = \theta_0,$$
$$a_1 = \dot{\theta}_0,$$

$$a_2 = \frac{3}{t_f^2}(\theta_f - \theta_0) - \frac{2}{t_f}\dot{\theta}_0 - \frac{1}{t_f}\dot{\theta}_f,$$
$$a_3 = \frac{3}{t_f^3}(\theta_f - \theta_0) + \frac{1}{t_f^2}(\dot{\theta}_f - \dot{\theta}_0) \qquad (4.7)$$

Chapter 5

Simulations and Results

5.1 Introduction

Each controller discussed in this chapter will contain the simulation results for: the error between the desired and actual actuated joint angles, the location of the desired and actual end effector trajectory in Cartesian space along with their respective positional output error and the overall system torque required to achieve the actual results. A preliminary conclusion will then be drawn based on the pros and cons of each control technique.

5.2 Modeling Multi-body Systems

SimMechanics tool enables you to create libraries of components that can be reused in many different designs. You define bodies in terms of their mass, inertia, and connection points. To create complex shapes, you assemble sets of simple geometries, such as spheres, cylinders, and extrusions defined in MATLAB, and SimMechanics calculates the resulting mass and inertia automatically. The diagram that defines the body clearly indicates all connections to the body, making it easy to see your system's topology. Parameters such as length and mass can be calculated using MATLAB scripts and assigned using MATLAB variables.

You connect bodies using joints and constraints. These define the degrees of freedom permitted between the bodies in your system, which dictate how your system can move. You can define and connect actuators to these joints to enable your system to move. Actuating these joints with electrical, hydraulic, pneumatic, or other physical systems modeled using *Simscape* tool enables you to model your entire multi-domain physical system within the Simulink environment [13].

You can import a CAD assembly into SimMechanics using SimMechanics Link. The mass and inertia of each part in the assembly are imported as rigid

bodies in SimMechanics. Geometry from the CAD assembly is saved to geometry files and associated with the proper body in SimMechanics. The mate definitions in the CAD assembly are imported as joints in the SimMechanics model.

For SolidWorks, Pro/ENGINEER, and Autodesk Inventor models, you install a plug-in that lets you save the CAD assembly as an XML file that can be imported into SimMechanics. For other CAD systems, SimMechanics Link provides an API that you can connect to the API of your CAD system.

The SimMechanics Import XML Schema enables you to import models into SimMechanics from any CAD system or modeling environment that exports an XML file that follows this schema.

5.3 DELTA Robot CAD Modelling

As mentioned in the last section, delta robot designed in Autodesk Inventor. Table 5.1 summarizes the lengths, inertias and masses of the links and moving platform.

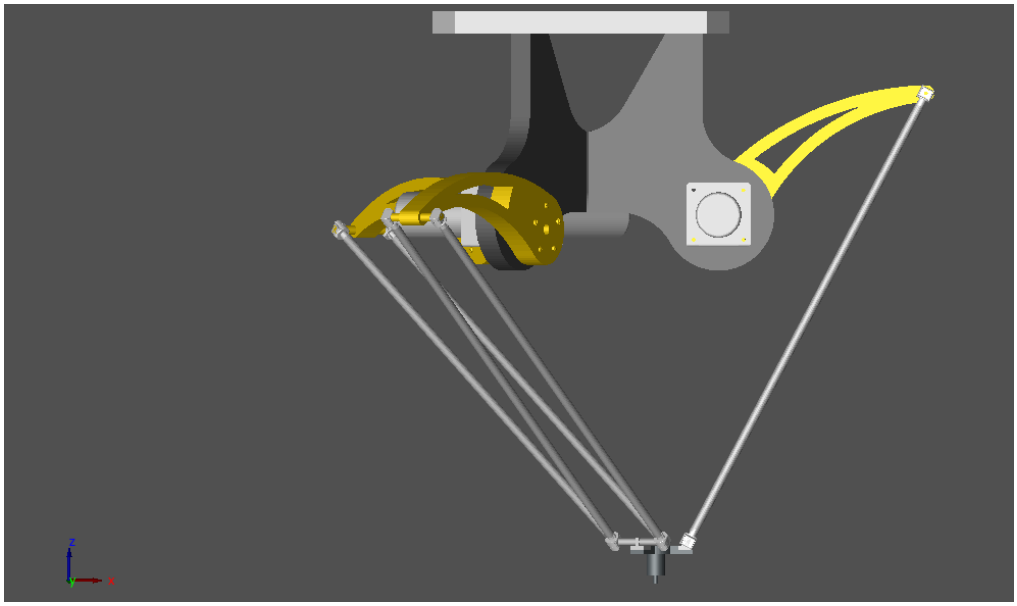


Figure5.1: CAD model of Delta Robot

With the help of Autodesk Inventor program, the following tables were utilized.Table 5.1, describes the Mechanical Properties of Robot parts. Masses and inertia depends on the material type, so, for more flexibility and robot speed, I chose little weight materials in building the Robot architecture.

Table 5.1: Delta Robot Mechanical Parts properties

Mechanical Part	Material	Mass(kg)
Upper Arm	Aluminum	1.58957
Forearm	Aluminum	0.0408236
Moving Platform	Echelon	0.110875

Table5.2. summarizes the mechanical part lengths and offset translation between frame {Oc} and frame {Or} as described in Fig. 5.2.

Table 5.2: Delta Robot 3-D dimensions

D-Robot Dimensions	Length(mm)	Description
L_f	185.032	Offset length from Frame $\{O_c\}$ to Joint J_i.
La	334.058	Length of the upper arm links.
L_b	684.083	Length of the Lower arm links.
L_e	44	Offset length from joint C_i to point D.
L_r	281.8	Offset length from Frame $\{O_c\}$ to frame $\{O_r\}$ in z- direction.
L_{tcp}	30	Offset length in z-direction from D point to Center of Gravity of the moving platform.

5.4 Dynamic Model of Delta Robot

After exporting the *.XML file generated from Autodesk Inventor Program, Matlab Simulink tool converted the exported *.XML file to Simulink model as illustrated in Fig.5.2.

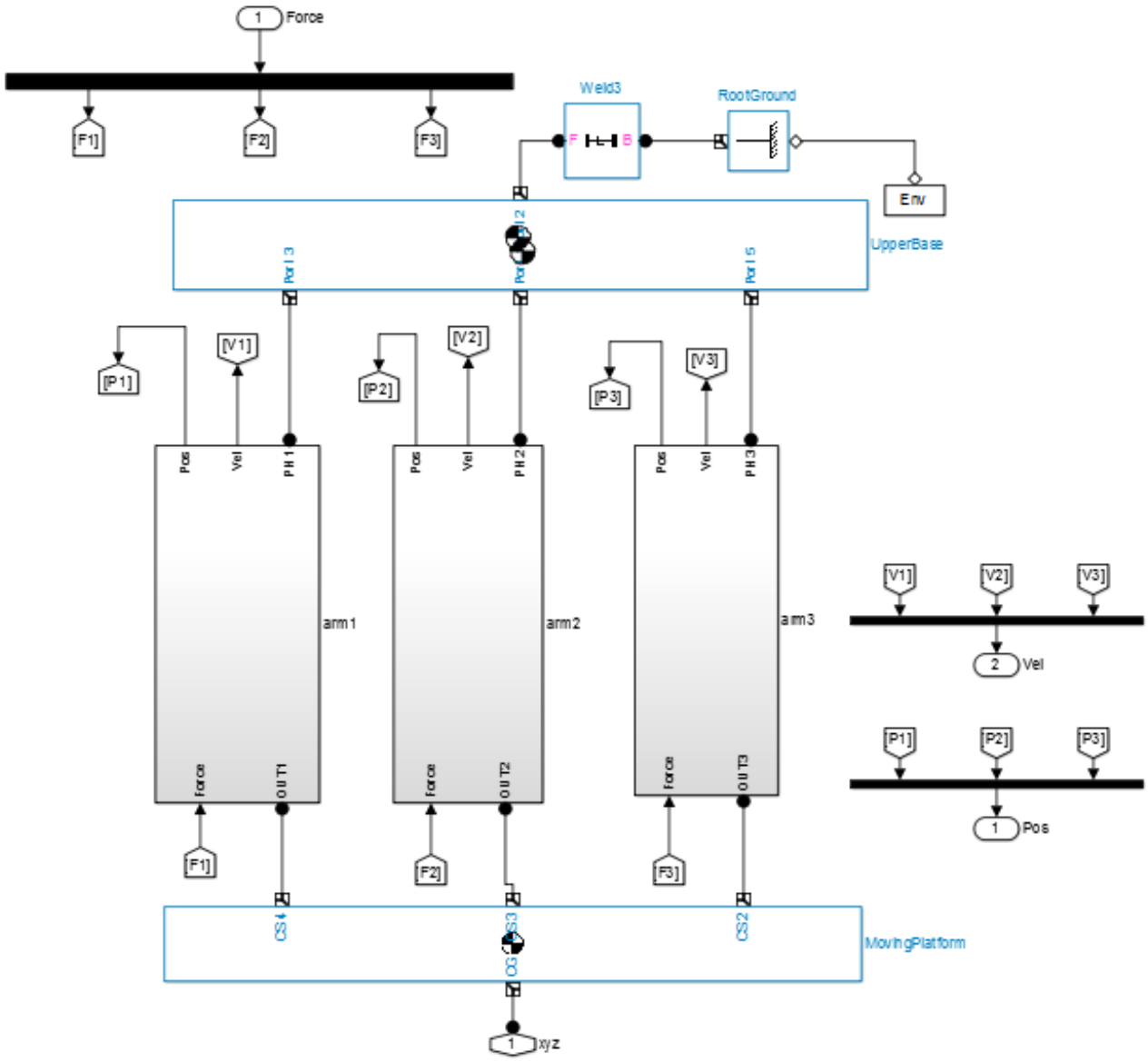


Figure5.2: Dynamic Model of Delta Robot

From the above Fig. 5.2 , the Simulink model represents the three arms of the robot, each arm is forced with torque at each private joint, speed and position measurements is calculated from each joint via joint sensors.

The next figure illustrates the Simulink block diagram of single arm which is composed of Upper Arm, Forearms, and its actuated joint and passive joint.

From the single arm dynamic model figure, Fig. 5.3 , the upper tip of the upper arm is actuated by Joint actuator which is placed in place of DC motor model, where the recent one do the same job.

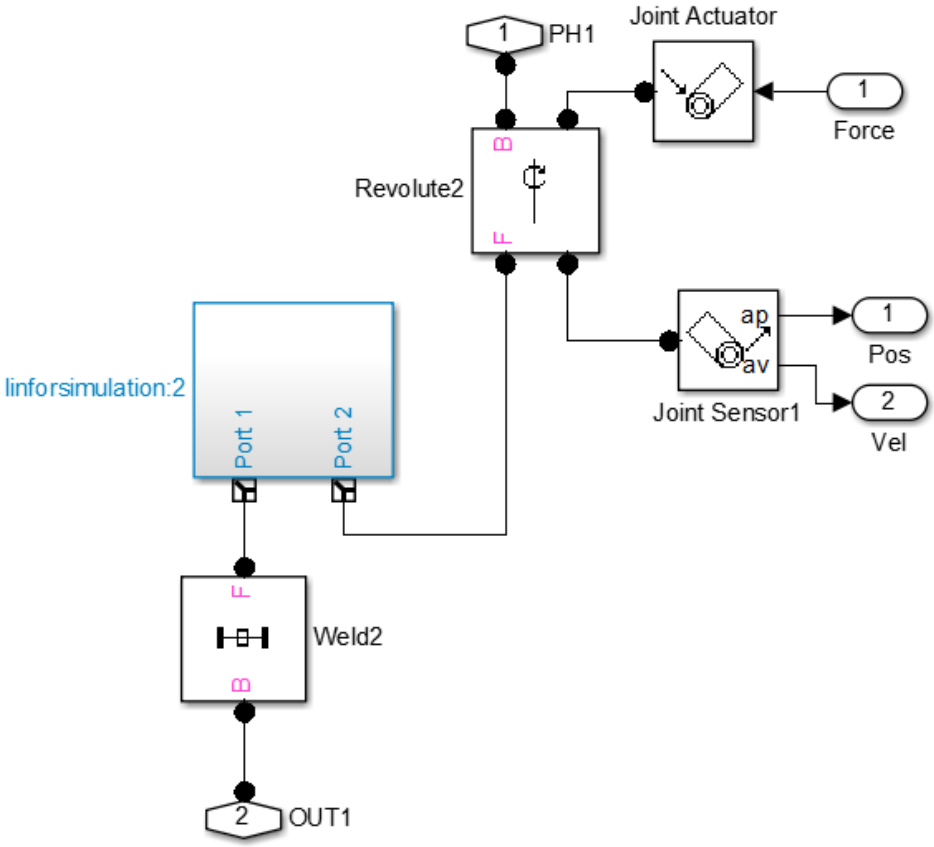


Figure5.3: Single Arm Dynamic Model

5.5 Model Linearization

Linear analysis tool embedded in Matlab Simulink, can deliver the linearized model of the nonlinear MIMO Delta Robot system dynamics. For linearization process, this block diagram shown in Fig.5.4 is connected. The output linearized model is arranged as

$$\dot{X} = Ax + Bu$$
$$Y = Cx + Du \tag{5.1}$$

Where A, B, C and D are the state space matrices.

And where, $[x_1, x_2]=[q_1, \dot{q}_1]$, $[x_3, x_4]=[q_2, \dot{q}_2]$, and $[x_5, x_6]=[q_3, \dot{q}_3]$. Where $q_i, \dot{q}_i$ are the joint phase and joint angular speed respectively

$$\dot{X} = \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{bmatrix} = [\mathrm{A}_{6X6}] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} + [\mathrm{B}_{6X3}] \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} \tag{5.2}$$

$$Y = [\mathrm{C}_{3X6}] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} + [D_{3X3}] \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} \tag{5.3}$$

5.5.1 Trimming and Linearizing Through Inverse Dynamics

Trimming is determination of the forces/torques necessary to produce the specified motion. These motion states constitute a trim or operating point. Trimming problems can have one solution, more than one, or none.

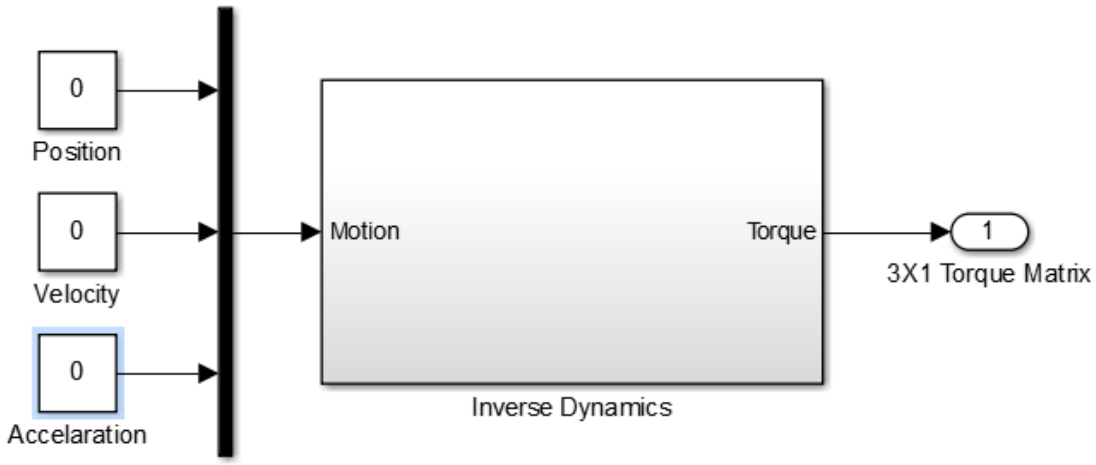


Figure 5.4: Inverse Dynamics DELTA Robot Model.

Each DELTA Robot leg outputs the computed leg force needed to maintain the motion specified by the motion actuation. After simulating the previous model Fig. 5.4, the steady states Torques keeping the platform sill is,

$$[\tau_1, \tau_2, \tau_3] = [2.128 \;\; 2.128 \;\; 2.128]$$

5.5.2 Linearizing at an Operating Point

Analysis tab in Matlab Simulink, linear analysis tool, introduces a simplified method for non-linear systems linearization. The state space result of

linearization at the operating point [2.128, 2.128, and 2.128] is stated in Eq.5.2 to Eq.5.5.

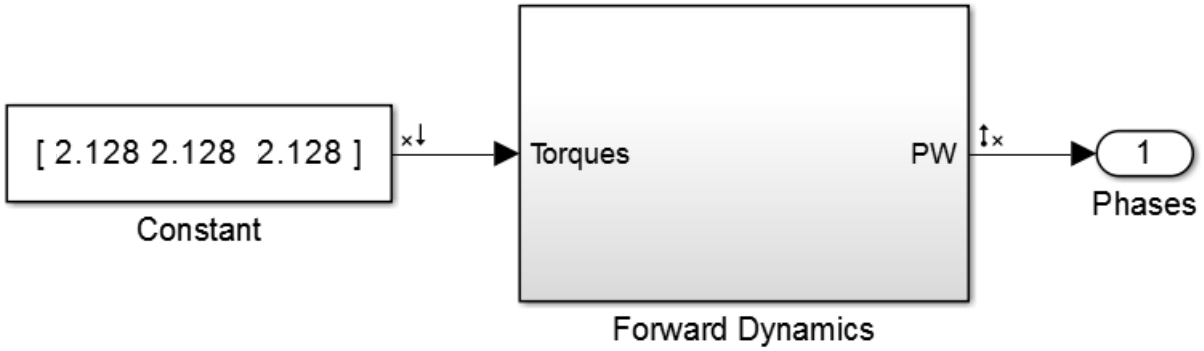


Figure 5.5: Simulink Model for Linearization

From Fig. 5.5, Linearization was according to position vector feedback. The output state spaces linear matrices A, B, C and D are,

$$A=\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0.6618 & 0 & -4.6386 & 0 & 10.5497 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0.1717 & 0 & -2.4943 & 0 & 0.8218 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 3.0516 & 0 & -4.9436 & 0 & 9.1532 & 0 \end{bmatrix} \quad (5.2)$$

$$B=\begin{bmatrix} 0 & 0 & 0 \\ -0.8109 & 15.5792 & -0.8109 \\ 0 & 0 & 0 \\ -3.6961 & 3.2042 & 6.9002 \\ 0 & 0 & 0 \\ -8.6789 & 4.5069 & -4.0151 \end{bmatrix} \tag{5.3}$$

$$C=\begin{bmatrix} 30.9897 & 0 & -40.9965 & 0 & -88.2854 & 0 \\ 57.2958 & 0 & 0 & 0 & 0 & 0 \\ -10.0065 & 0 & 100.6680 & 0 & -47.2894 & 0 \end{bmatrix} \tag{5.4}$$

$$D=\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{5.5}$$

5.6 Closed Loop Step Response

In this Simulink model, is tested and the response of the robot tracking without adding of a controller to the system, the model and results are shown below in the next two figures, Fig 5.6 and Fig.5.7

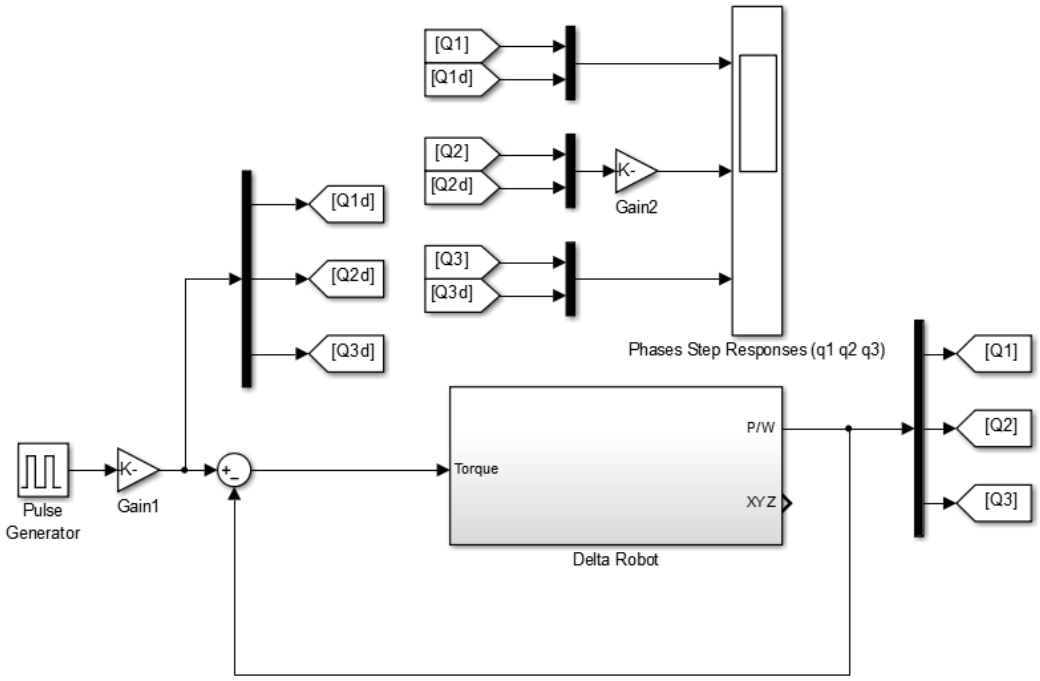


Figure 5.6: Simulink Model with No Controller

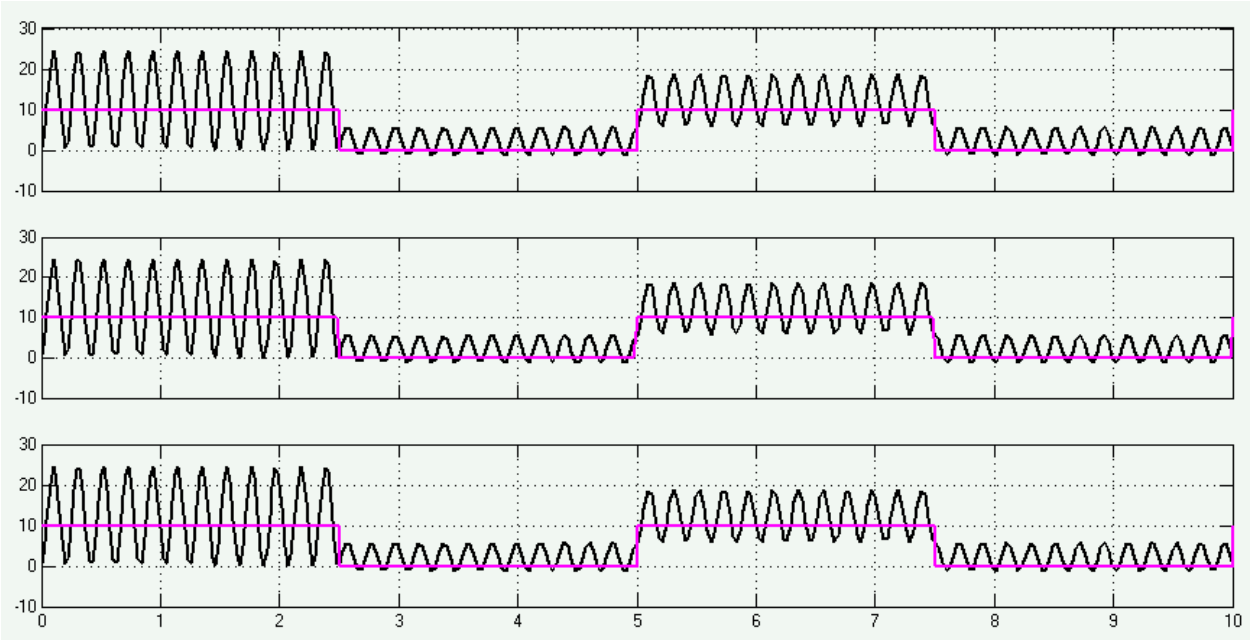


Figure 5.7: phase's response at Joints q1, q2 and q3.

From the above figure , a concolusion was drawn that the MIMO Delta Robot Dynamics can not reach the desired trajectory without help of suitable controller. In the next sectin , the examines of a calssical PID controller shown and illustrated in Fig.5.8.

5.7 Classical PID Controller

From Simulink library, a PID block is added to the Simulink model of my project. This block linearize the plat seen by PID controller and check the step response of the linearized plant. Fig.5.7 shows the Simulink model with PID controller. You must pay attention that PID controller block cannot linearize MIMO systems, therefore, one robot link was chosen for linearization and Control. After that, the result of the controller parameters was copied to the other two link controllers of Delta robot for actuation.

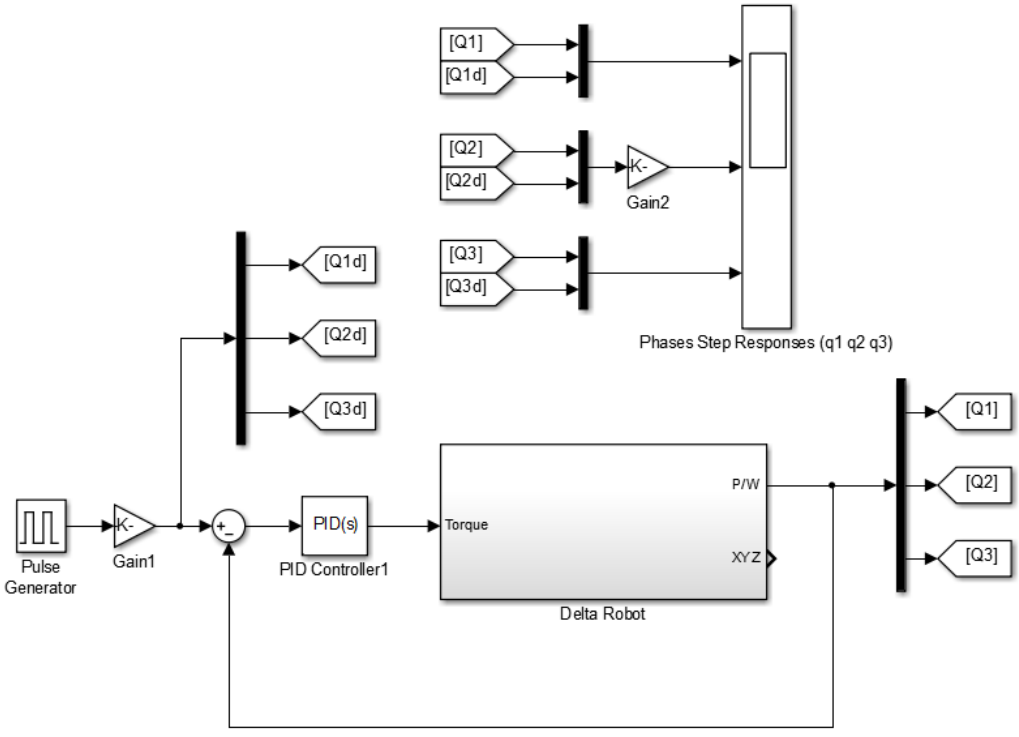


Figure 5.8: Delta Robot Model with PID controller.

5.7.1 Joint Angles

The following figures depict the error between the desired and actual joint angles of q_1, q_2, and q_3 respectively. These are the only three angles directly controllable by the actuators of the parallel robot.

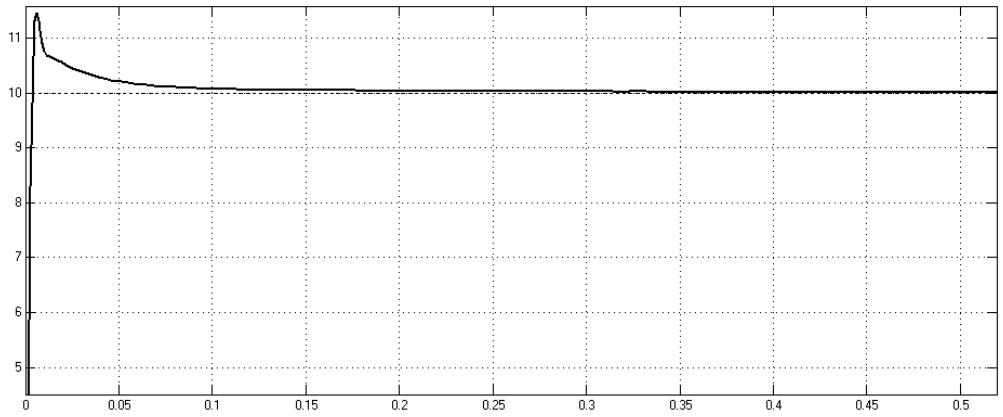


Figure 5.9: Joint Angle error for PID controller.

It is crucial that the difference between the desired angles and the actual angles of q_1, q_2, and q_3 be as small as possible, in order for the end effector error to be minimized. Fig. 5.9 displays a clearly overshoot, approximately 15%. This value of overshoot is not desired in robotics in general. So, more tuned parameters are got to reach the minimum overshoot. The next table 5.3 lists the values of PID parameters, K_P, K_I, K_D, and N, which guarantee the closed loop stability, and ensures the lowest value for overshoot and error.

Table 5.3: PID Controller Parameters

Controller Parameters	Tuned Values
Kp	*50*
ki	*0*
Kd	*3*
N	*1000*

The step response of the moving platform where, (q_1, q_2, q_3) = (-20, -20,-20).

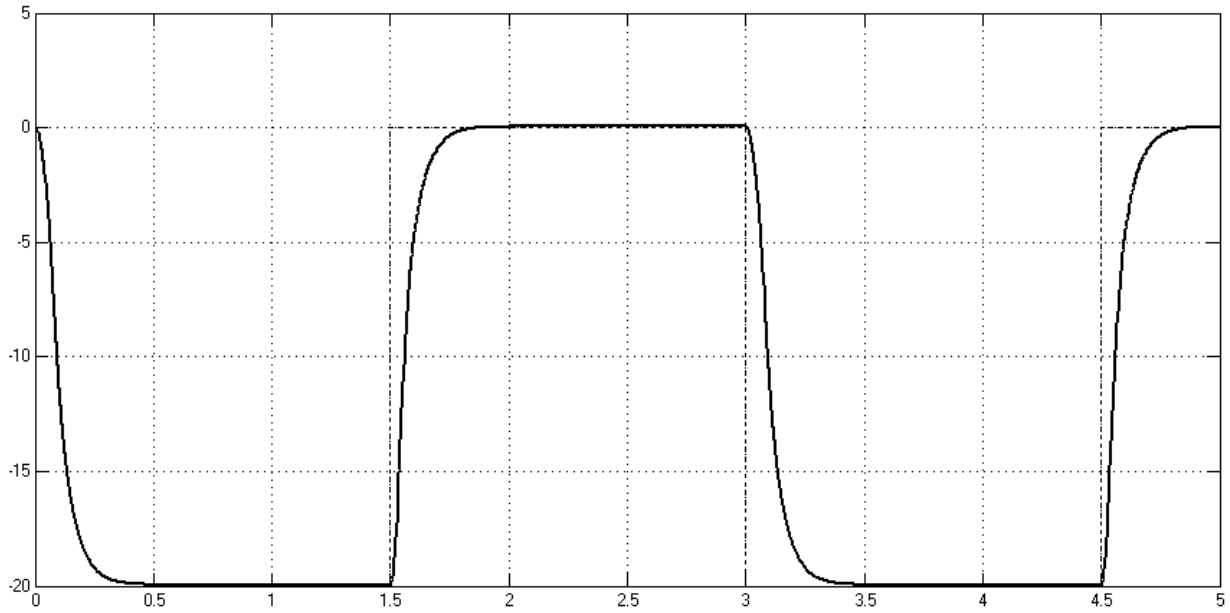


Figure 5.10: q_1 Joint Angle for PD controller in Table 5.3.

The constraint that ensured unbiased results was the fact that the motors utilized in the physical model of the parallel robot could output a maximum torque of 5 Newton meters. This will be confirmed in the subsequent torque plots.

5.7.2 Controller Output - Torque

The next figure illustrates the torque generated by the controller which will move the links to their actual position. The next figure, Fig.5.11, displays the output of PID controller during two seconds with PID parameters listed in Table 5.3.you must note that no actuator is considered in the previous Simulink model. Torque is already delivered by controller, in practical, this situation is not true. The next Simulink models will consider the servo motor actuators.

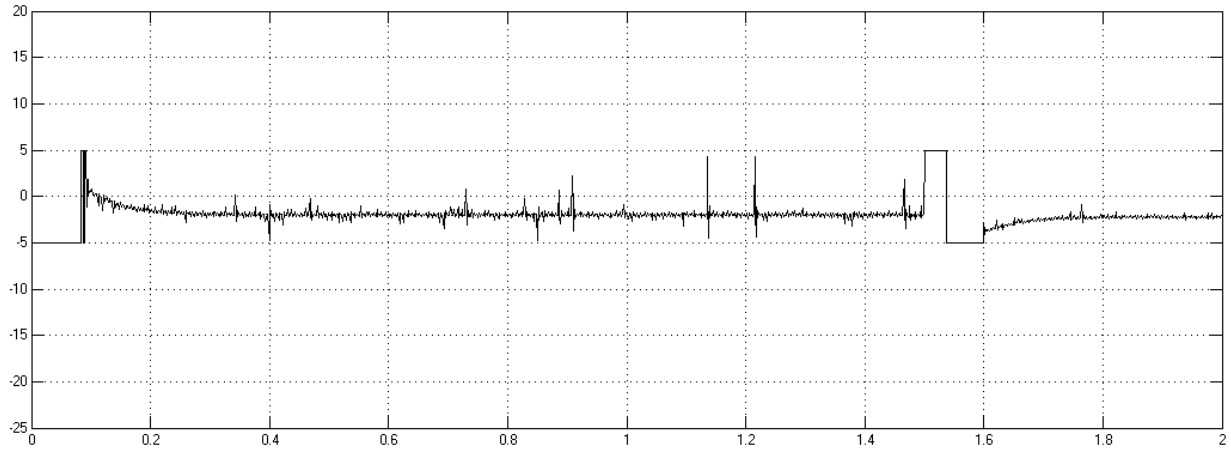


Figure 5.11: Torque output for PID controller for q_1 Joint.

5.8 Actuator Simulation

For Delta robot mechanics, each joint is actuated with a highly geared DC servo Motors, their mechanical and electrical parameters are listed in table 5.4. I got the help of SimDrive tool embedded in Matlab. The following Figure illustrates the Simulink model of DC motor with 31:1 gear ratio.

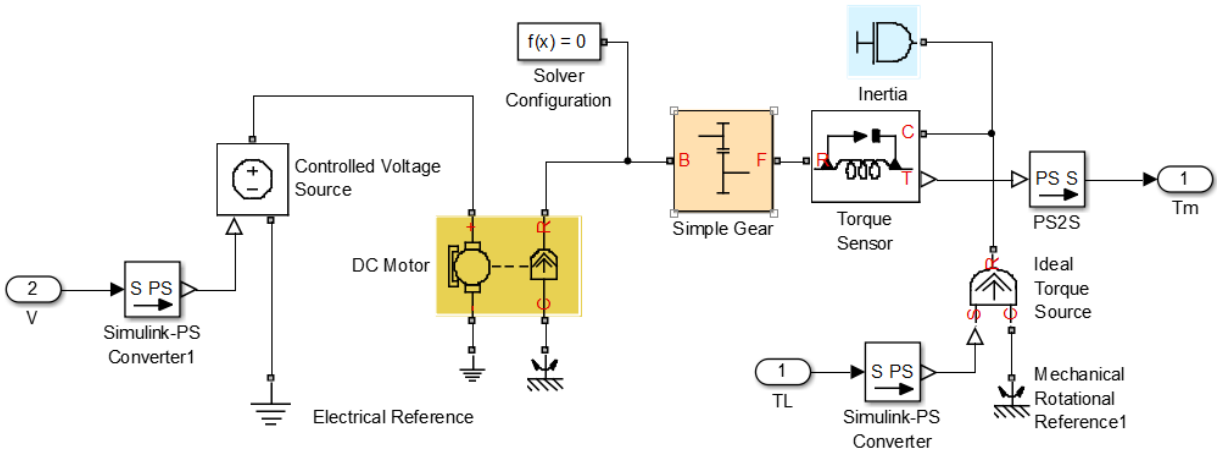


Figure 5.12: DC motor with simple gear Model.

The following table, table 5.4, summarizes the expected DC motor parameters.

Table 5.4: DC motor electrical and mechanical Parameterization

Electrical Parameterization	
Armature Inductance (L)	1600e-6
No-load Speed (rpm)	1200
Rated Speed at rated Load (rpm)	100
Rated Load (Mechanical Power/Watt)	75
Rated DC Supply Voltage(V)	24
Rotor Damping Parameterization	
No-Load Current (I)	0.2
DC Supply Voltage When Measuring No-Load Current (V)	1.5
Mechanical Parameterization	
Rotor Inertia (kg*m^2)	2.5e-5
Gear Ratio	31:1
Gear Inertia (kg*m^2)	0.05

For testing the behavior of the actuator dynamics, the next figures, Fig.5.13 and Fig. 5.14, illustrates the voltage step response for speed and torque, with no load. Fig.5.13 illustrates the voltage step response of the geared DC motor, with gear ratio 10:1. The dashed curve in Fig.5.13 shows the rotor speed. And the other curve shows the gear's shaft speed.

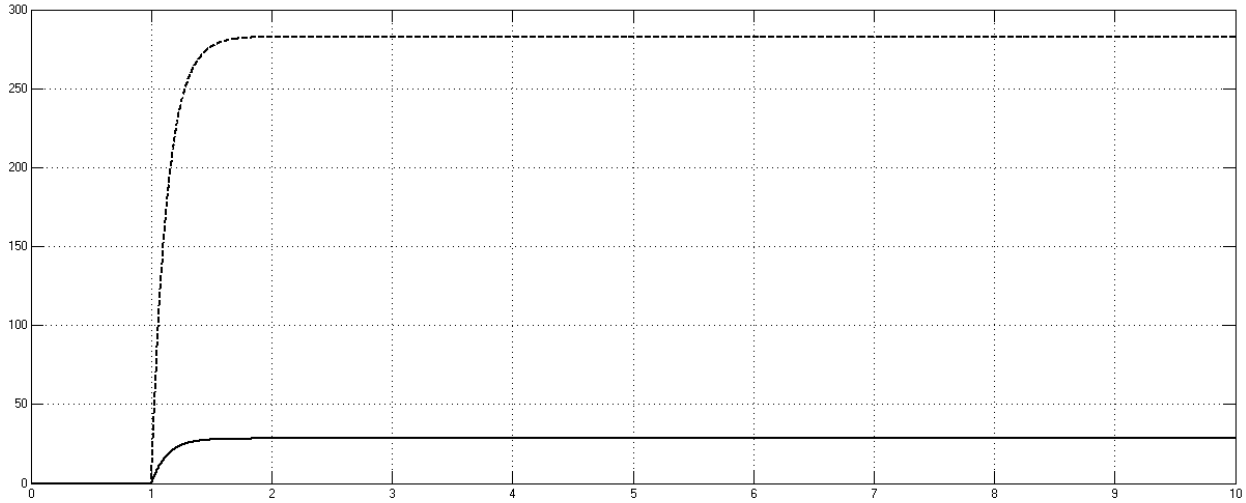


Figure 5.13: rotor's shaft speed V.S gear's shaft speed for DC Motor

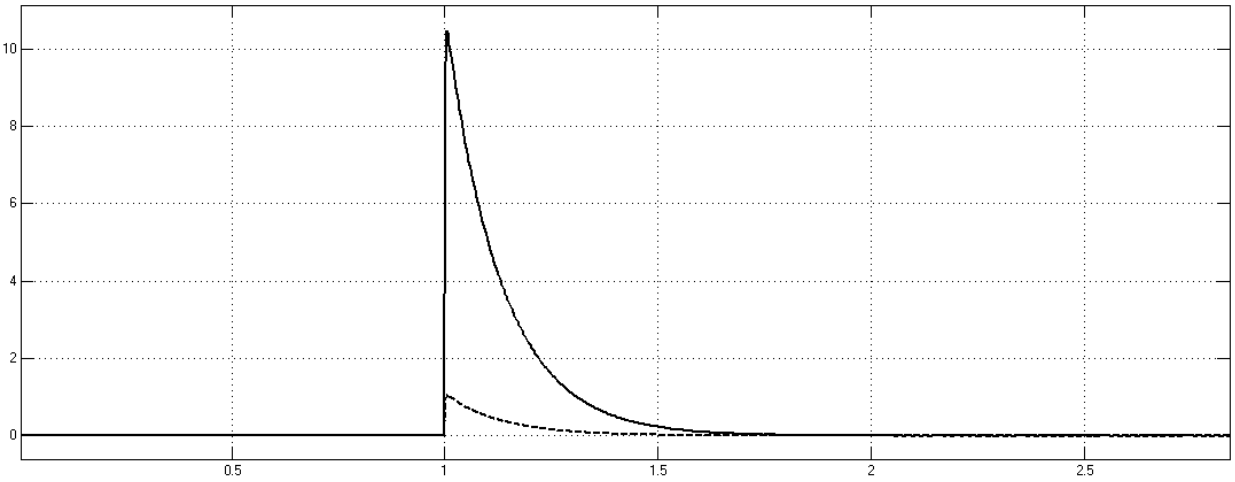


Figure 5.14: rotor's shaft torque V.S gear's shaft torque for DC motor.

From Fig. 5.13 and Fig. 5.14, we notice that,

$$\begin{aligned} w_f &= r^{-1} w_B \\ \tau_f &= r\tau_B \end{aligned} \tag{5.6}$$

Where W_F, T_F is the gear's follower angular speed and torque respectively, and W_B, T_B is the gear's base angular speed and torque respectively. And r is the gear ratio.

5.9 Actuated DELTA Robot Simulation

In the previous Simulink models, that torque generation is considered to be the duty of the controller, but, in practical, the controller generates PWM voltage varies from 0 volt to + 5 volt, which are the Microcontroller voltages. H-bridge circuit amplifies the input PWM voltage and actuates the DC motor directly. Each link is energized with highly geared servo DC motor. Fig.5.15, illustrates the PWM voltage-based Simulink model.

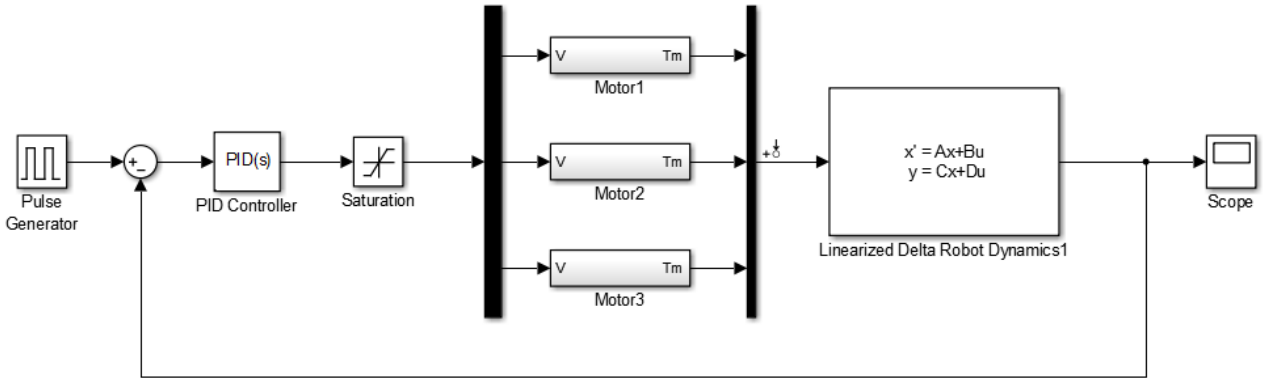


Figure 5.15: Linearized Delta Robot model with DC motor Actuators.

You can see from Fig. 5.14 that the motor block gets the voltage as an input and generates mechanical torque.

5.9.1 Joint Angles

Addition of a DC motor models in the simulink model , pushed me to re-tune PID parameters to get the optimal one approximately. As shown in the next table.

Table 5.5: PID controller parameters

Controller Parameters	Tuned Values
Kp	*90*
ki	*35*
Kd	*2*
N	*1000*

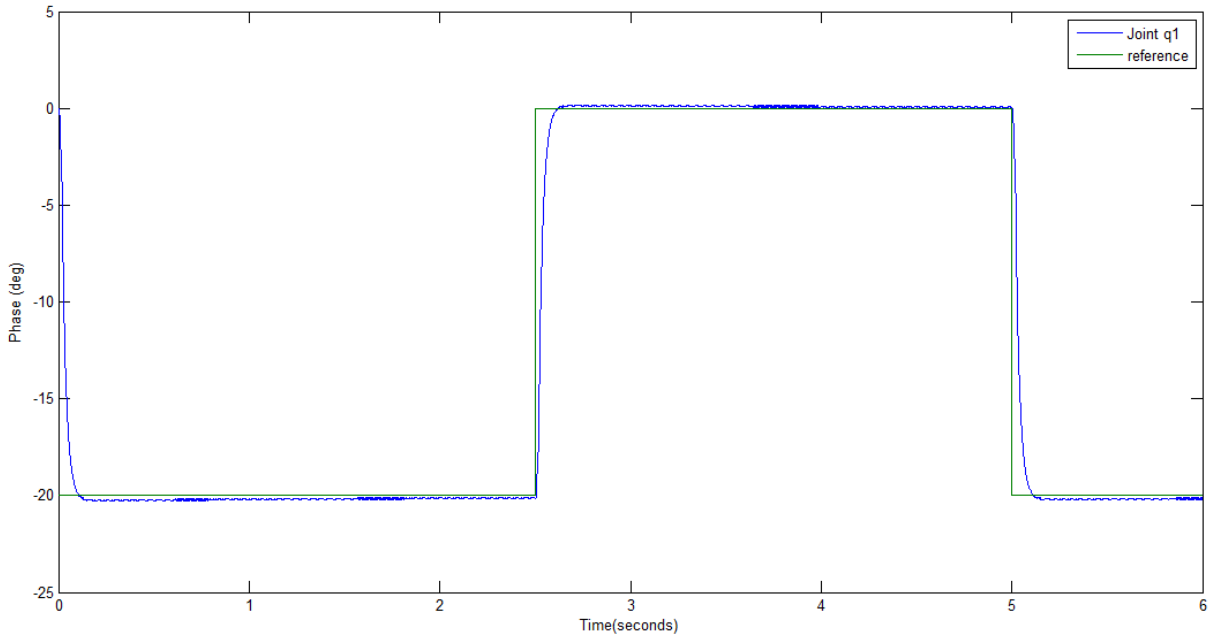


Figure 5.16: voltage pulse train Response for joint q_i.

The other two joints have the same response, because the step input is in the z direction.

5.10 Trajectory Generation

The end effector of the three degrees of freedom parallel robot was simulated to follow a circular trajectory based on the implementation of the desired controller. The tracking speed utilized is defined by the angular velocity formula: $\omega = 2\pi f$, where f is the tracking frequency of the end effector. The origin of the circle based on the Cartesian coordinate system in millimeters is defined as $(0, 0, z_0)$, where the distance of travelling in z- direction .the radius

of the circular trajectory is 100 millimeters and the frequency implemented is 0.5 Hertz. It should be noted that the trajectory defined in this report never impedes or approaches any singular point.

The next Simulink model generates a circular path trajectory in the x-y plane.

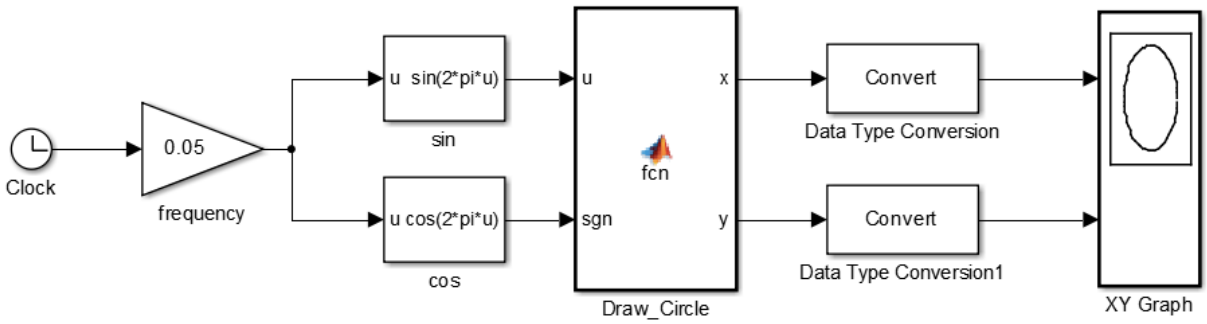


Figure 5.17: Circle path trajectory model.

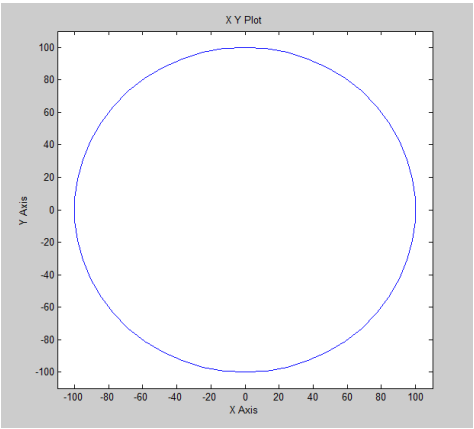


Figure 5.18: output circle for model in Fig.5.17

5.11 End Effector Trajectory

The next set of figures portrays the trajectory tracking of the end effector along with its respective error between its desired and actual position. In order to determine the actual location of the end effector, equations (2.7) and (2.10) were employed to solve for q1, q2 and q3 respectively. The Inverse kinematics equations were then applied based on all the available data to

definitively determine the actual location of the end effector in Cartesian coordinates. The results from the joint angles of q1, q2 and q3 play a crucial role in the determination of the actual location of the end effector, since a small angular error would not cause a large deviation when compared to the desired trajectory. The plots of the end effector error in the x-axis and y-axis are shown in order to easily identify the severity of the absolute accuracy.

The next Simulink Model illustrates the End effector tracking for a circle lies on X- Y plane with radius r= 100 mm.

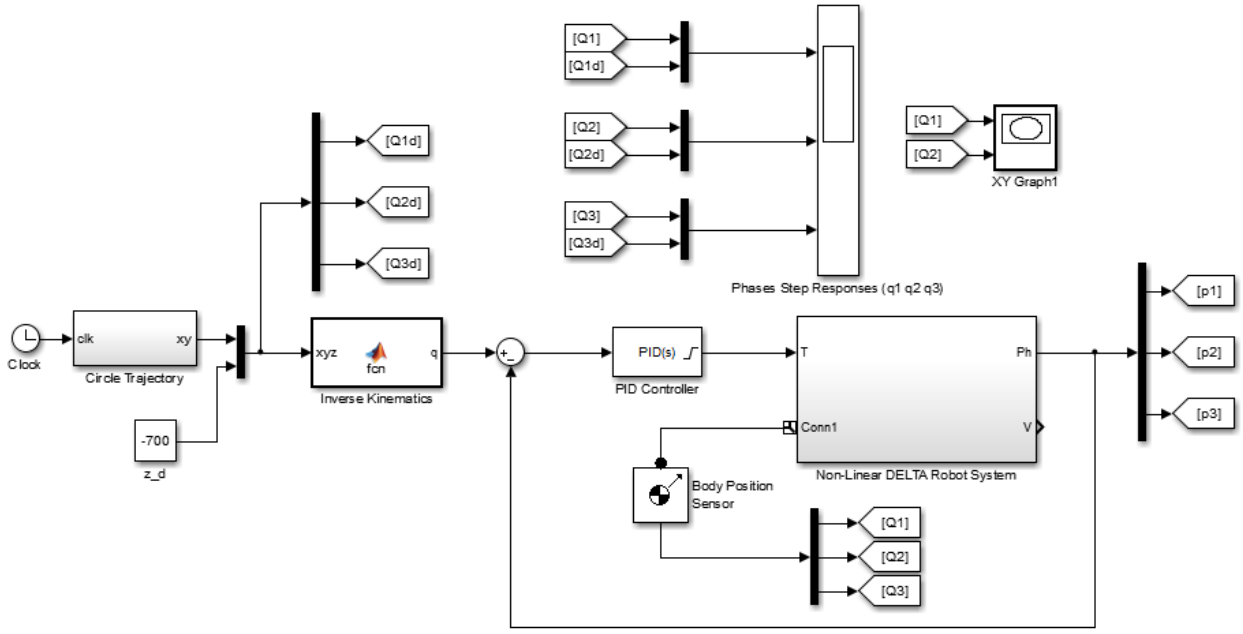


Figure 5.19: Circle Path Trajectory for DELTA Robot.

The next figures display the End Effector error on x-axis, y-axis and z-axis and display the End Effector trajectory.

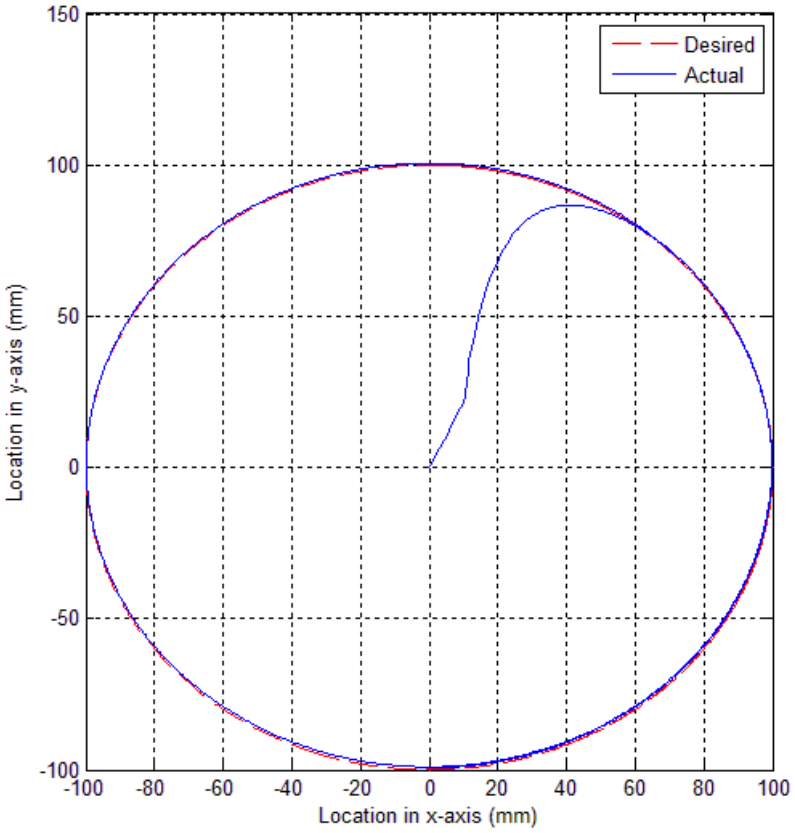


Figure 5.20: End Effector Trajectory for PID Controller.

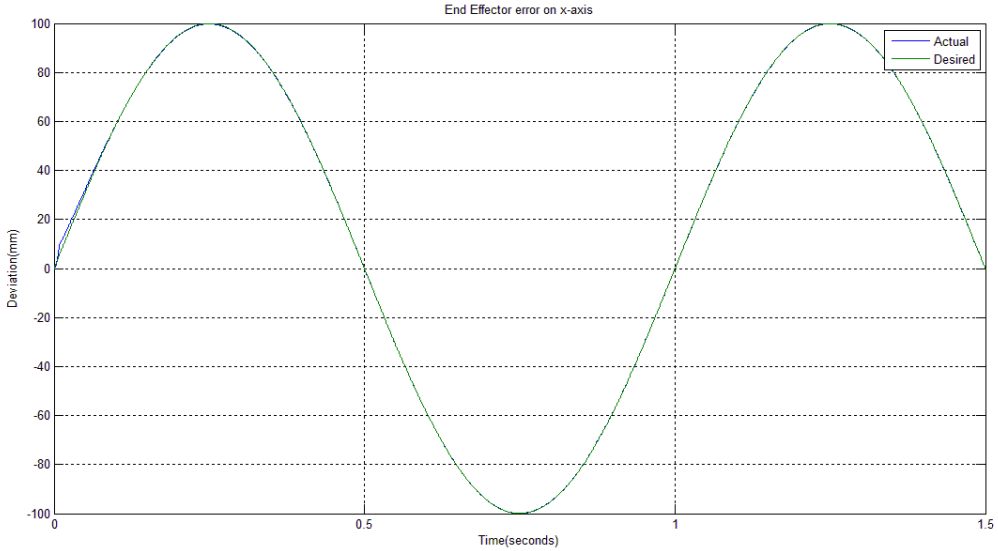


Figure 5.21: End Effector Trajectory error on x-axis for PID Controller.

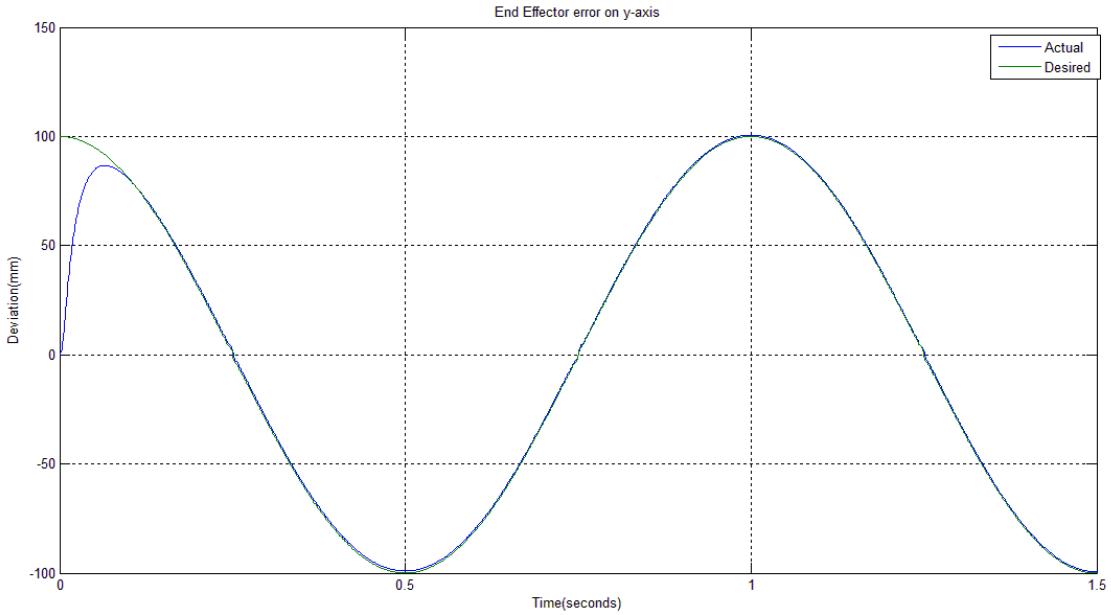


Figure 5.22: End Effector Trajectory error on y-axis for PID Controller.

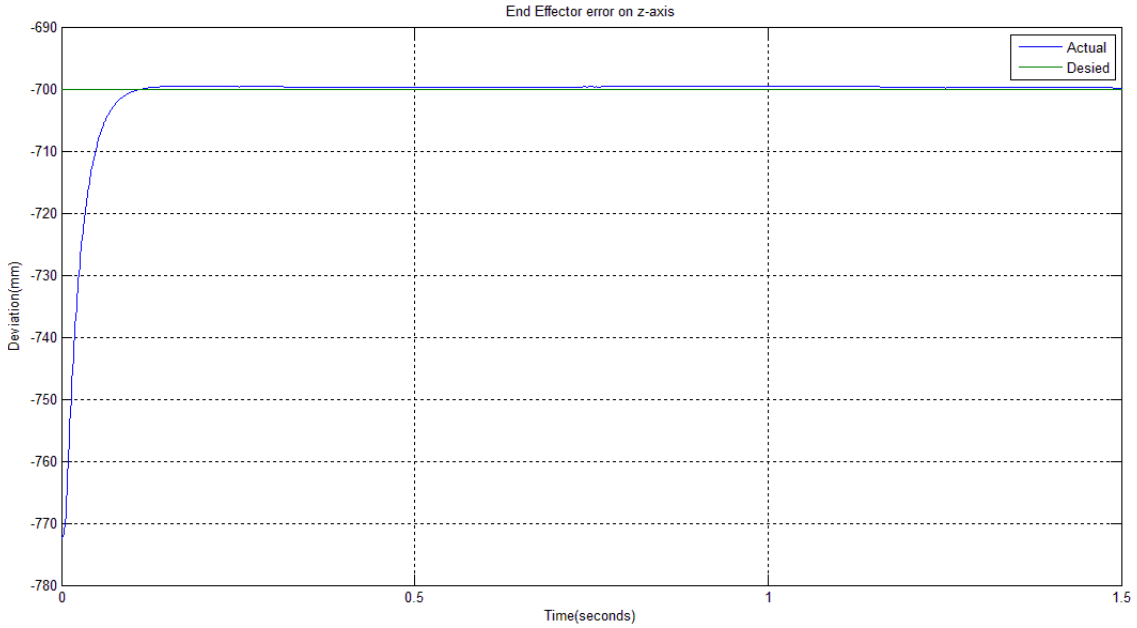


Figure 5.23: End Effector Trajectory error on z-axis for PID Controller.

5.12 Model Predictive Controller (MPC) Simulation

Model Predictive Control Toolbox™ provides tools for systematically analyzing, designing, and tuning model predictive controllers. You can design and simulate model predictive controllers using functions in MATLAB® or blocks in Simulink®. You can set and modify the predictive model, control and prediction horizons, input and output constraints, and weights. The toolbox enables you to diagnose issues that could lead to run-time failures and provides advice on changing weights and constraints to improve performance and robustness. By running different scenarios in linear and nonlinear simulations, you can evaluate controller performance. You can adjust controller performance as it runs by tuning weights and varying constraints.

In the next Simulink Model, I will test the MPC controller on a linearized model for DELTA robot and comparing the output results with the previous PID controller outputs.

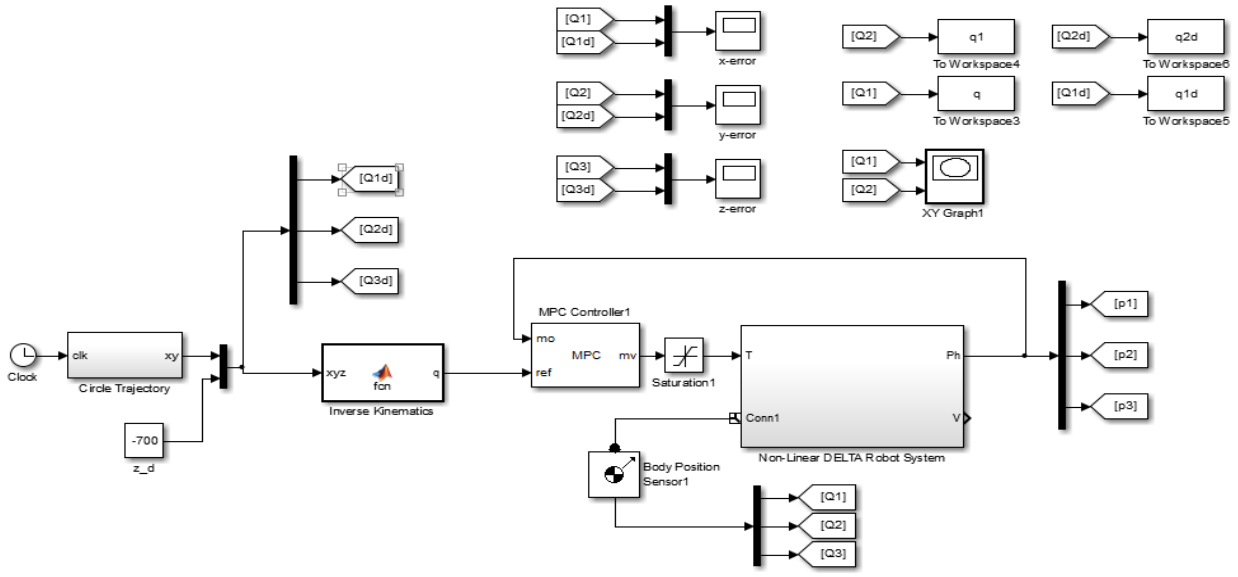


Figure 5.24: MPC Controller based – DELTA Robot Simulink Model

5.12.1 Joint Angles

The following figures depict the error between the desired and actual joint angles of q_1, q_2, and q_3 respectively using MPC Controller.

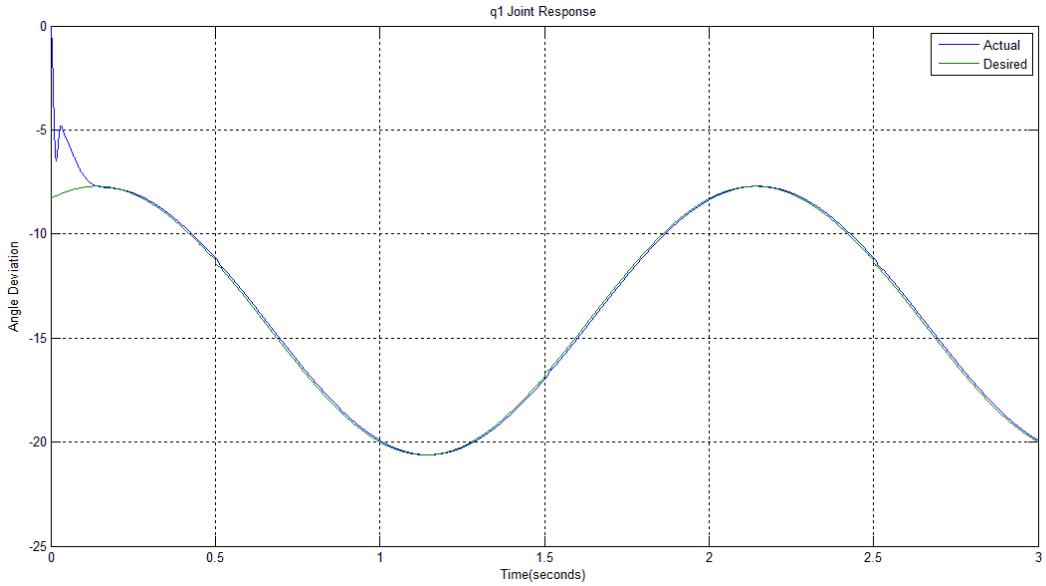


Figure 5.25: Joint q_1 Trajectory error for MPC Controller.

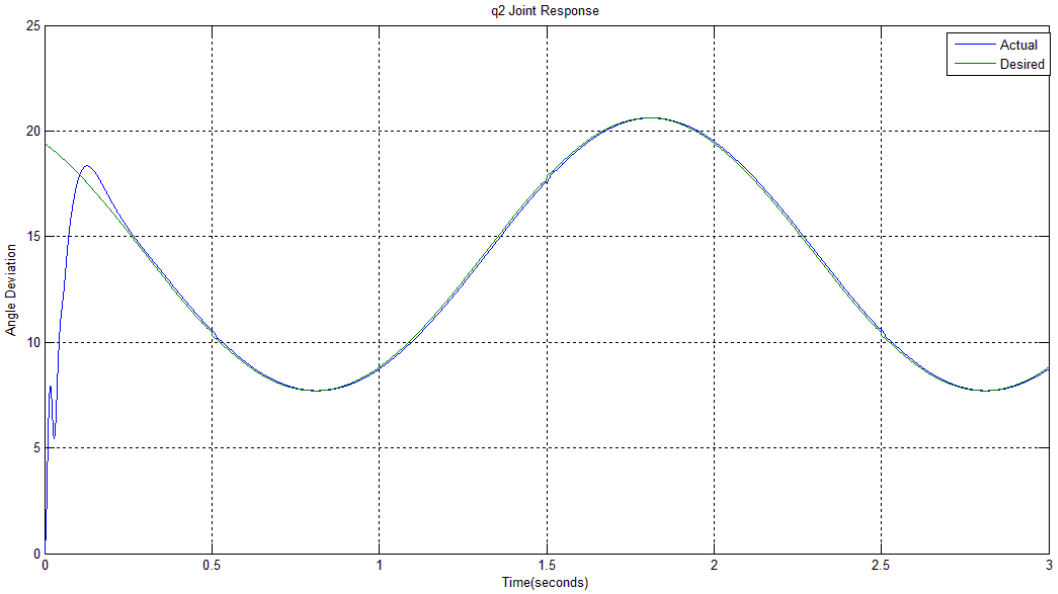


Figure 5.26: Joint q_2 Trajectory error for MPC Controller.

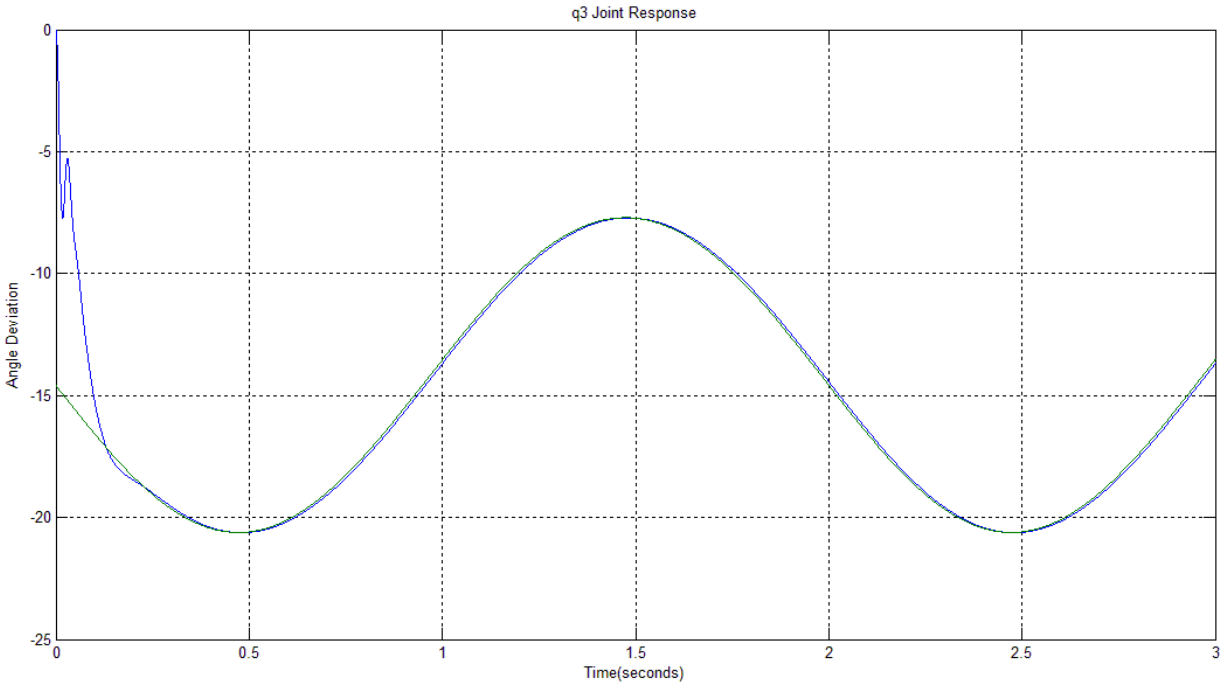


Figure 5.27: Joint q_3 Trajectory error for MPC Controller.

We can notice that MPC controller gave the same results of PID controller approximately. Then, a great contribution added at the response of DELTA robot. Robot arms are highly geared; this reason let the robot to be more robust.

5.13 Recommended Controller

The true decision for choosing the suitable controller between PID and MPC controllers which satisfy the desired criteria is taken after running the next Simulink model, as shown in Fig 5.28

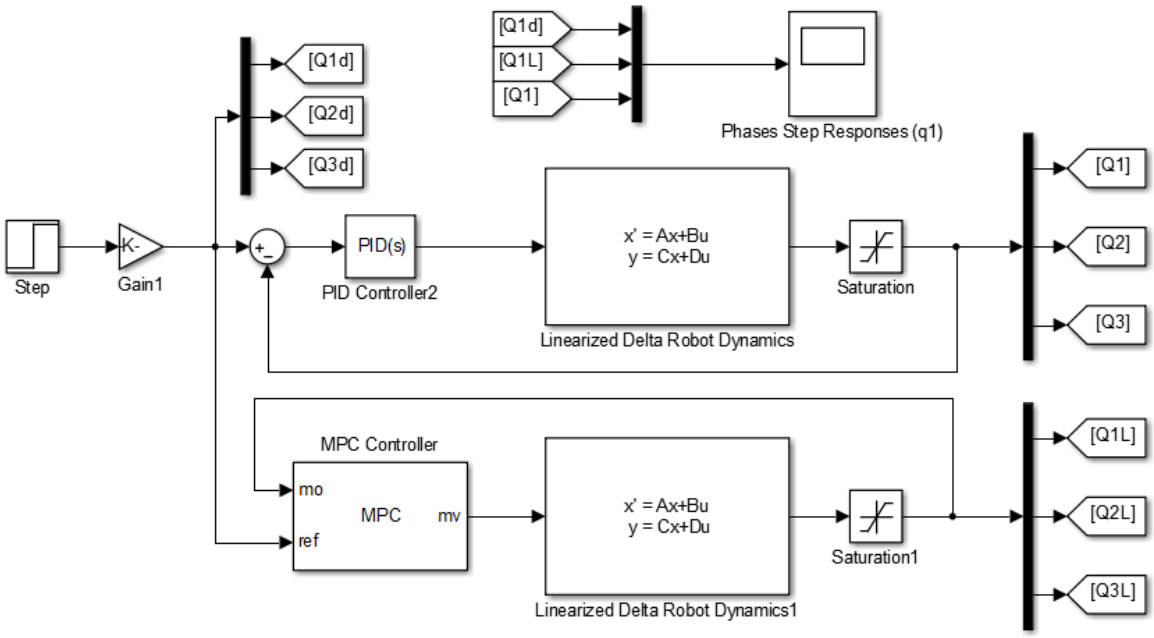


Figure 5.28: Linearized Delta Robot Model with PID and MPC Controllers.

The values of PID parameters which are choose in the previous Simulink model, listed in the following table 5.6.

Table 5.6: PID controller parameters

Controller Parameters	Tuned Values
Kp	*15*
ki	*90*
Kd	*.5*
N	*2000*

Fig.5.29, illustrates the result of q_1 joint response with input angle value equal 10 degrees.

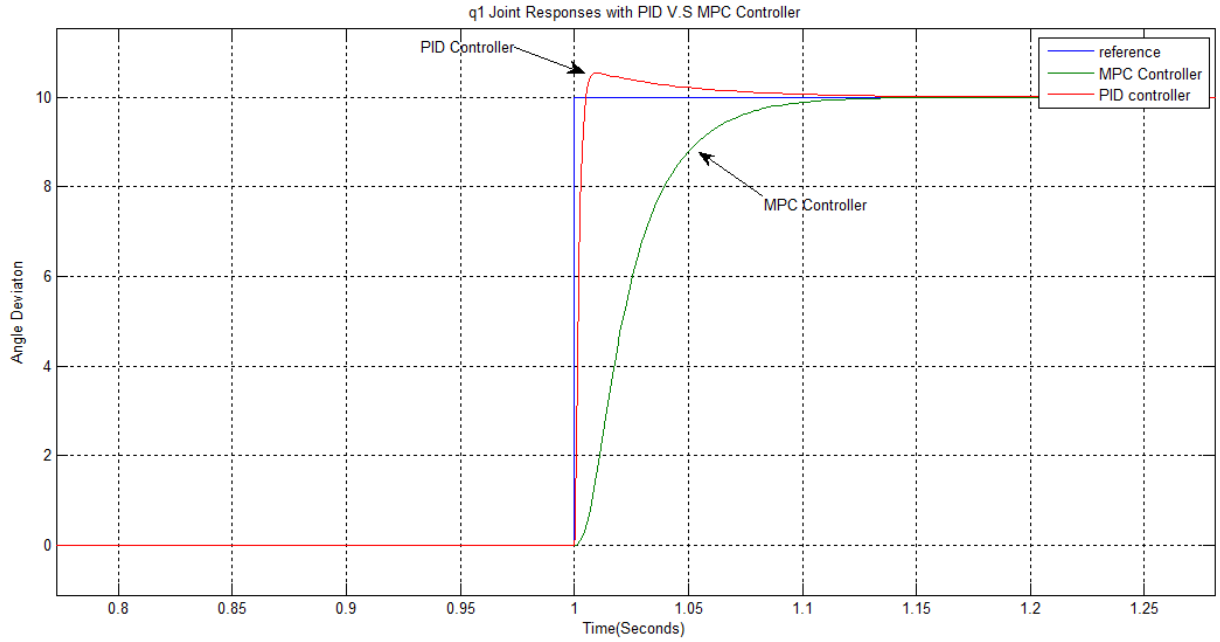


Figure 5.29: q_1 Joint Responses with PID and MPC Controllers.

The following table, compare the output results for q_1 joint responses with PID and MPC Controllers.

Table 5.7: Comparison table between PID and MPC controller responses

Response Characteristic	**PID Controller**	**MPC Controller**
Overshoot (%)	5.3%	0%
steady state error	0	0
Settling Time (sec.)	0.1	0.12

We can conclude that the joint response has no overshoot value with MPC controller comparing with PID controller as illustrated in Table 5.7.

5.14 Simulation Discussion

The final purpose of Simulink simulation is to test path tracking response of DELTA robot. Building the Nonlinear Dynamic Model using Autodesk Inventor program was the easiest solution to model the robot. Real and not approximated parameters were introduced via CAD dynamic Modeling which is one of the advantages of CAD modeling, but in the other side, the simulation speed is low. The simulation speed of Mathematical dynamic Model is more rapid than CAD dynamic model. Low simulation speed is annoying problem.

SimMechanics tool introduced an easy way to actuate robot joints by joint actuator block. Sensing computed torque, angle, angular speed, and acceleration was by joint sensor. Sensing of Cartesian coordinates of the moving platform was by Body sensor, which facilitated the process of simulation.

DELTA robot needed a high precision controller to control it, PID and MPC controllers were added. At the output of these two controllers, the output torque command was limited, because that Servo DC motors attached at each actuated joint generate a limited torque. PID parameters in Table 5.6 were optimally tuned by ***desired response block*** in Matlab Simulink. Fig. 5.28 shows that, with PID Controller and at an input angle of 10 degrees, DELTA robot upper arm can rotates with no steady state error and no overshoot when MPC controller is applied. But the value of an overshoot reached to 5.3% when PID controller was applied. Each of PID and MPC controllers guaranteed no steady state error as shown is Fig. 5.29.

Chapter 6

Conclusion and Future Work

8.1 Conclusion

The purpose of this thesis was to compare the simulation results of the non-adaptive PID and MPC controllers; MPC is the most suitable control technique to employ on the 3DOF Parallel DELTA structure. A summary of the differences between serial and parallel robot structures introduced the background of robotics, while the literature review provided a detailed account of the beginning of robotics.

The 3DOF Parallel DELTA robot was introduced and modeled using SimMechanics Matlab Tool, inverse kinematics and non-singular region in order to adequately define the parameters of the non-linear system. The derivations of the two controllers were solved to ensure stability of the closed loop system. This led to the simulation of the PID and MPC controllers in MATLAB to analyze whether the actual circular trajectory could satisfactorily track the desired circular trajectory. Once this task was completed, the electrical and mechanical design of the physical parallel robot structure was discussed in detail. The two controllers attained accurate end effector tracking results without compromising the amount of computation time and control effort usually found in more complex control techniques. It is highly recommended that the PID controller be utilized in various parallel robot structures to determine if similar results can be achieved, moreover, PID controller is easy to implement in Microcontrollers, familiar with us.

Finally, the proposed model in this thesis will open the road to master students who wish to continue their graduate study in the field of motion precision of the moving platforms of control systems.

8.2 Future Work

Each of PID and MPC controllers has proved the stability of the DELTA Robot, but the design was under the assumption that Mechanical parameters are certain, uncertainty is not considered in this thesis. The performance of the MPC and PID controllers can be improved if we designed adaptive controllers.
I started in building of DELTA robot structure, Experimental results and path trajectory tracking tests will be applied at real DELTA robot for validating these results with simulation results.

References

[1] John J. Craig, "Introduction to robotics, Mechanics and Control, Third Edition, 2005.

[2] Clavel, R., Delta, a Fast Robot with Parallel Geometry. 18th International Symposium on Industrial Robot, pp. 91-100, April 1988.

[3] YangminLi, Qingsong Xu," Dynamic modeling and robust control of a 3 PRC translational parallel kinematic machine", Robotics and Computer-Integrated Manufacturing journal, Science Direct, 2009.

[4] André Olsson, Modeling and control of a Delta-3 robot, master thesis, Department of Automatic Control, Lund University, 2009.

[5] Angelo Liadis, fuzzy logic control of a two degree of freedom parallel robot, Master thesis, Department of Electrical Engineering, Lakehead University, 2010.

[6] Manuel Napoleon Cardona Gutierrez, "Kinematics Analysis of a Delta Parallel Robot", IEEE, University of Sonsonate, Salvador,2010.

[7] P.J. Zsombor Murray," An Improved Approach to the Kinematics of Clavel's DELTA Robot", Center for Intelligent Machines, McGill University, 2009.

[8] Xia Wu, Jun Lin and Zhencai Zhu," Inverse Kinematics and Singularity Analysis for a 3-DOF Hybrid-driven Cable-suspended Parallel Robot", International Journal of Advanced Robotic Systems, INTECH, 2012.

[9] Mohsen, Mahdi, Mersad," Dynamics and Control of a Novel 3-DoF Spatial Parallel Robot", International Conference on Robotics and Mechatronics,2013.

[10] Serdar Kucuk and Zafer Bingul,"Robot Kinematics: Forward and Inverse Kinematics, Industrial Robotics: Theory, Modelling and Control", Sam Cubero (Ed.), ISBN: 3-86611-285-8, InTech,2006.

[11] Reza N. Jazar," Theory of Applied Robotics, Kinematics, Dynamics and control", second edition, Springer, RMIT university,2010.

[12] Yangmin Li and Qingsong Xu ,"Dynamic Analysis of a Modified DELTA Parallel Robot for Cardiopulmonary Resuscitation", Department of Electromechanical Engineering, Faculty of Science and Technology University of Macau, 2004.

[13] Modeling Multi-body Systems, www.mathwork.com.