



# **MAE 547- Modeling and Control of Robots**

## **Project Report**

**Instructor- Prof. Hamidreza Marvi**

<b>Team Member</b>	<b>ASU ID</b>
<b>Amarnath Periyakaruppan</b>	<b>1225607929</b>
<b>Anirudh Iyengar</b>	<b>1225709589</b>
<b>Kyle Welsh</b>	<b>1218512607</b>
<b>Rohit Sanjay Ganesh</b>	<b>1232064459</b>
<b>Sherry Daniel Sajan</b>	<b>1225838460</b>

# Table of Contents:

<b>1. Introduction:</b>	<b>3</b>
1.1 Manipulator Dynamics:	3
1.2 Manipulator Control:	3
<b>2. Governing Equations</b>	<b>4</b>
<i>The output equations of motions from our MATLAB code are given below. The MATLAB file and its content is also attached.</i>	
Coriolis Matrix (C):	4
<b>3. Robotics Toolbox:</b>	<b>5</b>
<b>4. Simulating dynamics of the system:</b>	<b>6</b>
4.1. Forward Dynamics:	7
4.2. Motion Equations:	8
4.3. Results:	9
4.3.1. Forward Dynamics with Equations of Motions:	9
4.3.2. Equations of Motions:	10
<b>5. Indirect Force Controls:</b>	<b>11</b>
5.1. Compliance Control:	11
5.1.1. Results:	12
5.2. Impedance Control:	13
<b>6. Contributions:</b>	<b>15</b>
<b>7. References:</b>	<b>15</b>

# 1. Introduction:

## 1.1 Manipulator Dynamics:

Dynamics involves the study of forces, particularly in relation to manipulators. Manipulators, to execute tasks, necessitate movement of their components and joints. This movement requires the application of forces and torques in specific manners to achieve desired trajectories, which is the focus of Dynamics of Manipulators. When a manipulator is tasked with performing a specific action, it must be initially accelerated from rest. Subsequently, it may need to maintain a constant velocity before decelerating to rest at a designated location. Achieving such motion entails adjusting torques at the joints through actuators to align with the desired trajectory.

The primary objective in Dynamics of Manipulators is to determine the torque requirements for the manipulator joints. These torque functions are contingent upon several factors including the intended trajectory, the masses of the manipulator links, joint friction, and external forces applied at the end-effector. The analysis of manipulator dynamics typically involves two types of problems:

1. Given a trajectory with specified variations in position, velocity, and acceleration, the task is to ascertain the torques necessary at the manipulator joints to traverse this trajectory.
2. Given variations in torques, determining the motion of the manipulator, which may entail determining position, velocity, and acceleration, all in angular terms.

## 1.2 Manipulator Control:

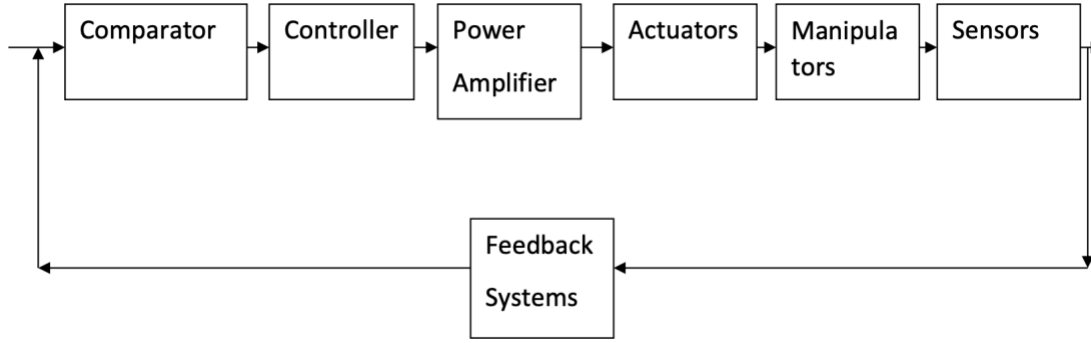
For a robot to effectively carry out tasks, it is essential that its end effectors move with precision and consistency. Manipulator control plays a crucial role in ensuring that the end effectors move as intended to accomplish the task at hand. This control system operates by taking the time history of joint locations as input and adjusts the manipulator's movement to follow the desired trajectory. Within this system, the desired task is provided as input to a trajectory planner. The trajectory planner then generates control set-points, including position, velocity, and acceleration, which are fed into the manipulator control system. Subsequently, the manipulator control system generates actuator commands, guiding the end effector to the desired location accordingly.

When a feedback system is incorporated into the manipulator control, it is referred to as closed-loop manipulator control; without it, it is termed open-loop manipulator control.

With a feedback system in place, the actual location and velocity of the manipulator are provided as feedback to the manipulator control system. This feedback mechanism enables the correction of errors in the end effector's position.

In cases where a robot features multiple joints or arms, a master joint control system becomes necessary. This system is responsible for providing the required set-points to each joint controller.

The key components of any manipulator system include the comparator, controller, power amplifier, actuators, manipulators, sensors, and feedback systems.



## 2. Governing Equations

The output equations of motions from our MATLAB code are given below. The MATLAB file and its content is also attached.

$$\begin{aligned} \text{Tau} = & qdd2*((ml2*a2^2)/4 + (a1*ml2*cos((pi*q2)/180)*a2)/2 + IL2 + Im2*kr2) + qdd1*(IL1 + IL2 + \\ & Im2 + Im1*kr1^2 + (a1^2*ml1)/4 + a1^2*ml2 + (a2^2*ml2)/4 + (a1^2*mm2)/4 + \\ & a1*a2*ml2*cos((pi*q2)/180)) + (981*a2*ml2*cos((pi*(q1 + q2))/180))/200 + \\ & (981*a1*ml1*cos((pi*q1)/180))/200 + (981*a1*ml2*cos((pi*q1)/180))/100 + \\ & (981*a1*mm2*cos((pi*q1)/180))/200 - (a1*a2*ml2*qd2*pi*sin((pi*q2)/180)*(qd1 + qd2))/360 - \\ & (a1*a2*ml2*qd1*qd2*pi*sin((pi*q2)/180))/360 \end{aligned}$$

$$(a1*a2*ml2*pi*sin((pi*q2)/180)*qd1^2)/360 + qdd1*((ml2*a2^2)/4 + (a1*ml2*cos((pi*q2)/180)*a2)/2 + IL2 + Im2*kr2) + qdd2*((ml2*a2^2)/4 + Im2*kr2^2 + IL2) + (981*a2*ml2*cos((pi*(q1 + q2))/180))/200$$

### Inertia Matrix (B):

The elements of the inertia matrix B are computed as the sum of contributions from each link of the manipulator:

$$\begin{aligned} B_{ij} = & \sum_{k=1}^n (ml_k \cdot JpL_{ki}^T \cdot JpL_{kj} + JoL_{ki}^T \cdot R_k \cdot IL_k \cdot R_k^T \cdot JoL_{kj} \\ & + mm_k \cdot Jpm_{ki}^T \cdot Jpm_{kj} + Jom_{ki}^T \cdot Rm_k \cdot Im_k \cdot Rm_k^T \cdot Jom_{kj}) \end{aligned}$$

### Coriolis Matrix (C):

The elements of the Coriolis matrix C are computed using the Christoffel symbols, which are calculated from the derivatives of the inertia matrix B:

$$C_{ij} = \sum_{k=1}^n \frac{1}{2} \left( \frac{\partial B_{ij}}{\partial q_k} + \frac{\partial B_{ik}}{\partial q_j} - \frac{\partial B_{jk}}{\partial q_i} \right) \cdot \dot{q}_k$$

## Gravity Vector (g):

The gravity vector  $g$  is computed by summing up the gravitational torques acting on each joint of the manipulator:

$$g_i = - \sum_{j=1}^n (ml_j \cdot g_0 \cdot JpL_{ji} + mm_j \cdot g_0 \cdot Jpm_{ji})$$

Where:

- $ml_k$  is the mass of the k-th link.
- $mm_k$  is the mass of the k-th motor.
- $JpL_{ki}$  is the Jacobian of the position of the i-th joint of the k-th link.
- $JoL_{ki}$  is the Jacobian of the orientation of the i-th joint of the k-th link.
- $Jpm_{ki}$  is the Jacobian of the position of the i-th joint of the k-th motor.
- $Jom_{ki}$  is the Jacobian of the orientation of the i-th joint of the k-th motor.
- $R_k$  is the rotation matrix representing the orientation of the k-th link.
- $Rm_k$  is the rotation matrix representing the orientation of the k-th motor.
- $IL_k$  is the inertia tensor of the k-th link.
- $Im_k$  is the inertia tensor of the k-th motor.
- $g_0$  is the gravitational acceleration.
- $q_k$  is the k-th joint position.
- $\dot{q}_k$  is the k-th joint velocity.

## 3. Robotics Toolbox:

This toolbox is designed for MATLAB versions 2019b and newer and relies on the Peter Corke toolbox, version 9.10 (the standard edition). To utilize the toolbox, you must have all the files from the folder, including the RVC tools file. Adjust the MATLAB path to the directory where you have stored the toolbox files, as illustrated in Figure 1.

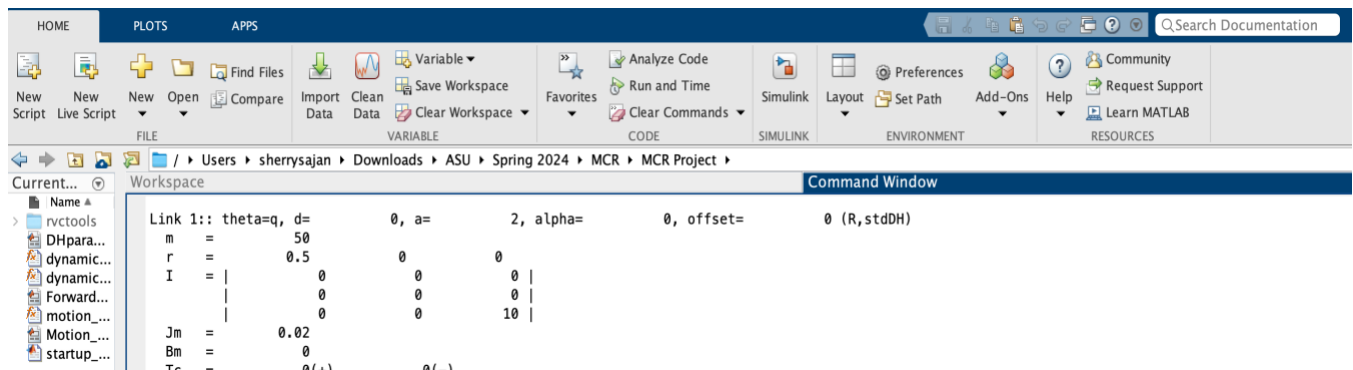


Figure 1-

## 4. Simulating dynamics of the system:

Now open the **DHparameters.mlapp** file using the open command in MATLAB as shown in Figure 1.

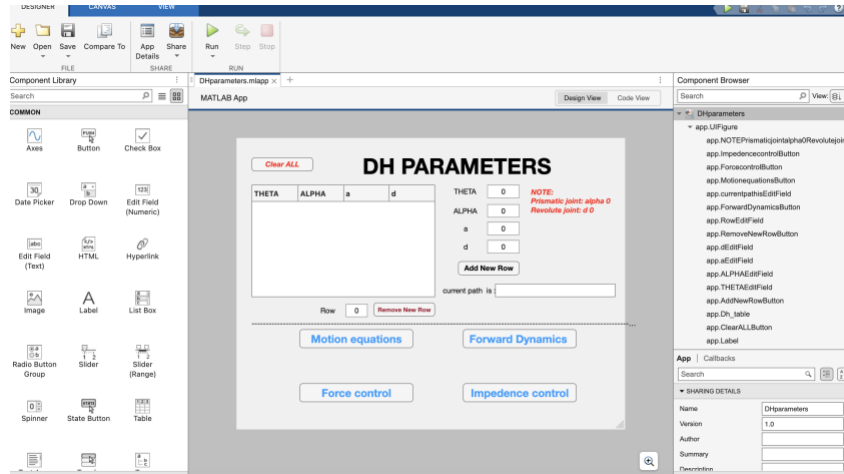


Figure 2

**Step 1.** After opening the DHparameters.mlapp file use the run command to start the Toolbox as shown in Figure 2.

**Step 2.** Once the toolbox is open as shown in figure 3, enter the DH Parameters for 3 Link robot that you want to create. For each link enter the 'Theta', 'Alpha', 'a', 'd' values and click on add new row and the link with the DH parameters will be added to the table. If you would like to remove a row, click on 'Remove New Row'.

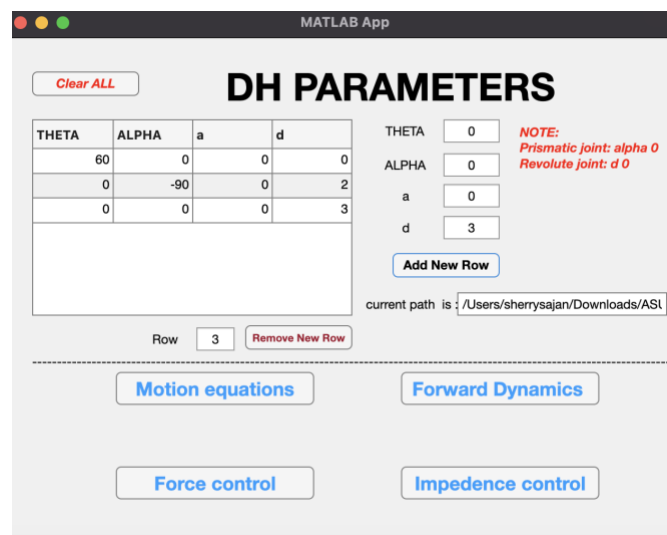


Figure 3

**Step 3.** Choose the Operation option (Motion Equations, Forward dynamics) that you would like to perform.

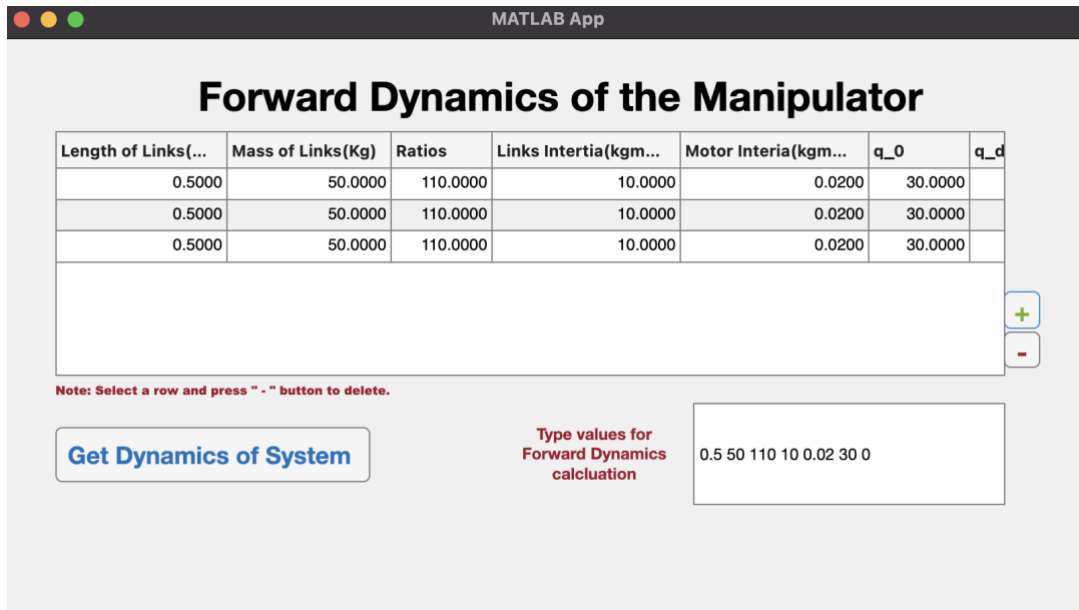
## 4.1. Forward Dynamics:

**Step 1.** If you chose this option, then a toolbox would appear as shown in Figure 4.

**Step 2.** Enter the values of each column in the same order in the box provided below the table. These columns include length of link, Mass of link, Gear ratio, Link Inertia, Motor Inertia, Initial Position of joint and velocity of joint.

**Step 3.** After entering these values click on '+' sign and the values will be automatically entered in the dynamics table. Follow the same procedure for the other 2 links as shown in Figure 4.

**Step 4.** After all values of links are provided, click on 'Get Dynamics of System'.



Length of Links(...)	Mass of Links(Kg)	Ratios	Links Inertia(kgm...)	Motor Inertia(kgm...)	q_0	q_d
0.5000	50.0000	110.0000	10.0000	0.0200	30.0000	
0.5000	50.0000	110.0000	10.0000	0.0200	30.0000	
0.5000	50.0000	110.0000	10.0000	0.0200	30.0000	

Note: Select a row and press "-" button to delete.

Get Dynamics of System

Type values for Forward Dynamics calculation

0.5 50 110 10 0.02 30 0

Figure 4

**Step 5.** After you click on 'Get Dynamics of System', you will get the results along with the plots. As shown in Figure 5 and 6.

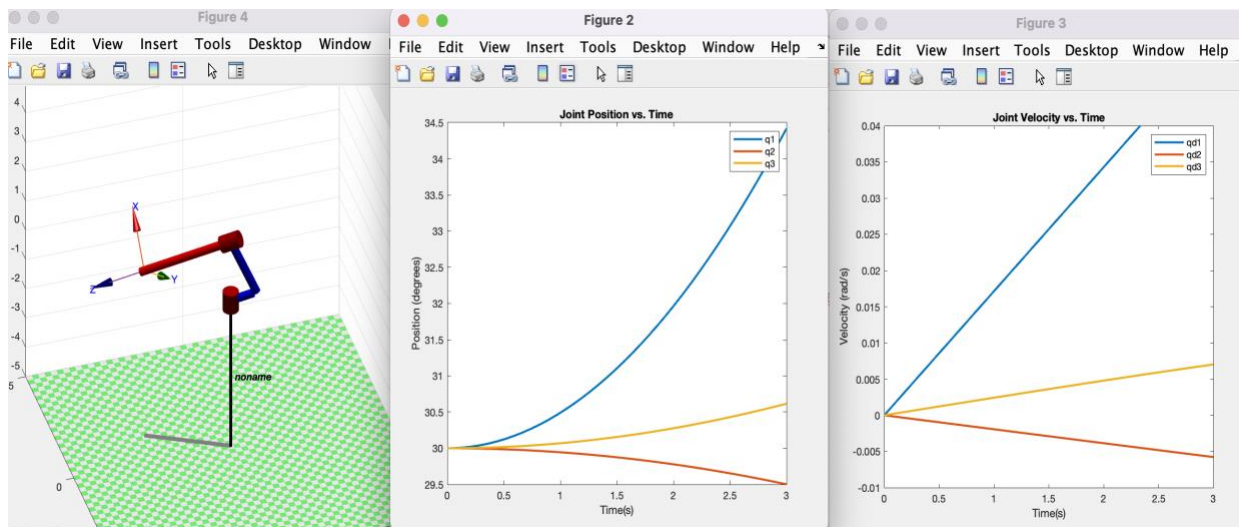


Figure 5

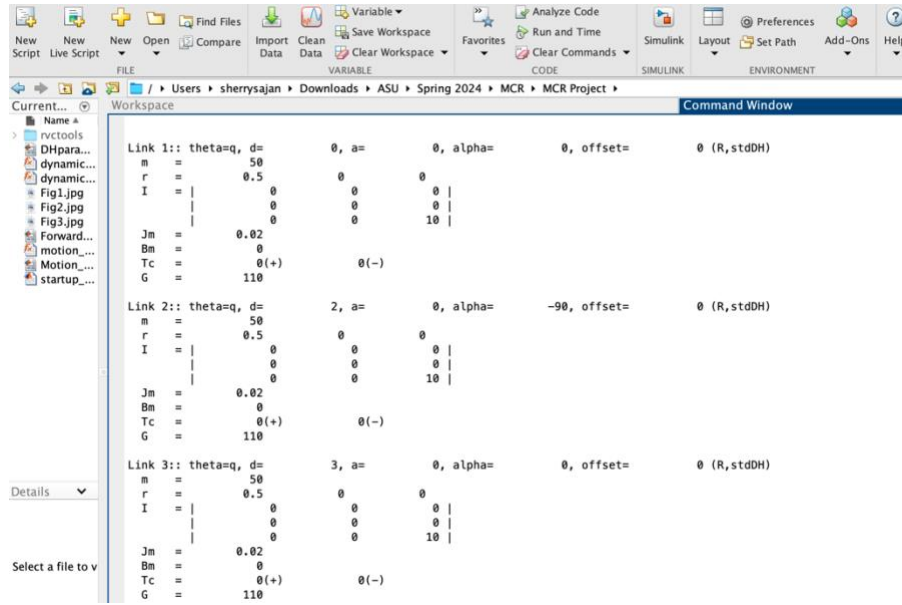


Figure 6

## 4.2. Motion Equations:

**Step 1.** If you chose this option, then a toolbox would appear as shown in Figure 7.

**Step 2.** Enter the values of each column in the same order in the box provided below the table. These columns include Gear ratio, Mass of link and motor, Inertia of Link and motor, Torque value and the type of Joint. Please type '1' (Prismatic) or '2' (Revolute) joints.

**Step 3.** After entering these values click on '+' sign and the values will be automatically entered in the dynamics table. Follow the same procedure for the other 2 links as shown in Figure 7.

**Step 4.** After all values of links are provided, click on 'Generate Motion equation'.

MATLAB App

### Motion Equation

Kr	ml	mm	iL	im	tau	type
0.5000	50.0000	110.0000	10.0000	0.0200	-50.0000	2.0000
0.5000	50.0000	110.0000	10.0000	0.0200	-50.0000	1.0000
0.5000	50.0000	110.0000	10.0000	0.0200	-50.0000	1.0000

Note: To extract motion equation please split your data with space bar  
Type = 1 (Prismatic) and =2 (revolute)

Put the values to extract motion equation: 0.5 50 110 10 0.02 -50 1

**Generate Motion equation**

Equations are:

Figure 7

**Step 5.** After you click on 'Generate Motion equation', you will get the results of torque, as shown in Figure 8.



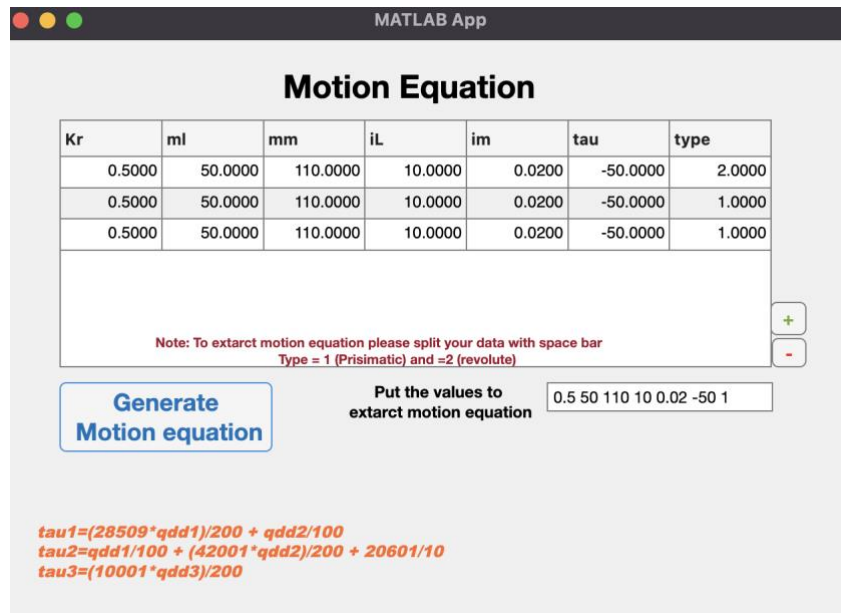


Figure 8

## 4.3. Results:

### 4.3.1. Forward Dynamics with Equations of Motions:

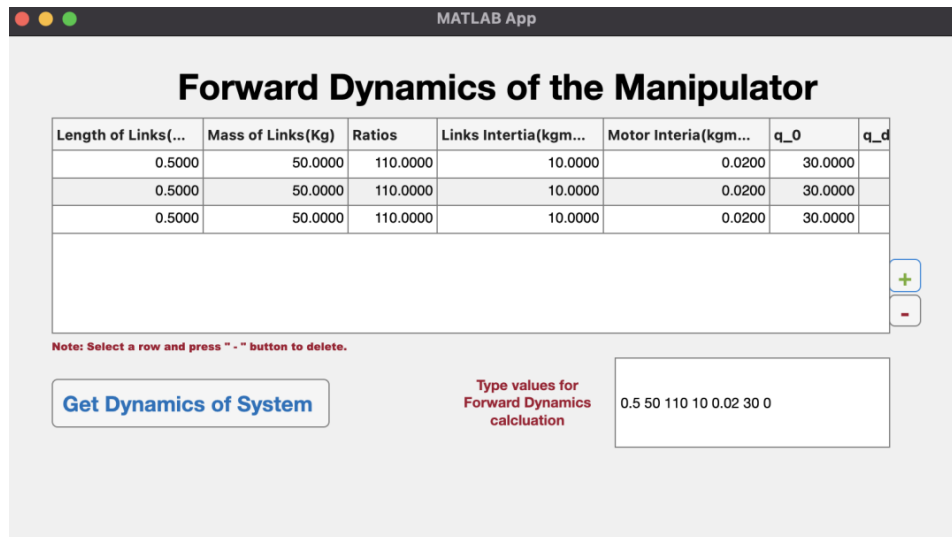


Figure 9

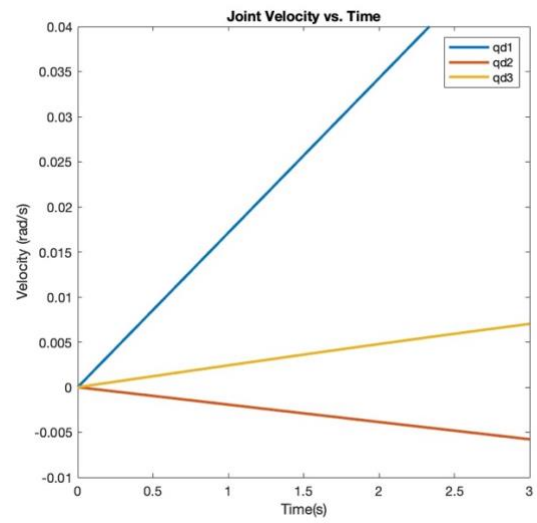
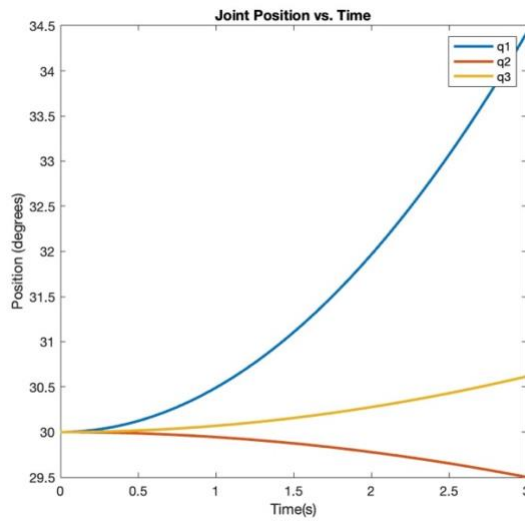
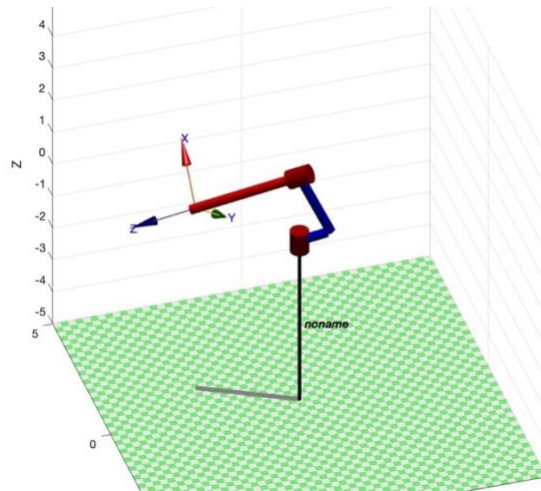


Figure 10

#### 4.3.2. Equations of Motions:

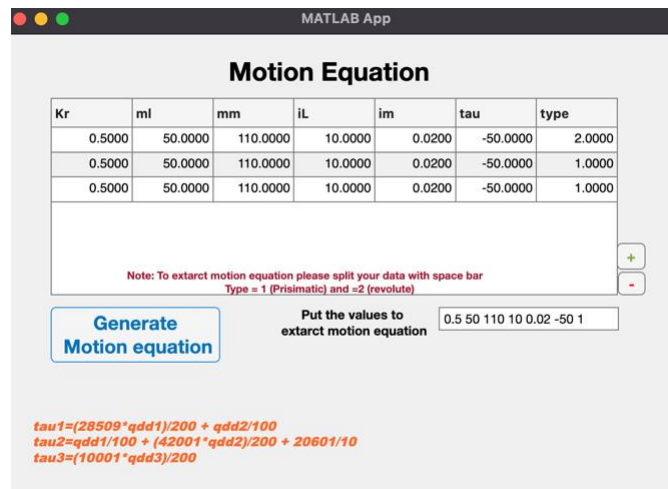


Figure 11

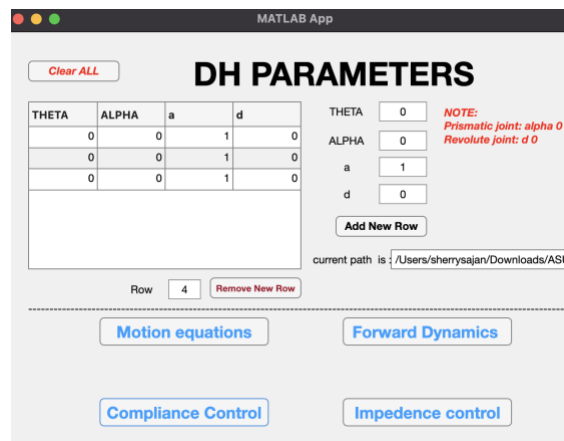
## 5. Indirect Force Controls:

### 5.1. Compliance Control:

Compliance control, on the other hand, focuses on regulating the compliance or flexibility of the robot's motion, often with the aim of ensuring safe and gentle interaction with the environment. Compliance control allows the robot to comply with external forces applied to it, thereby mitigating the risk of damage to the robot or its surroundings and ensuring smooth and natural interaction with humans or objects. This approach is particularly relevant in applications such as collaborative robotics, where robots work alongside humans, or in delicate tasks where precise force control is required, such as surgery or object assembly. Compliance control mechanisms can include force/torque sensors, soft materials, or control algorithms that adjust the robot's motion based on external force feedback.

**Step 1:** To run compliance control first come back to the DH table and select the required DH values and press button “Compliance Control” as shown in figure 12.

**Step 2:** Once the button is clicked it will take you to a screen labeled PD Controls. Once there plug in your values for Length (L), Gear ratio (Kr), mass of link (ml), mass of motor (mm), Inertia of link and motor (iL & im), K\_D, K\_P, friction, and finally desired position of joints (xd1,xd2,xd3).



The MATLAB App interface for DH PARAMETERS. It features a table for joint parameters (THETA, ALPHA, a, d) with three rows of zeros. To the right, individual input fields for THETA, ALPHA, a, and d are shown, along with a note: "NOTE: Prismatic joint: alpha 0 Revolute joint: d 0". Below the table is an "Add New Row" button and a "current path" field. At the bottom, there are four buttons: "Motion equations", "Forward Dynamics", "Compliance Control", and "Impedance control".

THETA	ALPHA	a	d
0	0	1	0
0	0	1	0
0	0	1	0

THETA: 0  
ALPHA: 0  
a: 1  
d: 0

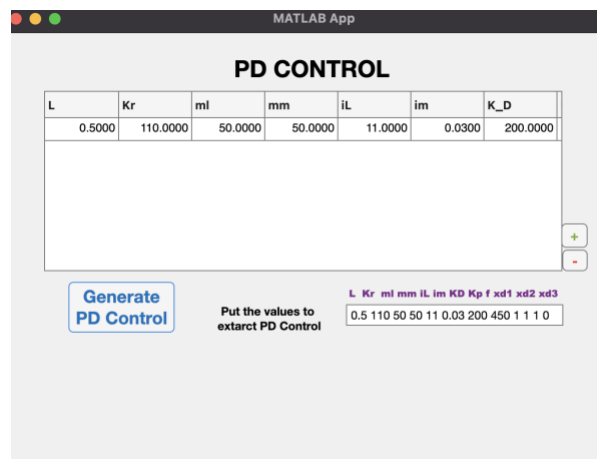
NOTE: Prismatic joint: alpha 0  
Revolute joint: d 0

current path is: /Users/sherrysajan/Downloads/ASL

Row: 4

Buttons: Motion equations, Forward Dynamics, Compliance Control, Impedance control

Figure 12



The MATLAB App interface for PD CONTROL. It features a table for joint parameters (L, Kr, ml, mm, iL, im, K\_D) with one row of values. Below the table is a "Generate PD Control" button. At the bottom, there is a text box for "Put the values to extract PD Control" and a display area showing the extracted values.

L	Kr	ml	mm	iL	im	K_D
0.5000	110.0000	50.0000	50.0000	11.0000	0.0300	200.0000

Buttons: Generate PD Control

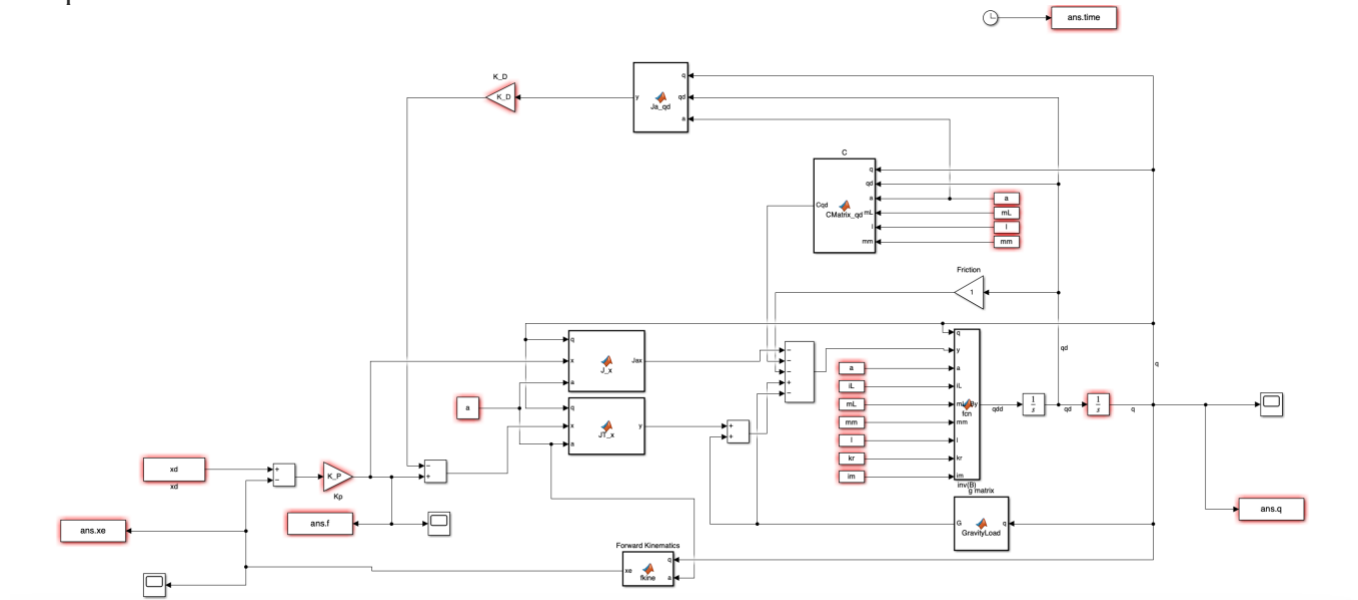
Put the values to extract PD Control

Display: L Kr ml mm iL im KD Kp f xd1 xd2 xd3  
0.5 110 50 50 11 0.03 200 450 1 1 1 0

Figure 13

**Step 3:** Once you have your values input click “Generate PD Control”. It will return 4 plots, 2 of them being the x, y positions of the actual and desired end effector as a function of time, and the other 2 plots as contact force as a function of time as shown in Figure 14.

**Simulink Block Diagram of Compliance Control:** PD control with gravity compensation under interactive forces with the environment



### 5.1.1. Results:

Indirect force control through compliance control: PD control with gravity compensation under interactive forces with the environment. Below are the results of plots for Desired Vs Actual End Effector Position and End-effector contact forces as a function of time:

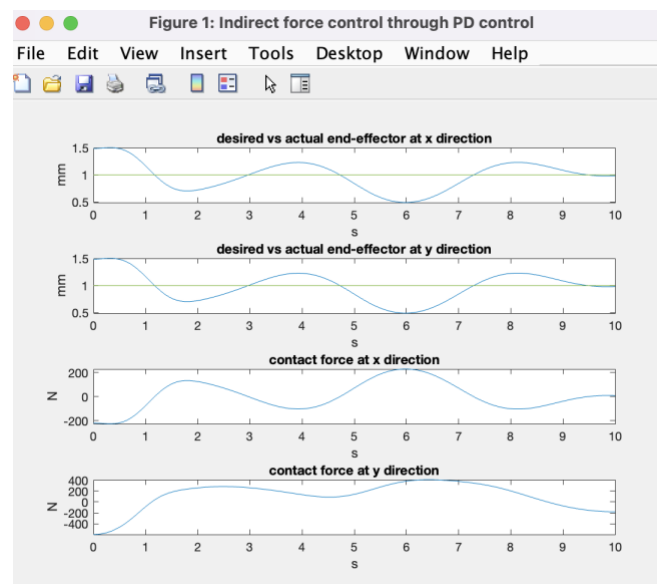


Figure 14

## 5.2. Impedance Control:

Impedance control refers to a control approach where the robot's behavior is modeled as an impedance, analogous to a mechanical spring-damper system. This allows the robot's response to external forces to be regulated by adjusting its impedance parameters, such as stiffness and damping coefficients. By dynamically adjusting these parameters, impedance control enables robots to adapt their behavior to varying environmental conditions and task requirements, providing robustness and flexibility in tasks like grasping, manipulation, and assembly.

**Step 1:** To run Impedance control again come back to the DH table and select the required DH values and press button “Impedance Control” as shown in figure 15.

**Step 2:** Once the button is clicked it will take you to a screen labeled Impedance Controls. Once there, plug in your values for Desired position of joints ( $x_{d1}, x_{d2}$ ), KD, KP, Friction, Inertia of link ( $IL1, IL2$ ), mass of link ( $mL1, mL2$ ), Mass of motors ( $mm1, mm2$ ), Gear ratios ( $Kr1, Kr2$ ), Inertia of motor ( $Im1, Im2$ ), Joint Positions ( $q1, q2$ ), velocity of joints ( $\dot{x}_{d1}, \dot{x}_{d2}$ ), acceleration ( $\ddot{x}_{d1}, \ddot{x}_{d2}$ ), Equilibrium position ( $x_{r1}, x_{r2}$ ), stiffness value and  $m_{dx}$  and  $m_{dy}$  (Positive definite matrix) as shown in Figure 16.

**DH PARAMETERS**

THETA	ALPHA	a	d
0	0	1	0
0	0	1	0

THETA: 0  
ALPHA: 0  
a: 1  
d: 0

NOTE: Prismatic joint: alpha 0  
Revolute joint: d 0

Add New Row

current path is: /Users/sherrysajan/Downloads/ASL

Row: 3 Remove New Row

Motion equations Forward Dynamics

Compliance Control Impedance control

Figure 15

**IMPEDANCE CONTROL**

xd1	xd2	KD	KP	friction	IL1	IL2
1	0	10	10	1	1	1

Put the values to extract Impedance control: 1 0 10 10 1 1 1 1 1 1 1 1 1 1 0.5 0.5 1 0 1 0 1 0 100 1 1

Generate Impedance Control

Figure 16

**Step 3:** Once you have your values input click “Generate Impedance Control”. It will return 2 plots of them being of the desired and actual end effector positions in the x, y direction as a function of time, and remaining 2 as the contact force as a function of time as shown in Figure 17.

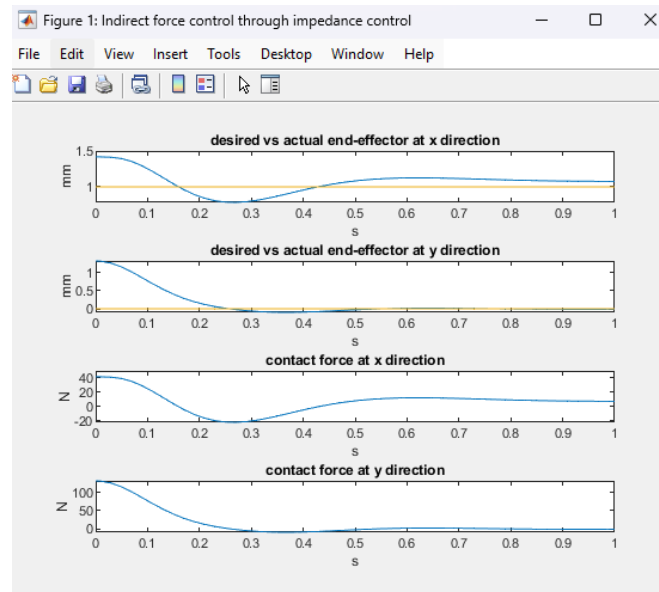
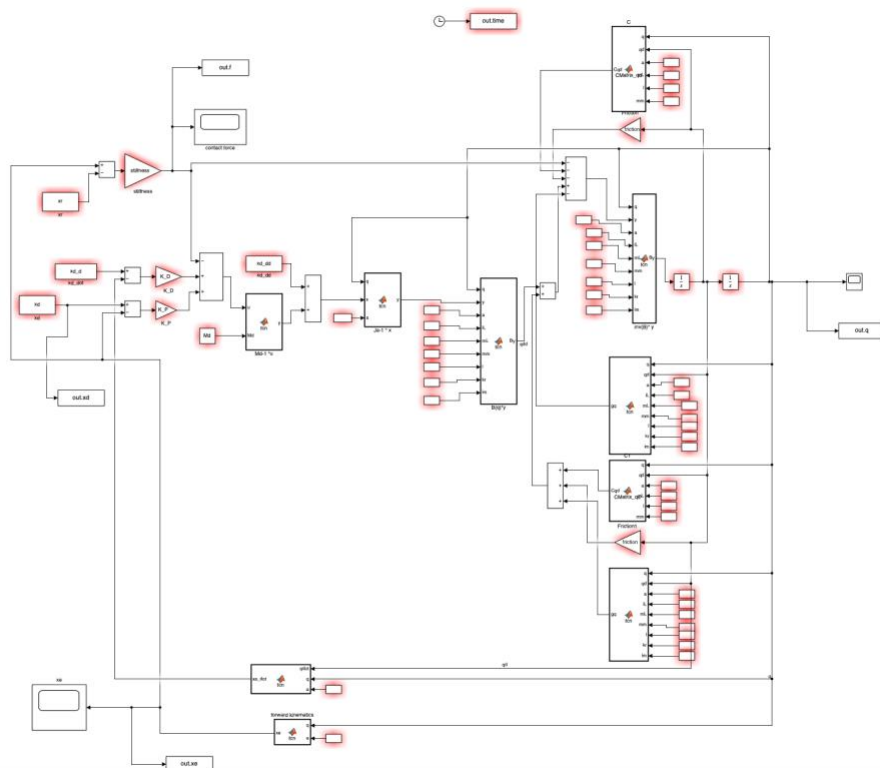


Figure 17

**Simulink Block Diagram of Impedance Control:** inverse dynamics control under interactive forces with the environment.



## 6. Contributions:

Member	Contribution
Amarnath Periyakaruppan	Compliance Control Simulink, Impedance Control Simulink
Anirudh Iyengar	Front end, app development, graphics
Kyle Welsh	Dynamics Equations, Functions, Compliance Control, Impedance Control, Report
Rohit Sanjay Ganesh	Forward Kinematics, Dynamics Equations, Compliance Control, Impedance Control
Sherry Daniel Sajan	Report, Compliance Control, Simulink, Impedance Control

## 7. References:

- Aramani3. "MAE-547-Final-Project/MAE 547 Final Project Report.pdf at Master · Aramani3/MAE-547-Final-Project." *GitHub*, 2020, [github.com/aramani3/MAE-547-Final-Project/blob/master/MAE%20547%20final%20project%20report.pdf](https://github.com/aramani3/MAE-547-Final-Project/blob/master/MAE%20547%20final%20project%20report.pdf). Accessed 3 May 2024.
- Ravendra275. "MAE-547-PR/the GUI at Master · Ravendra275/MAE-547-PR." *GitHub*, 2019, [github.com/ravendra275/MAE-547-PR/tree/master/The%20GUI](https://github.com/ravendra275/MAE-547-PR/tree/master/The%20GUI). Accessed 3 May 2024.
- Ni, Hsiao-Ping. "Ping830616/MAE\_547." *GitHub*, 3 Nov. 2021, [github.com/ping830616/MAE\\_547](https://github.com/ping830616/MAE_547). Accessed 3 May 2024.