Modeling and Control of Robots

(MAE 547)

Final Project Report

Fall ‘19

Dr. Hamidreza Marvi

By
Eshan Gaur (1217195785)
Prabal Bijoy Dutta (1217370453)
Shreya Dama (1217526427)
Shreya Reddy (1217394503)
Sriram Mudumbai Rangarajan (1217410051)

Table of Contents

Table of Figures

Introduction

The aim of this project is to create a robotics package on MATLAB R2019a. The user inputs information like the number of joints of their robot, the number of links, and the DH parameters into the GUI that has been created. With the use of Object Oriented Programming, the required outputs have been calculated. Instructions on how to run the code and the GUI have been included.

Furthermore, the following two assumptions are applied in the calculations - All of the joints are either revolute or prismatic and all of the links are straight.

Description of the Project

(a) Homogeneous Matrix Transformation (Eshan Gaur and Prabal)

The main advantage of Homogeneous transformations is the compact notation that exists within the position vectors and the rotation matrix combined. In these homogeneous matrices, a vector i_r represented in relation to the i^{th} coordinate frame can be expressed in relation to the j^{th} coordinate frame given that the orientation and the position relative to the i^{th} frame are in relation to the j^{th} frame.

The transformation T_i for each i such that $1 < i <= m$, is

$$T_i = Q_{i-1}R_i = \begin{pmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i \cos\alpha_{i-1} & \cos\theta_i \cos\alpha_{i-1} & -\sin\alpha_{i-1} & -\sin\alpha_{i-1}d_i \\ \sin\theta_i \sin\alpha_{i-1} & \cos\theta_i \sin\alpha_{i-1} & \cos\alpha_{i-1} & \cos\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

where θ and ɑ are the standard Denavit - Hartenburg parameters.

In our project, we have written a function that takes *p_tab* as an input. *p_tab* is a D-H parameter table. It then calculates all the A_i that are necessary to calculate the Homogeneous Transformation matrix, *T*. It then returns these values of A_i.

(b) Euler Angles (Shreya Dama and Sriram Mudumbai Rangarajan)

We know that three parameters are enough for us to work on the orientation of a manipulator in space. Hence, this minimal representation of the of only three parameters can be described as $\phi=[\varphi\ \nu\ \boldsymbol{\phi}]^T$. There are two types of Euler angles, *ZYZ* and *RPY.* We have included the functionality for both in our project.

In this project, we have created a function *EA* that takes three inputs, first one being whether the given frame is Fixed or Current, second being whether the user has entered angles or the rotation matrix and then the last being the values of said angles or the rotation matrix. The function then calculates the rotation matrix depending on the various choices made by the user and returns that value of rotation matrix.

(c) Forward Kinematics (Shreya Dama and Shreya Reddy)

Forward Kinematics of the manipulator here includes finding the DH parameters of the manipulator, finding the transformation matrices and finally making the robotic arm move using this information.

In this project, we have created a DH function that has the input as DH parameters, namely, *[a ,alpha, d, theta, n]* and uses those values to calculate the rotation and the Transformation matrix for the given values.These values are then returned.

(d) Workspace (Shreya Dama and Sriram Mudumbai Rangarajan)

Workspace, in its essential definition, enables us to find the three dimensional work space of a particular manipulator. This helps us to see the limits of the manipulator and which points in a three dimensional grid it can reach.

We have implemented a function “workspace” that takes as input the name of the robot and the D-H parameters *[a ,alpha, d, theta, n]* . This function then calculates the 3D operational workspace of the manipulator.

(e) Inverse Kinematics (Shreya Dama and Sriram Mudumbai Rangarajan)

In order to solve the inverse kinematics of the manipulator, a function called *inverseKinematics* was created. This function inputs the number of links, and the DH parameters. Using the Serial function from the robotics toolbox, a robot object is created. The functions fkine and ikunc are then used to obtain the forwards and inverse kinematics respectively. The joint parameters are obtained from the given end effector pose.

(f) Differential Kinematics (Prabal)

Differential kinematics give the relations between the joint velocities and the corresponding end-effector linear and angular velocity. The transformation matrix is obtained using function *HTM_i.* The function *Compute_jacobian* is used to obtain the jacobian matrix.
A robot singularity is a configuration in which the robot end-effector becomes blocked in certain directions. *Compute_singularity_1* is used to find the singularities of the robot manipulator.

(g) Inverse differential kinematics and inverse kinematics using Jacobians (Prabal, Shreya Dama and Sriram Mudumbai Rangarajan)

With Inverse Differential Kinematics, we are required to find the joint velocity vector that realizes a desired end-effector “generalized” velocity (both linear and angular). In our project, we input the joint coordinates, the rotation matrix, and time into the function *IDK*. The jacobian matrix is subsequently calculated and the joint velocity vector is obtained.

(h) Manipulator Dynamics (Shreya Dama, Prabal and Shreya Reddy)

In the Manipulator Dynamics section of the project, multiple inputs are required, hence the command *varargin* (Variable Argument Input) is used. The position, orientations, mass, and inertias of the link and motors need to be input, along with the gear ratio of the motors.

Correspondingly, the jacobian matrices of each of these parameters is found and the equations of motion are generated. Consequently, the christoffel symbols are calculated.

The inputs are allowed to be taken in both string form and in numerical values using the functions *str_process* and *str_process_22.* It has been tested for a 2-link robot with link properties RR,PR, and PR, with the *Robot_5* function as the test robot.

(i) Motion Control (Prabal, Sriram Mudumbai Rangarajan and Eshan Gaur)

One of the key foundations when it comes to industrial automation is Motion control. Here, the key issue and a point of concern is the supposed path of the tool and the calculated motion of the industrial robotic arm. For us to control the automated process, it is of high importance that the correct position of the object is observed. The next step in the implementation of the motion control system is the feedback comparison of the correct position and the target.

Instructions on usage of GUI and the Code

The toolbox that we have completed has been made possible using Matlab 2019a and the Corke toolbox. You have to be using Matlab 19 and the *startup_rvc.m* has to be in the same folder as the project code.

NOTE: Open all the GUI apps in using the open feature in MATLAB 2019a. Otherwise it won't work.

(1) Sample GUI Input to compute Jacobian,Singularity, and Homogeneous Transformation Matrix

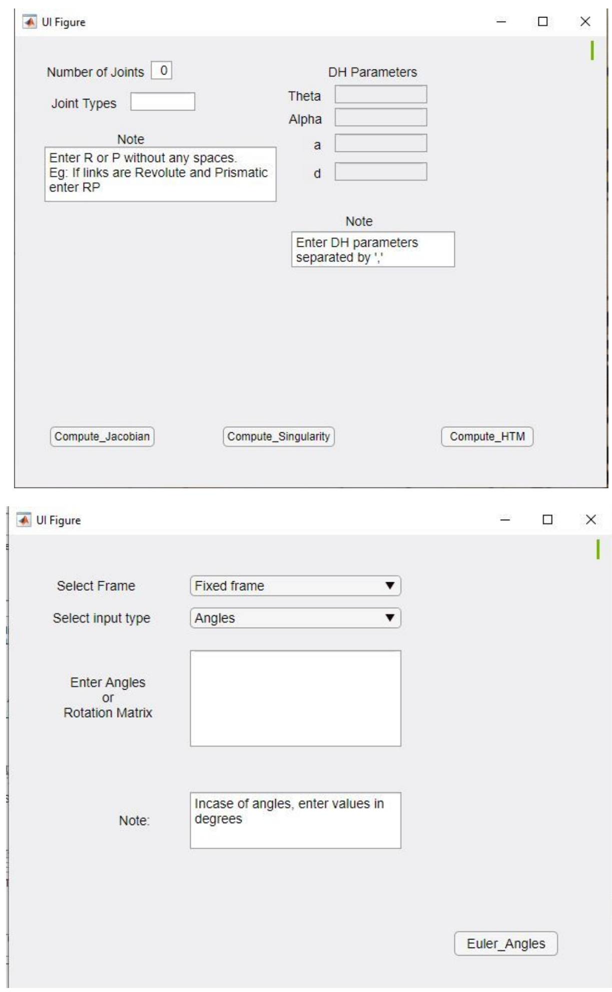


Fig 1. GUI interfaces

There are 2 GUI interfaces. The first one has the ability to calculate the Homogeneous Transformation Matrix, Singularities and Jacobian matrix. Inputs to this GUI are the number of joints, types of joints, and their DH parameters.

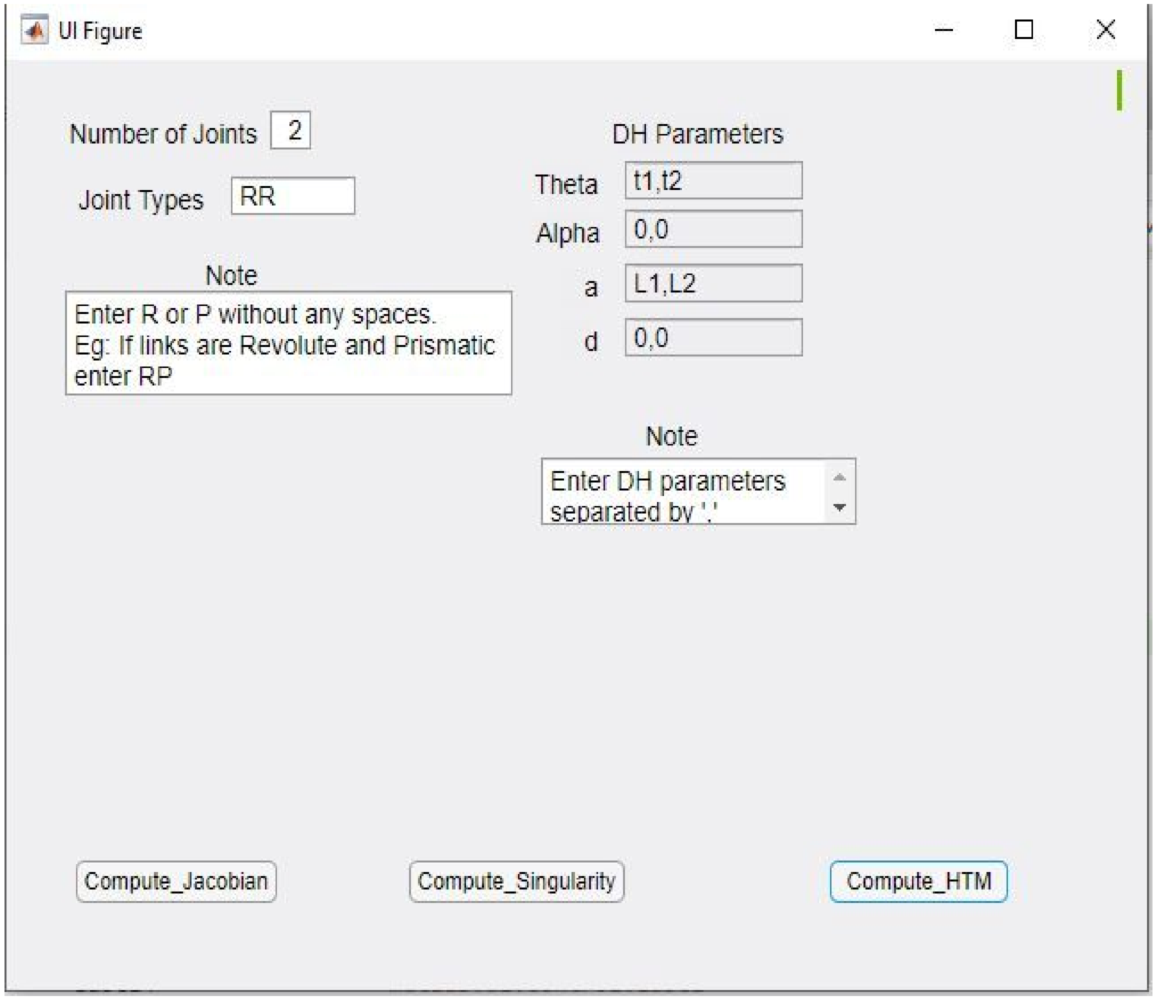
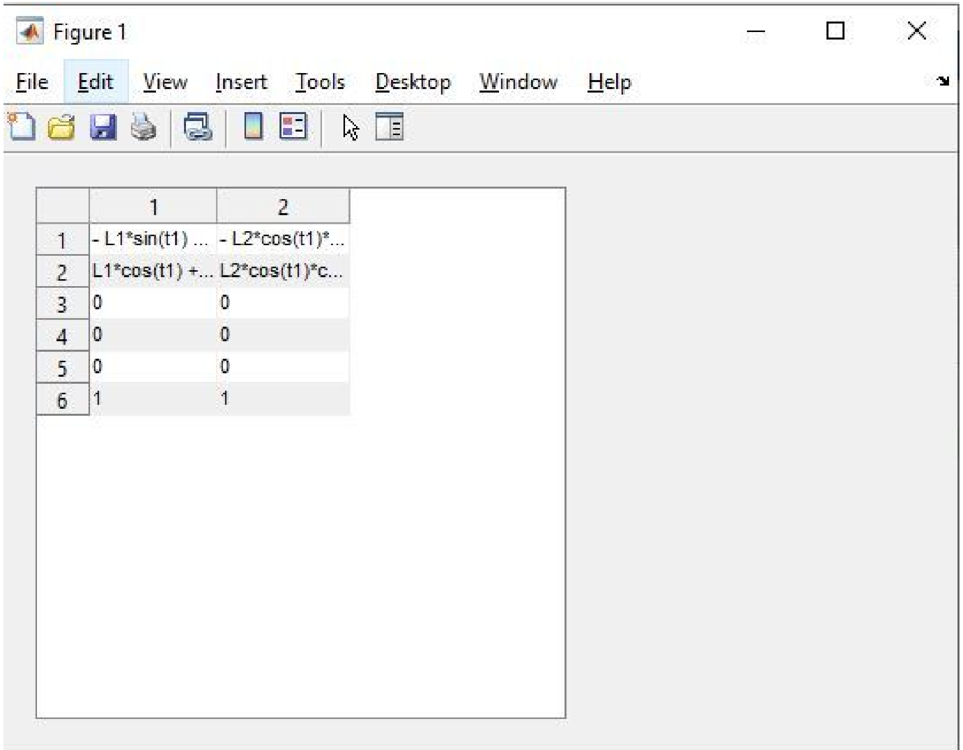


Fig 2. Sample GUI interface and the Values

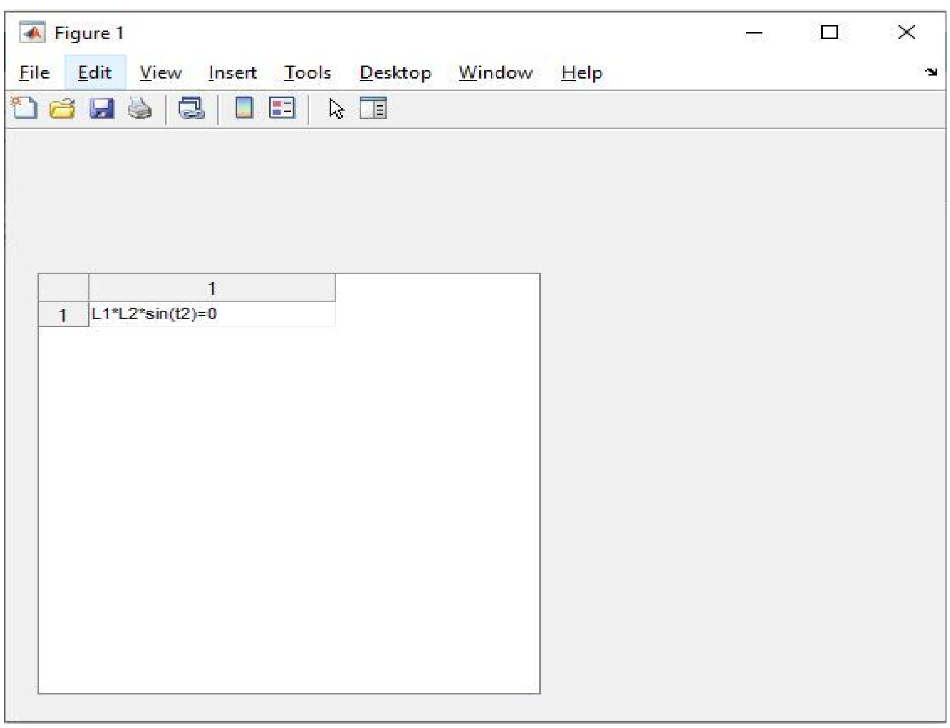
Steps to run the GUI:

1. Open /App/Jac_Sing_htm_2. This is the GUI interface.
2. Enter the number of Joints. (Integer only)
3. Enter the Joint Types. They need to be either R - Revolute or P - Prismatic. For multiple joints, use like “RRP”.
4. Enter the values of the DH parameters. The values should be separated by a comma (,).



	1	2
1	- L1*sin(t1) ...	- L2*cos(t1)*...
2	L1*cos(t1) +...	L2*cos(t1)*c...
3	0	0
4	0	0
5	0	0
6	1	1

Fig 3. Sample values of the Jacobian Matrix.



	1
1	L1*L2*sin(t2)=0

Fig 4. Sample values of the Singularity

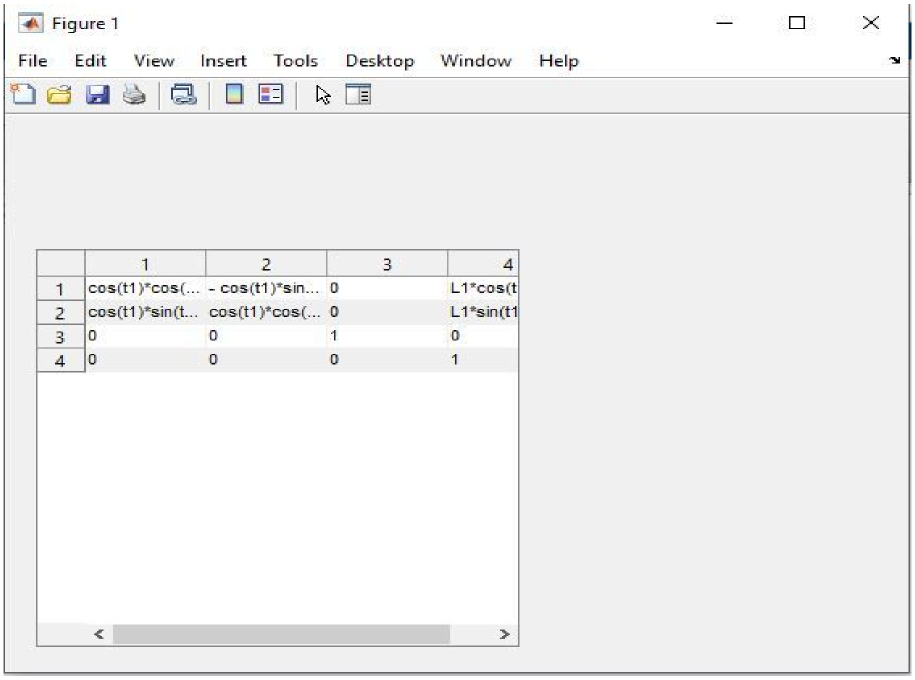


Fig 5. Sample values of the Homogeneous Matrix

(2) Sample GUI for Euler Angles (Apps/GUI_EA)

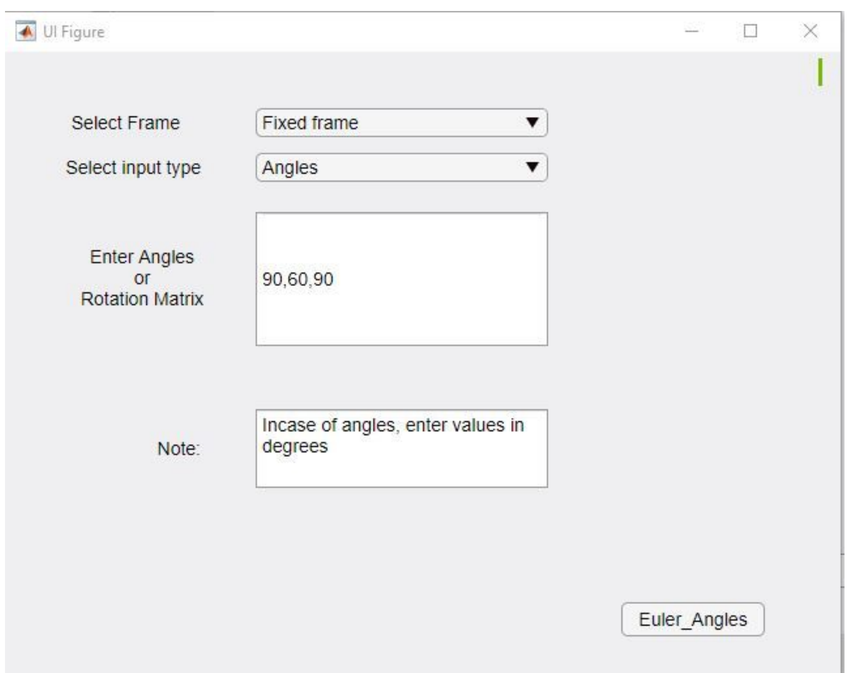
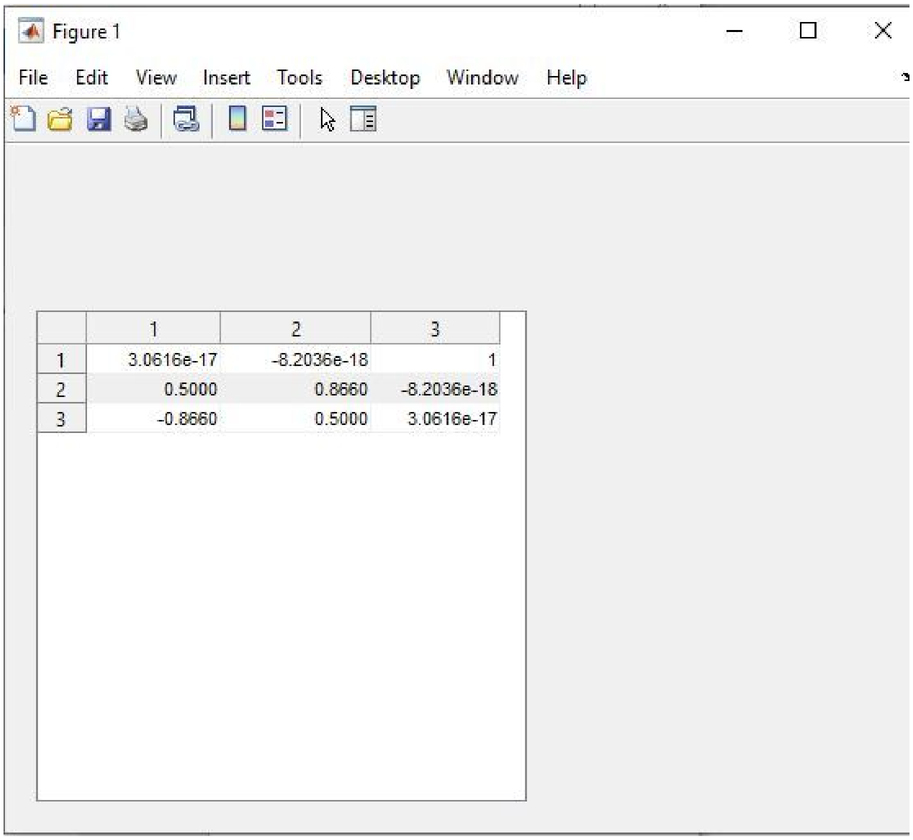


Fig 6. Sample Values for the Euler Angles



	1	2	3
1	3.0616e-17	-8.2036e-18	1
2	0.5000	0.8660	-8.2036e-18
3	-0.8660	0.5000	3.0616e-17

Fig 7 : Sample Euler Angle values

(3) GUI calculates the Forward Kinematics and Workspace of the manipulator(App/GUI_Forwardkinematics_and_workspace)

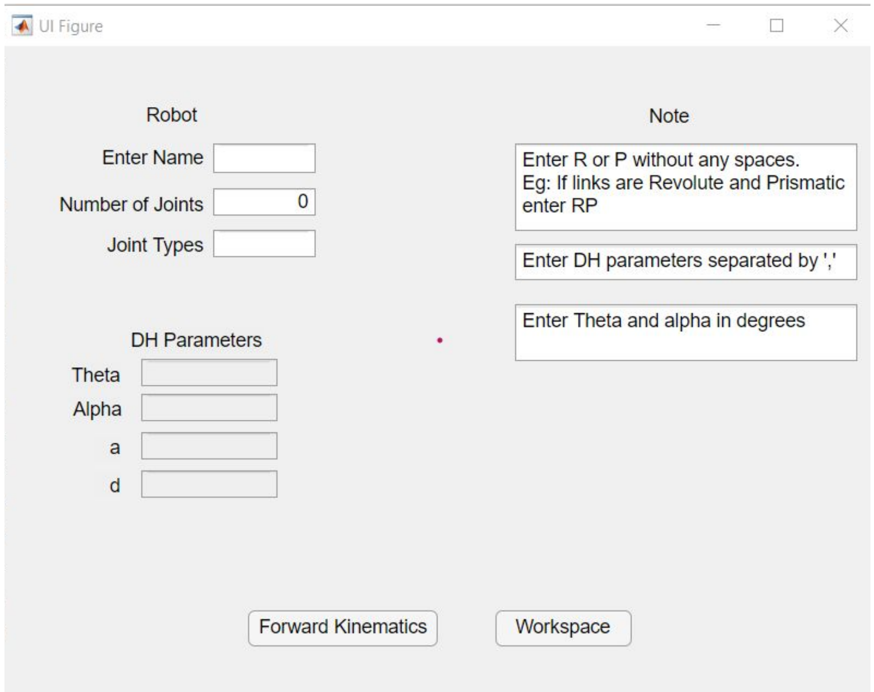


Fig 8 : GUI for forward kinematics

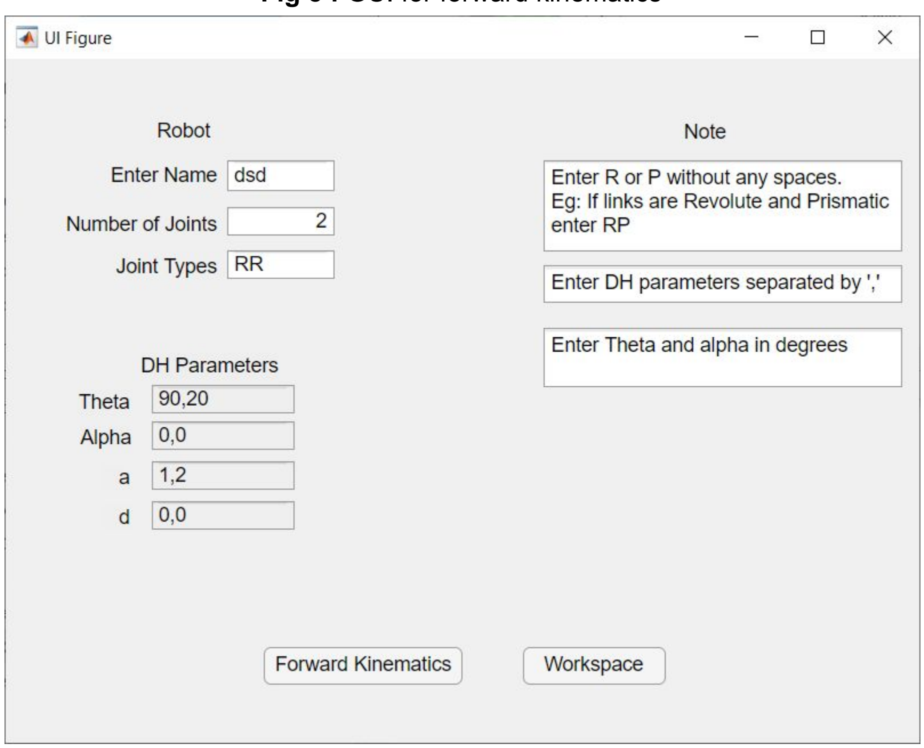


Fig 9 : GUI for forward kinematics with values input

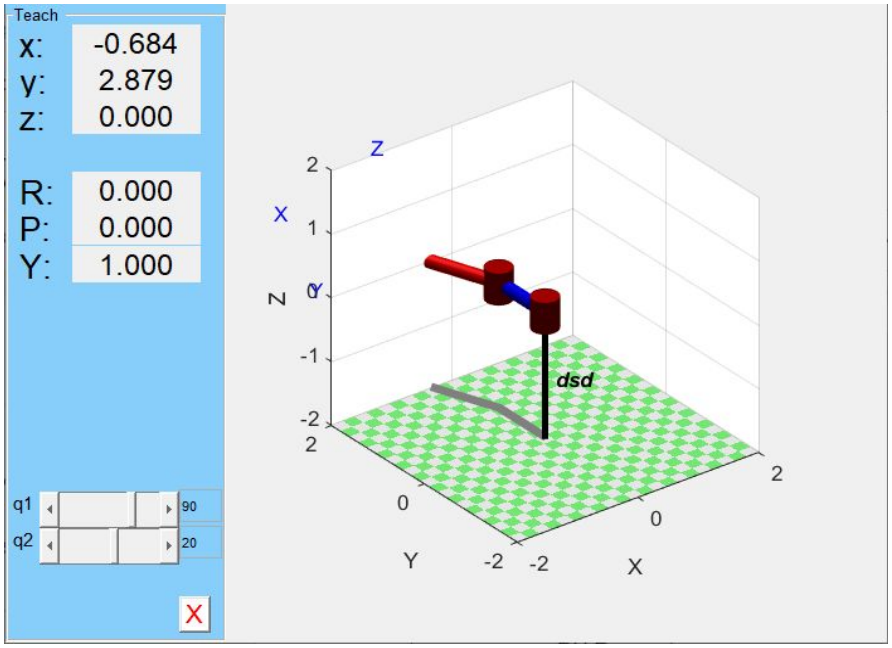
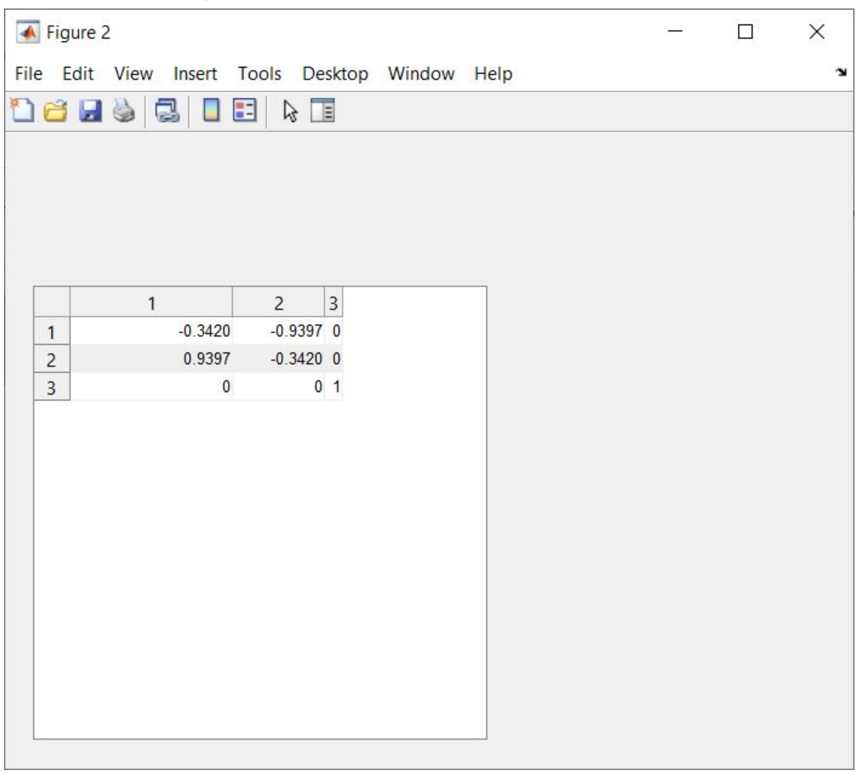


Fig 10 : Workspace for forward kinematics



	1	2	3
1	-0.3420	-0.9397	0
2	0.9397	-0.3420	0
3	0	0	1

Fig 11: Transformation matrix for Forward Kinematics

(4) GUI for Inverse Differential Kinematics (App/IDK_GUI)

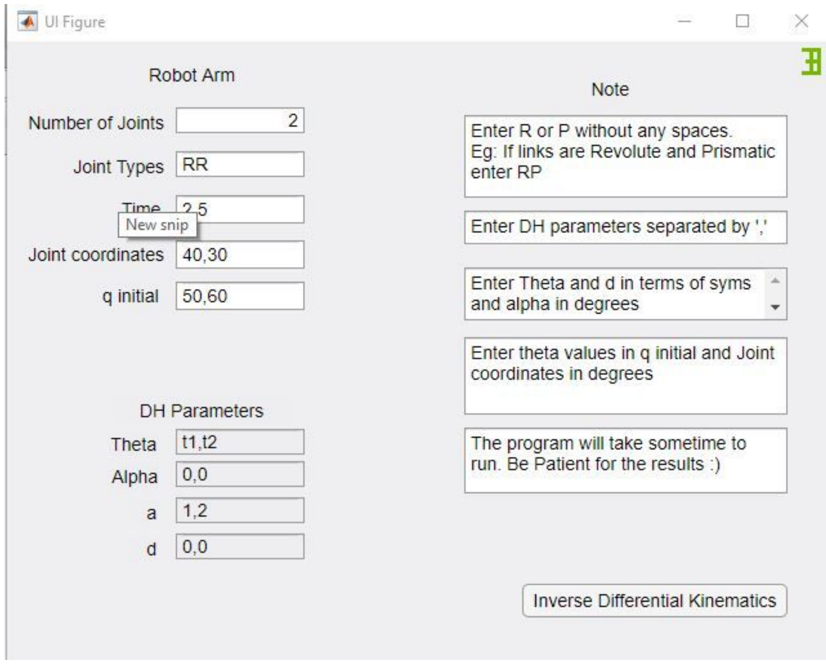


Fig 12 : GUI with sample values for Inverse Differential Kinematics

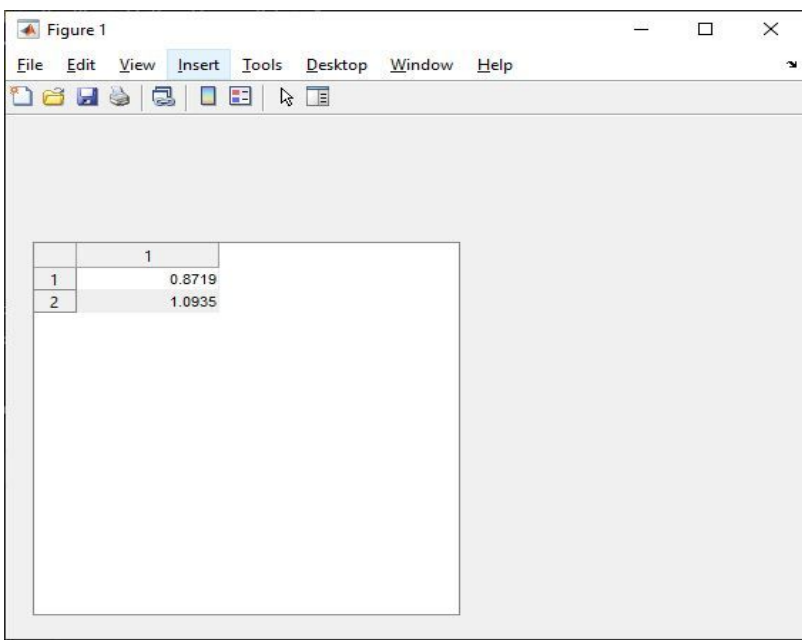


Fig 13 : GUI sample values for Inverse Differential Kinematics

(5) Sample Test for SCARA manipulator

We have included sample test cases for the SCARA manipulator

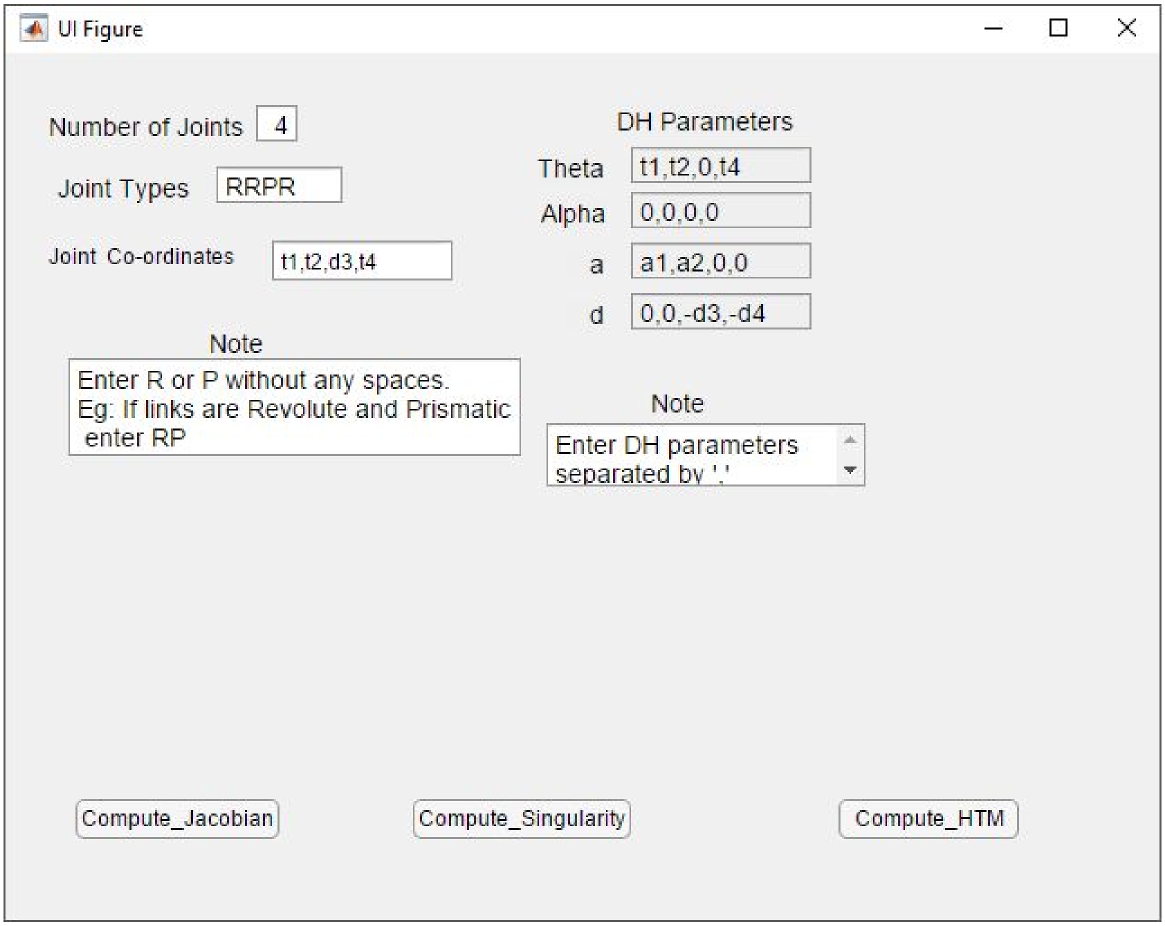


Fig 14 : Test case for SCARA manipulator

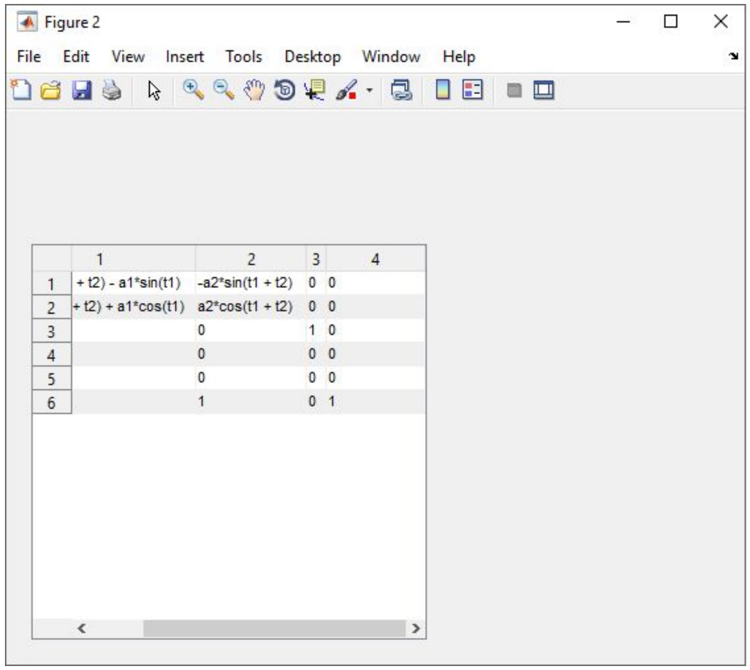


Fig 15 : Jacobian for SCARA

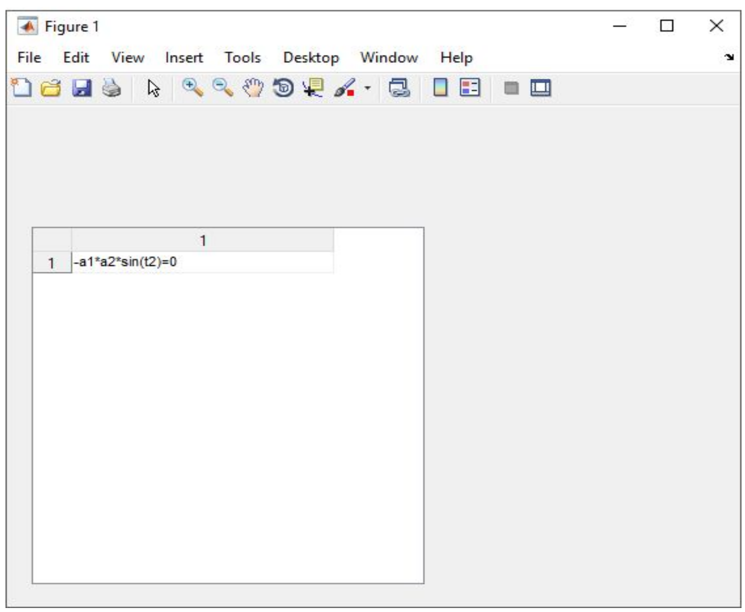


Fig 16: Singularity for SCARA

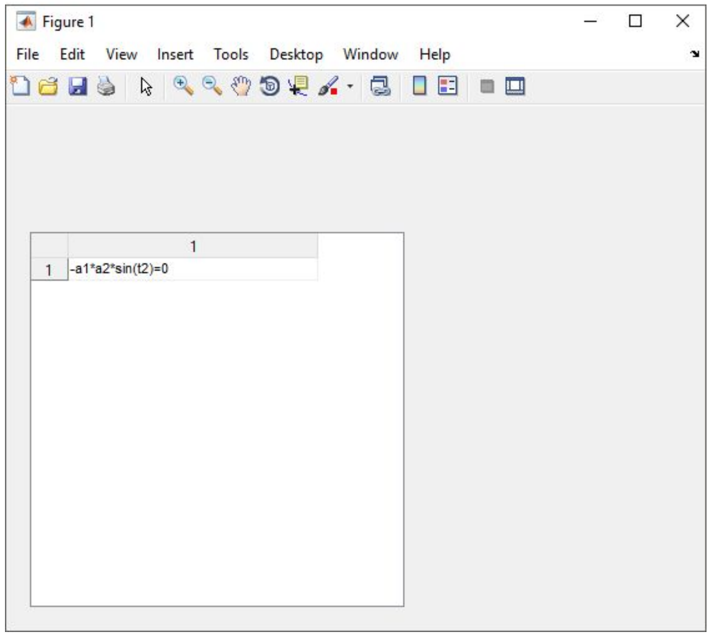


Fig 17 : Homogeneous Transformation Matrix for SCARA

(6) Operational Space Control of 2-link Planar Arm

Instruction to use the code and Simulink model:

1. Make the Folder Operational_Space_Control, the address of the MATLAB file directory.

2. Run the file i8_15.m

3. Run the Simulink File Operational_SpacePD.slx

4. Run the plot p8_15.m

The operational space control involves control scheme where the motion is specified in terms of operational space variables, and the measured joint space variables are converted into the operational space via use of direct kinematics.

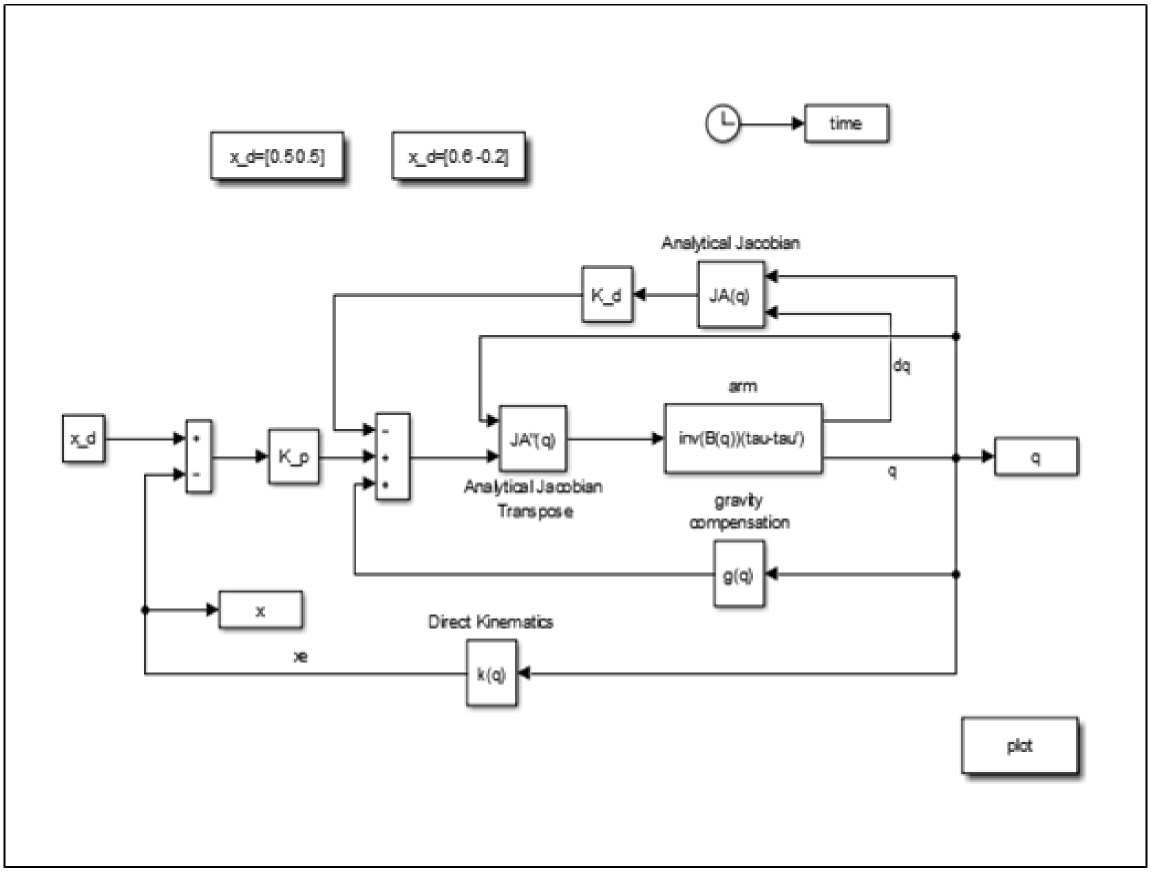


Fig 18 : Simulink Block Diagram for Operational Space Control

A 2-link planar arm, has two rotating components($\theta 1, \theta 2$) as joint variables. The purpose of the control law in the operational space is to perform a non-linear compensating action of joint space gravitational force and an operational space linear PD Control action [1]

The control law: $u = g(q) + J_A^T(q)K_P(x_d - x_e) - J_A^T(q)K_d \widehat{x_e}\, J_A(q)\dot{q}$ (1)

Is chosen so that the operational space error asymptotically tends to zero.

For the purpose of implementing the Operational Space Control, few blocks were designed, namely, the direct kinematic block and the Analytical Jacobian Block.

Direct Kinematic Block:

The direct kinematic block performs the forward kinematics returning the end-effector position for a given joint-coordinate values. The output of the direct kinematic block is used to compute the error from the desired position which is further used in developing the control law.

For a 2-link planar arm:

$$k(q) = \frac{a_1 cos(q_1) + acos(q_1 + q_2)}{a_1 sin(q_1) + a_2 sin(q_1 + q_2)}$$

(2)

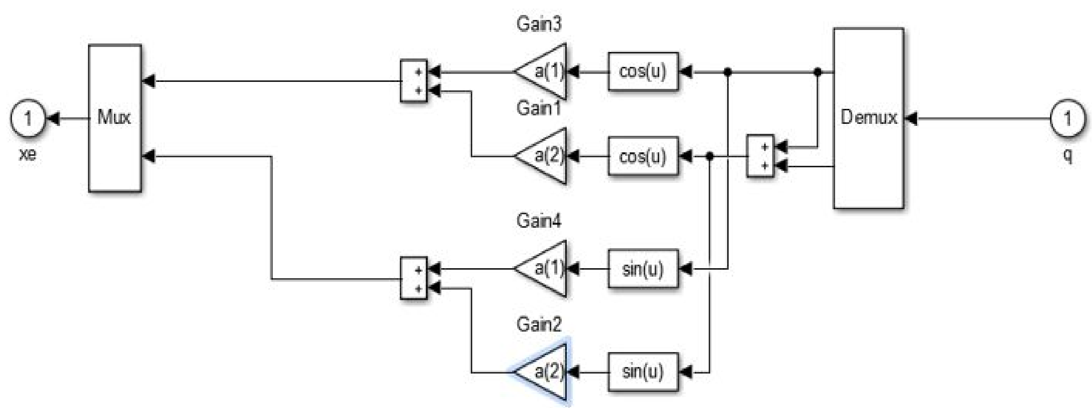


Fig 19 : Direct Kinematic Block

The above block diagram is the representative of the direct kinematic of the 2 planar arm.

Analytical Jacobian Block:

The analytical Jacobian is used to compute using the relation

$$\dot{x} = -J_A(q)$$

and to convert the operational space variables back to joint coordinate variables to perform the non-linear gravitational forces compensating actions.

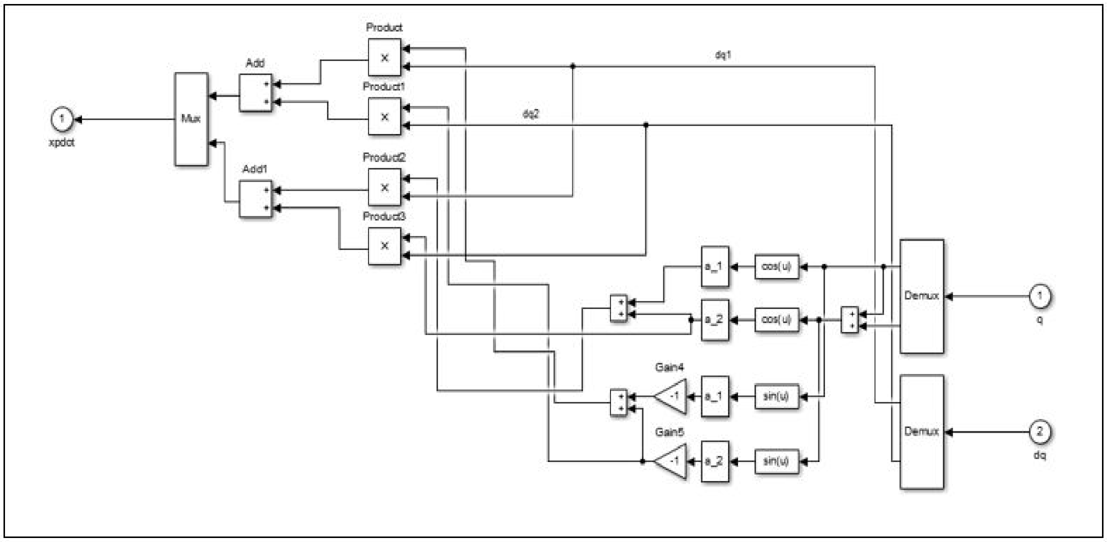


Fig 20 : Analytical Jacobian Block

The analytical Jacobian for a 2-planar arm is defined as

$$J_A(q) = \begin{bmatrix} -a_1 sin(q_1) - a_2 sin(q_1 + q_2) & -a_2 sin(q_1 + q_2) \\ a_1 cos(q_1) + acos(q_1 + q_2) & acos(q_1 + q_2) \end{bmatrix}$$

(4)

Assumptions in the modeling:

1. The initial state of the joint coordinate is taken as [0 0], also the derivative of the joint coordinate is taken as [0 0];

2. The value of K_p and K_d are respectively

K_p = 3250*eye(2);

K_d = 750*eye(2);

3. The desired positions are taken as

i) x_d = [0.5 0.5];

ii) x_d = [0.6 -0.2];

3.1 xd = [0.5 0.5]; and initial condition q = [0 0];

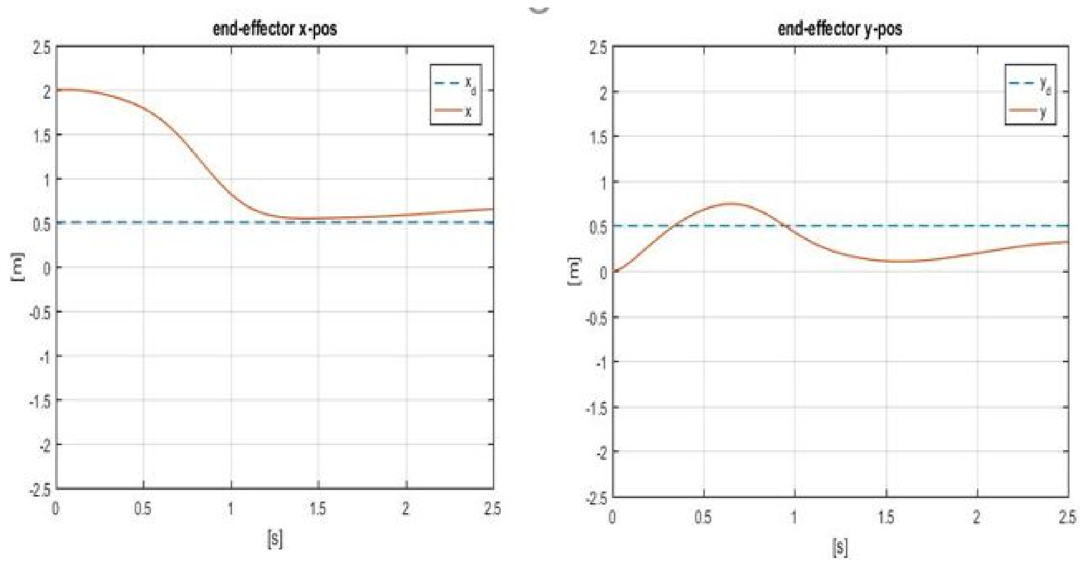


Fig 21 : End Effector X and Y Pose

3.2 xd = [0.6 -0.2]; and initial condition q = [0 0];

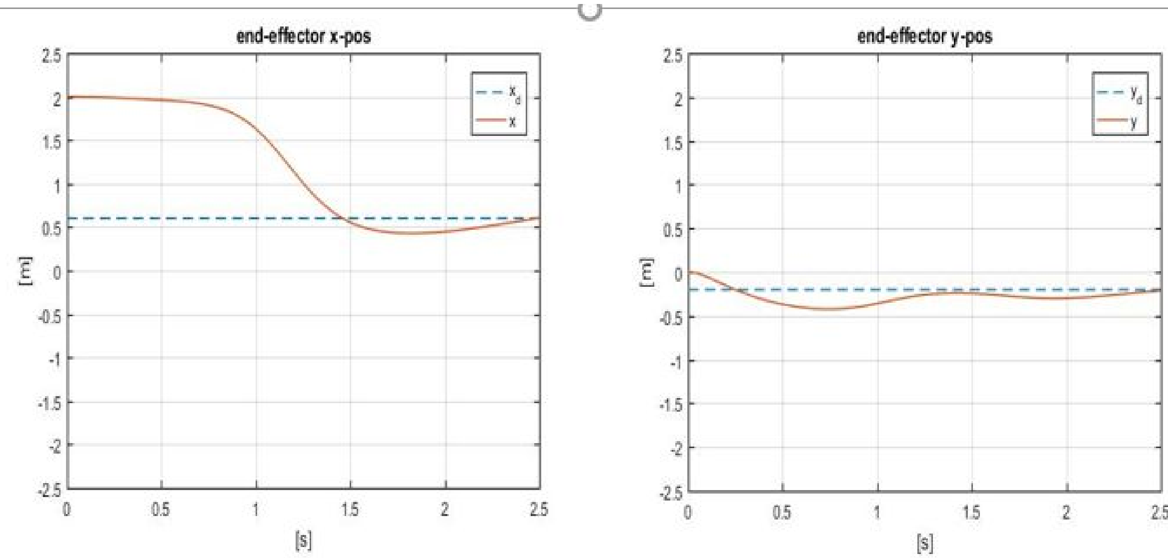


Fig 22 : End Effector X and Y Pose

Contributions of the Team

Below specified, are the contributions of each of the team members along with the specified work that they have accomplished.

A. Homogeneous Transformation Matrix : Eshan Gaur and Prabal Bijoy Dutta
B. Euler Angles : Shreya Dama and Sriram Mudumbai Rangarajan
C. Forward Kinematics : Shreya Dama and Shreya Reddy
D. Workspace : Shreya Dama and Sriram Mudumbai Rangarajan
E. Inverse Kinematics : Shreya Dama and Sriram Mudumbai Rangarajan
F. Differential Kinematics : Prabal Bijoy Dutta
G. Inverse Differential Kinematics using Jacobians : Prabal Bijoy Dutta, Shreya Dama and Sriram Mudumbai Rangarajan
H. Manipulator Dynamics : Shreya Dama, Sriram Mudumbai Rangarajan, Prabal and Shreya Reddy
I. Motion Control : Prabal Bijoy Dutta and Eshan Gaur
J. Graphic User Interface (GUI) : Prabal Bijoy Dutta, Shreya Dama and Sriram Mudumbai Rangarajan
K. Report : Eshan Gaur and Shreya Reddy

Citations and References

[1] Siciliano, Bruno, et al. *Robotics Modelling, Planning and Control*. Springer, 2009.