ME 598
Introduction to Robotics

Fall 2022

Robot Arm Kinematics

Group # R3
17 October 2022

Graduate students:
"I pledge that I have abided by the Graduate Student Code of Academic Integrity."

The report has been prepared by:

1. Lab leader: Sakshi Ranjeet Pol
2. Krupansh Manish Shah
3. Peera Tienthong
4. Shanmukha Sampath Kumar Tiriveedhi

Table of Contents

Abstract

In robotic manipulators, the important part is to attain desired position and orientation of the end effector or tool. To achieve the desired objective, we must have a brief idea of kinematics. In this report, we are computing the robot motion profile and reachable workspace of the Dobot desktop robot arm. In this we have used Forward as well as Inverse kinematics to calculate schematics of the robot with appropriately placed coordinate frames. All together we wrote a MATLAB code and have written the MATLAB code and simulations with all the required calculations.

Introduction

The dobot is a desktop robot arm consisting of five revolute joints and a vacuum end effector which can be used for lifting and moving small objects. A robot workspace is where it can reach and perform its actions, and an action profile determines the movement of the robot. Forward kinematics can be used to establish the workspace. Determining the inverse kinematics will also help calibrate the arm into a desired position.

There are many different types of the robotic arm used for different purposes. In this Lab we are using the Dobot desktop robot arm which consists of five revolute joints and a vacuum end-effector for lifting and moving small objects.

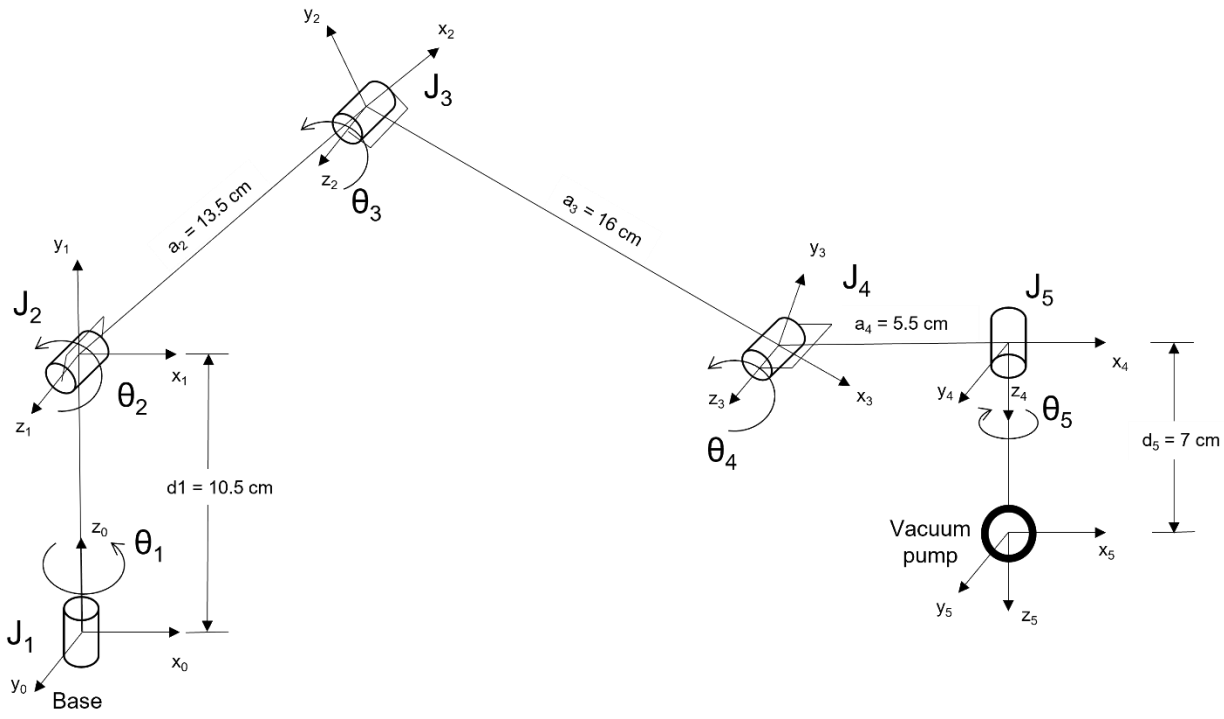


Figure 1: Dobot Schematics

Theory and Experimental Procedure

Part 1: Identify the Robot

The dobot is an articulated robotic arm with five revolute joints. Fig. 1 represents the schematics and configurations of the arm. The direction of arrows representing the joint angles shows the positive movement direction. Constraints are provided for the joints in the below table. These constraints should be considered before analyzing and modeling the workspace.

The end effector, vacuum pump, should always be oriented downward. So, this makes the axes, x_0 and x_3 to be parallel to the ground. And the angle at the third joint, θ_3, can be derived as the negative sum of the angles at joints two and four, θ_2 and θ_4. Once the constraints are established into the configuration, next step is to determine workspace of the robot. Fig. 2 & 3 show the overhead view and side view of the robotic workspace. The overhead view is determined by extending the arm to the furthest in

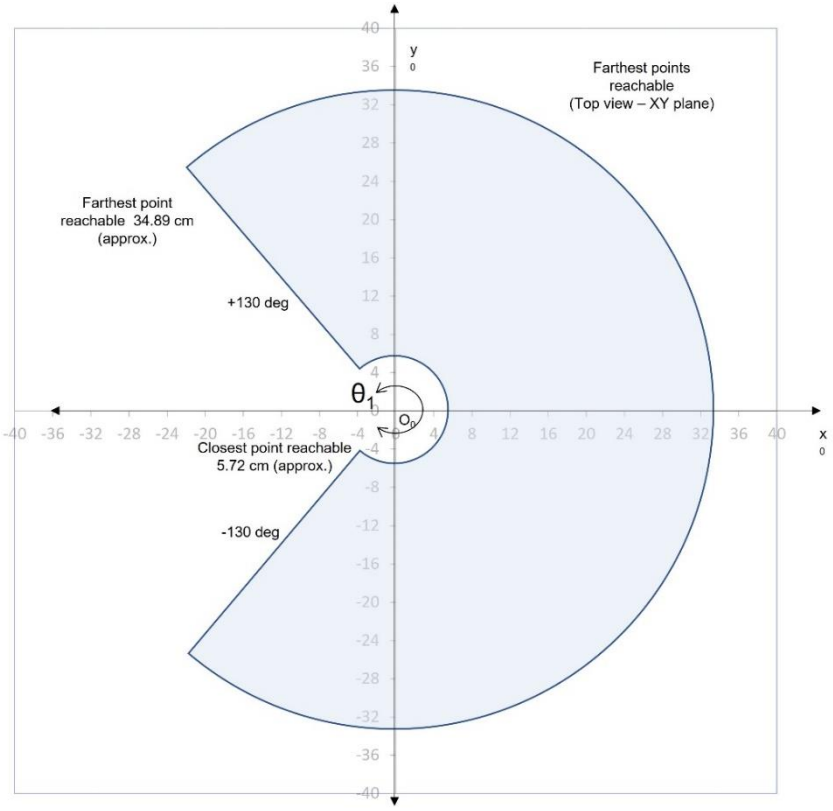


Figure 2: Dobot Workspace - Overhead View - XY Plane

the XY plane and rotating it about the Z-axis. The configuration would be $\theta_2 = 5$ deg. and $\theta_4 = -5$ deg. θ_1 is varied from -130 deg to 130 deg. The side view is determined by employing the limits of joints 2 & 4 and keeping the joint 1 at a constant, say 0 deg. The angle at joint 5 is of minimal use as the value cannot affect the final position of the end effector but it's orientation. So, it is safe to assume it to be always at 0 deg. The dobot

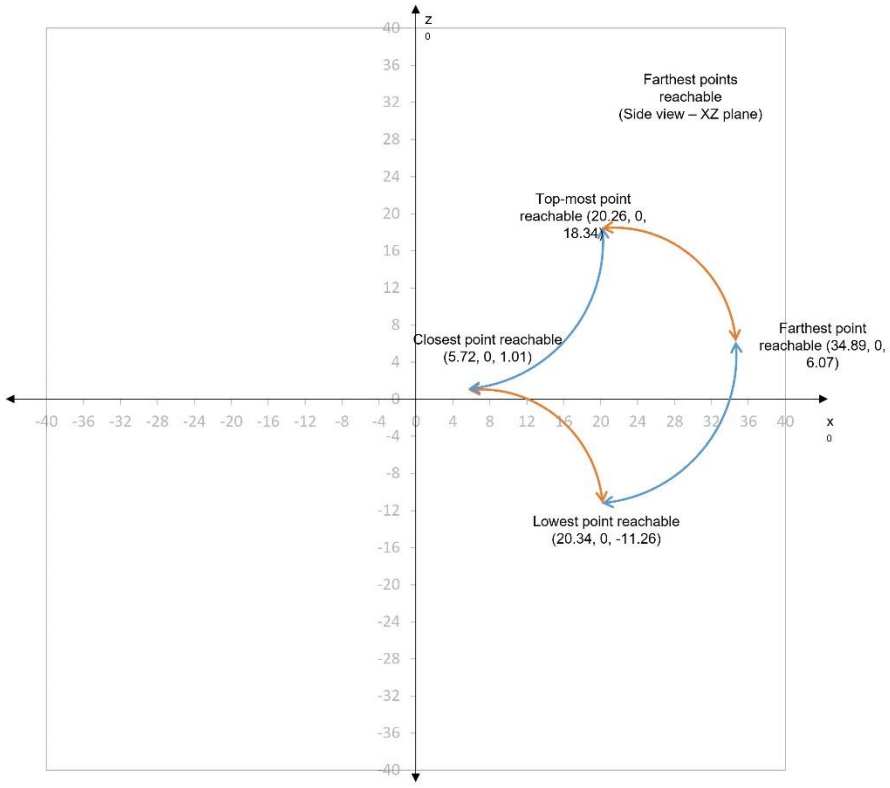


Figure 3: Dobot Workspace - Side View - XZ Plane

workspace is the area highlighted in the overhead view and between the blue and orange curves in the side view.

Part 2: Forward Kinematics

For this experiment, the Denavit Hartenberg (DH) convention was used to develop the forward kinematics of the Dobot.

Link	d	θ	a	a (alpha)
1	10.5	Θ_1^*	0	90^0
2	0	Θ_2^*	13.5	0
3	0	Θ_3^*	16	0
4	0	Θ_4^*	5.5	90^0
5	7	Θ_5^*	0	0

Table 1: DH Parameters with respect to the frames in dobot schematics

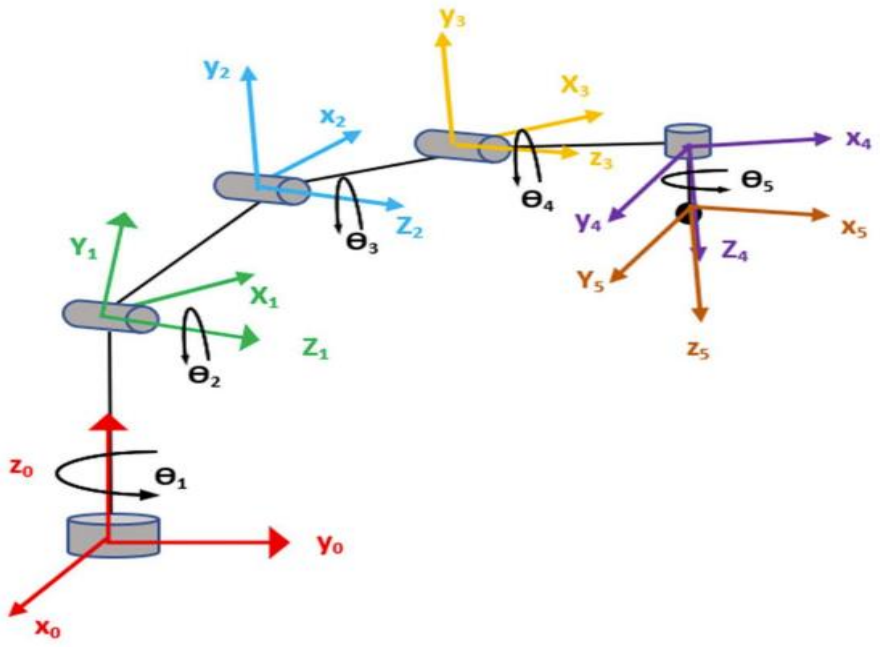


Figure 4: Dobot Schematics with Axes frames

The DH parameters were then defined for each link and summarized in a table as shown in Table 1. In the DH parameter table, each row can be represented by a homogenous transformation matrix as shown below:

a) Transformation matrix from frame 1 to frame 0:

$$T_0^1 = \begin{bmatrix} cos\theta_1 & 0 & sin\theta_1 & 0 \\ sin\theta_1 & 0 & -cos\theta_1 & 0 \\ 0 & 1 & 0 & 10.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

b) Transformation matrix from frame 2 to frame 1:

$$T_1^2 = \begin{bmatrix} cos\theta_2 & -sin\theta_2 & 0 & 13.5cos\theta_2 \\ sin\theta_2 & cos\theta_2 & 0 & 13.5sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

c) Transformation matrix from frame 3 to frame 2:

$$T_2^3 = \begin{bmatrix} cos\theta_3 & -sin\theta_3 & 0 & 16cos\theta_3 \\ sin\theta_3 & cos\theta_3 & 0 & 16sin\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

d) Transformation matrix from frame 4 to frame 3:

$$T_3^4 = \begin{bmatrix} cos\theta_4 & 0 & sin\theta_4 & 5.5cos\theta_4 \\ sin\theta_4 & 0 & -cos\theta_4 & 5.5sin\theta_4 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

e) Transformation matrix from frame 5 to frame 4:

$$T_4^5 = \begin{bmatrix} cos\theta_5 & -sin\theta_5 & 0 & 0 \\ sin\theta_5 & cos\theta_5 & 0 & 0 \\ 0 & 0 & 1 & 7 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Symbols cos and $sine$ represent the cosine and the sine functions, respectively, while R and d represent the rotational and the displacement matrices. Therefore, for links 1 to 5, the homogenous transformation matrices are as shown above.

f) Transformation matrix from frame 5 to frame 0:

$$T_0^5 = T_0^1\, T_1^2\, T_2^3\, T_3^4\, T_4^5$$

$$T_0^5 = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The five homogenous transformations above were multiplied to obtain the final homogenous transformation matrix T_0^5 which is read as the homogenous transformation of the reference frame 5 with respect to base frame 0 as represented above. According to the given conditions, from the dobot geometry, it is evident that $\theta_2 + \theta_3 + \theta_4 = 0$. So, the terms in the transformation matrix are,

$$\begin{aligned} a = {} & sin\theta_1 * sin\theta_5 - cos\theta_5 \\ & * \big(cos\theta_4 * (cos\theta_1 * sin\theta_2 * sin\theta_3 - cos\theta_1 * cos\theta_2 * cos\theta_3) + sin\theta_4 \\ & * (cos\theta_1 * cos\theta_2 * sin\theta_3 + cos\theta_1 * cos\theta_3 * sin\theta_2)\big) \\ & = sin\theta_1 * sin\theta_5 + cos\theta_5 * cos\theta_1 * \cos(\theta_2 + \theta_3 + \theta_4) = cos(\theta_1 - \theta_5) \end{aligned}$$

$$\begin{aligned} b = {} & cos\theta_5 * sin\theta_1 + sin\theta_5 \\ & * \big(cos\theta_4 * (cos\theta_1 * sin\theta_2 * sin\theta_3 - \cos\theta_1 * cos\theta_2 * cos\theta_3) + sin\theta_4 \\ & * (cos\theta_1 * cos\theta_2 * sin\theta_3 + cos\theta_1 * cos\theta_3 * sin\theta_2)\big) \\ & = cos\theta_4 * sin\theta_1 - sin\theta_5 * cos\theta_1 * \cos(\theta_2 + \theta_3 + \theta_4) = \sin(\theta_1 - \theta_5) \end{aligned}$$

$$\begin{aligned} c = {} & cos\theta_4 * (cos\theta_1 * cos\theta_2 * sin\theta_3 + cos\theta_1 * cos\theta_3 * \sin\theta_2) - sin\theta_4 * (cos\theta_1 * sin\theta_2 * sin\theta_3 - \\ & \cos\theta_1 * cos\theta_2 * cos\theta_3) = cos\theta_1 * \sin(\theta_2 + \theta_3 + \theta_4) = 0 \end{aligned}$$

$$d = \frac{27 * cos\theta_1 * cos\theta_2}{2} + 7 * cos\theta_4 * (cos\theta_1 * cos\theta_2 * sin\theta_3 + cos\theta_1 * cos\theta_3 * sin\theta_2) - \frac{11 * cos\theta_4 * (cos\theta_1 * \sin\theta_2 * sin\theta_3 - cos\theta_1 * cos\theta_2 * cos\theta_3)}{2} - \frac{11 * sin\theta_4 * (cos\theta_1 * cos\theta_2 * sin\theta_3 + cos\theta_1 * cos\theta_3 * sin\theta_2)}{2} - 7 * sin\theta_4 * (cos\theta_1 * sin\theta_2 * sin\theta_3 - cos\theta_1 * cos\theta_2 * cos\theta_3) - 16 * cos\theta_1 * sin\theta_2 * sin\theta_3 + 16 * cos\theta_1 * cos\theta_2 * cos\theta_3 = 13.5 * cos\theta_1 * cos\theta_2 + 5.5 * cos\theta_1 + 16 * cos\theta_1 * cos(\theta_2 + \theta_3)$$

$$e = -cos\theta_1 * sin\theta_5 - cos\theta_5 * \big(cos\theta_4 * (sin\theta_1 * sin\theta_2 * sin\theta_3 - cos\theta_2 * cos\theta_3 * sin\theta_1) + sin\theta_4 * (cos\theta_2 * sin\theta_1 * sin\theta_3 + cos\theta_3 * sin\theta_1 * sin\theta_2)\big) = -cos\theta_1 * sin\theta_5 + cos\theta_5 * sin\theta_1 * \cos(\theta_2 + \theta_3 + \theta_5) = \sin(\theta_1 - \theta_5)$$

$$f = sin\theta_5 * \big(cos\theta_4 * (sin\theta_1 * sin\theta_2 * sin\theta_3 - cos\theta_2 * cos\theta_3 * sin\theta_1) + sin\theta_4 * (cos\theta_2 * sin\theta_1 * sin\theta_3 + cos\theta_3 * sin\theta_1 * \sin\theta_2)\big) - cos\theta_1 * cos\theta_5 = -cos\theta_1 * cos\theta_5 - sin\theta_5 * sin\theta_1 * \cos(\theta_2 + \theta_3 + \theta_4) = -\cos(\theta_1 - \theta_5)$$

$$g = cos\theta_4 * (cos\theta_2 * sin\theta_1 * sin\theta_3 + cos\theta_3 * sin\theta_1 * sin\theta_2) - sin\theta_4 * (sin\theta_1 * sin\theta_2 * sin\theta_3 - cos\theta_2 * cos\theta_3 * sin\theta_1) = sin\theta_1 * \sin(\theta_2 + \theta_3 + \theta_4) = 0$$

$$h = \frac{27 * cos\theta_2 * sin\theta_1}{2} + 7 * cos\theta_4 * (cos\theta_2 * sin\theta_1 * sin\theta_3 + cos\theta_3 * sin\theta_1 * sin\theta_2) - \frac{11 * cos\theta_4 * (sin\theta_1 * sin\theta_2 * sin\theta_3 - cos\theta_2 * cos\theta_3 * sin\theta_1)}{2} - \frac{11 * sin\theta_4 * (cos\theta_2 * sin\theta_1 * sin\theta_3 + cos\theta_3 * sin\theta_1 * sin\theta_2)}{2} - 7 * sin\theta_4 * (sin\theta_1 * sin\theta_2 * sin\theta_2 * sin\theta_3 - cos\theta_2 * cos\theta_3 * sin\theta_1) - 16 * sin\theta_1 * sin\theta_2 * sin\theta_3 + 16 * cos\theta_2 * cos\theta_3 * sin\theta_1 = 13.5 * cos\theta_2 * sin\theta_1 + 5.5\, sin\theta_1 + 16 * sin\theta_1 * \cos(\theta_2 + \theta_3)$$

$$i = cos\theta_5 * \big(cos\theta_4 * (cos\theta_2 * sin\theta_3 + cos\theta_3 * sin\theta_2) + sin\theta_4 * (cos\theta_2 * cos\theta_3 - sin\theta_2 * sin\theta_3)\big) = cos\theta_5 * \sin(\theta_2 + \theta_3 + \theta_4) = 0$$

$$j = -sin\theta_5 * \big(cos\theta_4 * (cos\theta_2 * sin\theta_3 + cos\theta_3 * sin\theta_2) + sin\theta_4 * (cos\theta_2 * cos\theta_3 - sin\theta_2 * sin\theta_3)\big) = -sin\theta_5 * \sin(\theta_2 + \theta_3 + \theta_4) = 0$$

$$k = sin\theta_4 * (cos\theta_2 * sin\theta_3 + cos\theta_3 * sin\theta_2) - cos\theta_4 * (cos\theta_2 * cos\theta_3 - sin\theta_2 * sin\theta_3) = -\cos(\theta_2 + \theta_3 + \theta_4) = -1$$

$$l = \frac{(27 * sin\theta_2)}{2} + 16 * cos\theta_2 * sin\theta_3 + 16\, cos\theta_3 * sin\theta_2 + \frac{11 * cos\theta_4 * (cos\theta_2 * sin\theta_3 + cos\theta_3 * sin\theta_2)}{2} - 7 * cos\theta_4 * (cos\theta_2 * cos\theta_3 - sin\theta_2 * sin\theta_3) + 7 * sin\theta_4 * (cos\theta_2 * sin\theta_3 + cos\theta_3 * sin\theta_2) + \frac{\left(11 * sin\theta_4 * (cos\theta_2 * cos\theta_3 - sin\theta_2 * sin\theta_3)\right)}{2} + \frac{21}{2} = 10.5 + 13.5 * sin\theta_2 + 16 * \sin(\theta_2 + \theta_3) + 5.5 * \sin(\theta_2 + \theta_3 + \theta_4) - 7 * \cos(\theta_2 + \theta_3 + \theta_4) = 3.5 + 13.5 * sin\theta_2 + 16 * \sin(\theta_2 + \theta_3)$$

Part 3: Forward Kinematic Model Validation

For validating the forward kinematic model, functions are created in MATLAB and the output is compared with the output of the dobotPlot function which was provided in the assignment. About three functions have been created which can be used to calculate the position of the end effector of the dobot by using forward kinematics. A brief idea of what and how these functions work is provided below.

1. Function to find the transformation matrix from frame i to frame j.

Function Name: trans_f

Input: DH parameters (d, theta, a, alpha)

Output: transformation matrix from frame i to frame j

Description: This function does the transformation matrix from frame I to frame j. The transformation matrix composes of rotation and translation matrices both in z axis and x axis where T is the Translation Matrix and R is the Rotational Matrix. From the DH table, we take the parameters and put it into the forward Kinematic function.

Code: Provided in the appendix

2. Function to find the cartesian coordinates and the orientation of end effector.

Function Name: fwd_kin

Input: configurations (5 joints in degrees)

Output: cartesian coordinates and cup's direction

Description: This function receives the input joints (in degree), then does the transformation based on DH parameters by calling "trans_f" function above. It returns (X, Y, Z) corresponding to the input joints. Finally, the cup_dir contains the cup direction. For normal operation, the cup points downward, opposite to Z base, so that the value of cup_dir < 0.

Code: Provided in the appendix

3. Function to provide the final position of the end effector and validating the inputs according to the constraints. (Function to be called externally with the joint angles)

Function Name: ME598_GrpR3_FwdKin

Input: configurations (5 joints in degrees) in an array

Output: cartesian coordinates and error message if the joints are out of the workspace

Check joints limits

1. Check if the input vector(joints) is no less or more than 5 elements
2. Check base joint, shoulder, wrist and twist are within joint limits
3. Return X, Y, Z from fwd_kin() function
4. Also return the cup direction (to be certain that it points downward as required)

Description: This function basically checks if the input joints are within the limits given in the requirement. If not, it prints the error message to the screen. Otherwise, the (X,Y,Z) is returned.

Code: Provided in the appendix

Validations

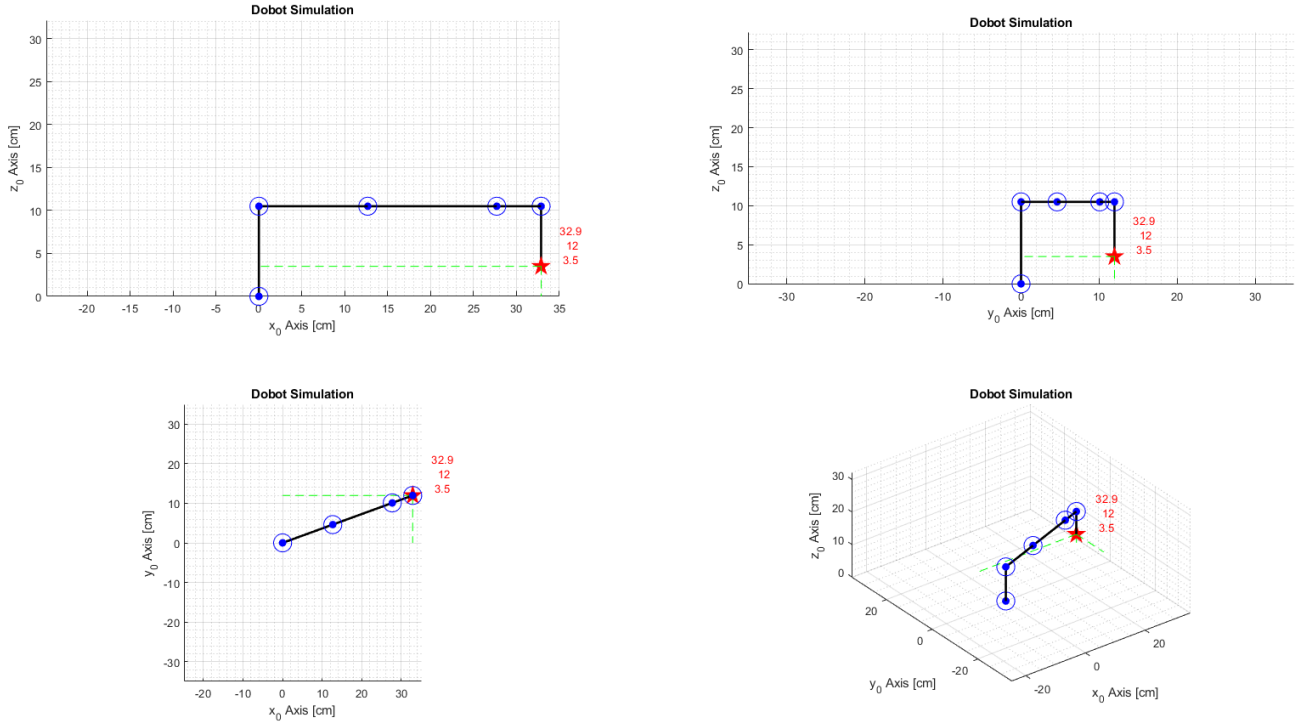
While there are constraints checked before calculating the forward kinematics in the custom function, the simulation one doesn't check for them. Hence, there will be an output for every case.

Example 1: Tested with angles of (20, 0, 0, 0, 0).

Function Output: Throws an error saying the shoulder angle is beyond robot workspace

```
>> ME598_GrpR3_FwdKin([20;0;0;0;0])
Error using ME598_GrpR3_FwdKin
Warning! Shoulder angle beyond robot workspace!
```

Simulation Output: Places the end effector to be at (32.8892, 11.9707, 3.5000)

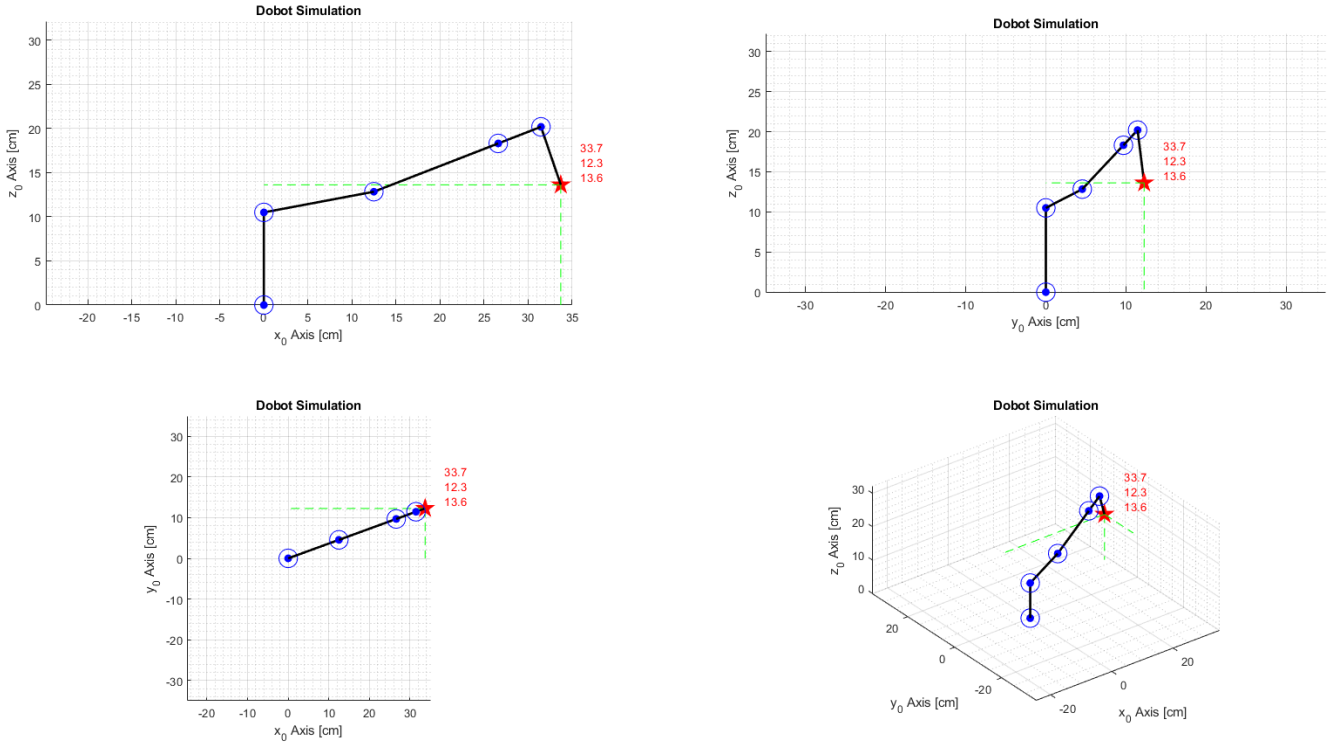


Example 2: Tested with angles of (20, 10, 10, 0, 0)

Function Output: Throws an error saying the joint angles violate wrist constraints

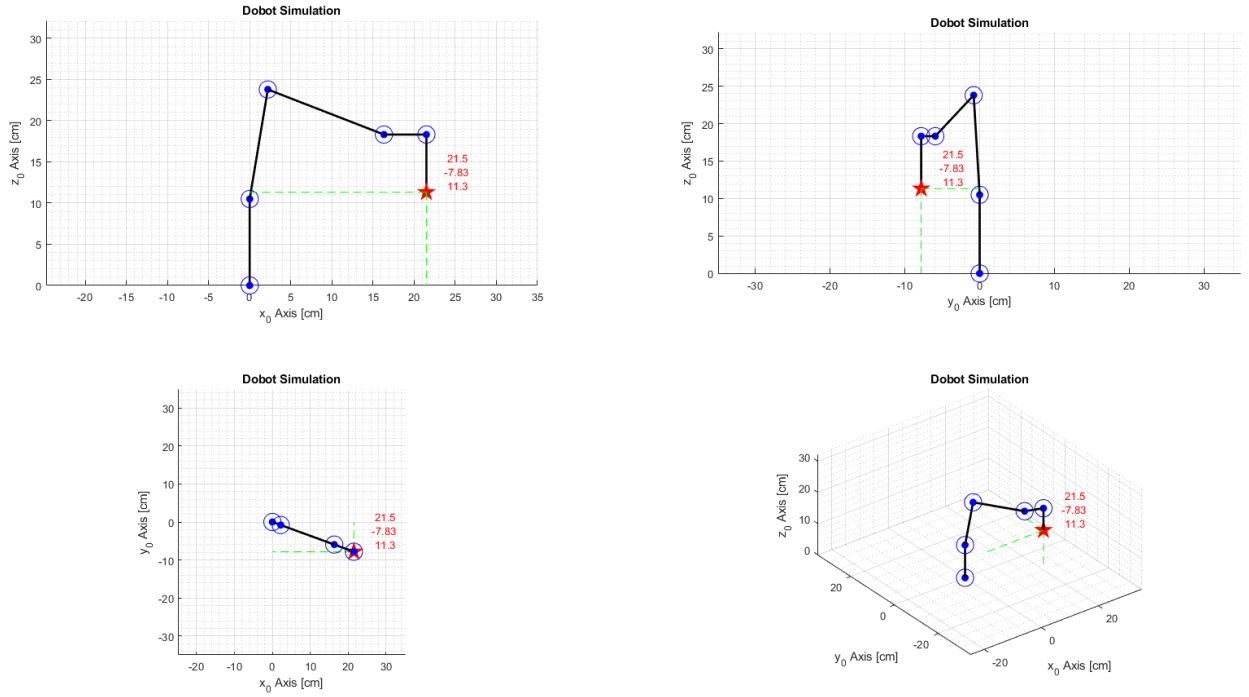
```
>> ME598_GrpR3_FwdKin([20;10;10;0;0])
Error using ME598_GrpR3_FwdKin
Warning! Joint angles violate mechanical constraints on wrist.
```

Simulation Output: Places the end effector to be at (33.7279, 12.2759, 13.6198)



Example 3: Tested with angles of (-20, 80, -100, 20, 0)

<u>Simulation Output:</u> Places the end effector to be at (21.4995, -7.8252, 11.3226)



Function Output: Predicts the end effector to be at (21.4995, -7.8252, 11.3226)

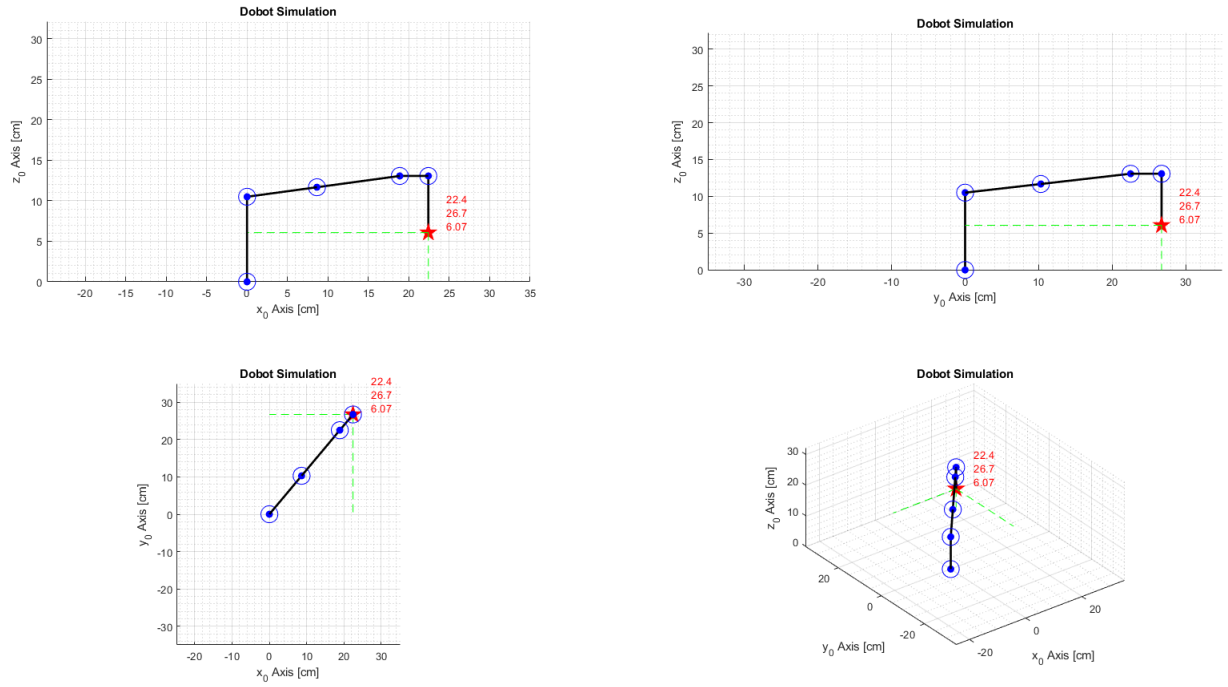
```
>> p = ME598_GrpR3_FwdKin([-20;80;-100;20;0])

p =

   21.4995
   -7.8252
   11.3226
```

Example 4: Tested with angles of (50, 5, 0, -5, 90)

Function Output: Predicts the end effector to be at (22.4254, 26.7256, 6.0711)

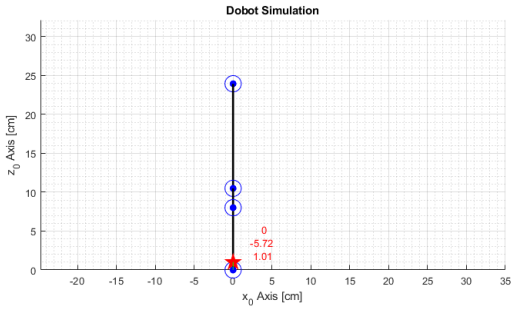
```
>> p = ME598_GrpR3_FwdKin([50;5;0;-5;90])

p =

   22.4254
   26.7256
    6.0711
```

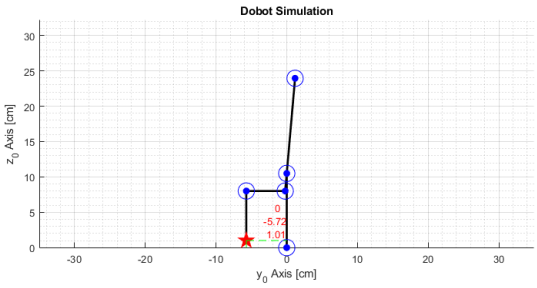
Simulation Output: Places the end effector to be at (22.4254, 26.7256, 6.0711)

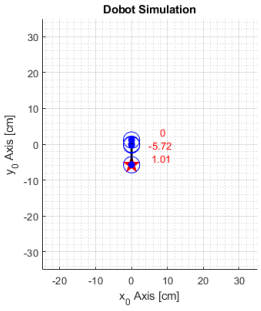


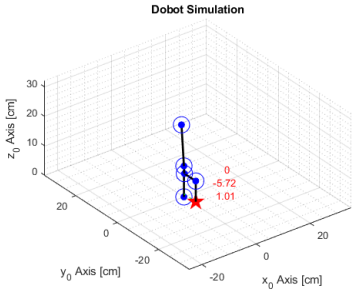
Example 5: Tested with angles (-90, 95, -180, 85, -30)

Simulation Output: Places the end effector to be at (0, -5.7179, 1.0095)









Function Output: Predicts the end effector to be at (0, -5.7179, 1.0095)

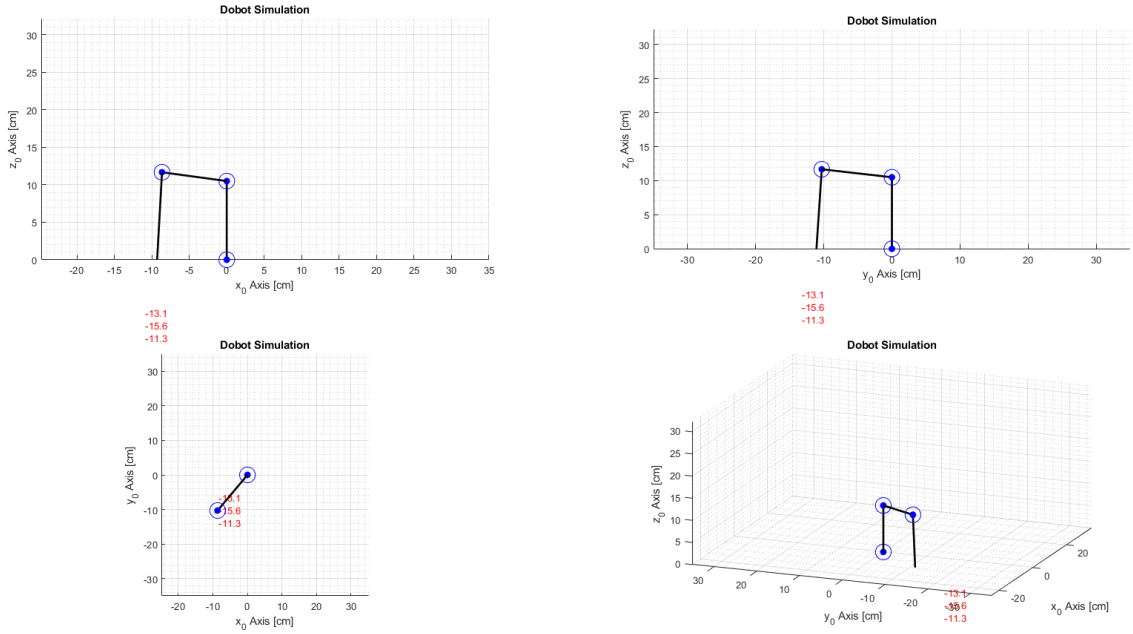
```
>> p = ME598_GrpR3_FwdKin([-90; 95; -180; 85; -30])

p =

   -0.0000
   -5.7179
    1.0095
```

Example 6: Tested with angles (-130, 5, 90, 85, 20)

Function Output: Predicts the end effector to be at (-13.0763, -15.5837, -11.2625)

```
>> p = ME598_GrpR3_FwdKin([-130; 5; -90; 85; 20])

p =

  -13.0763
  -15.5837
  -11.2625
```

Simulation Output: Places the end effector at (-13.0763, -15.5837, -11.2625)



The examples are taken to show how the model behaves within the mechanical constraints and when the input goes out of bounds of the constraints. As demonstrated, the output of the function is the same as the output of the simulation function provided.

Part 4: Inverse-Kinematic Model and Validation

Inverse-Kinematics is the use of kinematic equations to determine the motion of a robot to reach a desired position. The objective of inverse kinematics is to determine the robot configuration in terms of its joint angles given the position of the end-effector in space.

Derivation:

Given:

- $\boldsymbol{\theta_5 = 0}$
- Position of the end effector $P = (P_x, P_y, P_z)$
- $\theta_4 + \theta_3 + \theta_2 = 0$ as the end effector must be always perpendicular to the ground.

From the geometry (i.e., shadow of the dobot) it is evident that $\boldsymbol{\theta_1 = tan^{-1}\,{P_y}/{Px}}$

Taking reference from figure 5:

$$\boldsymbol{\theta_2 = tan^{-1}\left({y_1}/{x_1}\right) \pm tan^{-1}\left({y_2}/{x_2}\right)}$$

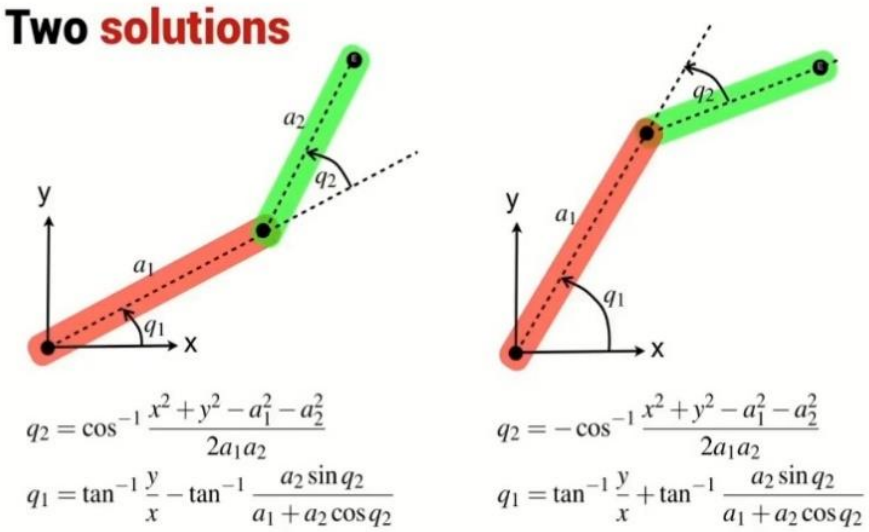


Figure 5: Ref: robotacademy.net.au

From the dobot's geometry

$$y_1 = P_y - d_1 - d_5$$

$$x_1 = \frac{P_x}{cos\theta_1} - a_4$$

$$y_2 = a_2\, sin\theta_3$$

$$x_2 = a_1 + a_2\, cos\theta_3$$

$$\boldsymbol{\theta_3 = -cos^{-1}\frac{x_1^2 + y_1^2 - a_1^2 - a_2^2}{2a_1a_2}}$$

$$\boldsymbol{\theta_4 = -\theta_3 - \theta_2}$$

Given below are 3 examples to demonstrate the working of our MATLAB code and Derivations

Example 1: Tested with the input of (-13.0763, -15.5837, -11.2625) which is the output of example 6 in forward kinematics validation. The output is same as the input of (-130, 45, -90, 85, 0)

```
>> q = ME598_GrpR3_InvKin(-13.0763, -15.5837, -11.2625)

q =

 -130.0000
    5.0001
  -90.0002
   85.0001
         0
```

Example 2: Tested with the output of example 4 in forward kinematics. (21.4995, -7.8252, 11.3226). The output is same as the input i.e. (-20, 80, -100, 20, 0)

```
>> q = ME598_GrpR3_InvKin(21.4995, -7.8252, 11.3226)

q =

   -20.0001
    80.0002
  -100.0001
    20.0000
         0
```

Example 3: Tested with the coordinates of (25, 12.7179, 12.0095). The output is within the workspace

```
>> q = ME598_GrpR3_InvKin(25, 12.7179, 12.0095)

q =

   26.9632
   59.4786
  -70.7231
   11.2445
         0
```

Example 4: Tested with coordinates of (10, 15.7179, 12.0095) which are beyond the robot workspace.

```
>> q = ME598_GrpR3_InvKin(10, 15.7179, 12.0095)
Error using ME598_GrpR3_InvKin
Warning! End effector beyond robot workspace!
```

Results

1) $T_5^0 =$

$$\begin{bmatrix} \cos(\theta_1 - \theta_5) & \sin(\theta_1 - \theta_5) & 0 & 13.5 * cos\theta_1 * cos\theta_2 + 5.5 * cos\theta_1 + 16 * cos\theta_1 * cos(\theta_2 + \theta_3) \\ \sin(\theta_1 - \theta_5) & -\cos(\theta_1 - \theta_5) & 0 & 13.5 * cos\theta_2 * sin\theta_1 + 5.5\, sin\theta_1 + 16 * sin\theta_1 * \cos(\theta_2 + \theta_3) \\ 0 & 0 & -1 & 3.5 + 13.5 * sin\theta_2 + 16 * \sin(\theta_2 + \theta_3) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The homogeneous transformation matrix above was derived by multiplying all the transformation matrices of the intermediate links from the base to the end-effector link. This matrix is further used to solve the forward and inverse kinematics.

2)The inverse kinematic function was determined using Geometric analysis and was verified using MATLAB Editor script.

- $\theta_1 = tan^{-1} {}^{P_y}/_{Px}$
- $\theta_2 = tan^{-1}({}^{y_1}/_{x_1}) \pm tan^{-1}({}^{y_2}/_{x_2})$
- $\theta_3 = -cos^{-1} \frac{{x_1}^2 + {y_1}^2 - a_1^2 - a_2^2}{2a_1a_2}$
- $\theta_4 = -\theta_3 - \theta_2$
- $\theta_5 = 0$

Discussion:

It is assumed that vacuum cup of the dobot always faces the ground, which implies that the fifth link's z axis is always perpendicular to the plane of the ground. The results obtained conform to the joint constrains and are same as those expected. There are minor deviations in terms of tenths of decimal values which can be caused by the way the trigonometric functions generate the outputs as this is dependent on the machine and some other constraints. If executed on the same machine, the output should be constant for both forward and inverse kinematic functions.

Conclusion

We completed the experiment successfully. It required an understanding of the concepts of Denavit - Hartenberg (DH) Conventions, Forward Kinematics, Inverse Kinematics, Homogeneous Matrix, and Workspace analysis. All of which was completed and achieved in the above parts. My team members and I were intrigued to do this experiment. We did face some difficulties, but we eventually overcame them by extensive research and teamwork.

References

[1] Mark W Spong, Seth Hutchinson, M. Vidyasagar – Robot Modeling and Control – 1st Edition, 2006, Tehseen F. Abaas, Ali A. Khlief, Mohanad Q. Abbood," Inverse Kinematics Analysis and Simulation of a 5 DOF Robotic Arm using MATLAB , AL-Khwarizmi Engineering Journal, Vol. 16, No.1, (2020)

[2] https://robotacademy.net.au/lesson/inverse-kinematics-for-a-2-joint-robot-arm-using-geometry/

Appendices

- MATLAB code for Forward Kinematics (Model Validation)

```
function coordis = ME598_GrpR3_FwdKin(confs)
%Error throw1 : the confs size must satisfy 5x1, check matrix dimension
```

```
    if length(confs) < 5 || length(confs) > 5
        error('The number of degrees input in array must be equal to 5 (base,
shoulder, elbow, wrist, twist)');
    end
%Error throw2 : check 1: Base, Shoulder, and Twist joints are within ranges
    if confs(1) < -130 || confs(1) > 130 %base joint beyond range
        error('Warning! Base angle beyond robot workspace!');
    elseif confs(2) < 5 || confs(2) > 95 %shoulder joint beyond range
        error('Warning! Shoulder angle beyond robot workspace!');
    elseif confs(4) < -5 || confs(4) > 85 %wrist joint beyond range
        error('Warning! Wrist angle beyond robot workspace!');
    elseif confs(5) < -90 || confs(5) > 90 %twist joint beyond range
        error('Warning! Twist angle beyond robot workspace!');
    elseif confs(2) + confs(3) + confs(4) ~= 0
        error('Warning! Joint angles violate mechanical constraints on wrist.')
    else %within robot workspace
        [x,y,z,cup_dir] = fwd_kin(confs);
        if cup_dir < 0
            coordis = [x;y;z];
        else
            error('cup is not in downward direction');
        end

    end

end

%% Function involves in part3
function T = trans_f(d,theta,a,alpha) %input unit in cm and rad
    R_z = [cos(theta)  -sin(theta)  0 0;
           sin(theta)  cos(theta)   0 0;
           0           0            1 0;
           0           0            0 1];
    T_z = [1 0 0 0;
           0 1 0 0;
           0 0 1 d;
           0 0 0 1];
    T_x = [1 0 0 a;
           0 1 0 0;
           0 0 1 0;
           0 0 0 1];
    R_x = [1       0       0        0;
           0 cos(alpha) -sin(alpha) 0;
           0 sin(alpha) cos(alpha)  0;
           0       0       0        1];

    T = R_z*T_z*T_x*R_x;
end


function [x,y,z,cup_dir] = fwd_kin(confs) %input unit in cm and degrees
    q_base = deg2rad(confs(1));
    q_shoulder = deg2rad(confs(2));
    q_elbow = deg2rad(confs(3));
    q_wrist = deg2rad(confs(4));
```

```
    q_twist = deg2rad(confs(5));

    %transforming frames
    T_01 = trans_f(10.5,q_base,0,deg2rad(90));
    T_12 = trans_f(0,q_shoulder,13.5,0);
    T_23 = trans_f(0,q_elbow,16,0);
    T_34 = trans_f(0,q_wrist,5.5,deg2rad(90));
    T_45 = trans_f(7,q_twist,0,0);

    T_05 = T_01 * T_12 * T_23 * T_34 * T_45;
    x = T_05(1,4);
    y = T_05(2,4);
    z = T_05(3,4);
    cup_dir = dot([0;0;1],[T_05(1,3);T_05(2,3);T_05(3,3)]);

end
```

- MATLAB code for Inverse Kinematics (PART 4)

```
function q = ME598_GrpR3_InvKin(Px, Py, Pz)

%Defining the the fixed values (i.e. arm length and theta_5)
d1 = 10.5;
a2 = 13.5;
a3 = 16;
a4 = 5.5;
d5 = 7;
theta_5d = 0;

try
    theta_1r = atan2(Py,Px);                    %theta 1 in radians
    theta_1d = theta_1r*180/pi;

    x1 = Px/cos(theta_1r) - a4;
    y1 = Pz - d1 + d5;
    x3 = (x1^2+y1^2-a2^2-a3^2) / (2*a2*a3);

    theta_3r = -acos(x3);                   %theta 3 in radians
    theta_3d = theta_3r*180/pi;

    x2 = a2 + a3*cos(theta_3r);
    y2 = a3*sin(theta_3r);
    theta_2r = atan2(y1,x1) - atan2(y2,x2);     %theta 2 in radians
                                                %due to the +5 to +95
                                                %constrain the two angles
                                                %are always subtracted.
    theta_2d = theta_2r*180/pi;

    theta_4d = -(theta_2d + theta_3d);          %theta 4 in degrees.

    if( (130 < theta_1d) || (theta_1d < -130) || (theta_2d < 5) || (theta_2d > 95) ||
(theta_4d < -5) || (theta_4d > 85) )
        error("Warning! End effector beyond robot workspace!");
```

```
    end

     q = [theta_1d; theta_2d; theta_3d; theta_4d; theta_5d];

catch
    error("Warning! End effector beyond robot workspace!")
end
```