MATRX5700

Assignment 1

470341137
470145351
470356706
470043677

14 Mar. 2020

Question 1

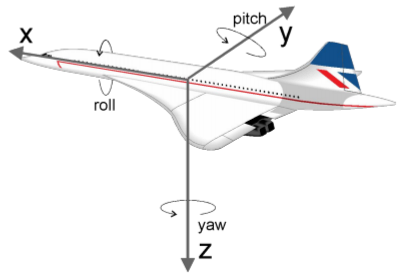

Figure 1: RPY standard in Engineering field

To determine the rotation matrix, an Engineering standard of Roll, Pitch, Yaw (R-P-Y) Fixed Angles is used. It is shown in Figure 1. The rotation matrix using the standard can therefor be obtained.

$$
\begin{aligned}
{}^{A}R_{BXYZ}(\gamma,\beta,\alpha) &= R_Z(\alpha)R_Y(\beta)R_X(\gamma) \\
&= \begin{bmatrix} c_\alpha & -s_\alpha & 0 \\ s_\alpha & c_\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\beta & 0 & s_\beta \\ 0 & 1 & 0 \\ -s_\beta & 0 & c_\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\gamma & -s_\gamma \\ 0 & s_\gamma & c_\gamma \end{bmatrix}
\end{aligned}
$$

The homogeneous transformation matrix is a compact representation of the translation and rotation.

$$
{}^{A}_{B}T = \begin{bmatrix} & {}^{A}_{B}R & & {}^{A}P_B \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

where ${}^{A}P_B$ is the translation matrix of B relative to A.

a.

$$
{}^{A}R_{BXYZ}(30°, 10°, 20°) = \begin{bmatrix} 0.9254 & -0.2146 & 0.3123 \\ 0.3368 & 0.8435 & -0.4184 \\ -0.1736 & 0.4924 & 0.8529 \end{bmatrix}
$$

With the translation matrix

$$^{A}P_{B} = \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix}$$

The homogeneous transformation matrix is

$$^{A}_{B}T = \begin{bmatrix} 0.9254 & -0.2146 & 0.3123 & 2 \\ 0.3368 & 0.8435 & -0.4184 & 3 \\ -0.1736 & 0.4924 & 0.8529 & 4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

b.

$$^{A}R_{BXYZ}(15°, 10°, 25°) = \begin{bmatrix} 0.8925 & -0.3675 & 0.2614 \\ 0.4162 & 0.8944 & -0.1637 \\ -0.1736 & 0.2549 & 0.9513 \end{bmatrix}$$

With the translation matrix

$$^{A}P_{B} = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}$$

The homogeneous transformation matrix is

$$^{A}_{B}T = \begin{bmatrix} 0.8925 & -0.3675 & 0.2614 & 2 \\ 0.4162 & 0.8944 & -0.1637 & 0 \\ -0.1736 & 0.2549 & 0.9513 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

c.

$$^{A}R_{BXYZ}(270°, 270°, 90°) = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

With the translation matrix

$$^{A}P_{B} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

The homogeneous transformation matrix is

$$ {}^{A}_{B}T = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{A}R_{BXYZ}(90°, -90°, -90°) = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} $$

With the translation matrix

$$ {}^{A}P_{B} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} $$

The homogeneous transformation matrix is

$$ {}^{A}_{B}T = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

The two transformation matrices are the same.

Question 2

a.

A matrix R is a valid rotational matrix if and only if it is orthogonal, meaning that

$$RR^T = R^T R = I$$

$$det(R) = 1$$

For matrix R_1, round up some decimal values

$$det(R_1) = 1.0065 \approx 1$$

$$R_1 R_1^T \approx R_1^T R_1 \approx I$$

So, R_1 is a valid rotation matrix.

For matrix R_2, round up some decimal values

$$det(R_2) = 1.0662$$

$$R_2^T R_2 = \begin{bmatrix} 1.0446 & 0.0194 & 0.004 \\ 0.0194 & 1.0558 & 0.0052 \\ 0.004 & 0.0052 & 1.0311 \end{bmatrix}$$

It is not strictly an identity matrix, and the determinant of R_2 is bigger than 1. So, we could say that R_2 is not a valid rotation matrix.

For matrix R_3,

$$det(R_3) = 1.0000$$

$$R_3 R_3^T = R_3^T R_3 = I$$

So, R_3 is a valid rotation matrix.

For matrix R_4,

$$det(R_4) = -1.0000$$

$$R_4 R_4^T = R_4^t R_4 = I$$

As the determinant of R_4 is -1, it is not a valid rotation matrix.

b.

Assume that the R-P-Y standard is the same as in Question 1. For matrix R_1,

$$Roll = 50.9977°$$

$$Pitch = 30.0035°$$

$$Yaw = -45.0000°$$

These values are valid since matrix R_1 is a valid rotation matrix, and it will not cause gimbal lock.

For matrix R_2,

$$Roll = 48.3247°$$

$$Pitch = 38.9339°$$

$$Yaw = 28.9797°$$

We might not trust these values as the rotation matrix in not valid.

For matrix R_3,

$$Roll = 0°$$

$$Pitch = -90.0000°$$

$$Yaw = 120.0007°$$

We might not trust these values. As when pitch angle approaches to 90/-90 degrees, it causes a 'gimbal lock', meaning there is a singularity in the R-P-Y Eular angles set.

For matrix R_4,

$$Roll = 98.4075°$$

$$Pitch = 9.0988°$$

$$Yaw = 71.8002°$$

We might not trust these values as the rotation matrix is not valid.

c.

The quaternion for each rotation matrix can be found by Matlab builtin function **Q = UnitQuaternion(R)**. The output of the function is in the $\mathbf{w} < \mathbf{x}, \mathbf{y}, \mathbf{z} >$ form.
The quoternions for each rotation matrix are:

$$Q1 = 0.76282 < -0.43657, 0.29477, 0.37498 >$$

$$Q2 = 0.87387 < 0.29218, 0.38069, 0.077815 >$$

$$Q3 = 0.35355 < 0.61237, -0.35355, 0.61237 >$$

$$Q4 = 0.58273 < 0.81166, 0.024621, -0.03212 >$$

Using quaternions is computationally more efficient. And there is no singularities ("gimbal lock") problems in quaternions.

d.

To estimate the values of roll, pitch, and yaw, we need to find the closest orthogonal matrix. Using singular value decomposition (Matlab function **[U,S,V] = svd(R)**), two rotations and a scaling matrix can be calculated. The nearest rotation matrix is the product of UV^* since an orthogonal matrix has the decomposition UIV^* where I is the identity matrix.

The normalised rotation matrix of R2 is

$$R_{2norm} = \begin{bmatrix} 0.6782 & 0.0964 & 0.7285 \\ 0.3702 & 0.8115 & -0.4520 \\ -0.6348 & 0.5763 & 0.5147 \end{bmatrix}$$

The estimated row-pitch-yaw of R2 is $[48.2288°, 39.4037°, 28.6289°]$.

The normalised rotation matrix of R4 is

$$R_{4norm} = \begin{bmatrix} 0.3084 & -0.0900 & -0.9470 \\ 0.9380 & 0.1943 & 0.2870 \\ -0.1581 & 0.9768 & -0.1444 \end{bmatrix}$$

The estimated row-pitch-yaw of R4 is $[98.4075°, 9.0988°, 71.8002°]$.

Question 3

For a 6R 6DOF robot arm, the Denavit-Hartenberg convension is used to derive the forward kinematics from the base to its end effector. The DH parameters are shown below. d_6 is not given in the question, assume $d_6 = 80mm$.

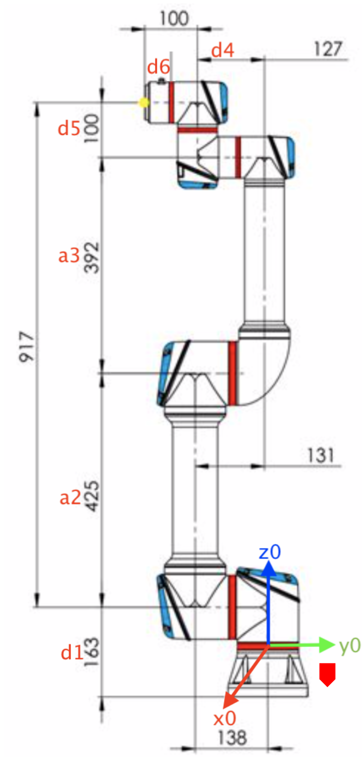

Figure 2: Denavit-Hartenberg parameters for UR5e

i	d_i(mm)	θ_i(deg)	a_i(mm)	α_i(deg)
1	163	θ_1	0	90
2	0	θ_2	425	0
3	0	θ_3	392	0
4	127	θ_4	0	90
5	100	θ_5	0	-90
6	80	θ_6	0	0

Table 1: Denavit-Hartenberg parameters for UR5e

Substitute into transformation matrix:

$$^{i-1}T_i = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 & a \\ sin(\theta) * cos(\alpha) & cos(\theta) * cos(\alpha) & -sin(\alpha) & -d * sin(\alpha) \\ sin(\theta) * sin(\alpha) & cos(\theta) * sin(\alpha) & cos(\alpha) & cos(\alpha) * d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The forward kinematic solution can be calculated as:

$$^0T_6 = {}^0T_1{}^1T_2{}^2T_3{}^3T_4{}^4T_5{}^5T_6$$

Simulate the solution in Matlab with $\theta_1 = 45, \theta_2 = 0, \theta_3 = 90, \theta_4 = 0, \theta_5 = 90, \theta_6 = 30$, we can get the position of the end effector $(x, y, z) = (859.07, -78.56, -352.0081)$ and a simulated plot of the robot arm.

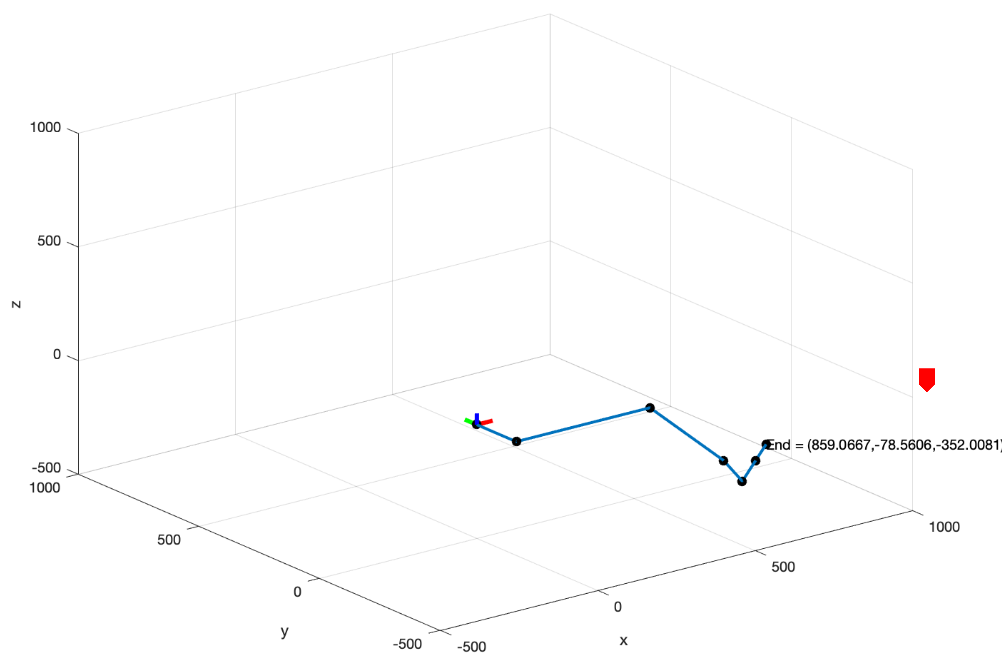

Figure 3: Simulation of the obtained kinematic equations

Question 4

The workspace of the the 3-link planar arm which describes the points in space that it can reach is shown as following. The generated MATLAB code is included in Appendix.

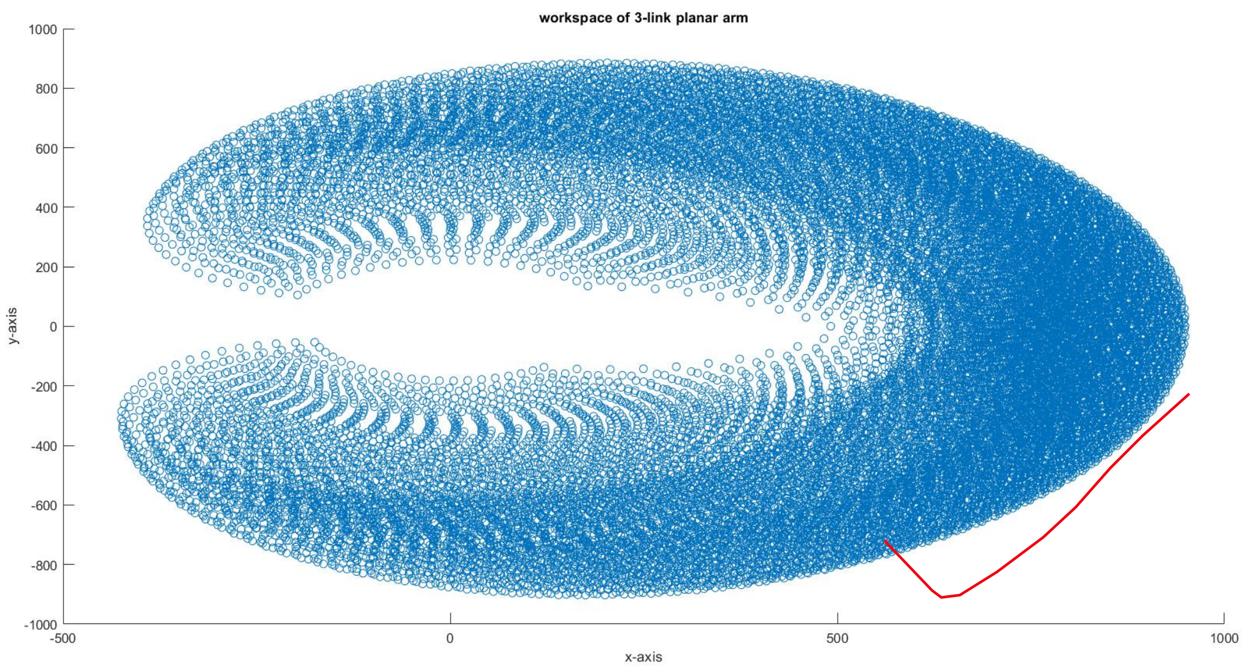

Figure 4: The workspace of 3-link planar arm

The configuration space of the 3-link planar arm are shown as following. The generated MATLAB code is included in Appendix.

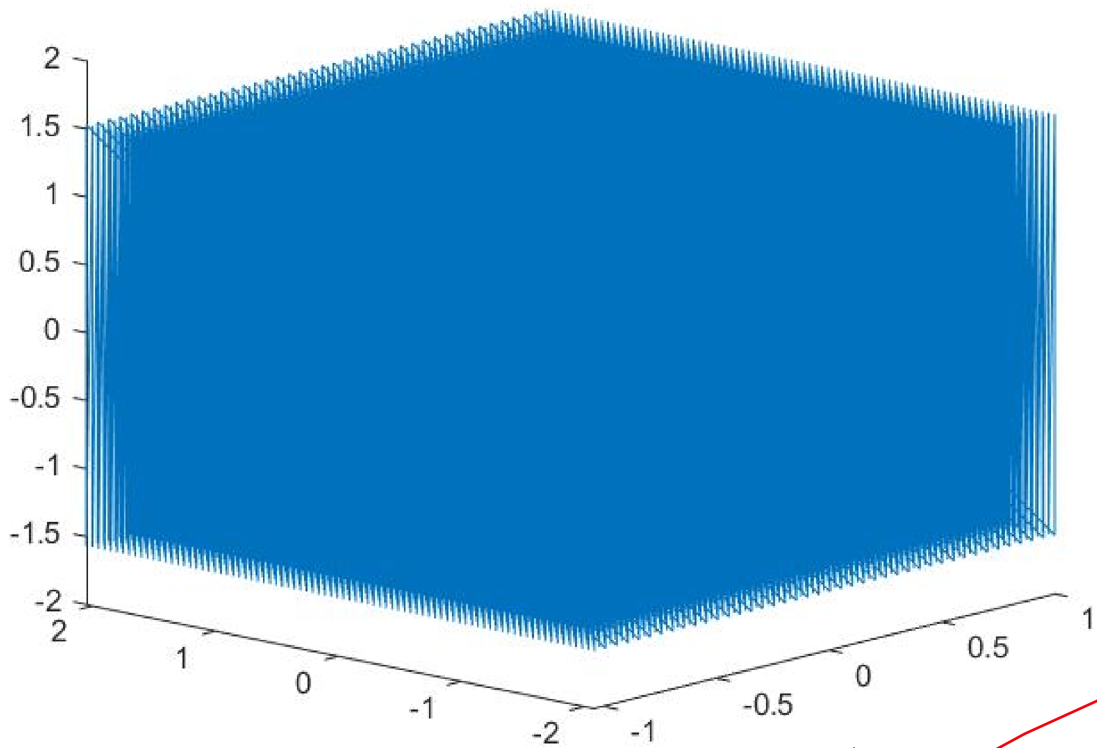

Figure 5: The configuration space of 3-link planar arm

The singularity of mechanism is defined by the determination of Jacobian matrix. However, for a 3-link planar arm, the Jacobian matrix is a 2x3 matrix shown below.

$$J = \begin{bmatrix} -L1S1 - L2S12 - L3S123 & -L2S12 - L3S123 & -L3S123 \\ L1C1 + L2C12 + L3C123 & L2C12 + L3C123 & L3C123 \end{bmatrix}$$

According to the definition of determination of matrix, the determination is only for the square matrix not rectangular matrix. Therefore, instead, the dependency of Jacobian matrix is proven by the MATLAB code which is included in Appendix.

1. The singularity of 3-link planar arm is the external boundary and internal boundary of the workspace shown above.

2. The singularity of 3-link planar arm is when $sin\theta_2$ and $sin\theta_3$ are both zero since the

configuration space limits the value of them to be π. This is the situation where the arm is at full extension. The arm cannot move radially in Cartesion space and this is loss of degree of freedom.

Question 5

(a) There are two approach to derive the inverse kinematics solutions for this UR5e robot arm,geometric method is used rather than algebraic method as it is more simple to find the most efficient way. Although using algebraic method can show multiple solutions, the most efficient way requires lest rotation on each joint.

As the block is placing 700m in front of the robot with 134mm offset. It means the most efficient way to reach the position is when the arm is perpendicular to the x-axis. Also we assume that the gripper should aligned with the y-axis with the tips vertically down at 40mm to pick up/place the first block. Thus the angle $\theta_1 = 90°$, $\theta_5 = -90°$, $\theta_6 = 90°$. The remaining angle can be solve using cosine rule on the side view shown in Figure.

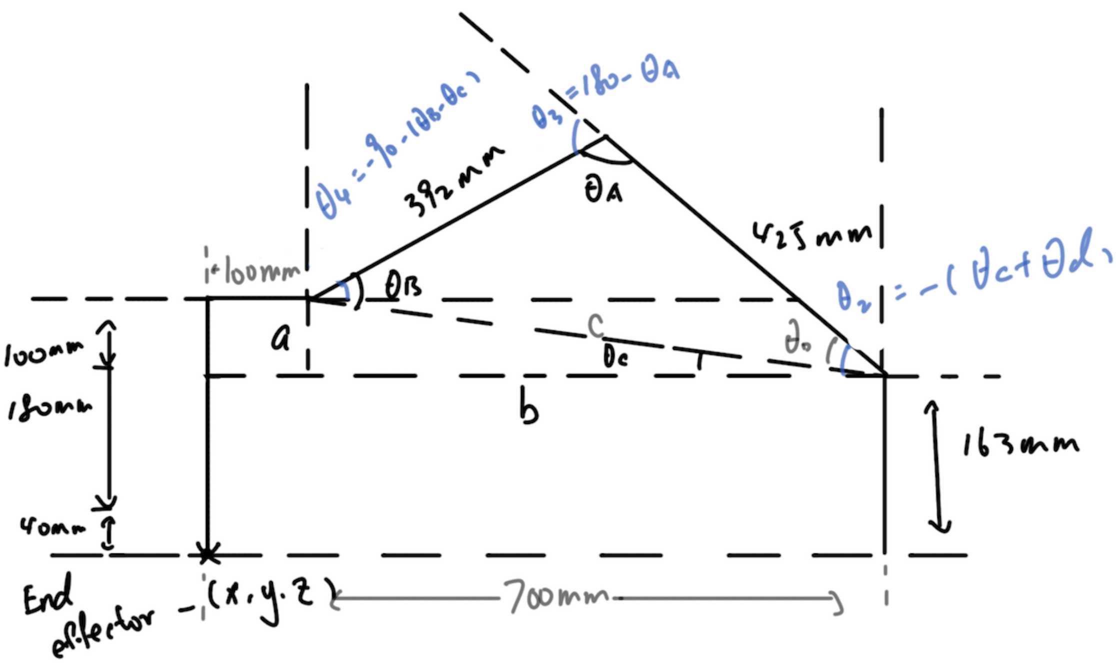

Figure 6: free body diagram of the robot arm

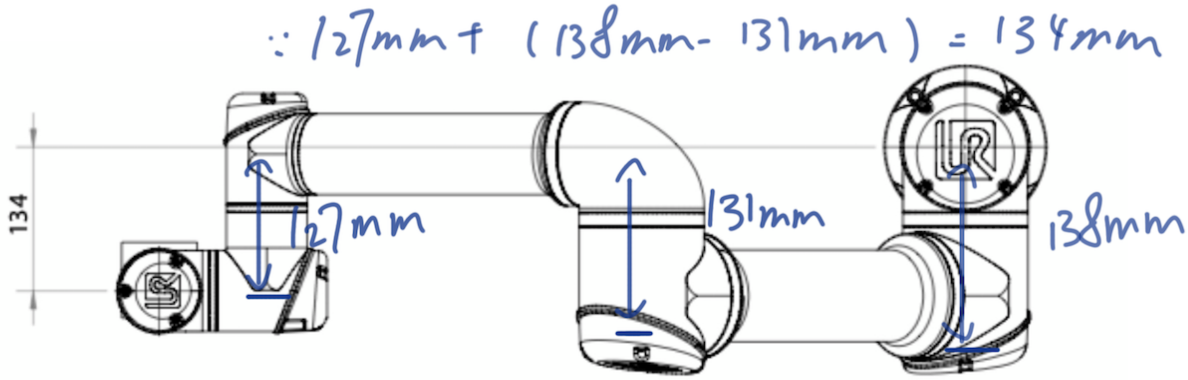

Figure 7: offset of the robot arm

With the end effector position $x = 134mm$, $y = 700mm$, $z = 40mm$

$$a = (100 + 180 + z) - 163 = 157mm$$

$$b = y - 100 = 600mm$$

$$c = \sqrt{(y-100)^2 + (117+z)^2} = 620mm$$

$$\theta_C = tan^{-1}\frac{117+z}{y-100} = tan^{-1}\frac{157}{600} = 14.66°$$

By cosine rule

$$392^2 + 425^2 - 2 \times 392 \times 452 \times cos\theta_A = c^2$$

$$\theta_A = cos^{-1}\frac{c^2 - 392^2 - 425^2}{-2 \times 392 \times 452} = 98.69°$$

By cosine rule

$$c^2 + 392^2 - 2 \times c \times 392 \times cos\theta_B = 425^2$$

$$\theta_B = cos^{-1}\frac{392^2 - c^2 - 425^2}{-2 \times c \times 392} = 42.64°$$

$$\theta_D = 180° - (\theta_A + \theta_B) = 38.67°$$

Thus

$$\theta_2 = -(\theta_C + \theta_D) = -53.33°$$

This is mostly right but the coordin

$$\theta_3 = 180° - \theta_A = 81.31°$$

$$\theta_4 = -90 - (\theta_B - \theta_C) = -117.98$$

(b) Simulation results using actual robot function are Figure4-7 shown below

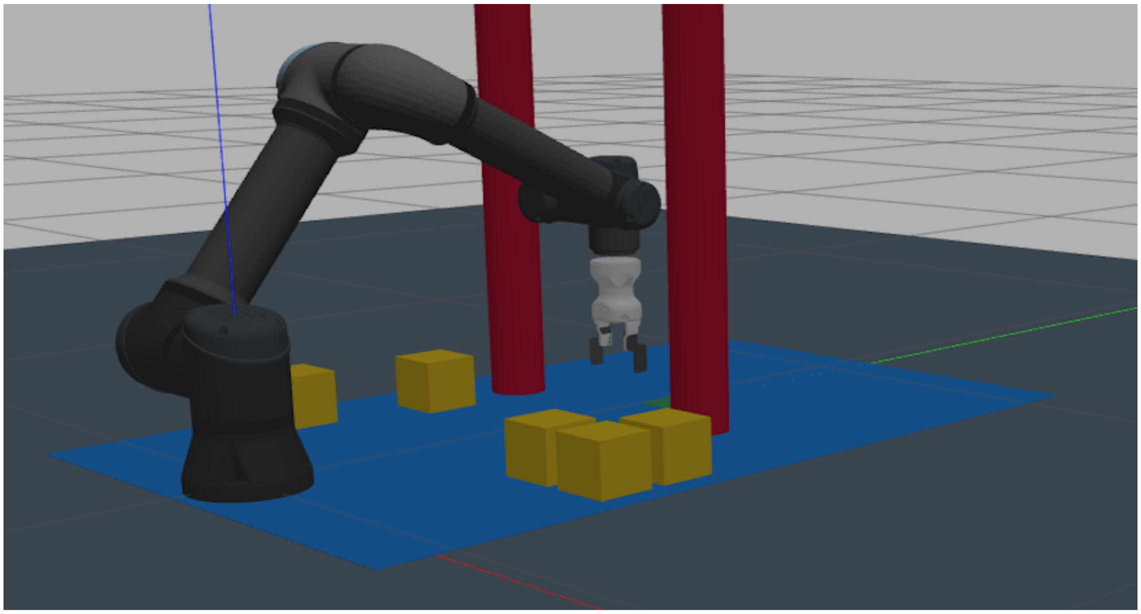

Figure 8: Robot simulation

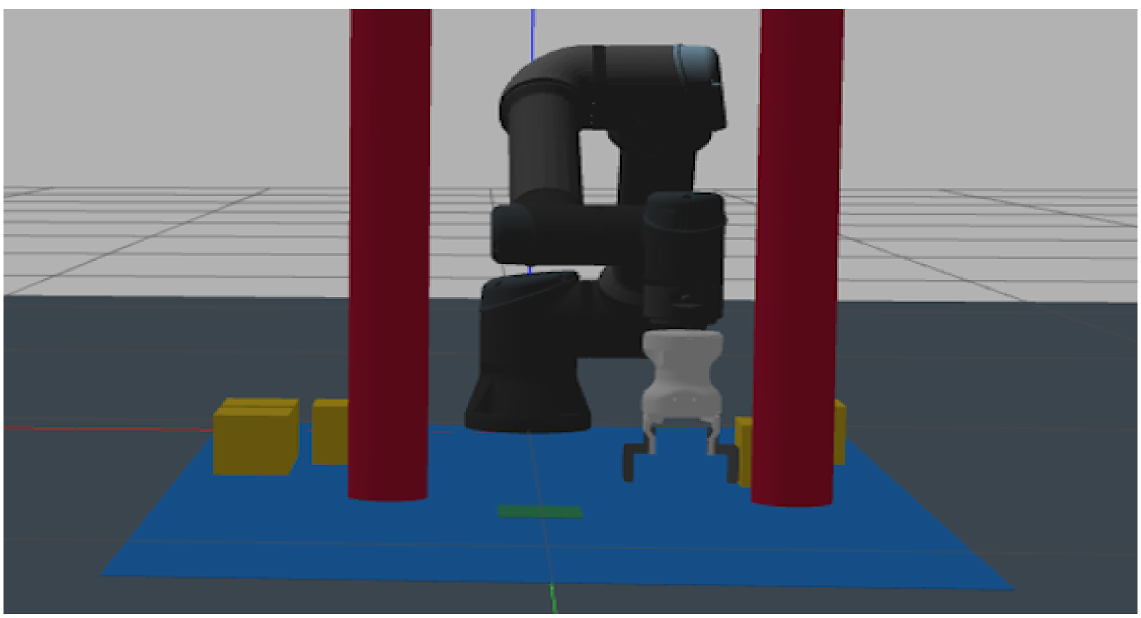

Figure 9: Front view of the robot

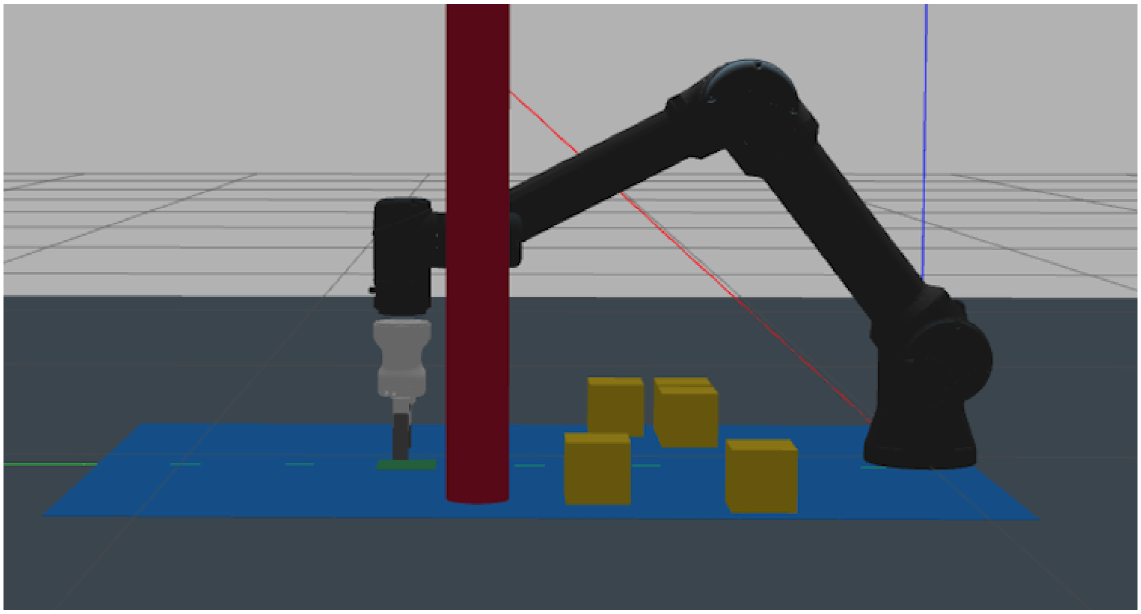

Figure 10: side view of the robot

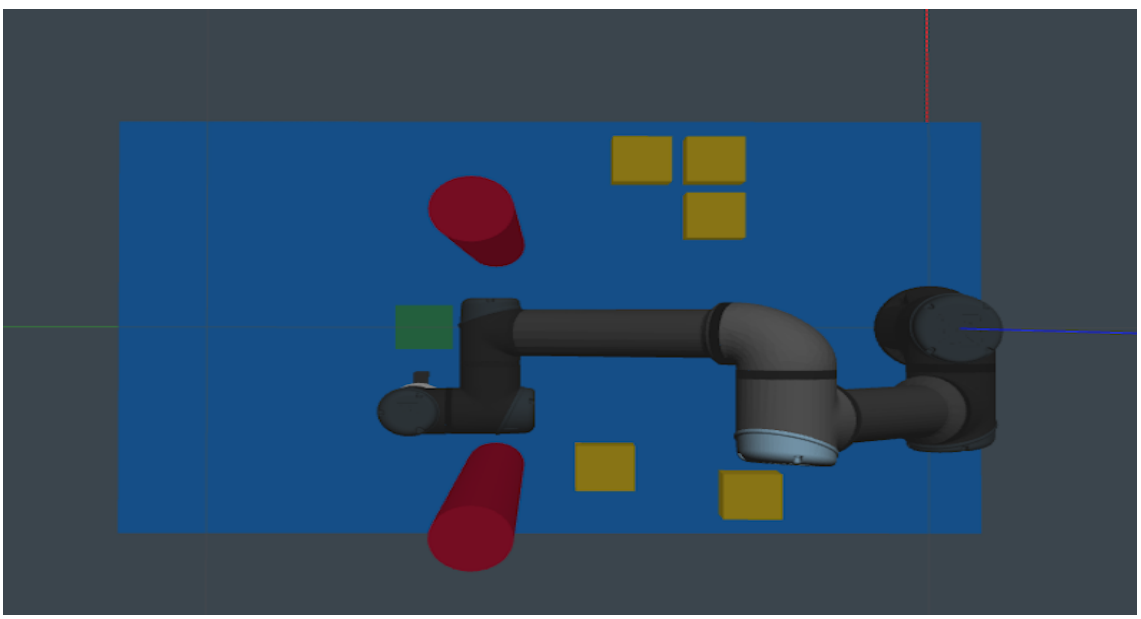

Figure 11: top view of the robot

Question 6

To stack the blocks and build a tower, the robot arm is set to pick up each block and return to home position, then put down the blocks at the goal position between two obstacle pillars. The coordinates of each block and the goal position is shown in the figure below.

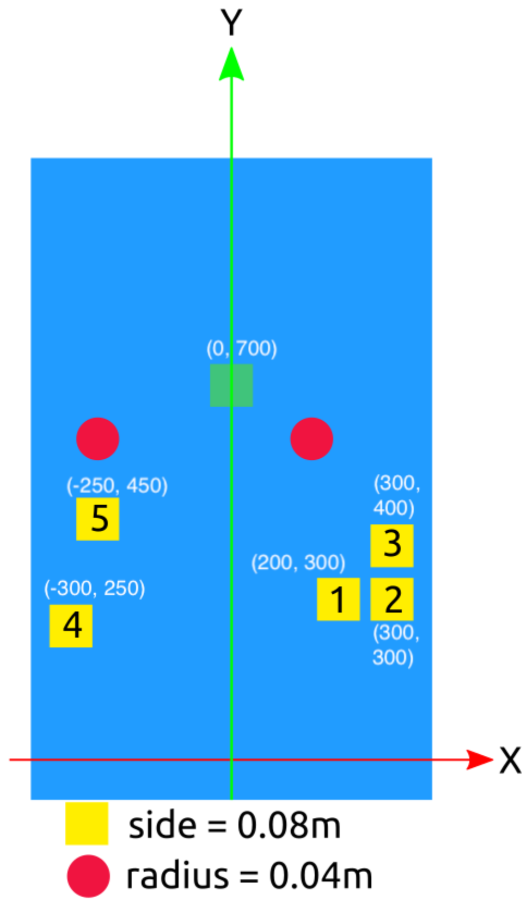

Figure 12: Coordinates of blocks and goal position in mm

The perpendicular distance L_0 from base to the end effector along center line can be calculated by Pythagorean theorem:

$$L_0 = \sqrt{d^2 + offset^2}$$

where d is the distance between base and the end effector, offset equals to $134mm$.

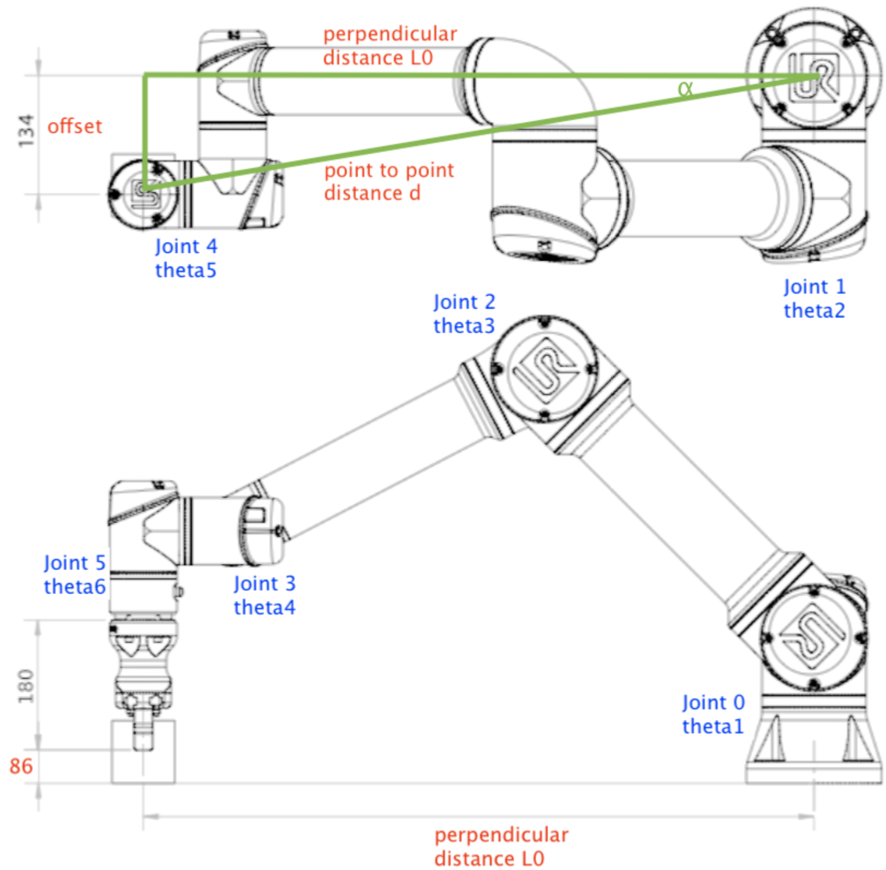

Figure 13: Robot arm pose

With the same method as in question 5, where $L_0 = 700mm$, joint angles of joints 1, 2 and 3 (i.e. $\theta_2, \theta_3 and \theta_4$) can be calculated. θ_5 remains $-90°$ to keep the tips vertically down. However, as the links no longer align with the center line, θ_1 of joint 0 is not necessarily 90°. Using the pythagorean triple, α can be calculated, thus

$$\begin{aligned}\theta_1 &= cos^{-1}(x/d) - \alpha \\ &= cos^{-1}(x/d) - cos^{-1}(L_0/d)\end{aligned}$$

In order to grip the block, the gripper should be aligned with the block, therefore

$$\theta_6 = \theta_1 - 90$$

In simulation, it is found that the robot arm cannot reach the goal position with the joint angles calculated, as the obstacle pillar in quadrant 1 blocks the transform. Since the pillar in quadrant 2 has more space away from the goal position, the robot arm can be mirrored about center line to place the blocks. The joint angles of the mirrored arm are calculated as:

$$\theta_1 = -(cos^{-1}(x/d) - cos^{-1}(L_0/d))$$
$$\theta_2 = -180 + (\theta_D + \theta_C)$$
$$\theta_3 = -(180 - \theta_A)$$
$$\theta_4 = -\theta_C + \theta_B - 90$$
$$\theta_5 = 90$$
$$\theta_6 = \theta_1 - 90$$

Question 7

The selected language is C++ for this question. To obtain information from bag file, the command rqt_bag is used to visualise the structure of folder structure. The linear velocity inside the folder /odom/twist is represented as linear.x, linear.y and linear.z and the angular velocity is respectively angular.x, angular.y and angular.z respectively.

The position of turtlebot is estimated by using the Euler differentiation rule as following and the data is published to toipc odometry.

$$Xposition = Xprevious + V_x * cos(\theta) * dt$$

$$Yposition = Yprevious + V_y * cos(\theta) * dt$$

$$V_x = linear.x * cos(\theta) - linear.y * sin(\theta)$$

$$V_y = linear.x * sin(\theta) + linear.y * sin(\theta)$$

$$\theta = angular.z * dt$$

As for this case, the robot didn't start moving from its origin, so the starting point of the robot needs to be manually set to correctly represent the estimated path.

To obtain the actual position of turtlebot, the data under file odom/pose/pose are attached and published. After integration of the estimated data, the estimated path and real path are then plotted on MATLAB. The plot of estimated position is shown below. As can be seen, the alienation happens mostly when the robot turns. It happens because the sensor measurements, especially for angular, may contain error. When the robot turns, the error may accumulate through time.

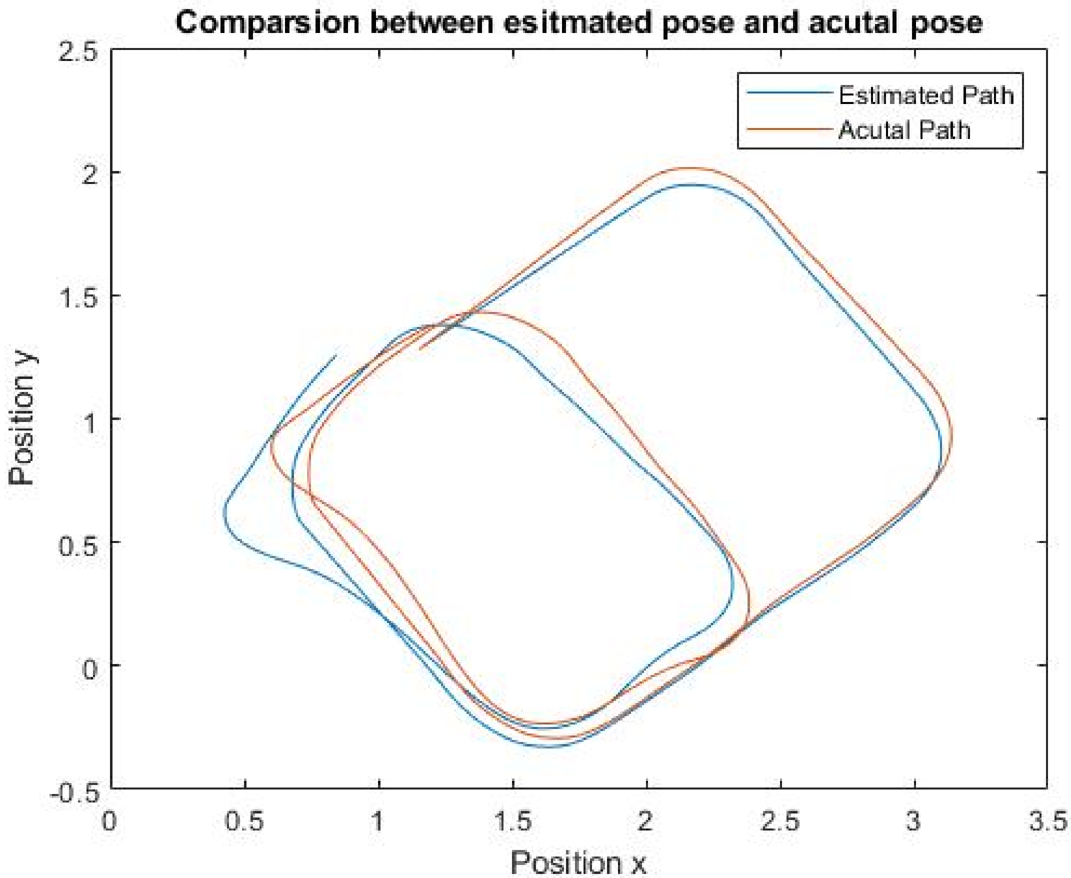

Figure 14: Comparison of two paths

Question 8

(a) The position of the center of mass of link 1 is:

$$x_1 = 0.5l_1c_1$$

$$y_1 = 0.5l_1s_1$$

The velocity is calculated by differentiating its position:

$$\dot{x}_1 = -0.5l_1s_1\dot{\theta}_1$$

$$\dot{y}_1 = 0.5l_1c_1\dot{\theta}_1$$

$$V_1^2 = \dot{x}_1^2 + \dot{y}_1^2$$

Similarly, the position of the center of mass of link 2 is:

$$x_2 = l_1c_1 + 0.5l_2c_{12}$$

$$y_2 = l_1s_1 + 0.5l_2s_{12}$$

The velocity can be calculated:

$$\dot{x}_2 = -l_1s_1\dot{\theta}_1 - 0.5l_2s_{12}(\dot{\theta}_1 + \dot{\theta}_2)$$

$$\dot{y}_2 = l_1c_1\dot{\theta}_1 + 0.5l_2c_{12}(\dot{\theta}_1 + \dot{\theta}_2)$$

$$V_2^2 = \dot{x}_2^2 + \dot{y}_2^2$$

The total kinetic energy of the system is:

$$K = K_1 + K_2 = [\frac{1}{2}I_1\dot{\theta}_1^2 + \frac{1}{2}m_2V_1^2] + [\frac{1}{2}I_2(\dot{\theta}_1 + \dot{\theta}_2)^2 + \frac{1}{2}m_2V_2^2]$$

Substituting V_D into K, we get

$$K = \dot{\theta}_1^2(\frac{1}{2}I_1 + \frac{1}{2}I_2 + \frac{1}{8}m_1l_1^2 + \frac{1}{8}m_2l_2^2 + \frac{1}{2}m_2l_1^2 + \frac{1}{2}m_2l_1l_2c_2) + \dot{\theta}_2^2(\frac{1}{2}I_2 + \frac{1}{8}m_2l_2^2) + \dot{\theta}_1^2\dot{\theta}_2^2(I_2 + \frac{1}{4}m_2l_2^2 + \frac{1}{2}m_2l_1l_2c_2)$$

The potential energy of the system is:

good!

$$P = \frac{1}{2}m_1gl_1s_1 + m_2g(l_1s_1 + \frac{1}{2}l_2s_{12})$$

The Lagrangian of the system is:

$$L = K - P = \dot{\theta}_1^2(\frac{1}{2}I_1 + \frac{1}{2}I_2 + \frac{1}{8}m_1l_1^2 + \frac{1}{8}m_2l_2^2 + \frac{1}{2}m_2l_1^2 + \frac{1}{2}m_2l_1l_2c_2) + \dot{\theta}_2^2(\frac{1}{2}I_2 + \frac{1}{8}m_2l_2^2) + \dot{\theta}_1^2\dot{\theta}_2^2(I_2 + \frac{1}{4}m_2l_2^2 + \frac{1}{2}m_2l_1l_2c_2) - \frac{1}{2}m_1gl_1s_1 - m_2g(l_1s_1 + \frac{1}{2}l_2s_{12})$$

Taking the derivatives of the Lagrangian we can compute the torque of the manipulator:

$$T_i = \frac{\delta}{\delta t}(\frac{\delta L}{\delta \dot{\theta}_i}) - \frac{\delta L}{\delta \theta_i}$$

The equations of motion is therefore:

You should show your calculations of the partial and full

$$T_1 = \ddot{\theta}_1(I_1 + I_2 + \frac{1}{4}m_1l_1^2 + \frac{1}{4}m_2l_2^2 + m_2l_1^2 + m_2l_1l_2c_2) + \ddot{\theta}_2(I_2 + \frac{1}{4}m_2l_2^2 + \frac{1}{2}m_2l_1l_2c_2) - \dot{\theta}_1\dot{\theta}_2(m_2l_1l_2s_2) - \dot{\theta}_2^2(\frac{1}{2}m_2l_1l_2s_2) + \frac{1}{2}m_1gl_1c_1 + m_2gl_1c_1 + \frac{1}{2}m_2gl_2c_{12}$$

$$T_2 = \ddot{\theta}_2(I_2 + \frac{1}{4}m_2l_2^2) + \ddot{\theta}_1(I_2 + \frac{1}{4}m_2l_2^2 + \frac{1}{2}m_2l_1l_2c_2) + \dot{\theta}_1^2(\frac{1}{2}m_2l_1l_2s_2) + \frac{1}{2}m_2gl_2c_{12}$$

(b) Based on the equation above, torque T_1 and T_2 should be inputs and the joint angles θ_1 and θ_2 returning as outputs. To build the Simulink model, the equation is rearranged and taking $\ddot{\theta}_1$ and $\ddot{\theta}_2$ as subject. Note that the Simulink block 'second order integrator' takes $\ddot{\theta}_1/\ddot{\theta}_2$ as input and output $\dot{\theta}_1/\dot{\theta}_2$ and θ_1/θ_2. As m_1/m_2, l_1/l_2, I_1/I_2 are provided in the questions, this can be simplified in the 'Euler constant' block.The rest of the model is basic mathematics operation.

(i) No torque applied

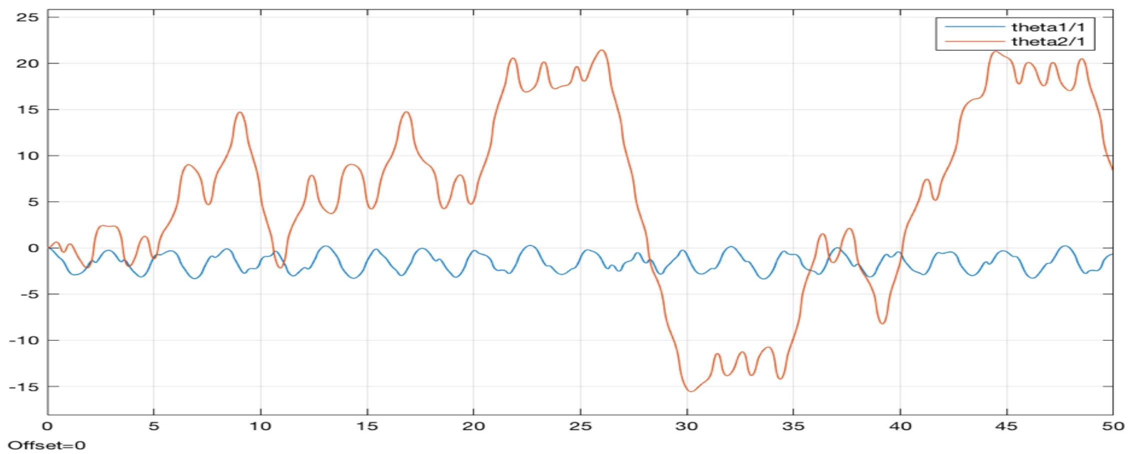

Figure 15: θ_1 and θ_2 behaviour with no torque

Plot of total energy.

(ii) Small torques (T = 5) applied

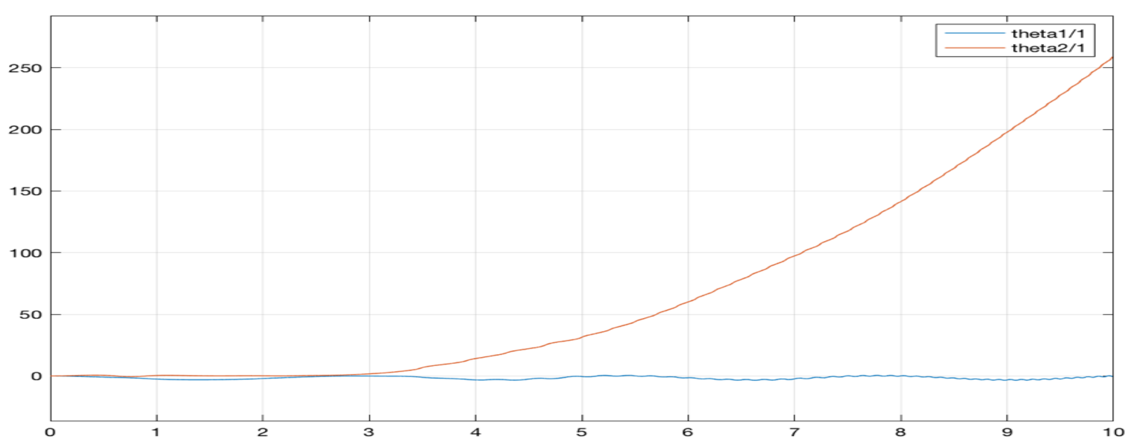

Figure 16: θ_1 and θ_2 behaviour with small torques

(c) To improve the performance of the two-link manipulator,two PID controllers have been applied to the system. The results shown how the joint angle θ_1 and θ_2 behave when holding horizontally and holding in some other angles.

Setting both angles reference to $0°$ i.e holding horizontally

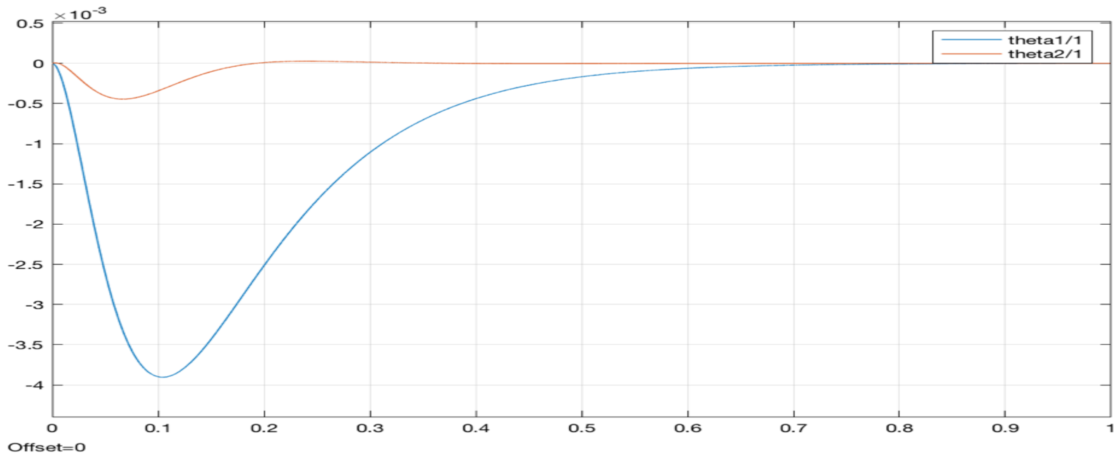

Figure 17: θ_1 and θ_2 behaviour under PID controller

Setting both angles reference to 1 ($57.29°$)

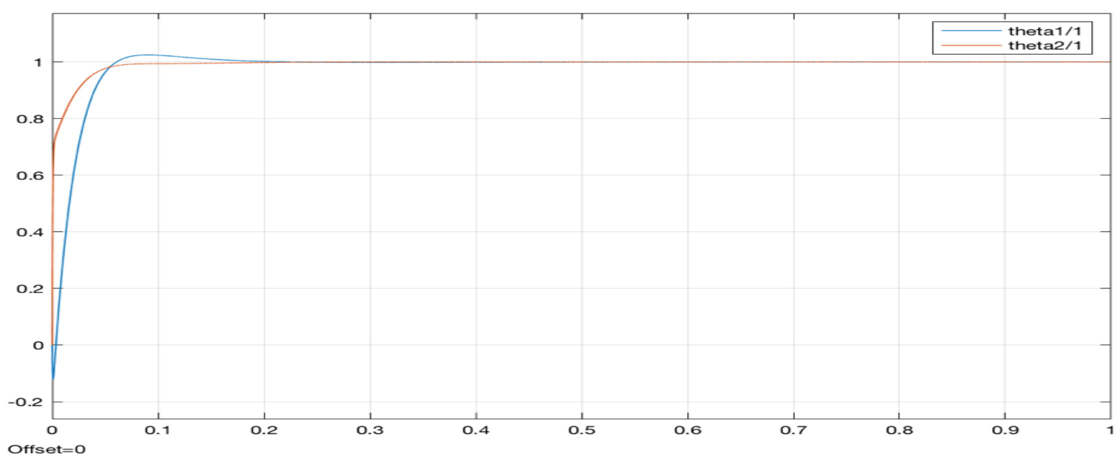

Figure 18: θ_1 and θ_2 behaviour under PID controller

Setting $\theta_1 = 180°$ and $\theta_2 = 0°$ i.e having the end point tracing out some trajectory

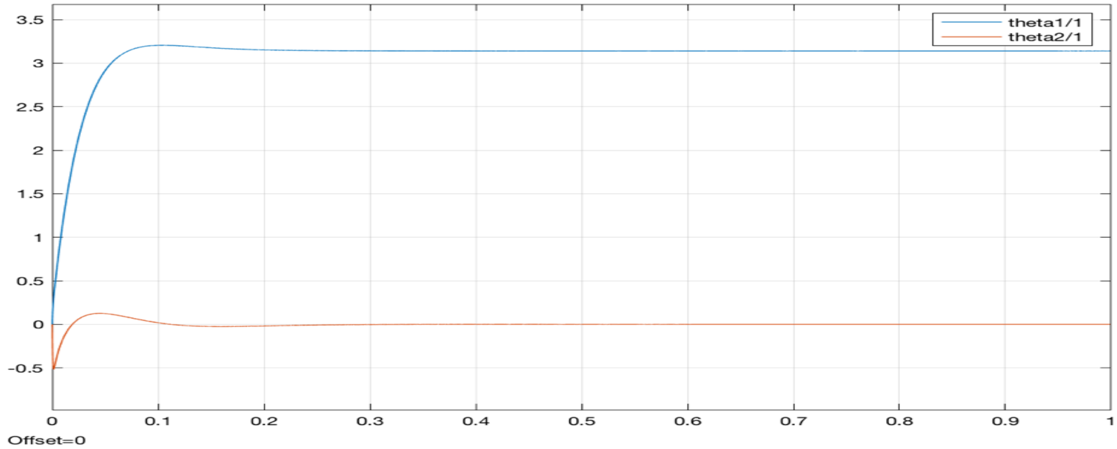

Figure 19: θ_1 and θ_2 behaviour under PID controller

(d) As the simulator and controller can shown how much torque should be use to move the manipulator, the motors with specific torque range can be designed. If the torque is too large, the motion of the robot will be unstable. The below figure shown how θ_1 performance is affected when larger torque applied. Choosing the motor with small velocity/RPM should be enough to drive this two-link manipulator in more efficient way.

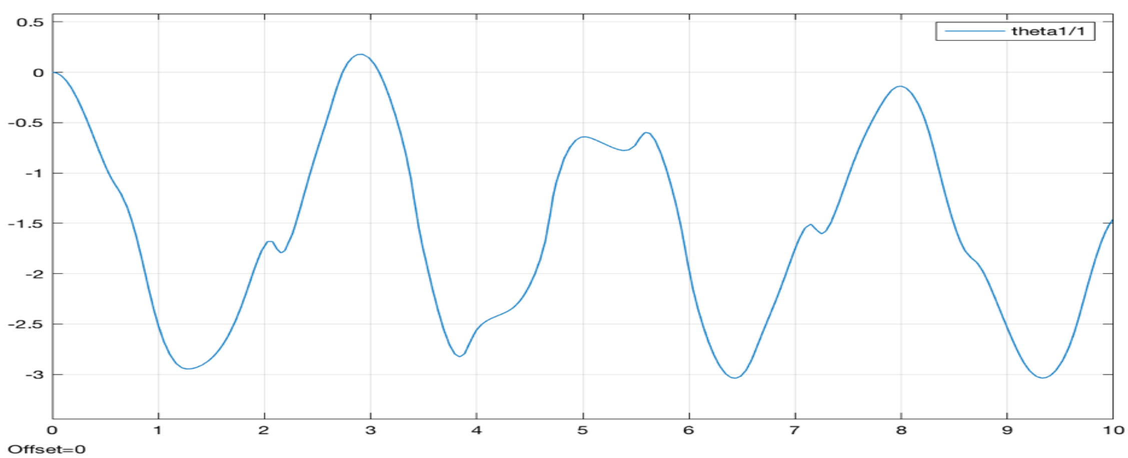

Figure 20: θ_1 with T =1

I don't really know what you are showing me in these figures.

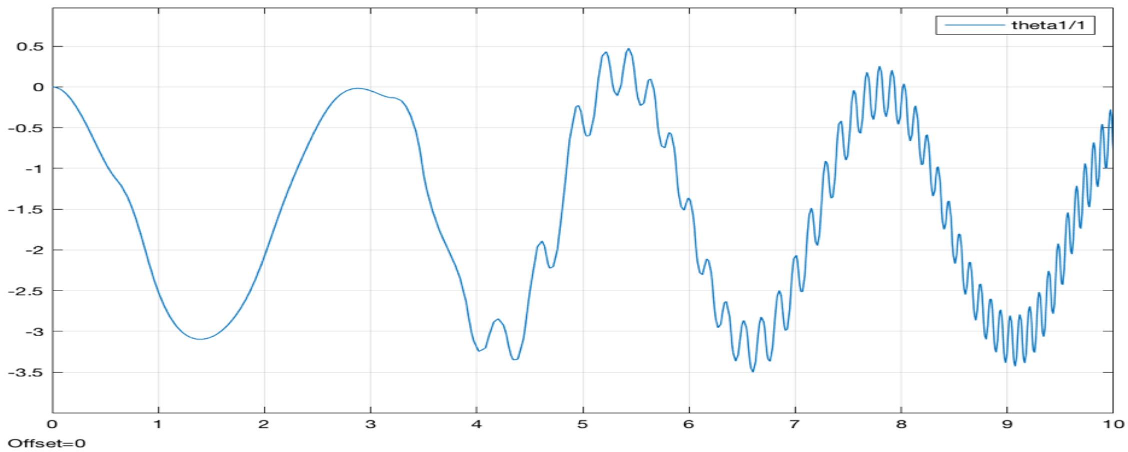

Figure 21: θ_1 with T =5

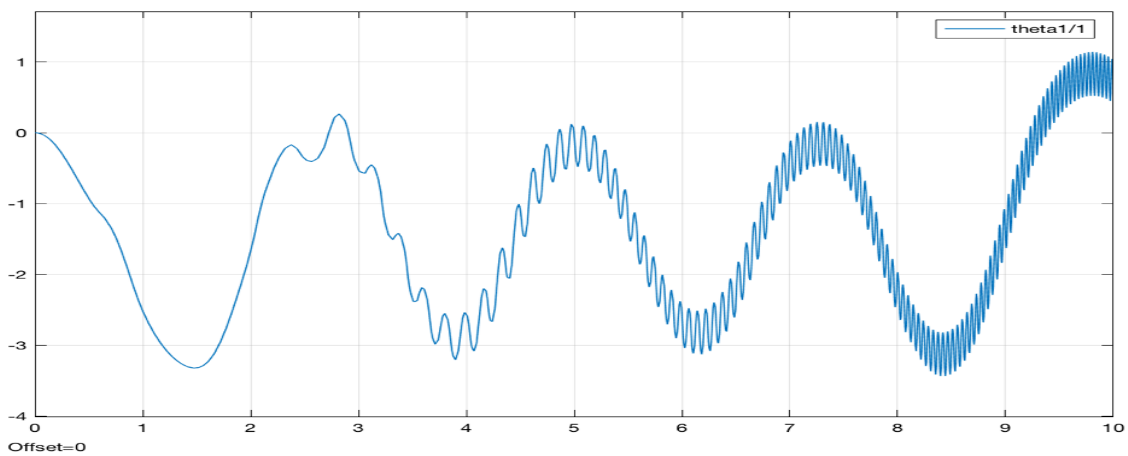

Figure 22: θ_1 with T =10

Simulink Model

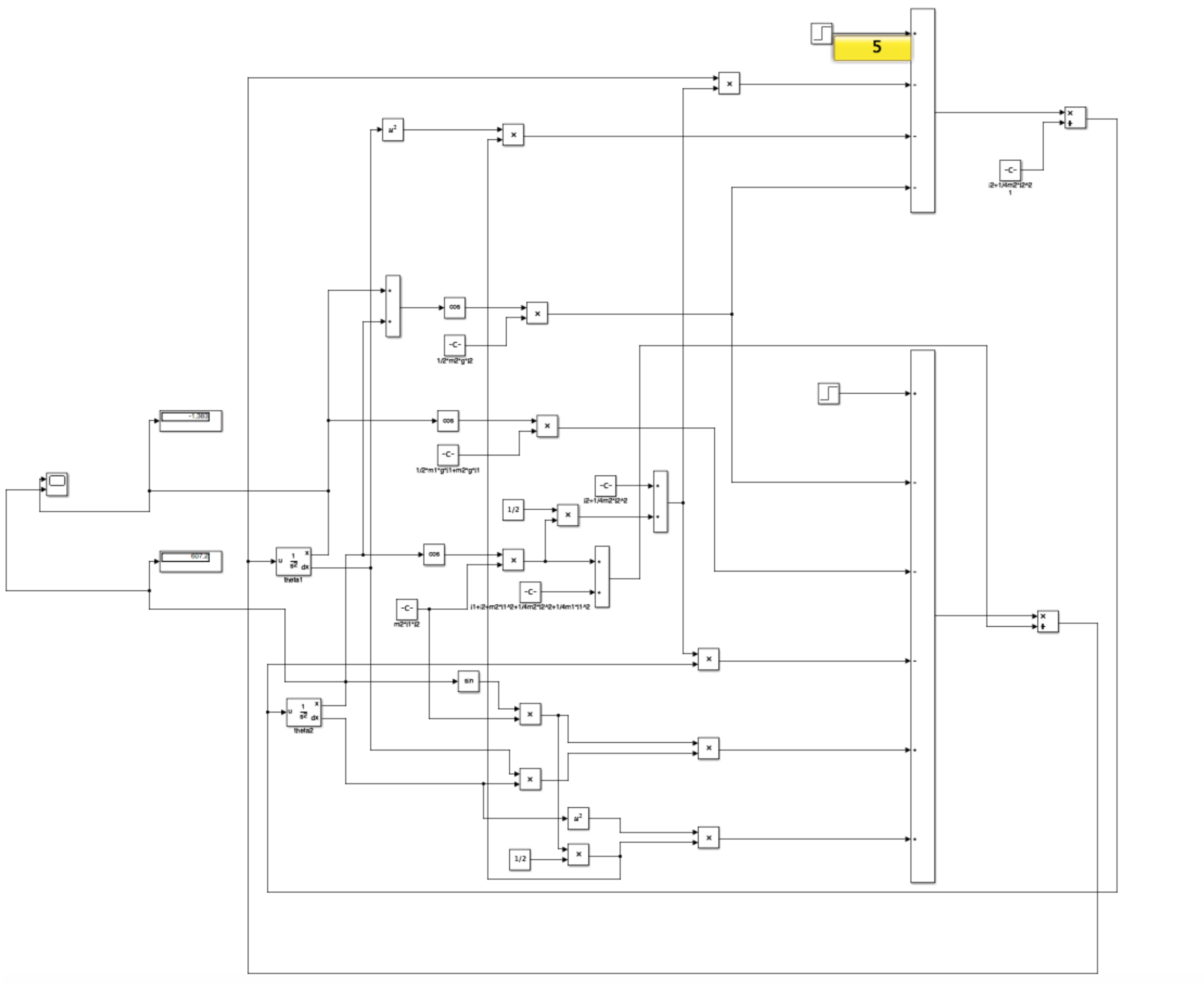

Figure 23: Motion equation modelling using Simulink

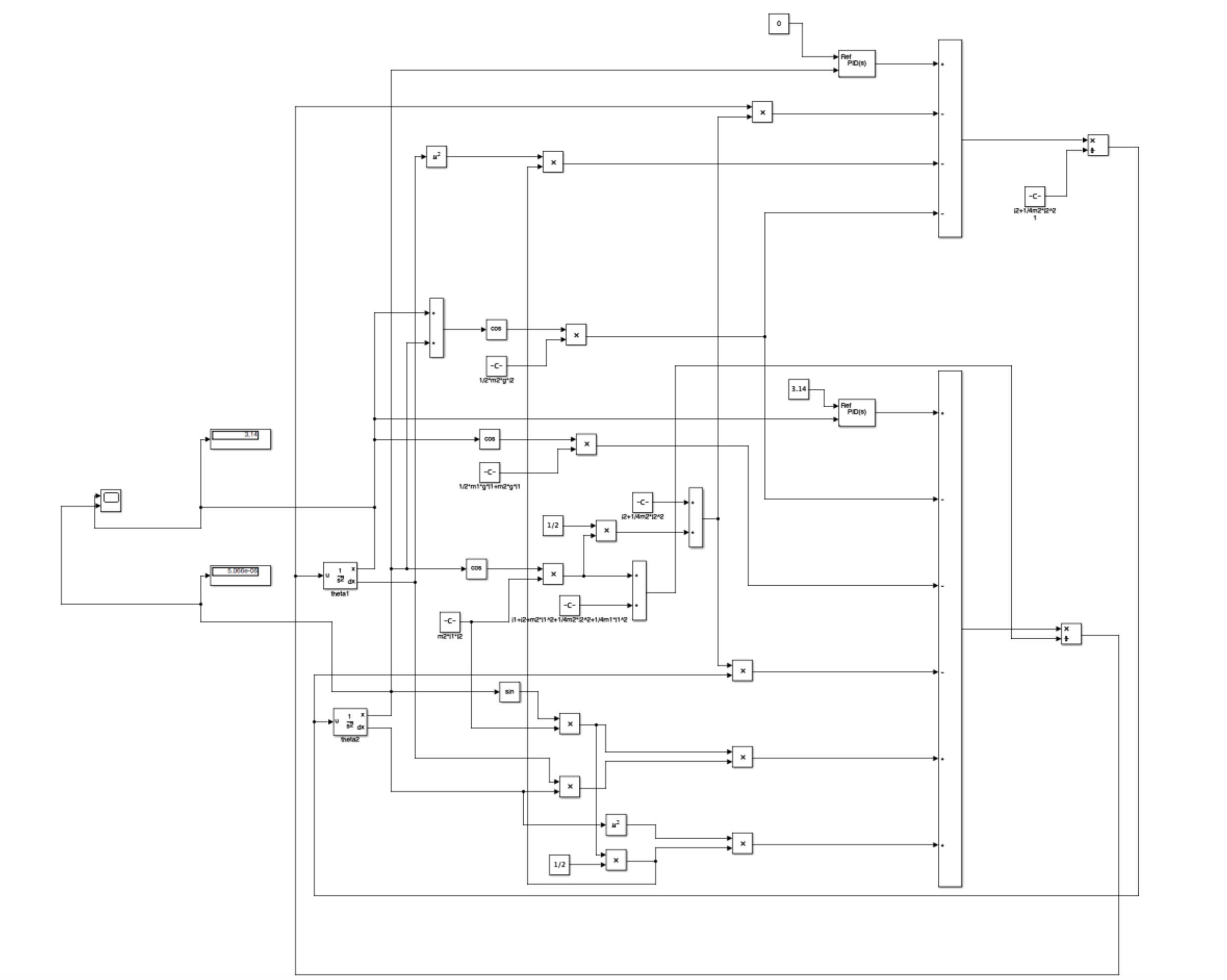

Figure 24: Motion equation model with PID controller