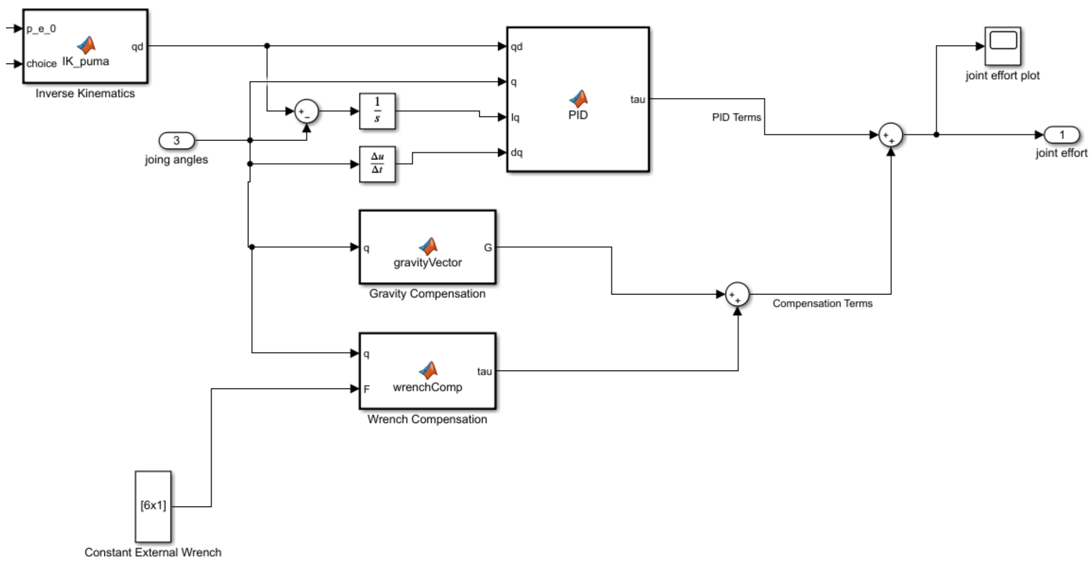
ME 6407 Robotics
Homework 05 Report

Question 1: End-point Control of The Simplified PUMA Robot
[Controller Design]
A **Joint-space PD Controller with Gravity& Wrench Compensation** is implemented for regulating the motion of the robot for both task1 and task2.



As shown in the block diagram, the control flow follows the steps below:
1) The “xedyedzed” task-space plan is first passed into the Inverse Kinematics block to be converted into joint-space plan, q_d.
2) The controller reads in the measured joint angles q at the current time step and compute the Gravity compensation and wrench compensation. The gravity vector is obtained according to Figure 1 (Appendix), and the wrench compensation is the same as we did in Homework 3.
3) The desired joint state (q_d) as well as the actual joint state from measurements (q, dq), are passed into a PID controller with zero KI (therefore just a PD controller).

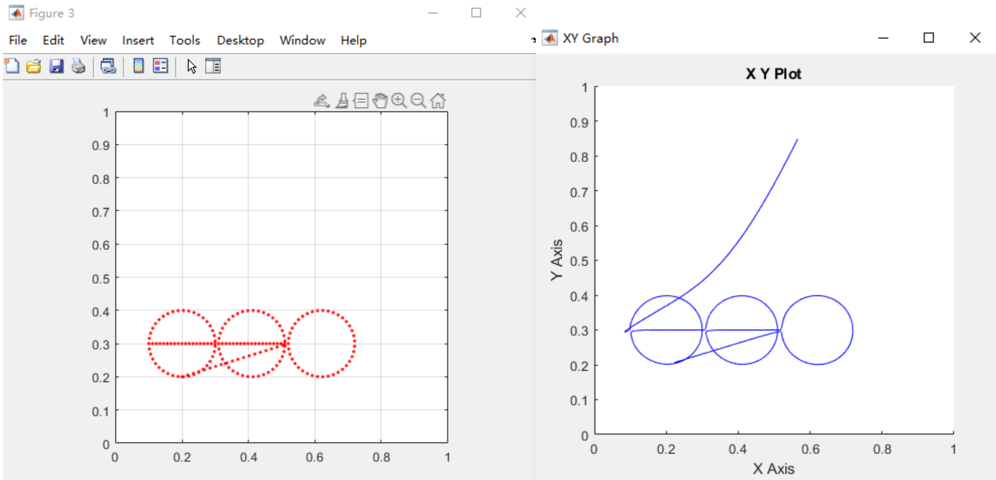
[Controller Gains, Question 1-1]

$$K_p = \begin{bmatrix} 6000 & 0 & 0 \\ 0 & 8000 & 0 \\ 0 & 0 & 6000 \end{bmatrix}$$

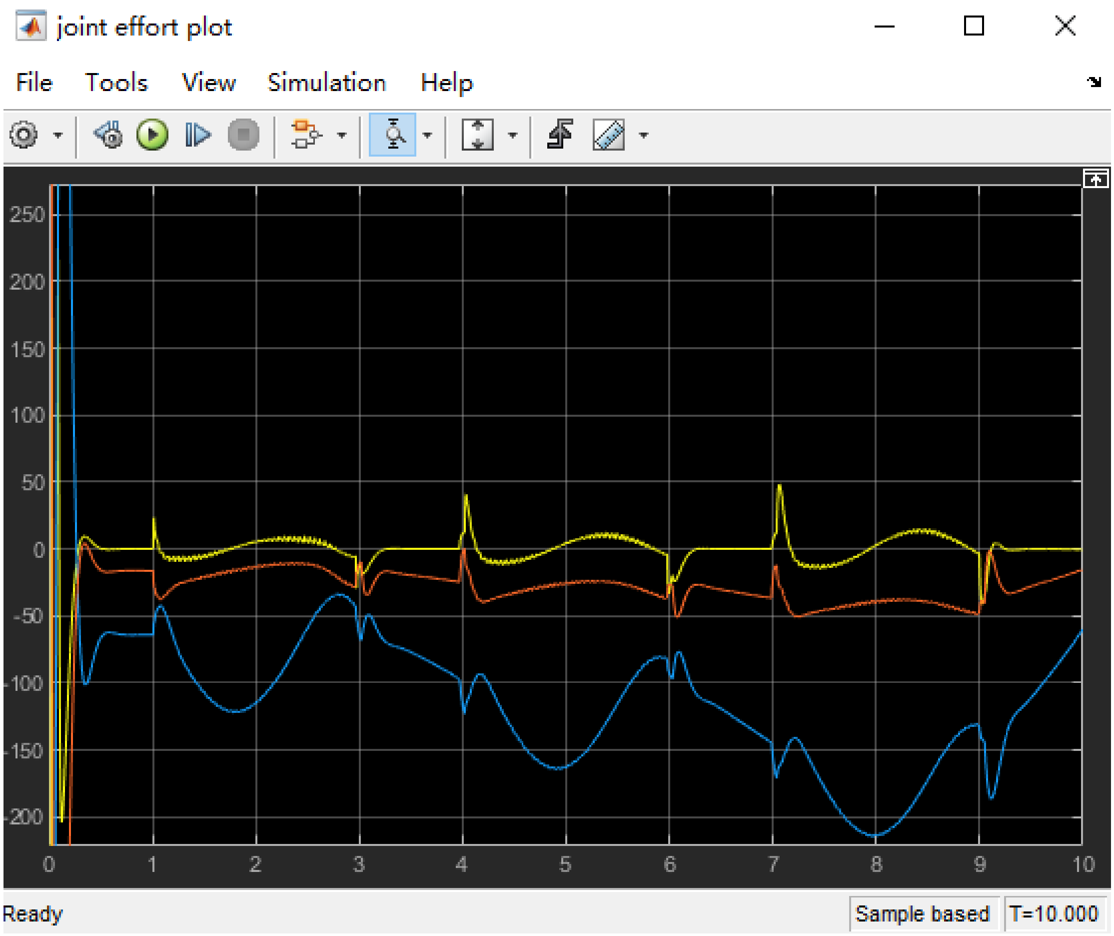
$$K_d = \begin{bmatrix} 400 & 0 & 0 \\ 0 & 600 & 0 \\ 0 & 0 & 400 \end{bmatrix}$$

[Reference and Reproduced Trajectory, Question 1 - 1]

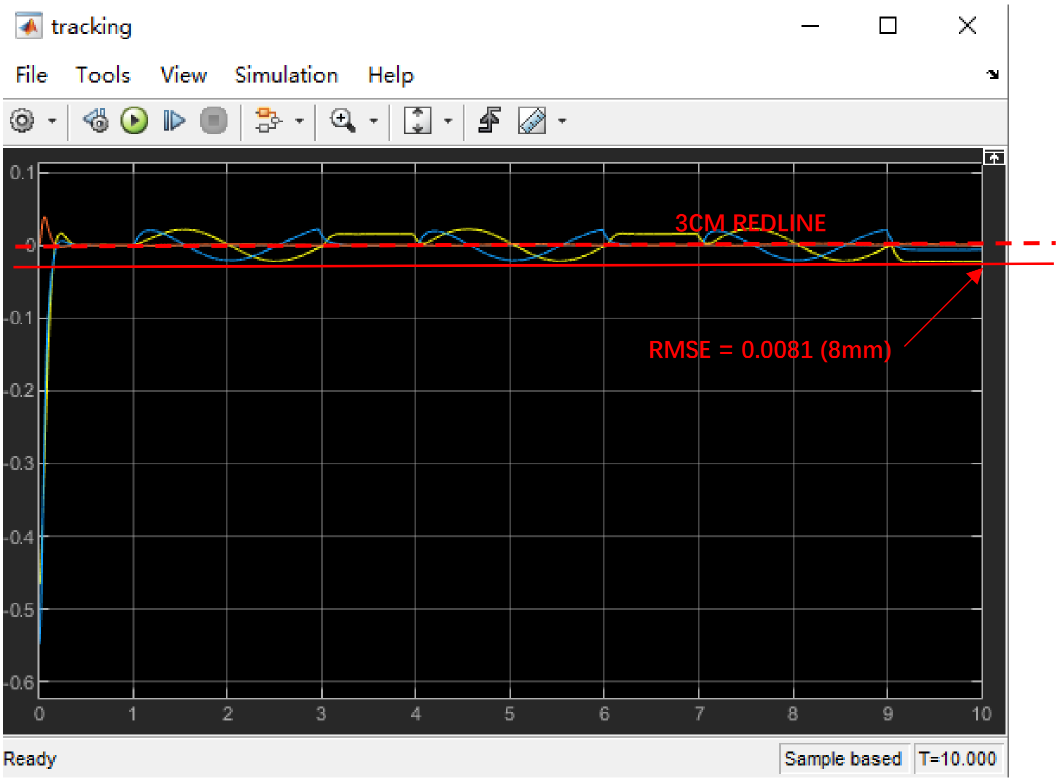
The planned trajectory and actual trajectory are plotted below for comparison:



[Plots, Question 1 - 1]



Joint Effort Versus Time (Please ignore the large transient response)



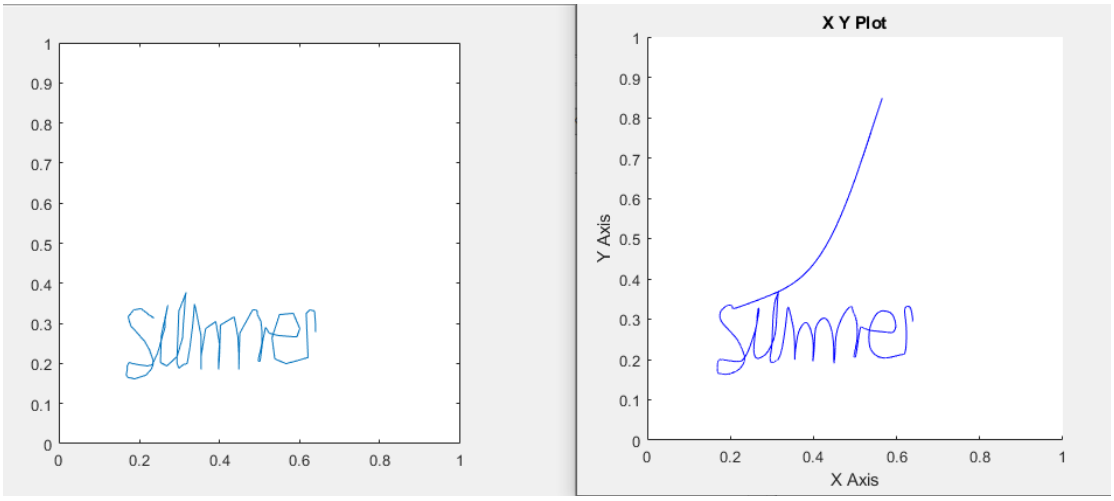
Joint Error Versus Time (Please Ignore Transient Response)

[Controller Gains, Question 1-2]

$$K_p = \begin{bmatrix} 9000 & 0 & 0 \\ 0 & 1200 & 0 \\ 0 & 0 & 9000 \end{bmatrix}$$

$$K_d = \begin{bmatrix} 300 & 0 & 0 \\ 0 & 450 & 0 \\ 0 & 0 & 300 \end{bmatrix}$$

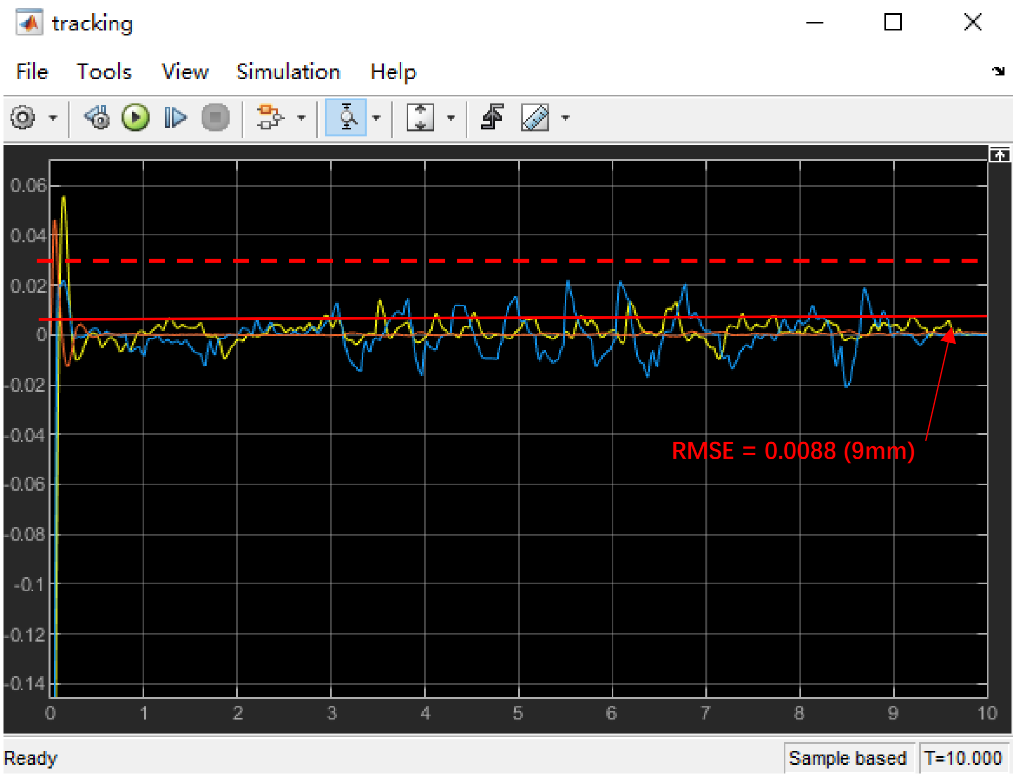
[Reference and Reproduced Trajectory, Question 1-2]



Robot Trying to Draw “summer”

[Plots, Question 1-2]

Joint Effort Versus Time



Joint Error Versus Time

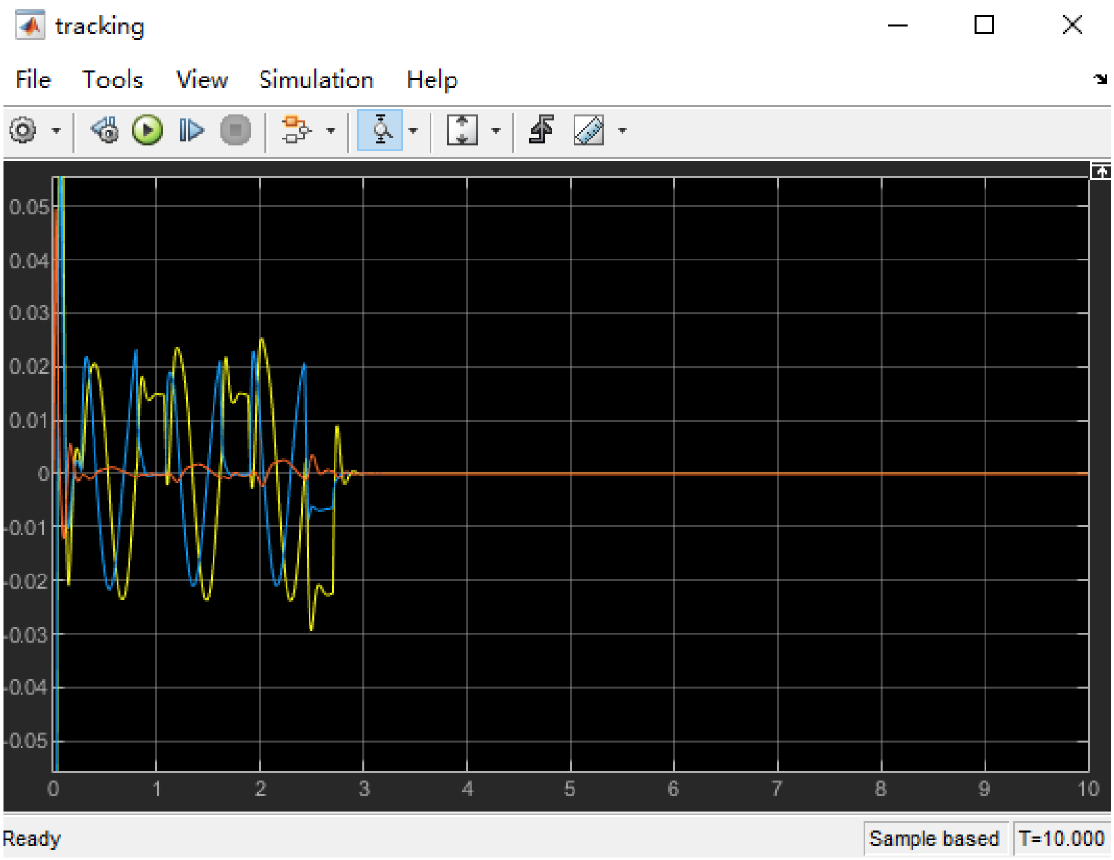
[BONUS: SPEED!!!!!!]

At the end I managed to make the robot complete the circle drawing task **within 2.7s** with the following set of gains (DEFINITELY NOT RECOMMEND!)

$$K_p = \begin{bmatrix} 30000 & 0 & 0 \\ 0 & 40000 & 0 \\ 0 & 0 & 30000 \end{bmatrix}$$

$$K_d = \begin{bmatrix} 540 & 0 & 0 \\ 0 & 810 & 0 \\ 0 & 0 & 540 \end{bmatrix}$$

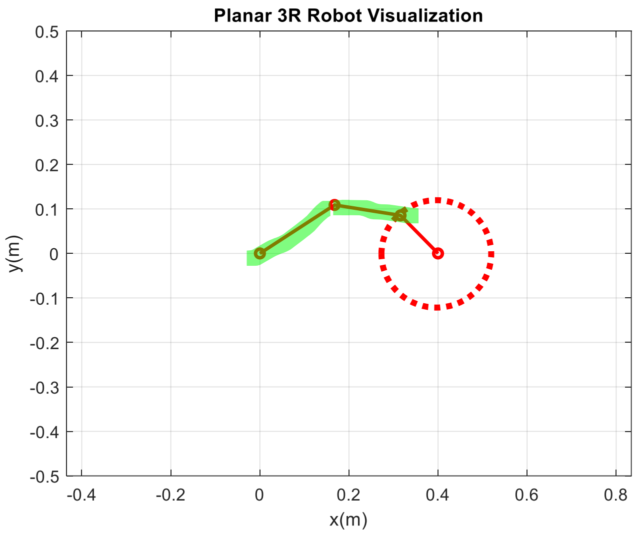
Here is the error plot:



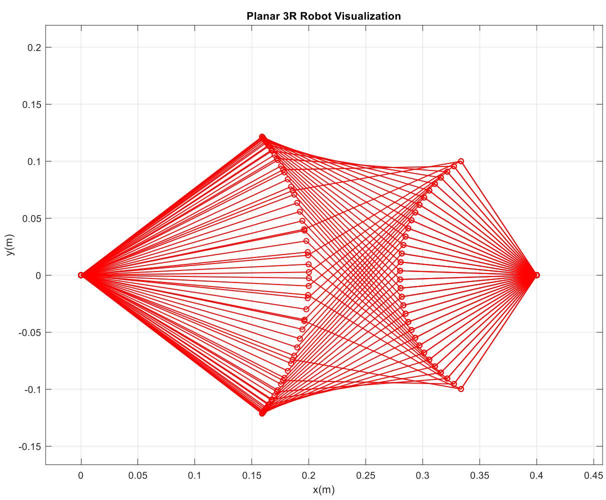
RMSE from 1-10s is calculated to be 0.0091 (9mm).

Question 2: Manipulability of Redundant Planar 3R Robot
[IK Approach]

Analytically obtaining the pose that'll maximize the robot's manipulability while satisfying the end effector position specification appears almost impossible to me after doing a bit of symbolic calculations. Therefore, a NUMERICAL approach to this problem is adopted. First, draw a circle centered at the end effector position p_e^0 with radius $r = a_3 = 0.12m$. Then we can reduce the IK problem to that of a planar 2R robot (as highlighted in green) by specifying the new "end effector" (now the location of the second joint) at a point on the circle we just drew. If we linearly interpolate those points on the circle, this iterative procedure is going to give us a bunch of IK solutions.

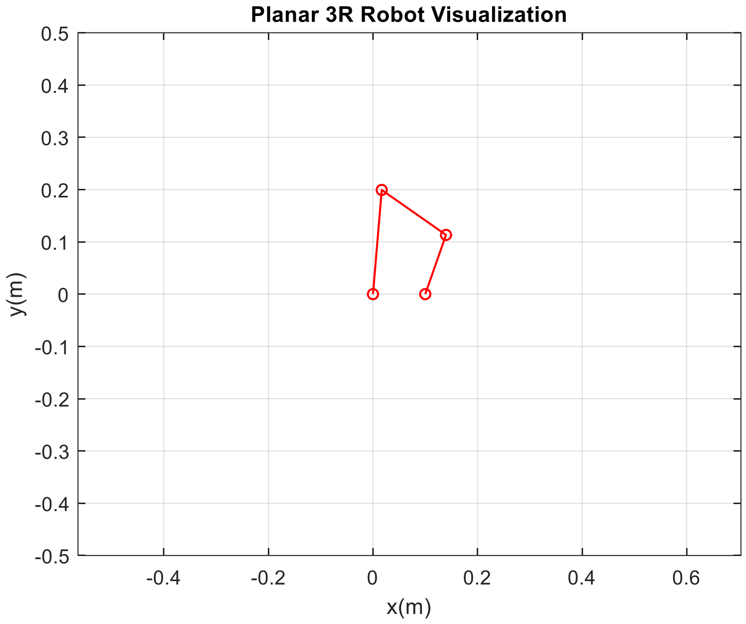


A detailed implementation of this IK procedure is shown in Figure 2. With an interpolation resolution of 100 points per cycle, below is a plot showing all the IK solutions it gives us with the end effector position x = 0.4, y = 0;

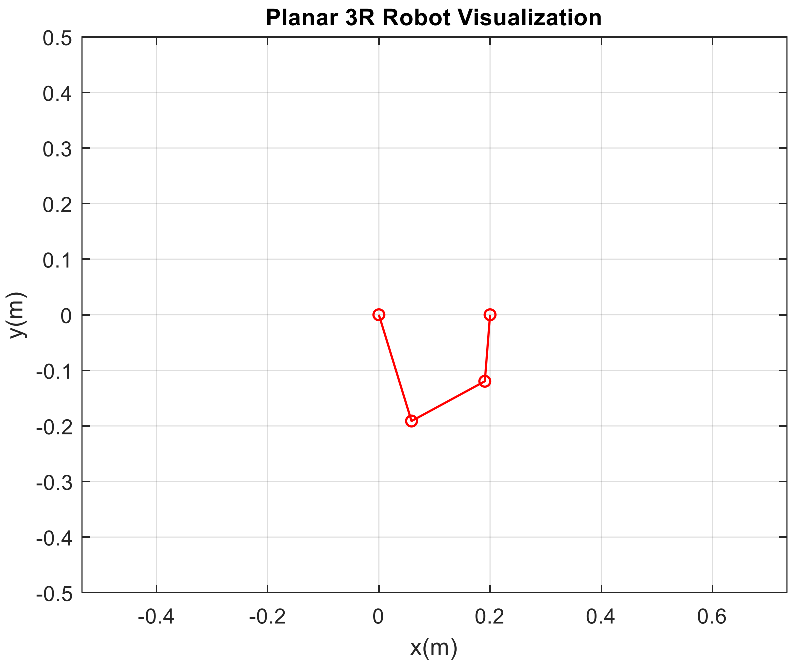


[Result + Screen Shot]

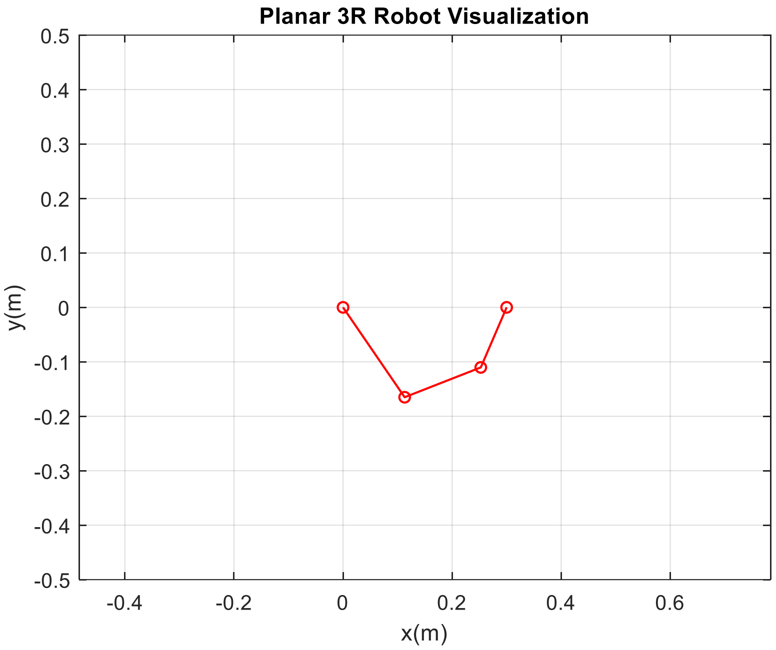
Position (x, y)	Joint1(deg)	Joint2(deg)	Joint3(deg)	Manipulability(w)
(0.1, 0)	85.1853	-120.1857	-74.2983	0.0008257



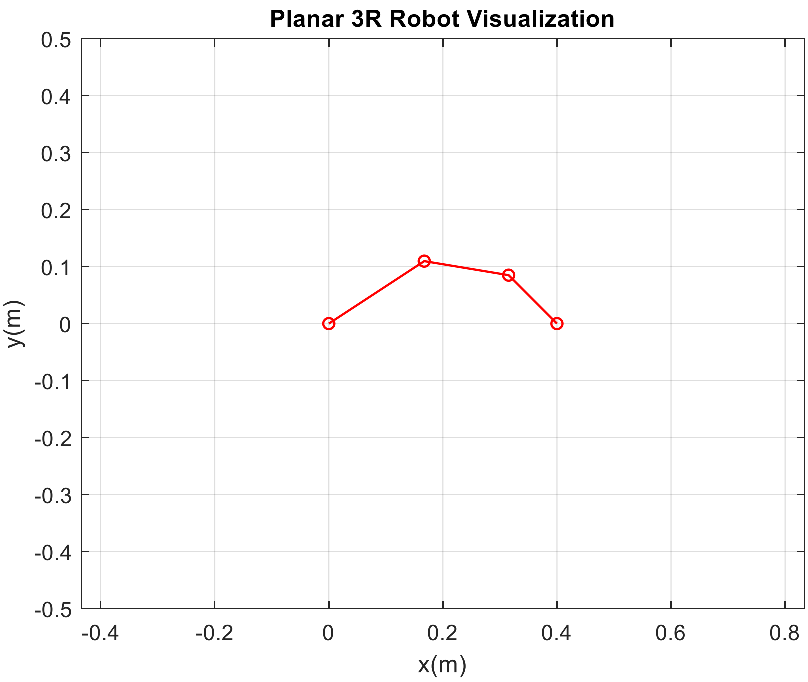
Position (x, y)	Joint1(deg)	Joint2(deg)	Joint3(deg)	Manipulability(w)
(0.2, 0)	-72.9227	101.4138	56.9999	0.0023



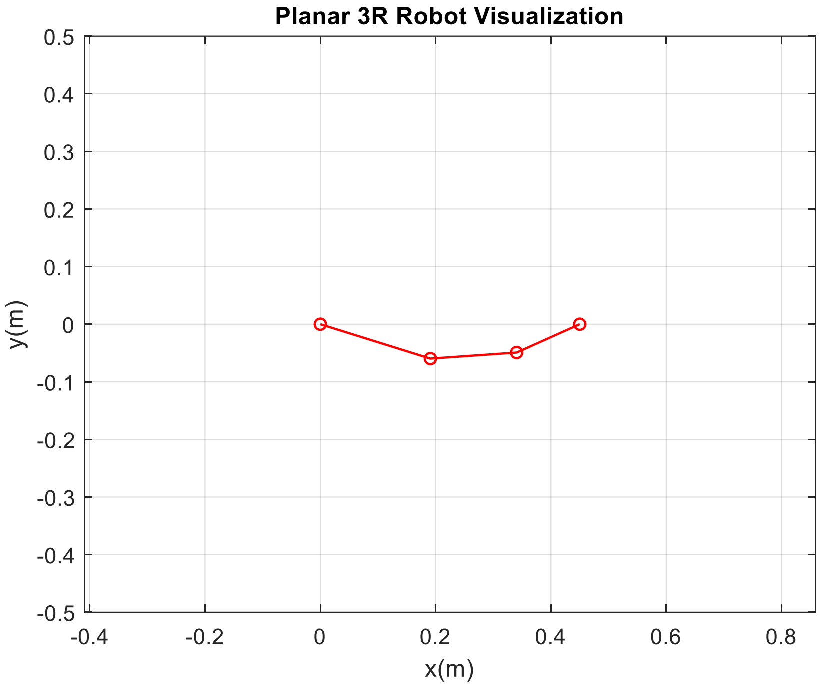
Position (x, y)	Joint1(deg)	Joint2(deg)	Joint3(deg)	Manipulability(w)
(0.3, 0)	-55.6091	77.0370	45.3056	0.0037



Position (x, y)	Joint1(deg)	Joint2(deg)	Joint3(deg)	Manipulability(w)
(0.4, 0)	33.2222	-42.6580	-35.6544	0.0032



Position (x, y)	Joint1(deg)	Joint2(deg)	Joint3(deg)	Manipulability(w)
(0.45, 0)	-17.3602	21.3916	20.1370	0.0012



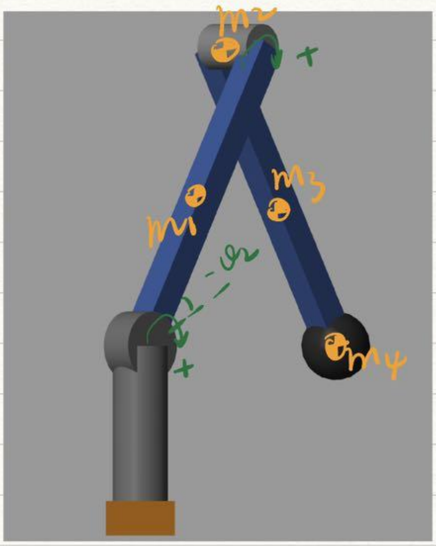
APPENDIX

Dynamics

Potential Energy and Gravity vector.

The potential energy of the PUMA Robot should be a func of θ_2 and θ_3 only.

$$m = \rho \cdot V \quad \checkmark \ \text{Simscape}$$



$$PE_1 = m_1 g \left(\frac{L_a}{2}\right) \sin(-\theta_2)$$

$$PE_2 = m_2 g L_a \sin(-\theta_2)$$

$$PE_3 = m_3 g\left(L_a \sin(-\theta_2) + \left(\frac{L_c}{2} \sin(-\theta_2-\theta_3)\right)\right)$$

$$PE_4 = m_4 g(L_a \sin(-\theta_2)) + (L_c \sin(-\theta_2-\theta_3))$$

$$m_1 = 2700 \cdot (1 \cdot 1/12 \cdot 1/12) = 18.75\text{kg}$$

$$m_2 = 2700 \cdot \pi(0.2/3)^2 \cdot 0.2 = 7.54\text{kg}.$$

$$m_3 = m_1 = 18.75\text{kg}$$

$$m_4 = 1000(4/3)\pi(0.1)^3 = 2.356\text{kg}.$$

$$\bar{q}(1) = \frac{\partial PE}{\partial \theta_1} = 0$$

$$\bar{q}(2) = \frac{\partial PE}{\partial \theta_2} = -4.905(76.042 \cos\theta_2 + 23.462 \cos(\theta_2+\theta_3$$

$$\bar{q}(3) = \frac{\partial PE}{\partial \theta_3} = -115.081 \cos(\theta_2+\theta_3).$$

Figure 1. Derivation of the gravity vector

```matlab
function [poses] = IK_3R(p0e)
    % Joint Length
    a1 = 0.2;
    a2 = 0.15;
    a3 = 0.12;
    precision = 100;
    poses = []; % [theta1; theta2; theta3; w]
    % Calculation
    for searchAngle = linspace(0, 2 * pi, precision)
       p02 = p0e + a3 * [cos(searchAngle); sin(searchAngle)];
       if (norm(p02) <= (a1 + a2))
          % Internal Solution Findable
           for choice = [1, -1]
               q = IK_2R(p02, choice);
              theta1 = q(1);
              theta2 = q(2);
              p2e = p0e - p02;
              theta3 = atan2(p2e(2), p2e(1)) - (theta1 + theta2);
              q = [theta1; theta2; theta3];
              poses = [poses, [q; w_3R(q); choice]];
           end
       end
    end
end

function [q] = IK_2R(p, choice)
    q = zeros(2, 1);
    % Robot Parameters
    l1 = 0.2;
    l2 = 0.15;
    % Derivation
    R = norm(p);
    theta_star = atan2(p(2), p(1));
    gamma = acos((l1^2 + R^2 - l2^2) / (2 * l1 * R)) * choice;
    q(1) = theta_star + gamma;
    pRela = p - [l1 * cos(q(1)); l1 * sin(q(1))];
    q(2) = atan2(pRela(2), pRela(1)) - q(1);
end
```

Figure 2. The Iterative IK Implementation in MATLAB for the Planar 3R Robot