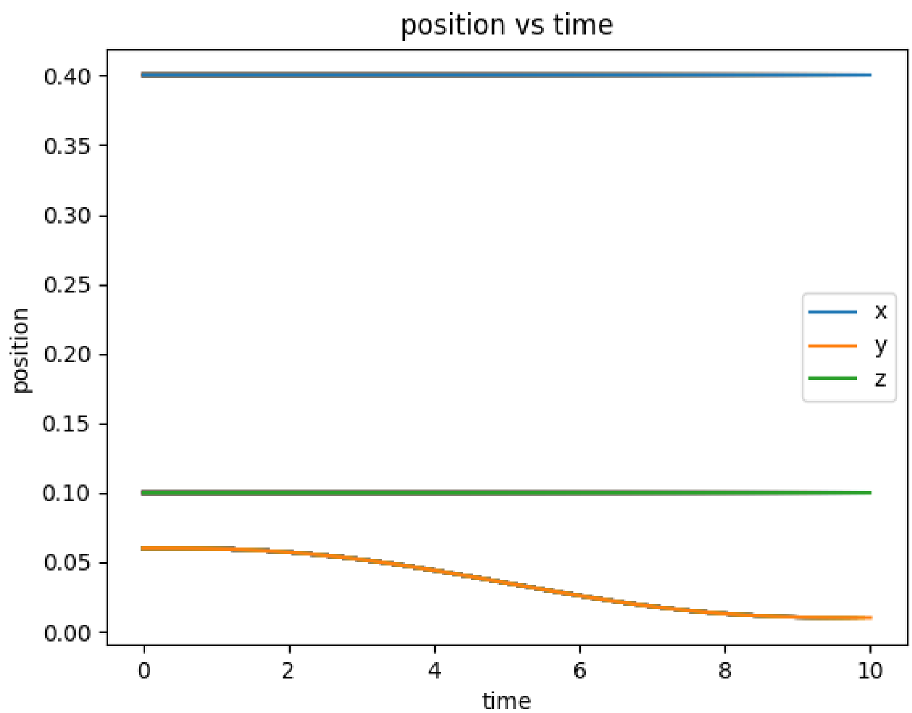
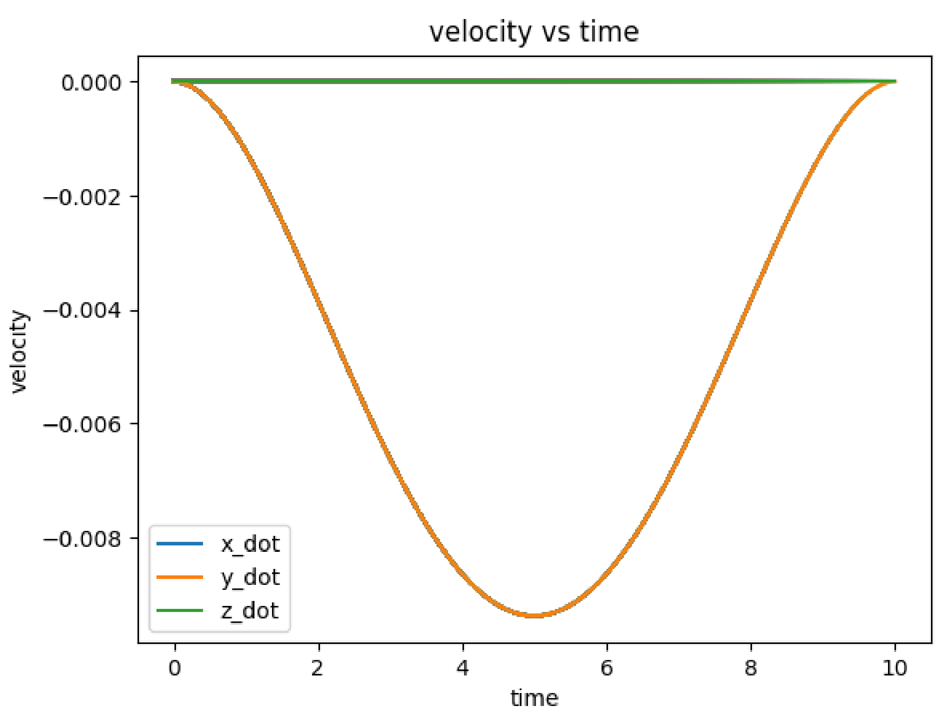
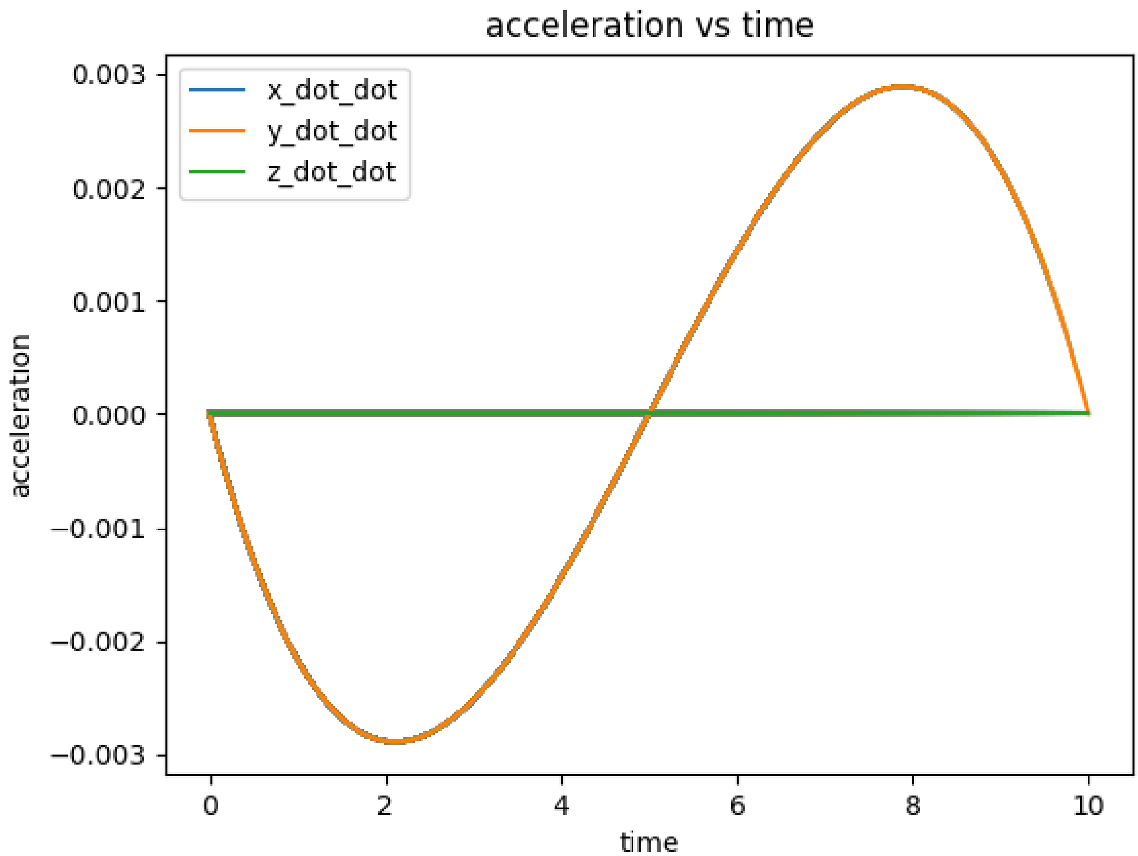
Name: Harsh Mandalia (19110186)
The secret word: DEV

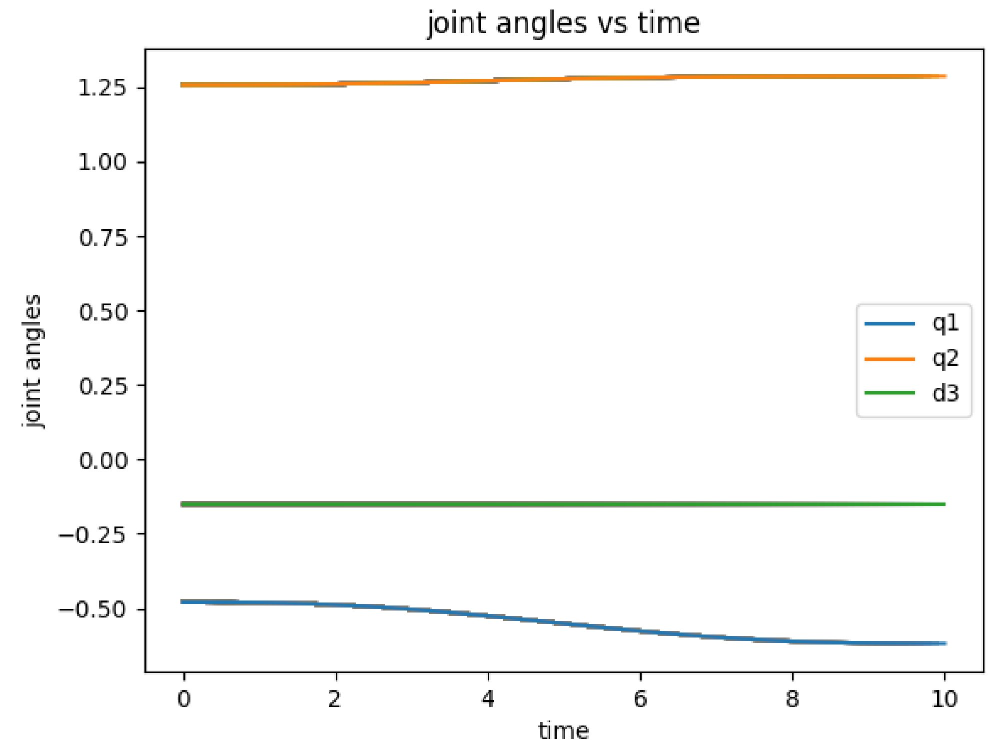
Task 1:

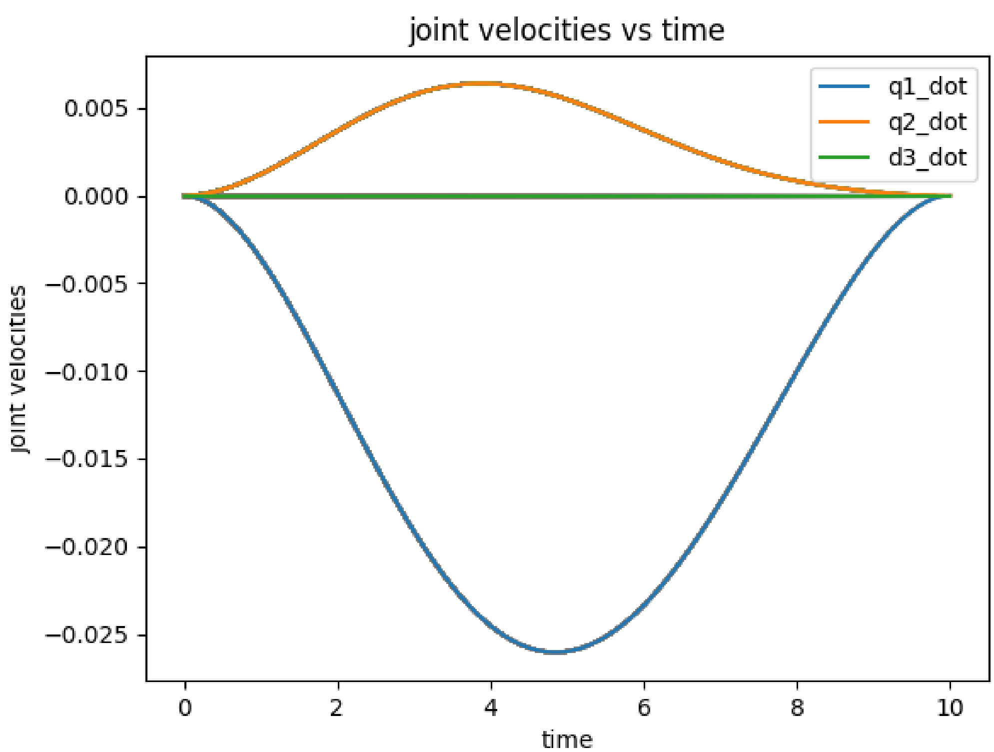


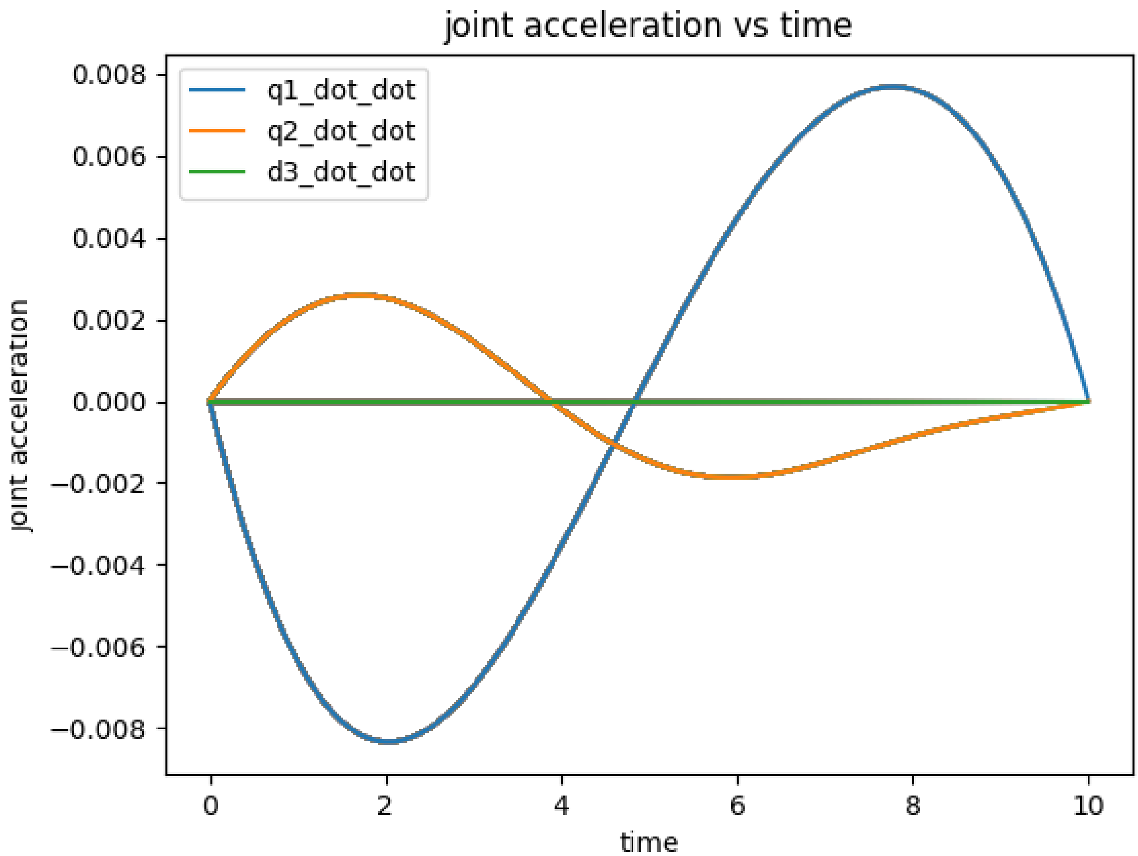



acceleration vs time
x_dot_dot
y_dot_dot
z_dot_dot
0.003
0.002
0.001
0.000
−0.001
−0.002
−0.003
acceleration
0
2
4
6
8
10
time

Task 2:


joint angles vs time
joint angles
time
q1
q2
d3
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
0
2
4
6
8
10


joint velocities vs time
joint velocities
time
q1_dot
q2_dot
d3_dot
0.005
0.000
−0.005
−0.010
−0.015
−0.020
−0.025
0
2
4
6
8
10



To find the joint angular acceleration from end-effector acceleration, I differentiated the end effector's velocities and got the end effector's acceleration in terms of joint acceleration. The image below shows the rough derivation of the equation used in the code.

⟹ Assignment - ~~06~~ 7,

Q - 2, for SCARA,

$$J_v = \begin{bmatrix} -l_3 \sin(q_1+q_2) - l_2 \sin q_1 & -l_3 \sin(q_1+q_2) & 0 \\ l_3 \cos(q_1+q_2) + l_2 \cos q_1 & l_3 \cos(q_1+q_2) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

so, $v_x = \left[-l_3 \sin(q_1+q_2) - l_2 \sin q_1\right] \dot{q}_1 + \left[-l_3 \sin(q_1+q_2)\right] \dot{q}_2$

$\therefore\ a_x = [\quad] \ddot{q}_1 + [\quad] \ddot{q}_2 - l_3 \cos(q_1+q_2)[\dot{q}_1+\dot{q}_2]\dot{q}_2$
$\quad - \left(l_3 \cos(q_1+q_2)[\dot{q}_1+\dot{q}_2] + l_2 \cos q_1 [\dot{q}_1]\right) \dot{q}_1$

similarly $a_y = J_v(2,1)\ddot{q}_1 + J_v(2,2)\ddot{q}_2 - l_3 \sin(q_1+q_2)[\dot{q}_1+\dot{q}_2]\dot{q}_2$
$\quad - \left(l_3 \sin(q_1+q_2)[\dot{q}_1+\dot{q}_2] + l_2 \sin q_1 [\dot{q}_1]\right)\dot{q}_1$

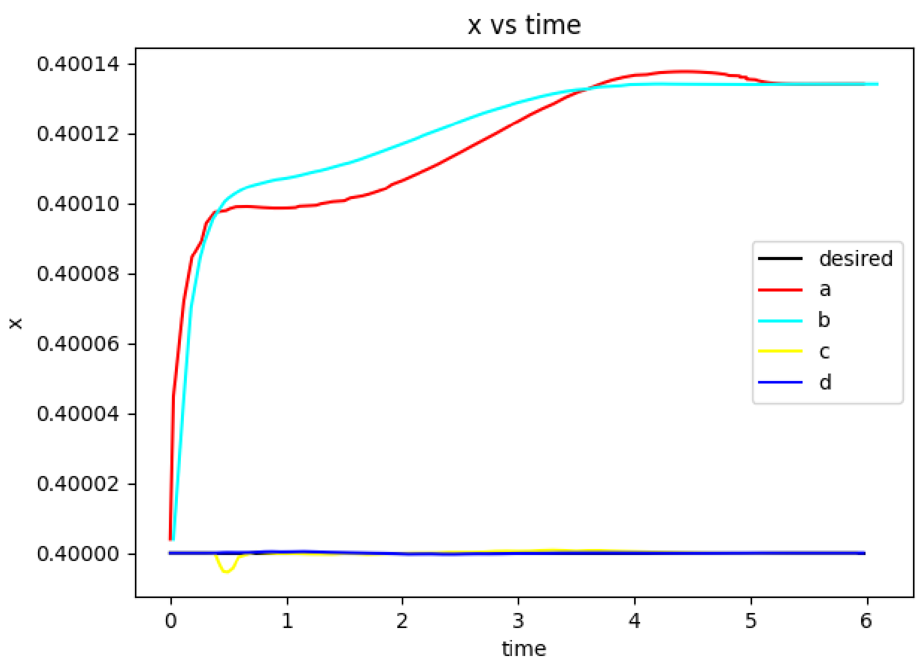
$v_z = \dot{d}_3$

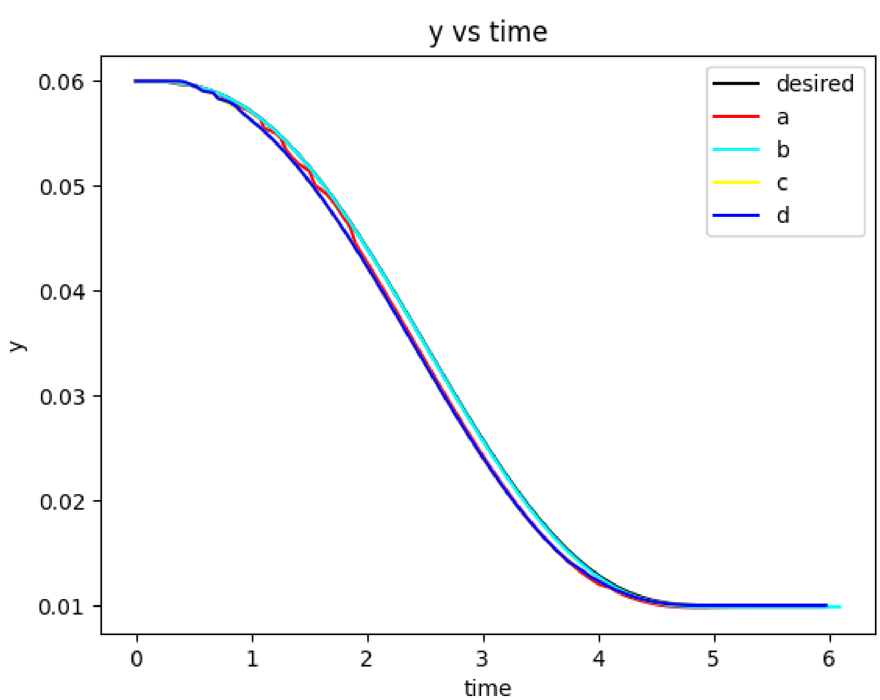
$a_z = \ddot{d}_3$

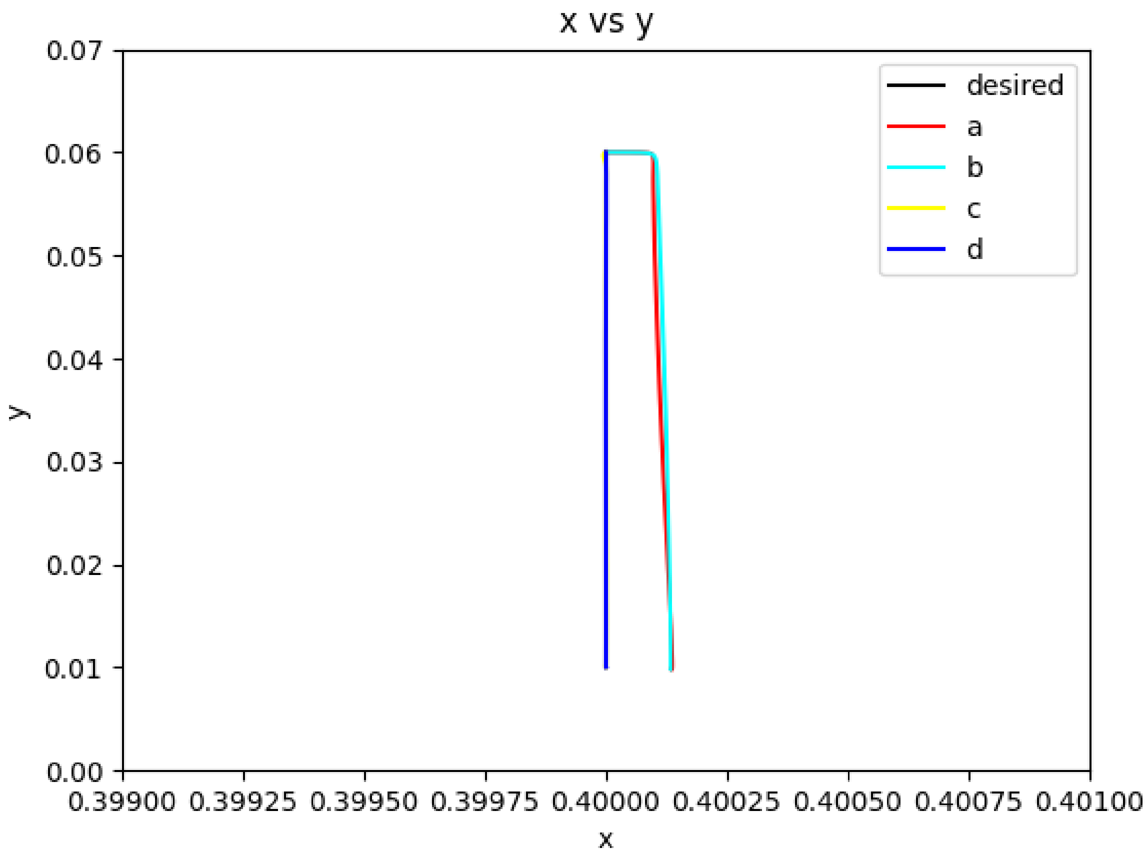
So, $\begin{bmatrix} a_x + l_3 c(q_1+q_2)(\dot{q}_1+\dot{q}_2)\dot{q}_2 + \left\{l_3 c(q_1+q_2)(\dot{q}_1+\dot{q}_2) + l_2 c(q_1)\dot{q}_1\right\}\dot{q}_1 \\ a_y + l_3 s(q_1+q_2)(\dot{q}_1+\dot{q}_2)\dot{q}_2 + \left\{l_3 s(q_1+q_2)(\dot{q}_1+\dot{q}_2) + l_2 s(q_1)\dot{q}_1\right\}\dot{q}_1 \\ a_z \end{bmatrix} = J_v \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \\ \ddot{d}_3 \end{bmatrix}$

Task 3:

I have used the dynamics equations in terms of joint angles(q) and not in terms of motor angles (θ_m). So, I divided motor dynamics by the gear ratio. kp=100, kd=19 is used for all the controllers.



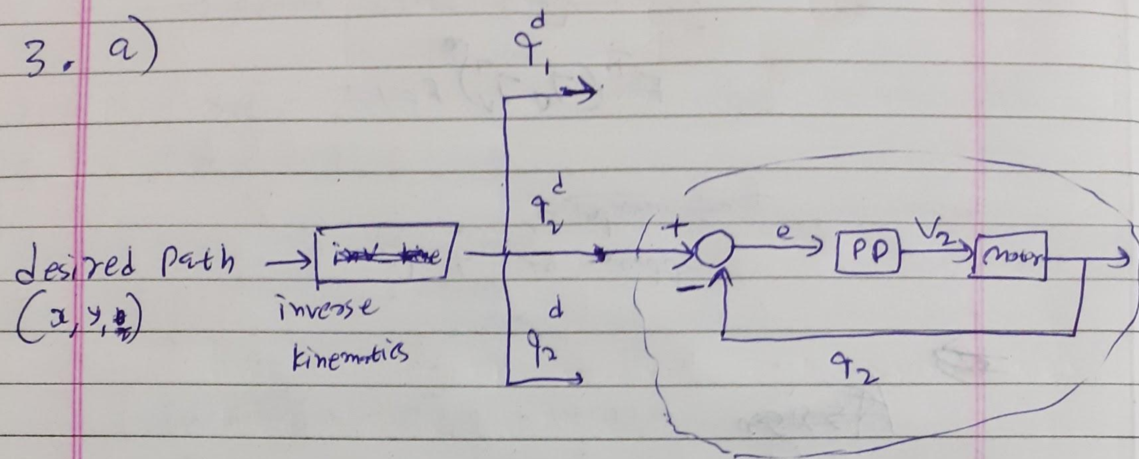



x vs y
desired
a
b
c
d
y
x
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0.39900
0.39925
0.39950
0.39975
0.40000
0.40025
0.40050
0.40075
0.40100

Assignment - 6-7

(~~DVE~~ (DEV)

3. a)



$$V_2 = k_{P_2}(q_2^d - q_2) + k_{D_2}(\dot{q}_2^d - \dot{q}_2)$$

→ similer block diagram will be for first and third joint variable.

$$V_1 = k_{P_1}(q_1^d - q_1) + k_{D_1}(\dot{q}_1^d - \dot{q}_1)$$

$$V_3 = k_{P_3}(q_3^d - q_3) + k_{D_3}(\dot{q}_3^d - \dot{q}_3)$$

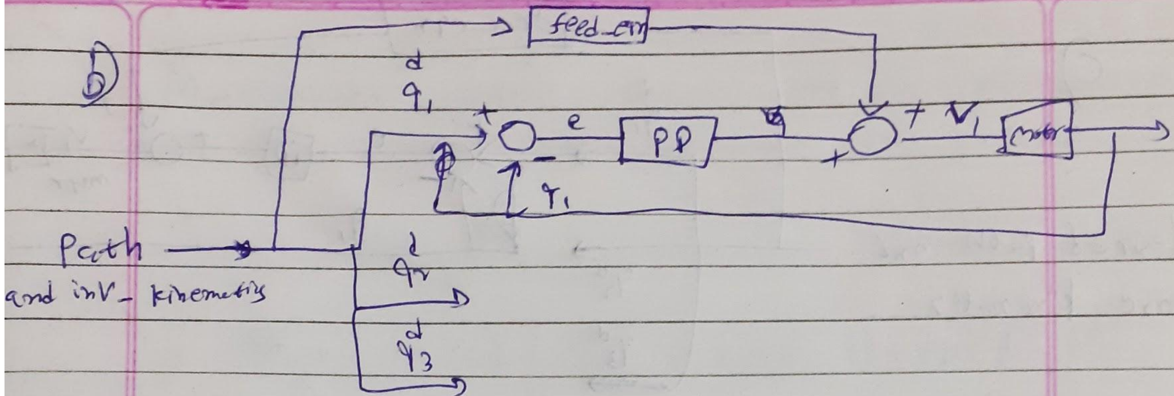
→ $k_{P_1} = k_{P_2} = k_{P_3} = 100$, $k_{D_1} = k_{D_2} = 19 = k_{D_3}$

~~$k_{P_3} = 10$~~ , ~~$k_{D_3} = 19$~~

⇒ from above k_P, k_D, and motor parameters (which are taken 1 for simplicity), we can find w_n and ξ.

so, $w_n = \sqrt{K_P} = 10$, $19 = \dfrac{2\xi(10) - 1}{1}$ ⇒ $\xi = 1$

b)



here, feed_eqn is Motor dynamics and D matrix from Robot dynamics

so, feed_eqn is $\left(\frac{J_m}{r} + D_{ii}\right)\ddot{q} + \left(\frac{B_{eff}}{r}\dot{q}\right) = \frac{K_m V}{R}$

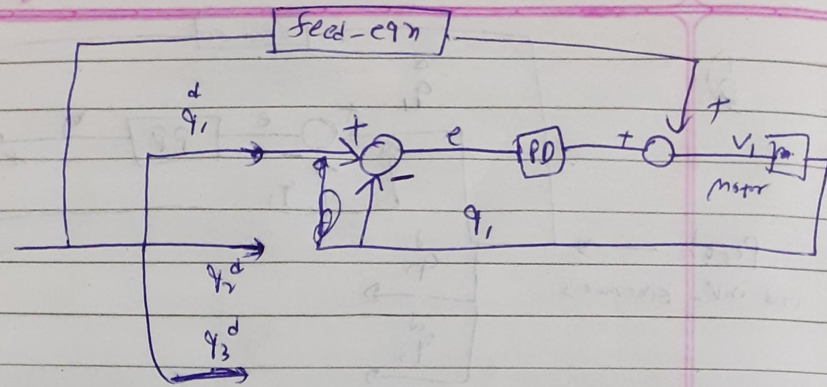
→ we can find Voltage ahead of time by substituting desired joint variables.

so, final eqn with PD controller is,

$$V_1 = feed\_eqn(V_1) + Kp_1(q_1^d - q_1) + Kp_1(\dot{q}_1^d - \dot{q}_1)$$
$$V_2 = feed\_eqn(V_2) + Kp_2(q_2^d - q_2) + Kp_2(\dot{q}_2^d - \dot{q}_2)$$
$$V_3 = feed\_eqn(V_3) + Kp_3(q_3^d - q_3) + Kp_3(\dot{q}_3^d - \dot{q}_3)$$

c)

desired path and inverse kinematics



→ here feed-eqn involves motor dynemics and Robot dynamics. ~~which~~

→ contaller eqn is,

$$V = \left\{\left(\frac{J_m}{r} + D_{ij}\right)\ddot{q} + \left(\frac{B_{eff}}{r}\right)\dot{q} + C(q,\dot{q})\dot{q} + g(q)\right\}\frac{R}{K_m}$$

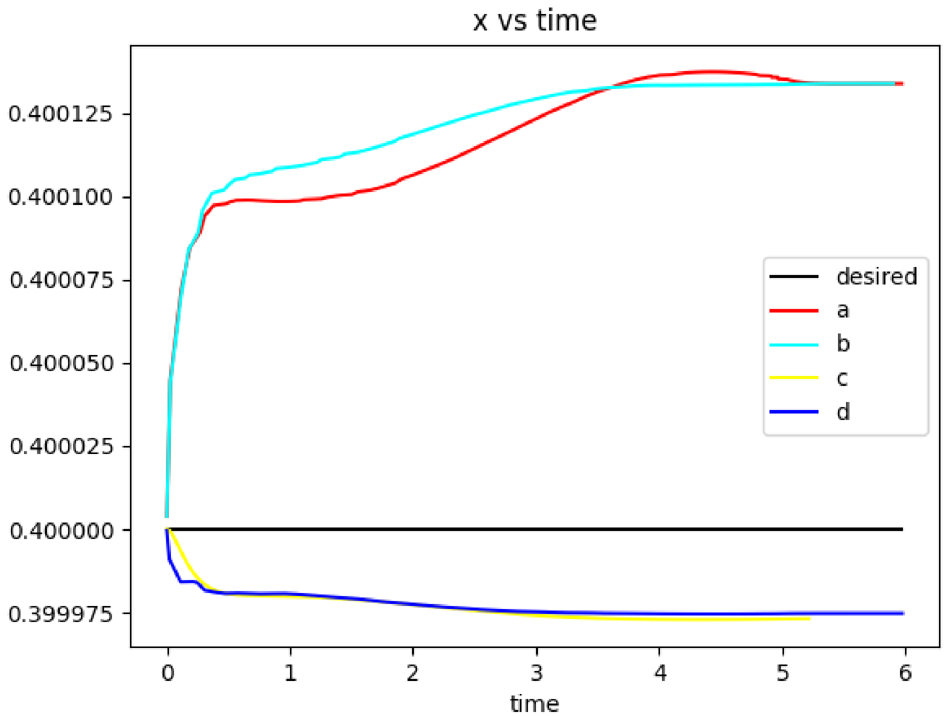
$$+ K_P(q^d - q) + K_P(\dot{q}^d - \dot{q})$$

d) Multivariable contalls involves the use of desired and realtime joint variables togather for voltage V_1,

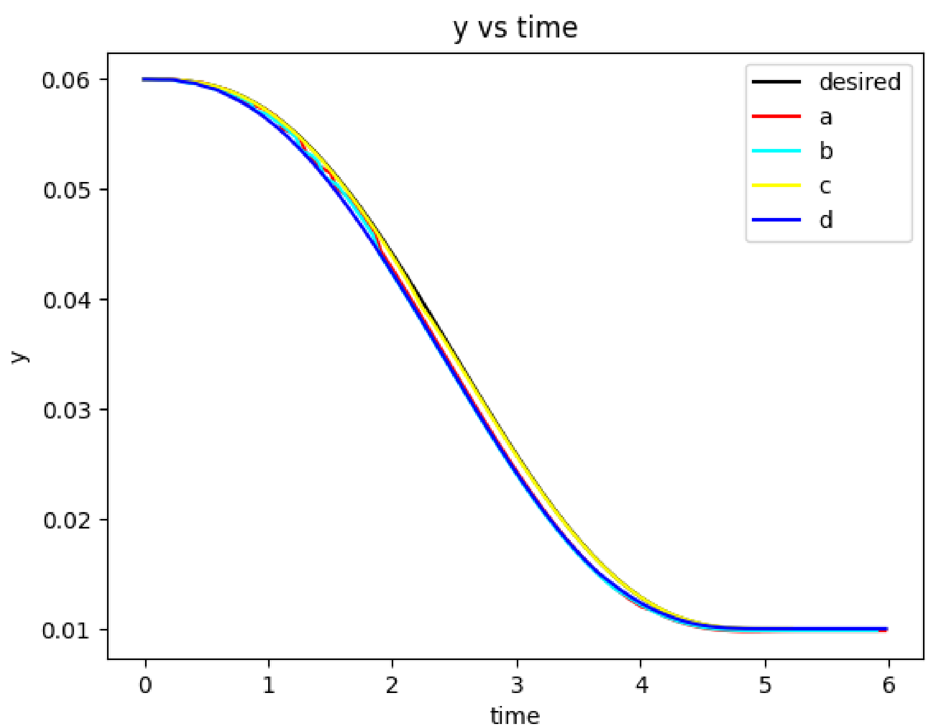
$$\frac{K_1}{r_1 R_1} V_1 = \left(\frac{J_1}{r_1} + D_{ij}\right)\ddot{q}_1 + \frac{B_{eff}}{r_1}\dot{q} + c(q,\dot{q})\dot{q} + g(q)$$

where $\ddot{q}_1 = \ddot{q}_1^d(t) + K_P(q^d - q) + K_P(\dot{q}^d - \dot{q})$

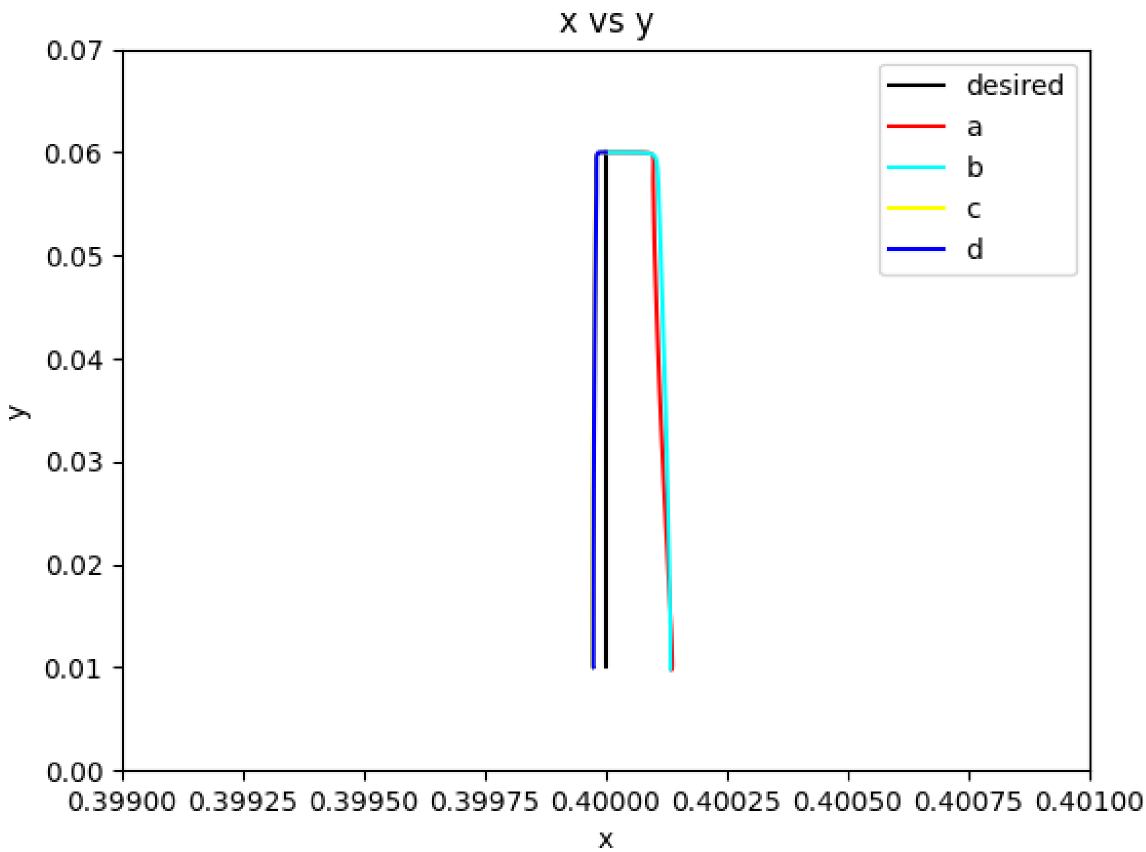
Task 4:



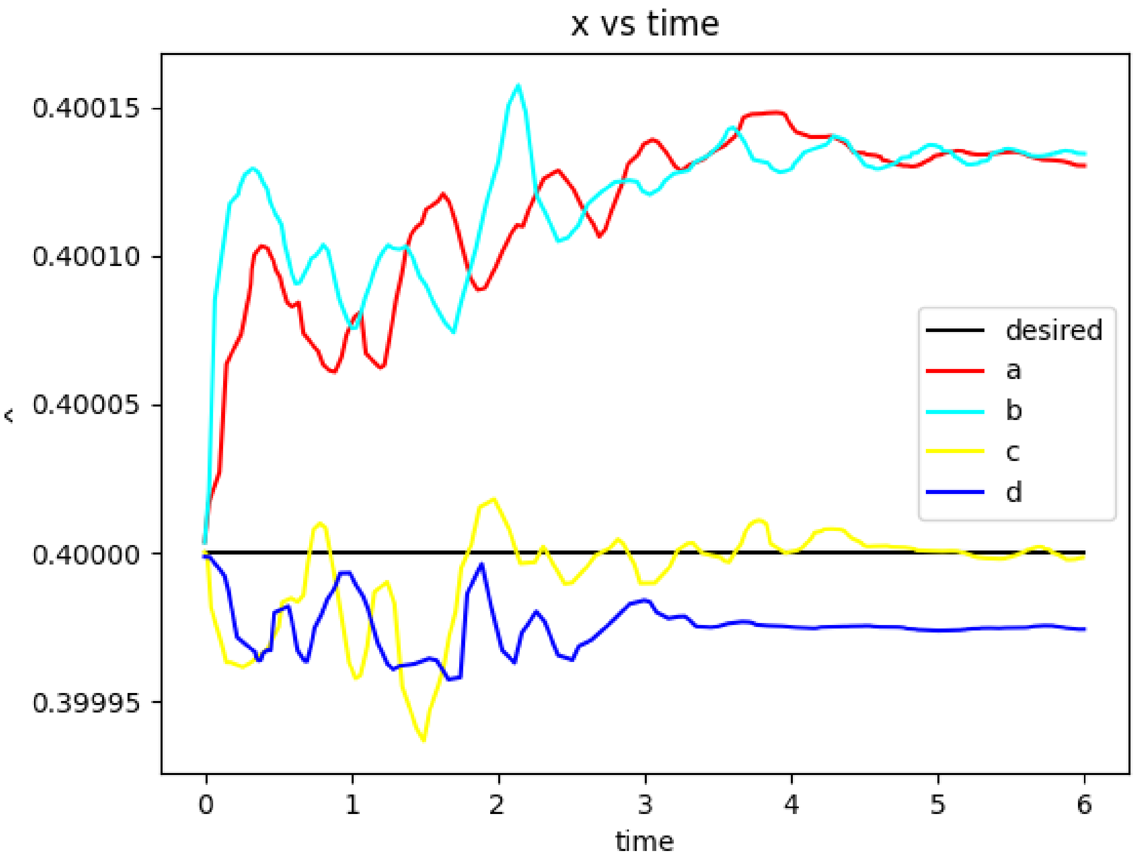
x vs time
0.400125
0.400100
0.400075
0.400050
0.400025
0.400000
0.399975
0
1
2
3
4
5
6
time
desired
a
b
c
d



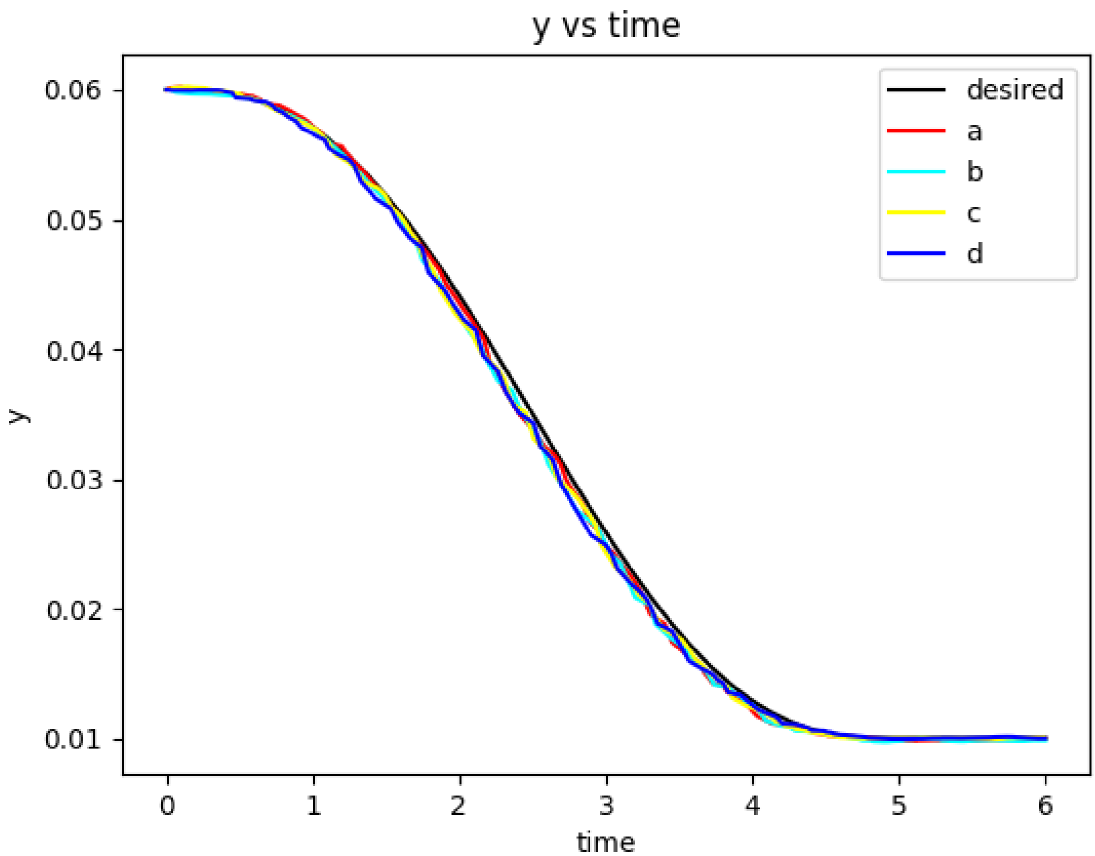
y vs time
0.06
0.05
0.04
0.03
0.02
0.01
y
0
1
2
3
4
5
6
time
desired
a
b
c
d


x vs y
desired
a
b
c
d
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
y
0.39900
0.39925
0.39950
0.39975
0.40000
0.40025
0.40050
0.40075
0.40100
x

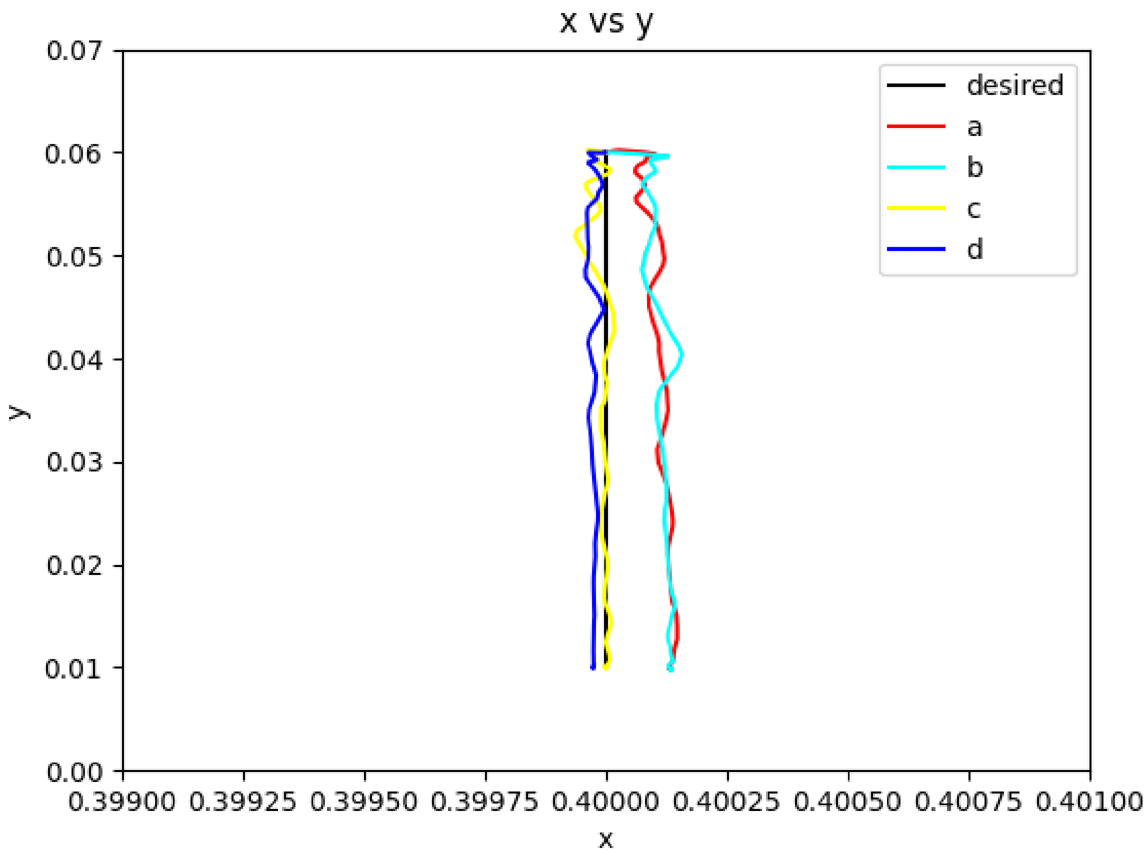
Task 5:



x vs time
0.40015
0.40010
0.40005
0.40000
0.39995
<
0
1
2
3
4
5
6
time
desired
a
b
c
d

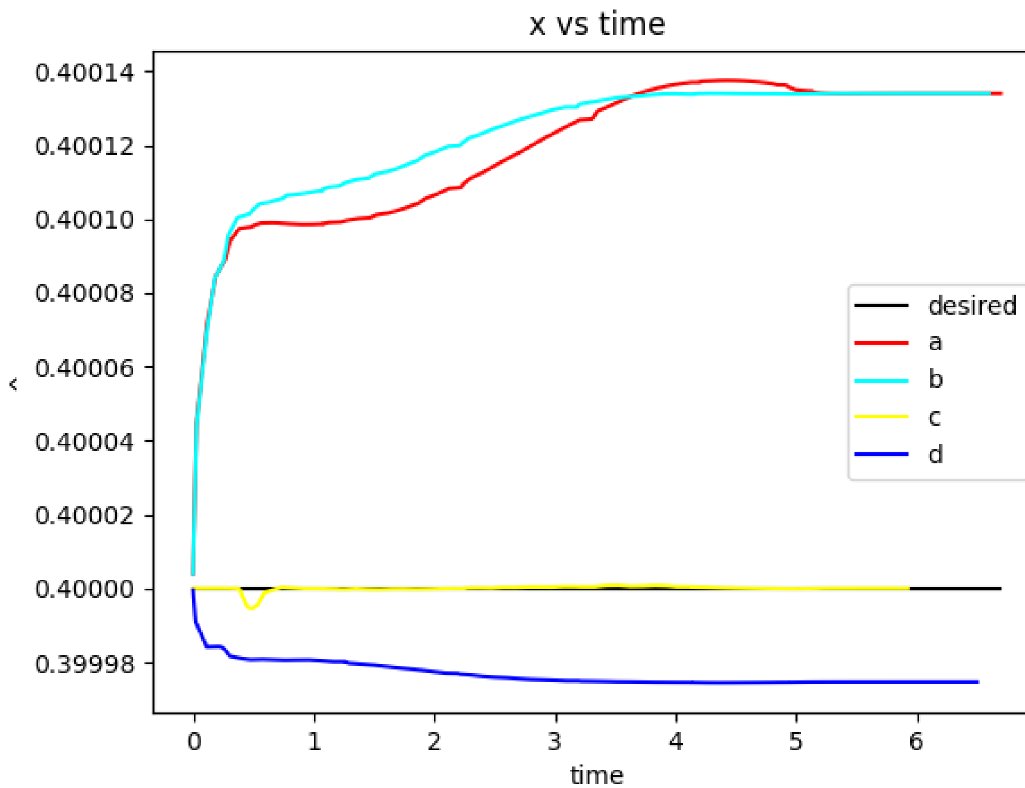


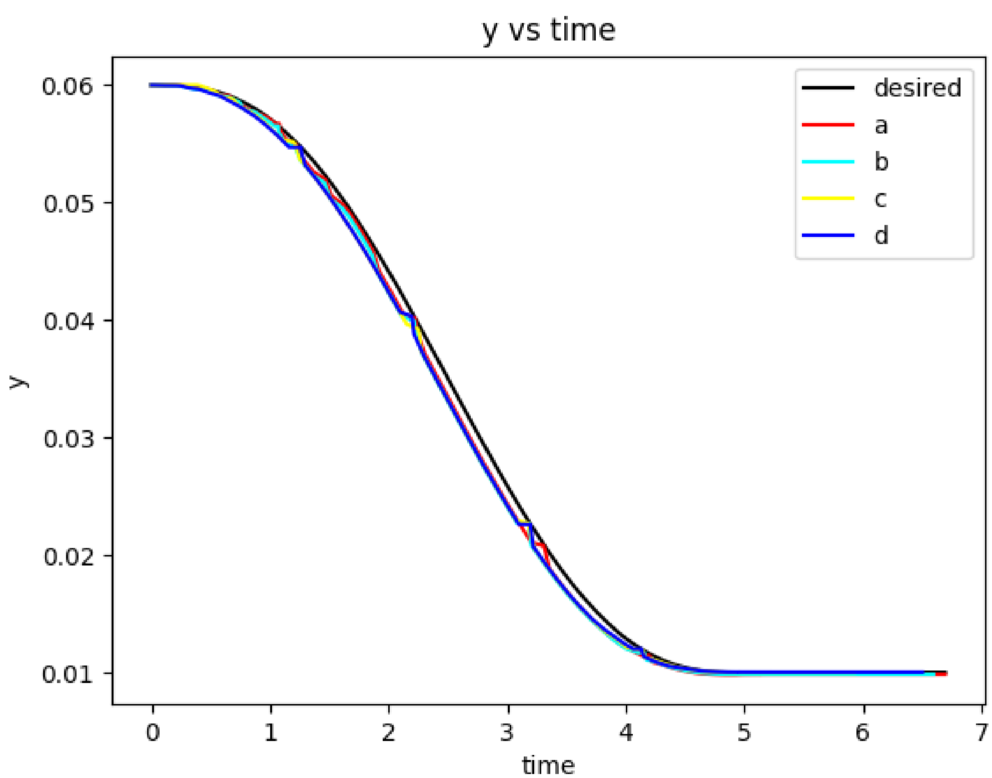
y vs time
0.06
0.05
0.04
0.03
0.02
0.01
y
0
1
2
3
4
5
6
time
desired
a
b
c
d

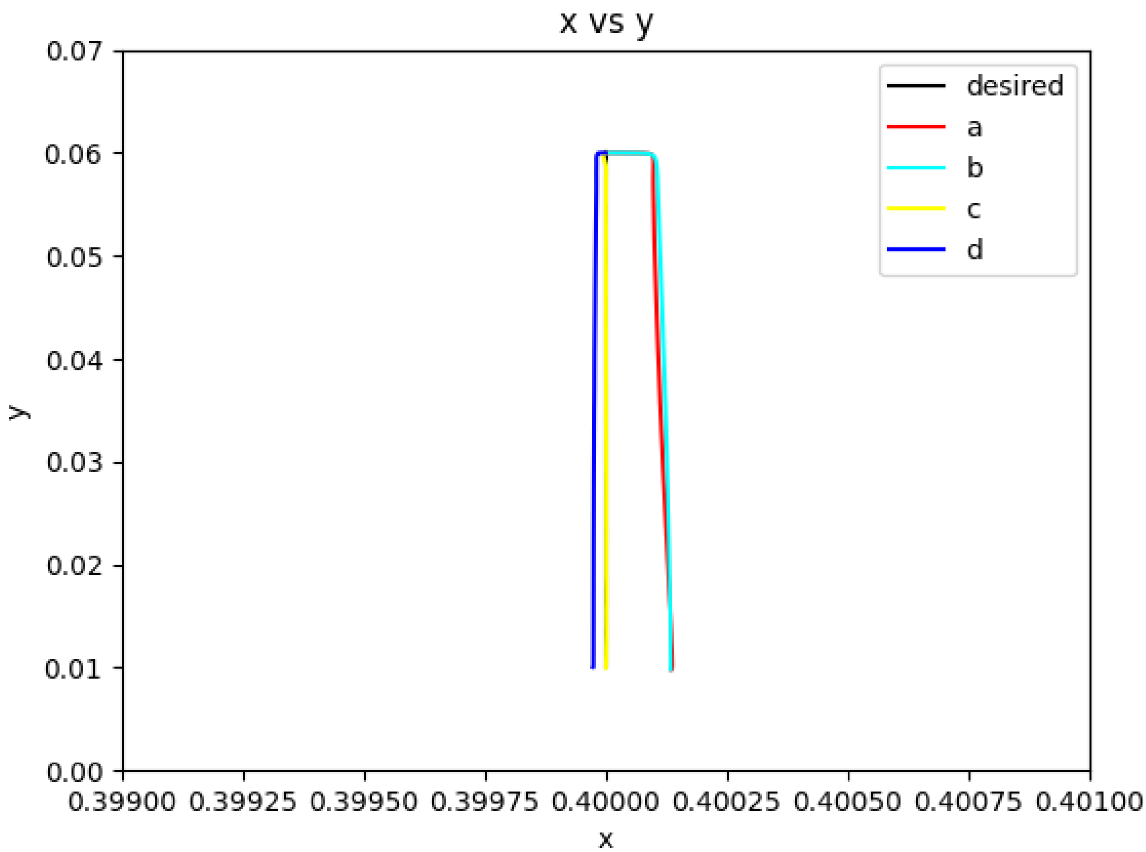


Task 6:

Adding impulse every second for one step in simulaiton time. So, at t=1,2,3,4,5 , I have added an impulse torque of magnitude 1e+10 N-m to all the three joints for 0.1 seconds. It can be treated as close to an impulsive disturbance.


x vs time
0.40014
0.40012
0.40010
0.40008
0.40006
0.40004
0.40002
0.40000
0.39998
x
0
1
2
3
4
5
6
7
time
desired
a
b
c
d


y vs time
0.06
0.05
0.04
0.03
0.02
0.01
y
0
1
2
3
4
5
6
7
time
desired
a
b
c
d



Task 7:

For the error in length, uppon zooming the graph of *y vs time*, I observed that feedforward disturbance cancellation using computed torque (c) is closest to the desired path throughout the trajectory among all the other methods.

From Task-5's feedforward disturbance cancellation using computed torque (c) controller, the steady state error is almost zero because all the dyenmics is being calcelled and PD part is taking disturbance into account. But when the error is introduced in the length, controller c is not able to reduce the steady state error because it is cancelling erroneous dynamics equation.

For the impulse one, I was able to get the impluses for all the controllers at t=2 seconds. Upone zooming at t=2s, all the controllers were working quite well but the PD controller was close to the desired trajectory, but after some time, it was the controller (c) which was close to the desired trajectory.

So, Multivariable controller (d) might be a good controller for a generalised case but if we are certain about the dimensions and properties of the robot, controller (c) would perform better with random disturbances and/or impulses. All the controllers leads to good result in general (atleast for the cases we tried)