RBE502 – Robot Control

Final Exam

Submitted by: Suketu Parekh

a) Generate a cubic polynomial trajectory for the first and second joint of the robot. The time span and the desired initial and final configuration and velocity of the robot as well are given by:

$$t0 = 0, tf = 10 \text{ sec}$$

$$\theta1(t0) = 180\circ, \theta1(tf) = 0, \theta2(t0) = 90\circ, \theta2(tf) = 0$$

$$\dot{\theta}1(t0) = \dot{\theta}1(tf) = \dot{\theta}2(t0) = \dot{\theta}2(tf) = 0$$

Ans.) Here, t1_des and t2_des stand for desired trajectories of theta1 and theta2 respectively. The following are the required equations:

```
t1_des = a1*t.^3 + b1*t.^2 + c1.*t + d1;
t2_des = a2*t.^3 + b2*t.^2 + c2.*t + d2;


[a1,b1,c1,d1] = [pi/500, -(3*pi)/100, 0, pi]
[a2,b2,c2,d2] = [pi/1000, -(3*pi)/200, 0, pi/2]
```

b) Consider the equations of motion derived for the robot in Programming Assignment 1. Transform the equations of motion (dynamics) of the robot to the standard Manipulator form:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau$$

Ans.) Equations of motion are as follows:

```
t2_ddot*(I2 + (m2*(2*r2^2 + 2*l1*cos(t2)*r2))/2) -
g*(m2*(r2*sin(t1 + t2) + l1*sin(t1)) + m1*r1*sin(t1))
+ t1_ddot*(m1*r1^2 + I1 + I2 + (m2*(2*l1^2 + 4*cos(t2)*l1*r2 +
2*r2^2))/2) - (m2*t2_dot*(2*l1*r2*sin(t2)*(t1_dot + t2_dot) +
2*l1*r2*t1_dot*sin(t2)))/2 = u1
```

-------Eq'n1

```
t2_ddot*(m2*r2^2 + I2) + t1_ddot*(I2 + (m2*(2*r2^2 +
2*l1*cos(t2)*r2))/2) - g*m2*r2*sin(t1 + t2) +
l1*m2*r2*t1_dot*sin(t2)*(t1_dot + t2_dot) -
l1*m2*r2*t1_dot*t2_dot*sin(t2) = u2
```

-------Eq'n2

Also, we know that:

```
[M1 M2; M3 M4]*[t1_ddot;t2_ddot] + [C1 C2; C3
C4]*[t1_dot;t2_dot] + [g1;g2] == [u(1);u(2)]
```

-------Eq'n3

From equation 1 and 3, we have:

```
M1 = (m1*r1^2 + I1 + I2 + (m2*(2*l1^2 + 4*cos(t2)*l1*r2 +
2*r2^2))/2)
M2 = (I2 + (m2*(2*r2^2 + 2*l1*cos(t2)*r2))/2)
C1 = - (m2*t2_dot*(2*l1*r2*sin(t2)+2*l1*r2*sin(t2)))/2
C2 = - (m2*t2_dot*(2*l1*r2*sin(t2)))
g1 = - g*(m2*(r2*sin(t1 + t2) + l1*sin(t1)) + m1*r1*sin(t1))
```

From equation 2 and 3, we have:

```
M3 = (I2 + (m2*(2*r2^2 + 2*l1*cos(t2)*r2))/2)
M4 = (m2*r2^2 + I2)
C3 = l1*m2*r2*t1_dot*sin(t2) - l1*m2*r2*t2_dot*sin(t2)
C4 = l1*m2*r2*t1_dot*sin(t2)
g2 = - g*m2*r2*sin(t1 + t2)
```

Replacing the above obtained expressions for M, C and g into equation 3, the required equation is obtained in Manipulator form.

- **Robust Controller:**

c) (5 points) Design a Robust Inverse Dynamics control law for trajectory tracking by the robot using the method described in Lecture 19. The control gains to be designed are Kp ∈R2×2 and Kd ∈ R2×2 (for the virtual control input v), and the Lyapunov matrix P = PT > 0 ∈ R4×4 for the robust control term vr.

– Use state-feedback control design to determine the control gains Kp and Kd to place the eigenvalues at {−3, −3, −4, −4}. You can use the place function in MATLAB to design the control gain matrix K ∈R2×4, where K = [Kp Kd].

– The matrix P is the solution to the Lyapunov equation A'P + PA = −Q. You can use the lyap function in MATLAB to solve for P.

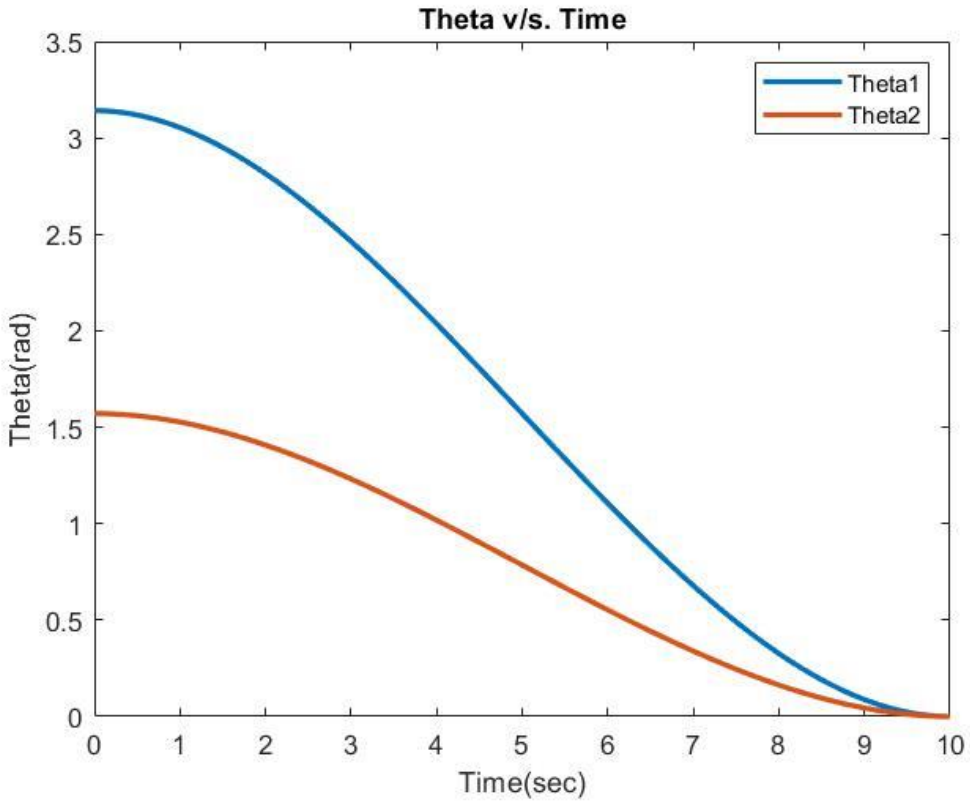
Moreover, initialize a constant value for the uncertainty upper bound ρ, which is used to compute the robust control term vr. This bound can be later tuned in simulation by trial and error. For the purpose of this assignment, you can use a constant ρ value.

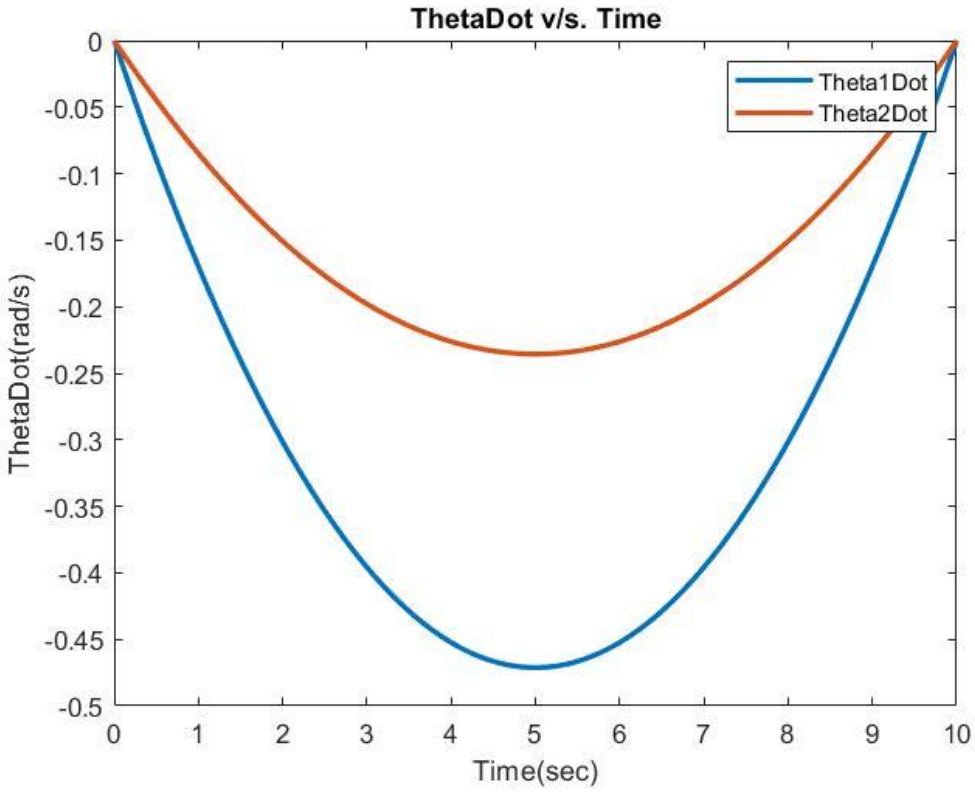
Ans.)

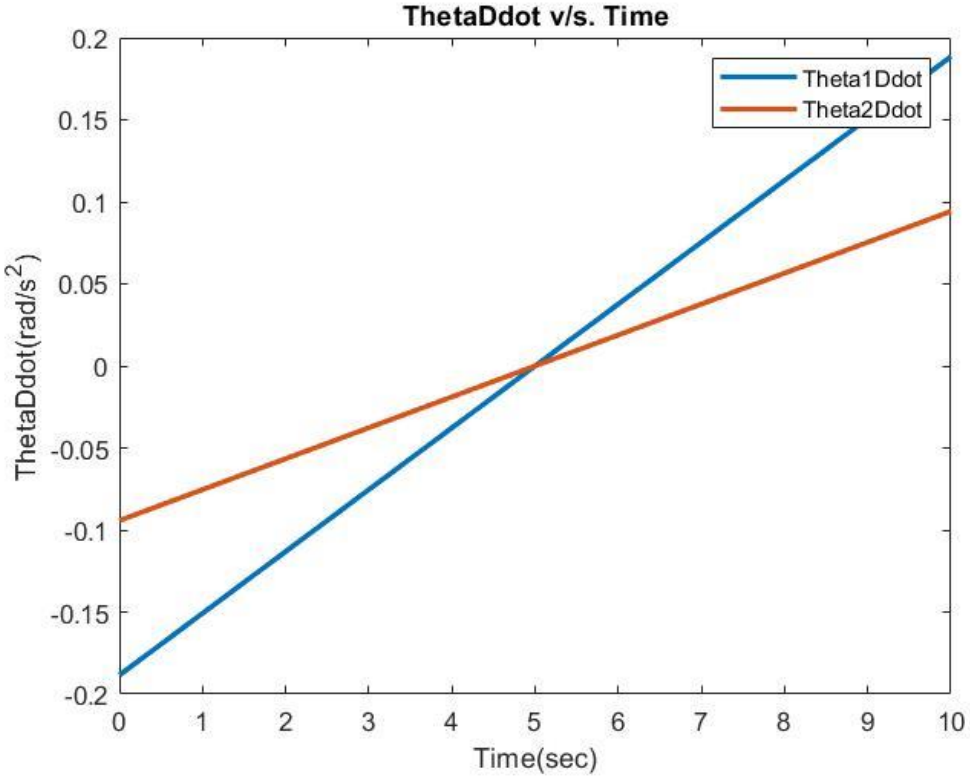
```
Kn =

    12     0     7     0

     0    12     0     7

P =

   13.4226         0    0.4583         0

         0   13.4226         0    0.4583

    0.4583         0    0.8512         0

         0    0.4583         0    0.8512


rho = eye(2).*11
```

d) (5 points) Update the ode function developed in Programming Assignment 3 to implement the robust inverse dynamics control law designed in Step (c). Note that you will need to evaluate the cubic polynomial trajectories inside the ode function to obtain the desired states at each point in time.

Ans.)




ThetaDot v/s. Time
Theta1Dot
Theta2Dot
ThetaDot(rad/s)
Time(sec)
0
-0.05
-0.1
-0.15
-0.2
-0.25
-0.3
-0.35
-0.4
-0.45
-0.5
0
1
2
3
4
5
6
7
8
9
10

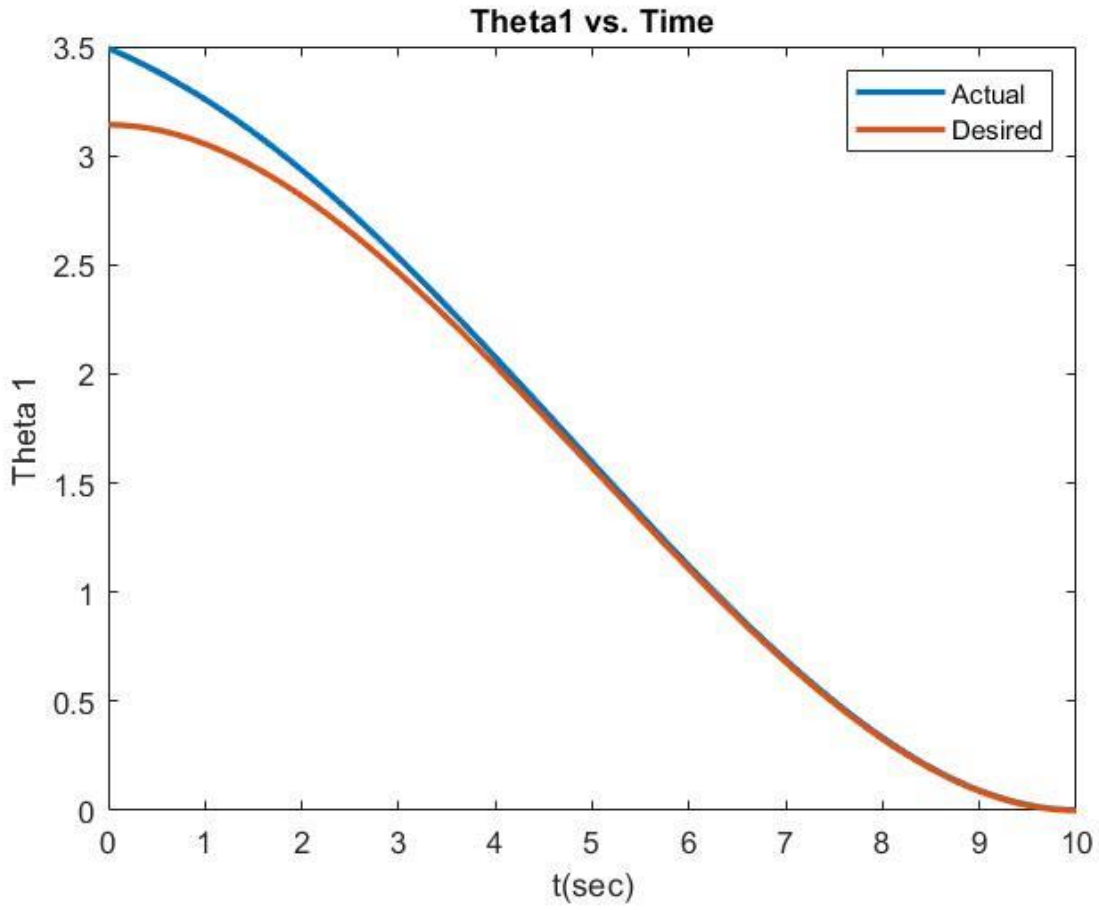

ThetaDdot v/s. Time
Theta1Ddot
Theta2Ddot
ThetaDdot(rad/s^2)
Time(sec)
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
0
1
2
3
4
5
6
7
8
9
10

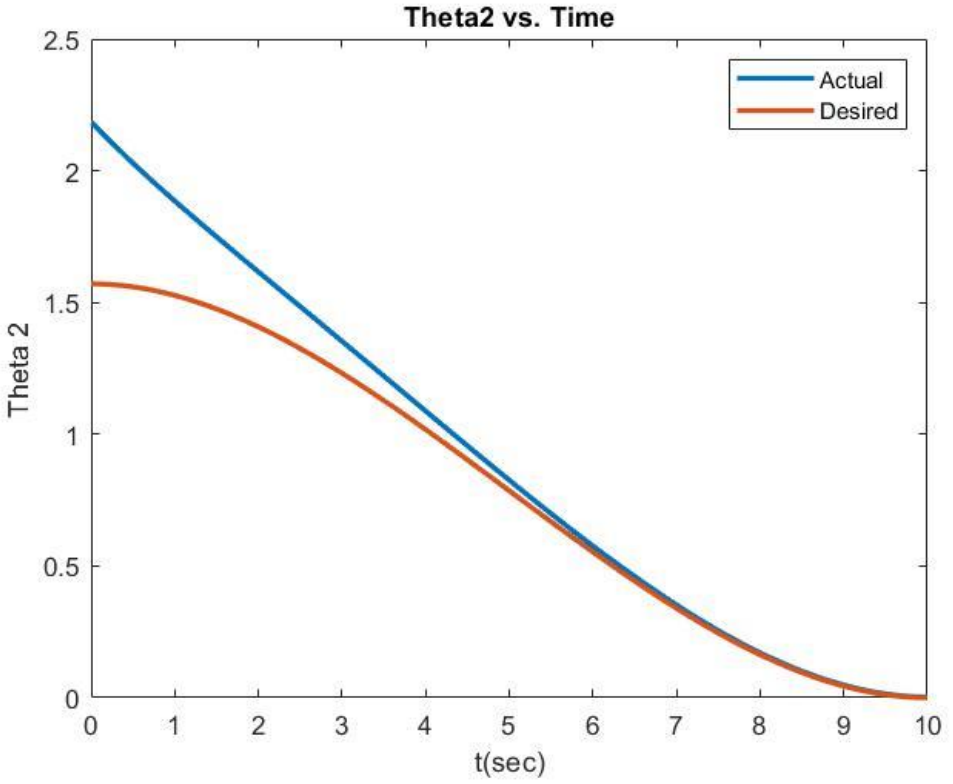
e) (5 points) Use ode45 and the ode function developed in Step (d) to construct a simulation of the system in MATLAB with the time span of [0,10] sec and initial conditions of:

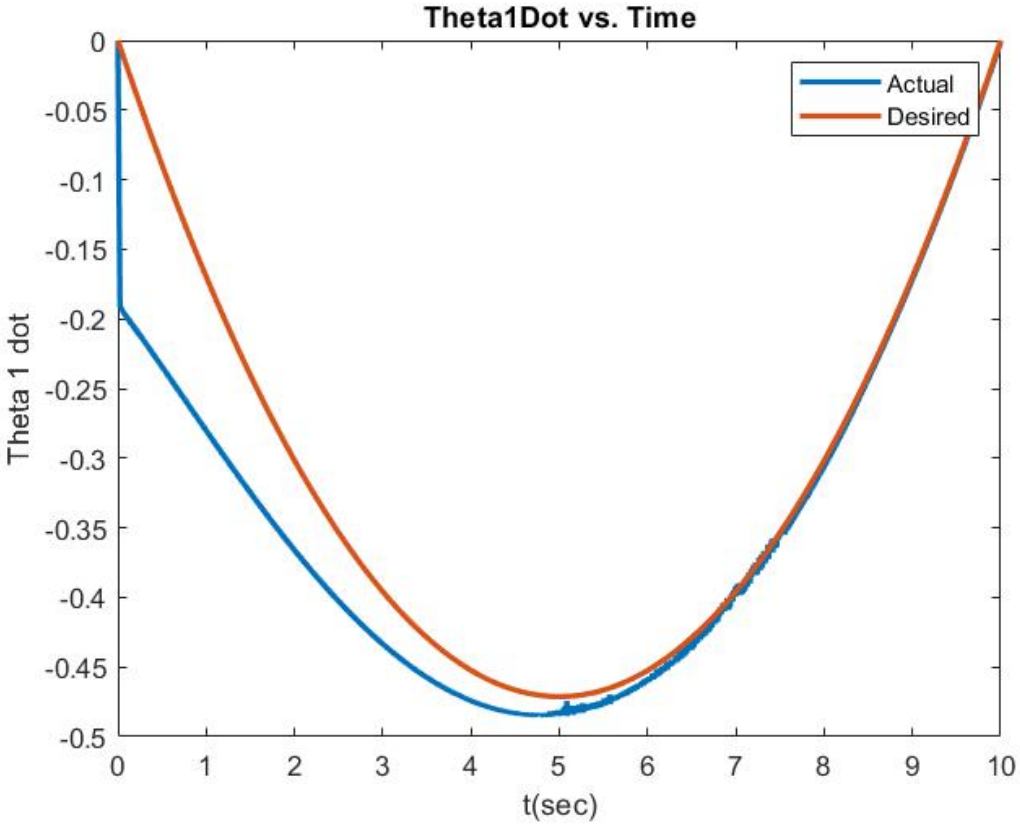
$$\theta 1(0) = 200^{\circ},\ \theta 2(0) = 125^{\circ},\ \dot{\theta}1 = 0,\ \dot{\theta}2 = 0$$

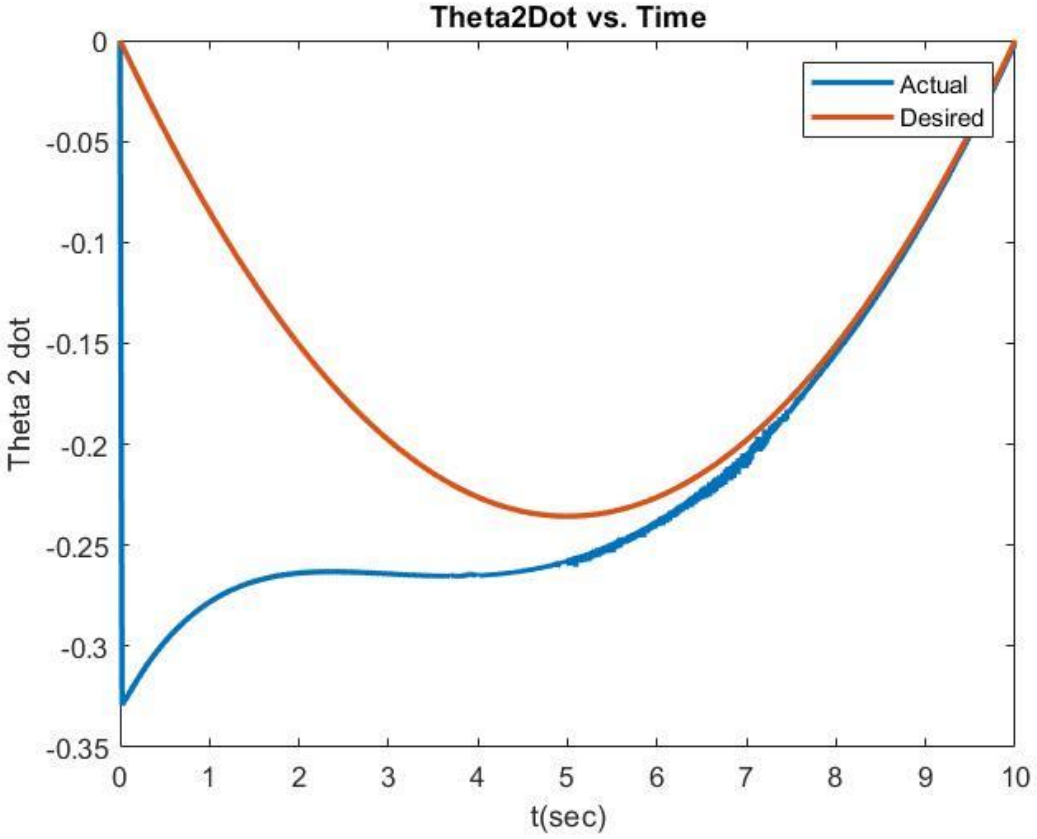
Plot the state trajectories, the control inputs trajectories, and the associated desired trajectories to evaluate the performance. If the performance is not satisfactory (i.e. the system does not track and converge to the desired trajectory), go back to Step (c) and tune the design parameters, including the uncertainty upper bound ρ, the state-feedback control gains, or the Lyapunov matrix P.

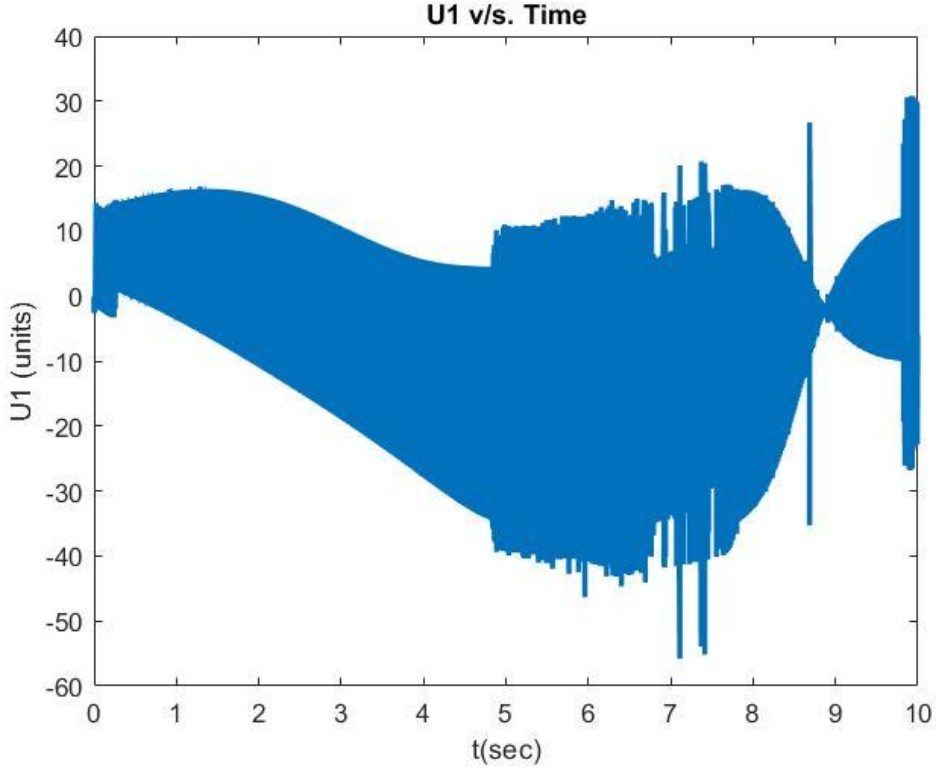
Ans.)

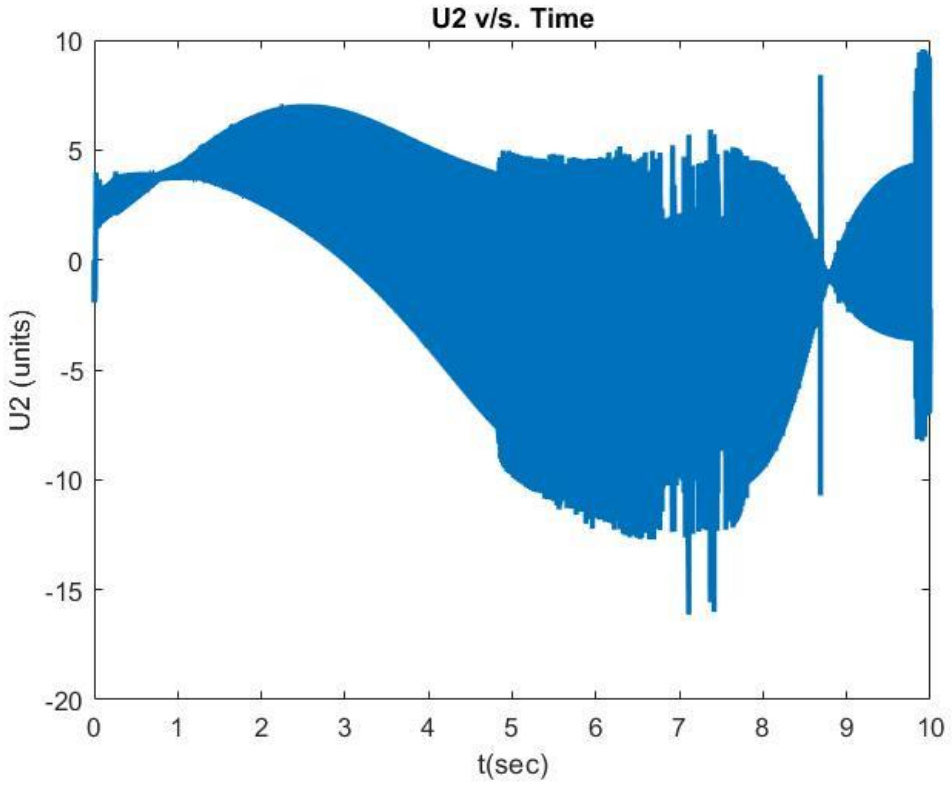



Theta2 vs. Time
Actual
Desired
Theta 2
t(sec)
2.5
2
1.5
1
0.5
0
0
1
2
3
4
5
6
7
8
9
10


Theta1Dot vs. Time
Actual
Desired
Theta 1 dot
t(sec)
0
-0.05
-0.1
-0.15
-0.2
-0.25
-0.3
-0.35
-0.4
-0.45
-0.5
0
1
2
3
4
5
6
7
8
9
10


Theta2Dot vs. Time
Actual
Desired
Theta 2 dot
t(sec)
0
-0.05
-0.1
-0.15
-0.2
-0.25
-0.3
-0.35
0
1
2
3
4
5
6
7
8
9
10

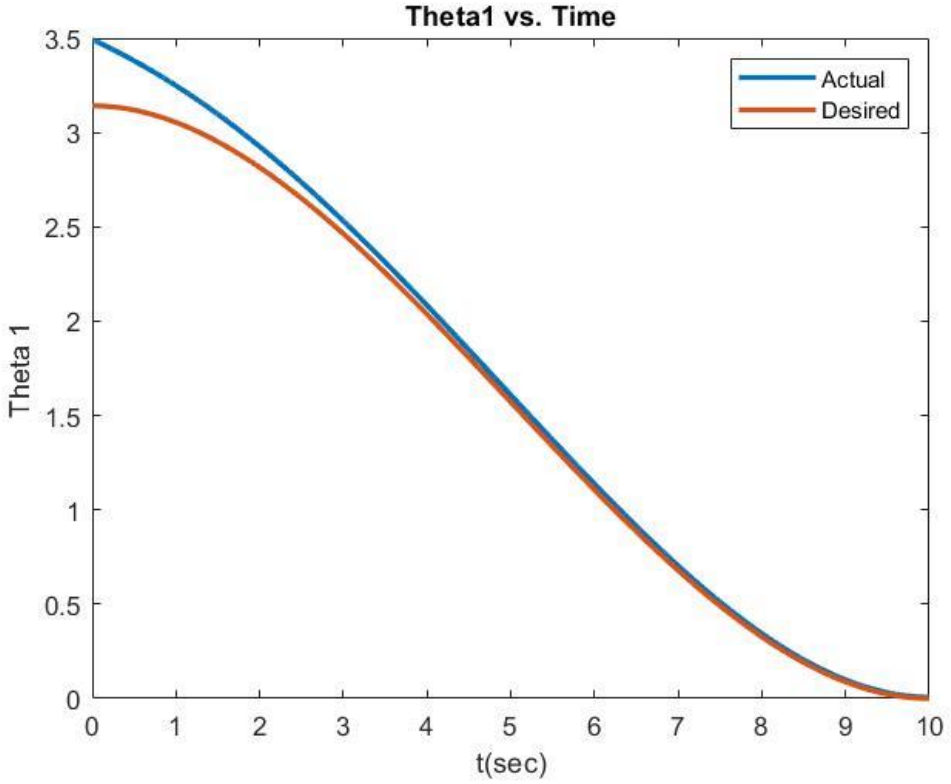

U1 v/s. Time
U1 (units)
t(sec)
40
30
20
10
0
-10
-20
-30
-40
-50
-60
0
1
2
3
4
5
6
7
8
9
10

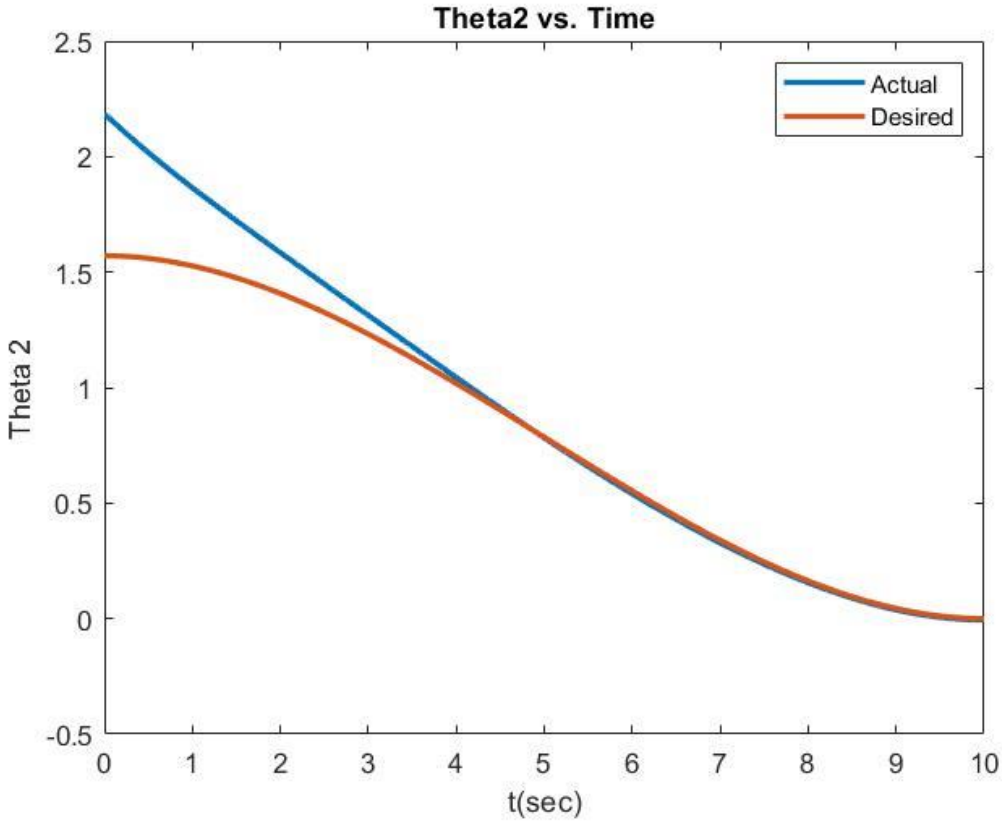


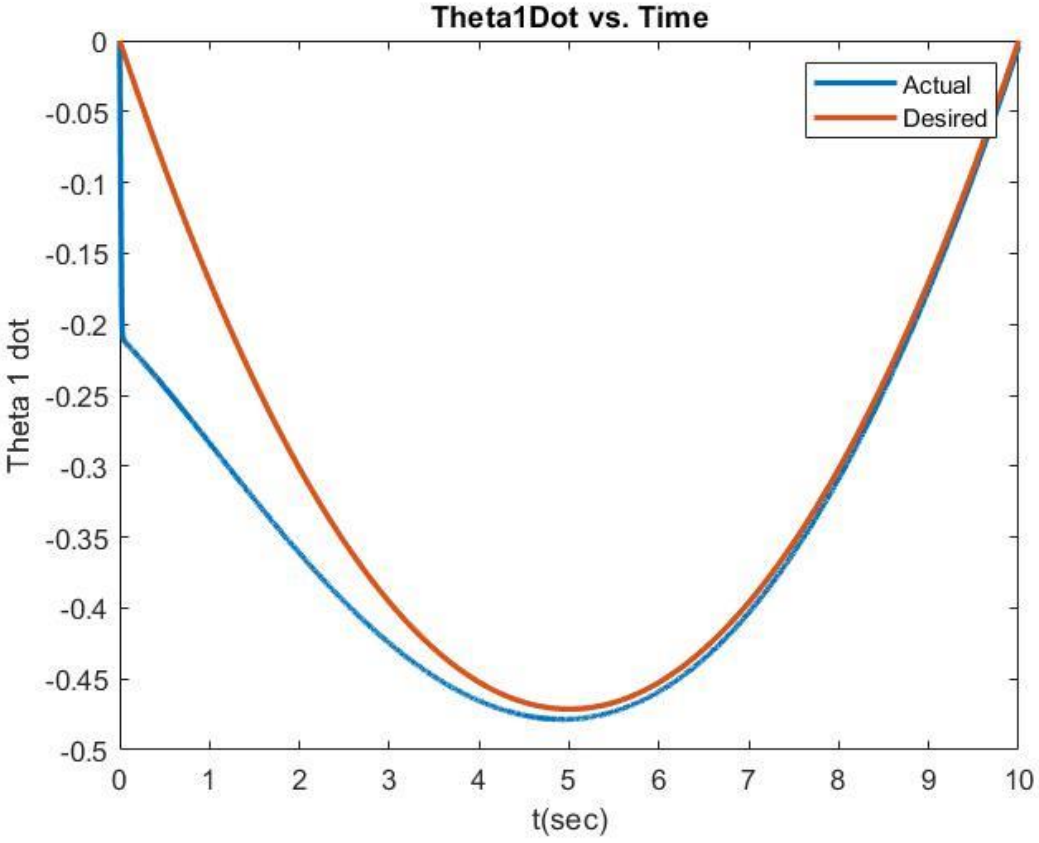
f) (5 points) Include a boundary layer in the robust control term vr to reduce the chattering effect observed in the control input in Step (e). Perform the same simulation as Step (e), and plot the state trajectories, the control inputs trajectories, and the associated desired trajectories to evaluate the performance. Discuss your findings in your final report.

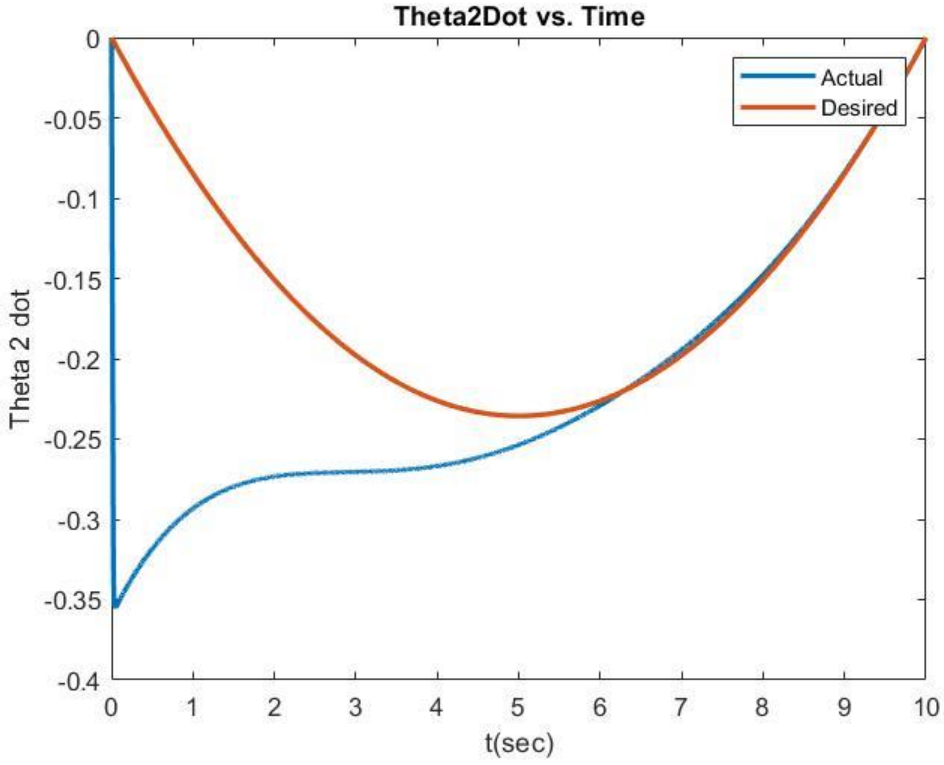
Ans.) `phi = 0.035;`

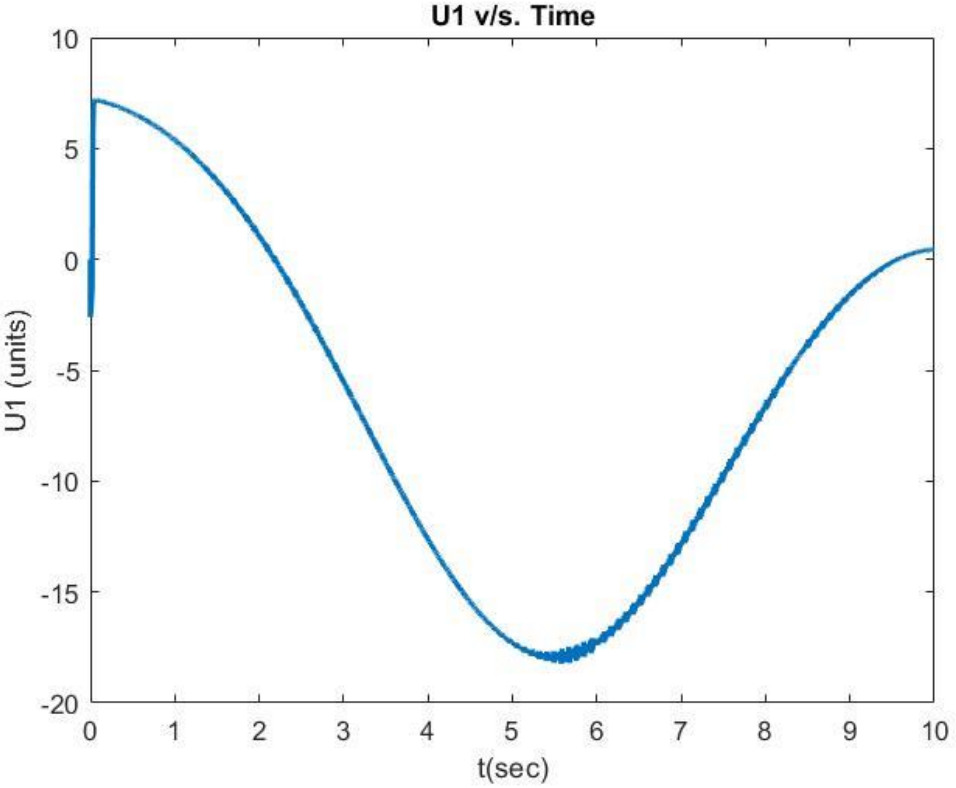
As can be seen from the following images, the inclusion of a boundary layer significantly reduces the number of zero-crossings to only 2 for both U1 and U2. Another observation was that the number of iterations also gets reduced significantly.

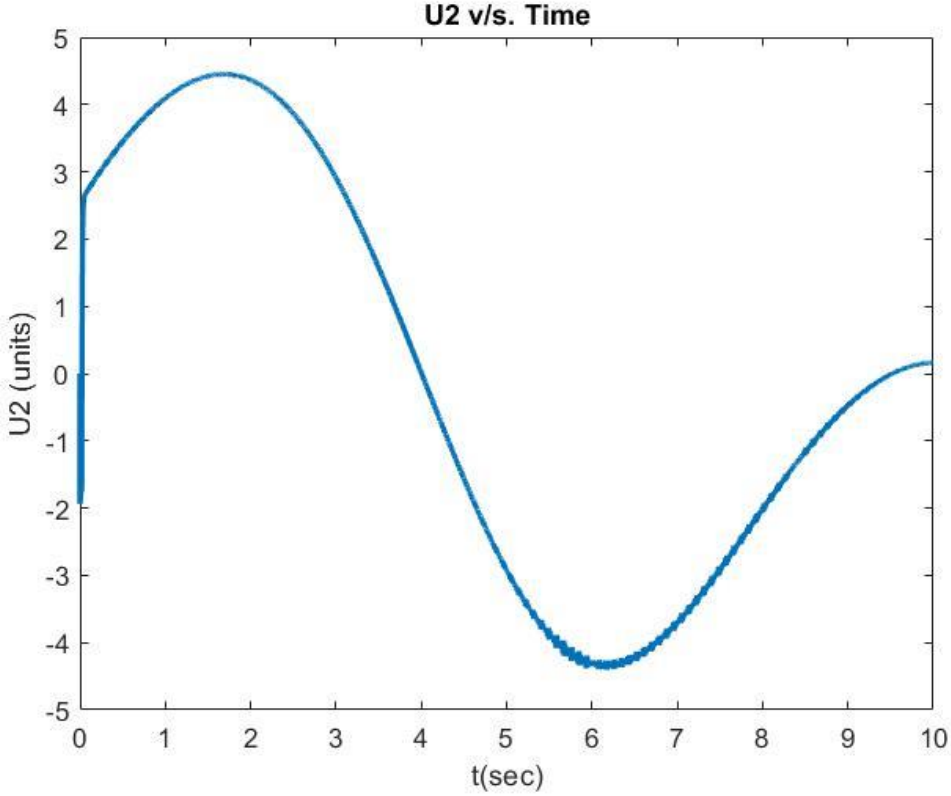

Theta1 vs. Time
Actual
Desired
Theta 1
t(sec)
0
0.5
1
1.5
2
2.5
3
3.5
0
1
2
3
4
5
6
7
8
9
10


Theta2 vs. Time
Actual
Desired
Theta 2
t(sec)
-0.5
0
0.5
1
1.5
2
2.5
0
1
2
3
4
5
6
7
8
9
10


Theta1Dot vs. Time
Actual
Desired
Theta 1 dot
t(sec)
0
-0.05
-0.1
-0.15
-0.2
-0.25
-0.3
-0.35
-0.4
-0.45
-0.5
0
1
2
3
4
5
6
7
8
9
10

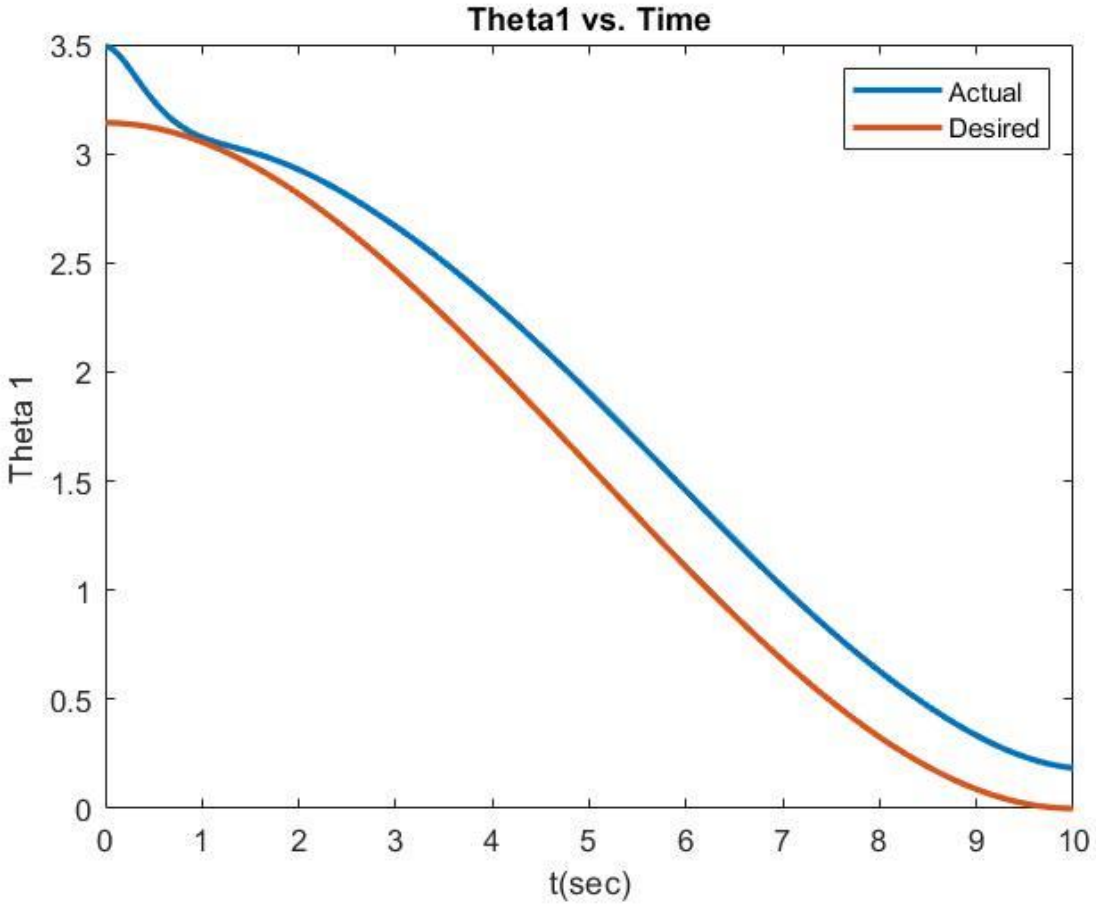

Theta2Dot vs. Time
Actual
Desired
Theta 2 dot
t(sec)
0
-0.05
-0.1
-0.15
-0.2
-0.25
-0.3
-0.35
-0.4
0
1
2
3
4
5
6
7
8
9
10

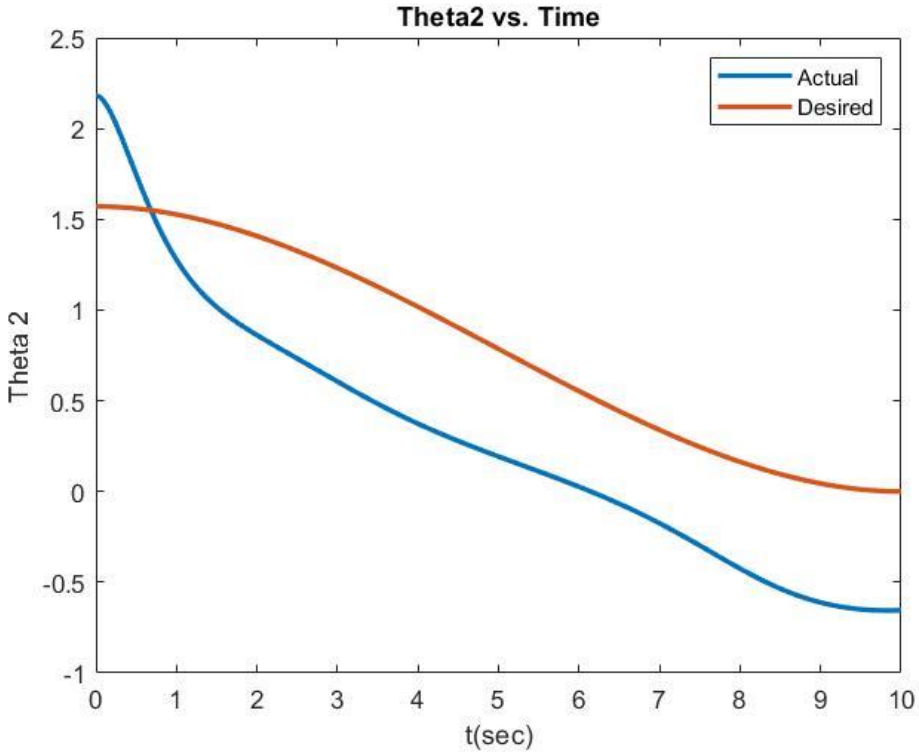

U1 v/s. Time
U1 (units)
t(sec)
10
5
0
-5
-10
-15
-20
0
1
2
3
4
5
6
7
8
9
10

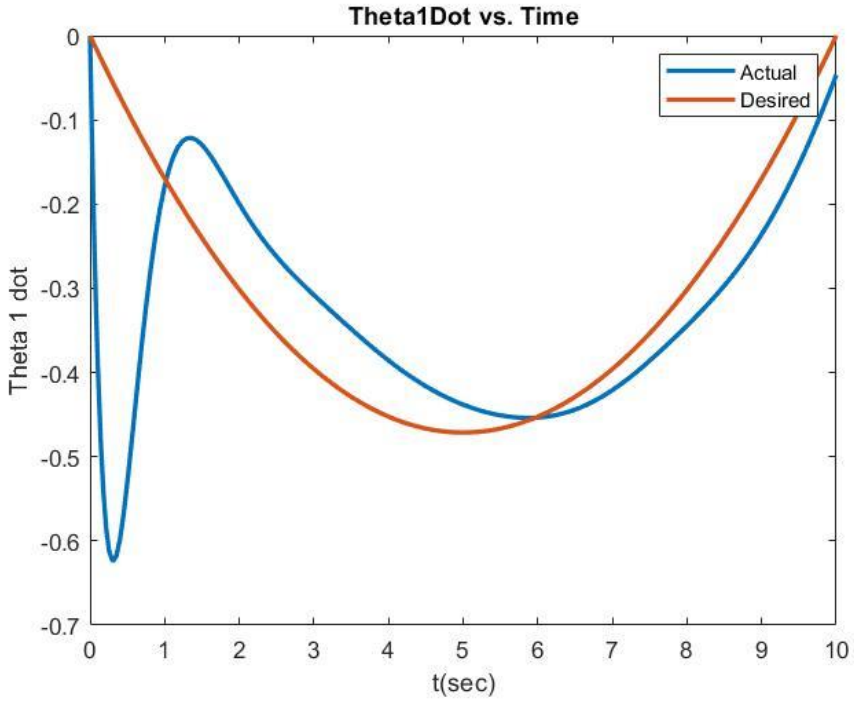

U2 v/s. Time
U2 (units)
t(sec)
5
4
3
2
1
0
-1
-2
-3
-4
-5
0
1
2
3
4
5
6
7
8
9
10

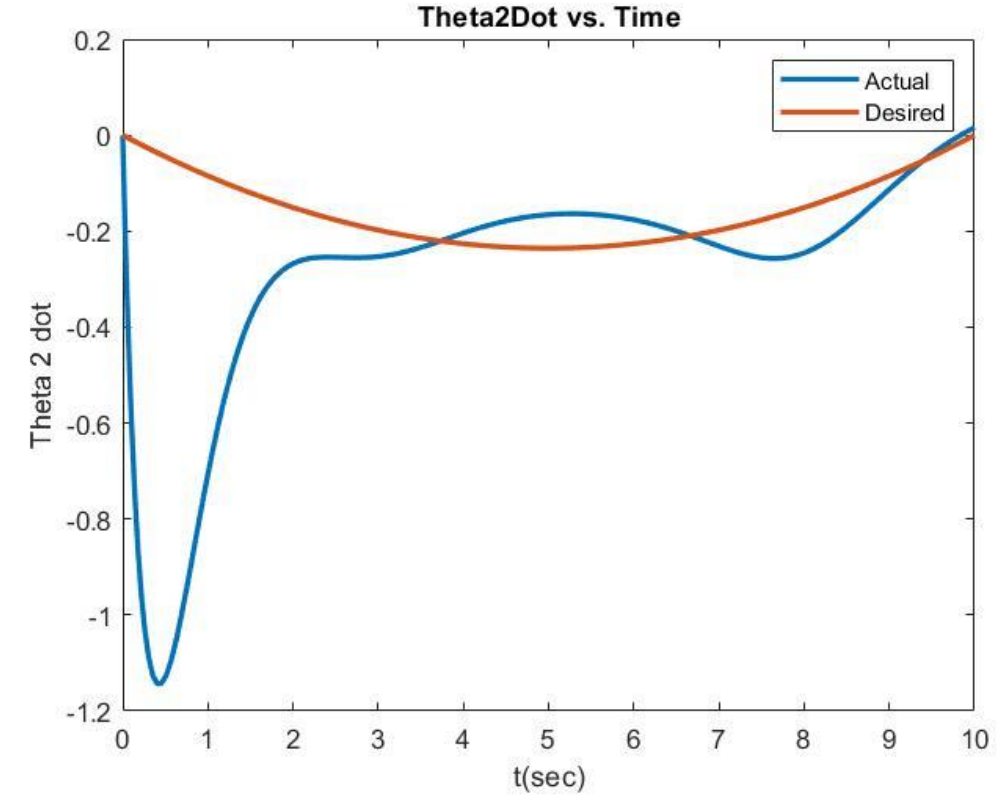
g) (2.5 points) To evaluate the performance of the robot without the robust inverse dynamics control, construct a simulation of the system in MATLAB with the same control law but with the vr term set to zero (do not change other design parameters such as Kp and Kd). Again, plot the state trajectories, the control inputs trajectories and the associated desired trajectories to compare the resulting performance with the performance obtained in Step (e). Discuss the results in your final report.

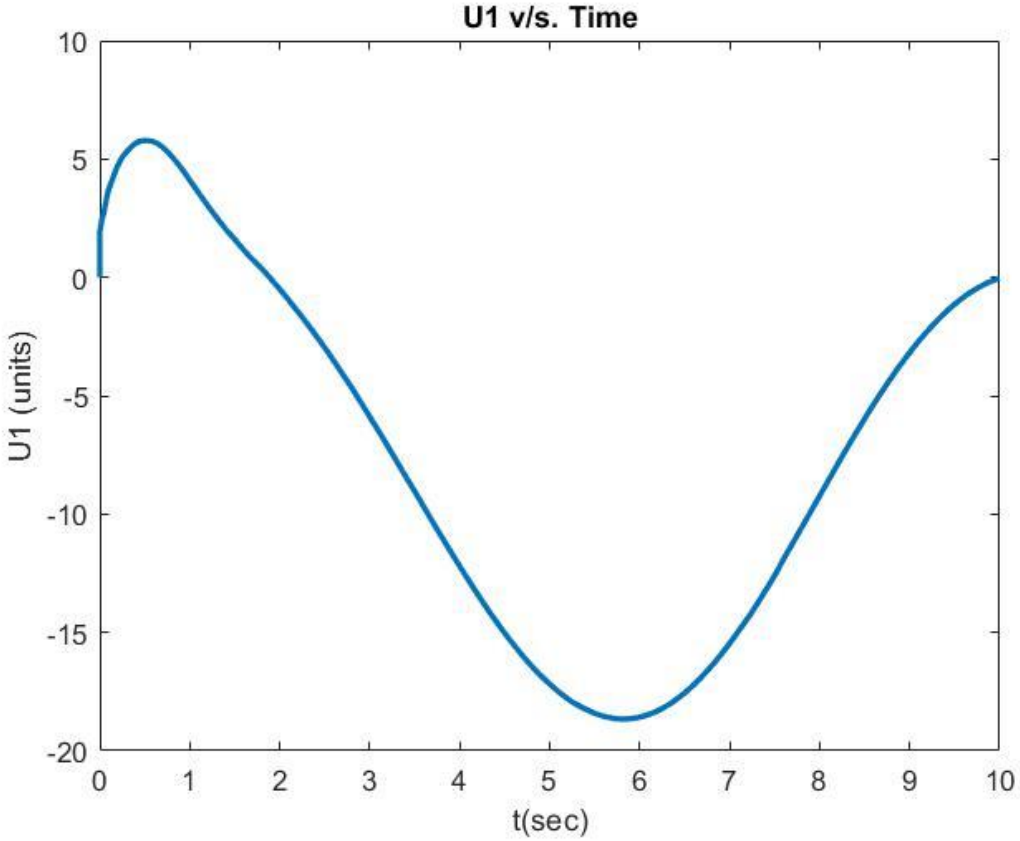
Ans.) When the value of vr is taken as zero, the states and trajectories never actually converge to zero and there appears to be a constant steady-state error in the trajectory. This may be due to the fact that the vr term holds the error term as well and the overall function works towards reducing the error value to zero.

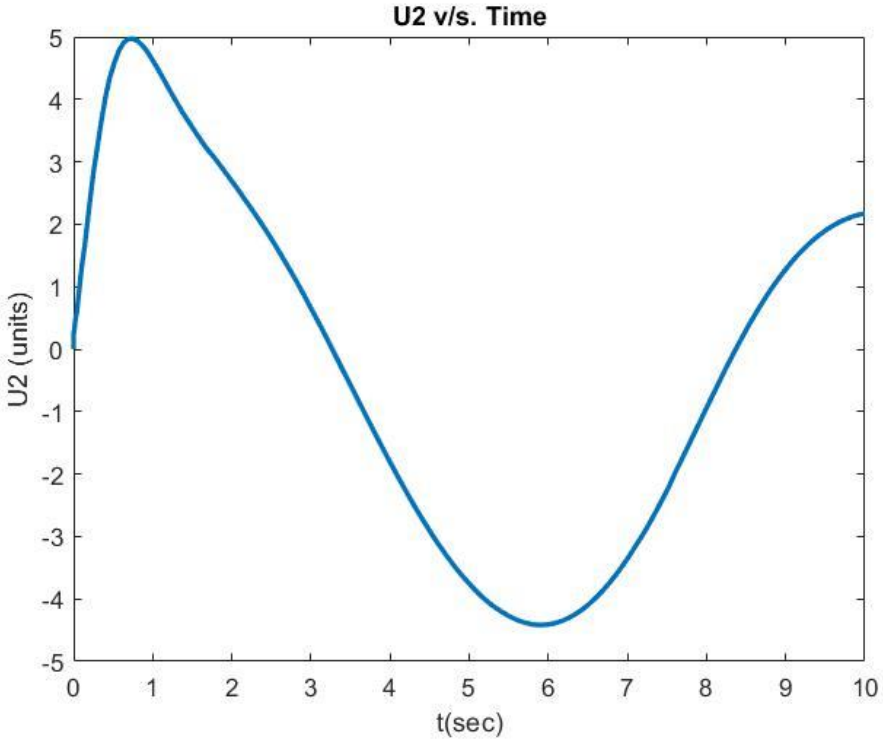



Theta2 vs. Time
Actual
Desired
Theta 2
t(sec)
2.5
2
1.5
1
0.5
0
-0.5
-1
0
1
2
3
4
5
6
7
8
9
10


Theta1Dot vs. Time
Actual
Desired
Theta 1 dot
t(sec)
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.6
-0.7
0
1
2
3
4
5
6
7
8
9
10

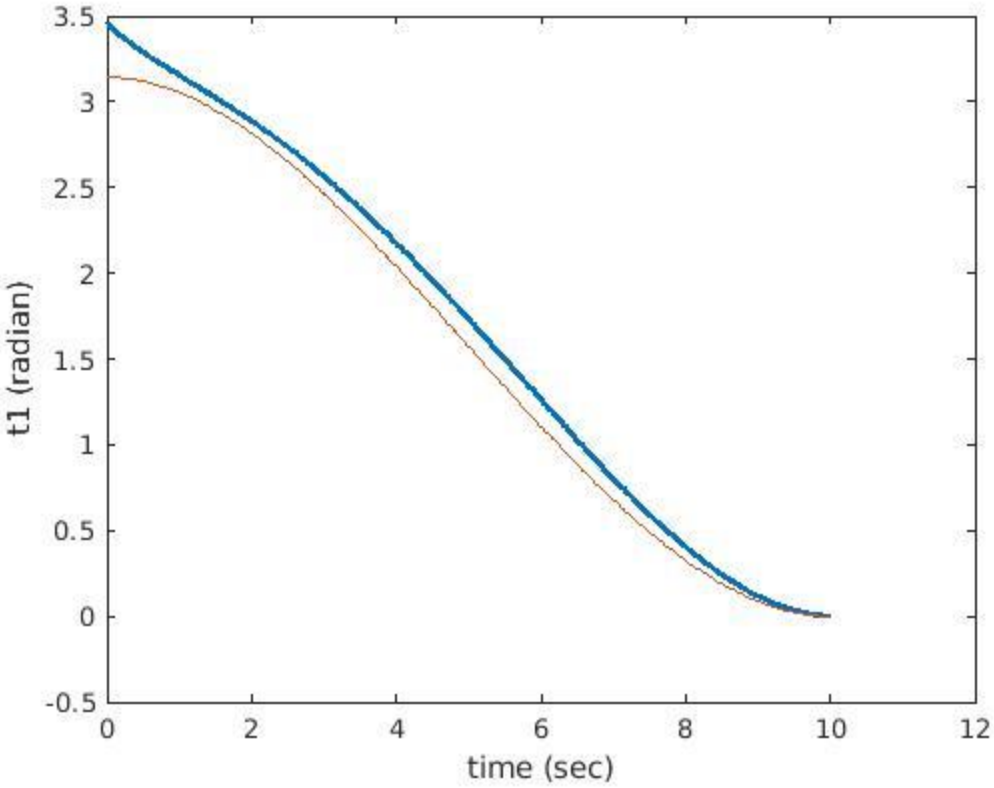

Theta2Dot vs. Time
Actual
Desired
Theta 2 dot
t(sec)
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
0
1
2
3
4
5
6
7
8
9
10

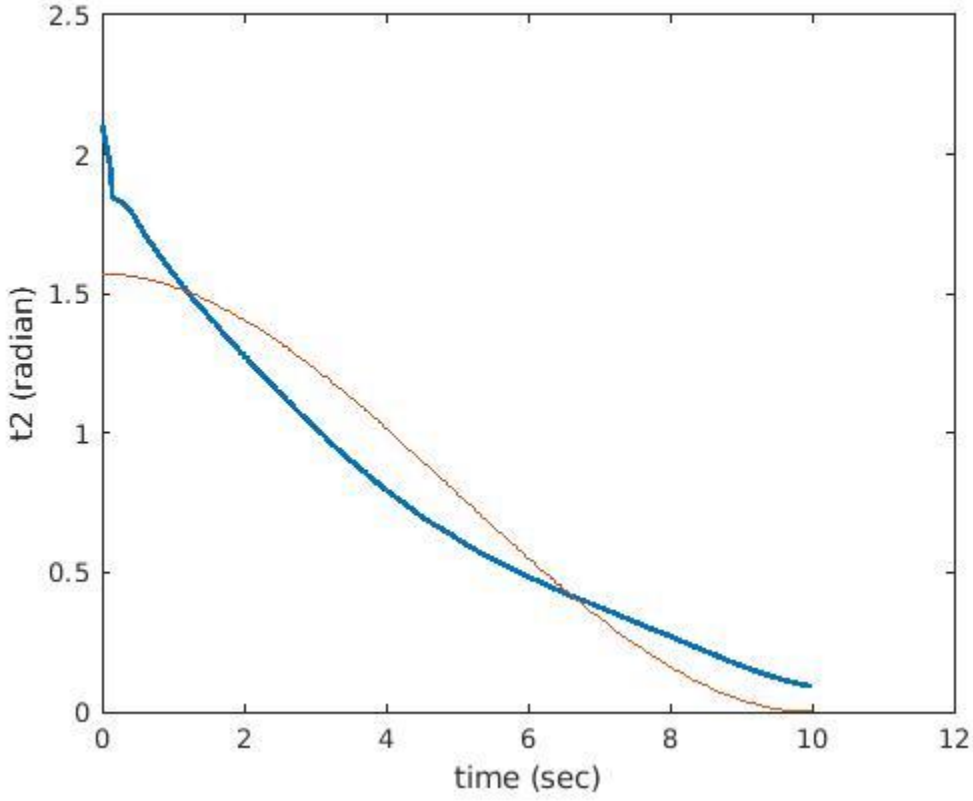

U1 v/s. Time
U1 (units)
t(sec)
10
5
0
-5
-10
-15
-20
0
1
2
3
4
5
6
7
8
9
10

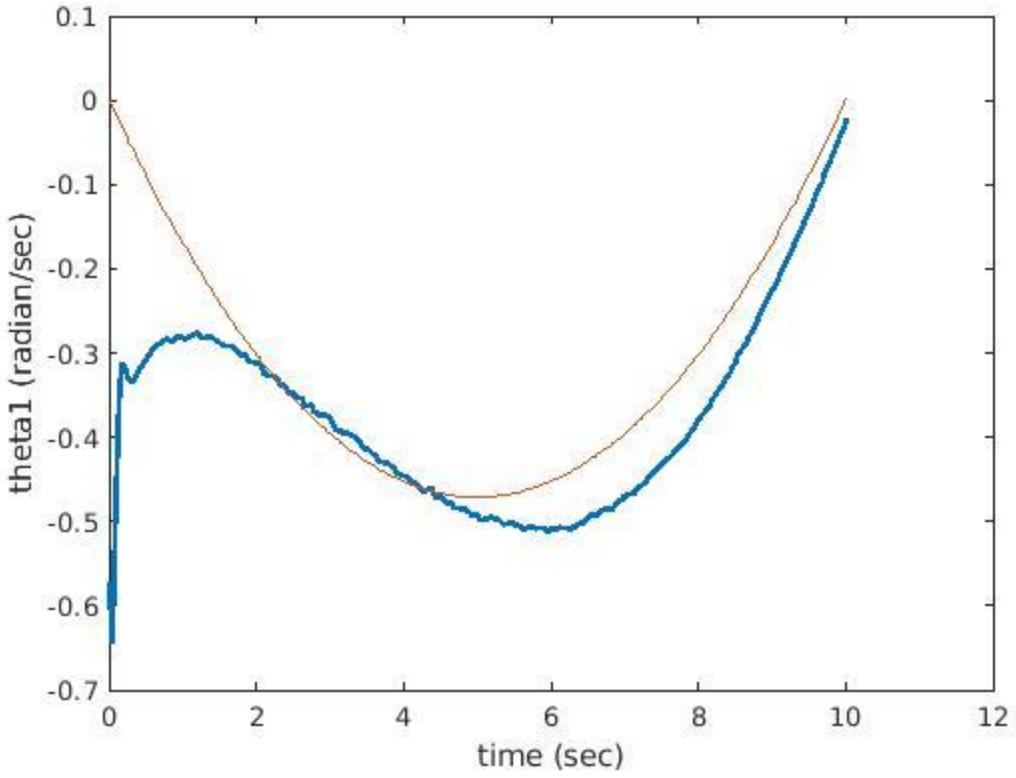


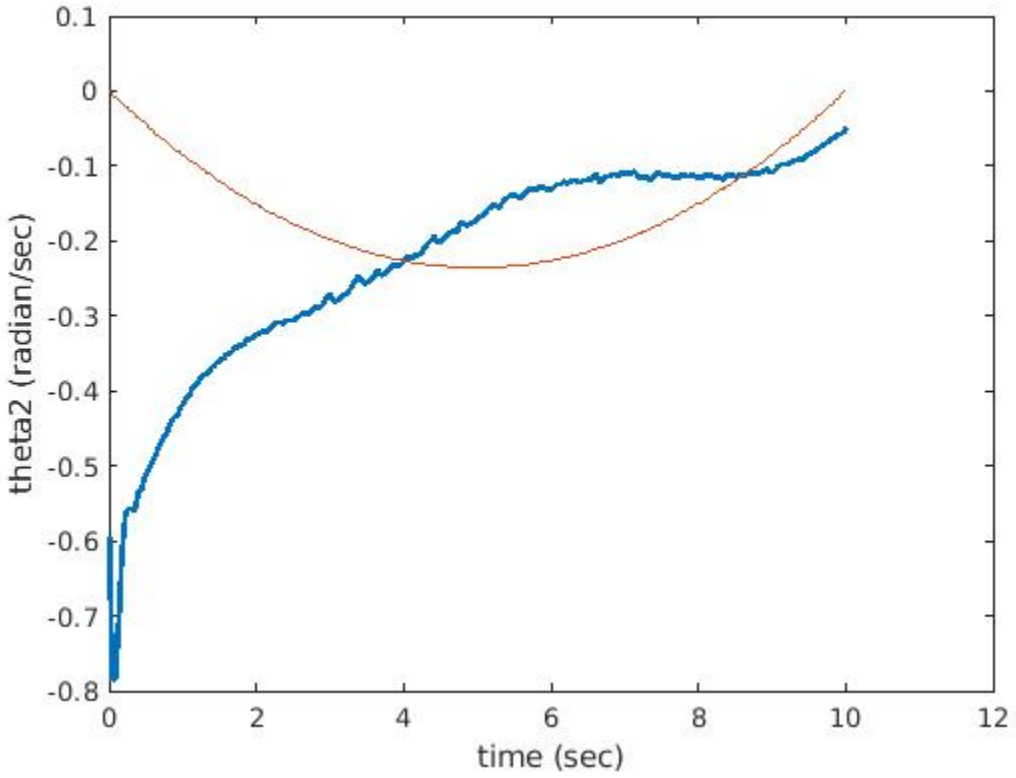
h) (5 points) Create a new copy of the rrbot_control.m file provided in Programming Assignment 2, and rename the new file to rrbot_robust_control.m. Update the code inside the while loop to control the RRbot robot in Gazebo for 10 seconds using your robust inverse dynamics controller with boundary layer designed in Step (f). Sample the data (joint angles, joint velocities and control inputs) at each loop to be plotted at the end.

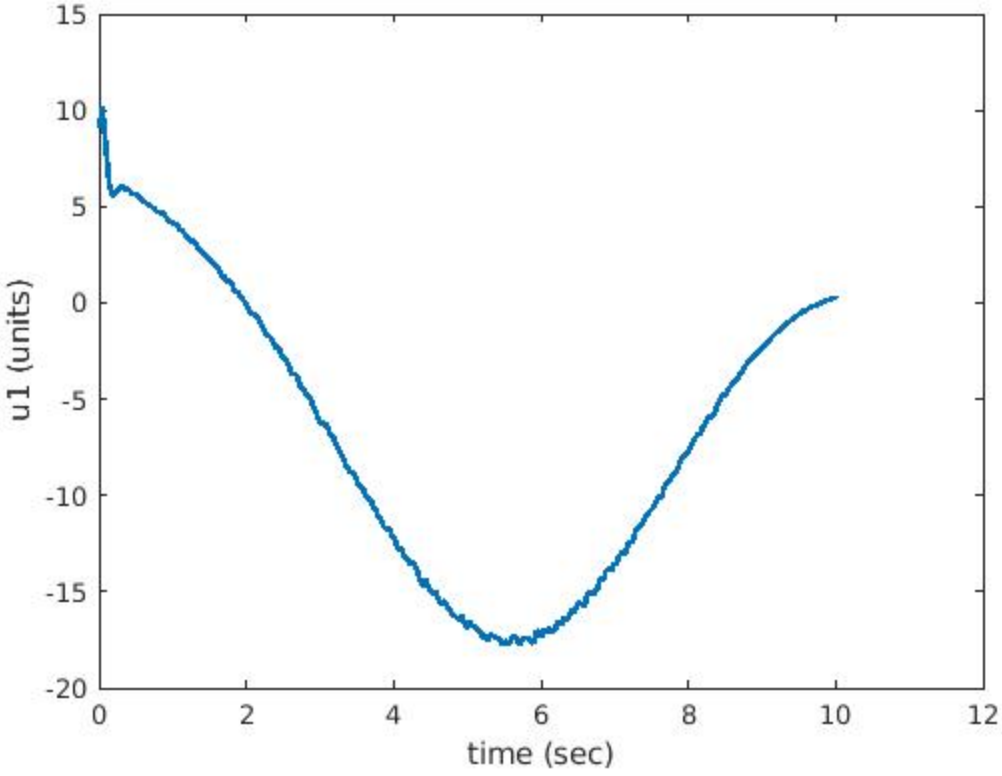
Ans.) As seen in the following images, the performance of the controller in Gazebo is in-line with the results obtained in MATLAB with the sole difference being in the ‘reaction-time’ of the simulation system for Gazebo.

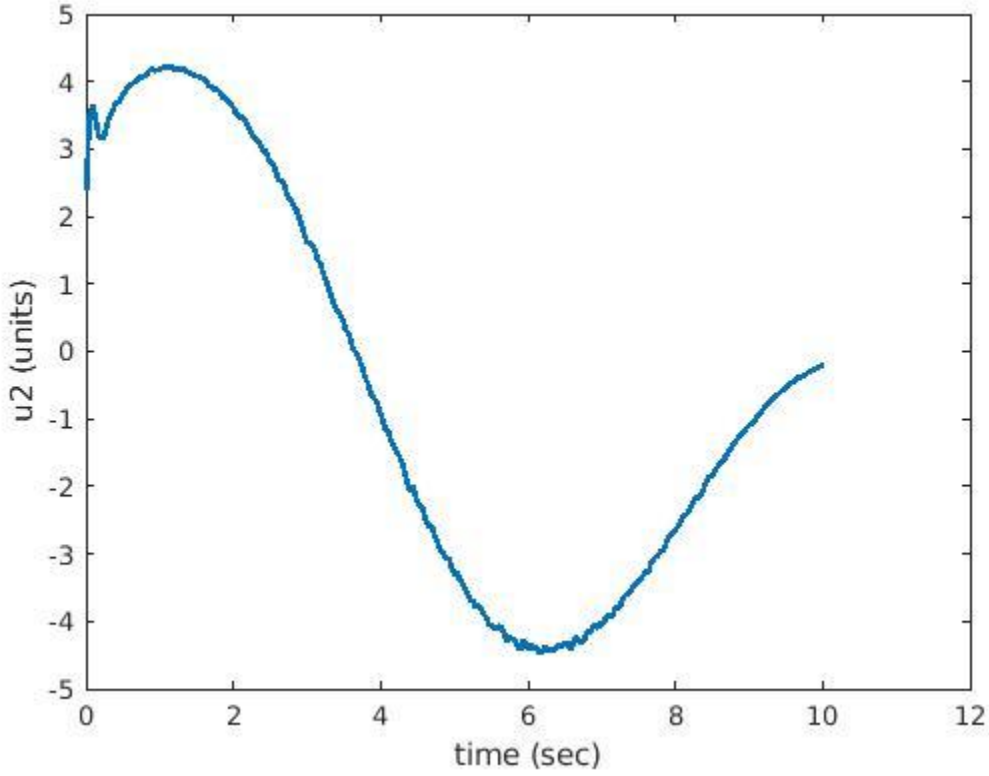

3.5
3
2.5
2
1.5
1
0.5
0
-0.5
t1 (radian)
0
2
4
6
8
10
12
time (sec)


2.5
2
1.5
1
0.5
0
t2 (radian)
0
2
4
6
8
10
12
time (sec)


0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.6
-0.7
theta1 (radian/sec)
0
2
4
6
8
10
12
time (sec)


0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.6
-0.7
-0.8
theta2 (radian/sec)
0
2
4
6
8
10
12
time (sec)


15
10
5
0
-5
-10
-15
-20
u1 (units)
0
2
4
6
8
10
12
time (sec)


5
4
3
2
1
0
-1
-2
-3
-4
-5
u2 (units)
0
2
4
6
8
10
12
time (sec)

- **Adaptive Controller:**

i) Re-write the dynamics of the robot in the linear parametric form of:

$$Y(q, \dot{q}, \ddot{q})\,\alpha = \tau$$

where $Y(q, \dot{q}, \ddot{q}) \in R2\times5$ is the regressor and $\alpha \in R5\times1$ is the parameter vector. Note that α will include lumped mass parameters from the original system.

j) (5 points) Design an Adaptive Inverse Dynamics control law along with the adaptation update law for trajectory tracking by the robot using the method described in Lecture 21. The control gains to be designed are $Kp \in R2\times2$ and $Kd \in R2\times2$ (for the virtual control input v), and the $P = P^T > 0 \in R4\times4$ and $\Gamma = \Gamma^T > 0 \in R5\times5$ for the adaptation update law $\dot{\hat{\alpha}}$.

– Use state-feedback control design to determine the control gains Kp and Kd to place the eigenvalues at $\{-3, -3, -4, -4\}$. You can use the place function in MATLAB to design the control gain matrix $K \in R2\times4$, where K = [Kp Kd].

– The matrix P is the solution to the Lyapunov equation $A^T P + PA = -Q$. You can use the lyap function in MATLAB to solve for P.

– The matrix Γ can be tuned by trial and error. An identity matrix could be used as the initial guess.

Ans.)

```
Kn =

    12     0     7     0
     0    12     0     7

P =

   13.4226         0    0.4583         0
         0   13.4226         0    0.4583
    0.4583         0    0.8512         0
         0    0.4583         0    0.8512

Gamma =

    0.4000         0         0         0         0
         0    0.4000         0         0         0
         0         0    0.4000         0         0
         0         0         0    0.4000         0
```

```
         0         0         0         0    0.4000
```

k) (5 points) Update the ode function developed in Programming Assignment 3 to implement the adaptive inverse dynamics control law and adaptation law designed in Step (j). Note that you will need to evaluate the cubic polynomial trajectories inside the ode function to obtain the desired states at each point in time.

Ans.) Control Law: `Tau = Yo*alpha_hat;`

Adaptive Law:

```
M1 = [1,2*cos(t2),0,0,0];
M2 = [0,cos(t2),1,0,0];
M3 = [0,cos(t2),1,0,0];
M4 = [0,0,1,0,0];
M_hat = [[M1;M3]*alpha_hat,[M2;M4]*alpha_hat];

B = [0,0;0,0;1,0;0,1];
Phi = M_hat\Y;
alpha_dot = -Gamma\(Phi'*B'*P*x);
```

l) (5 points) Use ode45 and the ode function developed in Step (k) to construct a simulation of the system in MATLAB with the time span of [0,10] sec and initial conditions of:

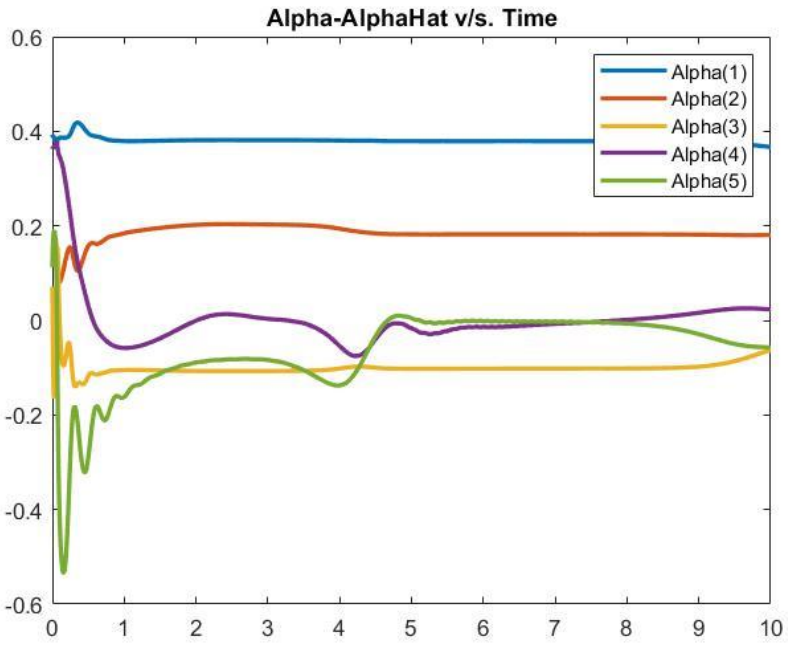
$$\theta 1(0) = 200^{\circ},\ \theta 2(0) = 125^{\circ},\ \dot{\theta} 1 = 0,\ \dot{\theta} 2 = 0$$

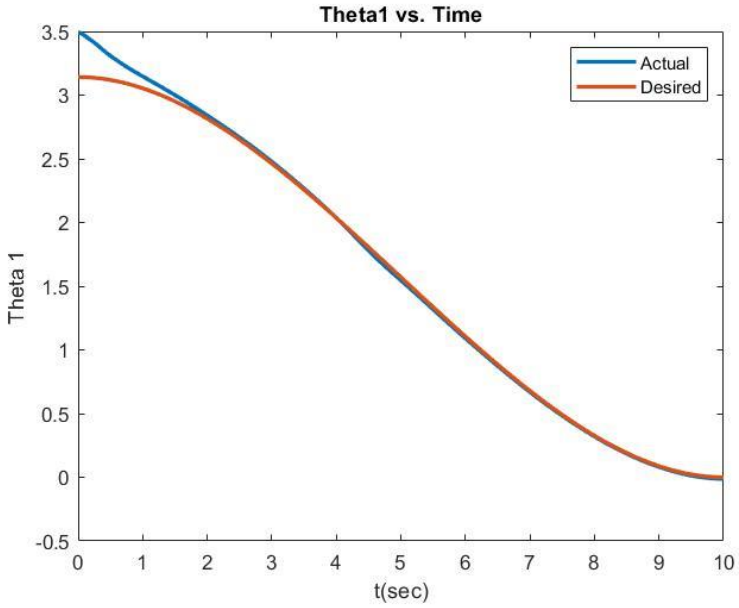
Set the initial values of the unknown parameter vector α to 75% of the actual values:

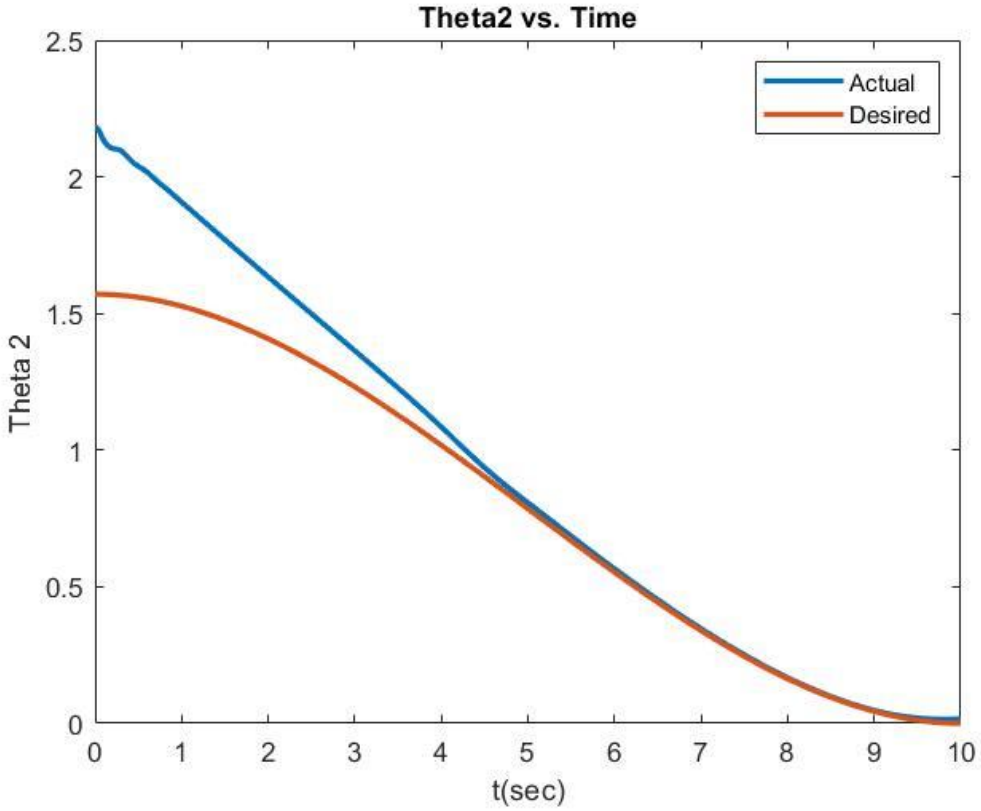
$$\hat{\alpha}(0) = 0.75\alpha$$

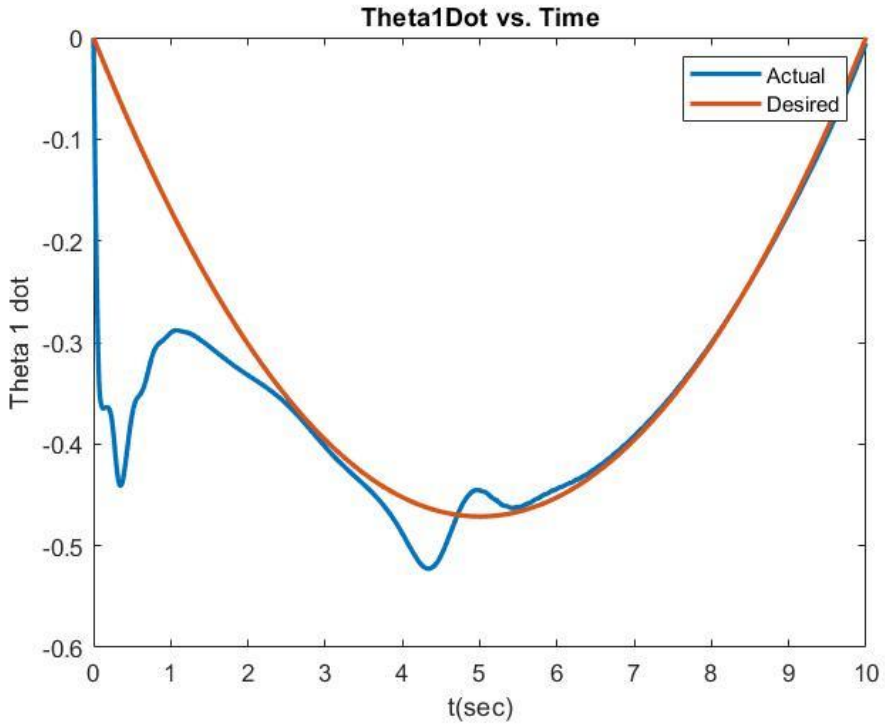
Plot the state trajectories, the control inputs trajectories, and the associated desired trajectories to evaluate the performance. Moreover, plot the time-varying estimated parameters (i.e. $\hat{\alpha}(t)$) over the time-span, and discuss the results in your final report (do the parameters converge to their actual values? please discuss). If the performance is not satisfactory (i.e. the system does not track and converge to the desired trajectory), go back to Step (j) and tune the design parameters.

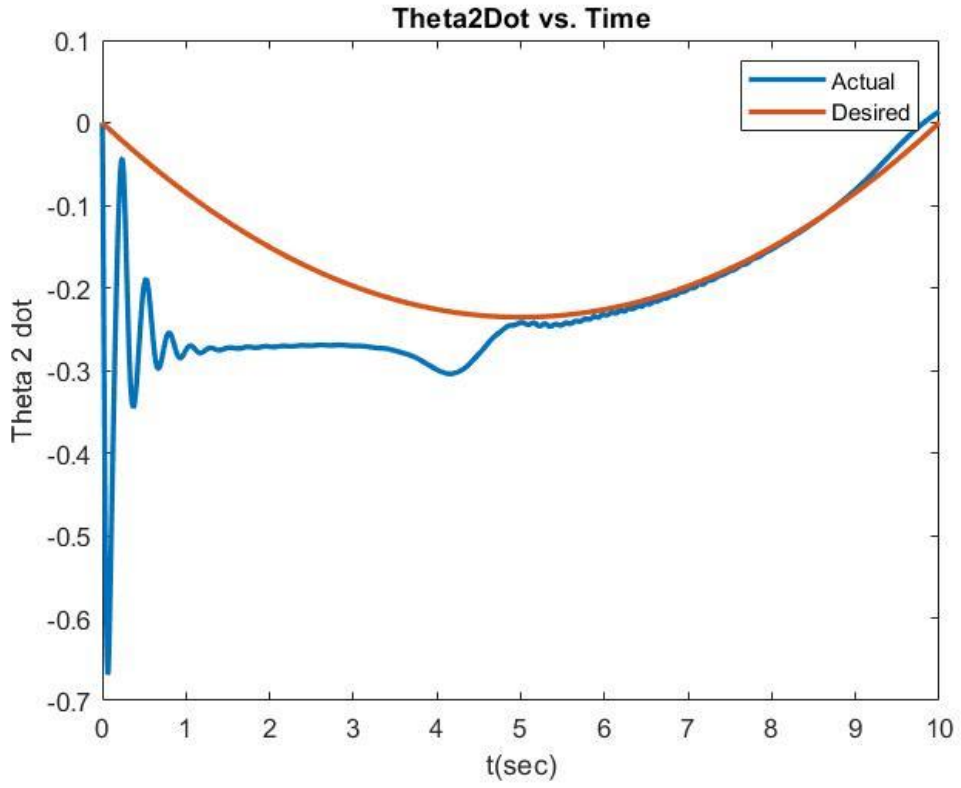
Ans.) The value of the parameters do not converge to the actual values but attain a steady state in between. The same can be seen in the following plot of Alpha – AlphaHat.

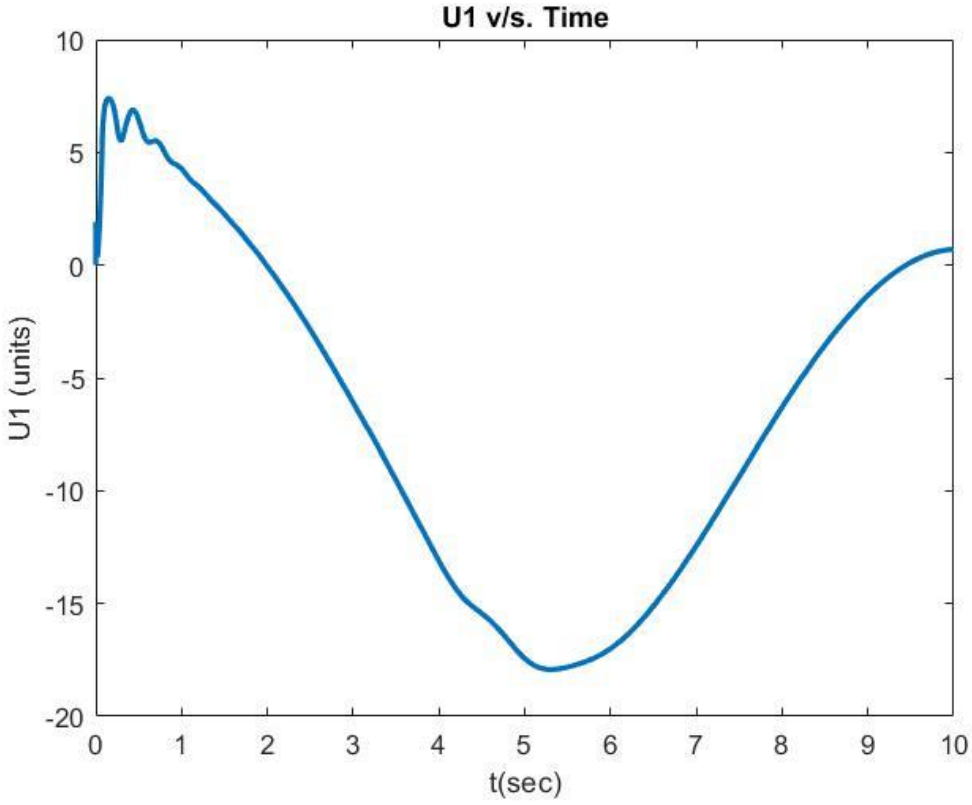

Alpha-AlphaHat v/s. Time
Alpha(1)
Alpha(2)
Alpha(3)
Alpha(4)
Alpha(5)
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
0
1
2
3
4
5
6
7
8
9
10

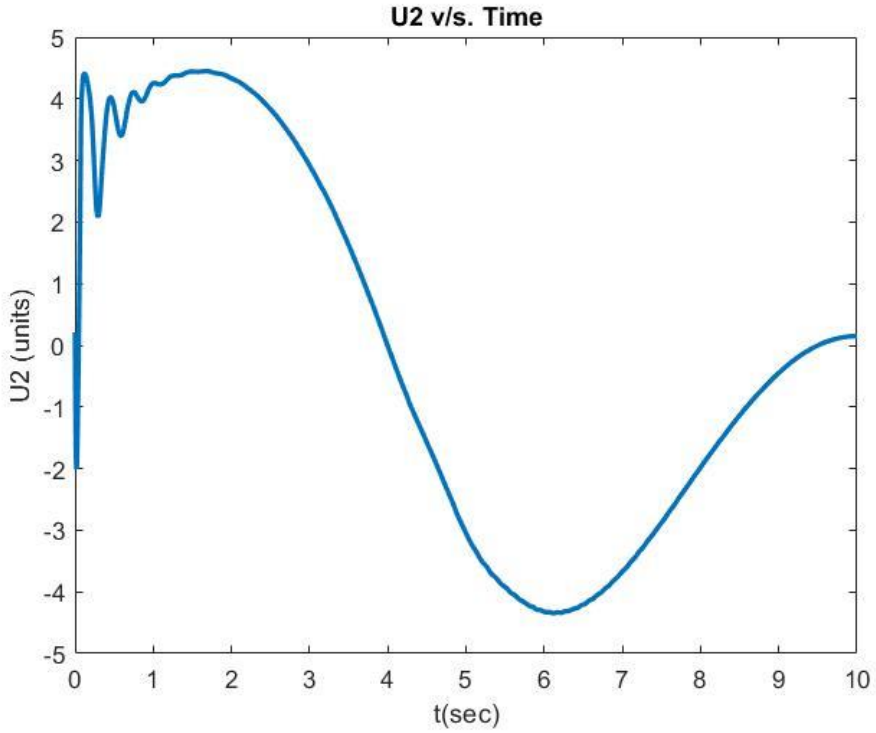

Theta1 vs. Time
Actual
Desired
Theta 1
t(sec)
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
0
1
2
3
4
5
6
7
8
9
10


Theta2 vs. Time
Actual
Desired
Theta 2
t(sec)
2.5
2
1.5
1
0.5
0
0
1
2
3
4
5
6
7
8
9
10


Theta1Dot vs. Time
Actual
Desired
Theta 1 dot
t(sec)
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.6
0
1
2
3
4
5
6
7
8
9
10


Theta2Dot vs. Time
Actual
Desired
Theta 2 dot
t(sec)
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.6
-0.7
0
1
2
3
4
5
6
7
8
9
10

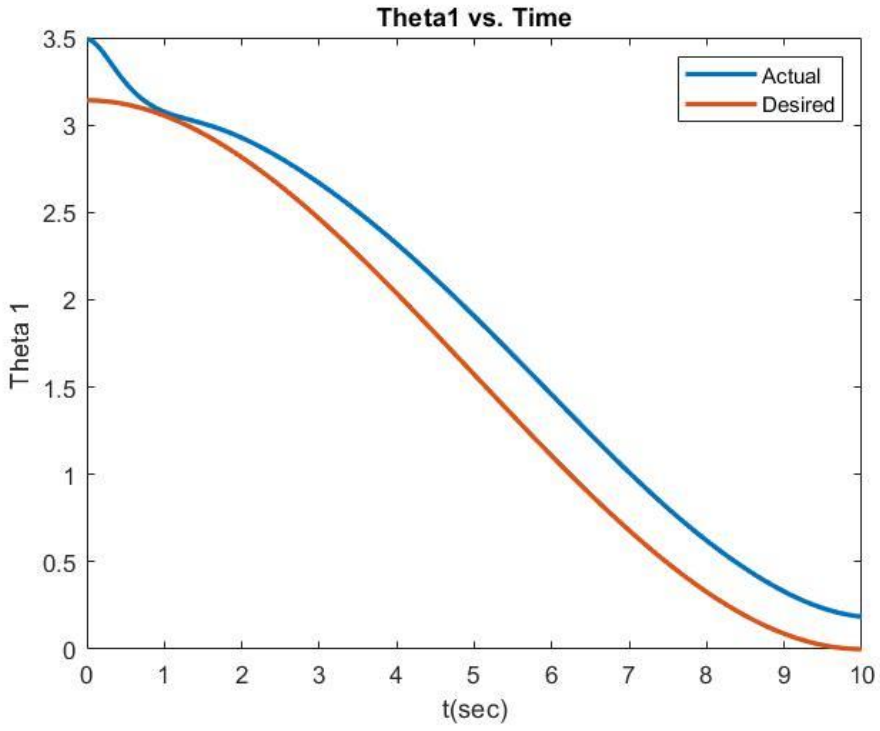

U1 v/s. Time
U1 (units)
t(sec)
10
5
0
-5
-10
-15
-20
0
1
2
3
4
5
6
7
8
9
10

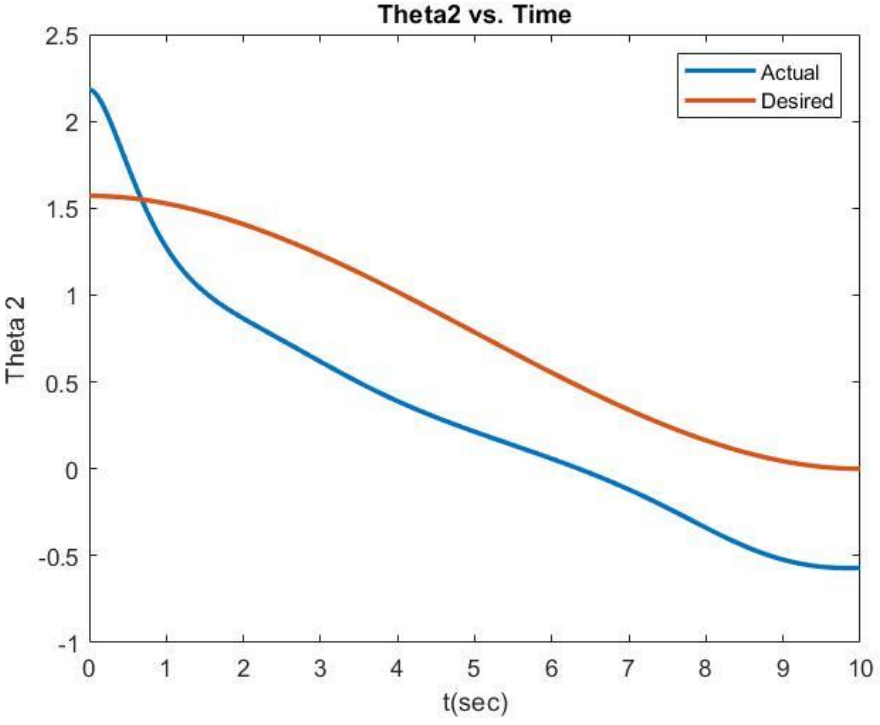


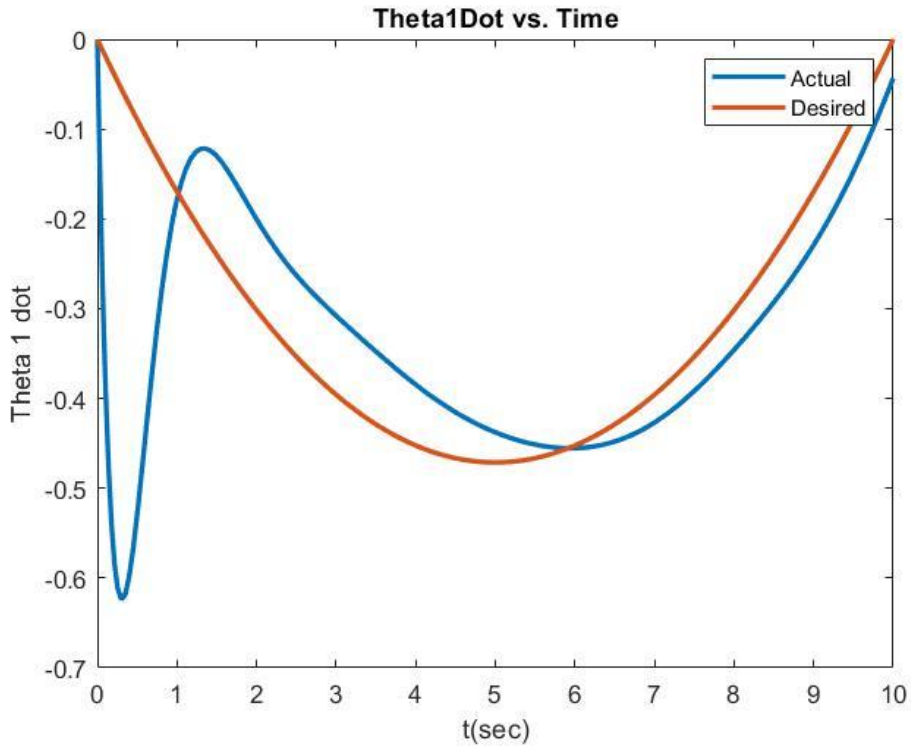
m) (2.5 points) To evaluate the performance of the robot without the adaptive inverse dynamics control, construct a simulation of the system in MATLAB with the same control law but with the P matrix set to zero (do not change other design parameters such as Kp and Kd). Again, plot the state trajectories, the control inputs trajectories, and the associated desired trajectories to compare the resulting performance with the performance obtained in Step (l). Discuss the results in your final report.

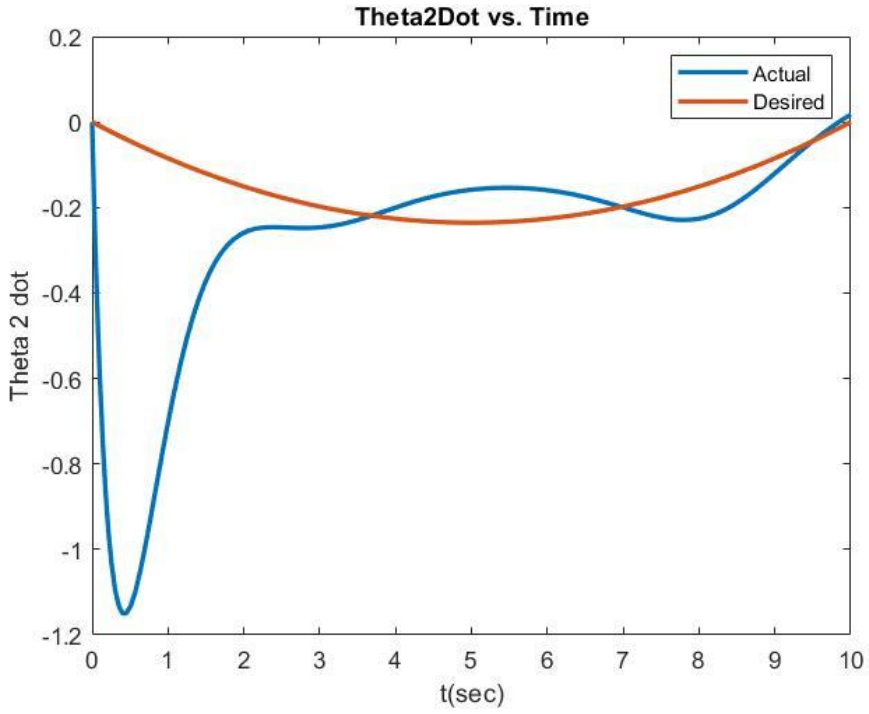
Ans.) `P = eye(4)*0;`

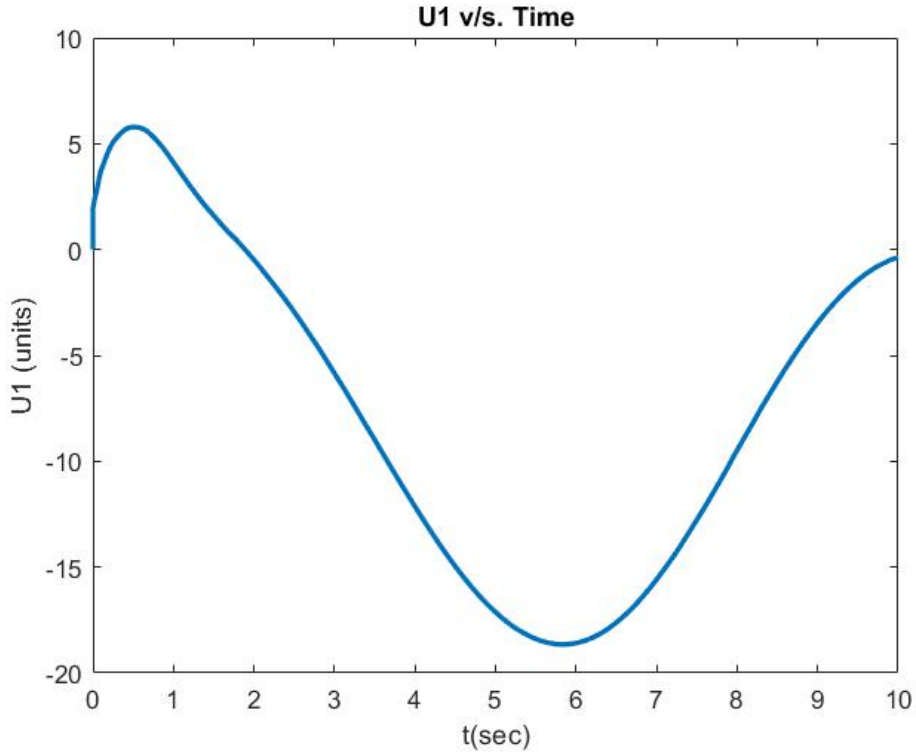
When the value of P is taken as zero, the states and trajectories never actually converge to zero and there appears to be a constant steady-state error in the trajectory. This may be due to the fact that the P term is present in the Adaption law which holds the error term as well and the overall function works towards reducing the error value to zero.

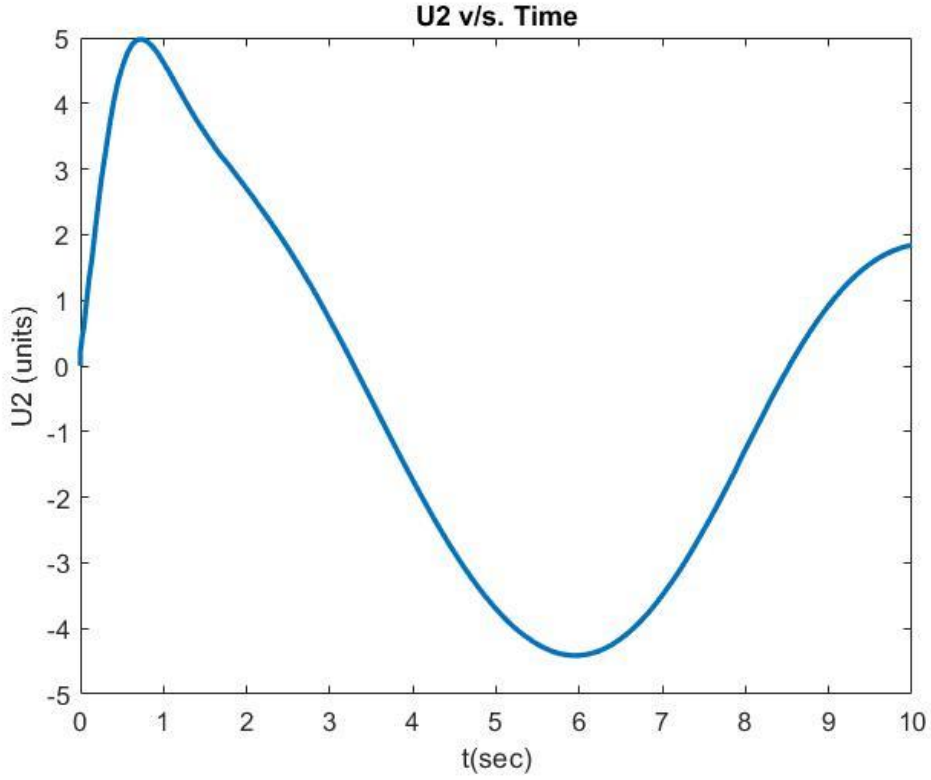

Theta1 vs. Time
Actual
Desired
Theta 1
t(sec)
3.5
3
2.5
2
1.5
1
0.5
0
0
1
2
3
4
5
6
7
8
9
10


Theta2 vs. Time
Actual
Desired
Theta 2
t(sec)
2.5
2
1.5
1
0.5
0
-0.5
-1
0
1
2
3
4
5
6
7
8
9
10


Theta1Dot vs. Time
Actual
Desired
Theta 1 dot
t(sec)
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.6
-0.7
0
1
2
3
4
5
6
7
8
9
10

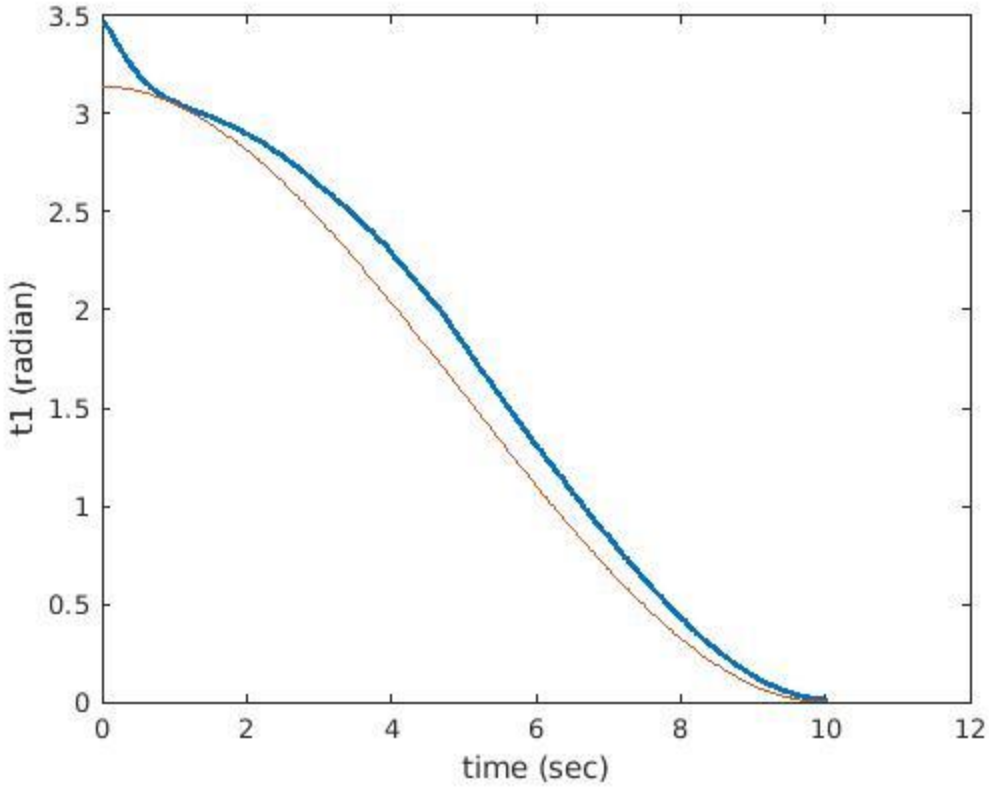

Theta2Dot vs. Time
Actual
Desired
Theta 2 dot
t(sec)
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
0
1
2
3
4
5
6
7
8
9
10

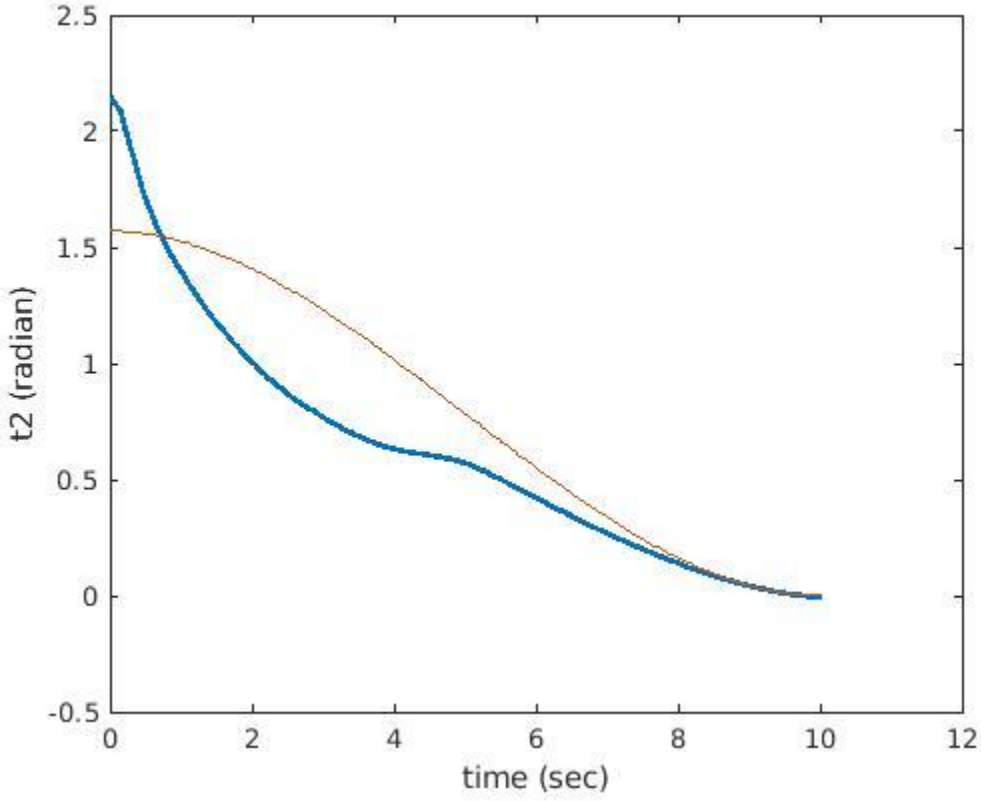

U1 v/s. Time
U1 (units)
t(sec)
10
5
0
-5
-10
-15
-20
0
1
2
3
4
5
6
7
8
9
10

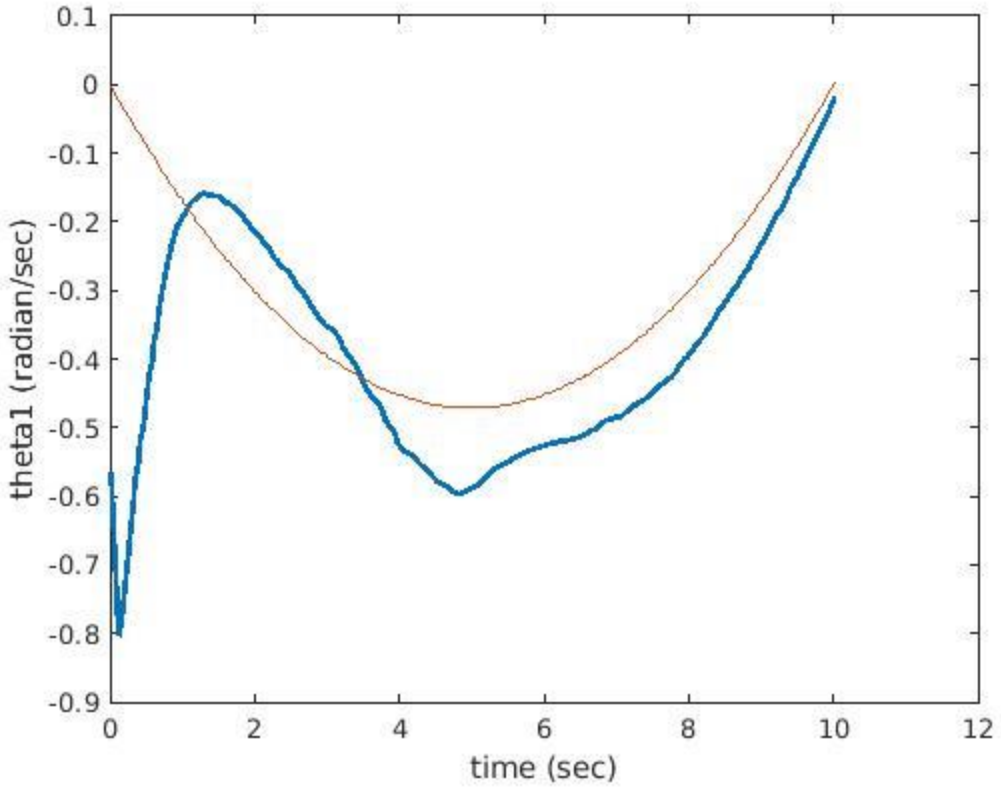

U2 v/s. Time
U2 (units)
t(sec)
5
4
3
2
1
0
-1
-2
-3
-4
-5
0
1
2
3
4
5
6
7
8
9
10

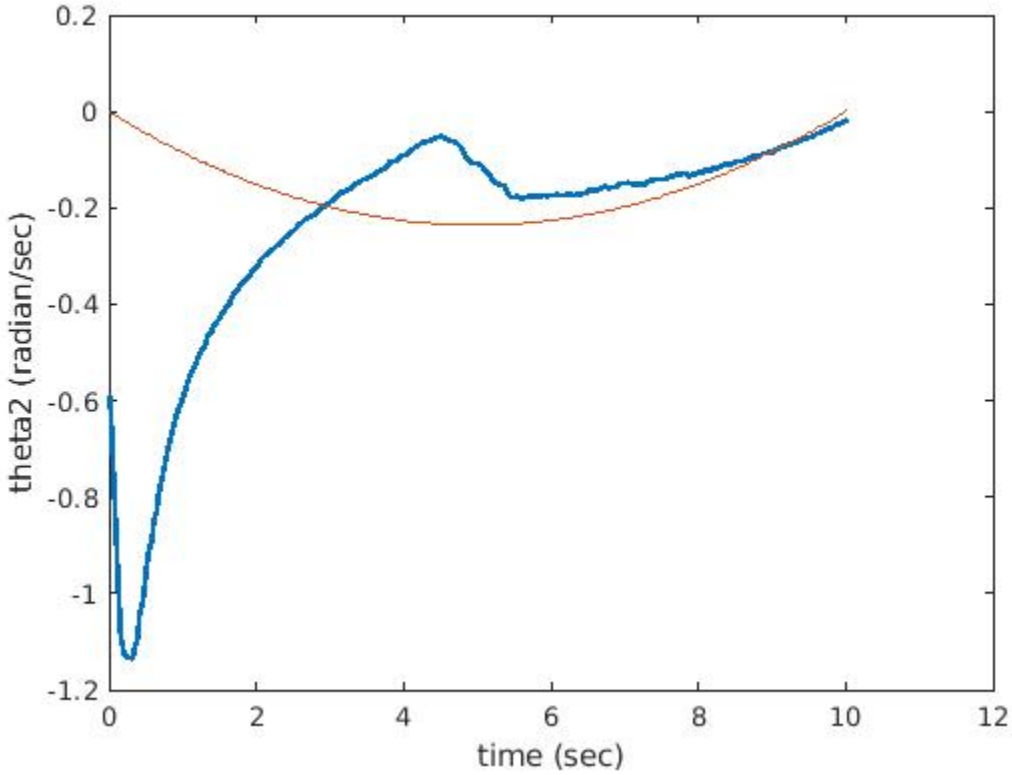
n) (5 points) Create a new copy of the rrbot_control.m file provided in Programming Assignment 2, and rename the new file to rrbot_adaptive_control.m. Update the code inside the while loop to control the RRbot robot in Gazebo for 10 seconds using your adaptive inverse dynamics controller designed in Step (k). Sample the data (joint angles, joint velocities and control inputs) at each loop to be plotted at the end. Feel free to define new functions and variables in your program if needed. Compare the resulting trajectories and control inputs in Gazebo with those obtained in Step (k) in MATLAB. If the Gazebo performance is not satisfactory, re-tune the design parameters to achieve the desired performance (as the last resort, you can reduce the initial uncertainty in $\hat{\alpha}(0)$ if needed, although not preferred). Discuss your findings in your final report.

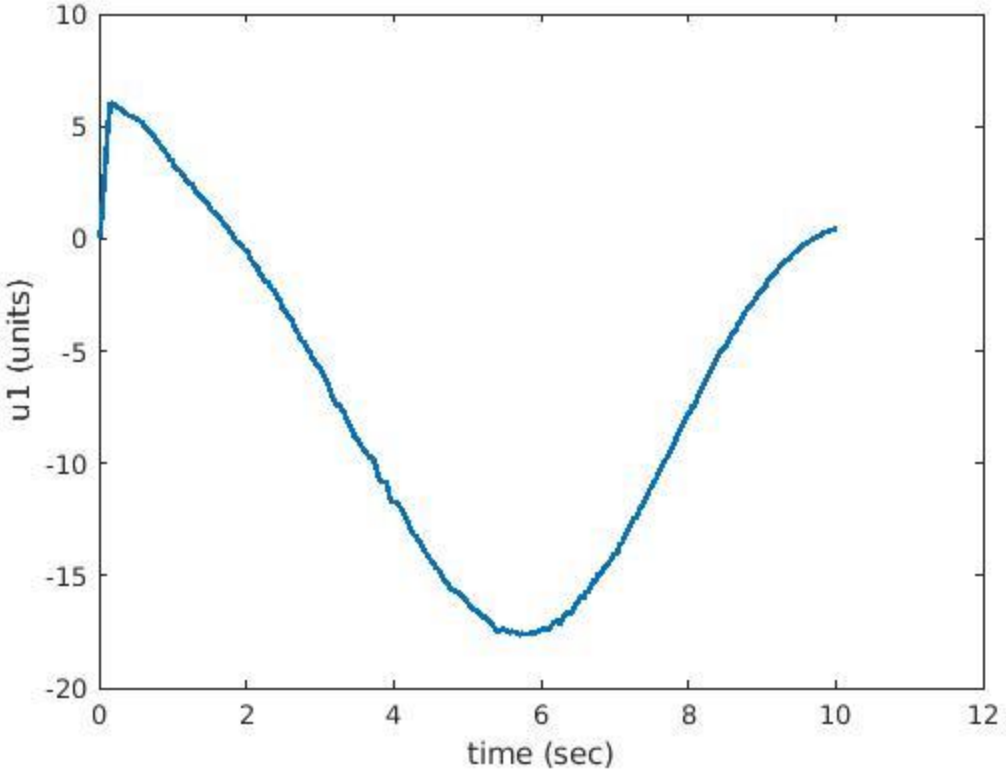
Ans.) As seen in the following images, the performance of the controller in Gazebo is in-line with the results obtained in MATLAB with the sole difference being in the ‘reaction-time’ of the simulation system for Gazebo.

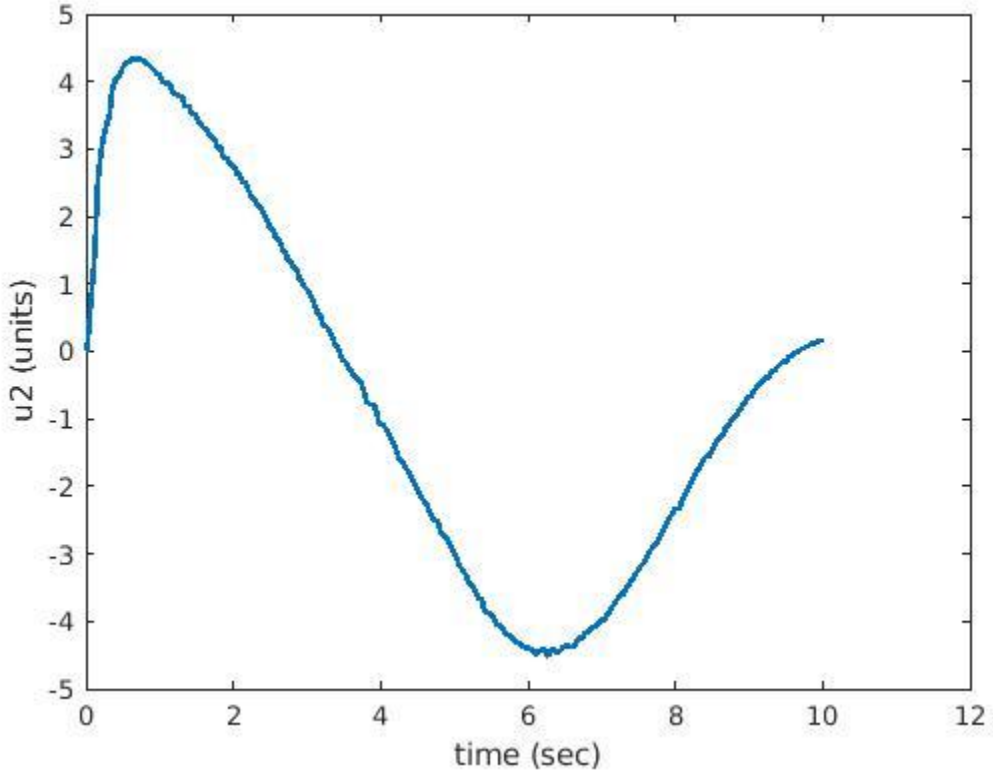

t1 (radian)
time (sec)
3.5
3
2.5
2
1.5
1
0.5
0
0
2
4
6
8
10
12


t2 (radian)
time (sec)
2.5
2
1.5
1
0.5
0
-0.5
0
2
4
6
8
10
12


0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.6
-0.7
-0.8
-0.9
theta1 (radian/sec)
0
2
4
6
8
10
12
time (sec)


0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
theta2 (radian/sec)
0
2
4
6
8
10
12
time (sec)


10
5
0
-5
-10
-15
-20
u1 (units)
0
2
4
6
8
10
12
time (sec)


5
4
3
2
1
0
-1
-2
-3
-4
-5
u2 (units)
0
2
4
6
8
10
12
time (sec)