RBE502 – Robot Control

Programming Assignment -3

Submitted by:
Suketu Parekh

a) (5 points) 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:

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

$$\theta 1(t0) = 180^{\circ},\ \theta 1(tf) = 0,\ \theta 2(t0) = 90^{\circ},\ \theta 2(tf) = 0$$

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

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) (5 points) 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.

c) (10 points) Derive the symbolic feedback linearization of the robot. Then, design a feedback linearization control for the robot, with a state-feedback control (or a PD control) for the virtual control input.

Ans.)
M*q_ddot + C*q_dot + g = T
Let virtual input v be such that:
T = M*v + C*q_dot + g
Thus, v = q_ddot
Let x = [q; q_dot]
Therefore, x_dot = [q_dot; q_ddot] = [q_dot; v] = [x3; x4; v1; v2]
x_dot = A*x + B*v

```
lambda = [-3;-7;-5-j;-5+j];
```

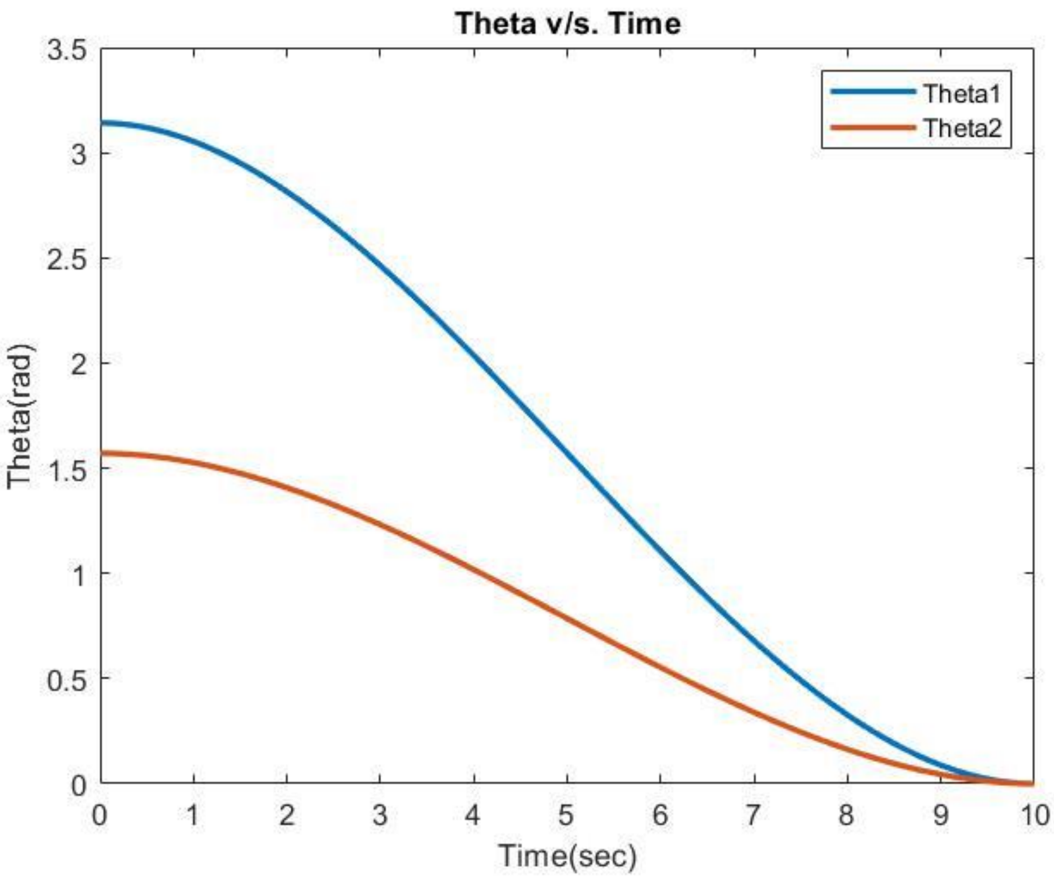
The value of k is as follows:

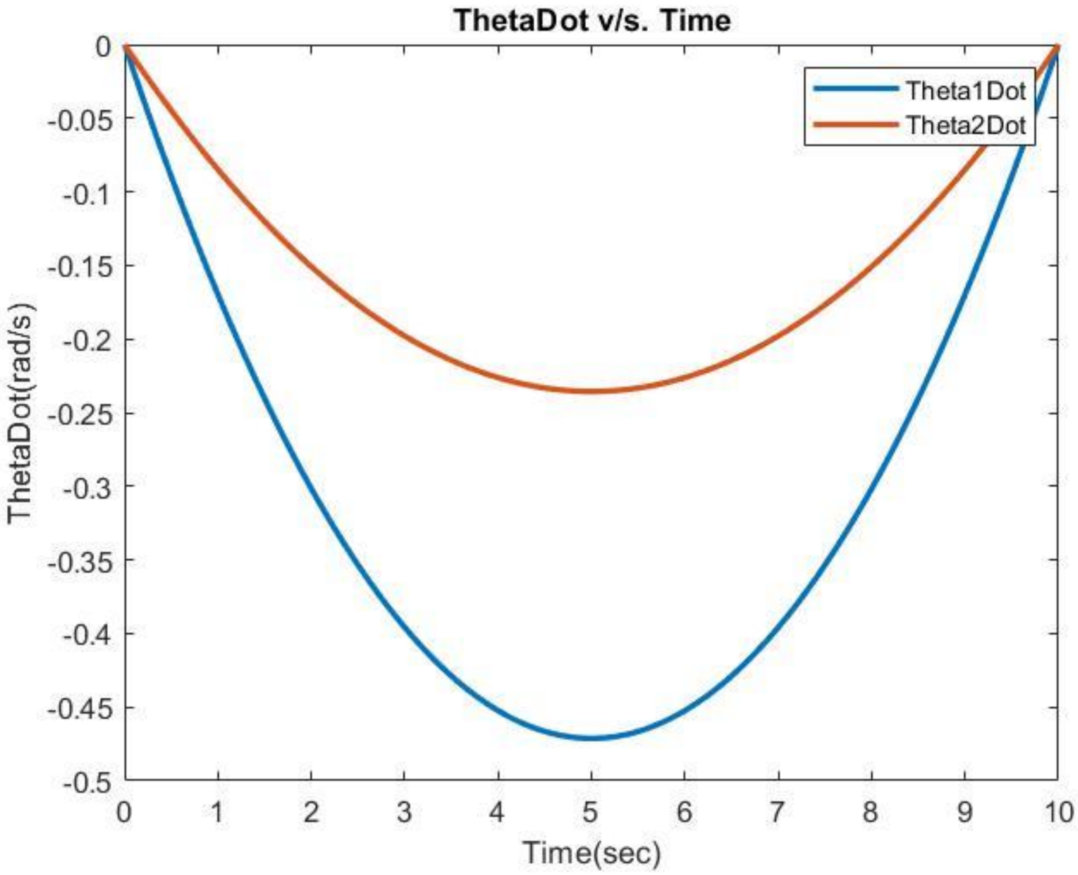
```
Kn =

   26.3636   -0.9759   10.2714   -0.0273
    0.8324   20.6796    1.1149    9.7286

v = [v1;v2]

v1 = (23306712829707163*pi*t^3)/4503599627370496 00 -
(32188277915878674 9*pi*t^2)/450359962737049600 -
(8492962916105073 9*pi*t)/140737488355328000 +
(29065837042723396 31*pi)/112589990684262400 -
(7420691724886443*t1)/281474976710656 +
(2197503449647273*t2)/2251799813685248 -
```

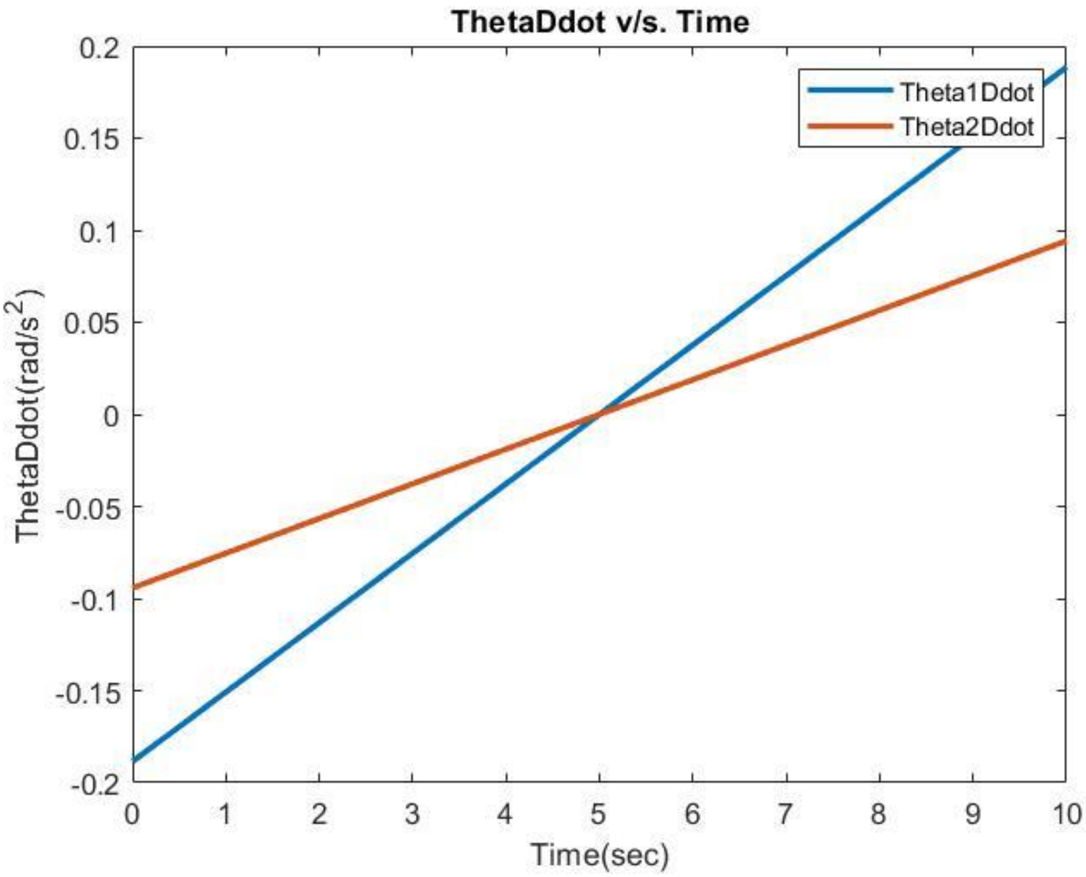
```
(5782259037395969*t1_dot)/562949953421312 +
(1923442327081*t2_dot)/70368744177664


v2 = (2515758614575471 3*pi*t^3)/1125899906842624000 -
(3369714910289478 63*pi*t^2)/1125899906842624000 -
(6205743939573009*pi*t)/17592186044416000 +
(6272508037836038 89*pi)/56294995342131200 -
(1874465541119461*t1)/2251799813685248 -
(5820780151158813*t2)/281474976710656 -
(627655080932601*t1_dot)/562949953421312 -
(2738370015515135*t2_dot)/281474976710656
```

d) (5 points) Update the ode function developed in Programming Assignment 2 to include the feedback linearization control law designed in part (c). Moreover, 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

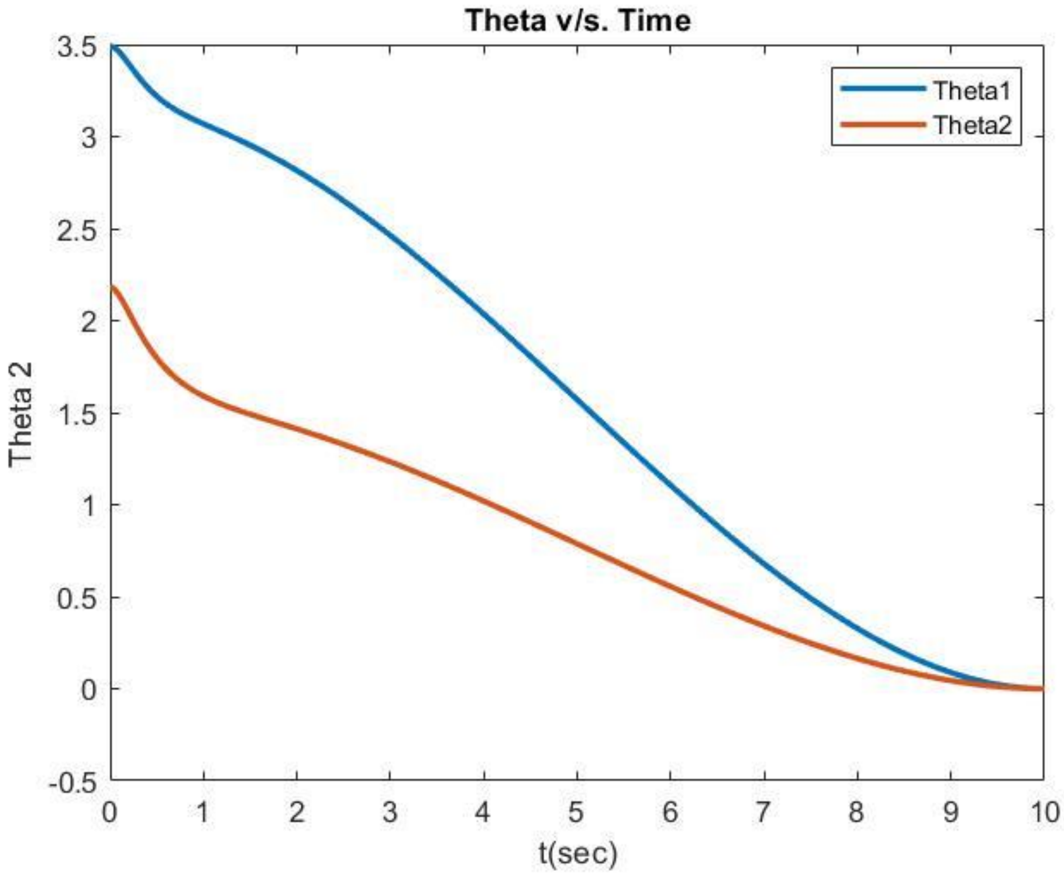
e) (10 points) Use ode45 and the ode function developed in part (d) to construct a simulation of the system in MATLAB with the time span of [0,10] sec and initial conditions of:

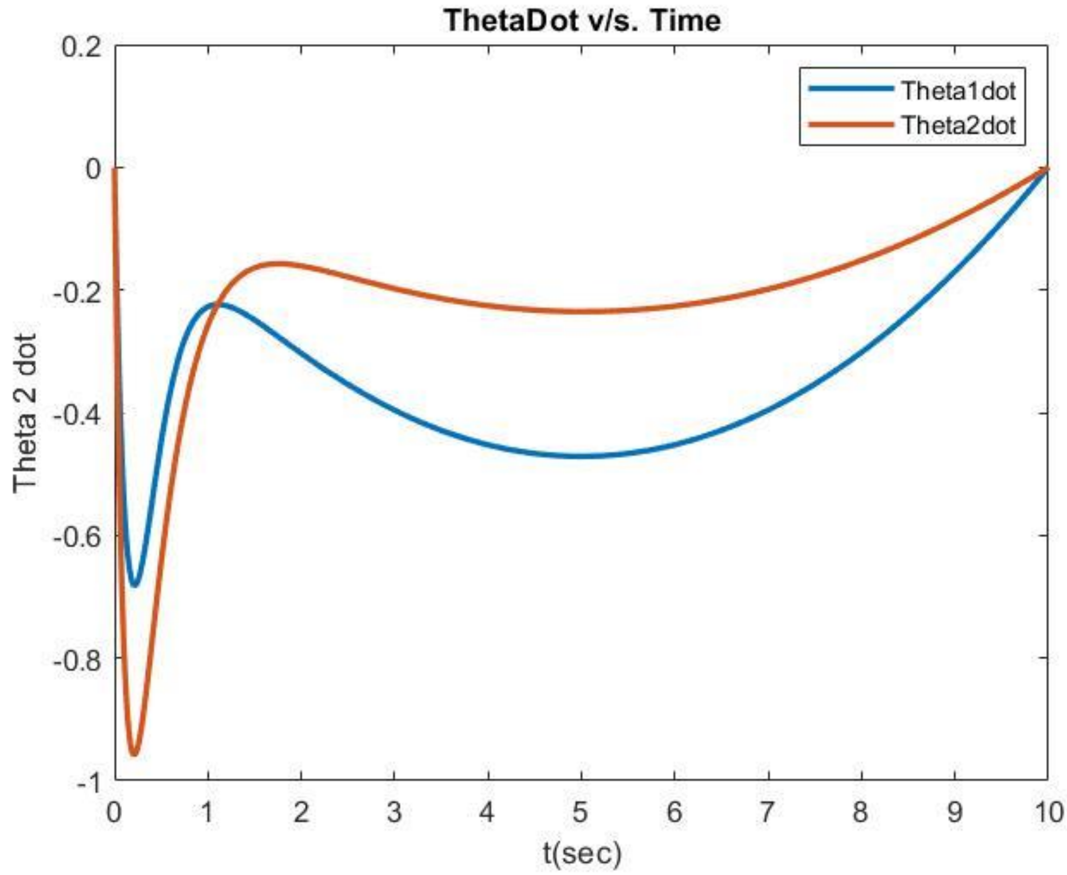
$$\theta_1(0) = 200^\circ, \theta_2(0) = 125^\circ, \dot{\theta}_1 = 0, \dot{\theta}_2 = 0$$

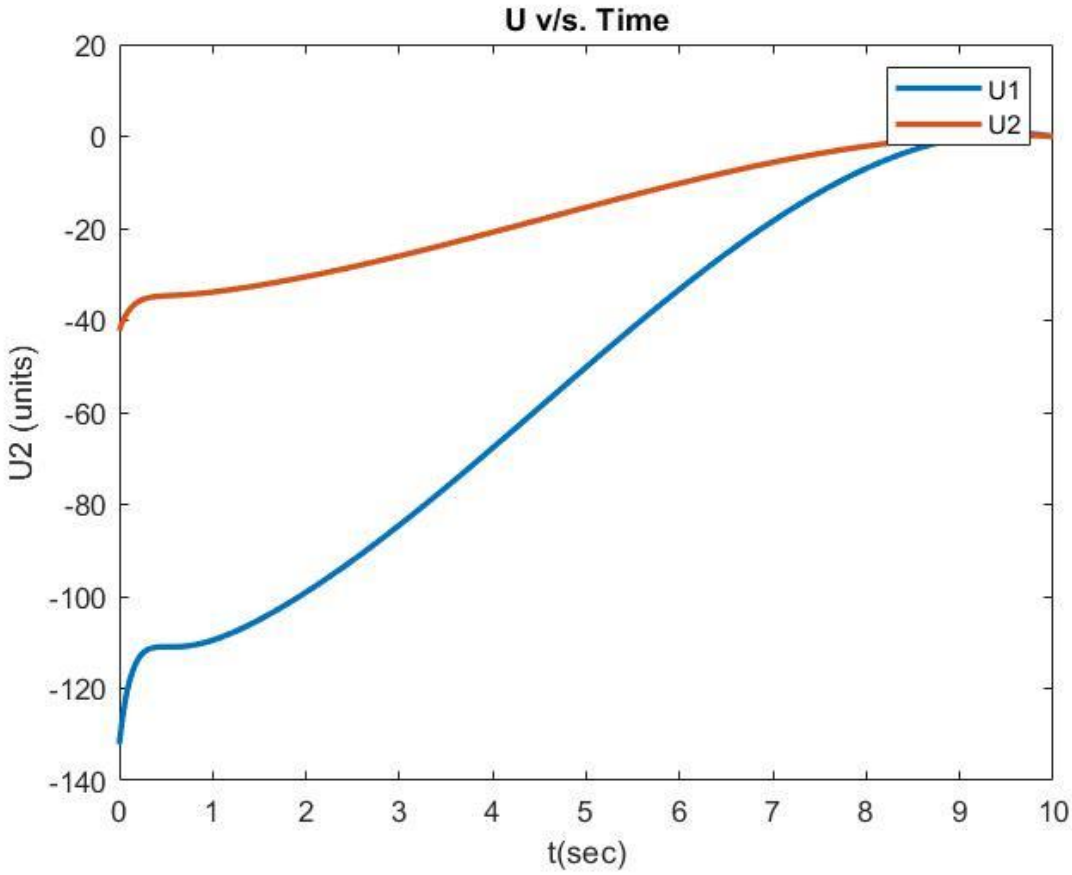
Plot the state trajectories and the control inputs trajectories and the associated desired trajectories to evaluate the performance. If the performance is not satisfactory (i.e. the system does not track and converge to the desired trajectory), go back to Step (c) and update the state-feedback control gains for the virtual control input.

Hint: The control inputs can be reconstructed after the solution is returned by ode45.

Ans.)

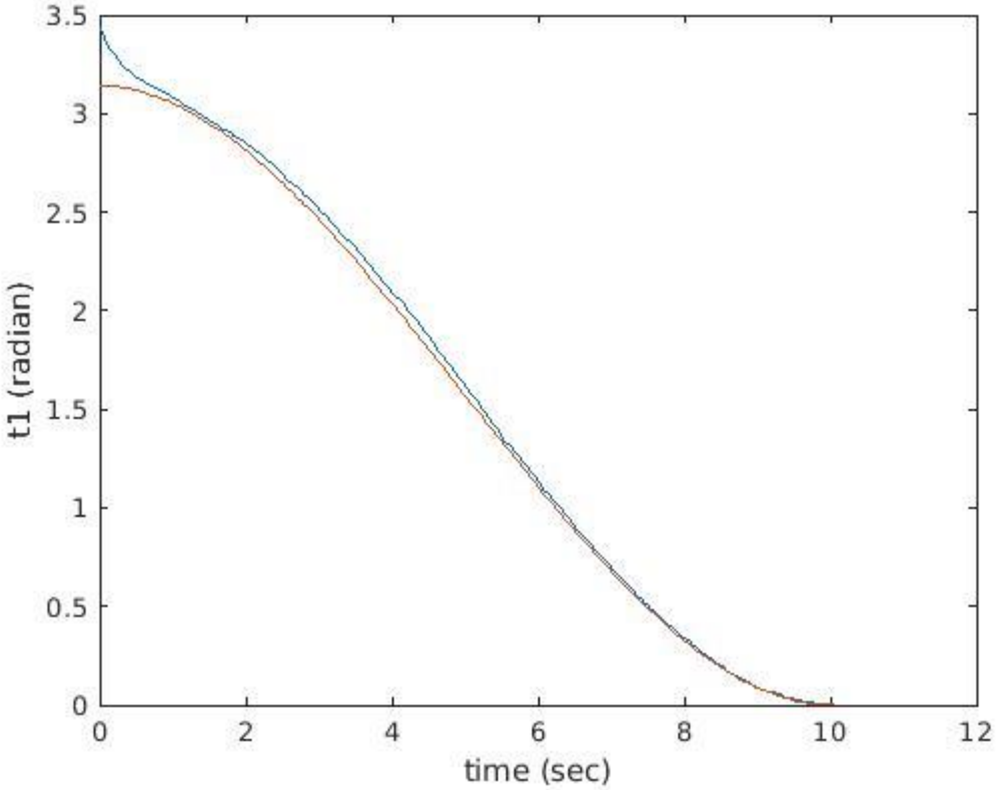


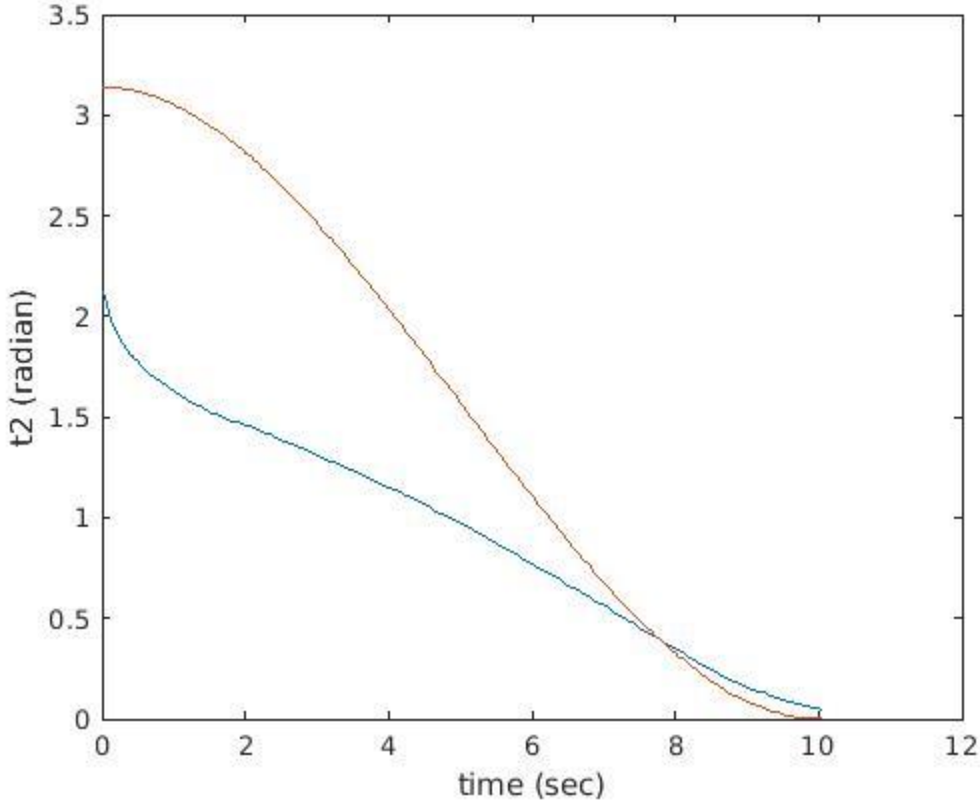

ThetaDot v/s. Time
Theta1dot
Theta2dot
Theta 2 dot
t(sec)
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6
7
8
9
10

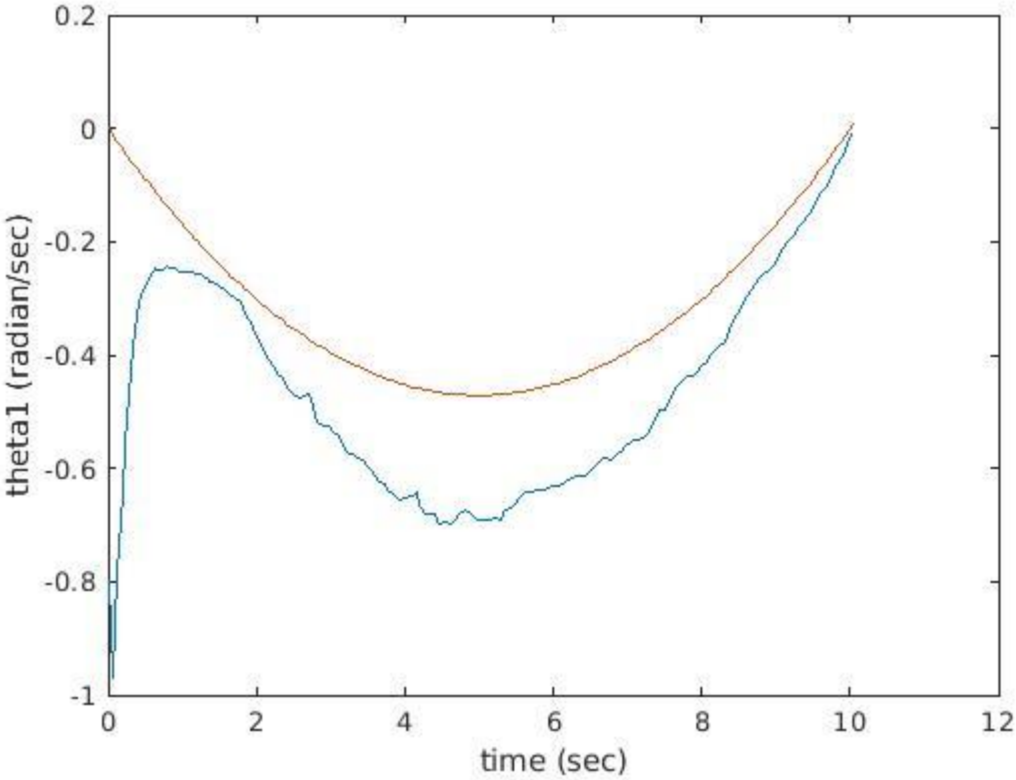

U v/s. Time
U1
U2
U2 (units)
t(sec)
20
0
-20
-40
-60
-80
-100
-120
-140
0
1
2
3
4
5
6
7
8
9
10

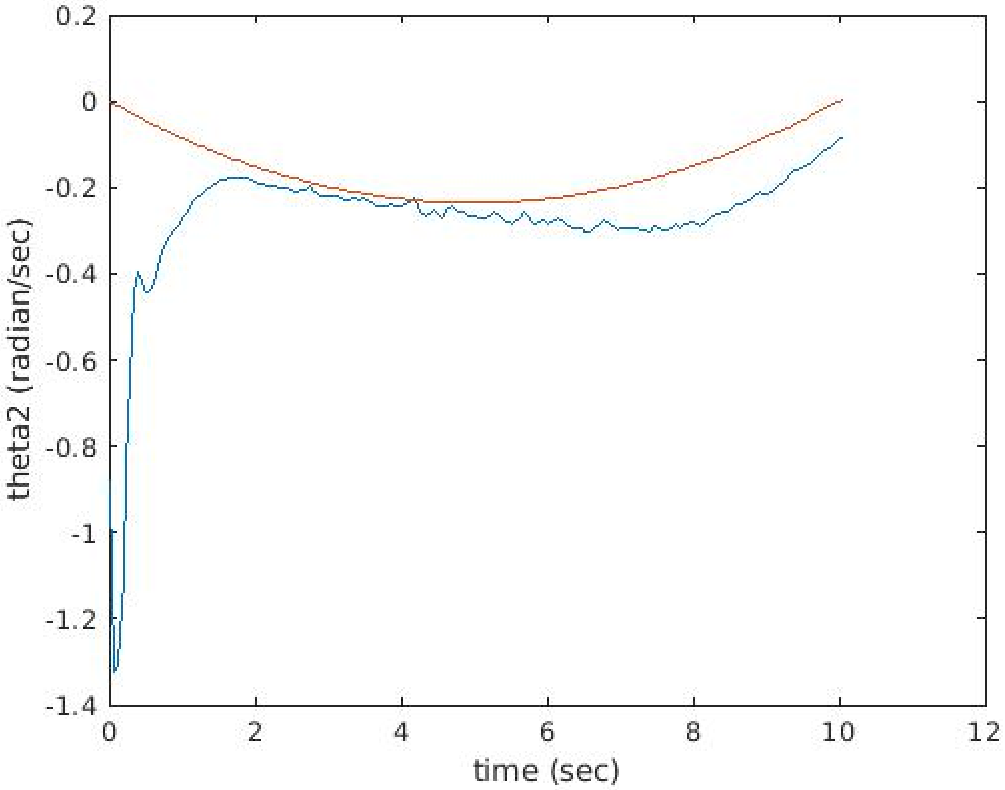
f) (15 points) Create a new copy of the rrbot_control.m file provided in Programming Assignment 2, and rename the new file to rrbot_traj_control.m. Update the code inside the while loop to control the RRbot robot in Gazebo for 10 seconds using your feedback linearization controller designed in Step (e). Sample the data 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 (e) in MATLAB. Discuss your findings in your final report.

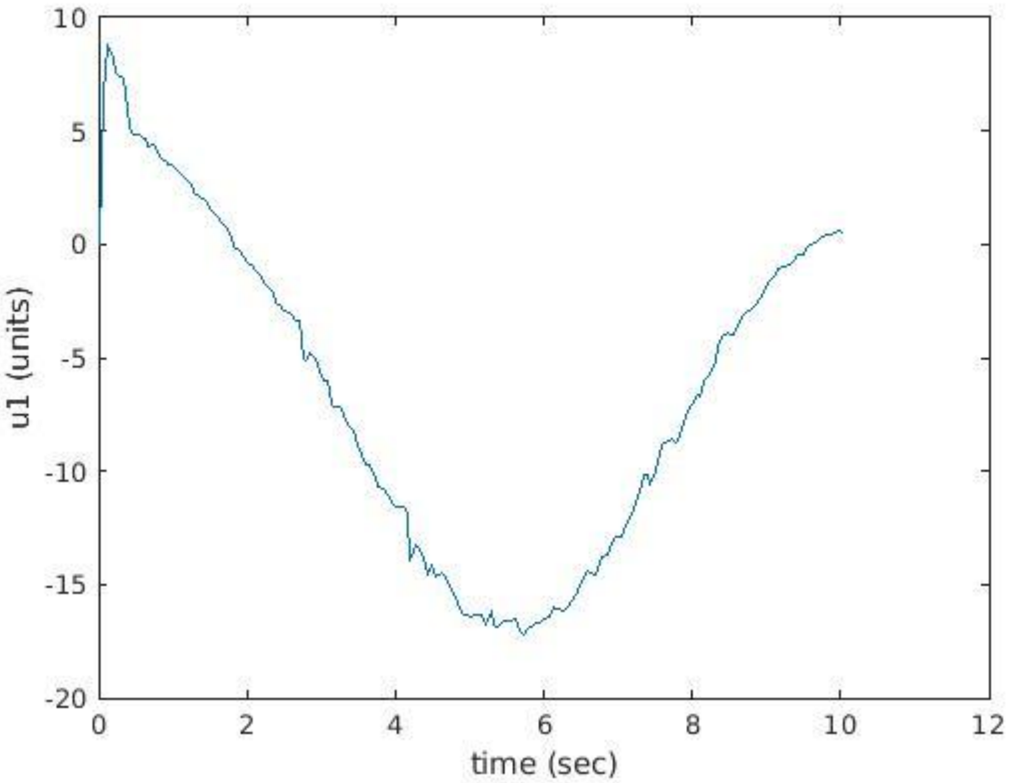
Ans.)

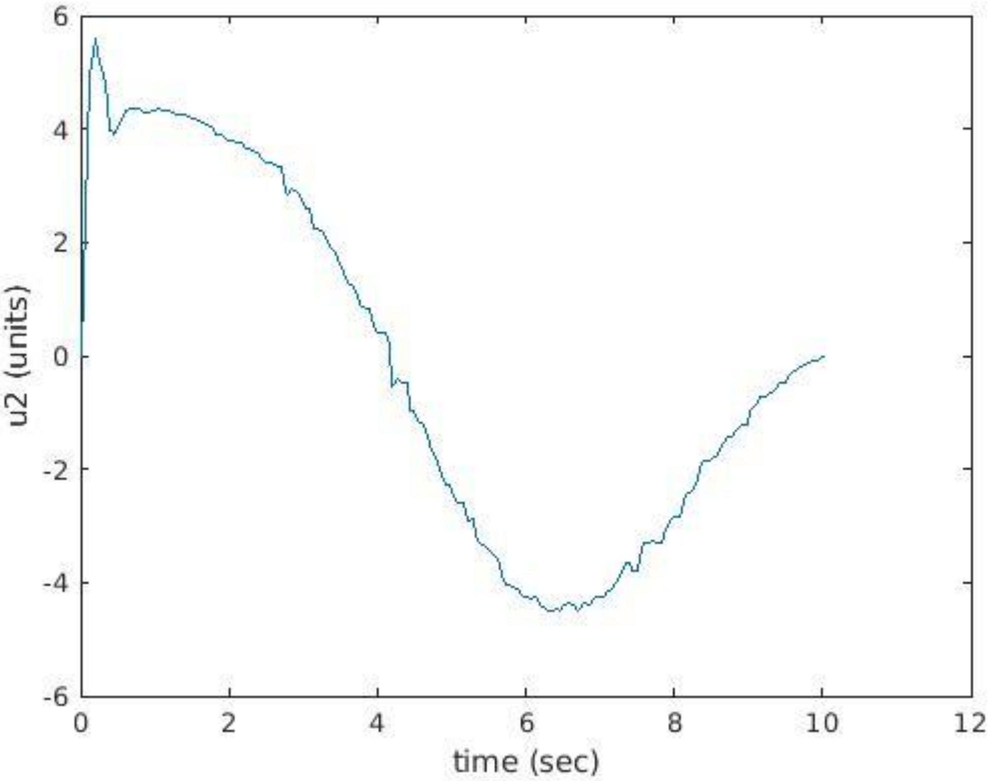



3.5
3
2.5
2
1.5
1
0.5
0
t2 (radian)
0
2
4
6
8
10
12
time (sec)


0.2
0
-0.2
-0.4
-0.6
-0.8
-1
theta1 (radian/sec)
0
2
4
6
8
10
12
time (sec)


theta2 (radian/sec)
time (sec)
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-1.4
0
2
4
6
8
10
12


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


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