RBE502 – Robotic Control

Programming Assignment -2

Submitted by:
Suketu Parekh

a) **(5 points) Using the symbolic equations of motion derived in Programming Assignment 1, determine the equilibrium points of the robot in MATLAB.**

t1_star =	t2_star =
0	0
pi	0
0	pi
pi	pi

b) **(5 points) Derive the Jacobian linearization of the symbolic state-space representation (derived in Programming Assignment 1) in MATLAB to obtain the symbolic representation of the state matrix A and the input matrix B. MATLAB function jacobian can be used for linearization. Substitute the physical parameters of the robot (listed below) in the state and input matrices.m1=m2= 1 (kg), l1=l2= 1 (m), r1=r2= 0.45 (m)I1=I2= 0.084 (kg·m2), g= 9.81 (m/s2)**

A =

```
[ 0, 0, 1, 0]
[ 0, 0, 0, 1]
[ -((16301277*cos(t1))/4000000 - (79461*cos(t1 + t2)*cos(t2))/40000)/((81*cos(t2)^2)/400 - 1474329/4000000), - ((79461*sin(t1 + t2)*sin(t2))/40000 + (5157*t1_dot^2*cos(t2))/40000 + (5157*t2_dot^2*cos(t2))/40000 + (81*t1_dot^2*cos(t2)^2)/400 - (81*t1_dot^2*sin(t2)^2)/400 + (9*u2*sin(t2))/20 - (79461*cos(t1 + t2)*cos(t2))/40000 + (5157*t1_dot*t2_dot*cos(t2))/20000)/((81*cos(t2)^2)/400 - 1474329/4000000) - (81*cos(t2)*sin(t2)*((573*u1)/2000 - (573*u2)/2000 + (16301277*sin(t1))/4000000 + (5157*t1_dot^2*sin(t2))/40000 + (5157*t2_dot^2*sin(t2))/40000 - (9*u2*cos(t2))/20 - (79461*sin(t1 + t2)*cos(t2))/40000 + (5157*t1_dot*t2_dot*sin(t2))/20000 + (81*t1_dot^2*cos(t2)*sin(t2))/400))/(200*((81*cos(t2)^2)/400 - 1474329/4000000)^2), -((5157*t1_dot*sin(t2))/20000 + (5157*t2_dot*sin(t2))/20000 + (81*t1_dot*cos(t2)*sin(t2))/200)/((81*cos(t2)^2)/400 - 1474329/4000000), -((5157*t1_dot*sin(t2))/20000 + (5157*t2_dot*sin(t2))/20000)/((81*cos(t2)^2)/400 - 1474329/4000000)]
[-((22717017*cos(t1 + t2))/4000000 - (16301277*cos(t1))/4000000 - (256041*cos(t1)*cos(t2))/40000 + (79461*cos(t1 + t2)*cos(t2))/40000)/((81*cos(t2)^2)/400 - 1474329/4000000), ((79461*sin(t1 + t2)*sin(t2))/40000 - (22717017*cos(t1 + t2))/4000000 + (14157*t1_dot^2*cos(t2))/20000 + (5157*t2_dot^2*cos(t2))/40000 - (256041*sin(t1)*sin(t2))/40000 + (81*t1_dot^2*cos(t2)^2)/200 + (81*t2_dot^2*cos(t2)^2)/400 - (81*t1_dot^2*sin(t2)^2)/200 - (81*t2_dot^2*sin(t2)^2)/400 - (9*u1*sin(t2))/20 + (9*u2*sin(t2))/10 - (79461*cos(t1 + t2)*cos(t2))/40000 + (5157*t1_dot*t2_dot*cos(t2))/20000 + (81*t1_dot*t2_dot*cos(t2)^2)/200 - (81*t1_dot*t2_dot*sin(t2)^2)/200)/((81*cos(t2)^2)/400 - 1474329/4000000) + (81*cos(t2)*sin(t2)*((573*u1)/2000 - (1573*u2)/1000 - (22717017*sin(t1 + t2))/4000000 + (16301277*sin(t1))/4000000 + (14157*t1_dot^2*sin(t2))/20000 +
```

(5157*t2_dot^2*sin(t2))/40000 + (256041*cos(t2)*sin(t1))/40000 + (9*u1*cos(t2))/20 - (9*u2*cos(t2))/10 - (79461*sin(t1 + t2)*cos(t2))/40000 + (5157*t1_dot*t2_dot*sin(t2))/20000 + (81*t1_dot^2*cos(t2)*sin(t2))/200 + (81*t2_dot^2*cos(t2)*sin(t2))/400 + (81*t1_dot*t2_dot*cos(t2)*sin(t2))/200))/(200*((81*cos(t2)^2)/400 - 1474329/4000000)^2), ((14157*t1_dot*sin(t2))/10000 + (5157*t2_dot*sin(t2))/20000 + (81*t1_dot*cos(t2)*sin(t2))/100 + (81*t2_dot*cos(t2)*sin(t2))/200)/((81*cos(t2)^2)/400 - 1474329/4000000), ((5157*t1_dot*sin(t2))/20000 + (5157*t2_dot*sin(t2))/20000 + (81*t1_dot*cos(t2)*sin(t2))/200 + (81*t2_dot*cos(t2)*sin(t2))/200)/((81*cos(t2)^2)/400 - 1474329/4000000)]

B =

[0, 0]
[0, 0]
[-573/(2000*((81*cos(t2)^2)/400 - 1474329/4000000)), ((9*cos(t2))/20 + 573/2000)/((81*cos(t2)^2)/400 - 1474329/4000000)]
[((9*cos(t2))/20 + 573/2000)/((81*cos(t2)^2)/400 - 1474329/4000000), -((9*cos(t2))/10 + 1573/1000)/((81*cos(t2)^2)/400 - 1474329/4000000)]

c) **(4 points) Investigate the stability properties of the linearized system around each of the equilibrium points found in Step (a).**

Eigenvalues for equilibrium point: 0,0 are:

EV1 =

2.7129

7.1676

-2.7129

-7.1676

System is unstable at this equilibrium point

Eigenvalues for equilibrium point: pi,0 are:

EV2 =

0.0000 + 7.1676i

0.0000 + 2.7129i

0.0000 - 7.1676i

0.0000 - 2.7129i

System is marginally stable at this equilibrium point

Eigenvalues for equilibrium point: 0,pi are:

EV3 =

2.7129

7.1676

-2.7129

-7.1676

System is unstable at this equilibrium point

Eigenvalues for equilibrium point: pi,pi are:

EV4 =

2.7129

7.1676

-2.7129

-7.1676

System is unstable at this equilibrium point

d) (1 point) Investigate the controllability of the linearized system around the equilibrium point corresponding to the "upward" configuration.

Rank =

4

Since Controllability matrix is full rank, system is Controllable at 0,0

e) **(5 points) Design a state-feedback control to stabilize the system around the upward equilibrium point. Feel free to use MATLAB function place.**
State-Feedback control is given by u = -kx. The value of k is as follows:

Eigen values taken = [-1;-5;-3-j;-3+j]

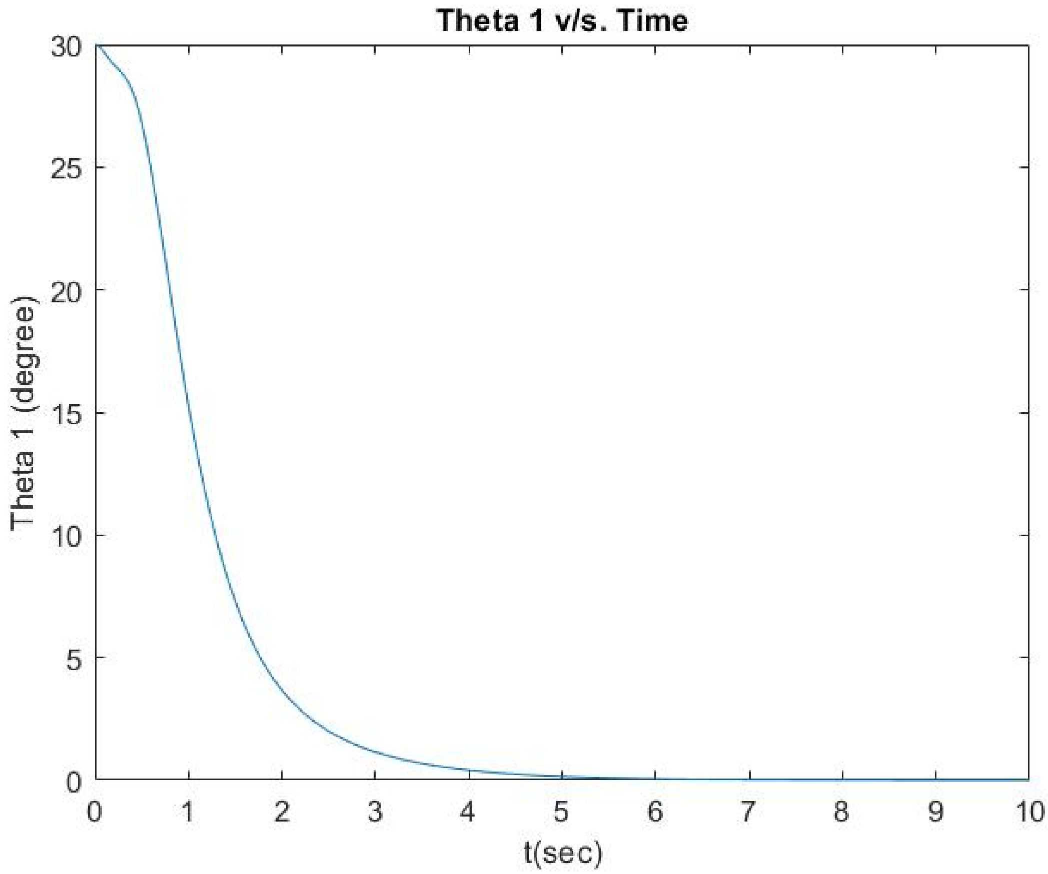
Kn =

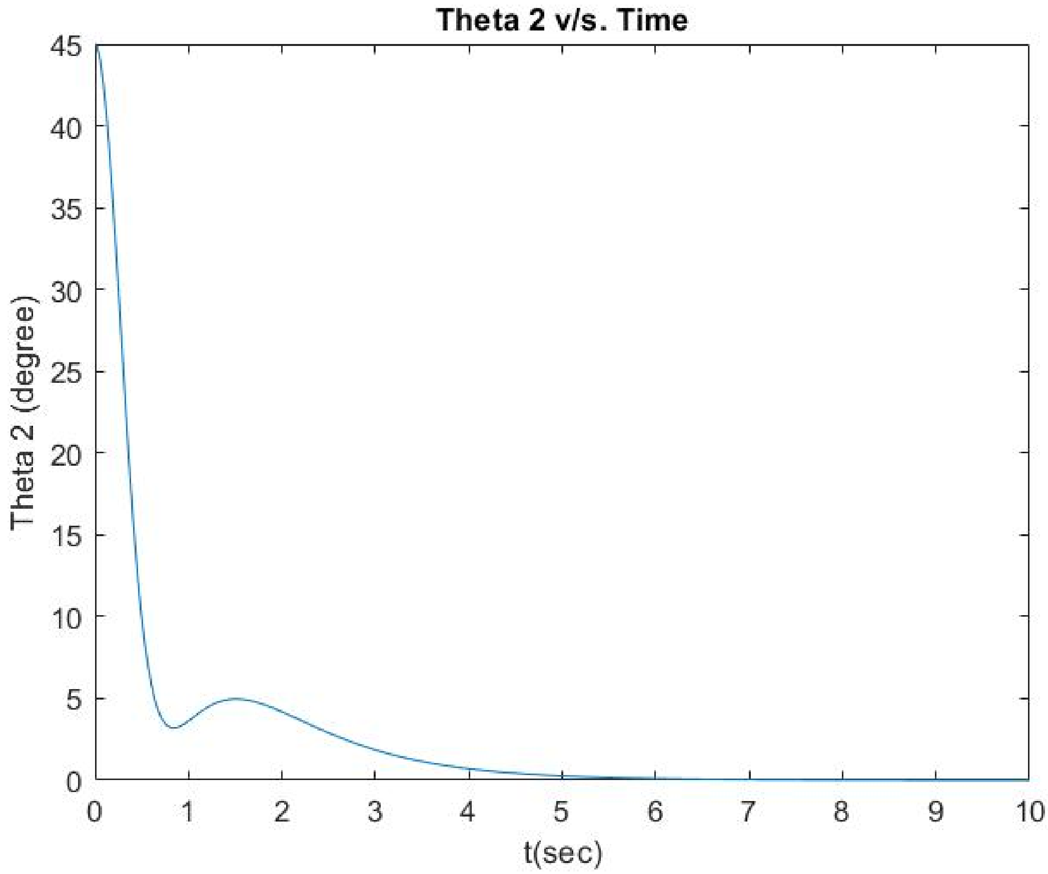
```
   33.8023    6.4772   14.8380    4.4190
    8.6340    5.5361    4.4190    1.7190
```

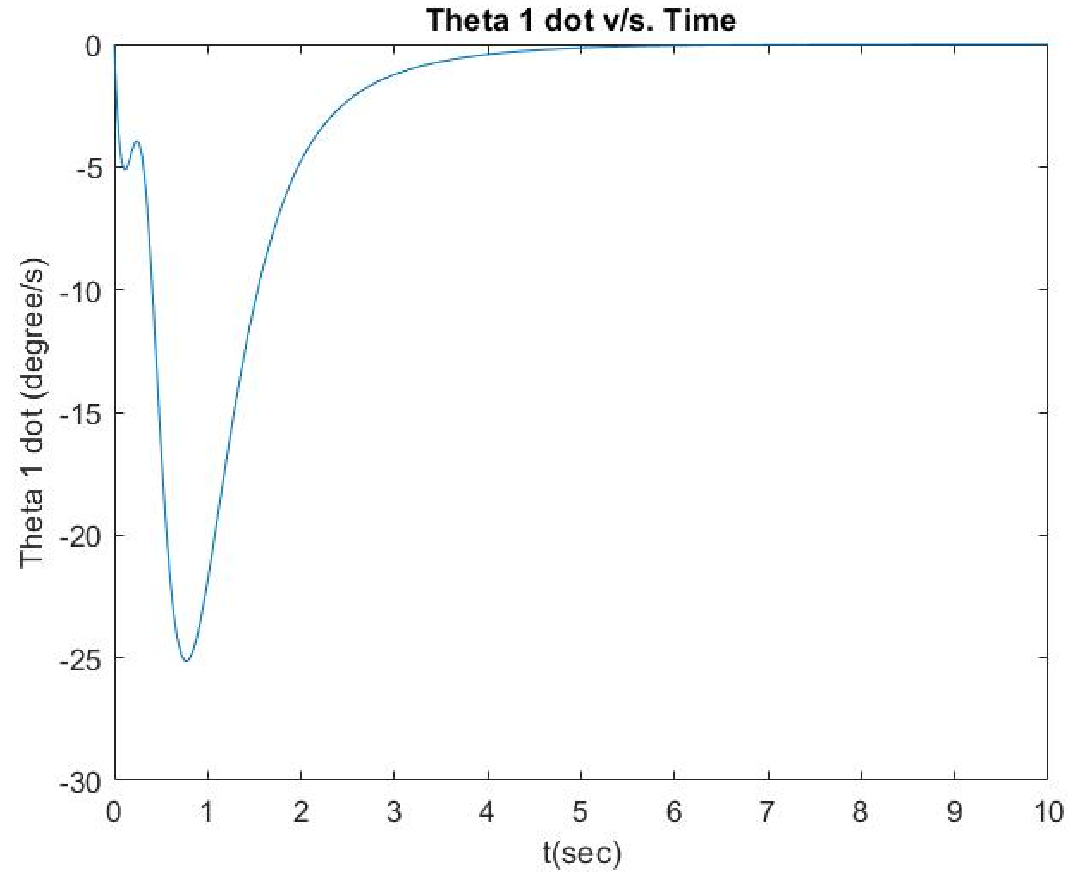
u =

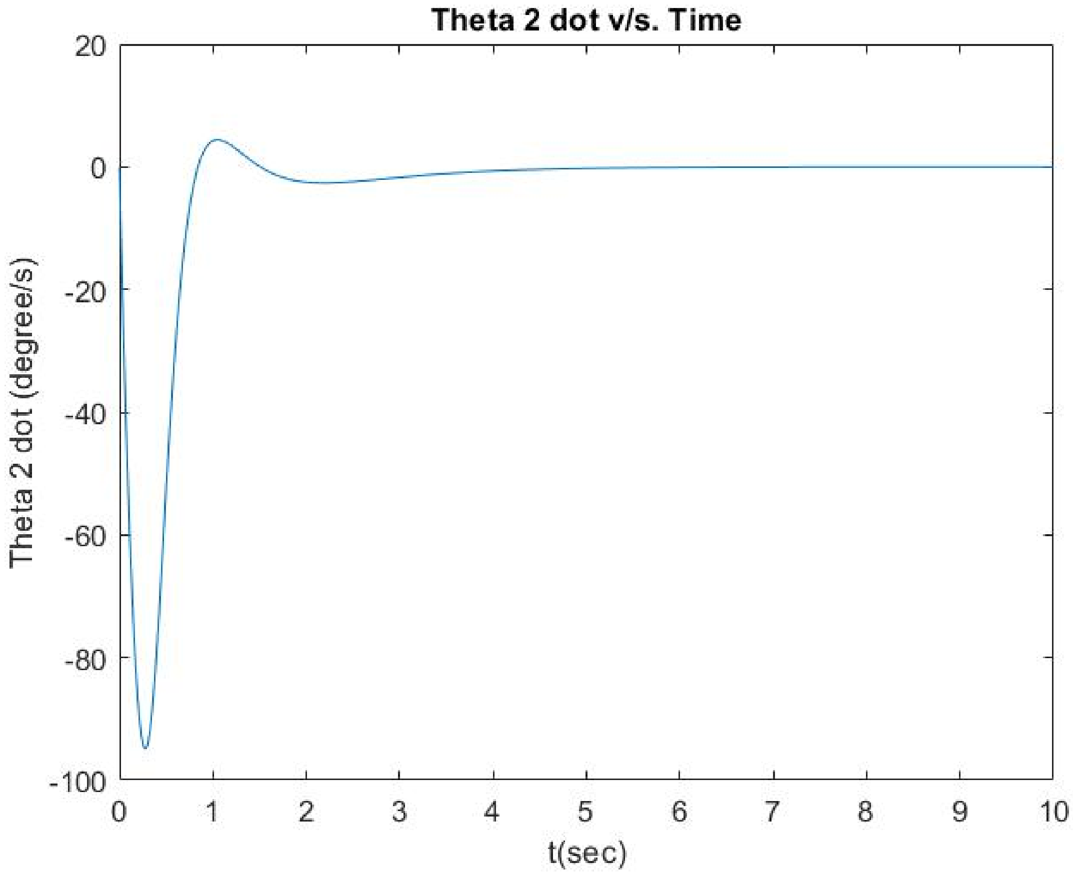
- (1189312871105339*t1)/35184372088832 - (7292630729812249*t2)/1125899906842624 - (7419*t1_dot)/500 - (4419*t2_dot)/1000
- (2430265662578031*t1)/281474976710656 - (6233124955178157*t2)/1125899906842624 - (4419*t1_dot)/1000 - (1719*t2_dot)/1000

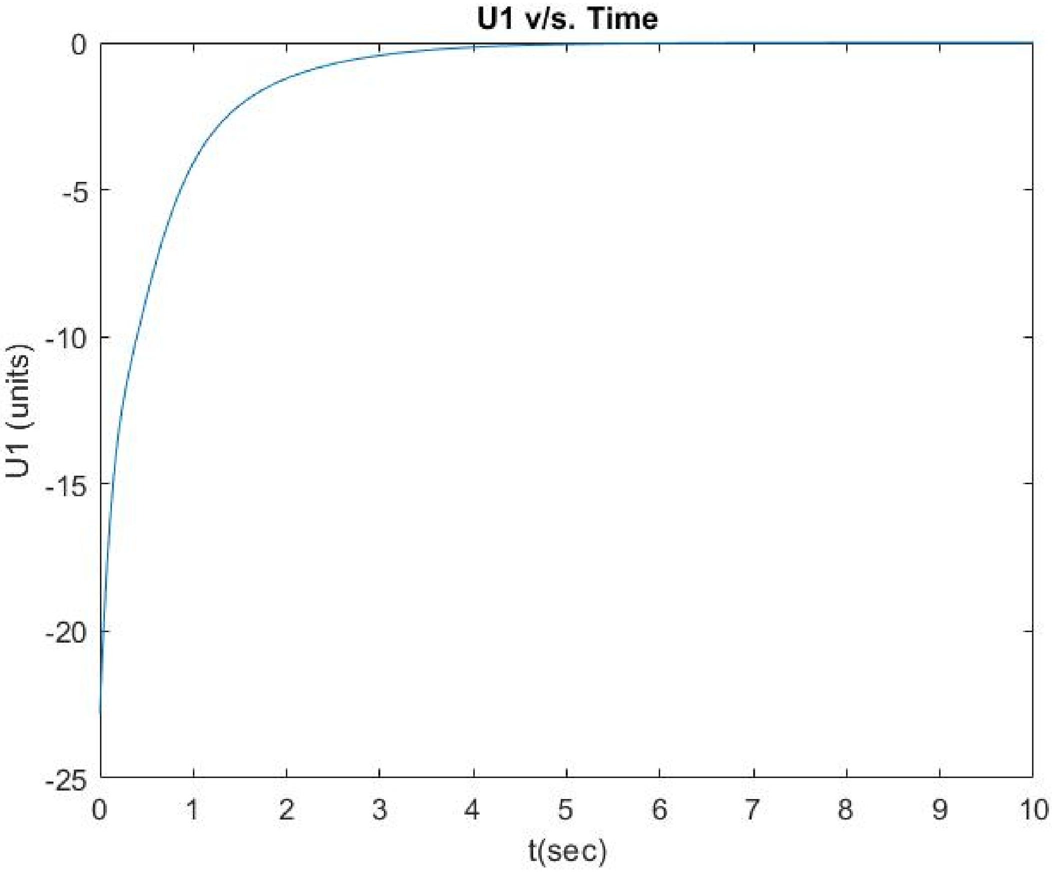
f) **(15 points) Use ode45 and the ode function developed in Programming Assignment 1 to construct a simulation of the system in MATLAB with the initial conditions $\theta_1(0) = 30^\circ$, $\theta_2(0) = 45^\circ$, $\dot\theta_1(0) = 0$ and $\dot\theta_2(0) = 0$ and time-span of 10 seconds. Do not forget to implement your feedback control law (designed in Step (e)) inside the ode function. Plot the state trajectories (of the state space form) and the control inputs trajectories to evaluate the performance.**

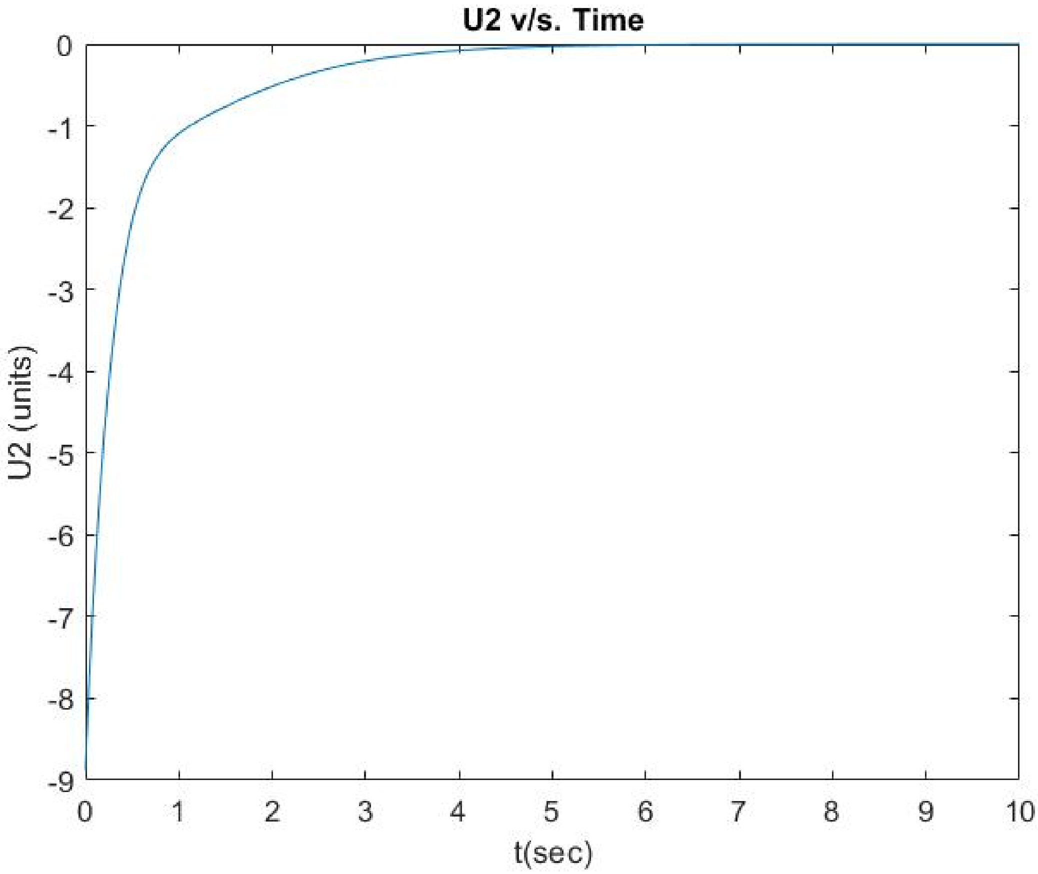

Theta 1 v/s. Time
Theta 1 (degree)
t(sec)
0
5
10
15
20
25
30
0
1
2
3
4
5
6
7
8
9
10


Theta 2 v/s. Time
Theta 2 (degree)
t(sec)
0
5
10
15
20
25
30
35
40
45
0
1
2
3
4
5
6
7
8
9
10


Theta 1 dot v/s. Time
Theta 1 dot (degree/s)
t(sec)
0
-5
-10
-15
-20
-25
-30
0
1
2
3
4
5
6
7
8
9
10

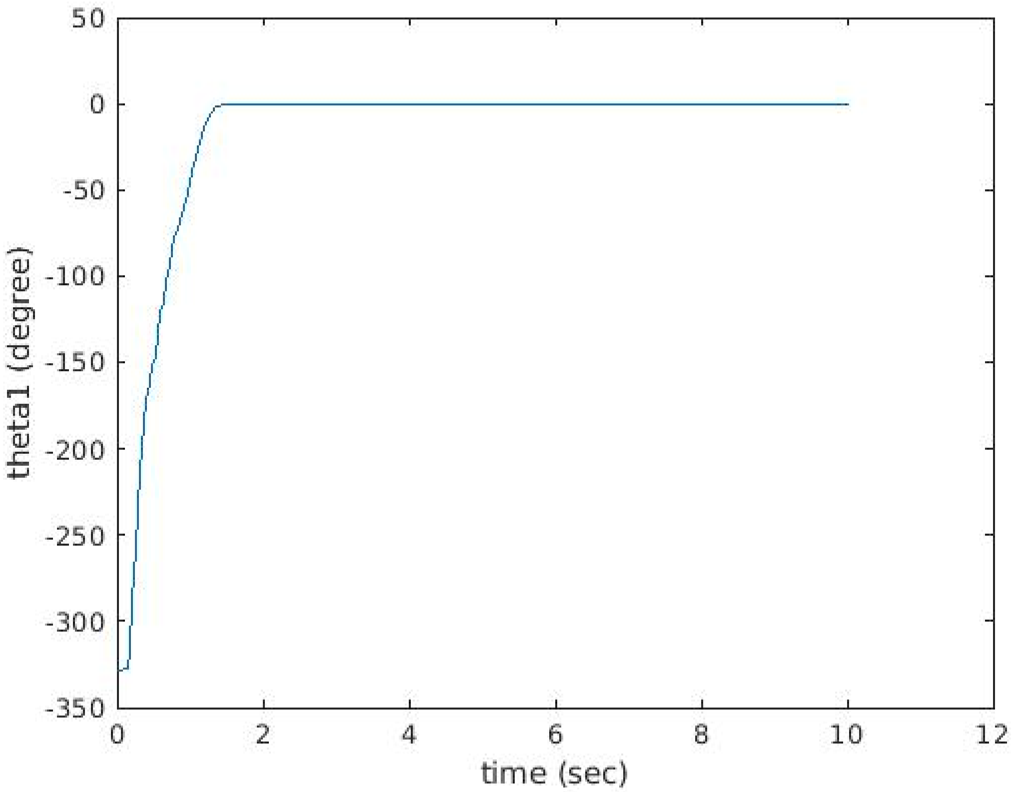

Theta 2 dot v/s. Time
Theta 2 dot (degree/s)
t(sec)
20
0
-20
-40
-60
-80
-100
0
1
2
3
4
5
6
7
8
9
10


U1 v/s. Time
U1 (units)
t(sec)
0
-5
-10
-15
-20
-25
0 1 2 3 4 5 6 7 8 9 10

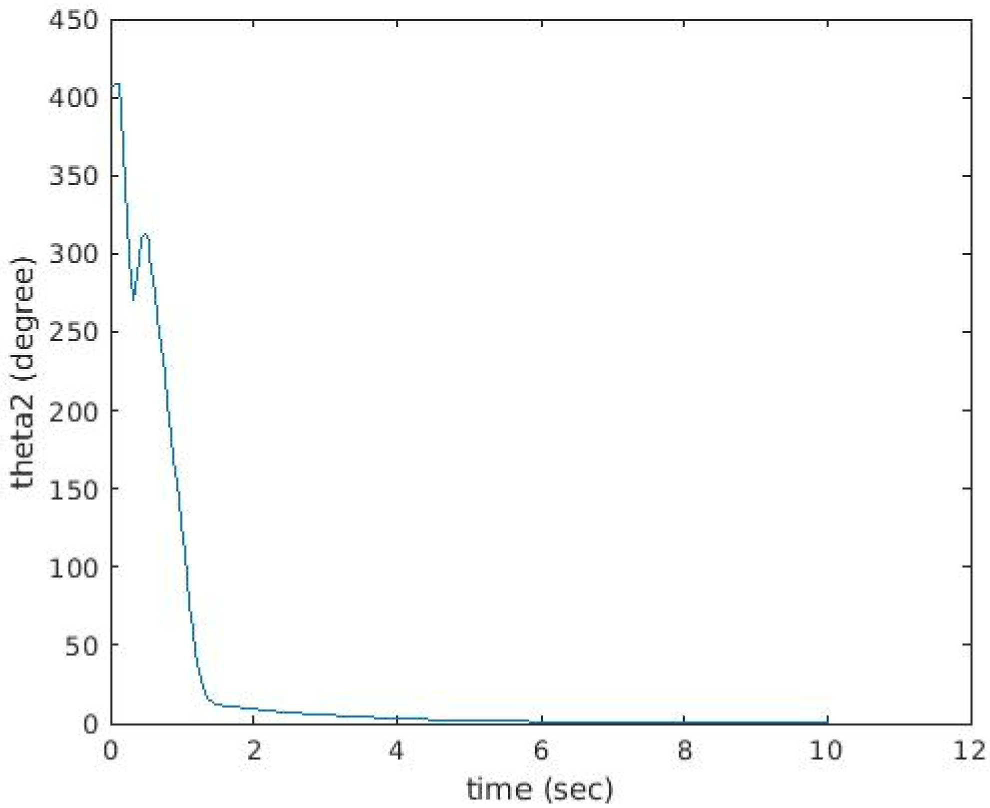

U2 v/s. Time
U2 (units)
t(sec)
0
-1
-2
-3
-4
-5
-6
-7
-8
-9
0 1 2 3 4 5 6 7 8 9 10

g) (15 points) Copy and paste the following code into a new MATLAB script named rrbot_control.m. Complete the code inside the while loop to control the RRbot robot inGazebo for 10 seconds using your state-feedback controller designed in Step (f). Sample thedata at each loop to be plotted at the end. Feel free to define new functions and variablesin your program if needed. Compare the resulting trajectories and control inputs in Gazebo with those obtained in Step (f) in MATLAB. Discuss your findings in your final report.

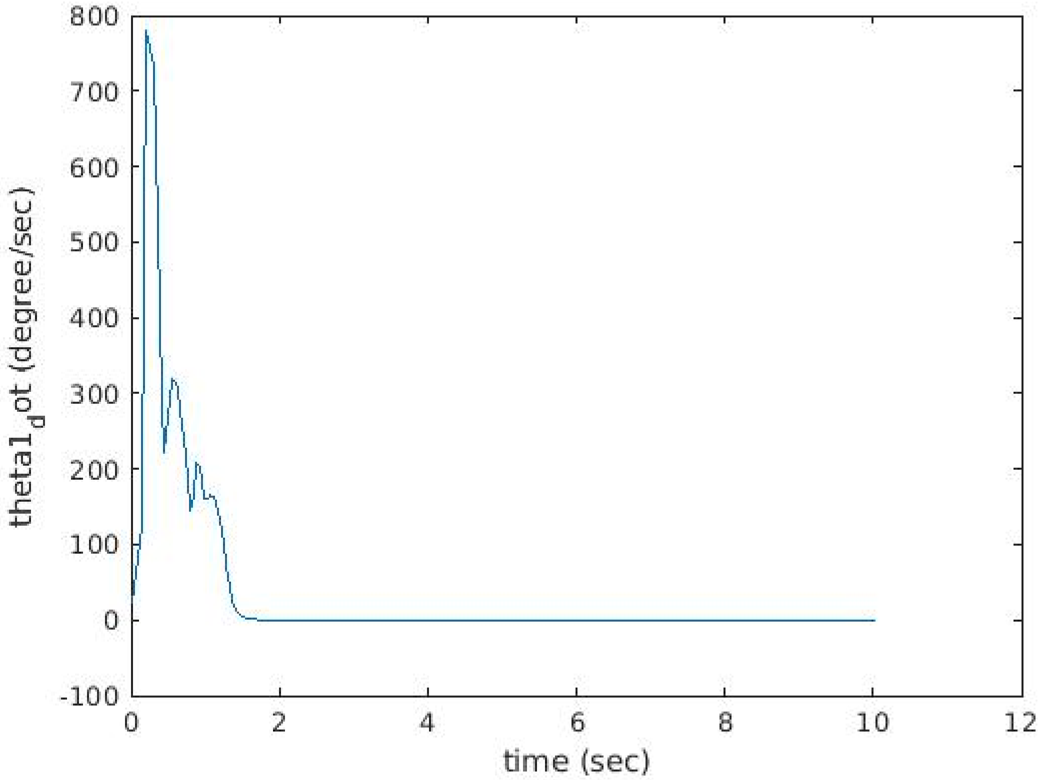
The plots of Theta v/s. time for both the MATLAB and Gazebo seems to be in good enough closeness.
The plots of Theta_dot v/s. time for MATLAB seems to be a lot smoother than that of Gazebo. This can be due to the fact the Gazebo simulates the system and in reality, there might be some differences in the actuation of the actuators than the ideal value.
The plots for the Control inputs v/s. time for MATLAB is also smoother than that of Gazebo. This is due to the compensation done by the simulator to account for the undue variations of the angular velocities.
The Gazebo plots are as follows:



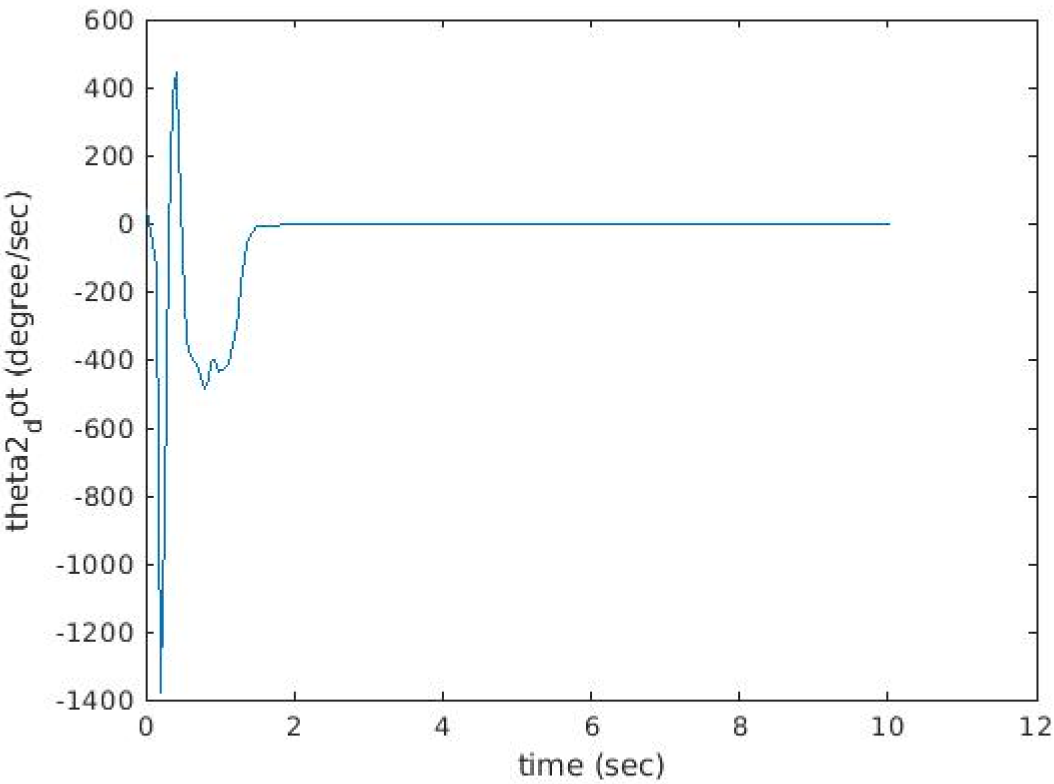
50
0
-50
-100
-150
-200
-250
-300
-350
theta1 (degree)
0
2
4
6
8
10
12
time (sec)



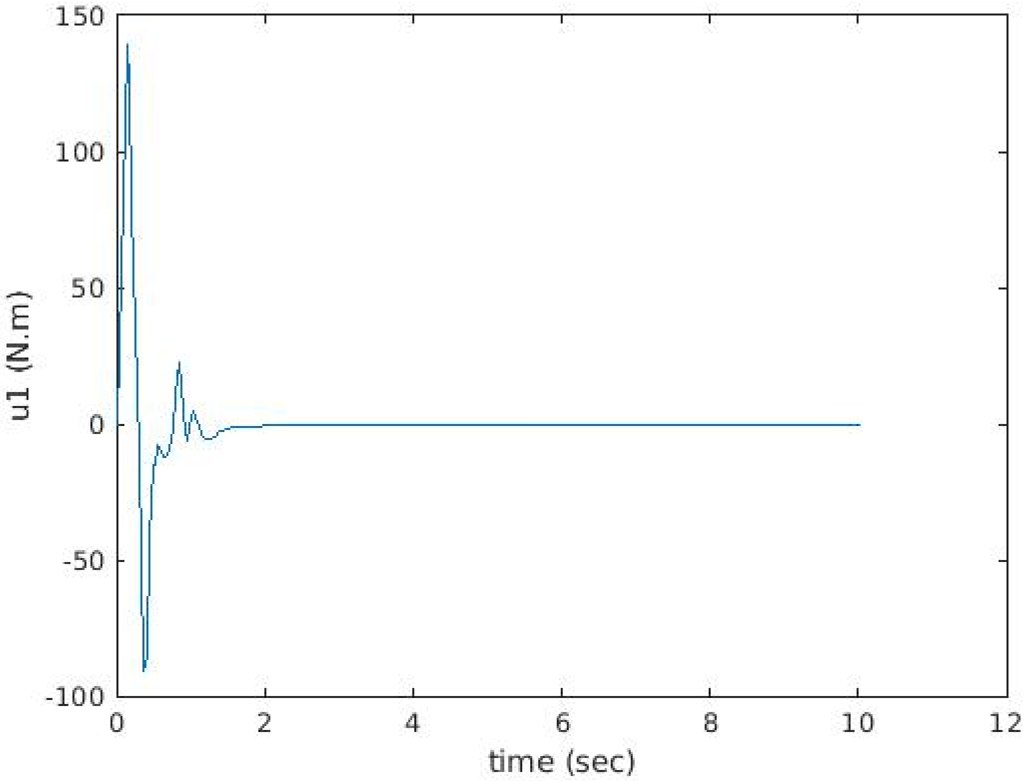
450
400
350
300
250
200
150
100
50
0
theta2 (degree)
0
2
4
6
8
10
12
time (sec)

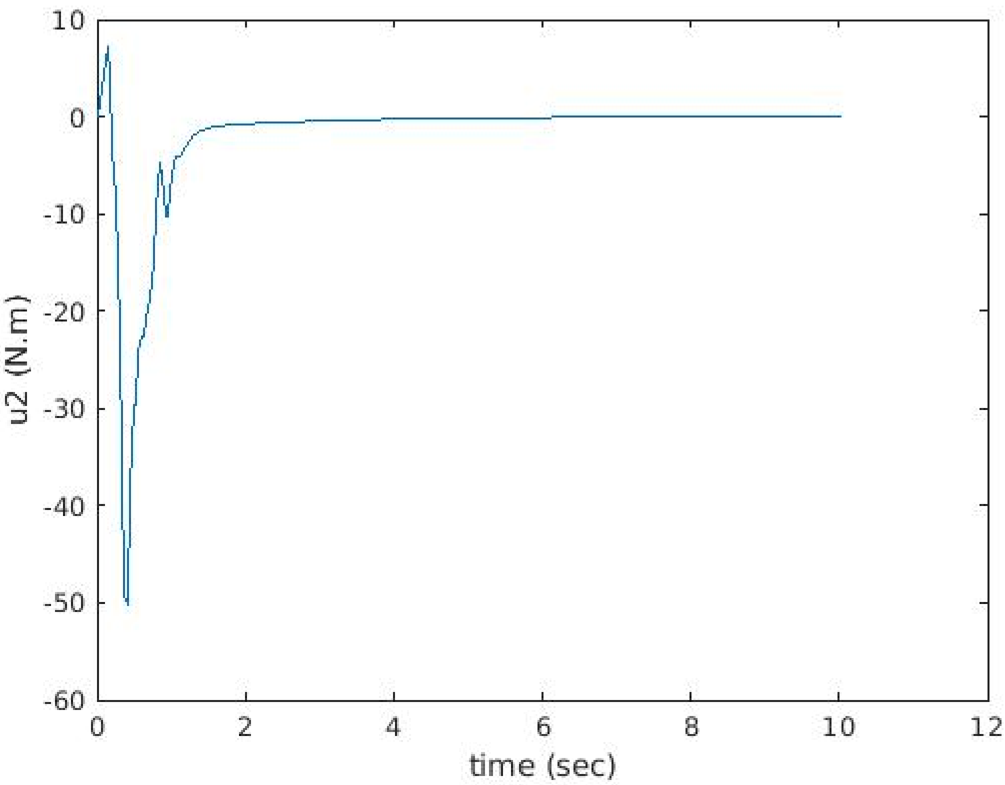


800
700
600
500
400
300
200
100
0
-100
0
2
4
6
8
10
12
theta1_dot (degree/sec)
time (sec)



600
400
200
0
-200
-400
-600
-800
-1000
-1200
-1400
0
2
4
6
8
10
12
theta2_dot (degree/sec)
time (sec)


150
100
50
0
-50
-100
u1 (N.m)
0
2
4
6
8
10
12
time (sec)


10
0
-10
-20
-30
-40
-50
-60
u2 (N.m)
0
2
4
6
8
10
12
time (sec)