Programming Assignment 3

Submitted by: Prasham Patel

<u>How to run:</u> Store all the files in a single folder and run main to run ODE45 of MATLAB. For Gazebo simulation run rrbot_control.

<u>Cubic polynomials trajectory generator</u>

The trajectory_generator function takes in initial and final state of one the variable of the system as a single matrix and start and end time of the trajectory. The equation of motion for states in the "step a" are show below:

th1_des = (pi*t∧3)/500 - (3*pi*t∧2)/100 + pi;

th1dot_des = (3*pi*t∧2)/500 - (3*pi*t)/50;

th2_des = (pi*t∧3)/1000 - (3*pi*t∧2)/200 + pi/2;

th2dot_des = (3*pi*t∧2)/1000 - (3*pi*t)/100;

<u>Standard manipulator form</u>

Below mentioned is the standard manipulator form the dynamic equation of the 2-link robot:

T = m*[theta_ddot1; theta_ddot2] + c*[theta_dot1; theta_dot2] + G

T = [theta_ddot1*(m1*r1∧2 + I1 + I2 + (m2*(2*l1∧2 + 4*cos(theta2)*l1*r2 + 2*r2∧2))/2) + theta_ddot2*(I2 + (m2*(2*r2∧2 + 2*l1*cos(theta2)*r2))/2) - g*m2*(r2*sin(theta1 + theta2) + l1*sin(theta1)) - g*m1*r1*sin(theta1) + l1*r2*theta_dot1*sin(theta2) - l1*m2*r2*theta_dot2*sin(theta2)*(theta_dot1 + theta_dot2)

;

theta_ddot2*(m2*r2∧2 + I2) + theta_ddot1*(I2 + (m2*(2*r2∧2 + 2*l1*cos(theta2)*r2))/2) - g*m2*r2*sin(theta1 + theta2) + l1*m2*r2*theta_dot1*sin(theta2)*(theta_dot1 + theta_dot2) - l1*m2*r2*theta_dot1*theta_dot2*sin(theta2)]

Feedback Linearization

Virtual inputs are calculated as shown below:

```
v1 = -kp1*(theta1 - q1_t(1)) -kd1*(theta_dot1 - q1_t(2)) + vd(1)

v2 = -kp2*(theta2 - q2_t(1)) -kd2*(theta_dot2 - q2_t(2)) + vd(2)
```

Then v1 and v2 are substituted into the original equation of motion calculated in assignment1 to get T1 and T2:

```
T1 = subs(u1, {theta_ddot1, theta_ddot2}, {v1, v2})

T2 = subs(u2, {theta_ddot1, theta_ddot2}, {v1, v2})
```

After obtaining T1 and T2 we form new equation as shown below, by subtracting T1 and T2 from u1 and u2 respectively. And solve those equation for theta double dot, which are used in the ODE function.

```
eq1 = u1-T1;

eq2 = u2-T2;

sol = solve([eq1==0, eq2==0], [theta_ddot1, theta_ddot2]);

th_ddot1 = sol.theta_ddot1 ;

th_ddot2 = sol.theta_ddot2 ;
```

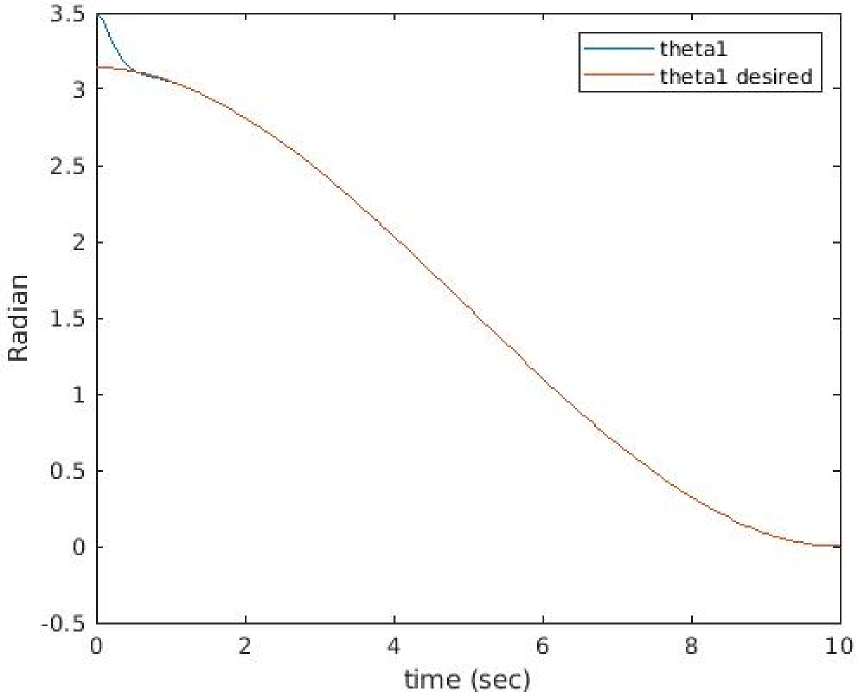
ODE function

The only change in ODE function is the replacement of the th1_dd and th2_dd by the values that we calculated above:

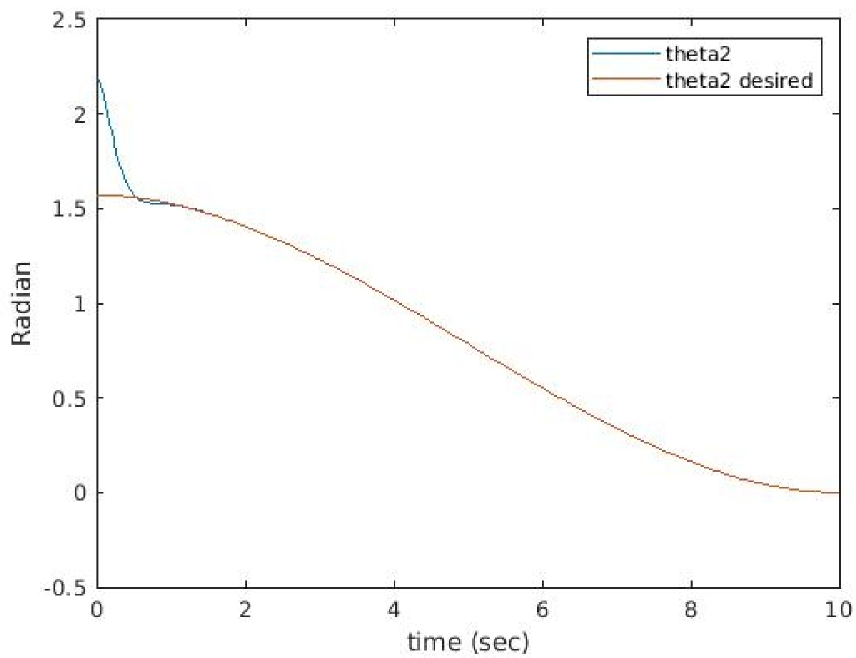
```
theta_ddot1 = (10*pi*kp1)/9 - pi/15 + (pi*t)/75 - kd1*theta_dot1 - kp1*theta1 - (pi*kd1*t)/15 +
              (pi*kd1*t^2)/150 - (pi*kp1*t^2)/30 + (pi*kp1*t^3)/450;

theta_ddot2 = (25*pi*kp2)/36 - pi/24 + (pi*t)/120 - kd2*theta_dot2 - kp2*theta2 - (pi*kd2*t)/24
              + (pi*kd2*t^2)/240 - (pi*kp2*t^2)/48 + (pi*kp2*t^3)/720;
```

ODE function output

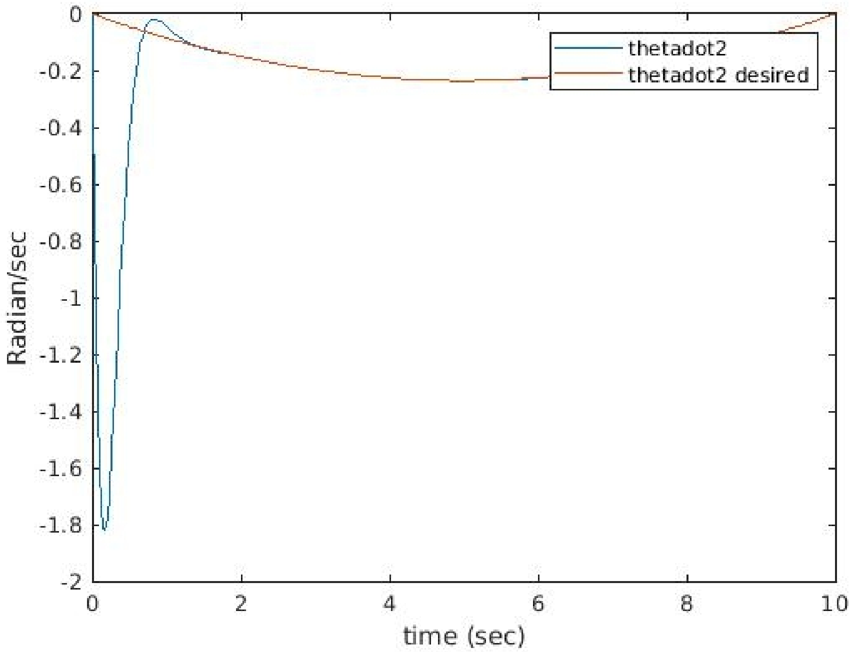
Theta1:



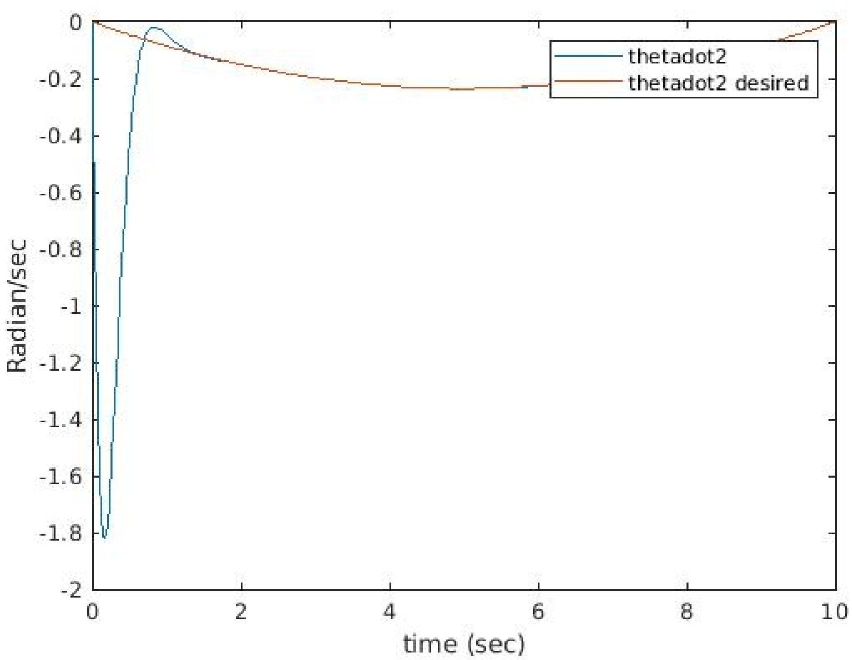
Theta2:



Theta1_dot:

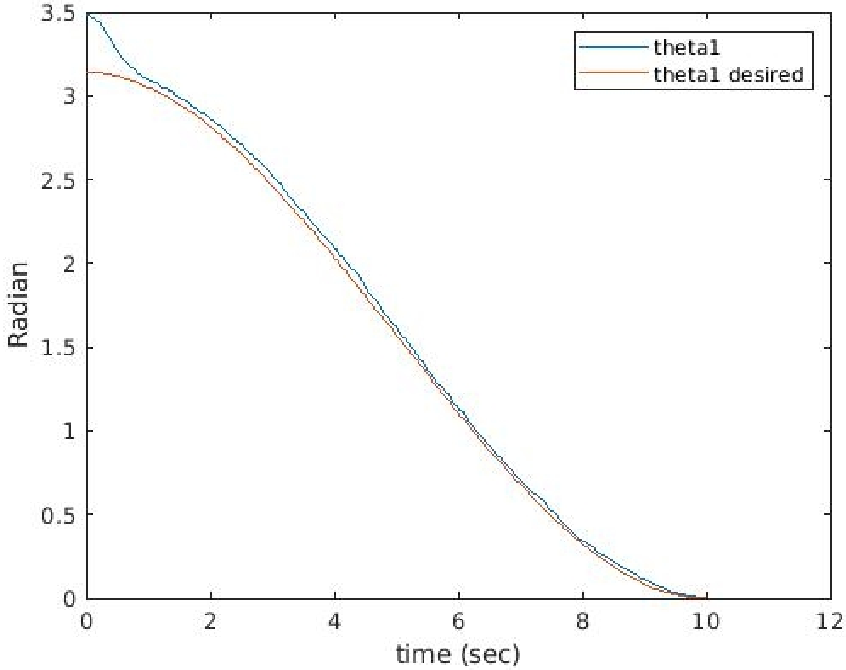


Theta2_dot:

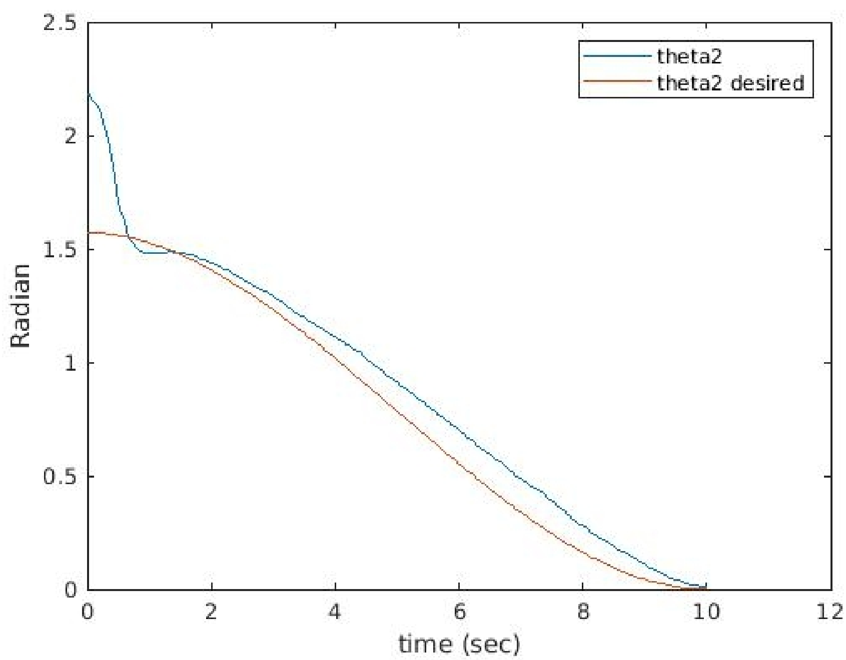


Gazebo Simulation

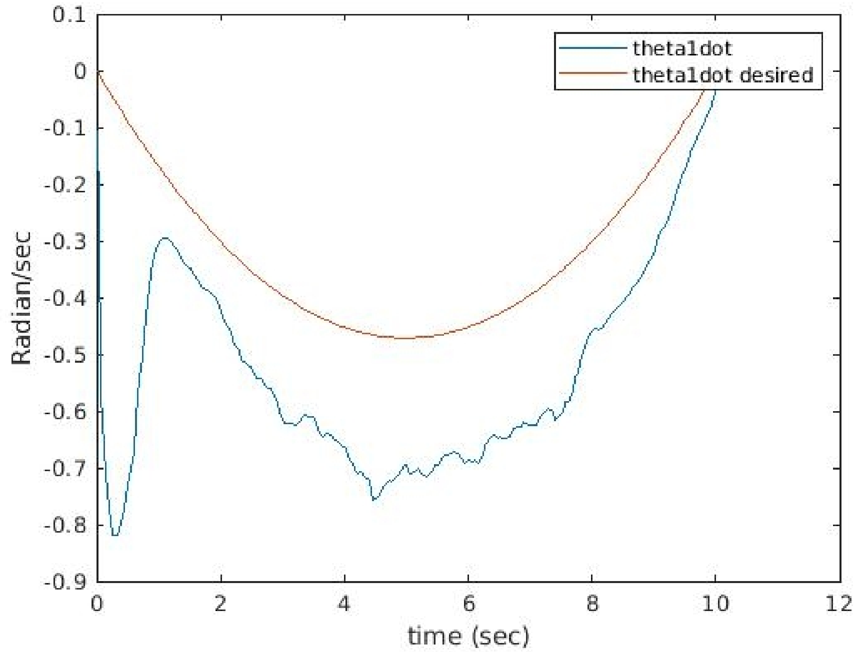
Theta1:



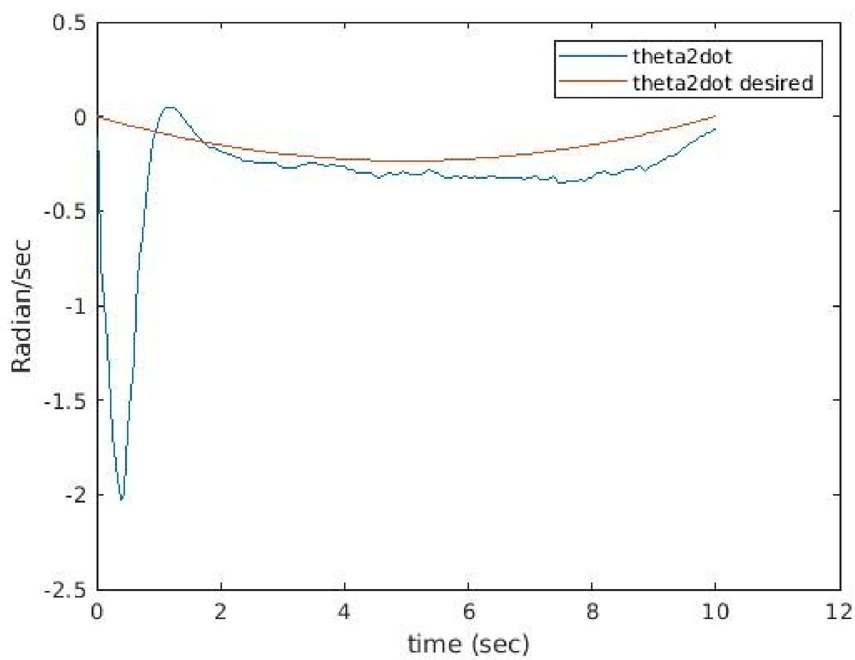
Theta2:



Theta1_dot:



Theta2_dot:



Torque input: