Programming Assignment-2

Submitted by: Prasham Patel
Student Id: 871563809

I. Equilibrium points

The commands for finding equilibrium points are in equilibrium_points.m. Total number of equilibrium points for revolute joint planar robots are 2^n, here n is the number of joints. Thus, we know there will be 4 equilibrium points for the two-link planar robot. And to find the same 'vpasolve' is used instead of 'solve'. vpasolve is called multiple times to find all the possible values of equilibrium.

```
while i < 5

    flag = 0;

    sol_th= vpasolve([eq_th_ddot1==0, eq_th_ddot2==0], [theta1, theta2], [-1 3.5; -1
3.5], "Random", true );

    for n = 1:i

        if ([round(sol_th.theta1, 4), round(sol_th.theta2, 4)] == [eqi_th(n, 1),
eqi_th(n, 2)])

            flag = 1;

        end

    end

    if flag == 0

    eqi_th(i, 1) = round(sol_th.theta1, 4);

    eqi_th(i, 2) = round(sol_th.theta2, 4);

    i = i+1

    end

end
```

While loop is used to check if the same value of equilibrium point is observed before, if no, add the current value to the list or else continue.

II. Linearization, stability and controllability

Jacobian function is used to linearize A and B matrix. After linearization value substitution is done. The values of variable theta1, theta2 and so on are imported from the script generated for the first assignment.

```
patel_progassignment1_matlab;
x = [theta1; theta2; theta_dot1; theta_dot2];
xd = [theta_dot1; theta_dot2; theta_ddot1; theta_ddot2]
A = jacobian(xd, x);
A = subs(A, {l1 l2 r1 r2 m1 m2 I1 I2 g}, {1, 1, 0.45, 0.45, 1, 1, 0.084, 0.084, 9.81})
```

```
% Linearization of B

B = jacobian(xd, T)
B = subs(B, {l1 l2 r1 r2 m1 m2 I1 I2 g}, {1, 1, 0.45, 0.45, 1, 1, 0.084, 0.084, 9.81})
```

Eigen values are found for A matrix obtained by substituting different equilibrium points. Below mentioned are the eigen values.

```
% Finding Eigen Values to check Stability

A1 = subs(A, {theta1, theta2, theta_dot1, theta_dot2}, {0, 0, 0, 0});
A1 = double(A1);
B1 = subs(B, {theta1, theta2, theta_dot1, theta_dot2}, {0, 0, 0, 0});
B1 = double(B1);
e1 = eig(A1);

A2 = subs(A, {theta1, theta2, theta_dot1, theta_dot2}, {pi, pi, 0, 0});
A2 = double(A2);
B2 = subs(B, {theta1, theta2, theta_dot1, theta_dot2}, {0, 0, 0, 0});
B2 = double(B2);
e2 = eig(A2);

A3 = subs(A, {theta1, theta2, theta_dot1, theta_dot2}, {pi, 0, 0, 0});
A3 = double(A3);
B3 = subs(B, {theta1, theta2, theta_dot1, theta_dot2}, {0, 0, 0, 0});
B3 = double(B3);
e3 = eig(A3);

A4 = subs(A, {theta1, theta2, theta_dot1, theta_dot2}, {0, pi, 0, 0});
A4 = double(A4);
B4 = subs(B, {theta1, theta2, theta_dot1, theta_dot2}, {0, 0, 0, 0});
B4 = double(B4);
e4 = eig(A4);

e1 = [7.16755185370863; 2.71285643835756; -7.16755185370864; -2.71285643835756]
e2 = [-4.9863 + 0.0000i; 0.0000 + 3.89953872825144i; 0.0000- 3.8995i; 4.9863 + 0.0000i]
e3 = [0.0000+ 7.16755; 0.0000- 7.1675i; 0.0000 + 2.7128i; 0.0000 - 2.7128i]
e4 = [-3.8995 + 0.0000i; 3.8995 + 0.0000i; 0.0000 + 4.9863i; 0.0000 - 4.9863i]
```

All positions accept the downward position(e3) have positive real part of the eigen values thus they are unstable. However, for e3 it has real part 0 for all eigen values. Thus, it is marginally stable.

Controllability for the upward position equilibrium point is calculated as shown below. As the rank of the matrix is 4 the system is controllable.

```
% Controllability for upward position

Co = ctrb(A1, B1);
rank(Co)
```

III. State Feedback control

'p' is the eigen values taken to form the 'k' matrix.

```
p = [-4 -6 -3+2i -3-2i];
k = place(A, B, p);
```

$\dot{x}$ is calculated as shown below in the ODE function. The rest of the function is the same as in programming assignment 1.

```
xd = (A-(B*k))*x;
```

IV. Gazebo implementation

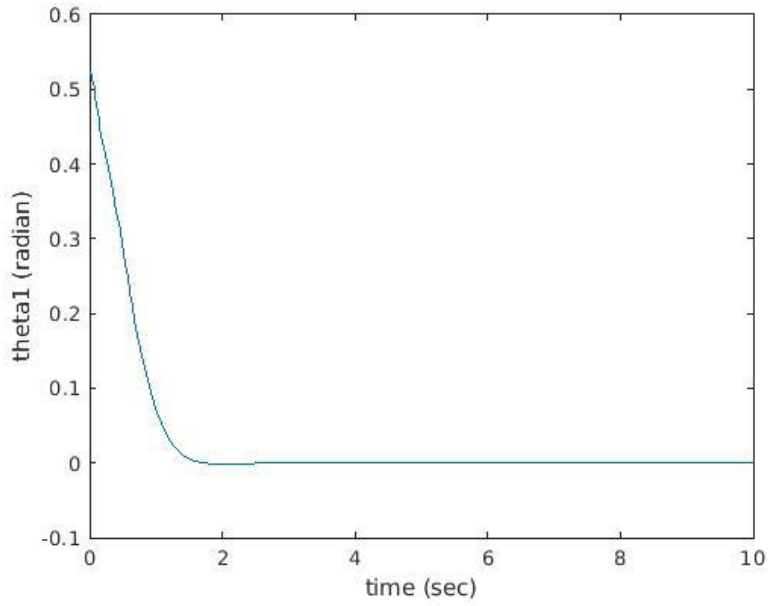
Defined 'U' matrix to store the value of 'u' for each iteration, similarly, 'Y' is defined to store the value of states to further plot them.

```
u = -k*[th1; th2; th_dot1; th_dot2];
tau1.Data = u(1);
tau2.Data = u(2);
U(length(U)+1, 1) = u(1)
U(length(U), 2) = u(2)
Y(length(Y)+1, :) = [th1, th2, th_dot1, th_dot2];
time(length(time)+1) = t
```

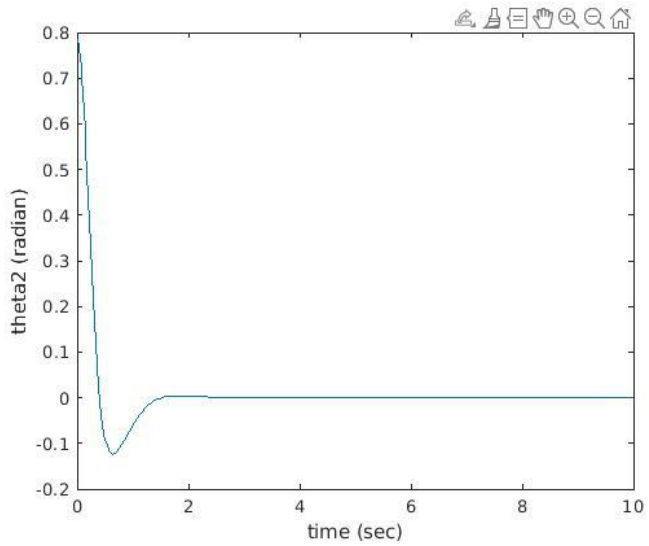
V. Plots

a. MATLAB ODE45 plots

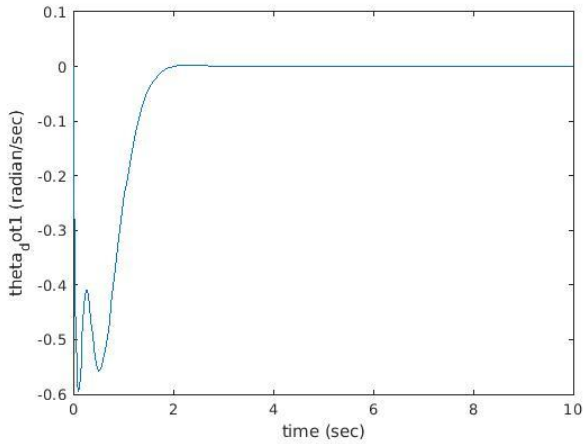
1. theta1 vs time



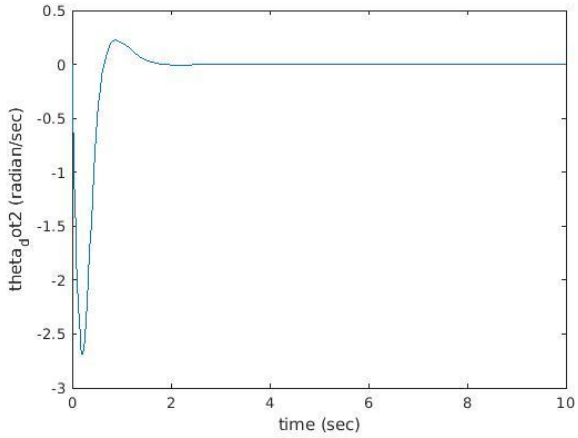
2. theta2 vs time



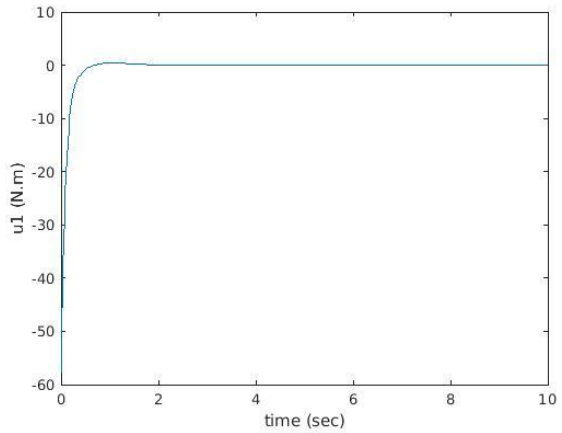
3. theta_dot1 vs time



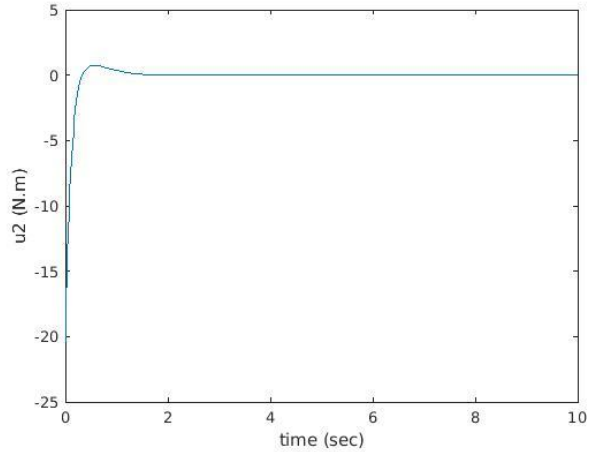
4. theta_dot2 vs time



5. U1 vs time

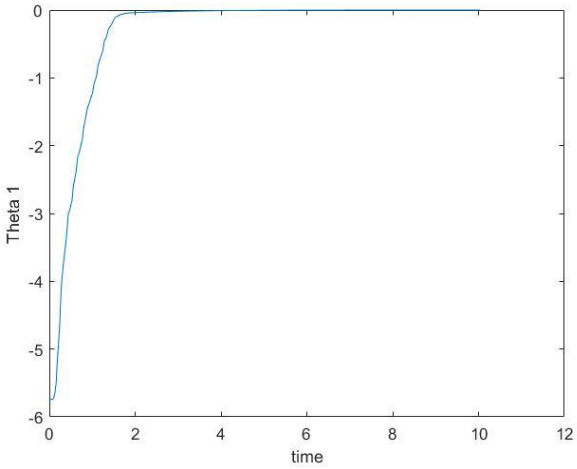


6. U2 vs time

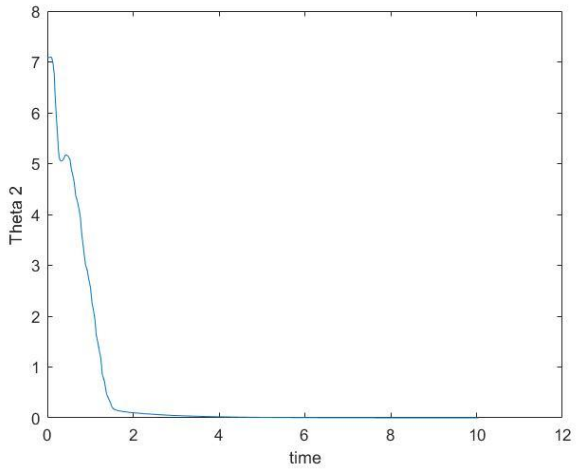


b. <u>Gazebo plots</u>

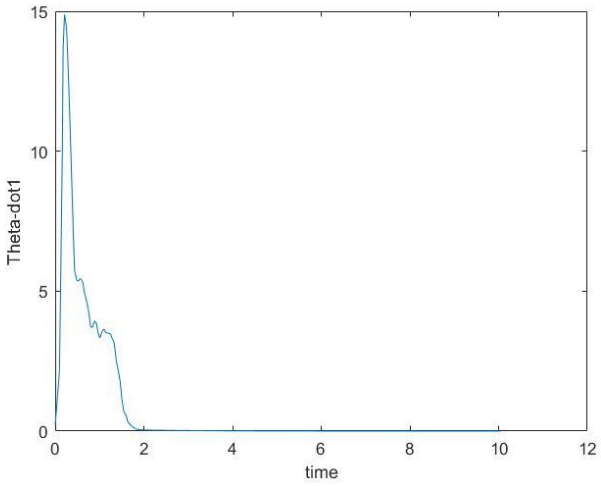
1. theta1 vs time



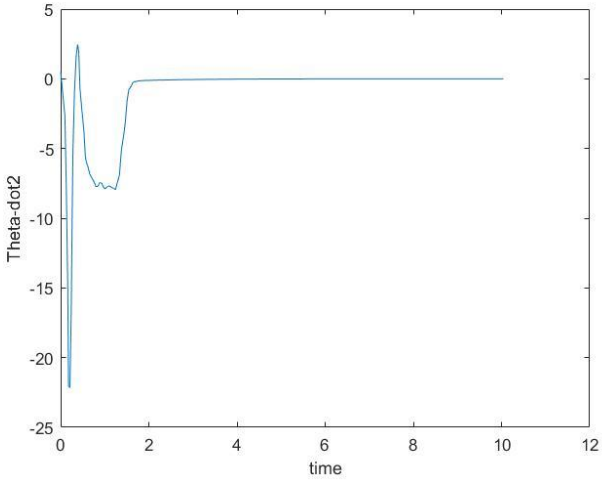
2. theta2 vs time



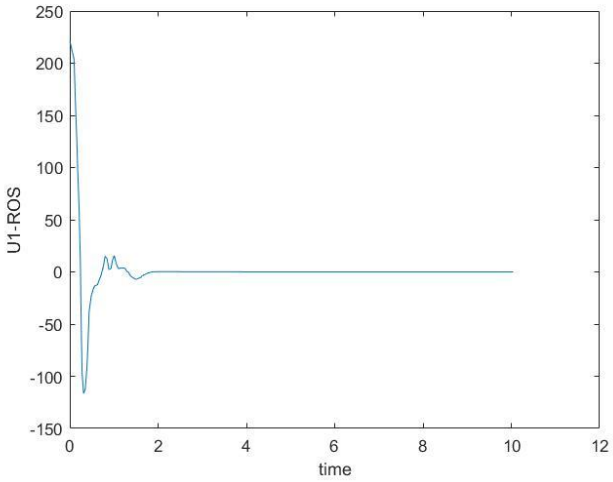
3. theta_dot1 vs time



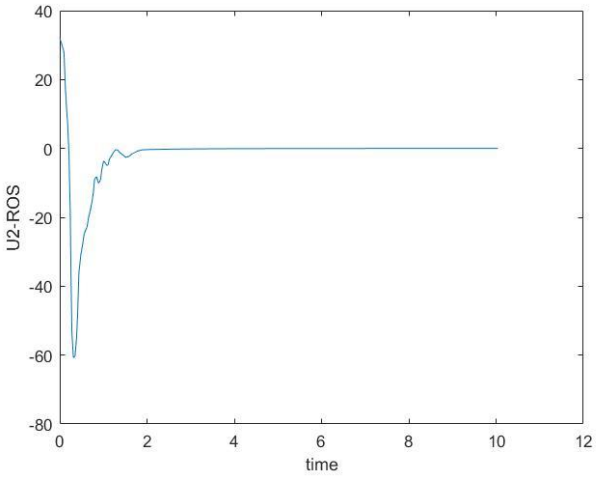
4. theta_dot2 vs time



5. U1 vs time



6. U2 vs time



- A clear difference between the theta1 and theta2 plots of the MATLAB ODE and Gazebo can be seen. Theta1 and theta2 plots obtained from Gazebo travel more than 5.7 radian to reach equilibrium.
- The reason is that Gazebo tracks the angles absolutely. Thus, when Gazebo is initialized, the robot drops to the downward position by rotating in a negative direction.
- When the rrbot_control is run, it again goes in negative direction to get the desired angles of pi/6 and pi/3.
- Finally, when the feedback control is applied it tries to go back to upward position by travelling in the positive direction to compensate for the initial travel when gazebo was initialized as discussed above.
- As for the control inputs, they also differ for the MATLAB ODE and Gazebo due to the same reason.