Trajectory Generation and Feedback Linearization Control for the RRBot Robotic Arm

The physical parameters of the robot are listed below:

$$m_1 = m_2 = 1\ (kg), \quad l_1 = l_2 = 1\ (m), \quad r_1 = r_2 = 0.45\ (m)$$
$$I_1 = I_2 = 0.084\ (kg \cdot m^2), \quad g = 9.81\ (m/s^2)$$

a) ***(2 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 are given by:

$$t_0 = 0, \quad t_f = 10\ sec$$
$$\theta_1(t_0) = 180^\circ, \quad \theta_1(t_f) = 0, \quad \theta_2(t_0) = 90^\circ, \quad \theta_2(t_f) = 0$$
$$\dot{\theta}_1(t_0) = \dot{\theta}_1(t_f) = \dot{\theta}_2(t_0) = \dot{\theta}_2(t_f) = 0$$

```
%--- Generate cubic polynomial trajectories for both the joints ------%

q1 = (pi*t^3)/500 - (3*pi*t^2)/100 - (6189958033024885*t)/10141204801825835211973625643008 + pi;
q2 = (pi*t^3)/1000 - (3*pi*t^2)/200 - (6189958033024885*t)/2028240960365167042394725128 6016 + pi/2;
q1_dot = (3*pi*t^2)/500 - (3*pi*t)/50 - 6189958033024885/10141204801825835211973625643008;
q2_dot = (3*pi*t^2)/1000 - (3*pi*t)/100 - 6189958033024885/2028240960365167042394725128 6016;
q1_ddot = (3*pi*t)/250 - (3*pi)/50;
q2_ddot = (3*pi*t)/500 - (3*pi)/100;
```

b) ***(2 points)*** Consider the equations of motion derived for the robot in Programming Assignment 1. Transform the equations of motion (i.e. the dynamics) of the robot to the standard Manipulator form:

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

⇒ After substituting the given physical parameters.

```
%--------- Manipulator Equation Form ---------%
M = [(9*cos(theta2))/10 + 1573/1000, (9*cos(theta2))/20 + 573/2000;
    (9*cos(theta2))/20 + 573/2000,                      573/2000];


C =[- (9*cos(theta2))/10 - 1573/1000, - (9*cos(theta2))/20 - (9*sin(theta2))/20 - 573/2000;
   (9*sin(theta2))/20 - (9*cos(theta2))/20 - 573/2000, -573/2000];


G =[- (8829*sin(theta1 + theta2))/2000 - (28449*sin(theta1))/2000;
                          -(8829*sin(theta1 + theta2))/2000];
```

c) ***(6 points)*** Derive the symbolic feedback linearization of the robot. Then, design a feedback linearization control for the robot, with a *state-feedback control* for the virtual control input, designed by the eigenvalue placement method.

⇒ Desired eigen values:

$\lambda_d = \{-10, -20, -30, -40\}$

⇒ Matrices A & B for "Upward configuration."

```
% As calculated in Programming Assignment 2
A = [0         0    1.0000         0;
     0         0         0    1.0000;
     12.5769  -11.9611         0         0;
    -16.9227   46.1565         0         0];

B = [0         0;
     0         0;
     1.7250   -4.4345;
    -4.4345   14.8902];
```

=) Calculated Gain Matrix ;

```
K =

   1.0e+03 *

    1.1604   -0.2123    0.1151    0.0120
    0.3281   -0.0221    0.0336    0.0070
```

=) Now, using the equations below the virtual control input & the overall control law are calculated.

```
%---------------- Virtual control input -----------------------%
V = -K*([theta1; theta2; theta1_dot; theta2_dot] - [q1; q2; q1_dot; q2_dot]) + [q1_ddot; q2_ddot];

%--------------- The overall control law ---------------%
Tau = M*V + C*[q1_dot; q2_dot] + G;
```

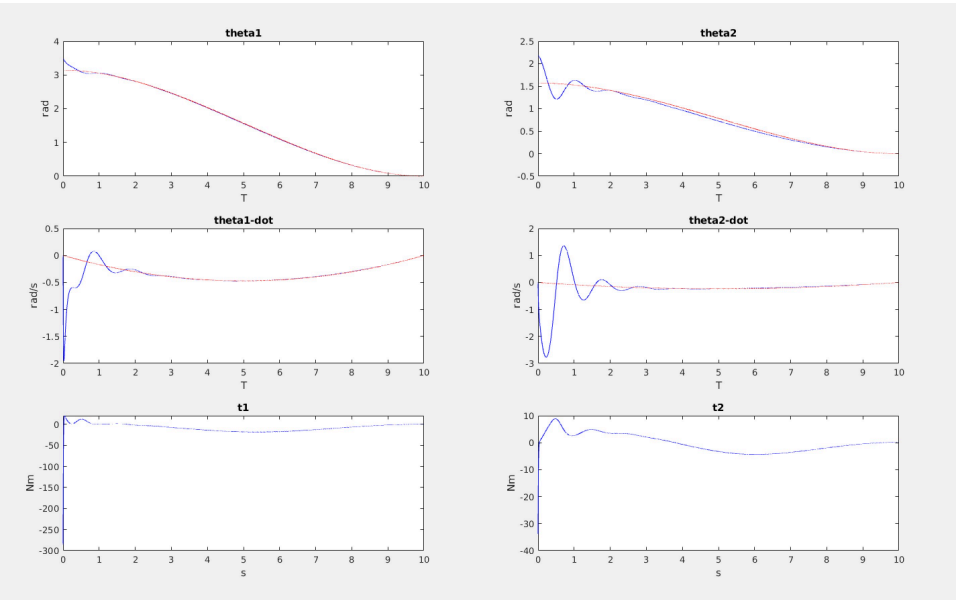
d) ***(2 points)*** Update the `ode` function developed in Programming Assignment 2 to include the feedback linearization control law designed in part (c).

```
%--------------- Get the current state values ---------------%
dX = zeros(4,1);
X = num2cell(X);
[theta1, theta2, theta1_dot, theta2_dot] = deal(X{:});

%--- Generate cubic polynomial trajectories for both the joints ------%

q1 = (pi*t^3)/500 - (3*pi*t^2)/100 - (6189958033024885*t)/10141204801825835211973625643008 + pi;
q2 = (pi*t^3)/1000 - (3*pi*t^2)/200 - (6189958033024885*t)/20282409603651670423947251286016 + pi/2;
q1_dot = (3*pi*t^2)/500 - (3*pi*t)/50 - 6189958033024885/10141204801825835211973625643008;
q2_dot = (3*pi*t^2)/1000 - (3*pi*t)/100 - 6189958033024885/20282409603651670423947251286016;
q1_ddot = (3*pi*t)/250 - (3*pi)/50;
q2_ddot = (3*pi*t)/500 - (3*pi)/100;

%--------- Manipulator Equation Form ---------%
M = [(9*cos(theta2))/10 + 1573/1000, (9*cos(theta2))/20 + 573/2000;
    (9*cos(theta2))/20 + 573/2000,                    573/2000];


C =[- (9*cos(theta2))/10 - 1573/1000, - (9*cos(theta2))/20 - (9*sin(theta2))/20 - 573/2000;
   (9*sin(theta2))/20 - (9*cos(theta2))/20 - 573/2000, -573/2000];


G =[- (8829*sin(theta1 + theta2))/2000 - (28449*sin(theta1))/2000;
                          -(8829*sin(theta1 + theta2))/2000];


%---------------- Virtual control input -----------------------%
V = -K*([theta1; theta2; theta1_dot; theta2_dot] - [q1; q2; q1_dot; q2_dot]) + [q1_ddot; q2_ddot];

%--------------- The overall control law ---------------%
Tau = M*V + C*[q1_dot; q2_dot] + G;

t1 = Tau(1);
t2 = Tau(2);

dX(1) = theta1_dot;
dX(2) = theta2_dot;
dX(3) = (I2*t1 - I2*t2 + m2*r2^2*t1*cos(theta1 + theta2)^2 - m2*r2^2*t2*cos(theta1 + theta2)^2 + m2*r2^2*t1*sin(theta1 + theta2)^2 - m2*r2^2*t2*sin(theta1 + theta2)^2 +
dX(4) = (I1*t2 - I2*t1 + I2*t2 - m2*r2^2*t1*cos(theta1 + theta2)^2 + m2*r2^2*t2*cos(theta1 + theta2)^2 - m2*r2^2*t1*sin(theta1 + theta2)^2 + m2*r2^2*t2*sin(theta1 + the
```

e) ***(3 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:

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

Evaluate the performance of the controller for tracking the desired trajectories generated in part (a). Plot the state trajectories, the control inputs trajectories, and the associated desired trajectories to evaluate the performance.



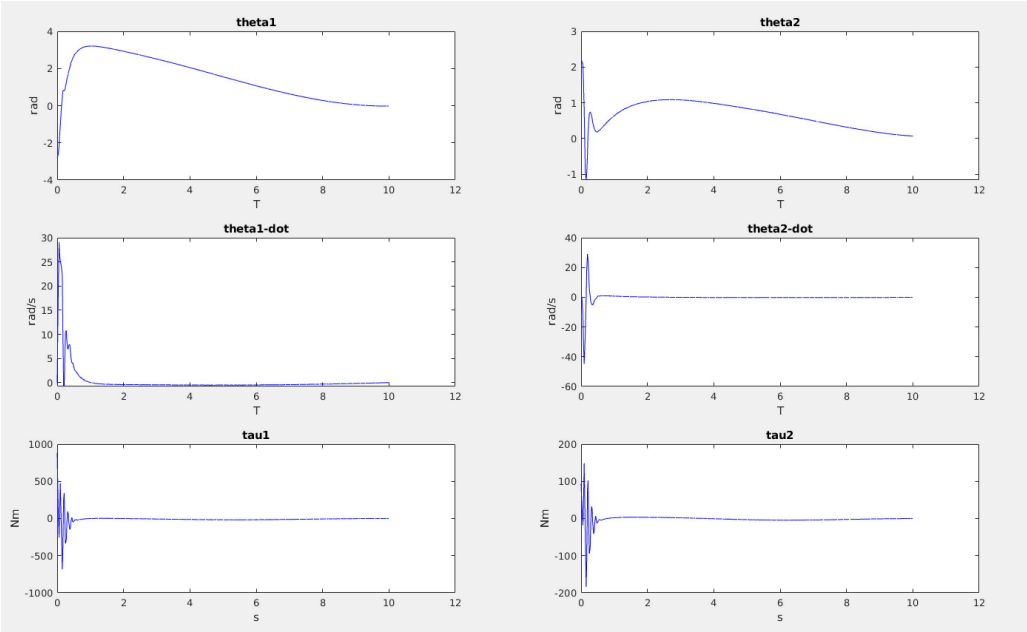
(Trajectories From Ode45)

⇒ Here, Red line represents the desired trajectory & the blue line represents the generated trajectory.

⇒ As no acceleration constraints given, the initial torque requirement is high. This can be optimized by tuning the desired eigen values.

f) ***(5 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`.
Note: as an alternative, you can use the ROS Python script posted on Canvas for this purpose. Update the code inside the `while` loop to control the RRbot robot in Gazebo for 10 seconds using your feedback linearization controller designed in part (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 part (e) in MATLAB. Discuss your findings in your final report.

```
% implementing the trajectory feedback controller
theta1 = jointData.Position(1,1);
theta2 = jointData.Position(2,1);
theta1_dot = jointData.Velocity(1,1);
theta2_dot = jointData.Velocity(2,1);

%Get the trajectory

X = [theta1, theta2, theta1_dot, theta2_dot];

%--- Generate cubic polynomial trajectories for both the joints ------%

q1 = (pi*t^3)/500 - (3*pi*t^2)/100 - (6189958033024885*t)/10141204801825835211973625643008 + pi;
q2 = (pi*t^3)/1000 - (3*pi*t^2)/200 - (6189958033024885*t)/20282409603651670423947251286016 + pi/2;
q1_dot = (3*pi*t^2)/500 - (3*pi*t)/50 - 6189958033024885/10141204801825835211973625643008;
q2_dot = (3*pi*t^2)/1000 - (3*pi*t)/100 - 6189958033024885/20282409603651670423947251286016;
q1_ddot = (3*pi*t)/250 - (3*pi)/50;
q2_ddot = (3*pi*t)/500 - (3*pi)/100;

%--------- Manipulator Equation Form ---------%
M = [(9*cos(theta2))/10 + 1573/1000, (9*cos(theta2))/20 + 573/2000;
    (9*cos(theta2))/20 + 573/2000,                     573/2000];


C =[- (9*cos(theta2))/10 - 1573/1000, - (9*cos(theta2))/20 - (9*sin(theta2))/20 - 573/2000;
   (9*sin(theta2))/20 - (9*cos(theta2))/20 - 573/2000, -573/2000];


G =[- (8829*sin(theta1 + theta2))/2000 - (28449*sin(theta1))/2000;
                           -(8829*sin(theta1 + theta2))/2000];


%--------------- Virtual control input ----------------------%
V = -K*([theta1; theta2; theta1_dot; theta2_dot] - [q1; q2; q1_dot; q2_dot]) + [q1_ddot; q2_ddot];

%-------------- The overall control law ---------------%
Tau = M*V + C*[theta1_dot; theta2_dot] + G;

tau1.Data = Tau(1);
tau2.Data = Tau(2);
```



From Gazebo Simulation