Aditya Jagani

RBE 502 — Robot Control

Instructor: Siavash Farzan

Fall 2021

Programming Assignment 3:
Trajectory Generation and Feedback Linearization Control for the RRBot Robotic Arm

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:

$$t_0 = 0, \quad t_f = 10\ sec$$

$$\theta_1(t_0) = 180^\circ, \quad \theta_1(t_f) = 0, \quad \theta_2(t_0) = 90^\circ, \quad \theta_2(t_f) = 0$$

$$\dot\theta_1(t_0) = \dot\theta_1(t_f) = \dot\theta_2(t_0) = \dot\theta_2(t_f) = 0$$

```
Trajectory can be calculated as follows:

q_t = a0 + a1*t + a2*(t^2) + a3*(t^3)

    A = [1  t0    t0^2   t0^3;
         0  1     2*t0   3*t0^2;
         1  tf    tf^2   tf^3;
         0  1     2*tf   3*tf^2];

    q = [q0 qf q0_dot qf_dot]';

    a = [a0; a1; a2; a3];

    sol = solve([A*a==q],a);
```

We solve for ‘a’ with A and q matrix. Once we obtain the values for ‘a’ matrix, we substitute them in q(t) to find out the desired trajectory equation:

Once we get q(t), we can obtain q_dot(t) and q_ddot(t) by differentiating and double differentiating respectively with time. We do this for both joints.

```
q1_desired = (pi*t^3)/500 - (3*pi*t^2)/100 + pi;
q2_desired = (pi*t^3)/1000 - (3*pi*t^2)/200 + pi/2;
q1_dot_desired = (3*pi*t^2)/500 - (3*pi*t)/50;
q2_dot_desired = (3*pi*t^2)/1000 - (3*pi*t)/100;
q1_ddot_desired = (3*pi*t)/250 - (3*pi)/50;
q2_ddot_desired = (3*pi*t)/500 - (3*pi)/100;
```

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:

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

```
Solve for control inputs U = [u1 u2]
Substitute values of mass, inertia, gravity etc. in equations of motions
% Calculate g(q) by substituting q_dot = 0

G = - (8829*sin(theta1 + theta2))/2000 - (28449*sin(theta1))/2000 -
(theta2_dot*((9*theta1_dot*sin(theta2))/5 + (9*theta2_dot*sin(theta2))/10))/2

% Calculate C(q,q_dot) by substituting q_ddot = 0
C = -(qdot2*((9*qdot1*sin(q2))/5 + (9*qdot2*sin(q2))/10))/2
(9*qdot1^2*sin(q2))/20

% Calculate M(q) by tau - g(q) - C(q,q_dot)
M = [(9*cos(theta2))/10 + 1573/1000, (9*cos(theta2))/20 + 573/2000]
[ (9*cos(theta2))/20 + 573/2000,                          573/2000]


% Final Manipulator Form:  M(q)q_ddot + C(q,q_dot)q_dot + g(q) = tau
tau =
g + theta1_ddot*((9*cos(theta2))/10 + 1573/1000) +
theta2_ddot*((9*cos(theta2))/20 + 573/2000)
g + (573*theta2_ddot)/2000 + theta1_ddot*((9*cos(theta2))/20 + 573/2000)
```

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.

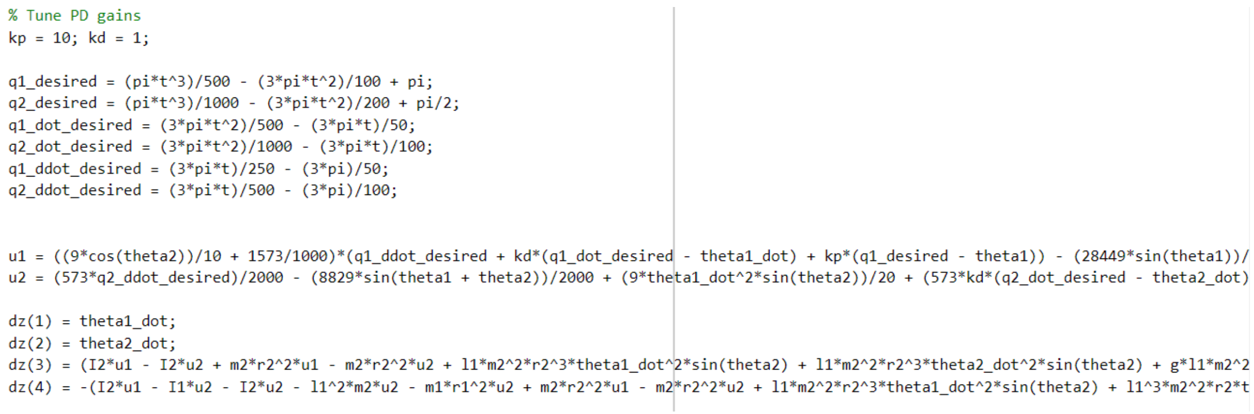
```
Using virtual control input v:
```

```
V =

q1_ddot_desired + Kp*q1_desired + Kd*(q1_dot_desired - theta1_dot)
q2_ddot_desired + Kp*q2_desired + Kd*(q2_dot_desired - theta2_dot)
```

```
The symbolic feedback linearization of the robot is obtained as follows:
tau =

((9*cos(theta2))/10 + 1573/1000)*(q1_ddot_desired + Kp*q1_desired +
Kd*(q1_dot_desired - theta1_dot)) - (28449*sin(theta1))/2000 - (8829*sin(theta1 +
theta2))/2000 + ((9*cos(theta2))/20 + 573/2000)*(q2_ddot_desired + Kp*q2_desired
+ Kd*(q2_dot_desired - theta2_dot)) - (theta2_dot*((9*theta1_dot*sin(theta2))/5 +
(9*theta2_dot*sin(theta2))/10))/2

(573*q2_ddot_desired)/2000 - (8829*sin(theta1 + theta2))/2000 +
(9*theta1_dot^2*sin(theta2))/20 + (573*Kp*q2_desired)/2000 + ((9*cos(theta2))/20
+ 573/2000)*(q1_ddot_desired + Kp*q1_desired + Kd*(q1_dot_desired - theta1_dot))
+ (573*Kd*(q2_dot_desired - theta2_dot))/2000
```

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.

We implemented the feedback linearization control law using an ode function which is located in the file 'myode_RRbot.m'.



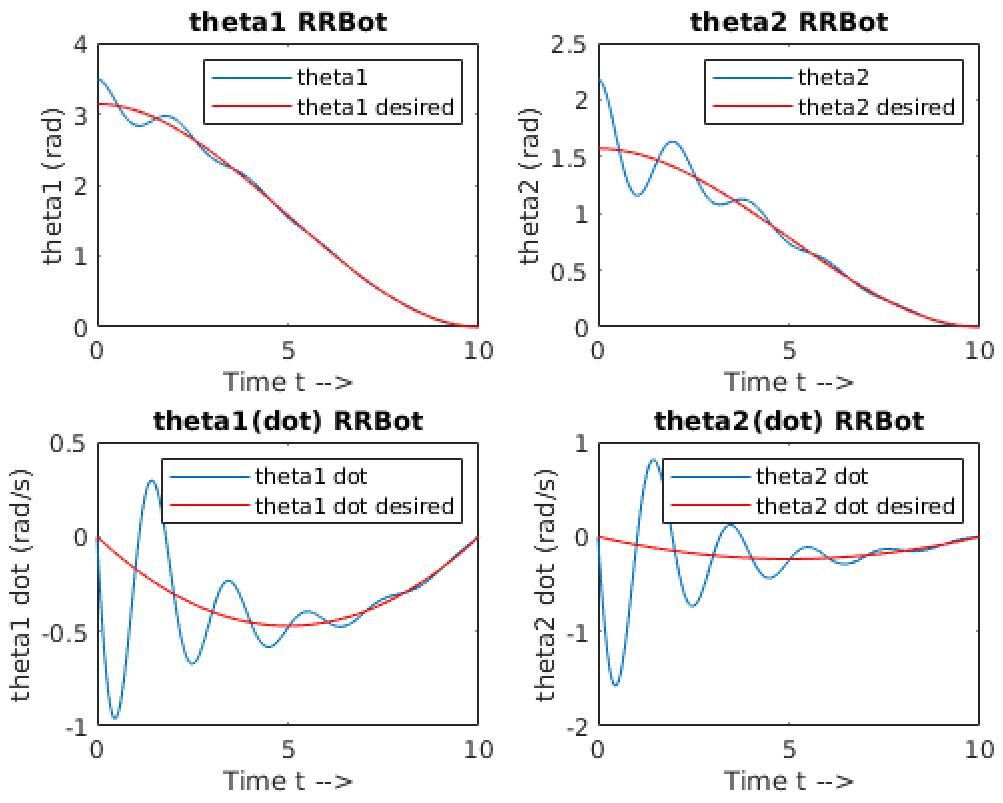
```
% Tune PD gains
kp = 10; kd = 1;

q1_desired = (pi*t^3)/500 - (3*pi*t^2)/100 + pi;
q2_desired = (pi*t^3)/1000 - (3*pi*t^2)/200 + pi/2;
q1_dot_desired = (3*pi*t^2)/500 - (3*pi*t)/50;
q2_dot_desired = (3*pi*t^2)/1000 - (3*pi*t)/100;
q1_ddot_desired = (3*pi*t)/250 - (3*pi)/50;
q2_ddot_desired = (3*pi*t)/500 - (3*pi)/100;


u1 = ((9*cos(theta2))/10 + 1573/1000)*(q1_ddot_desired + kd*(q1_dot_desired - theta1_dot) + kp*(q1_desired - theta1)) - (28449*sin(theta1))/
u2 = (573*q2_ddot_desired)/2000 - (8829*sin(theta1 + theta2))/2000 + (9*theta1_dot^2*sin(theta2))/20 + (573*kd*(q2_dot_desired - theta2_dot)

dz(1) = theta1_dot;
dz(2) = theta2_dot;
dz(3) = (I2*u1 - I2*u2 + m2*r2^2*u1 - m2*r2^2*u2 + l1*m2^2*r2^3*theta1_dot^2*sin(theta2) + l1*m2^2*r2^3*theta2_dot^2*sin(theta2) + g*l1*m2^2
dz(4) = -(I2*u1 - I1*u2 - I2*u2 - l1^2*m2*u2 - m1*r1^2*u2 + m2*r2^2*u1 - m2*r2^2*u2 + l1*m2^2*r2^3*theta1_dot^2*sin(theta2) + l1^3*m2^2*r2*t
```

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:

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

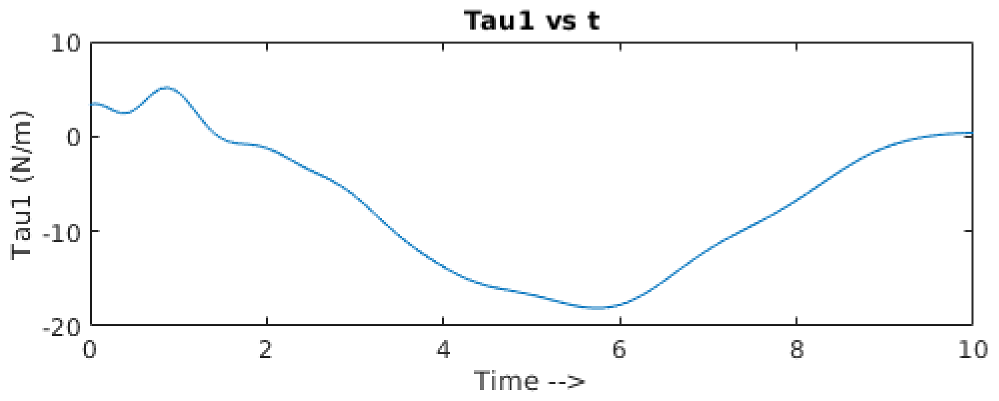
Plot the state trajectories and the control inputs trajectories and the associated desired trajectories to evaluate the performance.
If the performance is not satisfactory (i.e. the system does not track and converge to the desired trajectory), go back to Step (c) and update the state-feedback control gains for the virtual control input.
Hint: The control inputs can be reconstructed after the solution is returned by `ode45`.

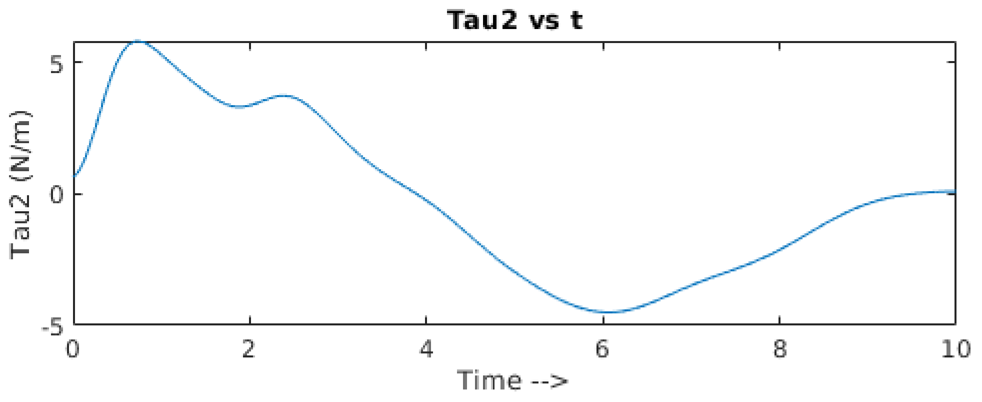
Given the desired joint angles and their corresponding velocities, we have implemented a PD controller using state feedback linearization to follow a generated cubic trajectory and follow it. The results are as depicted below:



We can observe, that despite oscillations in the beginning, our controller is able to cope up to the desired trajectory and follow it by the end of 10 seconds. In order to fine tune the graph, we can change Kp and Kd gain values to observe a better performance.

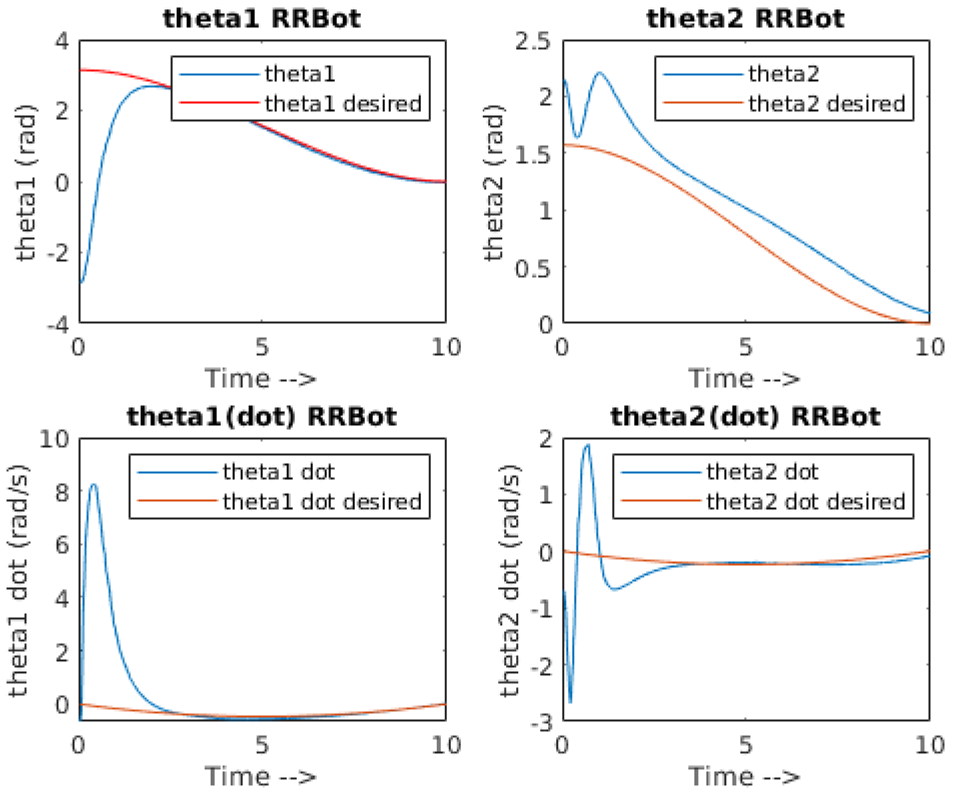
The plot of control inputs U1 and U2 is as follows:



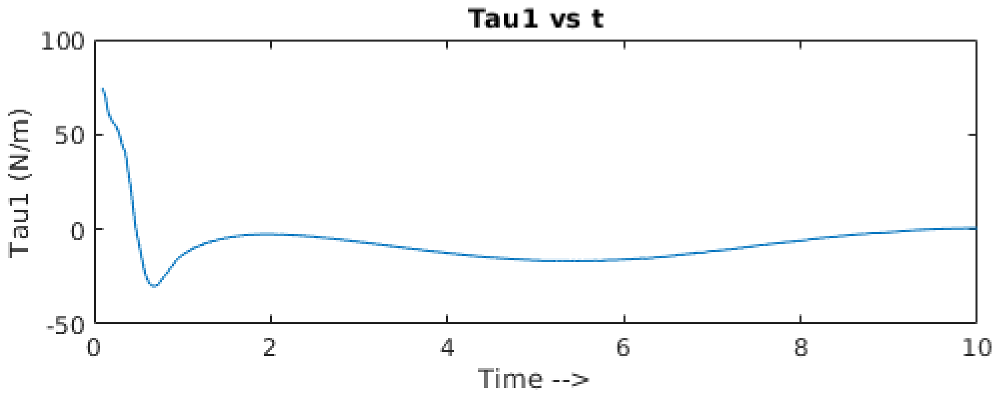


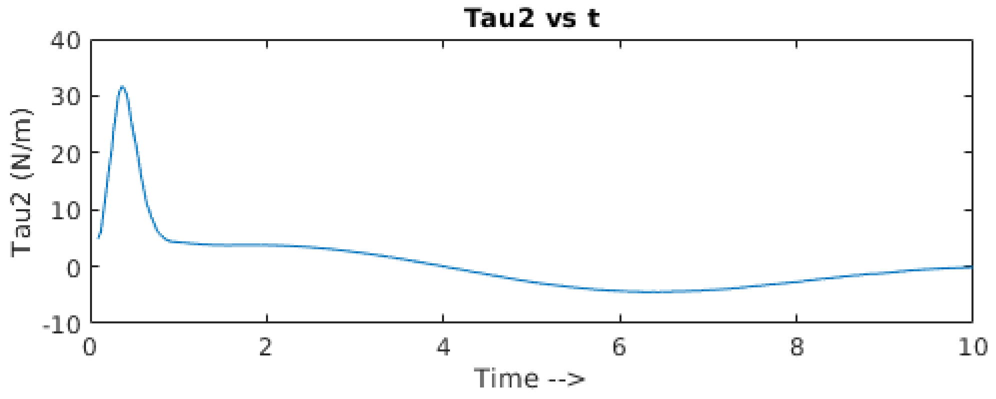
f) *(15 points)* Create a new copy of the `rrbot_control.m` file provided in Programming Assignment 2, and rename the new file to `rrbot_traj_control.m`. Update the code inside the `while` loop to control the RRbot robot in Gazebo for 10 seconds using your feedback linearization controller designed in 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

When we implement our designed controller on the RRBot, we observe the following trajectories. On closely observing them we can notice that we are able to track the desired trajectory with time. We have plotted them as follows:



The below graph represents the plot of Control Inputs U1 and U2 with time.


Tau1 vs t
Tau1 (N/m)
100
50
0
-50
0
2
4
6
8
10
Time -->


Tau2 vs t
Tau2 (N/m)
40
30
20
10
0
-10
0
2
4
6
8
10
Time -->