RBE 502 — Robot Control

Instructor: Siavash Farzan

Fall 2022

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

3.1 Overview

The RRBot robot – for which we studied its dynamics and state-feedback control in Programming Assignments 1 and 2 – is a simple two-link robot arm with two revolute joints.

You should have already installed and built the RRBot ROS package in Programming Assignment 0. To visualize the robot in Gazebo, you can run the following command in the Ubuntu terminal:

```
roslaunch rrbot_gazebo rrbot_world.launch
```

In Programming Assignment 2, we linearized the system and designed a state-feedback control to stabilize the robot to the upward equilibrium point. In this assignment, we will generate a dynamically feasible trajectory for the robot and derive a feedback linearization control to track the trajectory and control the robot motion in Gazebo, as a physics engine-based simulator.

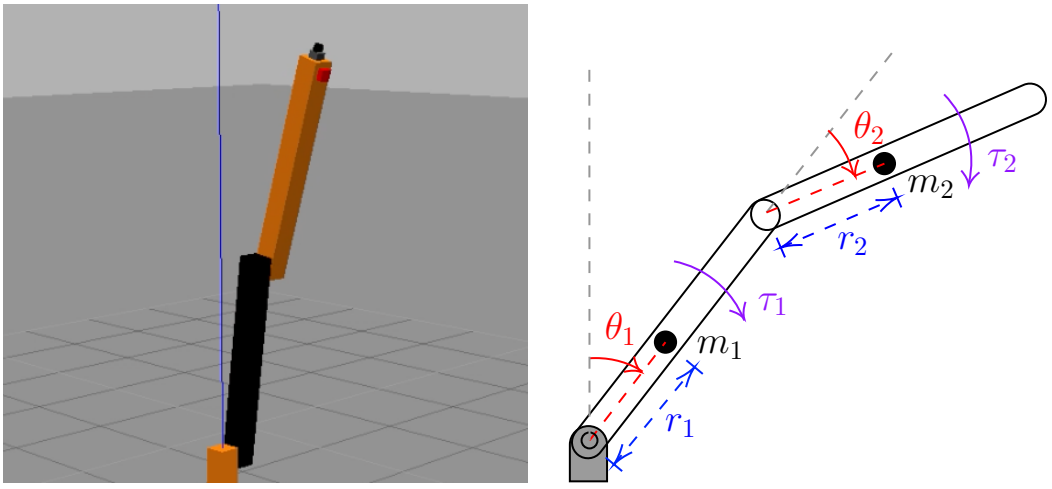

Figure 1: 2-DoF Revolute-Revolute robot arm (RRBot)

3.2 Problem Statement

Consider the 2-DoF Revolute-Revolute robot arm (RRBot) shown in Figure 1. Each joint is equipped with an actuator, applying control torques τ_1 and τ_2 to joint 1 and joint 2, respectively. Lengths of the first and second link are l_1 and l_2, respectively. The links have distributed mass, and r_1 and r_2 denote the location of the center of mass for each link (with mass of m_1 and m_2). The moments of inertia of link 1 and link 2 are I_1 and I_2, respectively. θ_1 is the angle of the first link with respect to the vertical axis, and θ_2 is the angle of the second link with respect to the first link.

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:

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

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:

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

Hint: Feel free to derive the term $C(q, \dot{q})\dot{q}$ in its entirety. There would be no need to isolate the $C(q, \dot{q})$ matrix for this assignment.

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.

d) ***(2 points)*** Update the `ode` function developed in Programming Assignment 2 to include the feedback linearization control law designed in part (c).
Hint: You will need to evaluate the cubic polynomial trajectories inside the `ode` function to obtain the desired states at each point in time.

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.
If the performance is not satisfactory (i.e. the system does not track and converge to the desired trajectory), go back to part (c) and update the state-feedback control gains for the

virtual control input.
Tune the location of eigenvalues in the state-feedback control design such that the joint torques (i.e. control inputs) stay within the bounds below, while tracking the trajectory successfully.

$$-20 \leq u_1 \leq 20\ (Nm), \qquad -10 \leq u_2 \leq 10\ (Nm)$$

Hint: The control inputs can be reconstructed after the solution is returned by `ode45`.

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 porpose.
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.

3.3 Robot Controller Setup in Gazebo

Before testing your control design on the RRbot robot in Gazebo, you need to setup your ROS controller nodes for the robot by following the steps in Section 2.3 of Programming Assignment 2.

We are now ready to evaluate the controller performance in Gazebo.

3.4 Performance Testing in Gazebo

i) Open a terminal in Ubuntu, launch the Gazebo simulator and spawn a new robot:

```
roslaunch rrbot_gazebo rrbot_world.launch
```

ii) Once the robot is successfully spawned in Gazebo, in a new terminal, launch the rrbot control node:

```
roslaunch rrbot_control rrbot_effort_control.launch
```

iii) We can now test the feedback linearization control script by running the `rrbot_traj_control.m` in MATLAB (or `rrbot_control.py` in ROS).

iv) Make sure to check the performance of your script for all the feedback linearization controllers that deemed satisfactory in MATLAB (determined in part (f)).

3.5 Submission

- Submission: individual submission via Gradescope. Submit the MATLAB (`.m`) files you have written as a zip folder and a detailed report including:
  - mathematical formulation and derivation of the trajectory generation method in part (a);
  - expressions for the manipulator form derived in part (b);

  - mathematical formulation and derivation of the feedback linearization control law and the virtual control input design in part (c);
  - discussion on the location of eigenvalues selected in part (c);
  - discussion on the ODE function development in part (d).

  Moreover, include the trajectory plots generated in part (e) and part (f) in your report, and discuss the performance. Compare the control performance in Gazebo in part (f) with the MATLAB performance in part (e).

- Due: as specified on Canvas.
- Files to submit: (please use exactly the same filename, and do NOT include the PDF file in the zip folder)
  - `LastName_ProgAssignment3_report.pdf`
  - `LastName_ProgAssignment3_matlab.zip`