RBE 502 — Robot Control

Instructor: Siavash Farzan

Fall 2022

Programming Assignment 2:

State Feedback Control of the RRBot Robotic Arm

2.1 Overview

The RRBot robot – for which we studied the dynamics in Programming Assignment 1 – 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 1, we derived the dynamics model and state-space representation of the robot in MATLAB, and implemented a passive simulation of the system. We will use the resulting equations of motion in this assignment to design a state-feedback controller for the robot and control the robot motion in Gazebo.

2.2 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 links 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. 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.

a) ***(2 points)*** Using the symbolic equations of motion derived in Programming Assignment 1, determine the equilibrium points of the robot in MATLAB.
Hint: It might be easier to use the original *second-order* EoMs for this purpose, and not the state-space representation. MATLAB functions `subs` and `solve` may come in handy!

b) ***(2 points)*** Derive the Jacobian linearization of the symbolic state-space representation (derived in Programming Assignment 1) in MATLAB to obtain the symbolic representation of the state matrix A and the input matrix B. MATLAB function `jacobian` can be used for linearization.
Substitute the physical parameters of the robot (listed below) in the state and input matrices.

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

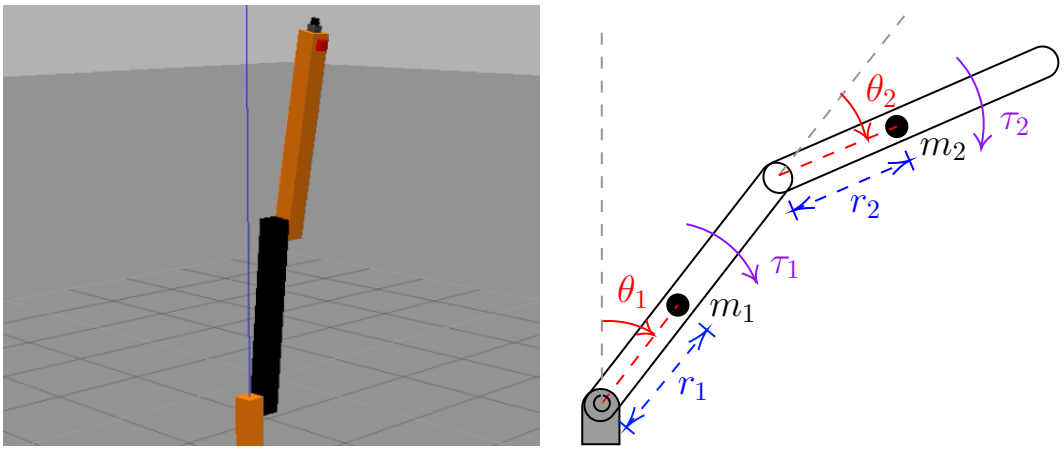

Figure 1: 2-DoF Revolute-Revolute robot arm (RRBot)

c) ***(2 points)*** Investigate the stability properties of the linearized system around each of the equilibrium points found in Step (a).

d) ***(0.5 points)*** Investigate the controllability of the linearized system around the equilibrium point corresponding to the "upward" configuration.

e) ***(2.5 points)*** Design a state-feedback control to stabilize the system around the upward equilibrium point. Feel free to use MATLAB function `place`. Include the selected eigenvalues and the resulting gains in your report.

f) ***(6 points)*** Use `ode45` and the `ode` function developed in Programming Assignment 1 to construct a simulation of the system in MATLAB with the initial conditions $\theta_1(0) = 30°$, $\theta_2(0) = 45°$, $\dot{\theta}_1(0) = 0$ and $\dot{\theta}_2(0) = 0$ and a time-span of 10 seconds.
Do not forget to implement your feedback control law (designed in Step (e)) inside the `ode` function. Plot the state trajectories (of the state space form) and the control inputs trajectories to evaluate the performance.
If the performance is not satisfactory (e.g. the system does not converge to the "upward equilibrium point), go back to Step (e) and change the location of the assigned eigenvalues.
Optional: Try to choose the eigenvalues such that the joint torques (i.e. control inputs) stay within the bounds of:

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

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

g) ***(5 points)*** Copy and paste the following code into a new MATLAB script named `rrbot_control.m`. Complete the code inside the `while` loop to control the RRbot robot in Gazebo for 10 seconds using your state-feedback controller designed in Step (f). 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 Step (f) in MATLAB. Discuss your findings in your final report.

```
clear; close; clc;

% ROS Setup
rosinit;

j1_effort = rospublisher('/rrbot/joint1_effort_controller/command');
j2_effort = rospublisher('/rrbot/joint2_effort_controller/command');
JointStates = rossubscriber('/rrbot/joint_states');

tau1 = rosmessage(j1_effort);
tau2 = rosmessage(j2_effort);

tau1.Data = 0;
tau2.Data = 0;

send(j1_effort,tau1);
send(j2_effort,tau2);

client = rossvcclient('/gazebo/set_model_configuration');
req = rosmessage(client);
req.ModelName = 'rrbot';
req.UrdfParamName = 'robot_description';
req.JointNames = {'joint1','joint2'};
req.JointPositions = [deg2rad(30), deg2rad(45)];
resp = call(client,req,'Timeout',3);

tic;
t = 0;

while(t < 10)
    t = toc;

    % read the joint states
    jointData = receive(JointStates);

    % inspect the "jointData" variable in MATLAB to get familiar with its
       structure

    % implement your state feedback controller below

    tau1.Data = ...;
    tau2.Data = ...;

    send(j1_effort,tau1);
    send(j2_effort,tau2);

    % sample the time, joint state values, and calculated torques here to be
       plotted at the end

end

tau1.Data = 0;
tau2.Data = 0;
```

```
send(j1_effort,tau1);
send(j2_effort,tau2);

% disconnect from roscore
rosshutdown;

% plot the trajectories
```

2.3 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. We downloaded the `gazebo_ros_demos` package and installed the control dependencies in Programming Assignment 0.

For this assignment, we will need two new files related to the joint effort controllers.

Download the `rrbot_control_effort.zip` from Canvas and extract the zip folder. Move the `rrbot_control_effort.yaml` file to the following directory

```
~/rbe502_ros/src/gazebo_ros_demos/rrbot_control/config/
```

Similarly, move the `rrbot_effort_control.launch` file to the following directory

```
~/rbe502_ros/src/gazebo_ros_demos/rrbot_control/launch/
```

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

2.4 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 state-feedback control script by running the `rrbot_control.m` in MATLAB.

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

2.5 Submission

- Submission: individual submission via Gradescope. Submit the MATLAB (`.m`) files you have written as a zip folder, and a detailed report. In your report:
  - describe the mathematical background for the linearization and state feedback control design of the two-link robot;
  - include all the results of Step (a) to Step (e);
  - include the trajectory plots generated in Step (f) and Step (g);
  - discuss your findings from Step (g) in Gazebo, and include a screenshot of the robot stabilized to the upward configuration.

  Submissions without MATLAB codes will not be graded.
- Due: as specified on Canvas.
- Files to submit: (please use the exact file name, and do NOT include the PDF file in the zip folder)
  - `LastName_ProgAssignment2_report.pdf`
  - `LastName_ProgAssignment2_matlab.zip`