RBE 502 — Robot Control

Instructor: Siavash Farzan

Fall 2022

Programming Assignment 1:

Dynamic Modeling and State-Space Representation of the RRBot Robotic Arm

1.1 Overview

For the next programming assignments, we will be using the RRBot robot, which 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 this assignment, we will derive the dynamics model and state-space representation of the robot in MATLAB. The resulting equations of motion will be used in the next assignments to design control algorithms for the robot and control the motion of robot in Gazebo.

1.2 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.

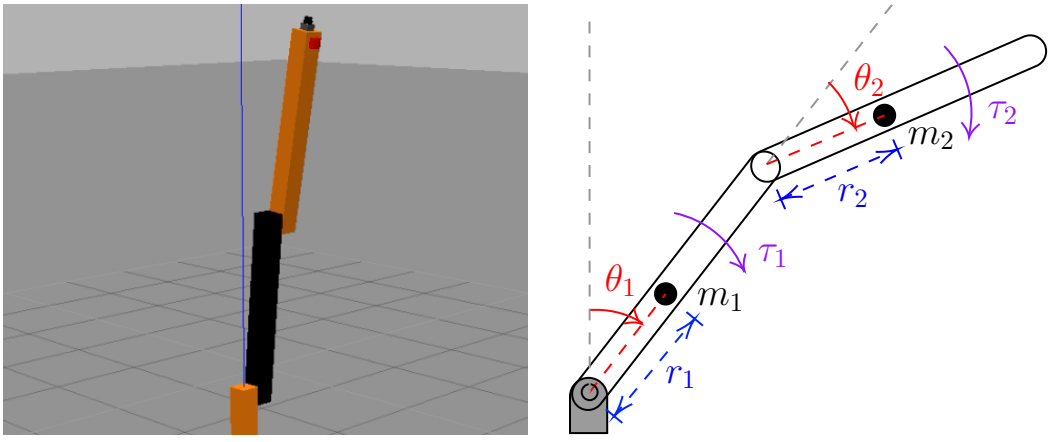

Figure 1: 2-DoF Revolute-Revolute robot arm (RRBot)

a) ***(10 points)*** Using the symbolic expressions in MATLAB, derive the equations of motion for the robot in MATLAB. That is, derive the dynamics in MATLAB by explicitly writing kinetic and potential energy symbolically and taking derivatives of the Lagrangian function, according to the Euler-Lagrange method.

Note 1: For taking partial derivatives (e.g. $\frac{\partial L}{\partial q}$), you can use the `jacobian` function in MATLAB:
`https://www.mathworks.com/help/symbolic/sym.jacobian.html`

Note 2: For taking time derivatives symbolically (e.g. $\frac{d}{dt}\frac{\partial L}{\partial \dot{q}}$), you can again use the `jacobian` function along with the *chain rule*:

$$\text{recall:} \quad \frac{dy}{dt} = \frac{dy}{dq}\frac{dq}{dt}$$

Therefore in MATLAB, $\frac{d}{dt}\frac{\partial L}{\partial \dot{q}}$ can be implemented as the following:

```
dDL_dtDdq = jacobian(DL_Ddq,[q;dq])*[dq;ddq];
```

Display the resulting equations of motion to be included in your report.

b) ***(5 points)*** Using the `solve` function in MATLAB, find the state-space representation for the equations of motion derived in step (a).

Display the resulting first order equations to be included in your report.

c) ***(5 points)*** Use `ode45` to construct a simulation of the system in MATLAB (set $\tau_1 = \tau_2 = 0$) given the initial conditions $\theta_1(0) = 30°$, $\theta_2(0) = 45°$, $\dot{\theta}_1(0) = 0$ and $\dot{\theta}_2(0) = 0$ and time-span of 10 seconds. The following physical parameters can be used for simulation:

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

After the simulation is done, plot the four state trajectories (associated to the states of the system, i.e. $\theta_1(t)$, $\theta_2(t)$, $\dot{\theta}_1(t)$ and $\dot{\theta}_2(t)$) with respect to time. Save the resulting trajectory plots to be included in your final report.

d) ***(Optional)*** 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/
```

Copy and paste the code block on the next page into a new MATLAB script named `rrbot_passive.m`. The code uses the MATLAB ROS Toolbox that allows the integration of ROS with MATLAB. Complete the code inside the `while` loop to collect the joint states data in Gazebo for 10 seconds, when the robot is passively released from the same initial states as Step (c). Sample the time and joint states data at each loop to be plotted at the end. Feel free to define new functions and variables in your program if needed.

Follow the steps below to test the performance of your MATLAB-ROS script:

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 MATLAB script by running the `rrbot_passive.m` in MATLAB.

Save the resulting trajectories to be included in the final report. Compare the Gazebo trajectories with those obtained in simulation in Step (c) (heads-up: the trajectories will be different). Discuss your findings in your final report.

Code block for Step (d):

```
clear; close; clc;

% ROS Setup
rosinit;

JointStates = rossubscriber('/rrbot/joint_states');

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

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

end

% disconnect from roscore
rosshutdown;

% plot the trajectories
```

1.3 Submission

- Submission: individual submission via Gradescope. Submit the MATLAB (`.m`) files you have written as a zip folder, and a brief report explaining the dynamic modeling approach developed in Step (a). Include the resulting equations of motion derived in Step (a), State-space representation derived in Step (b), and the trajectory plots generated in Step (c) in your report. The equations can be retyped or copied/pasted directly from MATLAB in the report. Submissions without MATLAB codes will not be graded.

  Optional: Report the actual trajectory plots from Gazebo generated in Step (d), and compare them with the trajectories simulated in Step (c). Discuss any discrepancy and what might be the cause of it.

- 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_ProgAssignment1_report.pdf`
  - `LastName_ProgAssignment1_matlab.zip`