EECS 106B Lab 4: Soft Robotics Modeling *

Due date: Monday, May 6th at 11:59pm

Goal

System Identification of Soft Robotics

You will be conducting system identification of a soft robotic finger. First you'll calibrate some sensors and then you'll fit parameters to static and dynamic data in order to build a model of the finger.

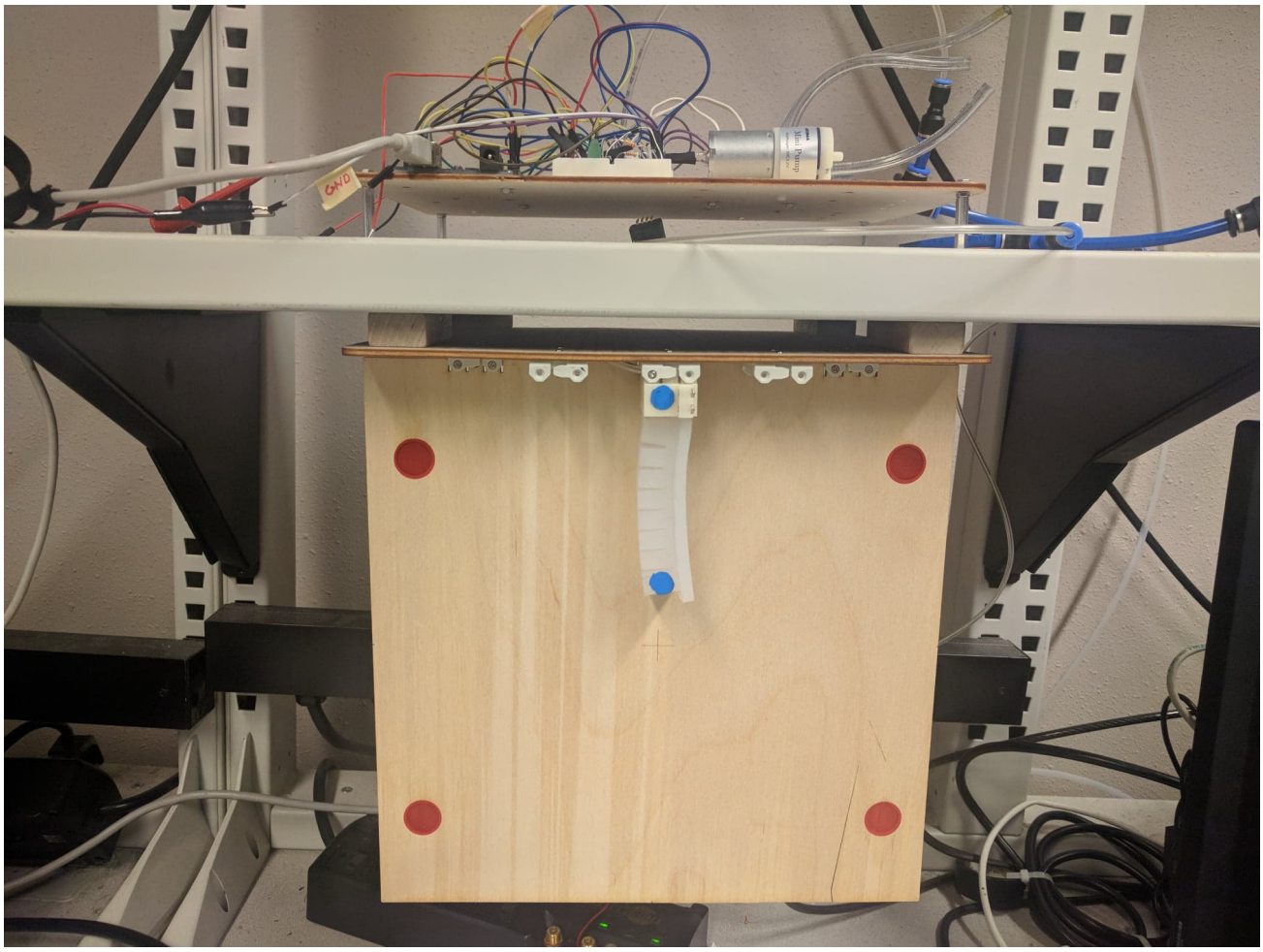

Figure 1: The soft robotic system

*Developed by Valmik Prabhu and Chris Correa, Spring 2019

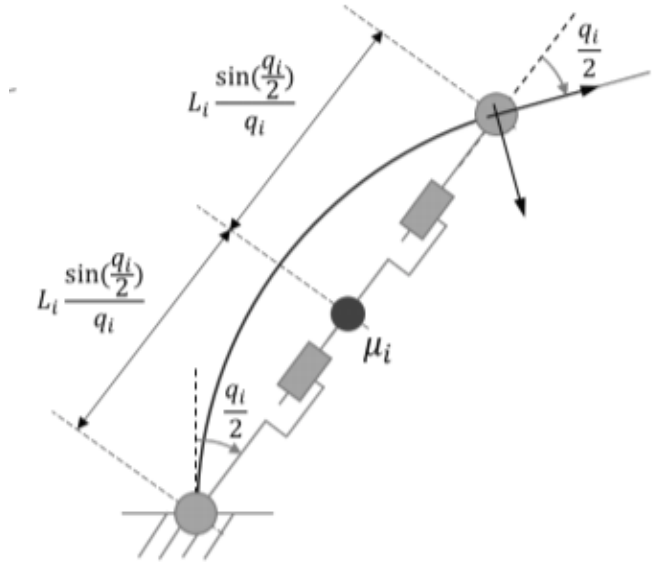

Figure 2: A soft robotic joint modeled as four rigid joints

Contents

1 Theory

1.1 Soft Robot Modeling

Modeling the soft robotic finger will be done using the methodology from this paper from Daniella Rus's lab at MIT. Here each soft joint is modeled as four rigid joints as shown in 2.
We've already seen how to find the dynamics for a rigid manipulator, so we'll start there

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = B\tau + J(\theta)f_{ext} \tag{1}$$

Here M is the inertia matrix, C the coriolis matrix, G the gravity matrix, and B the input dynamics map. We're ignoring external force. Note that in the paper Rus names the inertia matrix B, and ignores the input map (in this case, it's simply the identity matrix). Of course, this model is only an approximation, so we add stiffness and damping terms to take into account the compliant nature of the joint.

$$M(q)\ddot{q} + [C(q,\dot{q}) + D]\dot{q} + G(q) + Kq = B\tau \tag{2}$$

Here we model the stiffness (Young's modulus) and damping as constants, but in reality rubbers are hyperelastic. Walsh et al use a fourth order stiffness model where

$$\sigma(\lambda) = \frac{\lambda^4 - 1}{\lambda^3}\left(2C_1 + 4C_2\left(\lambda - \frac{1}{\lambda}\right)^2\right) \tag{3}$$

and

$$\lambda = \frac{q}{\sin(q)} \tag{4}$$

and

$$K = \frac{\sigma}{\lambda} \tag{5}$$

You'll be comparing the two models later in the lab. Furthermore we model the input dynamics with a second order filter

$$\tau = \frac{\alpha}{(\gamma s + 1)^2}u \tag{6}$$

which means that

$$\gamma^2\ddot{\tau} + 2\gamma\dot{\tau} + \tau = \alpha u \tag{7}$$

Thus our final dynamics are

$$M(q)\ddot{q} + [C(q,\dot{q}) + D]\dot{q} + G(q) + K(q)q = B\tau + J(\theta)f_{ext} \tag{8}$$

1.2 Least Squares

Your goal in the lab will be to identify values for K, D, α and γ, and you'll do that with a nonlinear least squares solver. The least-squares solver will solve the following problem:

$$\min_{K,D,\alpha,\gamma}(\|q_m - q_s(K,D,\alpha,\gamma,u)\|^2) \tag{9}$$

where q_m is your measured state and q_s is the simulated state, given a timeseries of input pwm values and the set of constants you'll be solving for. In order to find these constants, you'll need to simulate the dynamics of the system given K, D, α, γ, q_0, $\dot{q}_0$, τ_0, and $\dot{\tau}_0$ and compare them to data.

2 Logistics and Lab Safety

We expect that all students in this class are mature and experienced at working with hardware, so we give you much more leeway than in EECS 106A. Please live up to our expectations.

2.1 Groups, Robots and Accounts

Remember that groups must be composed of 2-3 students, at least one of whom must have completed EECS 106A. Robots should be reserved on the robot calendar here. The rules are

1. **Groups with a reservation have priority.** You can go in and use (or continue using) the hardware whenever you like if no one has a reservation. However, you must pass along the hardware once the next reservation starts. Please be respectful and try to plan ahead; it's not cool to take the first twenty minutes of another group's section winding down.

2. **Do not reserve more than two hours at a time.** This is shared hardware and we want to ensure that all groups have enough time to complete the lab.

3. **One computer per group (if needed)** We only have ten lab computers, and have around thirty students. If more than ten students want to use the computers, please try to limit yourself to one computer per group. In addition, groups with robot reservations have priority on the computers next to their respective robots. Groups can accomplish working in parallel by using personal computers, file-sharing software like git, remoting into the lab computers, and/or using simulators like Gazebo or RViz.

4. **Please remember to charge the robots after use**. That way others will be able to use the bot for all of their time slot.

2.2 Safety

Remember to follow the lab safety rules:

1. **Never work on hardware alone.** Robots are potentially dangerous and very expensive, so you should always have someone on hand to help if something goes wrong. This person should be another 106B student, though in cases when group schedules are highly incompatible, you may talk to a TA to arrange an alternate solution, such as bringing a friend (note that you would be entirely responsibe for this person's conduct).

2. **Do not leave the room while the robot is running.** Running robots must be under constant observation.

3. **Always operate the Soft robot with the "Output On/Off" button within reach.** If the finger starts to inflate too much, or something else goes wrong, immediately press the "Output On/Off" button on the power supply.

4. **Power off the robot when leaving.** Unless you're trading off with another group, the robots should be powered down for safety. To power on/off the robot, press the white button on the back of the robot. For turtlebots, cycle power (to kill any remaining processes), then power on and plug into a charger. Turtlebots cannot charge when off.

5. **Terminate all processes before logging off.** Leaving processes running can disrupt other students, is highly inconsiderate, and is difficult to debug. Instead of logging off, you should type

   ```
   killall -u <username>
   ```

   into your terminal, where `<username>` is your instructional account username (`ee106b-xyz`). You can also use `ps -ef | grep ros` to check your currently running processes.

6. **Do not modify robots without consulting lab staff.** Last semester we had problems with students losing gripper pieces and messing with turtlebots. This is inconsiderate and makes the TAs' lives much more difficult.

7. **Tell the course staff if you break something.** This is an instructional lab, and we expect a certain amount of wear and tear to occur. However, if we don't know something is broken, we cannot fix it.

2.3 Questions and Help

If you experience software-related errors, please perform the following:

1. Document exactly what you did leading up to the error (commands, terminal output, etc.)

2. Post on Piazza with a description of the error, the above materials, and a description of what you did to try and fix it.

Chris and Valmik **Will Not** diagnose any programming errors without the above documentation. However, feel free to ask us theoretical questions on piazza, during office hours, or after lecture or discussion.

3 Project Tasks

You'll be attemping a couple tasks

1. Note that one of the two soft robot setups uses a finger made from EcoFlex-30 and the other a finger made from DragonSkin-30. Please note which material you're using and use the same setup for the duration of the lab.

2. First you'll be calibrating the bend sensor and pressure sensor. To calibrate the pressure sensor, simply look at the documentation here. To calibrate the bend sensor, input ten different pwm values and take data (making sure to ignore any initial transient response). Plot the mean resistance and variance as a function of bend angle, which you can get from the camera data. Then perform a regression to get bend angle as a function of resistance. This is likely a linear regression, but perhaps another model will fit the data better.

3. Second you'll be doing some system identification. Take data of step responses to 3-5 pwm values (with 3-5 samples of each). Find constants K, D, α, and γ for each of these datasets and comment on both the residuals and any discrepancies. Some potential ideas to make this easier:

   - Find initial values for K and α by using a static model ($\ddot{q} = \dot{q} = 0$). This can be done with a linear regression rather than a nonlinear one.
   - If you can get useful pressure data, find γ independently using pressure as a (proportional) measurement for torque.
   - Try using a full second order model for τ if the coupled inertia and damping isn't accurate enough.
   - By looking at the measured data of the step response, you should be able to figure out if the system is underdamped or overdamped (though the addition of the coriolis matrix and torque dynamics might overly complicate this).
   - A potentially valid assumption is that the motor dynamics are faster than the finger dynamics.
   - Try using gridding to find initial conditions for D and γ, if running the regression is still hard.

4. Then you'll do the same identification, but you'll use the hyperelastic model rather than the standard elastic model. You should use the same data, and comment on any differences between this model and the the elastic model. Some potential ideas:

   - α and γ should theoretically not change between the two models
   - Static analysis will likely still be useful for finding C_1 and C_2, but you'll likely need more data points.

5. **For grad students:** Perform an additional experiment of your choice. You should provide your rationale for the experiment and the data you're collecting, state a reasoned hypothesis, and document your results using appropriate figures. Feel free to use the results of any of these experiments when doing analysis for the other tasks. Some possible ideas are:

   - Explore the use of the camera for capturing dynamic data. Perhaps using a Kalman filter to fuse camera and flex sensor data would be useful.
   - Add an amplifier to the pressure sensor to increase resolution at a desired pressure range (taking into account slew rate, stability, etc), and use it to better characterize the motor dynamics. If you do this you should use an additional breadboard and return the circuit to its default configuration whenever you aren't using it. If a group does this, other groups are free to use the amplifier as long as they credit the designing group (this will not count as a custom experiment for these other groups)
   - Compare the properties of EcoFlex-30 and DragonSkin-30 fingers.

Remember that this is the most research-based lab in an already research-oriented course. It's likely that not everything you try will work, and please remember that a negative or null result is still a result. We don't expect any group to work more than 20-30 hours on this, so if this becomes a problem please reach out to the GSIs. Also remember that you're doing nonlinear optimization in this lab, which can take a long time to run and can fail quite often. Take your data early so you have time to do the analysis properly.

4 Deliverables

To demonstrate that your implementation works, deliver the following:

1. Videos of your your data collection. Provide a link to your video in your report. You don't need a video of each data set, but we should see from your videos that you actually took data. Also, please combine all videos into one video file on upload.
2. Provide the code by adding "berkeley-ee106b" as a collaborator to a *private* github repo.
3. Submit a detailed report with the following:
   (a) Describe your approach and methods in detail, including any models, simulations, or cost functions you used, and all the tests you conducted.
   (b) Summarize your results in both words and with figures. In particular, show:
   - Provide plots, tables, and equations for your sensor calibration.
   - Provide results, discussion, and comparison for both the elastic and hyperelastic models.
   - Results from your custom experiment along with any accompanying figures or tables.
4. **BONUS:** In addition to your report, write a couple paragraphs detailing how the lab documentation and starter code could be improved. This course is evolving quickly, and we're always looking for feedback. This should be a separate section in the report, and only well-considered, thoughtful feedback will merit full credit.

5 Getting Started

We assume that you've already completed all the configuration steps from labs 0, 1, and 2. There are some different steps for lab 3.

5.1 Configuration and Workspace Setup

Note: *we expect all students to already be familiar with configuration procedures, either through EECS 106A, Lab 0, or some other place. Therefore, we won't be providing any explanations in this section. Please consult Lab 0 if you require a refresher*
Type the following commands:

```
cd ~/ros_workspaces
mkdir ~/ros_workspaces/lab4_ws
mkdir ~/ros_workspaces/lab4_ws/src
cd ~/ros_workspaces/lab4_ws/src
catkin_init_workspace
cd src
git clone https://github.com/chriscorrea14/lab4_pkg
cd ..
catkin_make
source devel/setup.bash
```

5.2 Working With Robots

For this lab, you'll be working with the soft finger setup. Before working with the soft robot, make sure the circuits are put together as expected. There are a couple of extraneous parts on the breadboard that you won't need. The circuits should look like figures 3 and 4. To connect to the soft robot, first turn on the power supply. It should look like figure 5. Press the "Output On/Off" button if the display says "OUTPUT OFF". Press the +25V button if the flashing number is not the one's digit. Rotate the dial until the display says 12V. If something goes wrong with the soft finger, press the "Output On/Off" button immediately.
Make sure the power supply is connected to the wires labeled "12V" and "GND". Then connect the Arduino board to the computer with the printer cable. Finally, you can start the Soft Gripper Interface with the following launch file:

```
roslaunch lab4_pkg soft_gripper_interface.launch
```

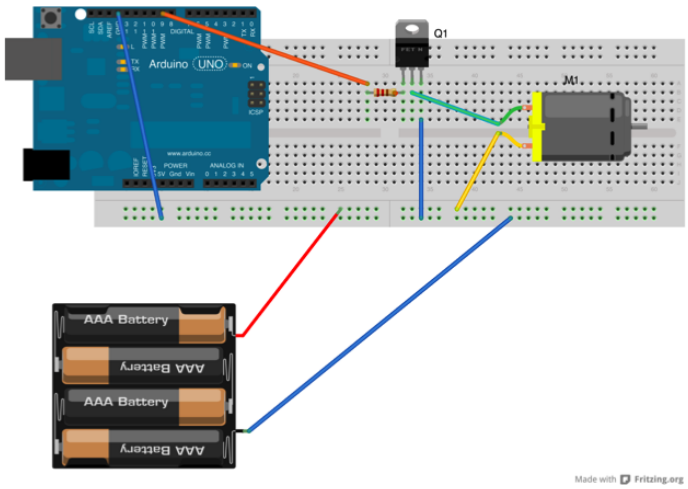

Figure 3: Pump Circuit

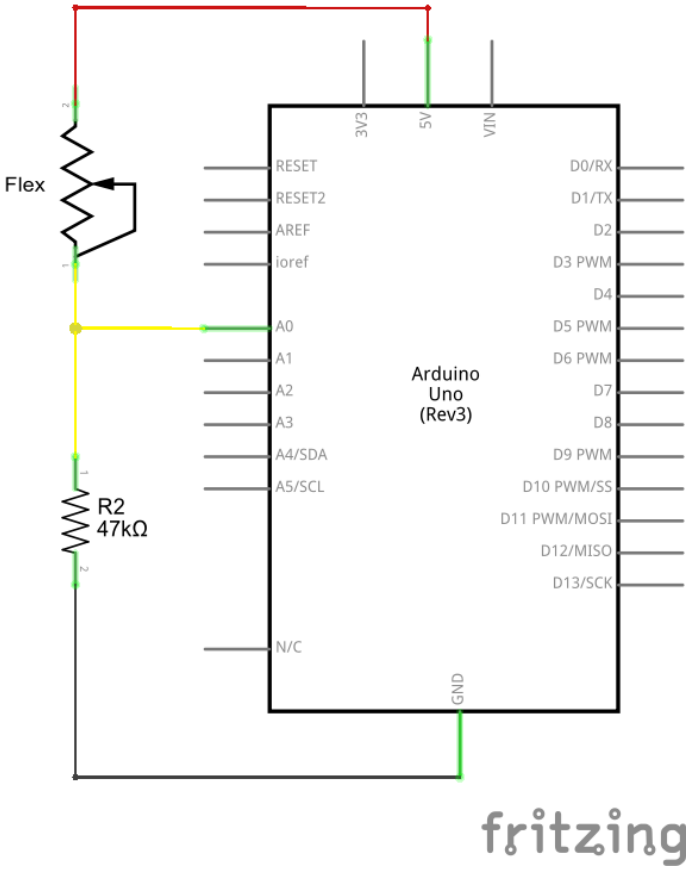

Figure 4: Flex Sensor Circuit

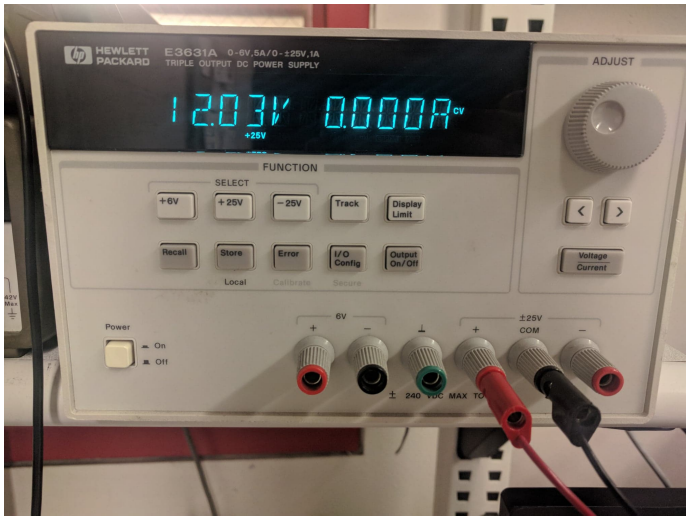

Figure 5: Power Supply.

This will open up the camera and start the node which listens to states and sends commands over Serial. If you have connection problems, apparently plugging in the printer cable into a different computer and then plugging it back into the original computer helps. Don't ask us why.

5.3 Starter Code

5.3.1 Data Collection

- `scripts/soft_gripper_interface.py` You should look through this file to make sure you know how it works, but hopefully you won't have to modify it at all. It publishes the soft gripper state to the `/soft_gripper_state` topic as a `SoftGripperState` message. If you're confused at what is in these messages look at `lab4.msg`. It also listens for `SoftGripperCmd` messages, which consist of the desired pump values for the left and right fingers. Note that all the code supports the use of two fingers, but we currently only use one. More info about how the state is calculated is listed below.
- `Arduino/motor/motor.ino` Hopefully you won't have to modify this file (for your own sake). This takes commands over serial represented as "⟨left⟩,⟨right⟩", such as "100, 75". It then returns back the state of the finger that the arduino can measure, also as comma separated values. You may find using the Serial Monitor useful for quick debugging of the Arduino code.
- `scripts/record_data.py` This script is what you'd be modifying the most. There are some examples of how to send commands to the finger in the `record_data()` method. It doesn't matter what you put for the left finger, since it is not connected. You only are controlling the right finger.

States: The state is represented by a couple things. We return the bend sensor analog output which you'll be converting to bend angle. We return the raw analog output from the pressure sensor readings, which you'll be converting to a pressure value in psi. Finally, we compute the homography between the 4 red circles on the backboard. We map these four points in pixel space to the x, y values in cm we measured in real life. We then project the blue circles to figure out the exact position of the base and tip of the soft finger. Remember that four-point homography assumes that all points lie on a plane, so try to situate the camera perfectly perpendicular to the sensor setup. The blue dots are quite close to the backboard, so hopefully small misalignments won't cause too many errors. If you are having problems with the homography calculation, there is commented out code in `soft_gripper_interface.py` to visualize the image with just the blue and red color (`plt.imshow(mask)`), to visualize the detected circles (`im_with_keypoints =`), and to display the projected image (`plt.imshow(new_image)` in `update_homography`). You probably won't need to use these, but in case you need to debug we left those commented out.

5.3.2 Data Analysis

- `matlab/finger_dynamics` Symbolically defines the dynamic equations for the finger and autogenerates some functions including elastic and hyper-elastic dynamics, inertia, gravity, and coriolis matrices, and forward

kinematics. You'll need to edit this if you want to change any models.

- `matlab/dhp`, `matlab/finger_transforms`, and `matlab/LagrangianDynamics` are all helper functions called by `finger_dynamics`.
- `tau_dynamics` is the second order dynamics of the motor. Change this file if you want to change the motor model.
- `find_tau` and `find_q` both do numerical integration of τ and q respectively using the `ode45` differential equation solver.
- `cost_function` simply computes and presents the error $q - \hat{q}$ and is called in the regression.
- `import_data` imports a csv file of the form returned by `record_data`. Autogenerated by matlab.
- `data_analysis` is the file you'll primarily be using. It's mostly empty, but features a least squares fit done on some pre-generated data to show you how to do it. Make sure to look at the documentation for `lsqnonlin`, as there are many potentially useful options.

6 Scoring

Table 1: Point Allocation for Lab 4

Section	**Points**
Video	3
Code	3
Methods	10
Results: Sensor Calibration	7
Results: Parameter Fitting	10
Results: Second Order Stiffness Model	7
Results: Custom Experiment	10*
Discussion of Results	10
Bonus Difficulties section	5*

Summing all this up, this mini-project will be out of 50 points, with an additional 15 points possible. For grad students, the mini-project will be out of 60 points, with an additional 5 points possible.

7 Submission

You'll submit your writeup(s) on Gradescope and your code by adding "berkeley-ee106b" as a collaborator to a github repo. Your code will be checked for plagiarism, so please submit your own work. Only one submission is needed per group. Please add your groupmates as collaborators on both Github and Gradescope.

8 Improvements

If you notice any typos or things in this document or the starter code which you think we should change, please let us know here. Next year's students will thank you.