Lab 1 Manual

Robotics & Automation
Dylan Losey, Virginia Tech

1 Introduction

Welcome to the first robotics lab! The GTAs are here to facilitate your learning and help with any technical challenges. Please be kind, professional, and respectful to the GTAs.

Objective. This lab has two objectives:

- Introducing the Geomagic Touch
- Calibrating the robot's joint position

Figure 1: Geomagic Touch at your workstation.

2 Degrees of Freedom

Geomagic Touch. Throughout the semester you will be working with the robot shown in Figure 1. You can think of this robot (the Geomagic Touch) as a table-top robot arm.

Question 1

Try picking up the pen and moving the robot arm. While you move the robot, answer the following questions with your team:

- How many joints does this robot have?
- What type of joints are they?
- How many degrees of freedom does this robot have?
- Is it a serial robot or a parallel robot?

3 Connecting to the Computer

During labs you will use MATLAB and Simulink to program your robot arm. Before we start programming the Geomagic Touch, we must check that the robot is connected to the computer and everything is running correctly.

Action 1

Perform the following checks:

- The computer licences are paired with specific Geomagic Touch robots. Confirm that your computer has the same number as your Geomagic Touch workstation.
- Make sure that the ethernet cable coming out of the Geomagic Touch is connected to the back of the computer, and the robot's charging cable is plugged in.
- Check that the computer is connected to the wifi "quanser_UVS" with password "UVS_wifi".

Hint. The computers are slow. Patience is a virtue. But if your connection to the robot is continually lagging throughout this lab, we recommend that you restart the computer.

Action 2

Now we are ready to initialize our robot arm. Open the Geomagic Touch Diagnostic Tool application shown in Figure 2. Then complete the following action items:

Figure 2: Application to check your connection to the Geomagic Touch.

- When you open the application, you will see a tab at the top called **Select**. Click on this tab to see a small 3D model of your robot (shown in Figure 3). When you move the actual robot this 3D model should move as well. *If the 3D model is not moving when you move the robot ask a GTA for help*.
- Once you've ensured that the robot is connected, click on the **Calibrate** tab. Follow all the instructions on the screen to calibrate the robot (i.e., plug the pen into the holder for several seconds until the calibrate button turns green).
- Make sure to close the Diagnostic Tool after calibration. Otherwise it will conflict with Simulink in later steps.

4 Reading the Joint Angles

Now that we have the robot connected, we can start to write programs for the robot. Our first program will read the robot's joint position.

Action 3

Let's open and run the initial simulink model for this lab.

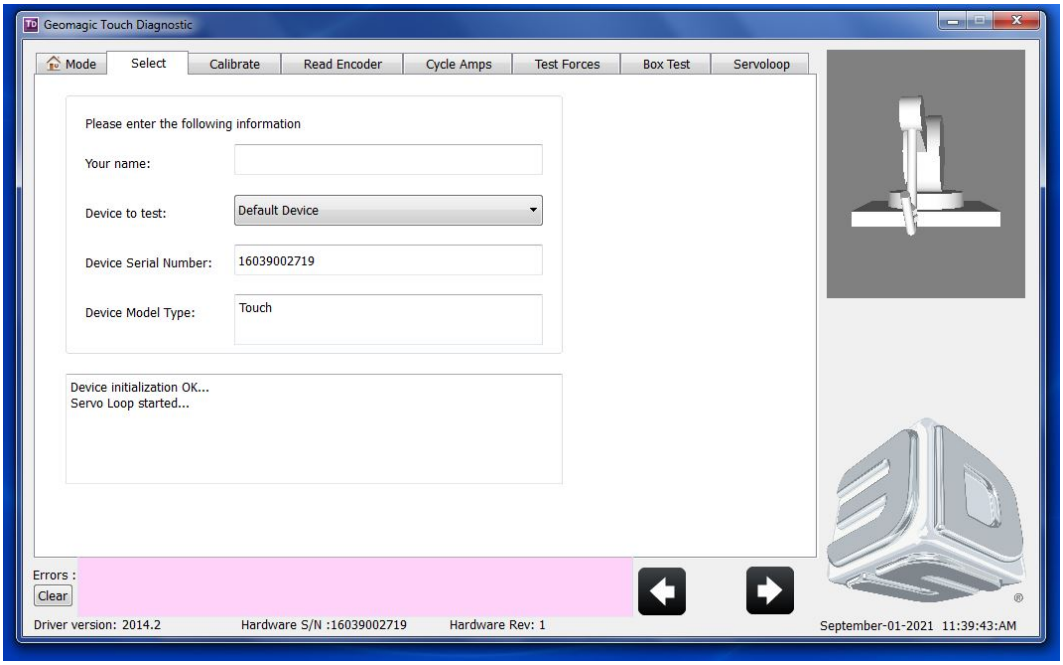

Figure 3: Geomagic Touch Diagnostic Tool. There is a 3D rendering of your robot arm in the top right. This should move in sync with your robot.

- In MATLAB, open the Joint Position Simulink model:
  `..\Desktop\01 Joint Position\Joint_Position_2015b_Start.mdl`
- You should see the Simulink diagram shown in Figure 4.

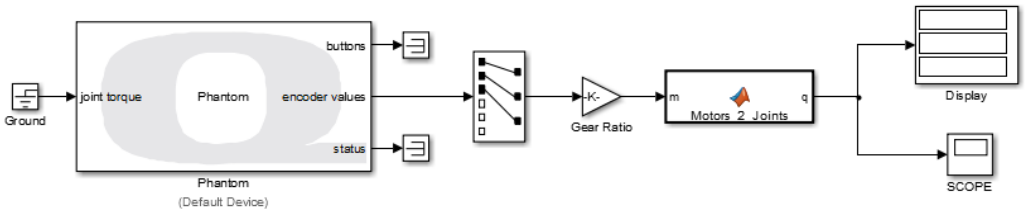

Figure 4: Initial Simulink diagram for the lab.

- Build the simulation by going to the QUARC menu and clicking on **Build**. When the MATLAB Command Window shows that the model has been downloaded to the target, run the simulation by opening the QUARC menu and clicking on **Start**.
- Open the **Scope**. Move around the robot arm. You will see a plot with 3 different lines — these lines show θ_1, θ_2, and θ_3 in real-time. You can also see these joint values directly on the **Display** block.

Question 2

As you move around the robot you notice that the joint values are very large. That's because we are currently reading the robot's encoders. The encoders on the robot's joints have 2048 counts. Write down the scalar that converts from counts to radians.

Action 4

Let's implement your solution to Question 2.

- Insert a **Gain** block between **Motors_2_Joints** block and the **Scope**. Enter the gain you found in Question 2, and check that the multiplication is "Element-wise." Name this block "Encoder."
- Your updated Simulink diagram should look like Figure 5.

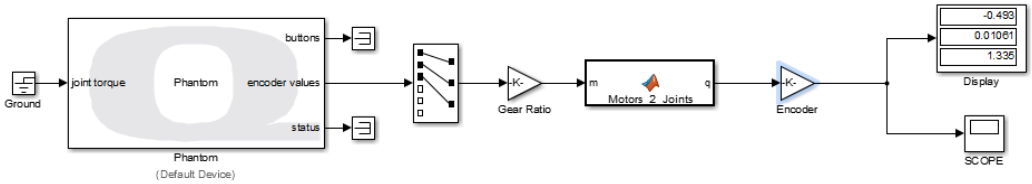

Figure 5: Updated diagram after converting from encoder counts to radians.

- Build and run the simulation using the same steps as before.
- Open the **Scope**. Move around the end-effector of the Geomagic Touch. You should now see joint values in radians — if you are getting values outside of the range $[-2\pi, 2\pi]$, then something is wrong.

5 Calibrating the Joint Angles

We have the joint positions in radians. However, we still need calibrate these joint positions to make sure they are aligned in the correct direction, and to account for any offsets.

Question 3

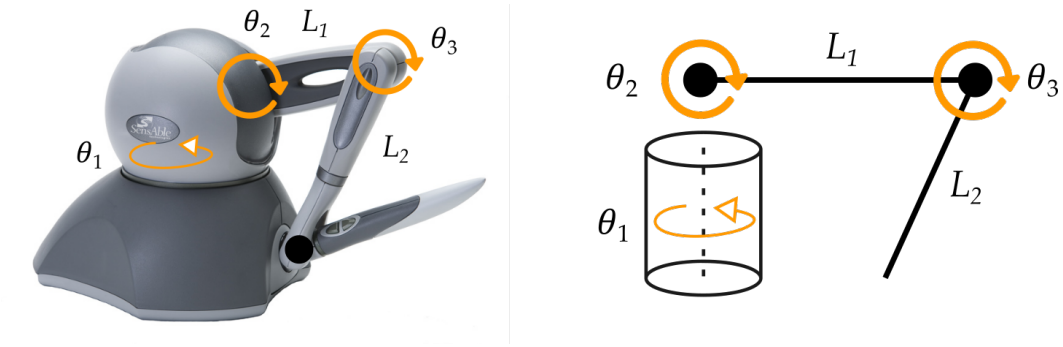

Figure 6: Joints of the Geomagic Touch. Arrows show positive direction of rotation (i.e., increasing joint position).

Starting with θ_1 try moving each joint in the *positive* direction defined by Figure 6. Does the corresponding signal on the scope also move in the positive direction? Or is it inverted? Record the relative direction (+/-) of each joint.

Action 5

Let's fix the joints so they are aligned in the correct direction.

- Insert a **Gain** block between the **Encoder** block and the **Scope**. Enter the three (+/−) gains you found in Question 3 as a vector. The joint angle calculation should look like what you see in Figure 7.
- Restart your model and open the scope. Each joint should now move according to positive orientation defined by Figure 6.

Question 4

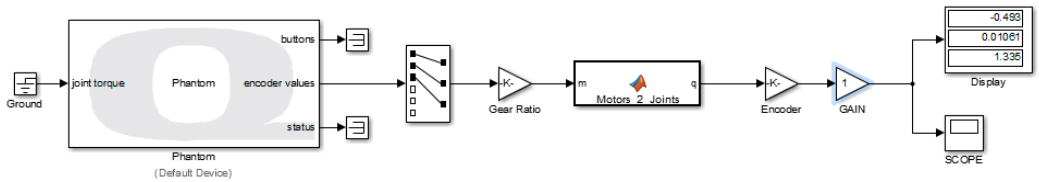

Figure 7: Updated diagram after aligning the joint directions.

We are almost calibrated. The last thing we need to do it check for any offset bias in the joint readings (i.e., reading $\theta_1 = 0.5$ when $\theta_1 = 0.0$). To check for this, place the robot's end-effector on **position 2** of the baseboard. Here the joint angles should read:

$$\theta_1 = 0.0 \quad \theta_2 = -0.3 \quad \theta_3 = 3.4 \tag{1}$$

Determine the biases (if any) that should be added to each joint in order to get these values.

Action 6

Let's fix the joint sensor to remove any bias in the joint readings.

- Insert a **Bias** block between the **Gain** block and the **Scope**. Enter the biases you found in Question 4 as a vector. Your diagram should look like Figure 8.

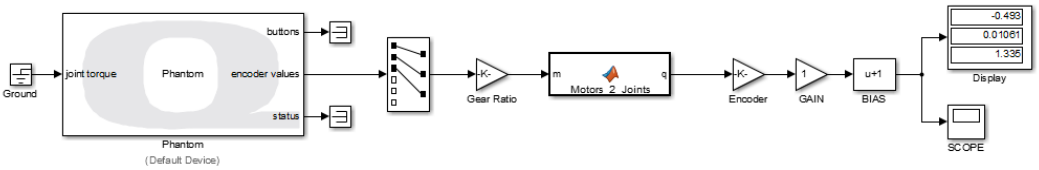

Figure 8: Updated diagram after adding a linear offset.

- Rebuild your model, start it, and open the **scope**.
- Place the robot's end-effector on **position 2** of the baseboard. The positions on the **scope** for each revolute joint should match the ones from Equation (1). If your positions do not match, either your **gain** or **bias** blocks are incorrect. Note — this won't be perfect, but try to get as close as you reasonably can.

Question 5

Show your calibrated robot to the GTA. Demonstrate that the joints are aligned in the correct direction and reading the appropriate values at **position 2**.

6 Finding the Robot's Joint Space

When we program the robot it's useful to know the minimum and maximum values for each joint. For example, imagine that we want the robot to reach $\theta_1 = 3\pi/4$. Before we tell the robot to go to that position, we should double check that this joint value is actually possible! In the final part of this lab you will measure the robot's joint space.

Question 6

While running the Simulink model you built, carefully move the robot as far as you can (*without pressing or pulling the robot too hard*). What is the maximum θ_1 value you can reach? What about the minimum θ_1 value? Separately looking at θ_1, θ_2, and θ_3, determine the minimum and maximum positions in radians.

Question 7

Within your team, brainstorm ways that you could increase the robot's joint space. Write down two of the solutions that you come up with.