Lab 4 Manual

Robotics & Automation
Dylan Losey, Virginia Tech

1 Introduction

Welcome to the fourth 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:

- Getting the Jacobian of your robot arm
- Comparing the output of your Jacobian to measured end-effector velocity

Figure 1: Geomagic Touch at your workstation.

2 Calibrating the Robot

Multiple groups use each workstation. Between sessions the robot needs to be re-calibrated. You must complete these steps at the start of *every* lab. On the plus side, you practiced these same steps during Lab 1, 2, &3, so you know what to do!

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.

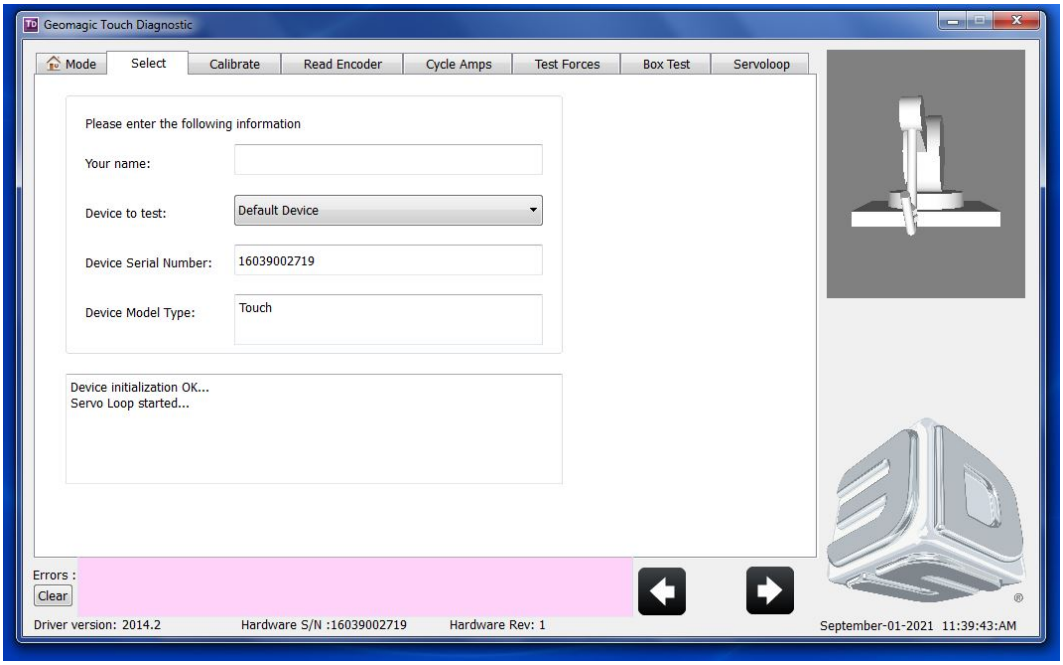

Figure 3: Geomagic Touch Diagnostic Tool. There is a 3D rendering of your robot arm in the top right. This should move in sync with your robot.

- 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.

Action 3

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

- In MATLAB, open the Forward Kinematics Simulink model:
  `..\Desktop\04 Jacobian\Jacobian_2015b_Start.mdl`
- You should see a Simulink diagram similar to Figure 4.

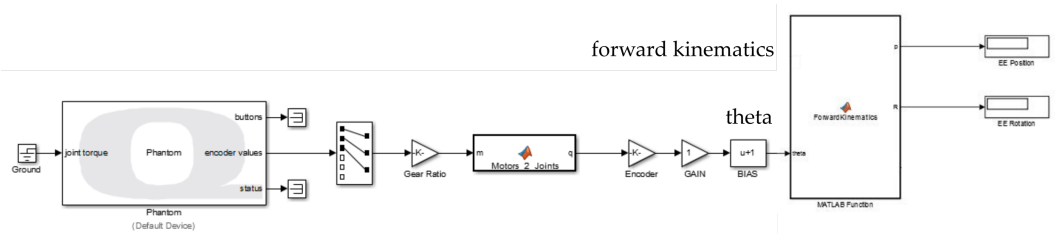

Figure 4: Initial Simulink diagram. Your team developed the forward kinematics in Lab 3.

- 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**.
- Place the robot's end-effector on **position 2** of the baseboard. The positions on the **display** for each revolute joint should read:

$$\theta_1 = 0.0 \quad \theta_2 = -0.3 \quad \theta_3 = 0.26 \tag{1}$$

  If your positions do not match, modify the **bias** block until the robot is correctly calibrated. You should also confirm that the forward kinematics function is correct. At **position 2**, the end-effector position p_{sb} should be close to:

$$p_x = 0.13 \quad p_y = 0 \quad p_z = -0.09 \tag{2}$$

  If the joint position is correct but the end-effector position is wrong, first double check the forward kinematics block, and then ask a GTA for assistance.

3 Finding the Jacobian

Question 1

Referring to Figure 5, find the space Jacobian for your robot arm. Your answer should be the matrix J_s as a function of θ (leave θ, L_1, and L_2 as variables).

Question 2

Write an equation that converts the space Jacobian J_s to the geometric Jacobian J. Then evaluate your equation to find J as a function of θ. Your answer should be the 6×3 geometric Jacobian for this robot arm (leave θ, L_1, and L_2 as variables).

Question 3

When the robot is in its home position ($\theta = 0$) we can move the end-effector along the y_s axis by actuating joint 1. Find a joint position θ where the robot **cannot** instantaneously move in the y_s axis no matter what we choose for $\dot{\theta}$. Your answer does not need to be reachable by the physical robot (i.e., it could be outside of the robot's joint space).

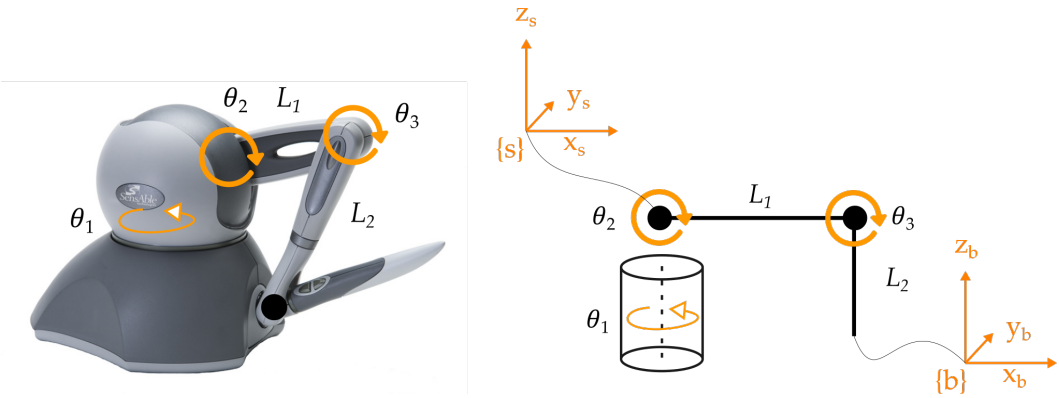

Figure 5: Joints of the Geomagic Touch. On right the robot is shown in its home position where $\theta_1 = \theta_2 = \theta_3 = 0$. Here $\{s\}$ is the fixed frame and $\{b\}$ is a coordinate frame at the end of link L_2. Both L_1 and L_2 are 0.132 meters in length.

4 Implementing the Jacobian

Action 4

Implement the geometric Jacobian J in your Simulink model. This block should input the joint position θ and output the 6×3 Jacobian matrix.

Action 5

We will use our geometric Jacobian to get the linear and angular velocity at the end-effector:

$$\begin{bmatrix} \omega_s \\ \dot{p} \end{bmatrix} = J(\theta)\dot{\theta} \tag{3}$$

You have θ. In this action you will change your Simulink model to also measure $\dot{\theta}$. When following these steps refer to Figure 6.

- We will obtain $\dot{\theta}$ by first using numerical differentiation, and then applying a low-pass filter to smooth the output.
- Insert a **Transfer Function** block into your model. Within the **Transfer Function** block, make the numerator $[1, 0]$ and the denominator $[0.01, 1]$. The numerator performs differentiation (s in the Laplace domain), and the denominator is a low-pass filter with time constant $\tau = 0.01$.
- Duplicate this **Transfer Function** block three times: the first will convert θ_1 to $\dot{\theta}_1$, the second will convert θ_2 to $\dot{\theta}_2$, and the third will convert θ_3 to $\dot{\theta}_3$.
- Add a **Demux** block to separate vector θ into the component scalars, and a **Mux** block to combine scalars into the vector $\dot{\theta}$.

Action 6

Using J and $\dot{\theta}$, implement Equation (3) in your Simulink model. Attach **display** blocks so you can visualize ω_s and $\dot{p}$ in real-time as your move the robot arm.

Question 4

Let's move the robot arm to make sure that the end-effector twist matches our intuition.

Which joints affect the end-effector's linear velocity in the x-axis? What about the y-axis and the z-axis? Independently move θ_1, θ_2, and θ_3. When you move the first joint, determine whether this has the largest affect on $\dot{p}_x$, $\dot{p}_y$, $\dot{p}_z$, or some combination. Try the same thing for joints two and three. Does your answer depend on θ?

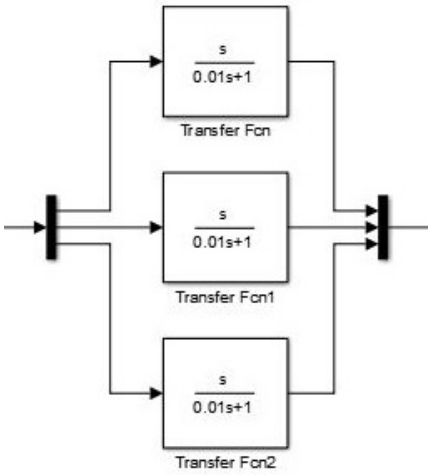

Figure 6: Section of your model for converting θ to $\dot{\theta}$. You will make a second copy of this to convert p to $\dot{p}$.

5 Comparing to Measured Velocity

The geometric Jacobian gives us linear velocity $\dot{p} = \dot{p}_{sb}$. You implemented forward kinematics to find p. If you perform numerical differentiation on p from your forward kinematics, you can get a measurement of $\dot{p}$. Does this measurement match your Jacobian output?

Action 7

Copy the blocks you made to convert θ to $\dot{\theta}$ (i.e., duplicate Figure 6). Use these blocks to convert p to $\dot{p}$, where p is the position vector output by the **forward kinematics** block. Attach a **display** to the output.

Question 5

The $\dot{p}$ from your Jacobian should match the $\dot{p}$ you measure by differentiating the forward kinematics. While moving the robot and keeping both scopes visible, show the GTA that you get approximately the same reading for both approaches.

6 Robot Design

For some robots (like surgical devices) we always want slow, precise motions. For other robots (like assembly line arms) we usually want fast, repeatable motions. What type of designs result in slower (or faster) serial robot arms?

Question 6

Refer back to the Jacobian J that you derived. Within your team, discuss how changing L_1, L_2, and the motors or gears would affect $\dot{p}$. Write down two things you could change to make the robot slower, and two things you could change to make the robot faster.