Lab 6 Manual

Robotics & Automation
Dylan Losey, Virginia Tech

1 Introduction

Welcome to the sixth 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:

- Understanding basic controllers for robot arms
- Implementing force control and PD control

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 Labs 1–5, 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.

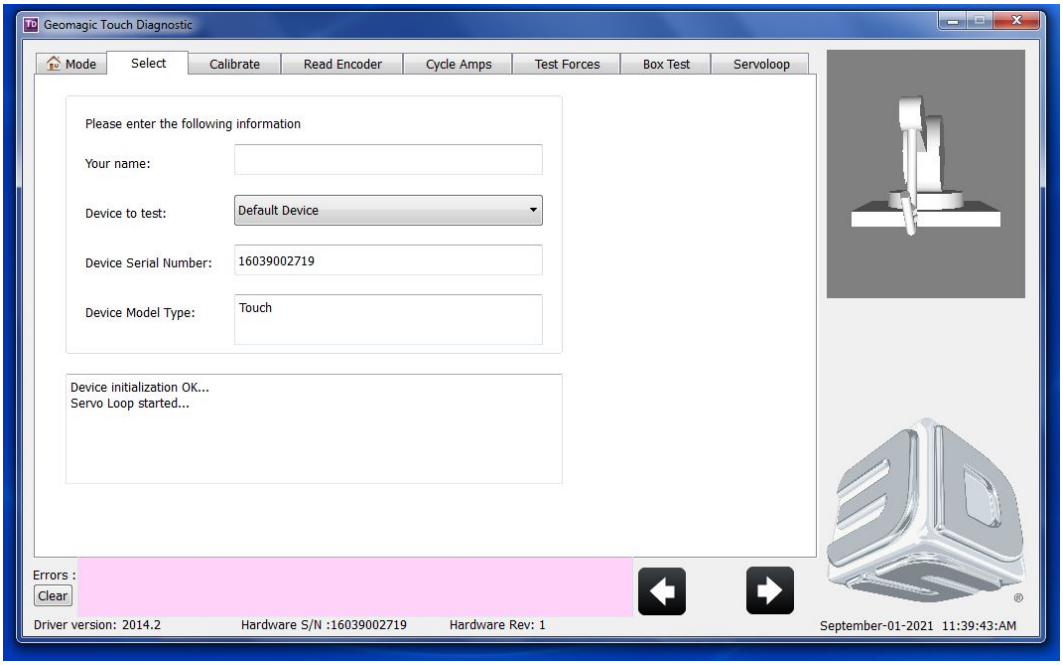

Figure 3: Geomagic Touch Diagnostic Tool. There is a 3D rendering of your robot arm in the top right. This should move in sync with your robot.

- 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 Control Simulink model:
  `..\Desktop\06 Control\Control_2015b_Start.mdl`
- You should see a Simulink diagram similar to Figure 4. This is the same model you started with for the last few labs.

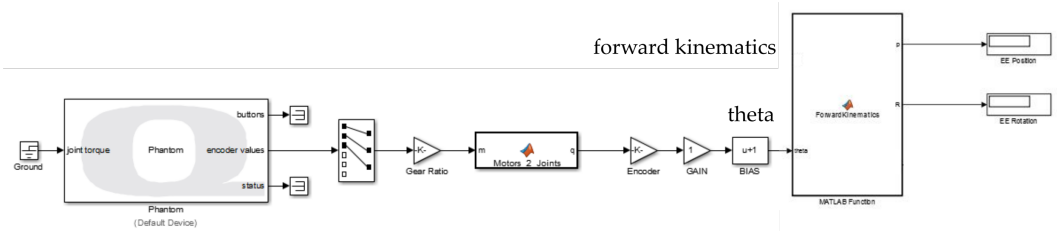

Figure 4: Initial Simulink diagram. Your team found the forward kinematics in Lab 3. Although not included in this diagram, you also found the geometric Jacobian in Lab 4. The geometric Jacobian is listed below in Equation (3).

- 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 Force Control

In the first part of this lab we will control the robot to apply *constant* forces and torques at the end effector. The concepts covered here are also related to statics.

Action 4

To get started, we will apply constant torques at each of the robot's joints. Look at the left side of the **Phantom** block; the input to this block is the commanded *joint torque*.

- Add a **Constant** block and connect its output to the **Phantom** joint torque.
- Here the joint torque is a 3×1 vector. The direction of torques is flipped: include a **Gain** block between the **Constant** and the joint torque. Set the **Gain** as -1.

- Add a **Unit Delay** block between the **Gain** and the **Phantom** joint torque. The **Unit Delay** block is needed to prevent an "algebraic loop" error in later steps.
- **Important:** *Do not apply torques larger than* ±3 Nm *at each of the joints. Attach a* ***Display*** *to the commanded torque so that you can keep an eye on this.*
- Set the **Constant** as a torque vector τ of your choice. *Make sure you hold the pen when you run the model!* You should feel the constant joint torques from the robot pushing against your hand. (**Hint**: You do not need to rebuild the model every time you change the value of a Constant block.)

Question 1

You are currently applying constant torques at each joint. But it is often useful to control the applied forces in end-effector space. Say you want to make the robot apply a wrench:

$$F = [0, 0, 0, f_x, f_y, f_z]^T$$

What equation would you use to convert this wrench to a joint torque? Throughout this lab you can assume all wrenches are in the intuitive geometric space.

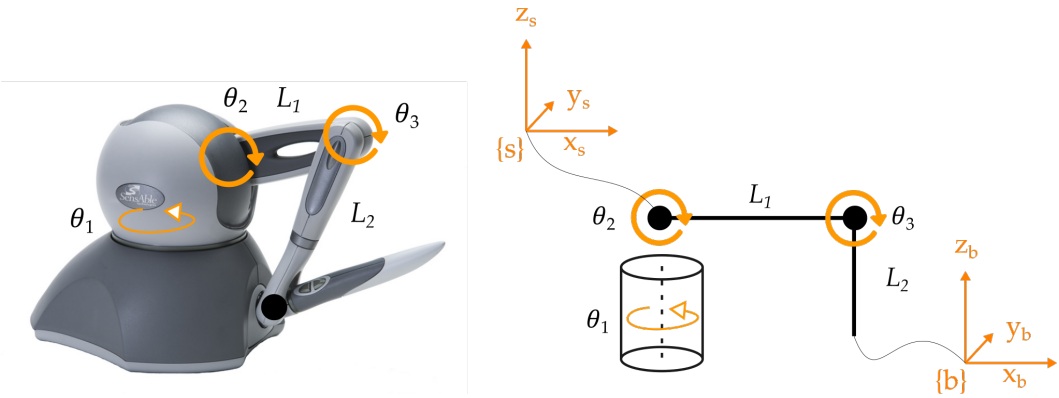

Figure 5: Joints of the Geomagic Touch. On right the robot is shown in its home position where $\theta_1 = \theta_2 = \theta_3 = 0$. Here $\{s\}$ is the fixed frame and $\{b\}$ is a coordinate frame at the end of link L_2. Both L_1 and L_2 are 0.132 meters in length.

Action 5

Below is the geometric Jacobian found in Lab 4. This Jacobian matrix is also provided for you in the **ForwardKinematics** block:

$$J(\theta) = \begin{bmatrix} 0 & -s_1 & -s_1 \\ 0 & c_1 & c_1 \\ 1 & 0 & 0 \\ s_1(L_2 s_{23} - L_1 c_2) & -c_1(L_2 c_{23} + L_1 s_2) & -L_2 c_{23} c_1 \\ -c_1(L_2 s_{23} - L_1 c_2) & -s_1(L_2 c_{23} + L_1 s_2) & -L_2 c_{23} s_1 \\ 0 & L_2 s_{23} - L_1 c_2 & L_2 s_{23} \end{bmatrix} \tag{3}$$

Use this Jacobian to implement your answer to ***Question 1***. Overall, you need to modify the block diagram so that you control the robot to apply commanded forces along the x, y, or z axes. How you do this is up to you. Below are some suggestions:

- Modify your Forward Kinematics function so that it also takes in a desired force vector $f_d = [f_x, f_y, f_z]^T$ and outputs the corresponding joint torques.
- Connect these commanded joint torques to the **Phantom** input. Remember to reverse the direction by multiplying by −1.
- Add a **Constant** block where you can set the desired $f_d = [f_x, f_y, f_z]^T$. This block should be an input to the modified forward kinematics function.

- **Important:** *Do not apply forces larger than* ±3 N *at each of the joints.*
- Try running the model with different values for f_x, f_y, and f_z. *Make sure you hold the pen when you run the model!* You should feel a constant force from the robot.

Question 2

Refer back to Figure 5. When you select a force of `[2; 0; 0]`, does this push in the positive x-axis? What about `[0; 2; 0]` and `[0; 0; 2]`? State whether or not the forces are aligned with the home axes. If not, describe any misalignment.

4 PD Control

Next you will control your robot to move to a desired position or track a reference trajectory. We will do this by implementing PD control in end-effector space.

Question 3

In lecture we discussed PD control in joint space:

$$\tau = K_P(\theta_d - \theta) - K_D\dot{\theta}$$

where K_P and K_D are 3×3 matrices of proportional gains and derivative gains, and θ_d is the joint position we want the robot to reach. Instead of reaching a joint position, what if we want to reach a specific (x, y, z) end-effector position? Start with controller:

$$f = K_P(p_d - p) - K_D\dot{p}$$

where $p_d \in \mathbb{R}^3$ is the target position we want to reach. Write an equation to convert this end-effector force into joint torques τ.

Action 6

Implement a PD controller in end-effector space. Set the desired position as:

$$p_d = \texttt{[0.15; 0.05; 0.005]}$$

and start with gain matrices:

$$K_P = \texttt{diag([30,30,30])} \quad K_D = \texttt{diag([0.5, 0.5, 0.5])}$$

As part of your implementation you will need to obtain $\dot{p}$. We get this using numerical differentiation (same process as Lab 4). Refer to Figure 6 and the steps below:

- 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 p_x to $\dot{p}_x$, the second will convert p_y to $\dot{p}_y$, and the third will convert p_z to $\dot{p}_z$.
- Add a **Demux** block to separate vector θ into the component scalars, and a **Mux** block to combine scalars into the vector $\dot{\theta}$.

Final Checks before Running: *Before running the model, double check all your signs! Getting these signs wrong could cause your system to go unstable.* Put the end-effector tip on Position 1. Try running the model (**hold the pen the first time you run the system**). The robot should move the pen out of the plate and into the desired position.

Question 4

While your Simulink model is running, record the position of the end-effector after the robot has stabilized. Does p correspond to p_d along the x, y, and z axes? If you notice a

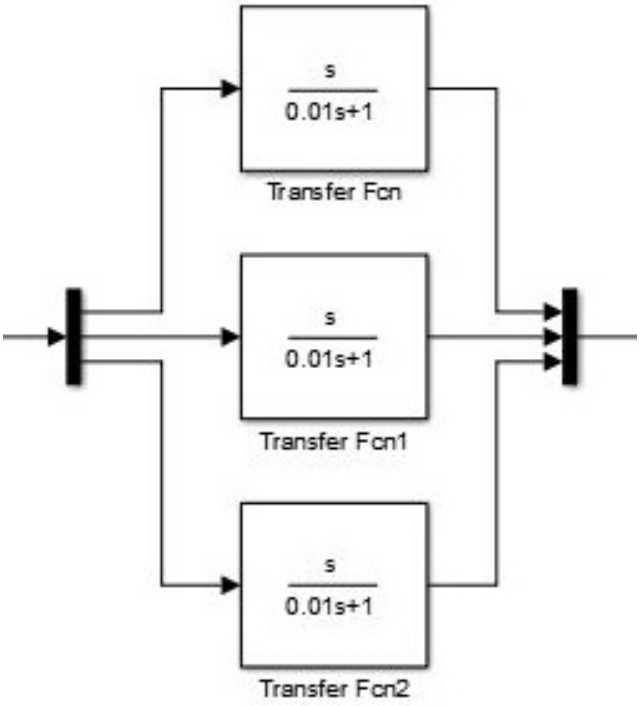

Figure 6: Section of your model for converting p to $\dot{p}$.

steady state error along one of these axes, explain why — i.e., is there anything you are not compensating for?

Question 5

Tune K_P and K_D to get the best performance you can. Do not make the gains unreasonably high (i.e., no proportional gain should exceed 50). Try pushing and pulling on the robot arm as you tune these gains, and observe how the robot responds to your external disturbances for different choices of K_P and K_D. *Write down* your best performing gains.

Action 7

Using your working gains, control the robot to move in a circle. The desired radius and frequency of this circle are left up to your team. Consider using $\cos(t)$ and $\sin(t)$ to set the desired end-effector position $p_d(t)$. Show your final system to a GTA.

Hint. You should insert a **Clock** to involve t in the system. Be careful when selecting the radius, it should be within the scale of the baseboard.