Lab 5 Manual

Robotics & Automation
Dylan Losey, Virginia Tech

1 Introduction

Welcome to the fifth 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 three objectives:

- Solving for the numerical inverse kinematics
- Comparing the inverse kinematics to measured joint positions
- Understanding the challenges of inverse kinematics

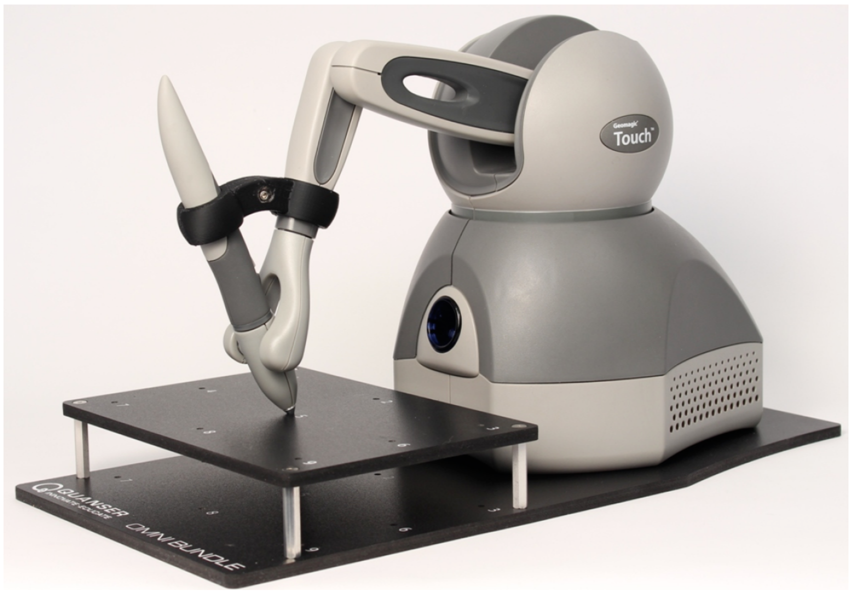


Figure 1: Geomagic Touch at your workstation.

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 1–4, 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:

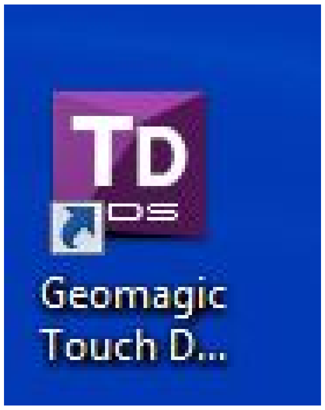
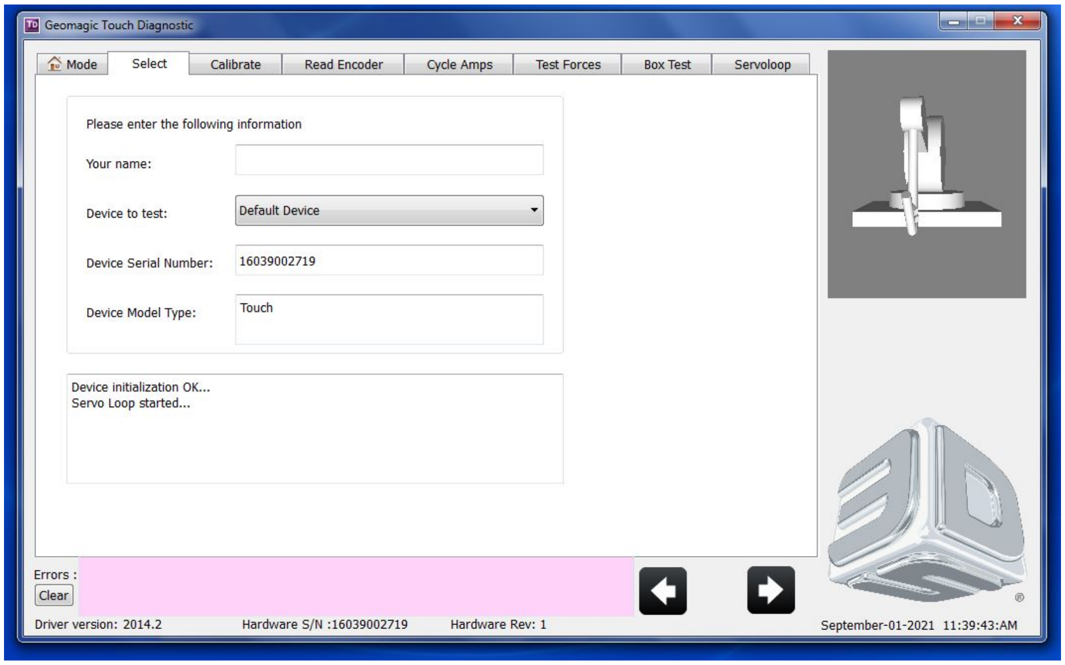


Figure 2: Application to check your connection to the Geomagic Touch.



Please enter the following information

Your name:

Device to test: Default Device

Device Serial Number: 16039002719

Device Model Type: Touch

```
Device initialization OK...
Servo Loop started...
```

Driver version: 2014.2 Hardware S/N :16039002719 Hardware Rev: 1 September-01-2021 11:39:43:AM

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\05 Inverse Kinematics\Inverse_Kinematics_2015b_Start.mdl`
- You should see a Simulink diagram similar to Figure 4. This is the same model you started with for Lab 4.

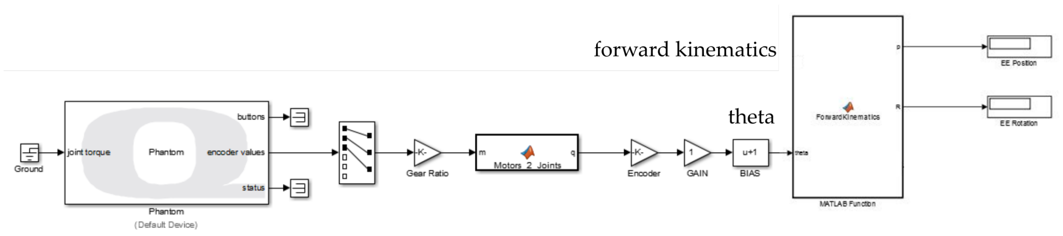


Figure 4: Initial Simulink diagram. Your team found the forward kinematics in Lab 3. Although not included in this diagram, you also found the geometric Jacobian in Lab 4. The geometric Jacobian is listed below in Equation (3).

- 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 Numerical Inverse Kinematics

Here you will update the Simulink model to compute the numerical inverse kinematics. Given just the end-effector's position p_{sb}, your goal is to get the corresponding joint position θ using the algorithm we outlined in lecture. **Do not use** the orientation of the end-effector for these numerical inverse kinematics.

Overall, you will be mapping $p_{sb} = [x, y, z]^T$ to the joint positions $\theta = [\theta_1, \theta_2, \theta_3]^T$.

Question 1

In lecture we leveraged $X \in \mathbb{R}^{6\times1}$ and $X_d \in \mathbb{R}^{6\times1}$ to represent the end-effector's orientation and position. Now that we are only using position, how do these terms change? Your modified answers for X and X_d should be 3×1 vectors.

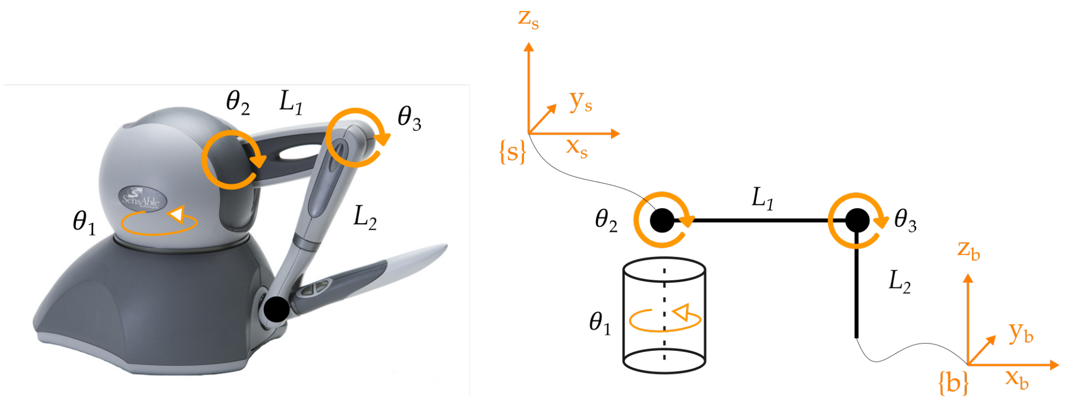


Figure 5: Joints of the Geomagic Touch. On right the robot is shown in its home position where $\theta_1 = \theta_2 = \theta_3 = 0$. Here $\{s\}$ is the fixed frame and $\{b\}$ is a coordinate frame at the end of link L_2. Both L_1 and L_2 are 0.132 meters in length.

Question 2

Below is the geometric Jacobian you previously found in Lab 4. As a reference, the robot arm is shown in Figure 5.

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

If $X_d - X$ is the error between the desired and actual end-effector position, what part of the Jacobian J would you use to map this error into changes in joint position $\Delta\theta$?

Action 4

Implement numerical inverse kinematics in your Simulink model. Your code should input the end-effector position p_{sb} and an initial guess at the solution θ_0. The block should output the 3×1 vector of joint positions θ. The initial solution θ_0 should be from a **Constant** block (we will tune this later in the lab). For now, set $\theta_0 = [0, 0, 0]^T$.

Attach a **display** block to the output so you can see the solution for θ in real-time.

Question 3

The θ that you solve for using numerical inverse kinematics should match the actual joint angles θ of the robot arm. While moving the robot and keeping both displays visible, show a GTA that your inverse kinematics solution matches the actual joint position.

4 Challenges of Inverse Kinematics

The inverse kinematics of serial robot arms brings a unique set of challenges. In this part of the lab you will explore some of these challenges.

Question 4

Move the robot's end-effector to each of the marked positions on the board. At each position, record the error e between the inverse kinematics solution and the actual joint position:

$$e = \|\theta_{actual} - \theta_{inverse\_kinematics}\|^2 \tag{4}$$

Question 5

Ideally we should have no error between the actual joint position and the inverse kinematics solution. But if there is an error, what could be the cause? Within your team, brainstorm **three things** that could contribute to discrepancies between θ_{actual} and $\theta_{inverse\_kinematics}$.

Question 6

Choose an end-effector position p_{sb}. Using Figure 5 as a reference, draw the robot arm in **two different joint positions** that both reach p_{sb}.

Question 7

Different choices of θ_0 may cause the robot to output different answers for $\theta_{inverse\_kinematics}$. Place the robot's end-effector in **position 2** on the baseboard and then set θ_0 to each of the values listed below. Determine whether that choice of θ_0 results in a different solution than the one you obtained with $\theta_0 = [0, 0, 0]^T$.

- Set $\theta_0 = [2\pi, 0, 0]^T$
- Set $\theta_0 = [0, 0, -\pi/4]^T$
- Set $\theta_0 = [0, 0, \pi]^T$

Action 5

In practice, we want to obtain inverse kinematics solution as quickly and accurately as possible. Tune your numerical inverse kinematics code to **minimize** the number of iterations it takes to find a solution. There is no one right answer to this question: your goal is to understand how each parameter trades-off between precision and speed.

- Add a second output to the Inverse Kinematics block that displays the total number of iterations.
- Tune the parameters for step size and termination condition.
- List your optimal parameters on the answer sheet.