Lab 2: Jacobians and Full Velocity Kinematics

Pre Lab Due Date: Monday 10/07/2024 @ 11:59pm
Code Due Date: Thursday 10/10/2024 @ 11:59pm
Report Due Date: Wednesday 10/16/2024 @ 11:59pm
Robot Lab Time: Friday 10/11/2024 and Monday 10/14/2024

This assignment consists of a written Prelab (due one week from assignment release), an individual coding assignment, and a partner report and robot lab. You will submit all parts of the assignment through Gradescope. Late submissions will be accepted per the standard late submission policy, but they will be penalized by 25% for each partial or full day late. After the late deadline, no further assignments may be submitted; post a private message on Ed Discussion to request an extension if you need one due to a special situation such as illness.

The Prelab is worth 5 points, and the report and code are each worth 20 points. The robot lab portion is included in the report grade. You may talk with other students about this assignment, ask the teaching team questions, use a calculator and other tools, and consult outside sources such as the Internet. **If you make use of outside sources, you are expected to cite them in your report.** When you get stuck, post a question on Ed Discussion or go to office hours!

1 Prelab (individual, due 10/07/2024 @ 11:59pm)

This lab is about Jacobians and velocity IK. For now, we'll start by focusing on the underlying theory.

1.1 Your Task

1. Derive the forward velocity kinematics for the Panda arm for when you are tracking the end effector. You do not have to do all of the computations by hand, but you should explain your steps and your final result. Your submission should include a copy of any code, MATLAB calculations, Mathematica notebooks, etc., that you write for this step if you choose not to do computations by hand.

2. Do a sanity check: Let's say that the Panda arm starts in the zero configuration and only one of the joints moves. What do you expect the corresponding velocity of the end effector to be? Does your FK reflect this velocity?

3. Let's start thinking about the inverse velocity kinematics problem. Under what conditions will the velocity IK problem for the Panda arm have no solution? One unique solution? Infinite solutions?

4. Read through the instructions for the Code and Lab assignments and develop a preliminary testing plan for your lab.

Submit the Prelab on Gradescope. You must complete and submit your prelab assignment individually, and your writeup should be clear and concise.

2 Code (individual, due 10/10/2024 @ 11:59pm)

In this part of the lab, you will implement the velocity FK and IK for the Panda arm.

2.1 Update Your Simulation Setup

To update your own private fork of the `meam520_labs` repo for Lab2 please do the following:

```
$ cd ~/meam520_ws/src/meam520_labs/
$ git pull upstream main
```

This is going to look at the public TA repository and grab the code stubs and launch files that we've added since the last lab. Occasionally when running this command, you might encounter a merge conflict. We don't expect that to occur, but if it does, don't hesitate to reach out to the TAs for help.

2.2 Your Task: Coding Implementation

1. You will need to implement your Jacobian calculation in `calcJacobian.py`. Open the `lib` folder using your preferred text editor, for example:

   ```
   atom ~/meam520_ws/src/meam520_labs/lib/
   ```

   and implement the `calcJacobian` function. The required inputs and outputs are outlined in the file. You are free to define other functions to modularize your solution, to help with this we have provided two function outlines in `calculateFK.py` for you to implement and use with `calcJacobian`. It is important that your solution run relatively quickly. With this in mind, your final solution cannot use the python library sympy. You are welcome to use it for testing but this package is not included in the autograder and will cause issues with your submitted code.

2. Using your `calcJacobian`, implement forward velocity kinematics in the `FK_velocity` method in the file `FK_velocity.py` located in the ∼`/meam520_ws/src/meam520_labs/lib/` directory.

3. **Linear IK velocity:** For implementing the inverse velocity kinematics, we will focus on the linear part first. Using your `calcJacobian`, implement inverse velocity kinematics in the `IK_velocity` method in the file `IK_velocity.py` located in the ∼`/meam520_ws/src/meam520_labs/lib/` directory. (Hint: you might want to look up Python's existing `numpy.linalg.lstsq` method.)

   If the velocity IK problem has **no solutions**, i.e. there exists no joint velocity which will exactly produce the given target velocity, then you should choose the joint velocity which minimizes the l^2 norm of the error between the target velocity and the achieved velocity. The term for this sort of solution is the *least squares solution*.

   If the velocity IK problem has **infinite solutions**, i.e. many choices of joint velocity will exactly produce the target end effector velocity, then you should minimize the l^2 norm of the joint velocities. The term for this sort of solution is also the least squares solution.

   Note that instead of a decimal number, some entries of both the target linear and angular velocities may be `nan`, this stands for Not a Number. This input means that the particular coordinate of the end effector velocity should be left unconstrained, meaning it can be anything. For example, calling

   ```
   IK_velocity(q,v,np.array([np.nan,np.nan,np.nan]))
   ```

   would constrain the linear velocity to be `v` while leaving the angular velocity completely unconstrained.

4. **Angular IK velocity:** By setting the `np.nan` in the `IK_velocity` function call, you are ignoring the angular velocity tracking command and only tracking the linear velocity (This is a good starting point for testing the linear part!). In order to perform angular velocity tracking, an extra step is needed to calculate the desired angular velocity. You will implement the function `calcAngDiff()` in `calcAngDiff.py`. This function calculates the target angular velocity given the current and the target end-effector orientations. Note that the function takes in rotation matrices as input and outputs the rotation vector that takes the current orientation to the target orientation (**Hint: Simply doing subtraction on two rotation matrices won't work. Think more carefully about how to find the difference in rotation matrices**). You may find it useful to know that given a single rotation R, the corresponding axis of rotation with magnitude $\sin\theta$ can be found by first computing the skew symmetric part of R as

   $$S = \frac{1}{2}(R - R^T)$$

   and then extracting the coefficients of the corresponding vector under the skew operation. Explicitly, since S is skew symmetric, $S = [a]_\times$ for some a, where $a \times b = [a]_\times b$ and $\times$ is the cross product in $\mathbb{R}^3$. Since

   $$[a]_\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$$

   one can easily extract the vector coefficients, such that a points in the direction of the axis of rotation and has magnitude $\sin\theta$. What remains is to ensure you are considering the *relative rotation between the target and current frames*, but to ultimately **express the axis relative to the world frame** to comply with the function's defined behavior.

   $$\vec{\omega}^0_{1,2} = R^0_1 \vec{\omega}^1_{1,2}$$

   where R^0_1 is the current orientation. **Hint:** pay close attention to your signs and coordinate frames in this function!

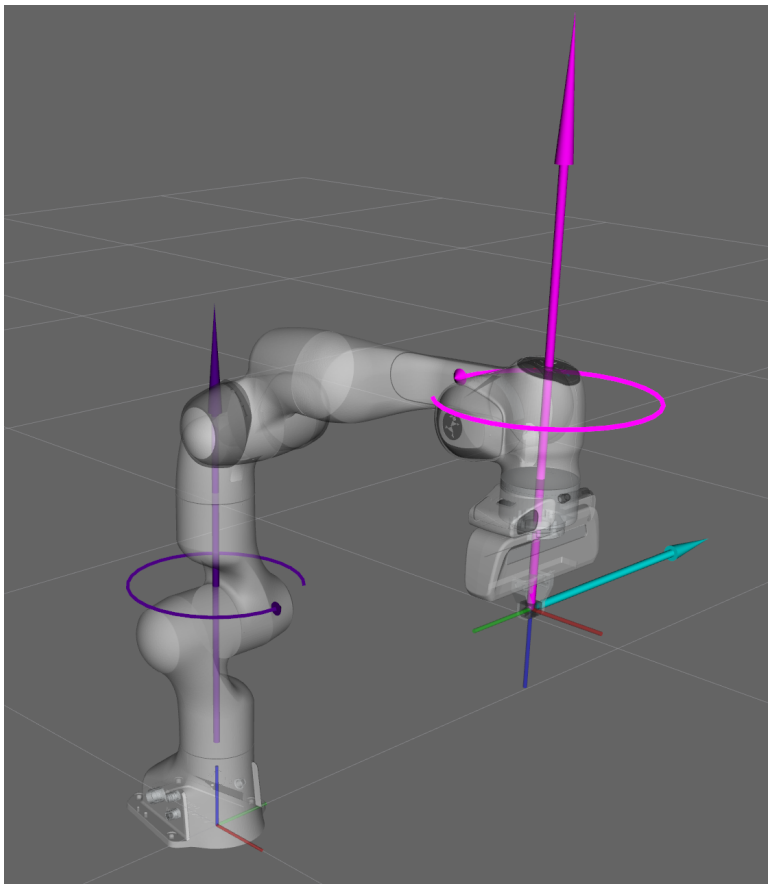

Figure 1: The visualization generated by the `visualize.py` script, showing the end effector linear velocity (in cyan) and angular velocity (in magenta) when moving a specific joint (in purple) at unit velocity for this configuration.

2.3 Provided Testing Scripts

We have provided the following scripts that you can use to help debug your code:

- `~/meam520_ws/src/meam520_labs/labs/lab2/visualize.py`

  This script will step through several configurations for the robot. For each, it will allow you to iterate through the joints, and moving only that joint at unit velocity. In RViz, the script will visualize the linear velocity (in cyan) and angular velocity (in magenta) of the end effector, as seen in Fig. 1.

- `~/meam520_ws/src/meam520_labs/labs/lab2/follow.py` This script contains several demos (some of which you will need to implement yourself). Each demo has the end effector of the Panda arm follow a periodic trajectory. All trajectories are centered at the end effector position corresponding to the neutral configuration of the robot. The trajectories are:

  - `ellipse`: an ellipse in the y-z with respective semiaxes `ry` and `rz`, traversed with a frequency `f`. The orientation is fixed. You will need to implement both the linear and angular trajectory tracking for this part. The linear velocity trajectory should follow the ellipse shape. Please design an orientation trajectory for the angular part. You can be creative for this part.
  - `line`: a line of length `L` in the z direction, traversed in an sinusoidal oscillatory pattern with angular frequency `f`. The end effector will rotate around the x-axis in a sinusoidal oscillatory pattern during the line motion. An orientation trajectory example is given. You will need to implement the linear trajectory for following the line.
  - `eight`: a figure 8 in the x-y plane. The trajectory is computed using a Lissajous curve. The orientation is fixed. The linear part of the "eight" trajectory is given. Please design an orientation trajectory for the angular part. You can be creative for this part.

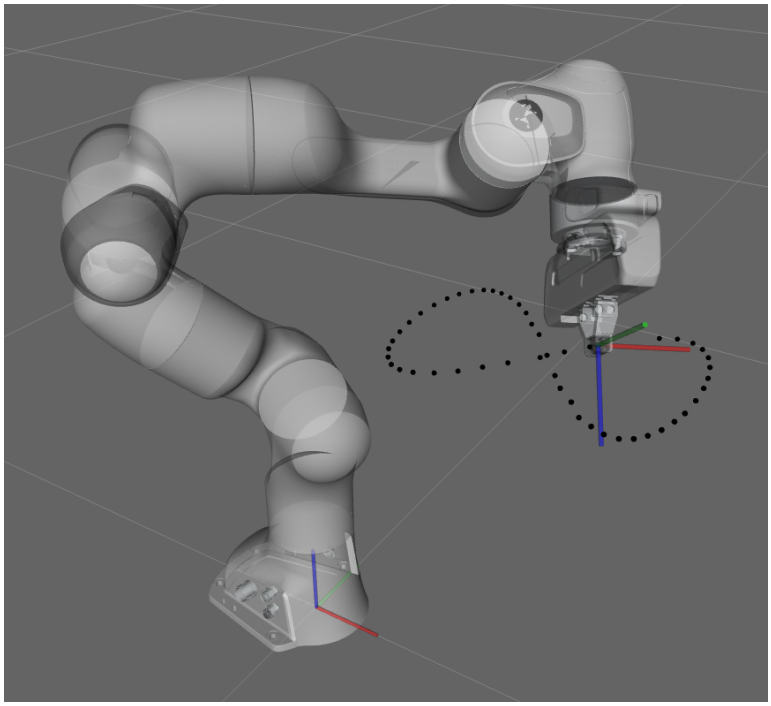

Figure 2: The robot tracking the `eight` trajectory using the `follow.py` script

The `eight` trajectory has already been implemented for you, while the `ellipse` and `line` demos have not. You can find stub code in the `JacobianDemo` class in `follow.py`. Their inputs are the current time `t` and some shape parameters, and the output is the desired position and velocity for the end effector at that instant.

The simulator tracking the `eight` trajectory can be seen in Fig. 2.

To do the testing:

1. In a terminal, type the following:

```
$ cd meam520_ws
$ roslaunch meam520_labs lab2.launch
```

2. In a second terminal do the following:

```
$ cd meam520_ws/src/meam520_labs/labs/lab2
$ python <insert file name>.py
```

2.4 Submit

Each student should submit their individual code. Remember that all coding assignments will be run through an originality checker, so copying code from your partner, another student, or anywhere else is not permitted. Please submit `calcJacobian.py`, `FK_velocity.py`, and `IK_velocity.py`, along with any other files needed to run your code to the Lab2 Code assignment on Gradescope. Once submitted, we will run automated tests to grade your code. You may submit as many times as you want, but any attempts to try to figure out what test cases we are running will result in your code getting a 0.

3 Lab (pairs, due 10/16/2024 @ 11:59pm)

3.1 Your Task

In pairs, compare your code against the results of simulation and hardware. If both partners have working code, you only need to test on one of the versions (specify whose in the report).

3.2 Robot Lab

Throughout the lab, always keep in mind the 2 rules of robotics:

1. A system that does not work in simulation will not work in hardware.
2. A system that worked in simulation may not work in hardware.

Preparing for Lab Before you come to robot lab, you should complete the following:

- Ensure that at least one partner's code works in simulation with no bugs and decide whose code you will test.
- Bring a USB drive with all files you need to run your code.
- Prepare an evaluation plan for the robot lab. Remember that your time is limited to 30 minutes. Planning in advance will allow you to make the most of that time.
- Come to lab in standard lab attire: closed toed shoes, long pants, and hair tied back. Our lab space tends to be cold, so you may want to bring a jacket or coat.

Getting your `lib` folder onto the robot computer You will need to get your code onto the lab computers. To do this you should bring the code on a USB drive. Once you have the code on the computer, copy the code to the following folder: `~/meam520_ws/src/meam520_labs/lib`. *(Feel free to overwrite any existing files in that folder.)*

Testing Your Code The task here is to observe how your implementation works on the robot. To do this, we will use code from the lab, and we will also ask you to think about what test cases you should investigate. Some questions to ask yourself include:

- Are the test cases safe? Will they lead the robot to have self-collision or collision with the obstacles (e.g. tables)?
- Are those test cases sufficient for you to judge the correctness of your code?
- How will you evaluate the correctness of your code? Are you looking at any specific metrics?

Discuss with your teammate what good test cases might be. When you are ready, you are going to execute the following code. We will resume using the simulation and then robot pipeline.

Robot Lab Workflow To test the code, we will be using a two part workflow. First you will test everything in simulation and then you will test it on the robot.

STEP 1: VERIFY IN SIMULATION

1. In a terminal, type the following:

```
$ cd meam520_ws
$ roslaunch meam520_labs lab2.launch
```

2. In a second terminal do the following:

```
$ cd meam520_ws/src/meam520_labs/labs/lab2
$ python <insert file name>.py
```

Call a TA over and have them verify that your code works in simulation before you attempt to run it on hardware.

STEP 2: TEST ON HARDWARE
If the code works in simulation, then you can proceed testing it on hardware.

1. Make sure all of your teammates are clear of the robot workspace
2. Release the software button (black button) which will turn the light on the robot blue (if this step is not done before launching ROS then the robot will not move)

3. NOTE: If you are using the same two terminals as from your software stop, you do not need to perform the `./franka.sh master` step. This sets environmental variables once in the terminal and it persists as long as the terminal is open. If you closed those terminals then open two terminals and complete the next steps completely.

4. In the first terminal do the following:

```
$ cd meam520_ws
$ ./franka.sh master
$ roslaunch franka_interface interface.launch
```

5. In the second terminal do the following:

```
$ cd meam520_ws/src/meam520_labs/labs/lab#
$ python <insert file name>.py
```

6. When the script has finished executing, enable the software stop again (black button); the lights on the robot will turn white.

Collect any data that you need for your report. Once you have finished, delete your files from the `~/meam520_ws/src/meam520_labs/lib` folder.

3.3 Report Contents

Submit a written report describing your work in this lab. **The report must be no more than 8 pages.** You are free to use whatever typesetting workflow you are comfortable with. Handwritten submissions will not be accepted. If you need to sketch some diagrams or write very complex equations by hand, it is okay as long as your photograph/scan is clear and readable. Your report should include the following sections:

Methods Describe your strategy for computing the velocity kinematics for the Panda robot arm. This should be an expanded version of your pre-lab. Include in this section:

- Any relevant diagrams, measurements, or parameters used in your approach.
- Any relevant equations, theory, or pseudocode that will clarify your explanation.
- Whose code you used for experiments and a short (1 paragraph) summary of your code structure.

Evaluation Test your code extensively in simulation and on the hardware to ensure correctness. In this section, document your testing process: how can you be confident that your solutions are free of bugs? Report all relevant data to support your discussion. Questions to consider may include:

- Start in the zero configuration, assume only one joint moves. What do you expect the corresponding velocity of the end effector to be? Does your FK reflect this velocity?
- Using geometric intuition, are there singularities and, if so, where are they? Are these configurations reflected in your mathematical expressions?
- How well does your end effector track trajectories (which ones might you care about) when only position tracking is required? What if you also want to control orientation?

Analysis Discuss the Panda forward and inverse velocity kinematics in the context of your data and observations. Do your results make sense? What movements is the robot good at? What movements is it bad at? Under what circumstances might you want to use velocity control over position control (or vice versa)?

3.3.1 Submit

Each group should only submit once to Gradescope per Gradescope assignment. While going through the submission please ensure that you **add your group members to the submission**. Failure to do so could impact the grades of all parties involved. Submit your written report as a PDF to the Lab2 Report assignment on Gradescope.

IMPORTANT, please submit the correct thing to each assignment. Don't submit the code to the report assignment. Don't submit the report to the code assignment.