Advanced Robotic Manipulation (CS 327A) Homework #2

(Spring 2022/2023) Due: **Friday, April 28, 11:59pm**

In this homework you will implement an Operational Space Trajectory Tracking Controller on the KUKA-iiwa arm in simulation. The KUKA-iiwa is a 7 DOF torque controlled robot manipulator and, therefore, your controller will directly output joint torques. You will also explore how both joint damping and operational space dynamic compensation improve controller performance.

Code Update

You can find the new homework code by navigating to the cs327a directory in your computer and following the steps below:

1. Stash your changes and pull the new updates (which will include the solutions) by using the following commands:

```
$ git stash
$ git pull
```

2. Retrieve your own changes (that you previously stashed) by typing:

```
$ git stash pop
```

3. Build the updated repo by navigating to cs327a_2021/build and running:

```
$ cmake .. && make
```

4. The new code is available in the hw3 folder

Problem 1 - Trajectory and Controller Setup

The desired end effector position trajectory to be followed is represented by

$$x_d = 0$$
$$y_d = 0.5 + 0.1 \cos \frac{2\pi t}{5}$$
$$z_d = 0.65 - 0.05 \cos \frac{4\pi t}{5}$$

where t is time in seconds.

The desired end effector orientation trajectory is represented in Euler parameters as $\lambda_d = (\lambda_{0d}, \lambda_{1d}, \lambda_{2d}, \lambda_{3d})$ where

$$\lambda_{0d} = \lambda_{2d} = \frac{1}{\sqrt{2}} \sin \left(\frac{\pi}{4} \cos \frac{2\pi t}{5} \right)$$
$$\lambda_{1d} = \lambda_{3d} = \frac{1}{\sqrt{2}} \cos \left(\frac{\pi}{4} \cos \frac{2\pi t}{5} \right)$$

The desired operational space coordinates are thus given by: $\mathbf{x}_d = [x_d \; y_d \; z_d \; \lambda_{0d} \; \lambda_{1d} \; \lambda_{2d} \; \lambda_{3d}]^T$.

(a) Desired Trajectory and End Effector Error

(i) Find an expression for the desired velocity trajectory vector $\dot{\mathbf{x}}_d = [\dot{x}_d \; \dot{y}_d \; \dot{z}_d \; \dot{\lambda}_{0,d} \; \dot{\lambda}_{1,d} \; \dot{\lambda}_{2,d} \; \dot{\lambda}_{3,d}]^T$.

(ii) Find an expression for the desired acceleration trajectory vector $\ddot{\mathbf{x}}_d = [\ddot{x}_d \; \ddot{y}_d \; \ddot{z}_d \; \ddot{\lambda}_{0,d} \; \ddot{\lambda}_{1,d} \; \ddot{\lambda}_{2,d} \; \ddot{\lambda}_{3,d}]^T$.

(iii) Give an expression for the E and E^+ matrices which relate these trajectories to the desired operational space velocities and accelerations

(iv) Using these E and E^+ matrices, find an expression for the instantaneous angular error $\delta\phi$ between the current and desired orientations of the end effector.

(b) Operational Space Controller Design

(i) Write the symbolic expression for the control forces and moments required to:

- Track the desired operational space trajectory with dynamically decoupled PD control.
- Compensate for gravitational forces only in operational space

You may assume that you have perfect estimates of the system dynamics.

(note that in practice, we do not often compensate for Coriolis or centrifugal forces as adding these terms tends to destabilize the controller due to estimation errors.)

- *HINT: Use the pseudoinverse of the Jacobian to compute the operational space kinetic energy matrix $\Lambda(x)$ and the operational space gravity vector $p(x)$*

(ii) Write the expression that will allow you to convert your control-forces and moments into torque commands that can be sent to the robot's joint motors.

(c) Null Space

(**i**) What are the dimensions of the robot's Task Jacobian given our current 6DOF task and 7DOF KUKA-iiwa robot?

(**ii**) What is the dimensionality (DOF) of the Task Jacobian's null space?

(**iii**) Define the Null space projection matrix for this Task Jacobian using the pseudoinverse.

(**iv**) Would a 6DOF Puma robot be able to perform the task in this problem while also performing obstacle avoidance?

(**v**) How many DOF would our null-space have if we were simply trying to follow a trajectory, but didn't care about orientation?

Problem 2 - Implementing your controller

IMPORTANT SUBMISSION INSTRUCTIONS:

For parts (b), (c), and (d), you will upload your **hw2_control.cpp** to a seperate Gradescope assignment

Additionally, you should record a video of the simulation running in each part, upload this video somewhere accessible (Google Drive or Youtube both work well), and then include a link to the video for each part.

Alternatively, you may come by Wesley's office hours or email him to schedule a Zoom session in order to show your working simulations.

(a) Setup

If you have followed the code update instructions, you should see a new folder called **hw2** in your bin folder. Inside this folder you will see the **hw2_simviz** and **hw2_control** executable files, as well as a third executable file called **launch.sh**.

You can run the launch.sh file in order to start both the simulation and the controller at (nearly) the same time by running the command

```
./launch.sh
```

If you do so right now, you should observe a window pop up with a model of the KUKA-iiwa, which the rapidly falls to the ground under the effects of gravity. The robot collapses in this way because the controller by default is sending zero torque to all joints.

(b) Initial Implementation

You will be editing the **hw2_control.cpp** source file. Read through the file carefully, taking note of the suggested controller variables and additional comments denoting what each part of the code does

Then, **in the space labeled FILL ME IN (starting on line 142) implement your operational space controller from Problem 1 part (b). Comment on the overall behavior.**

NOTE - make sure that your final control torques are stored in the variable command_torques

(c) Control Improvements through Joint Damping

You may notice that the controller you implemented in part B does not perform quite as well as you might hope.

One way in which we can improve the controller is to add joint space damping and gravity compensation. One possible joint damping term is :

$$A(-k_{vj}\dot{q})$$

where $-k_{vj}$ is a joint velocity damping gain.

However, when adding this damping term, we should take care that this additional term does not affect the task, as it could impact task performance if we just added it directly. Thus, we can instead project it and the joint gravity compensation into the null space of the transpose of the robot's Jacobian matrix, J^T.

$$joint\_damping = (N_{proj})^T(A(-k_{vj}\dot{q}) + g) = (I - J^+J)^T(A(-k_{vj}\dot{q}) + g)$$

Add this additional term directly to your control torque, and **comment on the difference in performance compared to part (b). Explain why this term improves performance.**

(d) Operational Space Dynamic Compensation Effects

Starting with your controller from Part C, make the following modification:

Rather than setting your kinetic energy matrix estimate $\hat{\Lambda}(x)$ equal to the actual kinetic energy matrix, set it instead to be equal to the Identity matrix

$$\hat{\Lambda}(x) = I_6$$

This in effect is removing the part of the controller which performs operational space dynamic compensation.

Run your modified controller and **comment on the difference in compared to part (c). Explain why this controller performs worse.**

Extra Credit:

Using sai2-interfaces and the "plot" redis keys provided, generate a plot which compares the desired and current position of the end effector for both parts (c) and (d). If attempting this part, you should look through the README, docs, and tutorials in the sai2-interfaces folder.