Experimental Robotics (CS225A) Homework #2

(Spring 2022/2023) Due: **Monday, May 1**

Make sure to provide justification for your answers. This includes labeling all of your plots (title, axes, legend, etc.) and explaining what is shown in the plots. Otherwise, you will lose points.

In this homework assignment, you will implement model identification and operational space control for the Panda and experiment with dynamic decoupling (feedback linearization) in operational space.

To download the assignment, you'll have to pull the latest updates from `cs225a.git`. If you want to keep your progress from previous homeworks, first call `git status` to see what files you've modified, and then call `git add <filename>` for all the files you want to save. Next call `git commit -m "Your commit message here"` to commit the changes to these files. For the rest of the files you don't care about, call `git stash` to revert them back to the original version (if you ever decide you want to bring back the modified files, you can call `git stash pop`). At this point, `git status` should show no modified files (untracked files are fine).

Now, you are ready to download the assignment. Call `git pull`. This will likely ask you to save a commit message for merging `cs225a.git` with your local repository - you can simply save and exit. If there were merging issues, you'll have to go into the problem files, manually fix the merging, and then commit those changes again. If you get issues about linking errors when building, the sai2-common updates might not have pulled correctly.

1. In this question, we want to identify the inertial properties of the end-effector of the robot. We will implement the method seen in class to do this. You will implement a PD controller in joint space with gravity compensation and Coriolis compensation but without dynamic decoupling so the control equation will look like:

$$\Gamma = -K_p(q - q_d) - K_v\dot{q} + b + g$$

Where K_p and K_v are diagonal matrices of proportional and derivative gains respectively.

(a) The closed loop equation of motion for joint 7 looks like

$$m_{77}\ddot{q}_7 + (K_{v7} + d)\dot{q}_7 + K_{p7}(q_7 - q_{d7}) + M_{[7,1:6]}\ddot{q}_{[2:7]} = 0$$

Where d is a damping coefficient for the joint due to the transmission (or the simulation) and $M_{[7,1:6]}\ddot{q}_{[2:7]}$ represents the dynamic coupling terms. Identify the value of m_{77} knowing that:

- The mass of the last link is 1 kg
- The center of mass of the link is located at 10 cm in the Z direction of the joint's frame
- The inertial tensor in a frame located at the center of mass of the link is diagonal and its coefficients are $I_{xx} = I_{yy} = 0.1\,\mathrm{kg \cdot m^2}$ and $I_{zz} = 0.25\,\mathrm{kg \cdot m^2}$
- The axis of rotation of the last joint is Z

(b) Assume the dynamic coupling terms are negligible. How can you choose K_{v7} to get an oscillatory system with no damping for joint 7? What is the sign of your chosen K_{v7}? Would you recommend doing this on a real system? Why?

(c) What would be the natural oscillation frequency of this system (when you choose your value of K_{v7} from the previous question) assuming $K_{p7} = 50.0\,\mathrm{Nm \cdot rad^{-1}}$?

(d) Use SAI2 to implement the controller described with $K_{pi} = 400.0\,\mathrm{Nm \cdot rad^{-1}}$ for $i \in [1, 6]$, $K_{vi} = 50.0\,\mathrm{Nm \cdot s \cdot rad^{-1}}$ for $i \in [1, 6]$, $K_{p7} = 50.0\,\mathrm{Nm \cdot rad^{-1}}$ and choose K_{v7} such that you get an oscillatory behavior with no damping. Set $q_{7d} = 0.1\,\mathrm{rad}$.

(e) Plot the trajectory of joint 7 over at least 20 oscillations. Measure the oscillation frequency. Is this the result you expected? If there are differences, explain why. (*Note*: Be careful with the running frequency of your controller and your simulation. If you are running the slow version of the controller, remember that your simulation is running 5 times slower than real time.)

2. Now, you will implement operational space controllers. Let the task positions and velocities of the robot end effector be given by x and $\dot{x}$, respectively. Let x_d be the desired position of the robot. Implement the operational space control law with joint space gravity:

$$F = \Lambda(-k_p(x - x_d) - k_v\dot{x})$$
$$\Gamma = J_v^T F + g$$

where k_p and k_v are your scalar control gains, Λ is your operational space mass matrix (there is a function in `Sai2Model` to compute it), and g is your joint space gravity vector. The control point is attached to "`link 7`" and the point is located at 10.0 cm in the Z direction of the last joint. You can use the variables `link_name` and `pos_in_link` in the code.

(a) Tune your gains to achieve critical damping with $k_p = 200$ and report your chosen k_v. The desired end effector position is $[0.3, 0.1, 0.5]$. Plot the operational point trajectory as well as the joint trajectories.

(b) Notice that the robot reaches the goal position, but the robot still moves in a strange way. Why is this happening? (*Hint*: The task is 3DOF but the Panda is a 7DOF robot. Does this affect the motion?)

(c) We will add extra damping to remove this issue. Add some joint space damping to your commanded torques, that is a term $K_{vj}\dot{q}$. Choose a good K_{vj} value and report it. Plot the operational point trajectory as well as the joint trajectories now. Compare with before and explain what you see.

(d) Notice your robot moves a lot slower now because all your joints are more damped (a physical intuition is all your joints have more friction now). We will now implement joint damping in the nullspace of the control task only. Replace your damping term from the previous question by the nullspace damping term:

$$N^T M(-K_{vj}\dot{q})$$

You can find N (the projection matrix from joint space to task null space) by calling the function `nullspaceMatrix()` from `Sai2Model`. Plot the operational point trajectory as well as the joint trajectories now. Compare with before and explain.

3. We've been using the joint space gravity vector, which simultaneously cancels gravity in all joints. Instead, we will use the operational space gravity vector:

$$p = \bar{J}^T g$$
$$F = \Lambda(k_p(x_d - x) - k_v\dot{x}) + p$$
$$\Gamma = J_v^T F - N^T M K_{vj}\dot{q}$$

$\bar{J}$ is also a function in `Sai2Model`. This is the dynamically consistent Jacobian inverse (generalized inverse of Jacobian with mass matrix weighting). You can use the function `dynConsistentInverseJacobian()` to get it.

(a) Control the end effector to $[0.3, 0.1, 0.5]$. Plot the operational space trajectory as well as the joint angles. What do you see? (*Hint:* Wait!)

(b) Why do we not see this behavior with joint space gravity compensation?

4. We will run our controller from Problem 3 to track a circular trajectory.

   (a) Test your controller by drawing a circular trajectory with:

$$x_d = \begin{bmatrix} 0.3 \\ 0.1 \\ 0.5 \end{bmatrix} + 0.1 \begin{bmatrix} \sin(\pi t) \\ \cos(\pi t) \\ 0 \end{bmatrix}$$

   Choose gains that track the trajectory well while maintaining critical damping for the dynamically decoupled controller (do not change your gains between scenarios (i)-(iv)). For each one of the scenarios below, write your control equations, implement them and plot the trajectory and the desired trajectory for a few cycles as well as the joint angles. Explain what you see and why.

   i. Use the controller from Problem 3
   ii. Use the controller from Problem 3 without using the Lambda matrix (treat Λ as if it was identity)
   iii. Use the controller from Problem 3 with the Lambda matrix and replace the damping term in the nullspace by a dynamically decoupled PD controller that tries to bring the posture to $q_d = [0, 0, 0, 0, 0, 0, 0]$
   iv. Use the controller from scenario (iii), but also compensate for gravity in joint space (not only in the task space)

5. Submit your SAI code.