Experimental Robotics (CS225A) Homework #3

(Spring 2016/2017) Due: **Tuesday, May 2**

In this homework assignment, you will implement orientation, pose control, and velocity saturation for the Kuka.

To download the assignment, you'll have to pull the latest updates from `cs225a-dist.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 --recurse-submodules`. 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. Try `cd sai2-common` from the cs225 folder to go into the sai2-common folder and call `git pull origin master`. Go back into the cs225a folder and call the make script again with `./make.sh`.

1. You will implement posture control of the robot in the nullspace of the task.

   Let the task positions and velocities of the robot be given by x and $\dot{x}$, respectively. Let x_d be the desired position of the robot. Recall the operational space control law with joint space gravity:

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

   where k_p and k_v are your control gains, Λ is your operational space mass matrix, and g is your joint space gravity vector.

   When you implemented null space damping in the last homework, you noticed that we just projected joint space damping into the null space of the task to damp the joints in all directions except for the task direction with the null space projection matrix ($\mathcal{N}^T A(-K_v\dot{q})$). We will extend this idea but controlling the posture as well with:

   $$\Gamma = J_v^T F + \mathcal{N}^T A(K_p(q_d - q) - K_v\dot{q}) + g$$

   (a) Use appropriate gains for operational space from the last homework. Use small gains for joint space posture control but enough to reach the desired posture. Use a posture that is the same as the initial posture (bent shape) and a desired end effector position of $\begin{bmatrix} -0.1 \\ 0.4 \\ 0.7 \end{bmatrix}$. Plot the motion of the end-effector and joints.

(b) Try the same control but without the posture control (set K_P to 0). Use the right mouse button to drag the robot and explain the motion compared to dragging the robot with posture control.

2. We will implement orientation control with the full jacobian $\begin{bmatrix} J_v \\ J_\omega \end{bmatrix}$.

Recall orientation error and full control is:

$$\delta\phi = -\frac{1}{2}\sum_{n=1}^{3} R_i \times (R_d)_i$$
$$F = \Lambda_0 \begin{bmatrix} k_p(x_d - x) - k_v\dot{x} \\ k_p(-\delta\phi) - k_v\omega \end{bmatrix}$$
$$\Gamma = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix}^T F + g$$

where R_i and $(R_d)_i$ are the i-th columns of R and R_d, respectively, and ω is the angular velocity. Note that `ModelInterface::J` returns a flipped jacobian $\begin{bmatrix} J_\omega \\ J_v \end{bmatrix}$.

(a) Control your robot to orientation:

$$R_d = \begin{bmatrix} \cos\frac{\pi}{3} & 0 & \sin\frac{\pi}{3} \\ 0 & 1 & 0 \\ -\sin\frac{\pi}{3} & 0 & \cos\frac{\pi}{3} \end{bmatrix}$$

and position:

$$x_d = \begin{bmatrix} -0.1 \\ 0.4 \\ 0.7 \end{bmatrix}$$

from the initial position. Plot x, and $\delta\phi$ over the motion. Add some null space damping if you feel ambitious or your robot is moving too much. Choose gains appropriately to achieve good tracking but critical damping. If your plot is not precise enough because the robot is moving too fast, reduce your gains.

3. Velocity Saturation

Implement a simple operational space PD controller like in problem 1 with null space damping. From the initial configuration, reach a desired position of $\begin{bmatrix} -0.1 \\ 0.4 \\ 0.7 \end{bmatrix}$.

(a) With operational space $k_p = 150$ and a critically damped k_v, plot x, $\dot{x}$.

(b) Use velocity saturation with:

$$\dot{x}_d = \frac{k_p}{k_v}(x_d - x)$$
$$F = \Lambda(-k_v(\dot{x} - \nu\dot{x}_d))$$
$$\nu = sat(\frac{V_{max}}{|\dot{x}_d|})$$

Where $sat()$ saturates the number between -1 and 1, $V_{max} = 0.05$. Plot x, $\dot{x}$.

4. Attach your SAI code here.