CS327A Programming Assignment 2
Due: Wednesday May 20

In this assignment you will build upon your previous simulation and implement a prioritized controller with three tasks:

End-effector position (you already implemented this one). This task has 1^{st} priority.

End-effector orientation The end-effector frame should always have the same orientation as the ground frame, meaning the desired orientation is I_3. This task has 2^{nd} priority.

Null space damping The position and orientation task each have 3 DOF, the KUKA robot has 7 DOF. Therefore we need to apply damping to the remaining DOF in order to avoid null-space oscillations. This task has 3^{rd} priority.

Your control law should be implemented in the following shape

$$\Gamma = \Gamma_{\text{pos}} + N_{\text{pos}}^{\mathsf{T}} \left(\Gamma_{\text{ori}} + N_{\text{ori}}^{\mathsf{T}} \Gamma_{\text{nulldamp}} \right). \tag{1}$$

To implement this, you should modify the `cTaskOpPos` class. Essentially, your job is to compute `m_Gamma_task`. The two functions you need to modify are, again, `cTaskOpPos::computeTaskTorque()` and `cTaskOpPos::updateModel()`.

Keep in mind that expensive operations (such as computing Jacobians, matrix inversions, null-spaces) should be done within `cTaskOpPos::updateModel()`. Only the control law should be implemented in `cTaskOpPos::computeTaskTorque()`. (The simulator executes updateModel() only once for every 10 control cycles.)

You can access data structures you computed in `cTaskOpPos::updateModel()` from within `cTaskOpPos::computeTaskTorque()` by storing them as member variables. Simply initialize them in `cTaskOpPos.h`.

In your starter code, I had a null-space damping task running in the background, which you need to disable now. Please open `Application.cpp` and comment the line `my_op_controller->addTask(my_task_nulldamp, 1);`.

Hints

- You can access the angular velocity Jacobian with `m_dynamics_interface->getJw( m_link_name );` .
- Compile in debug mode during development, as this will apply error-checks (e.g. matching dimensions in your Eigen computations).
- Your controller does NOT need to compensate for gravity, coriolis, or centrifugal forces
- Use the left and right mouse buttons to rotate and zoom the scene camera
- If you are having low screen resolution, the scopes may appear very large and block the view on your robot. You can adjust their size by modifying `scope_size_horizontal` and `scope_size_vertical` in `Application.cpp`.
- Please post on Piazza if you discover any issues

You may not use or share this code with anyone outside the framework of CS327A, as this is experimental code and NOT an official release.

Grading and Collaboration

You will demonstrate your simulations and explain your code during an interactive grading session on the due date, for which I will set up an online sign up sheet. You must complete this assignment individually, you may discuss code concepts (such as "how do I multiply two matrices using Eigen?"), but you may not share your actual implementation.