CS327A Programming Assignment
Due: Monday May 11

In this assignment you will implement some of the controllers you derived in the previous assignments using a real-time simulation and control framework. You are set up with a dynamic simulation of the KUKA robot, a control framework (GCL), and a simple GUI.

This is a first-time assignment and somewhat experimental, if you experience issues with the installation process or coding part, please contact me at any time. Please follow the Piazza posts, as I am sure there will be a number of clarifications coming up. If you prefer, I can set you up with a working environment on one of our lab computers.

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.

Installation

This simulation environment runs on either MacOSX or Ubuntu (tested on 12.04 and 14.04). Before compiling the application, you need to install a few 3rd party libraries.

- Download and install Qt from https://www.qt.io/download-open-source (go to offline-installers and choose accordingly, and go with the standard install path)
- Mac: install homebrew (instructions here: http://brew.sh/)
- Mac: install libraries: open a terminal and type `brew install cmake libusb boost eigen`
- Linux: install libraries: open a terminal and type `sudo apt-get install cmake libglew-dev libusb-1.0-0-dev freeglut3-dev libxmu-dev libxi-dev libasound2-dev libboost-dev libeigen3-dev`

Now that you installed all the dependencies, you can download and install the actual simulation environment.

- Download the simulator: `https://www.dropbox.com/s/a3ebv7tkq3kkthf/GCL_Student.zip?dl=0`
- Open a terminal and navigate to `GCL_Student`
- Type `sh make_3rdParty.sh`. This will take a few minutes. Make sure it compiles without errors before moving on from here.

- The application you will work on is located in `GCL_Student/application/24-cs327hw`.
- In this folder, open `CMakeLists.txt`. Here you need to specify your Qt install paths to the directories `Qt5Widgets` and `Qt5OpenGL`. The two line numbers you need to modify are `107` and `110`.
- Now you can compile the actual application. Type `sh make_rel.sh` or `sh make_dbg.sh` to compile in release or debug mode.
- This will generate the build directories `build_rel` or `build_dbg`. To run the application, navigate to either of these folders and type `./24-cs327hw`.
- You should now be able to see the simulated KUKA robot and be able to use the basic GUI.

Generalized Coordinate Controller

Your first goal is to implement a simple PD position controller with non-linear dynamic decoupling. To do this, you need to modify the code in `24-cs327hw/gcl/tasks/cTaskGcPos.cpp`. Search for `CS327A TODO Part1`. After successful implementation, the robot should follow a sinusoidal trajectory (each joint). You can set and experiment with the k_p and k_v values by using the slider in the GUI.

Operational-Space Controller

Next, you will implement an OP-space controller that tracks the end-effector's linear position (a 3-dof task). Again, using non-linear dynamic decoupling in op-space. To do this, you need to modify two parts in `24-cs327hw/gcl/tasks/cTaskOpPos.cpp`. First, you need to compute a few data structures associated with the robot model (search for `CS327A TODO Part2`). Second, you need to implement the actual control law (search for `CS327A TODO Part3`). After successful implementation, the yellow sphere (current position) should track the red sphere (goal position).

Hints

- For your matrix/vector computations, you will use the c++ Eigen library. If you are unfamiliar with this one, get started here: http://eigen.tuxfamily.org/dox/GettingStarted.html
- You need to run `sh make_3rdParty.sh` only once. But whenever you modify your code, you need to recompile the application with `sh make_rel.sh` or `sh make_dbg.sh`.
- 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 GUI slider to set k_p and k_v. At initialization they are set to 0.0.
- Use the GUI radio buttons to switch between the two control modes
- Use the left and right mouse buttons to rotate and zoom the scene camera
- You can apply an external force to the robot by selecting the radio button and left-click-dragging the robot
- You can use and IDE but a simple text editor will do perfectly fine. (I recommend Sublime Text)
- Please post on Piazza if you run into any issues (expected)

Resolving Install and Compiling Issues

- To install Qt Creator on Linux: Download the executable (qt-opensource-linux-x65...), open a terminal and navigate to the folder this file is located in. Type `chmod +x qt-opensource-linux-x65..` and `./qt-opensource-linux-x65....`
- On Linux: If you get errors related to GL, try the following: `sudo apt-get install mesa-common-dev mesa-utils-extra libgl1-mesa-dev libglapi-mesa`
- On Linux: If you get errors related to "CMAKE_CXX_COMPILER-NOTFOUND", try the following: `sudo apt-get update && sudo apt-get install build-essential`
- After installing Ubuntu, make sure you activate the proprietary binary graphics drivers. To do that, go to Settings – Software & Updates – Additional Drivers.

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.