Marco Morales ME 449 Capstone Project

Date: December 9, 2021

Objective

For the Final Project, the goal was to control the Kuka youBot in CoppeliaSim and have it complete three objectives

1. Create a "Best" simulation where the youBot picks up a cube and moves it to a goal position with no oscillation and the error converging to 0 before the first trajectory segment is completed.
2. Create a "Overshoot" simulation where the Kp and Ki gains make the youBot overshoot, and oscillate, when picking up the cube but still pick it up successfully and then moving it to the goal position.
3. Create a "NewTask" simulation where the cube's initial and goal positions are changed. The robot has to move to the cube's new initial position and then place it at the new goal position.

Software

The general idea of this project is to create three functions, which are called TrajectoryGenerator, NextState, and FeedbackControl (which I defined as Feedbackcontrols), and then have them work together to complete the objective mentioned previously.

TrajectoryGenerator was created in Milestone 2. The purpose of this function is to create an array of 13 values that make up a transformation matrix, such as rotation and translation, as well as the gripper state.

NextState was created in Milestone 1. NextState takes the configuration of the youBot and the joint speeds to create the future configuration which includes the chassis phi,x and y value, the joint values for the arm and the wheel angles for the arm.

Finally, FeedbackControl was created in Milestone 3 and it calculates the control law based on kp,ki and the configurations. The control law is made up of the wheel speeds and the joint speeds that were calculated using the pseudoinverse of jacobian of the end effector and the feedforward reference twist.

Navigation

For each of the three objectives, a folder was made just for those results. For "Best" and "NewTask", there are 7 items and 8 items for "Overshoot". For each folder, there is a png file of the X_err plot over time, a csv file of the calculated configurations, a csv file of the X_err, a log file of the my program being called with the input with statements that occur as the code runs, a .txt file that has the same information as the log file, a brief README.pdf, and a video of the youBot in CoppeliaSim performing the task. For “Overshoot” there is another video to show joint limits being applied.

In the code folder, there are 7 python files. Milestone1.py, Milestone2.py, Milestone3.py, and Full_program.py can be called by calling `python3` and then the name of the python script. This applies for the other scripts, which are Best.py, Overshoot.py and NextState.py. Best.py has the conditions that would result in the getting the “Best” solution. Overshoot.py has the conditions that would result in getting the “Overshoot” results. Finally, NextState.py has the conditions that would result in the “NextState” simulation and results. The conditions are the kp,ki and the Tsc initial and goal for Next State.

Joint Limits and Singularities

Joint limits were implemented into the code and this was tested in the “Overshoot”. the idea of the Joint limits is that if a joint angle is greater than or less than a threshold, then make the column of that joint Je matrix zeros. This would result in the jacobian, Je, not using the joints that do not follow these joint limits. The joint limits were measured by using scene 3 from CoppeliaSim as well as trial and error.

As for singularities, setting a cutoff for small values was set and this cutoff point was 1e-02, or 0.01.

Results

Best

For the “Best” objective, the youBot was able to pick up the cube at a new initial configuration value for the youBot. This initial configuration includes a chassis rotation and displacement as well as joint and wheel angles that would require the use of a feedback control and PI control. The error converges to zero before the end effector reaches its first trajectory which is the stand off point above the cube at its initial position. The animation is smooth with no oscillation.

Overshoot

For the “Overshoot” objective, the youBot was able to pick up the cube with a large overshoot. More specifically, there is oscillation in both the simulation and the X_err graph. When the youBot approaches the cube at its initial position. The end effector misses the cube but it oscillates back leading it to successfully pick up the cube and continue with the trajectory. In the simulation, there is a large amount of collision with the youBot. This is where joint limits are introduced.

There is a second video in the “Overshoot” folder where the video shows the simulation when joint limits are introduced. The simulation shows that the robot does not collide with itself at all, unlike the original “Overshoot” video where the arm collides with the chassis. However, introducing the joint limits results in the youBot not being able to pick up the box. This could be the result of the joints being too constrained. The joint limits prevent the robot from reaching configurations where the arm collides with either itself or the chassis. It is important to note that the conditions were exactly the same but the test_joint_limits was not used for the original. You can uncomment this in Milestone3.py to enable it.

NewTask

The youBot was able to successfully pick up the cube when it is at a new initial position and then move it to a new goal position. This was accomplished by changing both the cube's initial and goal configurations. These new configurations would be the input into TrajectoryGenerator that would result in configurations for the end effector. The error converges to zero right before the end of the first trajectory point but then there is an instance of small spike in error as the youBot moves from one stand off position to another.