<u>Assignment 3 Report – Walter Stark</u>

Task 3:

In this task, I made a controller that causes the soft finger to follow a sinusoidal trajectory. I implemented this by adjusting the position effector goal X and Y values overtime using sinusoidal functions. Both functions had the same frequency but had different intercept and amplitude values that were tuned to reduce error. Since the finger doesn't move in the Z direction, that was kept constant at a value determined that closely matched the position of the finger (6.76272189649mm).

I implemented two separate PI controllers for both the X and Y values to decrease the steady-state error. I used an integral and proportional gain factor of 5 and 1.2 respectively to decrease the controllers rise time and help remove steady state error. These gain factors were chosen by slowly increasing the proportional gain first until the error slightly decreased and then increasing the integral gain by a large amount (since it had a larger effect on the error) iteratively until the median error would settle for both X and Y to an acceptable value (lower than 1). However, one downside of these gain values is that it took around 5 seconds to get to a sinusoidal motion with minimal errors (large settling time).

Due to the PI controller, the median error was small. The median error in X, Y, and Z settled to around -0.13452357308895913mm, 0.1014554950440889mm, 0.10027013733151924mm respectively over time. The median tracking error (distance) based on these values was 0.19607118511mm. These values were pulled from one instance in time since they slightly change over time, but they stay within +/- 0.02mm of the above value. Overall, the finger moves smoothly in a sinusoidal trajectory with minimal error.

Task 4:

In task 4, I was successfully able to pick up and move three objects from each of their starting positions to a predetermined end point. I split up the task into subcomponents: moving to the object, moving up and down, grasping/dropping the object, and moving to destination. By splitting the task into these subcomponents, this method made it easy to loop over these components and repeat this method for different object geometries.

While running the program, I found that the grabber and controller were quite robust. I designed the controller such that the fingers would compress more if the object was falling and still be able to hold onto it (as seen with the espresso cup). One issue I did run into was that my computer did not have enough computational power to run the program due to the high number of collisions. As such, the provided video for Task 4 is at 4.5x speed. In the future, I would attempt to optimize my program potentially using my GPUs CUDA. Additionally, I found that the objects would roll/move after dropping them. I mostly resolved this by having the gripper pause when reaching the destination and pausing after releasing. Lastly, I optimized the design of my gripper by decreasing the diameter of the palm to the smallest diameter object and increasing the size of the grippers. I found this made it easier to grasp objects and hold onto them while moving.

References:

Sofa-Framework. (n.d.-a). *Sofa/caduceus.scn at master · sofa-framework/sofa*. GitHub. https://github.com/sofa-framework/sofa/blob/master/examples/Demos/caduceus.scn

Sofa-Framework. (n.d.-b). *Sofapython3/example-scriptcontroller.py at master · sofa-framework/sofapython3*. GitHub. https://github.com/sofa-framework/SofaPython3/blob/master/examples/example-scriptcontroller.py

Tutorial: Making A soft gripper. Soft Robotics Toolkit. (n.d.). https://softroboticstoolkit.com/sofa/tutorial/