Follow the Leader: Exploring Self-Referencing Cost Functions in Trajectory Optimization

1st Grace Tang
Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, USA
gtang@mit.edu

***Abstract*—Drones are becoming equipped with more and more advanced sensors, and it is becoming apparent that they can work together to serve in even more powerful applications, such as search/rescue, carrying rather heavy objects, and defense. However, in order for drones to work in groups, they need to be capable of not crashing into each other. Thus, this project creates a simulation with 2 quadcopter drones and designates one as the leader and the other as the follower. It then discusses the constraints necessary for the follower to stay reasonably close to the leader without getting within a certain radius of collision.**

***Index Terms*—quadcopters, trajectory optimization**

I. Introduction

Drones are useful on their own, but have even more potential when working together. However, this requires them to maintain proper spacing. They can neither drift to far nor run into each other. Thus, this project focuses on getting 2 drones to maintain proper following distance even though the trajectory constraints only correspond to the first drone. The second drone is controlled only relative to the first one. This could save energy on a lot of repeated computations, if quadcopters only had to worry about where they were relative to other quadcopters.

II. Drake setup

A. Working with the Multibody Physics Plant

Although there were plenty of examples for Drake's built-in QuadcopterPlant, these models don't allow for much interaction between the quadcopters and other objects in the environment. For example, the original intent of this project was to simulate the effects of wind and an uneven payload on a quadrotor moving along a trajectory. Unfortunately, the project had to shift direction from this after wind turned out to be very difficult to model. Regardless, starting off with the more flexible Multibody physics plant would allow for more interesting simulations in the future.

The given QuadrotorGeometry was necessary to create a visual model in Meshcat, but in order to get it into the Multibody object, it had to be wrapped in a diagram with propellers, an input vector of length 4 (one for each propeller) and an output state vector of 12 (x, y, z, roll, pitch, yaw, and the derivatives of all 6). This diagram was then inserted into a larger diagram with a duplicate one to represent the other drone. After the two input and output vectors were multiplexed/demultiplexed appropriately so that the outer/multidrone diagram had just one input and output port, the multidrone diagram was inserted into a final diagram with the Meshcat attached.

It was necessary to condense all drones into one diagram with one input/output port because the next step was to feed this model into Drakes's direct collocation library. Direct collocation was chosen in this case because quadrotors generally don't have extreme changes or nondifferentiable points in their dynamics. Thus, direct collocation's piecewise approximations seemed to be a good fit.

B. Single-Quadrotor experimentation

To get more familiar with the Multibody quadrotor, it was also simulated in a single-quadrotor context. For example, before experimentation could go on, the quadrotor was connected to an LQR controller and made to converge to a given point from any position.

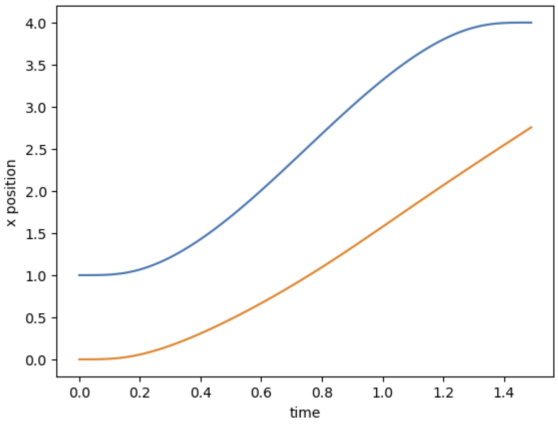
III. Cost Functions

This leader-follower system was optimized for both time and thrust/energy consumption. As a result, the cost expressions used were

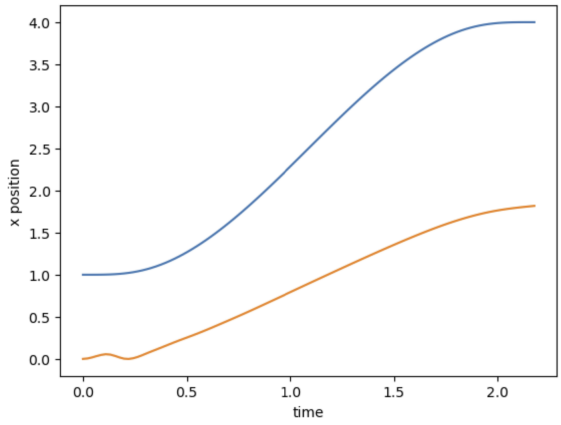
$$\Sigma_{i=0}^{i=3} xu[i]^2 \tag{1}$$

$$\Sigma_{i=4}^{i=7} yu[i]^2 \tag{2}$$

where the 0th-3rd input pertained to the leader quadcopter, and the 4th-7th input pertained to the follower quadcopter. Different x and y constants were experimented with in order to see how weighting one quadcopter's energy towards the cost affected the convergence rate of the paths. For example, below is a plot of the leader (blue) vs. follower (orange) drones' position while the leader traveled in a straight line. In this case, their thrust costs were weighted equally.



After this example, the cost scalar for the follower quadcopter y was scaled up from 10 to 100. The below position response was observed:



Note that the yellow curve (the follower) is less able to keep up with the leading curve, since energy spent in its propellers is weighted so heavily.

IV. Results/Output graphs

V. Challenges/Work Currently in Progress

A. Rendering

While the Meshcat rendering worked well in the case of the single-quadcopter trajectory optimization and LQR simulations, adding in a second quadcopter unfortunately ruined it, hence the need for position graphs instead of video simulations. The follower quadcopter would loop back to the start for no reason, despite this behavior not being outlined in the trajectory plots.

B. 3-Quadcopter Simulation

Experimentation was started with a third quadcopter in the diagram. Convergence was observed, but unfortunately, getting the quadcopters to maintain a 1 unit distance apart was very difficult due to the notebook reporting NaNs in the cost function.