**Trajectory Planning**

We used MATLAB to generate coefficients for quintic minimum jerk trajectories. For trajectories composed of successive straight line segments, each line was planned over seconds, with the quadrotor stopping at each waypoint. We calculated coefficients and in the x, y, and z directions based on the start and end positions of each line segment and start and end velocities and accelerations of zero. We then output the coefficients and length of time for each segment to a text file. After reading the text file into C, we calculated the time-parameterized position, velocity, and acceleration:

These values were the desired positions, velocities, and accelerations that we fed into our controller. We used this method to generate a closed square trajectory.

For curved trajectories, we used parametric equations for x and y and had the parameter vary according to a quintic. We created a figure 8 trajectory in the x-y plane with the position equations:

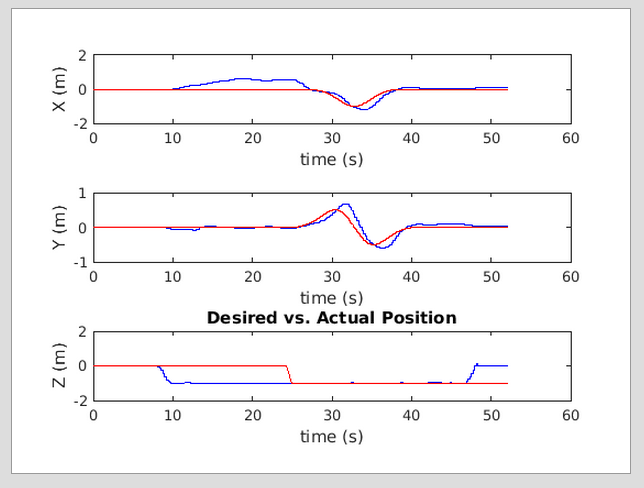
Similarly, we created a circle trajectory with the equations:

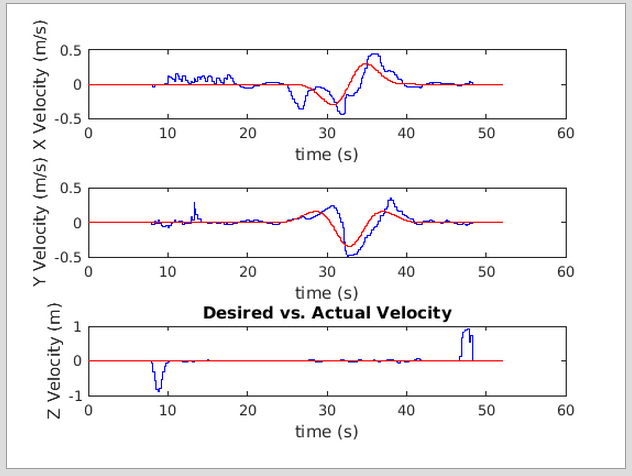
In MATLAB, we calculated coefficients and for as it varies from zero to and output these coefficients and the time to complete the entire trajectory to a text file. After reading these coefficients into C, we calculated and and used these to calculate x and y positions, velocities, and accelerations, holding z constant.

**Tracking Results:**

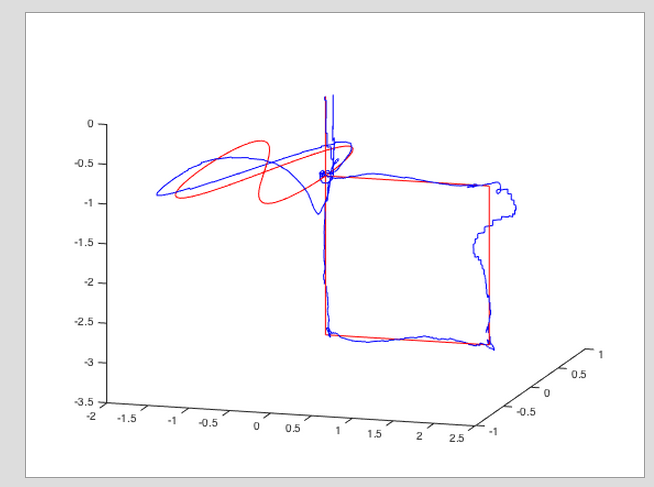
*Circle Trajectory*

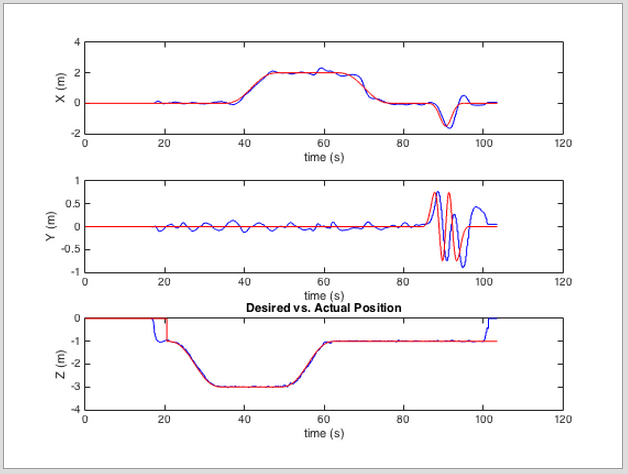
These are our plots for our circle trajectory with a diameter of 1 meter. Our actual position tracks our desired position relatively well. Our quadrotor was able to maintain its hover position at our desired z height – the offset before and after the trajectory is due to our bookkeeping of height, and the spikes in z velocity correspond to takeoff and landing. Our x and y positions also track well but they lag somewhat and overshoot the desired position. This seems mostly due to the aggressive trajectory. With 12 seconds to complete the trajectory, the quadrotor wasn’t quite able to keep up with the desired position and velocity inputs. There may also have been issues resulting from the linear nature of our controller. A radius of .5 meters is tight relative to the size of the quadrotor, and requires roll and pitch angles that may be too large to adhere strictly to our assumption of linearity. Finally, the irregularities in our velocity feedback could have resulted from how the Bebop measures velocity. It uses optical flow to provide velocity feedback, and the surface of the floor during this test did not have an abundance of good features to aid in optical flow calculations.

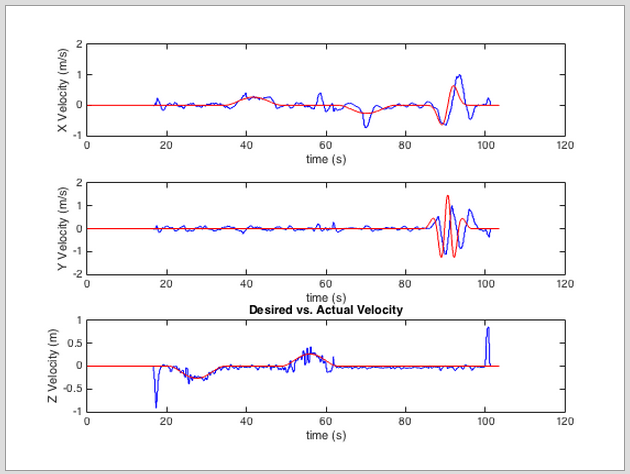




*Figure 8 and Square Trajectories (from demo)*







The figure 8 and square trajectories are the ones we ran during our demo. Figure 8 tracking suffered from a lot of the same issues as the circle – it was trying to follow an aggressive trajectory in a short amount of time (also 12 seconds) and couldn’t keep up. We see this especially in the y position and velocity graphs, where the figure 8 corresponds to the trajectory between 80 and 100 seconds. It also may have required roll and pitch angles that broke our assumption of linear dynamics.

Square – underdamped controller gains in x-y direction?