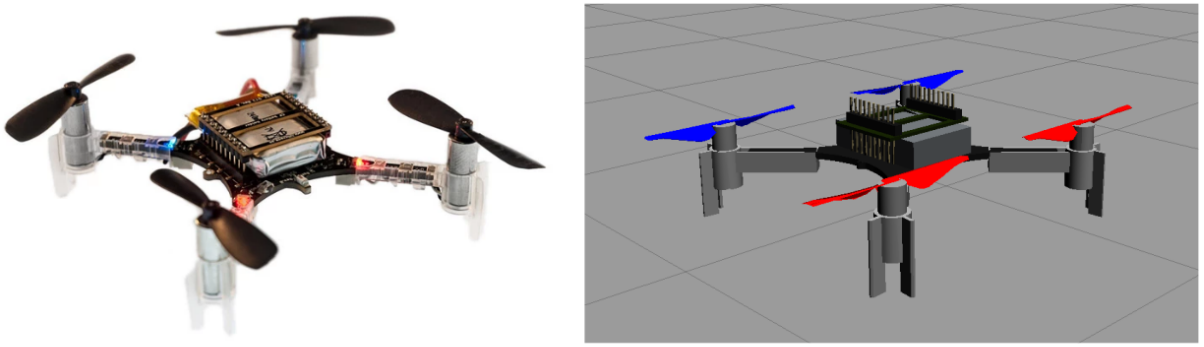
RBE 502 — Robot Control

Students: 1) Denny Boby (dboby@wpi.edu)
2) Wael Mohammed (wmohammed@wpi.edu)

Final Project:

1.1 Overview

This project aims to develop a robust control scheme to enable a quadrotor to track desired trajectories in the presence of external disturbances.



(a) (b)

Figure 1: Crazyflie 2.0 Quadrotor (a) hardware, (b) Gazebo physics based simulation.

1.4 Problem Statement

Design a sliding mode controller for altitude and attitude control of the Crazyflie 2.0 to enable the quadrotor to track desired trajectories and visit a set of desired waypoints.
The main components of the project are described below.

Part 1 Trajectory Generation

Consider a quintic trajectory of the form

$$qd(t) = a0 + a1*t + a2*t^2 + a3*t^3 + a4*t^4 + a5*t^5$$

The coefficients a0, a1, a2, a3, a4 and a5 can be found by solving the matrix equation below:

Deriving the equations and constraints:

$$\underbrace{\begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 & t_0^4 & t_0^5 \\ 0 & 1 & 2t_0 & 3t_0^2 & 4t_0^3 & 5t_0^4 \\ 0 & 0 & 2 & 6t_0 & 12t_0^2 & 20t_0^3 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 \\ 0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix}}_{a} = \underbrace{\begin{bmatrix} q_0 \\ \dot{q}_0 \\ \ddot{q}_0 \\ q_f \\ \dot{q}_f \\ \ddot{q}_f \end{bmatrix}}_{b} \Longrightarrow a = inv(A)*b$$

known

Six equations, six unknowns → solve for $a_0, a_1, a_2, a_3, a_4, a_5$

unknown given

Here except a0, a1, a2, a3, a3, a4 and a5 everything else is known and has values:

- $p_0 = (0, 0, 0)$ to $p_1 = (0, 0, 1)$ in 5 seconds
- $p_1 = (0, 0, 1)$ to $p_2 = (1, 0, 1)$ in 15 seconds
- $p_2 = (1, 0, 1)$ to $p_3 = (1, 1, 1)$ in 15 seconds
- $p_3 = (1, 1, 1)$ to $p_4 = (0, 1, 1)$ in 15 seconds
- $p_4 = (0, 1, 1)$ to $p_5 = (0, 0, 1)$ in 15 seconds

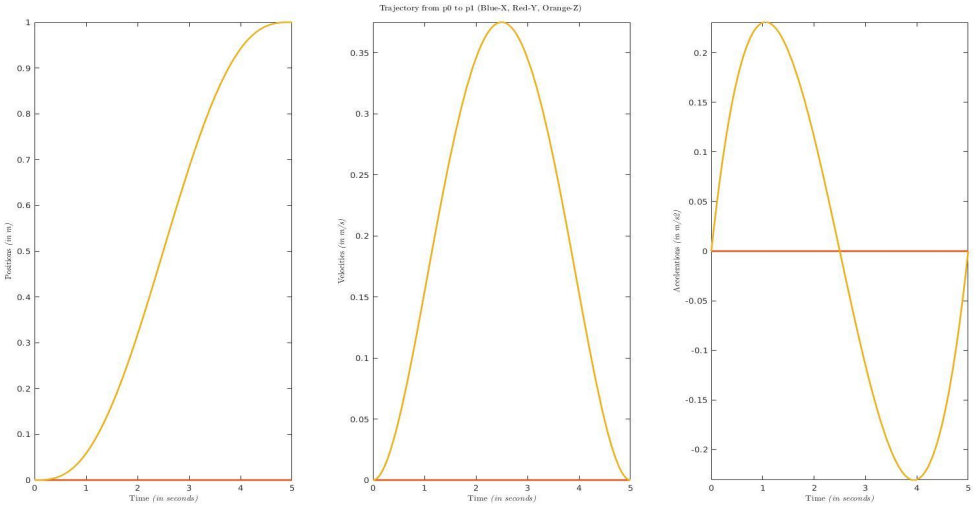
The sequence of visiting the waypoints does matter. The velocity and acceleration at each waypoint must be equal to zero.

Differentiating qd(t) w.r.t time we get velocity and again differentiating velocity w.r.t time we get acceleration.

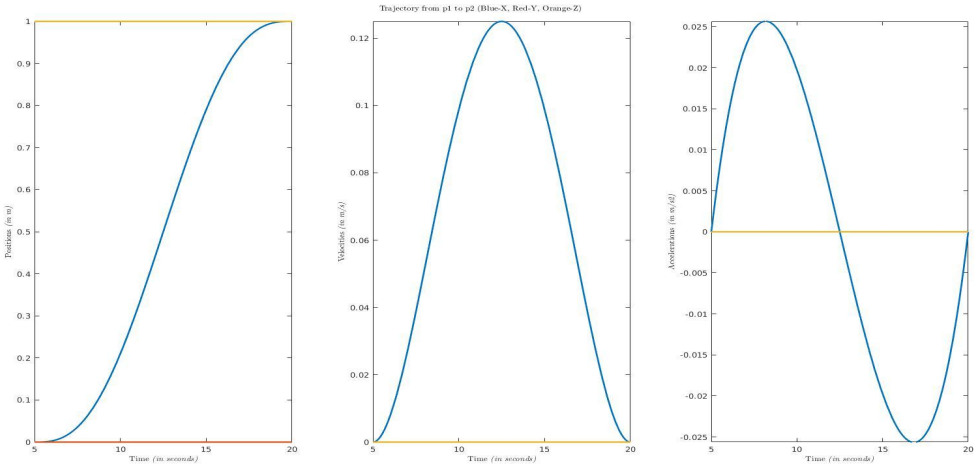
$$qd`(t) = a1 + 2*a2*t + 3*a3*t^2 + 4*a4*t^3 + 5*a5*t^4$$

$$qd``(t) = 2*a2* + 6*a3*t + 12*a4*t^2 + 20*a5*t^3$$

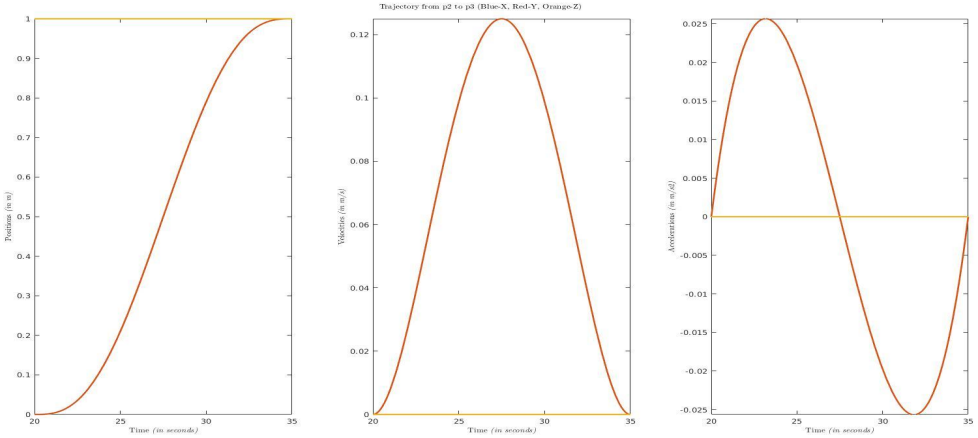
The plot for p0(0,0,0) to p1(0,0,1) in time 0 - 5 sec is the following:



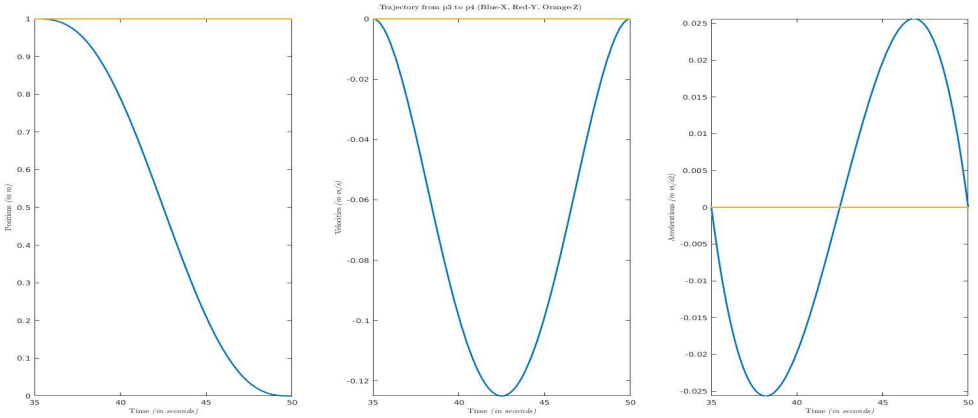
The plot for p1(0,0,1) to p2(1,0,1) in time 5 - 20 sec is the following:



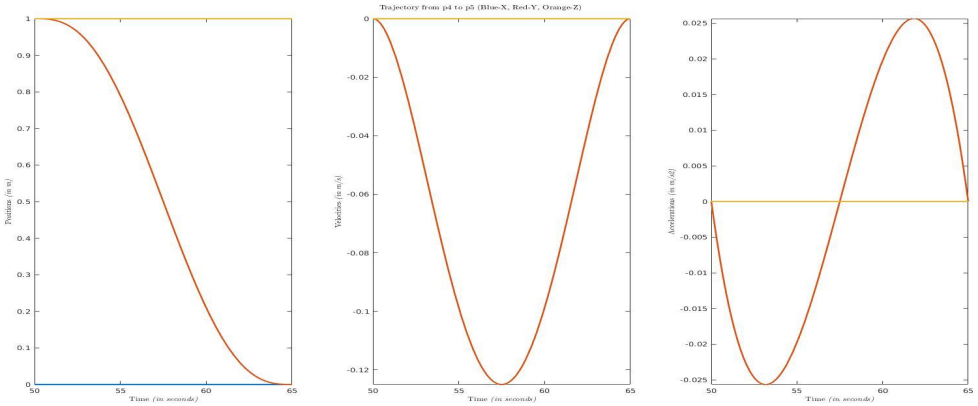
The plot for p2(1,0,1) to p3(1,1,1) in time 20 - 35 sec is the following:



The plot for p3(1,1,1) to p4(0,1,1) in time 35 - 50 sec is the following:



The plot for p4(0,1,1) to p5(0,0,1) in time 50 - 65 sec is the following:



Part 2 Controller Derivation

The derivation for the control laws is done below in the handwritten part.

Tuning Parameters:
The following tuning parameters were used.

a) PD Controller tuning parameters:-
   Kp and Kd are used for deciding how quickly the system converges to the trajectory. The larger the value, the faster the convergence.

   Kp = 85, Kd = 8

b) Sliding Mode Control Lambda parameters:-
   These tuning parameters are used in the sliding mode control to decide how quickly to take the system from initial condition to the designed sliding surface. Since we designed 4 sliding mode controllers, we have 4 of these parameters. The larger the value, the faster the convergence of the system from initial values to the sliding surface.

   lambd_z = 4, lambd_roll = 7, lambd_pitch = 7, lambd_yaw = 4

c) Sliding Mode Control Gain parameters:-
   These tuning parameters are used in the sliding mode control to decide how quickly to take the system down the sliding surface. Since we designed 4 sliding mode controllers, we have 4 of these parameters. The larger the value, the faster the system slides the sliding surface to reach the desired trajectories.

   k_z = 7, k_roll = 100, k_pitch = 100, k_yaw = 25

d) Saturation function constant:-
   This parameter is used to avoid chattering in sliding mode control and is used to select the boundary region along the desired trajectory. The smaller the value, the more closer the boundary is to the desired trajectory. In case chattering still occurs after the saturation function is used, we should increase the value of the constant a.

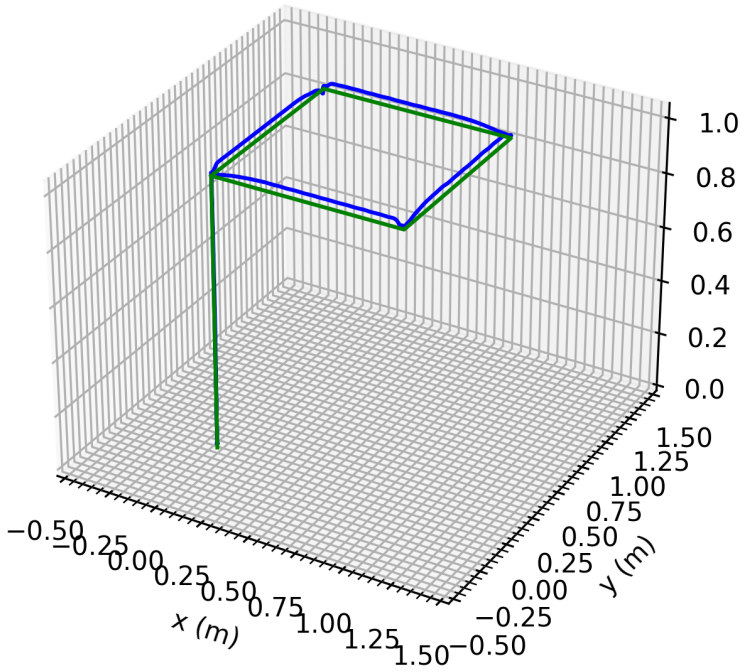
   a = 0.2

Part 3 Code Explanation and Trajectory Plot

Code Explanation:
When the odom_callback is called we initialize our time (self.t) and from the odometry message, the drone's current position, velocity, orientation, and angular velocity along the 3 axes are extracted. These values are passed to the smc_control function. smc_control first calls the traj_evaluate function which, based on the current time, predicts which trajectory the drone is following and calls generate_trajectory to get the desired position, velocity, and acceleration for the x,y,z axis for that time instant. The desired position, velocity, and acceleration are calculated based on the derivation in part 1. traj_evaluate returns these desired positions, velocity, and acceleration to smc_control. smc_control then calculates the desired roll, pitch, and yaw angles and then errors in altitude z, roll, pitch, and yaw angles are defined. Then the sliding surface is defined using the errors. Then u1, u2, u3, and u4 are utilized according to the calculations in part 2. Then motor_speed takes as input the values

of u1, u2, u3, and u4 and returns the motor angular velocities based on the allocation matrix. These angular velocities are clamped between motor min and max speeds. And finally, these motor velocities are published to the motor_speed topic to actuate the drone.

3D Plot Of the Actual Trajectory:



We can see from the 3D plot that the trajectory of the drone is a bit off from the actual trajectory. This is because of the saturation function that is used to avoid chattering. Thus we can say that the controller works efficiently and is robust to handle reasonable external disturbances and follow the desired trajectory pretty closely.