RBE 502 : Robot Controls

Robust Trajectory Tracking for Quadrotor UAVs using Sliding Mode Control

Aabha Tamhankar
Miheer Diwan

Problem Statement

To 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.

Part 1 : Trajectory Generation & Plots

We generated quintic trajectories for (x,y,z) co-ordinates of the quadrotor. It generated position, velocity and acceleration desired trajectories for visiting 5 waypoints in a particular sequence, as follows.

- p0 = (0, 0, 0) to p1 = (0, 0, 1) in 5 seconds
- p1 = (0, 0, 1) to p2 = (1, 0, 1) in 15 seconds
- p2 = (1, 0, 1) to p3 = (1, 1, 1) in 15 seconds
- p3 = (1, 1, 1) to p4 = (0, 1, 1) in 15 seconds
- p4 = (0, 1, 1) to p5 = (0, 0, 1) in 15 seconds

The first order and second order derivatives of the equations were then calculated for the trajectory plotting. The plotted trajectories are in the figure below.

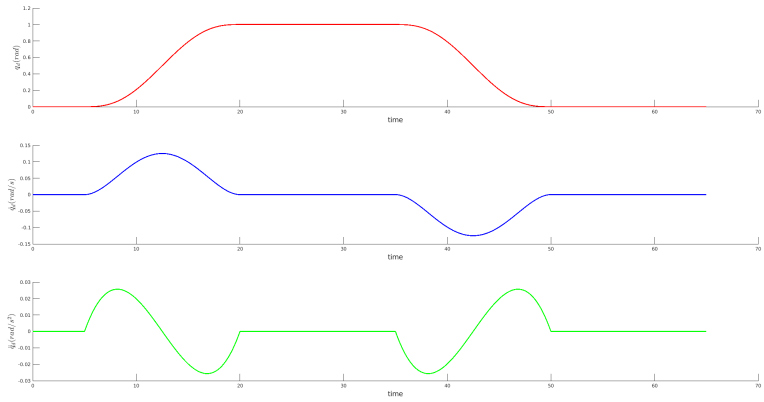

Figure 1: Graph of x Position, Velocity and Acceleration vs Time

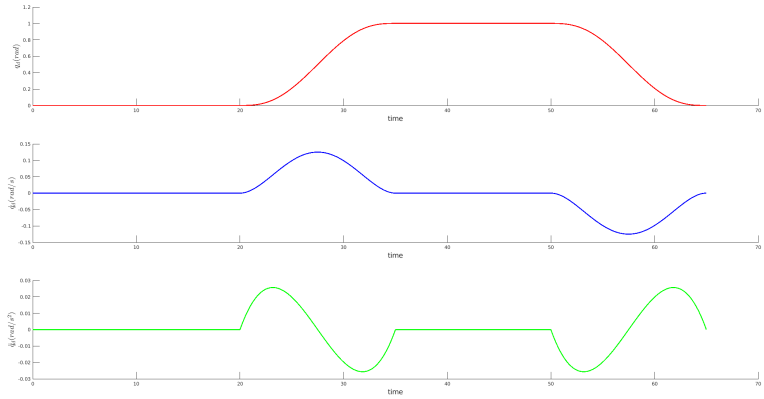

Figure 2: Graph of y Position, Velocity and Acceleration vs Time

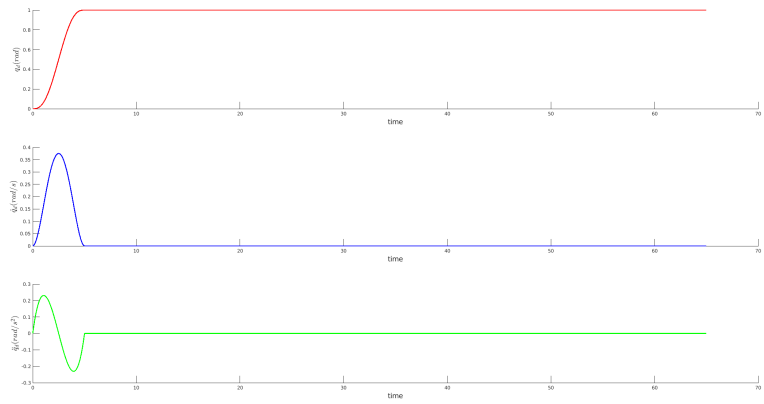

Figure 3: Graph of z Position, Velocity and Acceleration vs Time

Part 2 : Derivation of Sliding Mode Control Laws

Equations of Motion

$$\ddot{x} = \frac{1}{m}(cos\phi\ sin\theta\ cos\psi + sin\phi\ sin\psi)\ u_1$$
$$\ddot{y} = \frac{1}{m}(cos\phi\ sin\theta\ sin\psi - sin\phi\ cos\psi)\ u_1$$
$$\ddot{z} = \frac{1}{m}(cos\phi\ cos\theta)\ u_1 - g$$

$$F_x = m(-k_p(x - x_d) - k_d(\dot{x} - \dot{x}_d) + \ddot{x}_d)$$
$$F_y = m(-k_p(y - y_d) - k_d(\dot{y} - \dot{y}_d) + \ddot{y}_d)$$

$$\theta_d = sin^{-1}(\frac{F_x}{u_1})$$
$$\dot{\theta}_d = 0$$
$$\ddot{\theta} = \dot{\phi}\dot{\psi}\ \frac{I_z - I_x}{I_y} + \frac{I_p}{I_y}\ \Omega\ \dot{\phi} + \frac{1}{I_y}u_3$$

$$\phi_d = sin^{-1}(\frac{-F_y}{u_1})$$
$$\dot{\phi}_d = 0$$
$$\ddot{\phi} = \dot{\theta}\dot{\psi}\ \frac{I_y - I_z}{I_x} - \frac{I_p}{I_x}\ \Omega\ \dot{\theta} + \frac{1}{I_x}u_20$$

$$\psi_d = 0$$
$$\dot{\psi}_d = 0$$
$$\ddot{\psi} = \dot{\phi}\dot{\theta}\ \frac{I_x - I_y}{I_z} + \frac{1}{I_z}u_4$$

Using these equations of motion, we can now calculate the control inputs, i.e. u1, u2, u3, u4.

Calculations for u1

$$e = z_d - z$$

$$\dot{e} = \dot{z}_d - \dot{z}$$

$$\ddot{e} = \ddot{z}_d - \ddot{z} = \ddot{z}_d - \frac{1}{m}(cos\phi\ cos\theta)\ u_1 + g$$

$$s = \dot{e} + \lambda_1 e$$

$$\dot{s} = \ddot{e} + \lambda_1 \dot{e}$$

$$\dot{s} = \ddot{z}_d - \frac{1}{m}(cos\phi\ cos\theta)\ u_1 + g + \lambda_1 \dot{e}$$

$$s\dot{s} \leq -K_1|s|$$

$$u_1 = \frac{m}{(cos\phi\ cos\theta)}(\ddot{z}_d + g + \lambda_1 \dot{e} + K_1 sign(s))$$

By using the *sat* function in place of *sign* function, it was seen that we could avoid chattering.

$$u_1 = \frac{m}{(cos\phi\ cos\theta)}(\ddot{z}_d + g + \lambda_1 \dot{e} + K_1 sat(s, \rho_1))$$

Calculations for u2

$$e = \phi_d - \phi$$

$$\dot{e} = \dot{\phi}_d - \dot{\phi}$$

$$\ddot{e} = \ddot{\phi}_d - \ddot{\phi} = \ddot{\phi}_d - \ \dot{\theta}\dot{\psi}\ \frac{I_y - I_z}{I_x} + \frac{I_p}{I_x}\ \Omega\ \dot{\theta} - \frac{1}{I_x}u_2$$

$$s = \dot{e} + \lambda_2 e$$

$$\dot{s} = \ddot{e} + \lambda_2 \dot{e}$$

$$\dot{s} = \ddot{\phi}_d - \ \dot{\theta}\dot{\psi}\ \frac{I_y - I_z}{I_x} + \frac{I_p}{I_x}\ \Omega\ \dot{\theta} - \frac{1}{I_x}u_2 + \lambda_2 \dot{e}$$

$$s\dot{s} \leq -K_2|s|$$

$$u_2 = I_x\ddot{\phi}_d - \ \dot{\theta}\dot{\psi}\ (I_y - I_z) + I_p\ \Omega\ \dot{\theta} + I_x\lambda_2\dot{e} + I_x u_{2r}$$

$$u_2 = I_x\ddot{\phi}_d - \ \dot{\theta}\dot{\psi}\ (I_y - I_z) + I_p\ \Omega\ \dot{\theta} + I_x\lambda_2\dot{e} + I_x u_{2r}$$

$$-s(u_{2r}) \leq -K_2|s|$$

$$u_{2r} = K_2 sign(s)$$

$$u_2 = I_x\ddot{\phi}_d - \ \dot{\theta}\dot{\psi}\ (I_y - I_z) + I_p\ \Omega\ \dot{\theta} + I_x\lambda_2\dot{e} + I_x K_2 sign(s)$$

Calculations for u3

$$e = \theta_d - \theta$$
$$\dot{e} = \dot{\theta}_d - \dot{\theta}$$
$$\ddot{e} = \ddot{\theta}_d - \ddot{\theta} = \ddot{\theta}_d - \ \dot{\phi}\dot{\psi}\ \frac{I_z - I_x}{I_y} - \frac{I_p}{I_y}\ \Omega\ \dot{\phi} - \frac{1}{I_y}u_3$$
$$s = \dot{e} + \lambda_3 e$$
$$\dot{s} = \ddot{e} + \lambda_3\dot{e} = \ddot{\theta}_d - \ \dot{\phi}\dot{\psi}\ \frac{I_z - I_x}{I_y} - \frac{I_p}{I_y}\ \Omega\ \dot{\phi} - \frac{1}{I_y}u_3 + \lambda_3\dot{e}$$
$$s\dot{s} \leq -K_3|s|$$
$$u_3 = I_y\ddot{\theta}_d - \ \dot{\phi}\dot{\psi}\ (I_z - I_x) - I_p\ \Omega\ \dot{\phi} + I_y\lambda_3\dot{e} + I_y u_{3r}$$
$$-s(u_{3r}) \leq -K_3|s|$$
$$u_{3r} = K_3 sign(s)$$
$$u_3 = I_y\ddot{\theta}_d - \ \dot{\phi}\dot{\psi}\ (I_z - I_x) - I_p\ \Omega\ \dot{\phi} + I_y\lambda_3\dot{e} + I_y K_3 sign(s)$$

By using the *sat* function in place of *sign* function, it was seen that we could avoid chattering.

$$u_3 = I_y\ddot{\theta}_d - \ \dot{\phi}\dot{\psi}\ (I_z - I_x) - I_p\ \Omega\ \dot{\phi} + I_y\lambda_3\dot{e} + I_y K_3 sat(s, \rho_3)$$

Calculations for u4

$$e = \psi_d - \psi$$
$$\dot{e} = \dot{\psi}_d - \dot{\psi}$$
$$\ddot{e} = \ddot{\psi}_d - \ddot{\psi} = \ddot{\psi}_d - \ \dot{\phi}\dot{\theta}\ \frac{I_x - I_y}{I_z} - \frac{1}{I_z}u_4$$
$$s = \dot{e} + \lambda_4 e$$
$$\dot{s} = \ddot{e} + \lambda_4\dot{e} = \ddot{\psi}_d - \ \dot{\phi}\dot{\theta}\ \frac{I_x - I_y}{I_z} - \frac{1}{I_z}u_4 + \lambda_4\dot{e}$$
$$s\dot{s} \leq -K_4|s|$$
$$u_4 = I_z\ddot{\psi}_d - \ \dot{\phi}\dot{\psi}\ (I_x - I_y) + I_z\lambda_4\dot{e} + I_z u_{4r}$$
$$-s(u_{4r}) \leq -K_4|s|$$
$$u_{4r} = K_4 sign(s)$$
$$u_4 = I_z\ddot{\psi}_d - \ \dot{\phi}\dot{\psi}\ (I_x - I_y) + I_z\lambda_4\dot{e} + I_z K_4 sign(s)$$

By using the *sat* function in place of *sign* function, it was seen that we could avoid chattering.

$$u_4 = I_z\ddot{\psi}_d - \ \dot{\phi}\dot{\psi}\ (I_x - I_y) + I_z\lambda_4\dot{e} + I_z K_4 sat(s, \rho_4)$$

Design and Tuning Parameters

Design Parameters	Values
k1	12
k2	380
k3	400
k4	4

Increasing the K values will make the respective control inputs (i.e. u values) aggressive and it will lead to faster convergence.

Design Parameters	Values
K_{p1}	50
K_{d1}	10

This affects the θ_d values, and thus, they affect the error seen in the x-direction.

Design Parameters	Values
K_{p2}	30
K_{d2}	10

This affects the ϕ_d values, and thus, they affect the error seen in the y-direction.

Design Parameters	Values
λ_1	7
λ_2	9
λ_3	9
λ_4	7

The λ values will make the results aggressive exponentially towards the origin. Thus, increasing the values will ensure converging faster to the desired trajectory.

Boundary Layer Parameter ($\rho = 0$): Rho (ρ) is the boundary layer parameter that increases the robustness of the controller by adding a tracing error. To reduce chattering we tried multiple values of ρ and set it to 0.9 to reduce chattering.

Part 3 : Code Explanation

Our code consists of two python files (*traj.py* and *Visualize.py*) and a MATLAB file (*traj.m*)

1. First, we calculate and plot the desired trajectories using the *traj.m* file. It takes the initial position, final position, initial time and final time as inputs and gives us the desired position, velocity, and acceleration in the X, Y, and Z direction.

2. The *traj.py* file consists of the major part of our control implementation. We hard-coded the desired trajectory values obtained from the *traj.m* file.

3. The signum function is used to implement the boundary layer for the sliding mode controller. It also acts as the saturation function which improves the signum fn. and helps in reducing the chattering observed in our controller.

4. Using the system dynamics, equations of motion, and the equations derived above, we obtain the values of $u_1.u_2, u_3, u_4$. We multiply the u matrix with the allocation matrix to obtain the rotor speeds which are then relayed to Crazyflie.

   Note: As mentioned in the document, we also wrap the roll, pitch, and yaw errors between $[-\pi,\pi]$.

5. The *Visualize.py* file is used to Visualize the 3D plot of the tracked and the desired trajectory.

Part 4 : Results & Plots

Previously, the motion in the x direction was causing errors in the calculation of ψ, which in turn affected the accuracy of our trajectory tracking. However, we were able to overcome this issue by switching to an extended version of the controller, which made our trajectory tracking independent of ψ errors. As a result, we achieved improved tracking performance. Despite expecting ψ errors to be zero, we still observed a slight deviation in the trajectory. The following are the results.

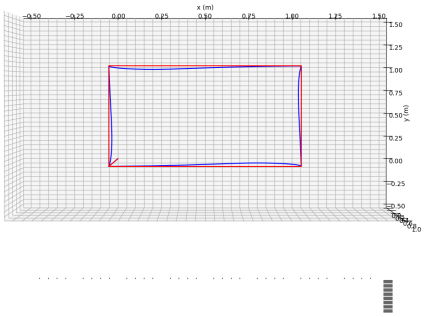

Figure 4: Top View of the Desired Trajectories

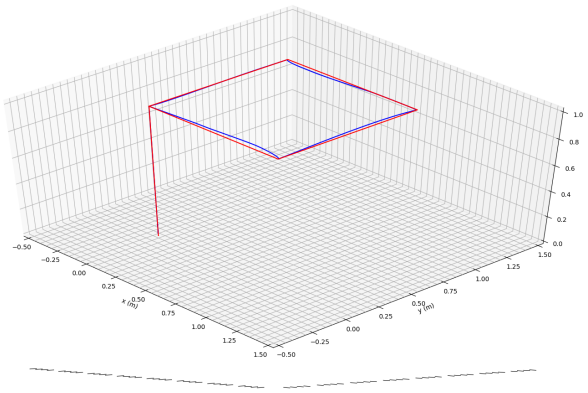

Figure 5: Orthogonal View of the Desired Trajectories

In order to achieve convergence of the actual trajectory of the Quadrotor with the desired trajectory, which was calculated using a quintic fifth-order equation, tuning of the controller was necessary. We were able to accomplish this by ensuring that the trajectory reached the sliding surface and slid on it until convergence, while also avoiding chattering by implementing a boundary layer. After carefully tuning the relevant parameters, we successfully achieved convergence of the actual trajectory with the desired one. Overall, we are satisfied with the performance of the designed controller.