RBE 502: Robot Control Project
Sliding Mode Controller Design for 3D Trajectory Tracking of Quadcopter

Ankush Singh Bhardwaj   Anuj Pradeep Pai Raikar

May 02, 2023

1 Generate Quintic Trajectories for Translational Coordinates

Figure 1: Quadcopter

$$q = \begin{bmatrix} x & y & z & \phi & \theta & \psi \end{bmatrix}^T$$

Figure 2: State

$$u = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 \end{bmatrix}$$

Figure 3: Control Inputs

Position, Velocity and Acceleration Trajectories are given by:
$x = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5$
$\dot{x} = a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4$
$\ddot{x} = 2a_2 + 6a_3 t + 12a_4 t^2 + 20a_5 t^3$

The velocity and acceleration at each waypoint must be equal to zero

$$p_0 = (0,0,0) \text{ to } p_1 = (0,0,1) \text{ in 5 seconds}$$
$$p_1 = (0,0,1) \text{ to } p_2 = (1,0,1) \text{ in 15 seconds}$$
$$p_2 = (1,0,1) \text{ to } p_3 = (1,1,1) \text{ in 15 seconds}$$
$$p_3 = (1,1,1) \text{ to } p_4 = (0,1,1) \text{ in 15 seconds}$$
$$p_4 = (0,1,1) \text{ to } p_5 = (0,0,1) \text{ in 15 seconds}$$

Figure 4: Waypoints to be Visited

a = inv (A) * b

Figure 5: Equations for Solving Trajectory

$$A = \begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 & t_0^4 & t_0^5 \\ 0 & 1 & 2t_0 & 3t_0^2 & 4t_0^3 & 5t_0^4 \\ 0 & 0 & 2 & 6t_0 & 12t_0^2 & 20t_0^3 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 \\ 0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3 \end{bmatrix}$$

The trajectory equations are:
From P_0 to P_1
$z = 0.00192t^5 - 0.024t^4 + 0.08t^3$
$\dot{z} = 0.0096t^4 - 0.096t^3 + 0.24t^2$
$\ddot{z} = 0.0384t^3 - 0.288t^2 + 0.48t$

From P_1 to P_2
$x = 0.0000079t^5 - 0.0002963t^4 + 0.002963t^3$
$\dot{x} = 0.0000395t^4 - 0.0011852t^3 + 0.0089t^2$
$\ddot{x} = 0.000158t^3 - 0.00356t^2 + 0.0178t$

From P_2 to P_3
$y = 0.0000079t^5 - 0.0002963t^4 + 0.002963t^3$
$\dot{y} = 0.0000395t^4 - 0.0011852t^3 + 0.0089t^2$
$\ddot{y} = 0.000158t^3 - 0.00356t^2 + 0.0178t$

From P_3 to P_4
$x = -0.0000079t^5 + 0.0002963t^4 - 0.002963t^3 + 1.0$
$\dot{x} = -0.0000395t^4 + 0.0011852t^3 - 0.0089t^2$
$\ddot{x} = -0.000158t^3 + 0.00356t^2 - 0.0178t$

From P_4 to P_5
$y = -0.0000079t^5 + 0.0002963t^4 - 0.002963t^3 + 1.0$
$\dot{y} = -0.0000395t^4 + 0.0011852t^3 - 0.0089t^2$
$\ddot{y} = -0.000158t^3 + 0.00356t^2 - 0.0178t$

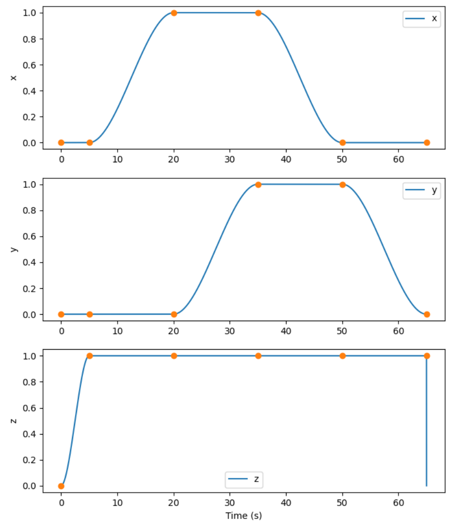


Figure 6: Desired Positional Trajectories

$$\psi_d = 0 \qquad \text{and} \qquad \dot{\phi}_d = \dot{\theta}_d = \dot{\psi}_d = 0 \qquad \text{and} \qquad \ddot{\phi}_d = \ddot{\theta}_d = \ddot{\psi}_d = 0$$

Figure 7: Desired Angular Velocities

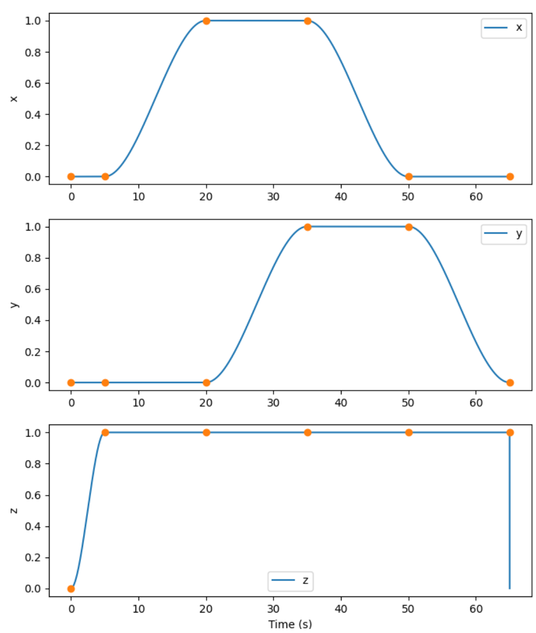


Figure 8: Desired Velocities Trajectories

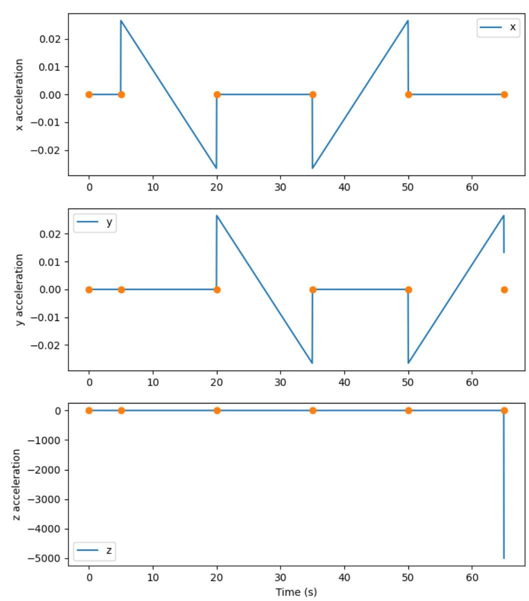
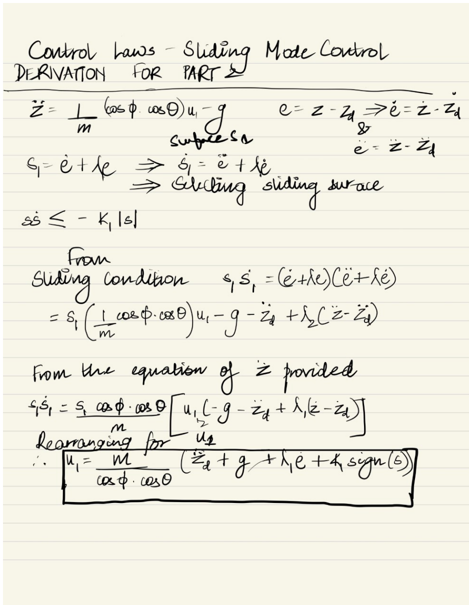


Figure 9: Desired Accelerations Trajectories

2 Design of Sliding Mode Control Laws


Control Laws - Sliding Mode Control
DERIVATION FOR PART 2

$\ddot z = \frac{1}{m}(\cos\phi \cos\theta)u_1 - g$ $\quad e = z - z_d \Rightarrow \dot e = \dot z - \dot z_d$ & $\ddot e = \ddot z - \ddot z_d$

surface s_1

$s_1 = \dot e + \lambda e \Rightarrow \dot s_1 = \ddot e + \lambda \dot e$

$\Rightarrow$ ~~Sliding~~ sliding surace

$s\dot s \le -K_1|s|$

From Sliding condition $s_1\dot s_1 = (\dot e+\lambda e)(\ddot e + \lambda \dot e)$

$= s_1\left(\frac{1}{m}\cos\phi\cdot\cos\theta\right)u_1 - g - \ddot z_d + \lambda_2(\dot z - \dot z_d)$

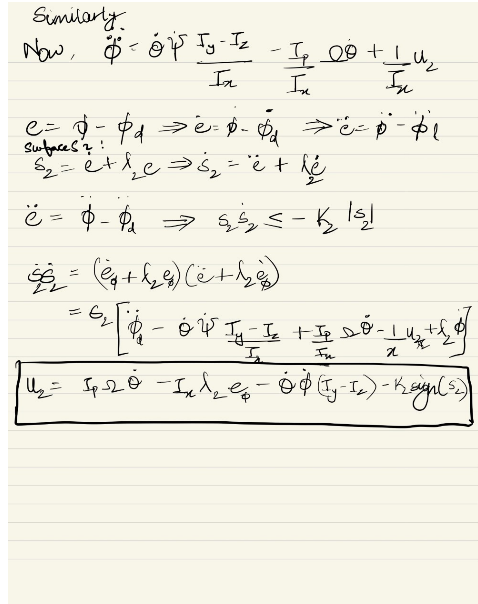
From the equation of $\ddot z$ provided

$s_1\dot s_1 = \frac{s_1\cos\phi\cdot\cos\theta}{m}\left[u_1(-g-\ddot z_d+\lambda_1(\dot z-\dot z_d)\right]$

Rearranging for u_1

$\therefore \boxed{u_1 = \frac{m}{\cos\phi\cdot\cos\theta}(\ddot z_d + g + \lambda_1 e + k_1 \, sign(s))}$

Figure 10: Derivation of Control Law for Z coordinates


Similarly

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

$e = \phi - \phi_d \Rightarrow \dot e = \dot\phi - \dot\phi_d \Rightarrow \ddot e = \ddot\phi - \ddot\phi_d$

surface s_2:

$s_2 = \dot e + \lambda_2 e \Rightarrow \dot s_2 = \ddot e + \lambda_2\dot e$

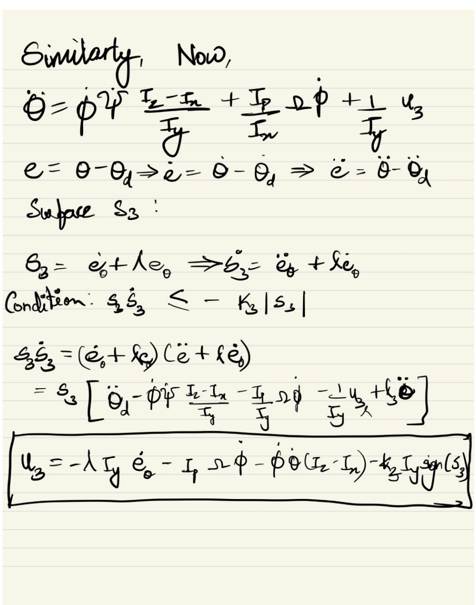
$\ddot e = \ddot\phi - \ddot\phi_d \Rightarrow s_2\dot s_2 \le -K_2|s_2|$

$s_2\dot s_2 = (\dot e_\phi + \lambda_2 e_\phi)(\ddot e + \lambda_2 \dot e_\phi)$

$= s_2\left[\ddot\phi_d - \dot\theta\dot\psi\frac{I_y-I_z}{I_x} + \frac{I_p}{I_x}\Omega\dot\theta - \frac{1}{I_x}u_2 + \lambda_2\dot\phi\right]$

$\boxed{u_2 = I_p\Omega\dot\theta - I_x\lambda_2 e_\phi - \dot\theta\dot\phi(I_y - I_z) - K_2 \, sign(s_2)}$

Figure 11: Derivation of Control Law for Roll Angle


Similarly, Now,

$\ddot\theta = \dot\phi\dot\psi\frac{I_z - I_x}{I_y} + \frac{I_p}{I_x}\Omega\dot\phi + \frac{1}{I_y}u_3$

$e = \theta - \theta_d \Rightarrow \dot e = \dot\theta - \dot\theta_d \Rightarrow \ddot e = \ddot\theta - \ddot\theta_d$

Surface s_3:

$s_3 = \dot e_\theta + \lambda e_\theta \Rightarrow \dot s_3 = \ddot e_\theta + \lambda\dot e_\theta$

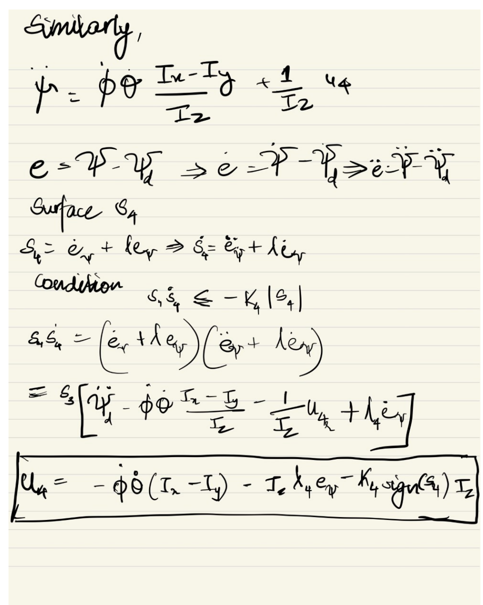
Condition: $s_3\dot s_3 \le -K_3|s_3|$

$s_3\dot s_3 = (\dot e_\theta + \lambda e_\theta)(\ddot e + \lambda\dot e_\theta)$

$= s_3\left[\ddot\theta_d - \dot\phi\dot\psi\frac{I_z-I_x}{I_y} - \frac{I_p}{I_y}\Omega\dot\phi - \frac{1}{I_y}u_3 + \lambda_3\ddot\theta\right]$

$\boxed{u_3 = -\lambda I_y \dot e_\theta - I_p\Omega\dot\phi - \dot\phi\dot\theta(I_z - I_x) - k_3 I_y \, sign(s_3)}$

Figure 12: Derivation of Control Law for Pitch Angle


Similarly,

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

$e = \psi - \psi_d \Rightarrow \dot e = \dot\psi - \dot\psi_d \Rightarrow \ddot e = \ddot\psi - \ddot\psi_d$

Surface s_4

$s_4 = \dot e_\psi + \lambda e_\psi \Rightarrow \dot s_4 = \ddot e_\psi + \lambda\dot e_\psi$

Condition $s_4\dot s_4 \le -K_4|s_4|$

$s_4\dot s_4 = (\dot e_\psi + \lambda e_\psi)(\ddot e_\psi + \lambda\dot e_\psi)$

$= s_3\left[\ddot\psi_d - \dot\phi\dot\theta\frac{I_x - I_y}{I_z} - \frac{1}{I_z}u_4 + \lambda_4\dot e_\psi\right]$

$\boxed{u_4 = -\dot\phi\dot\theta(I_x - I_y) - I_z\lambda_4 e_\psi - K_4 \, sign(s_4) I_z}$

Figure 13: Derivation of Control Law for Yaw Angle

3 ROS Node to Design the Control Design

A quadrotor controller is implemented in this Python script. It publishes rotor speeds to the Actuators subject and subscribes to an Odometry topic to obtain the quadrotor's present state. The controller creates the appropriate locations, velocities, and accelerations for the quadrotor to follow using a trajectory generator. It determines the rotor speeds necessary to get the quadrotor to the desired state using a feedback control loop.

The '*init*' constructor of the 'Quadrotor' class sets some initial settings and initializes the publisher and subscriber. The '$traj_e valuate$' method creates the ideal trajectory for the quadrotor to follow based on the current time 't'. Using a quintic polynomial, the desired locations, velocities, and accelerations are computed.

The gain values used for the controller is:
$\lambda_1 = 7$
$\lambda_2 = 12$ $\lambda_3 = 12$
$\lambda_4 = 8$

$K_1 = 10$
$K_2 = 120$ $K_3 = 120$
$K_4 = 10$

$K_p = 10$
$K_d = 120$

4 Visualize the Trajectory

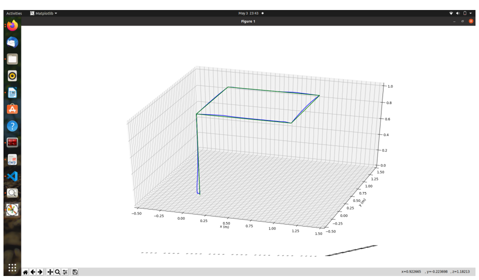


Figure 14: Viewpoint 1

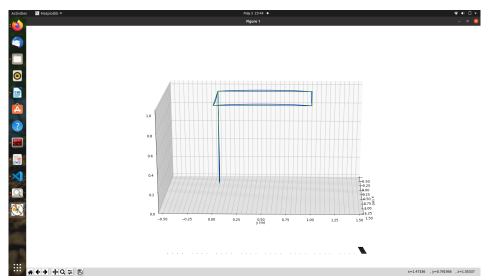


Figure 15: Viewpoint 2

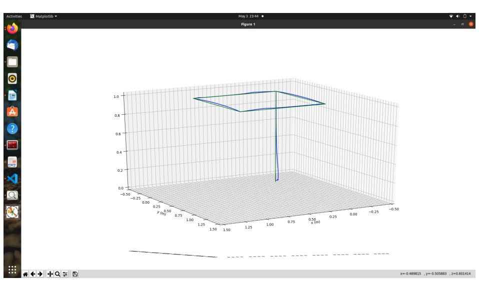


Figure 16: Viewpoint 3

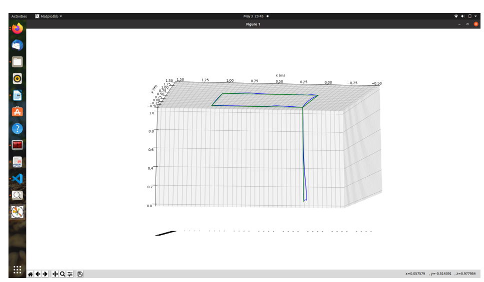


Figure 17: Viewpoint 4

5 Discussion

Initially when using the sign function, chattering was observed in the trajectories. Therefore we took the decision of replacing the sign function in the control laws with the saturation function.
Errors are noticed in the trajectory during lift off and trajectory is not tracked very accurately. Perhaps finer tuning of the gains would result in better tracking. The nature of gazebo of being a physics simulator also might play a role in accurate trajectory tracking.