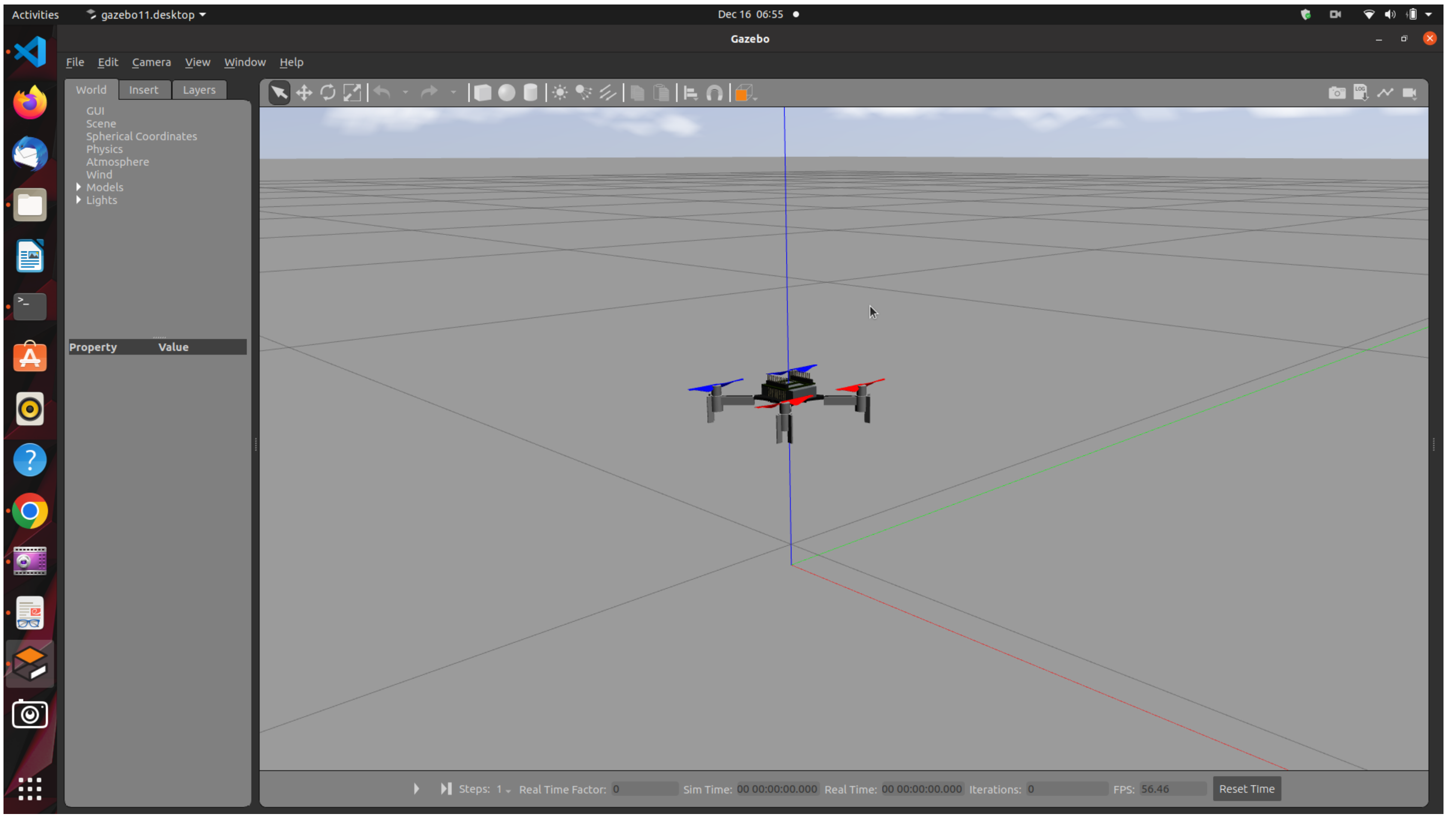
Vaishnavi Gejji | Himanshu Gautam
16 Dec 2022

RBE 502 — Robot Control

Final Project:

Robust Trajectory Tracking for Quadrotor UAVs using Sliding Mode Control

1.2 Crazyflie 2.0 Setup in Gazebo



1.4 Problem Statement

Part 1: Trajectory Generation:

- General form of quintic polynomial trajectory is:

$$q = a_5 t^5 + a_4 t^4 + a_3 t^3 + a_2 t^2 + a_1 t + a_0$$

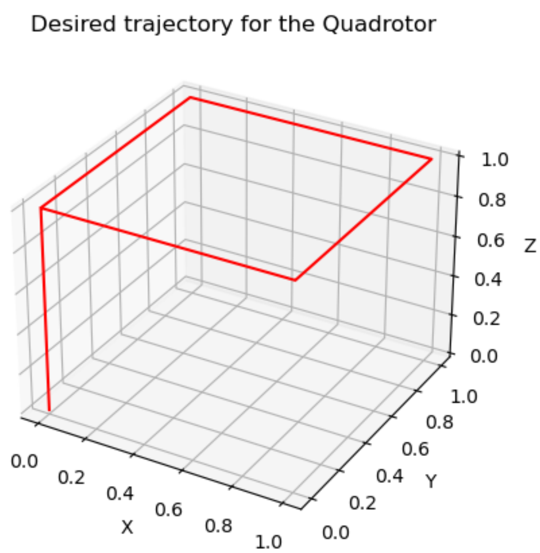
$$q\_dot = 5 a_5 t^4 + 4 a_4 t^3 + 3 a_3 t^2 + 2 a_2 t + a_1$$

$$q\_ddot = 20 a_5 t^3 + 12 a_4 t^2 + 6 a_3 t + 2 a_2$$

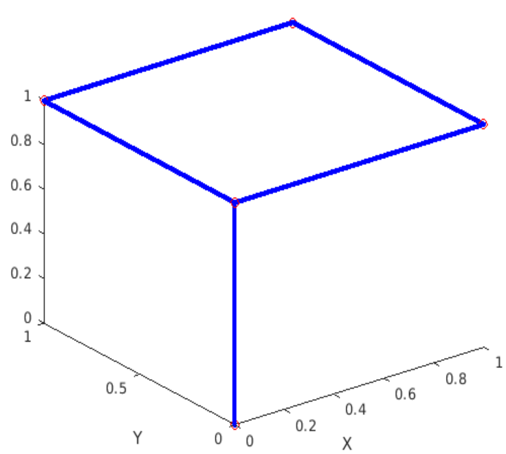
- To generate a quintic polynomial trajectory between two waypoints, we calculate the coefficients (a1,a2,a3,a4,a5) for every joint.

$$A = \begin{pmatrix} 1 & t_0 & t_0^2 & t_0^3 & t_0^4 & t_0^5 \\ 0 & 1 & 2\,t_0 & 3\,t_0^2 & 4\,t_0^3 & 5\,t_0^4 \\ 0 & 0 & 2 & 6\,t_0 & 12\,t_0^2 & 20\,t_0^3 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\ 0 & 1 & 2\,t_f & 3\,t_f^2 & 4\,t_f^3 & 5\,t_f^4 \\ 0 & 0 & 2 & 6\,t_f & 12\,t_f^2 & 20\,t_f^3 \end{pmatrix} \qquad a = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{pmatrix} \qquad b = \begin{pmatrix} q_0 \\ \dot{q}_0 \\ \ddot{q}_0 \\ q_f \\ \dot{q}_f \\ \ddot{q}_f \end{pmatrix}$$

- Using the above equations, we calculate each intermediate trajectory between waypoints for x,y,z coordinates separately.
- Each intermediate trajectory is appended to the respective x_traj, y_traj, z_traj lists to later obtain the complete desired trajectory.
- We generate the desired trajectory and visualize it using the python script *traj_generation.py.*



- Similarly, we visualize the waypoints and corresponding desired trajectory in MATLAB using the *quiticpolytraj(wayPoints, timePoints, tSamples)*function.
- The resulting trajectory is as follows:



Vaishnavi Gejji | Himanshu Gautam
16 Dec 2022

Part 2: Designing sliding mode control laws for the quadrotor

Control-affine form:

- It is ensured that the EOMs of the system are in control affine form which is represented as:

$$\ddot{q} = f(q, \dot{q}) + g(q, \dot{q})u$$

- For the given system the EOMs are already in this form with the f and g matrices as follows:

$$f = \begin{pmatrix} 0 \\ 0 \\ -g \\ -\frac{I_p\,\Omega\,\dot{\theta} - \dot{\psi}\,\dot{\theta}\,(I_y - I_z)}{I_x} \\ \frac{I_p\,\Omega\,\dot{\phi} - \dot{\phi}\,\dot{\psi}\,(I_x - I_z)}{I_y} \\ \frac{\dot{\phi}\,\dot{\theta}\,(I_x - I_y)}{I_z} \end{pmatrix} \qquad g = \begin{pmatrix} \frac{\sin(\phi)\sin(\psi) + \cos(\phi)\cos(\psi)\sin(\theta)}{m} & 0 & 0 & 0 \\ -\frac{\cos(\psi)\sin(\phi) - \cos(\phi)\sin(\psi)\sin(\theta)}{m} & 0 & 0 & 0 \\ \frac{\cos(\phi)\cos(\theta)}{m} & 0 & 0 & 0 \\ 0 & \frac{1}{I_x} & 0 & 0 \\ 0 & 0 & \frac{1}{I_y} & 0 \\ 0 & 0 & 0 & \frac{1}{I_z} \end{pmatrix}$$

Selecting sliding surface:

- For the quadrotor system, only the altitude (z), roll-pitch-yaw angles. Thus, we define our error dynamics as follows:

$$e = \begin{pmatrix} z - z_d \\ \phi - \phi_d \\ \theta - \theta_d \\ \psi - \psi_d \end{pmatrix} \qquad e\_dot = \begin{pmatrix} \dot{z} - \dot{z}_d \\ \dot{\phi} - \dot{\phi}_d \\ \dot{\theta} - \dot{\theta}_d \\ \dot{\psi} - \dot{\psi}_d \end{pmatrix}$$

- General form of sliding surface selected is:

$$s = \dot{e} + e\,\lambda$$

- This sliding surface is implemented for each control input.
- Also, given that desired yaw = desired angular velocities = desired angular accelerations = 0

s =

$$\begin{pmatrix} \dot{z} - \dot{z}_d + \lambda\ (z - z_d) \\ \dot{\phi} - \dot{\phi}_d + \lambda\ (\phi - \phi_d) \\ \dot{\theta} - \dot{\theta}_d + \lambda\ (\theta - \theta_d) \\ \dot{\psi} - \dot{\psi}_d - \lambda\ (\psi_d - \psi) \end{pmatrix}$$

s =

$$\begin{pmatrix} \dot{z} - \dot{z}_d + \lambda\ (z - z_d) \\ \dot{\phi} + \lambda\ (\phi - \phi_d) \\ \dot{\theta} + \lambda\ (\theta - \theta_d) \\ \dot{\psi} + \lambda\, \psi \end{pmatrix}$$

Designing u such that sliding condition is satisfied:

- The general form for calculating the control input is:

$$u = \frac{-f(q, \dot{q}) + \ddot{q}_d - \lambda \dot{e} + u_r}{g(q, \dot{q})}$$

- Here, lambda is a tuning parameter which determines how fast the system converges from the sliding surface to the origin.
- For convenience, lambda is set to 1 here.
- It is given that there are no unknown parameters. Thus the robust control term can be calculated as follows:

$$u_r = -(\rho + k) sat(\tfrac{s}{\phi_s})$$

- Final control inputs are as follows:

u =

$$\begin{pmatrix} \dfrac{\cos(\phi)\cos(\theta)\ \left(g + \ddot{z}_d - \lambda\ \left(\dot{z} - \dot{z}_d\right) - k\ \mathrm{sign}(s_1)\right)}{m} \\ -\dfrac{\lambda\,\dot{\phi} - \dfrac{I_p\,\Omega\,\dot{\theta} - \dot{\psi}\,\dot{\theta}\ (I_y - I_z)}{I_x} + k\ \mathrm{sign}(s_2)}{I_x} \\ -\dfrac{\lambda\,\dot{\theta} + \dfrac{I_p\,\Omega\,\dot{\phi} - \dot{\phi}\,\dot{\psi}\ (I_x - I_z)}{I_y} + k\ \mathrm{sign}(s_3)}{I_y} \\ -\dfrac{\lambda\,\dot{\psi} + k\ \mathrm{sign}(s_4) + \dfrac{\dot{\phi}\,\dot{\theta}\ (I_x - I_y)}{I_z}}{I_z} \end{pmatrix}$$

Solving the problem of chattering:

- Because of the definition of the sign function, the control law becomes discontinuous across s(t).
- Due to this, the controller flips sign as it reaches close to the sliding surface. This causes chattering and is undesirable.
- Hence, we replace *sign(s)* with *sat(s/phi).*

$$u = \begin{pmatrix} \frac{\cos(\phi)\cos(\theta)}{m}\left(g + \ddot{z}_d - \lambda(\dot{z} - \dot{z}_d) - \mathrm{sat}\left(\frac{s_1}{\phi_s}\right)(k+\rho)\right) \\ -\frac{1}{I_x}\left(\lambda\dot{\phi} - \ddot{\phi}_d + \mathrm{sat}\left(\frac{s_2}{\phi_s}\right)(k+\rho) - \frac{I_p\Omega\dot{\theta}}{I_x} + \frac{\dot{\psi}\dot{\theta}(I_y - I_z)}{I_x}\right) \\ -\frac{1}{I_y}\left(\lambda\dot{\theta} - \ddot{\theta}_d + \mathrm{sat}\left(\frac{s_3}{\phi_s}\right)(k+\rho) + \frac{I_p\Omega\dot{\phi}}{I_y} - \frac{\dot{\phi}\dot{\psi}(I_x - I_z)}{I_y}\right) \\ -\frac{1}{I_z}\left(\lambda\dot{\psi} - \ddot{\psi}_d + \mathrm{sat}\left(\frac{s_4}{\phi_s}\right)(k+\rho) + \frac{\dot{\phi}\dot{\theta}(I_x - I_y)}{I_z}\right) \end{pmatrix}$$

- Design and control parameters used in the control laws:
  - k => how fast does the system converge to sliding surface = tuning parameter
  - rho => bounds for the unknown parameters
  - phi_s => boundary layer
  - lambda => determines how fast the system converges from sliding surface to origin

- Values chosen for parameters:
  Kp = 100
  Kd = 5
  lambda_1 = lambda_z = 5
  lambda_2 = lambda_phi = 12
  lambda_3 = lambda_theta = 12
  lambda_4 = lambda_psi = 5
  K1 = 10
  K2 = 140
  K3 = 140
  K4 = 25

Calculations for sliding mode control design:

Implementing control on $\ddot{z}$, $\ddot{\phi}$, $\ddot{\theta}$, $\ddot{\psi}$ for the quadrotor.

① Sliding mode control design for z

Given: $\ddot{z} = \frac{1}{m}(\cos\phi . \cos\theta)u_1 - g$

Step 1: Selecting a sliding surface (s)

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

Error and error dynamics:

$$e = z - z_d \ , \ \dot{e} = \dot{z} - \dot{z}_d \ , \ \ddot{e} = \ddot{z} - \ddot{z}_d$$

$\therefore$ Sliding surface $S = \dot{e} + \lambda_z e$
$= \dot{z} - \dot{z}_d + \lambda_z (z - z_d)$

$\therefore \ \dot{s} = \ddot{e} + \lambda_z \dot{e}$
$= \ddot{z} - \ddot{z}_d + \lambda_z (\dot{z} - \dot{z}_d)$

Checking if valid sliding surface:
(i) $\dot{s}$ contains $\ddot{z}$. $\therefore$ $\dot{s}$ contains u
(ii) $s \rightarrow 0 \Rightarrow e \rightarrow 0, \ \dot{e} \rightarrow 0, \ \ddot{e} \rightarrow 0$

$\therefore$ S is a valid surface.

Step 2: Designing u such that it satisfies sliding condition

Sliding condition : $s\dot{s} \leq -k|s| \ , \ k > 0$

$\therefore \ s\dot{s} = s[\ddot{z} - \ddot{z}_d + \lambda \dot{e}]$

$$= s\left[\frac{1}{m}(\cos\phi . \cos\theta)u_1 - g - \ddot{z}_d + \lambda \dot{e}\right]$$

Taking the coeff of u outside:

$$s\dot{s} = \frac{s(\cos\phi.\cos\theta)}{m}\left[u_1 - \frac{m(g+\ddot{z}_d - \lambda\dot{e})}{\cos\phi.\cos\theta} \right]$$

$$u_1 = \frac{-m}{\cos\phi.\cos\theta}\left(\lambda\dot{e} - \ddot{z}_d - g + k_1 \,\mathrm{sgn}(s)\right)$$

Substituting u_1 in $s\dot{s}$, we get the following:

$$s\dot{s} = s(-k_1\,\mathrm{sgn}(s))$$

For sliding condition to be satisfied,

$$s\dot{s} = -k_1|s| < -k|s|$$

This will be satisfied if : $k_1 > k > 0$

To avoid chattering, we use $\mathrm{sat}\left(\frac{s}{\phi}\right)$

Step 3: Control law for z

$$\therefore u_1 = \frac{-m}{\cos\phi.\cos\theta}\left[\lambda\dot{e} - \ddot{z}_d - g\right] - \underbrace{\boxed{\frac{m}{\cos\phi.\cos\theta}}\,.\,k_1}_{k_z}\,\mathrm{sat}\left(\frac{s}{\phi_1}\right)$$

Vaishnavi Gejji | Himanshu Gautam
16 Dec 2022

(2) Sliding mode control for ϕ

Given: $\ddot{\phi} = \dot{\theta}\dot{\psi}\dfrac{I_y - I_z}{I_x} - \dfrac{I_p}{I_x}\Omega\dot{\theta} + \dfrac{1}{I_x}u_2$

Step 1: Sliding surface (s)

As shown in part (1): $s = \dot{e} + \lambda e$

$$\dot{s} = \ddot{e} + \lambda\dot{e} = \ddot{\phi} - \ddot{\phi}_d + \lambda\dot{e}$$

Error dynamics: $e = \phi - \phi_d$, $\dot{e} = \dot{\phi} - \dot{\phi}_d$, $\ddot{e} = \ddot{\phi} - \ddot{\phi}_d$

Step 2: Designing u to satisfy sliding condition

Sliding condition: $s\dot{s} \leq -k|s|$, $k>0$

$$s\dot{s} = s\left[\ddot{\phi} - \ddot{\phi}_d + \lambda\dot{e}\right]$$

$$= s\left[\dot{\theta}\dot{\psi}\frac{I_y - I_z}{I_x} - \frac{I_p}{I_x}\Omega\dot{\theta} + \frac{1}{I_x}u_2 - \ddot{\phi}_d + \lambda\dot{e}\right]$$

Taking out the u coefficient:

$$\therefore s\dot{s} = \frac{s}{I_x}\left[\dot{\theta}\dot{\psi}(I_y - I_z) - I_p\Omega\dot{\theta} + (\lambda\dot{e} - \ddot{\phi}_d)I_x + u_2\right]$$

$$u_2 = I_x\left[\frac{\dot{\theta}\dot{\psi}(I_z - I_y)}{I_x} + \frac{I_p\Omega\dot{\theta}}{I_x} + \ddot{\phi}_d - \lambda\dot{e} - k_2\,\mathrm{sgn}(s)\right]$$

Substituting u_2 in $s\dot{s}$, we get the following:

$$s\dot{s} = s\left[-k_2\,\mathrm{sgn}(s)\right] = -k_2|s|$$

For satisfying the sliding condition:

$$s\dot{s} = -k_2|s| \leq -k|s|, \quad k>0$$

$\therefore$ $k_2 > k > 0$ will satisfy the condition

* Replacing sgn(s) with sat(s/ϕ) to avoid chattering problem.

Step 3: Control law for ϕ

$$u_2 = -I_x\left[\frac{\dot{\theta}\dot{\psi}(I_y - I_z)}{I_x} - \frac{I_p\Omega\dot{\theta}}{I_x} - \ddot{\phi}_d + \lambda\dot{e}\right] - \underbrace{\boxed{I_x k_2}}_{k_\phi}\mathrm{sat}\left(\frac{s}{\phi_2}\right)$$

Vaishnavi Gejji | Himanshu Gautam
16 Dec 2022

③ Sliding mode control for θ

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

Step 1: Sliding surface (s)

As shown in part ①: $\boxed{s = \dot{e} + \lambda e}$

$$\dot{s} = \ddot{e} + \lambda\dot{e} = \ddot{\theta} - \ddot{\theta}_d + \lambda\dot{e}$$

Error dynamics: $e = \theta - \theta_d$, $\dot{e} = \dot{\theta} - \dot{\theta}_d$, $\ddot{e} = \ddot{\theta} - \ddot{\theta}_d$

Step 2: Designing u such that sliding condition is satisfied

Sliding condition: $s\dot{s} \leq -k|s|$, $k > 0$

$$s\dot{s} = s\left[\ddot{\theta} - \ddot{\theta}_d + \lambda\dot{e}\right]$$

$$s\dot{s} = s\left[\dot{\phi}\dot{\psi}\frac{I_z - I_x}{I_y} + \frac{I_p}{I_y}\Omega\dot{\phi} + \frac{1}{I_y}u_3 - \ddot{\theta}_d + \lambda\dot{e}\right]$$

let $$u_3 = -I_y\left[\dot{\phi}\dot{\psi}\frac{I_z - I_x}{I_y} + \frac{I_p}{I_y}\Omega\dot{\phi} - \ddot{\theta}_d + \lambda\dot{e} + K_3\,\mathrm{sgn}(s)\right]$$

Substituting u_3 in $s\dot{s}$, we get:

$$s\dot{s} = s\left[-K_3\,\mathrm{sgn}(s)\right] = -K_3|s|$$

∴ For satisfying the sliding condition, select $K_3 > k > 0$

Step 3: Control law for θ:

$$u_3 = -I_y\left[\dot{\phi}\dot{\psi}\frac{I_z - I_x}{I_y} + \frac{I_p}{I_y}\Omega\dot{\phi} - \ddot{\theta}_d + \lambda\dot{e}\right] - \underbrace{\boxed{I_y K_3}}_{K_\theta}\,\mathrm{sat}\left(\frac{s}{\phi_3}\right)$$

* Replacing sgn(s) with sat(s/ϕ) to avoid chattering problem.

④ Sliding mode control for ψ

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

Step 1: Sliding surface (s)

As shown in part ①: $s = \dot{e} + \lambda e$

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

Error dynamics: $e = \psi - \psi_d$, $\dot{e} = \dot{\psi} - \dot{\psi}_d$, $\ddot{e} = \ddot{\psi} - \ddot{\psi}_d$

Step 2: Designing u such that sliding condition is satisfied

Sliding condition: $s\dot{s} \leq -k|s|$, $k>0$

$$\therefore s\dot{s} = s[\ddot{\psi} - \ddot{\psi}_d + \lambda\dot{e}]$$

$$= s\left[\dot{\phi}\dot{\theta}\frac{I_x - I_y}{I_z} + \frac{1}{I_z}u_4 - \ddot{\psi}_d + \lambda\dot{e}\right]$$

let $u_4 = -I_z\left[\dot{\phi}\dot{\theta}\dfrac{I_x - I_y}{I_z} - \ddot{\psi}_d + \lambda\dot{e} + k_4\,\mathrm{sgn}(s)\right]$

Substituting u_4 in $s\dot{s}$, we get:

$$s\dot{s} = s[-k_4\,\mathrm{sgn}(s)] = -k_4|s|$$

$\therefore$ For satisfying the sliding condition, select $k_4 > k > 0$

Step 3: Control law for ψ:

$$u_4 = -I_z\left[\dot{\phi}\dot{\theta}\frac{I_x - I_y}{I_z} - \ddot{\psi}_d + \lambda\dot{e}\right] - \underbrace{\boxed{I_z k_4}}_{k_\psi}\,\mathrm{sat}\left(\frac{s}{\phi_4}\right)$$

* Replacing sgn(s) with sat(s/ϕ) to avoid chattering problem.

$$\left.\begin{aligned} \psi_d &= 0 \\ \dot{\phi}_d = \dot{\theta}_d &= \dot{\psi}_d = 0 \\ \ddot{\phi}_d = \ddot{\theta}_d &= \ddot{\psi}_d = 0 \end{aligned}\right\} \text{Given Assumptions}$$

$\therefore$ Final control laws are :

$$u_1 = \frac{-m}{\cos\phi \cdot \cos\theta}\left[\lambda\dot{e} - \ddot{z}_d - g\right] - \frac{m}{\cos\phi \cdot \cos\theta} \cdot K_1 \, sat\left(\frac{s}{\phi_1}\right)$$

$$u_2 = -I_x\left[\frac{\dot{\theta}\dot{\psi}(I_y - I_z)}{I_x} - \frac{I_p \Omega \dot{\theta}}{I_x} + \lambda\dot{e}\right] - I_x K_2 \, sat\left(\frac{s}{\phi_2}\right)$$

$$u_3 = -I_y\left[\dot{\phi}\dot{\psi}\,\frac{I_z - I_x}{I_y} + \frac{I_p}{I_y}\Omega\dot{\phi} + \lambda\dot{e}\right] - I_y K_3 \, sat\left(\frac{s}{\phi_3}\right)$$

$$u_4 = -I_z\left[\dot{\phi}\dot{\theta}\,\frac{I_x - I_y}{I_z} + \lambda\dot{e}\right] - I_z K_4 \, sat\left(\frac{s}{\phi_4}\right)$$

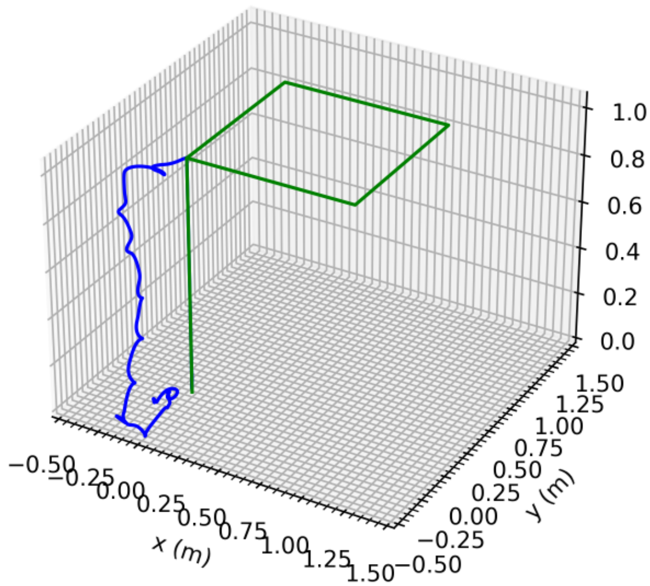
****note: the lambda for each u is different (lambda_1, lambda_2, lambda_3, lambda_4).***

Part 3: Implementing ROS node for performance evaluation

- The Quadrotor object is in the *ros_node.py* code.
- Here, we define the physical parameters and also tune different tunable parameters.
- Using some functions from *traj_generation.py* the desired values for position, velocity, acceleration are computed.
- Using the *forces_to_rp function*, force inputs are converted to desired roll and pitch angles using the F_x and F_y equations.
- In order to prevent chattering, *sat function* is used in developing the control laws.
- Within the *smc_control function* the control laws derived above are implemented. Also, from the control laws, motor speeds are computed using the allocation matrix.
- The actual and desired trajectory values are saved to the *log.pkl* file.

Part 4: Visualizing the results

- Using the given *visualize.py* script, desired and actual trajectories are plotted using the values from *log.pkl* file.
- The results are as shown below:
  (green: desired trajectory, blue: actual trajectory)



1.5 Performance Testing in Gazebo

- The performance test has been recorded and the video can be located in the submission folder.

Discussion about controller performance:

- The quadrotor reaches the second waypoint i.e. from p0 = (0,0,0) to p1 = (0,0,1) as expected, within the 5 sec duration.
- Also, at this point p1, the velocity of the quadrotor becomes zero.
- However, it then de-stabilizes and starts falling towards the X-Y plane.