RBE 502 : Robot Control Project Report

Sumeet Dinesh Shanbhag, Upasana Mahanti

May 2023

1 Introduction

The goal of this project is to develop a robust control scheme that enables a quad-rotor to track desired trajectories despite the presence of external disturbances. To achieve this, we will use the Crazyflie 2.0 Micro Aerial Vehicle as a testing platform in a simulated environment using the Gazebo simulator and ROS middleware.

2 Trajectory Generation

Since the question provides us with 5 waypoints to track and the time it should take between 2 successive waypoints, we can use the given data to generate a quintic trajectory for the quadrotor to track. Below is the formula to obtain the quintic trajectory.

$$\begin{bmatrix} a0 \\ a2 \\ a2 \\ a3 \\ a4 \\ a5 \end{bmatrix} = \begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 & t_0^4 & t_0^5 \\ 0 & 1 & 2t_0 & 3t_0^2 & 4t_0^3 & 5t_0^4 \\ 0 & 0 & 2 & 6t_0 & 12t_0^2 & 20t_0^3 \\ 1 & t_f & t_f^2 & t_f^3 & t_f^4 & t_f^5 \\ 0 & 1 & 2t_f & 3t_f^2 & 4t_f^3 & 5t_f^4 \\ 0 & 0 & 2 & 6t_f & 12t_f^2 & 20t_f^3 \end{bmatrix}^{-1} \begin{bmatrix} q_0 \\ \dot{q}_0 \\ \ddot{q}_0 \\ q_f \\ \dot{q}_f \\ \ddot{q}_f \end{bmatrix}$$

Where $a0$, $a1$, $a2$, $a3$, $a4$, and $a5$ are the coefficients of the quintic polynomial equation.

$$q(t) = at^5 + bt^4 + ct^3 + dt^2 + et + f$$

Here, q_0, $\dot{q}_0$, $\ddot{q}_0$, q_f, $\dot{q}_f$, and $\ddot{q}_f$ are the initial and final positions, velocities, and accelerations, respectively, and t_0 and t_f are the start and end times.

2.1 Plots

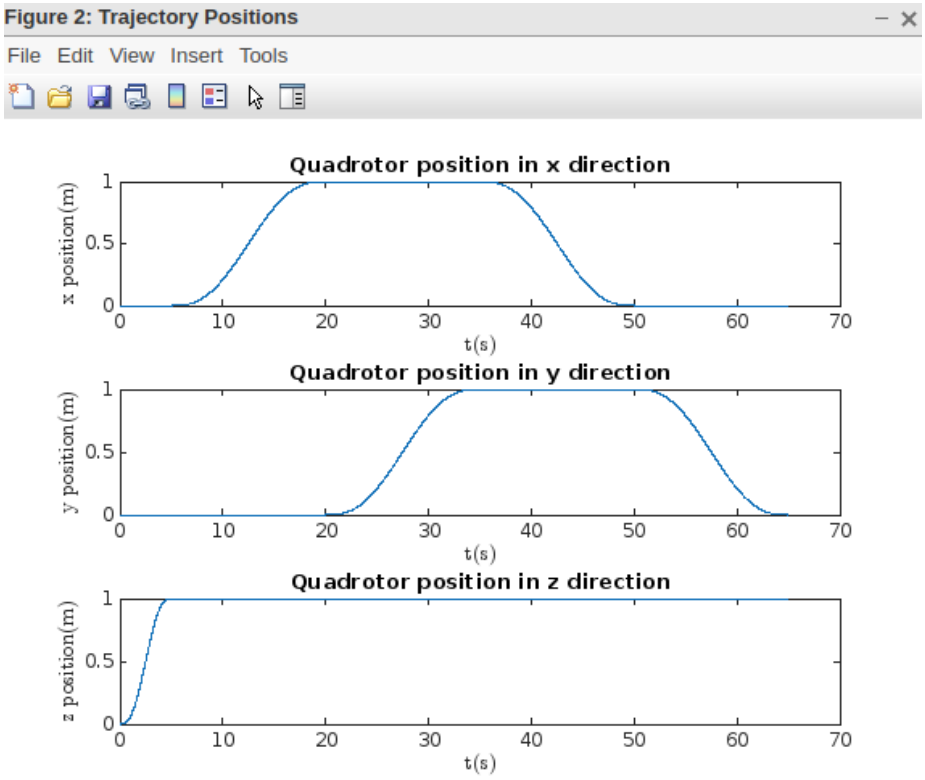

Figure 1: Desired Position Trajectories

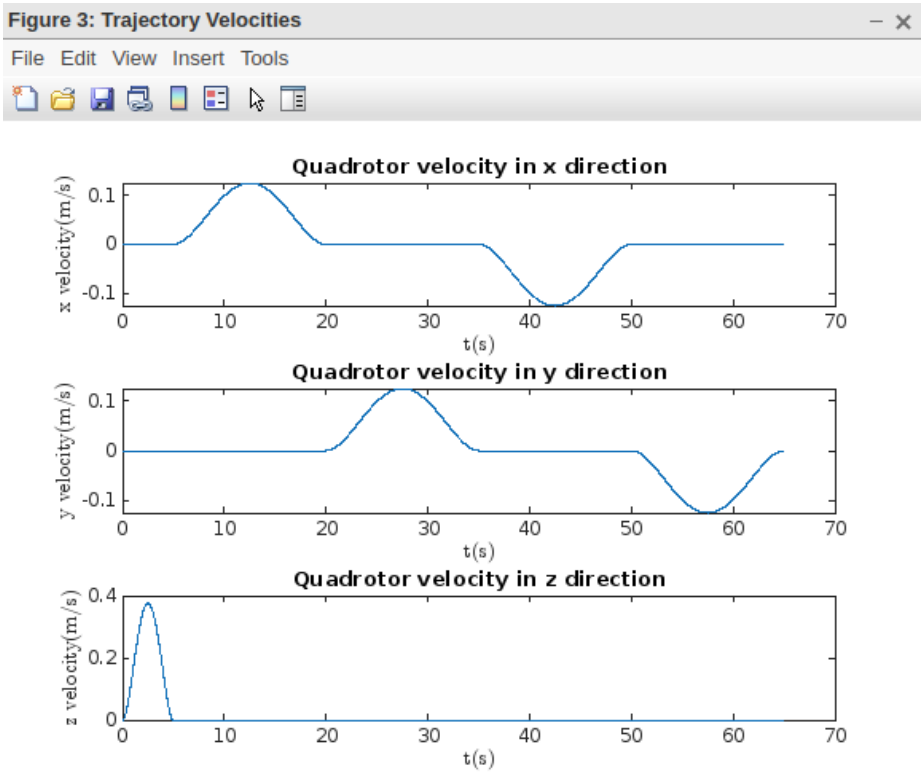

Figure 2: Desired Velocity Trajectories

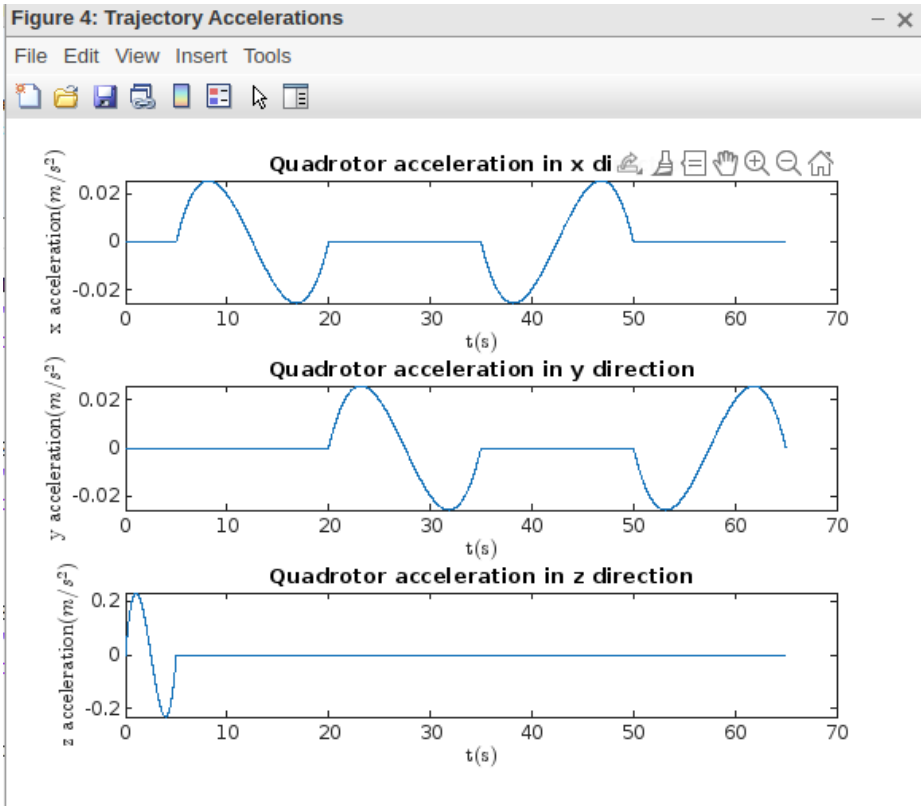

Figure 3: Desired Acceleration Trajectories

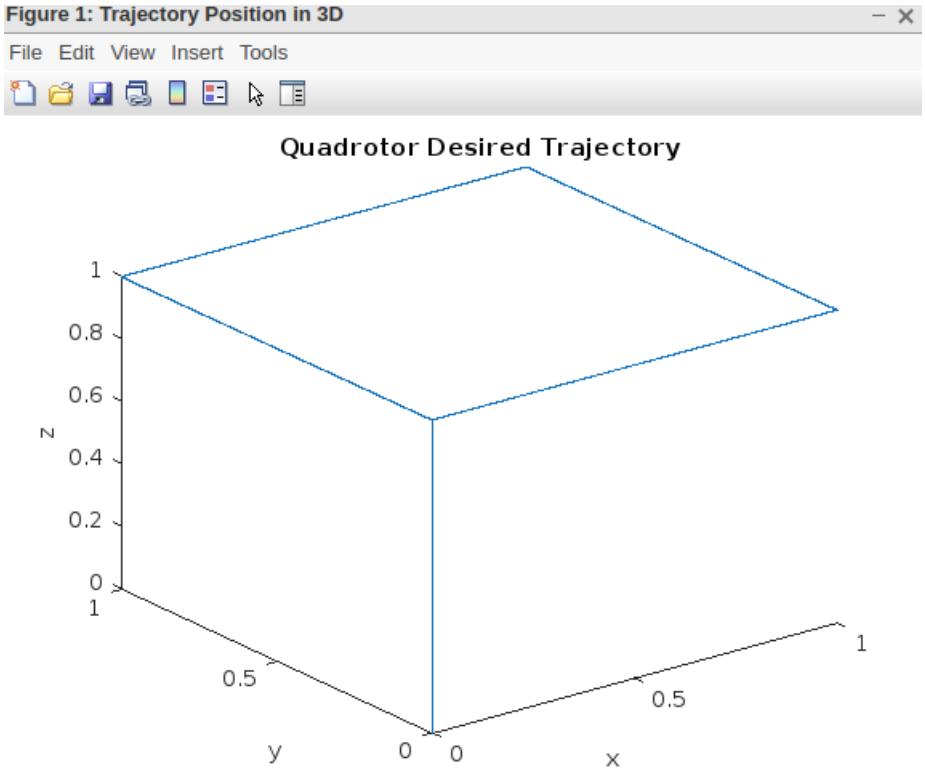

Figure 4: Desired Position Trajectories in 3D

3 Sliding Mode Control Law Derivation

3.1 Derivation

Given a second-order nonlinear system in the control-affine form

$$\ddot{q} = f(q, \dot{q}) + g(q, \dot{q})u \tag{1}$$

We define the error of the system as

$$e = q - q_d \tag{2}$$

$$\dot{e} = \dot{q} - \dot{q}_d \tag{3}$$

With surface equation

$$s = \dot{e} + \lambda * e \tag{4}$$

$$s\dot{s} = s * [f(q, \dot{q}) + g(q, \dot{q}) * u - \ddot{q}_d + \lambda * \dot{e}] \tag{5}$$

To satisfy the sliding condition

$$s\dot{s} \leq -K * |s| \tag{6}$$

We chose

$$u = \frac{(-\hat{f}(q, \dot{q}) + \ddot{q}_d - \lambda * \dot{e} + u_r)}{\hat{g}(q, \dot{q})} \tag{7}$$

Because we assume all the model parameters are known:

$$\hat{f}(q, \dot{q}) = f(q, \dot{q}), \quad \hat{g}(q, \dot{q}) = g(q, \dot{q}) \tag{8}$$

Thus,

$$s\dot{s} = u_r * s \tag{9}$$

Chose $u_r = -K * \text{sign}(s)$. Since $s\dot{s} = -K * \text{sign}(s) \leq -K * |s|$, the sliding condition is satisfied. Then, to remove chattering, replace $\text{sign}(s)$ with saturation function $\text{sat}(\frac{s}{\phi})$ where ϕ is the boundary layer around the sliding surface:

$$u_r = -K * \text{sat}(\frac{s}{\phi}) \tag{10}$$

3.2 Sliding controller design for each of the input

3.2.1 Input Z

To control the z direction, we design the control law for u_1 as follows,

$$u_1 = \frac{-f_z(q, \dot{q}) + \ddot{z}d - \lambda_z \dot{e}z + urz}{g_z(q, \dot{q})} \tag{11}$$

where,

$$f_z(q, \dot{q}) = -g, \tag{12}$$

$$g_z(q, \dot{q}) = \frac{\cos(\phi)\cos(\theta)}{m} \tag{13}$$

$$u_{rz} = -K_z \cdot \text{sat}\left(\frac{s_z}{\phi_z}\right) \tag{14}$$

3.2.2 Input ϕ, ψ and θ

After $u1$ has been calculated, we use the provided equations to calculate θ_d and ϕ_d.

$$F_x = m\left(-k_p\left(x - x_d\right) - k_d\left(\dot{x} - \dot{x}_d\right) + \ddot{x}_d\right),$$

$$F_y = m\left(-k_p\left(y - y_d\right) - k_d\left(\dot{y} - \dot{y}_d\right) + \ddot{y}_d\right),$$

$$\theta_d = \sin^{-1}\left(\frac{F_x}{u_1}\right)$$

$$\phi_d = \sin^{-1}\left(\frac{-F_y}{u_1}\right)$$

After u_1 has been calculated, we use the provided equations to calculate θ_d and ϕ_d. Then, we use the same approach as u_1 to obtain the control laws for u_2, u_3, and u_4:

$$u_2 = \frac{-f_\phi(q,\dot{q}) + \ddot{\phi}_d - \lambda_\phi^* \dot{e}_\phi + u_{r_\phi}}{g_\phi(q,\dot{q})} \quad \text{where}$$

$$f_\phi(q,\dot{q}) = \frac{\dot{\psi}\dot{\theta}(I_y - I_z)}{I_x} - \frac{I_p\dot{\theta}(w_1 - w_2 + w_3 - w_4)}{I_x} \quad \text{and} \quad g_\phi(q,\dot{q}) = 1/I_x$$

$$u_{r_\phi} = -K_\phi \cdot \text{sat}(\frac{s_\phi}{\phi_\phi})$$

$$u_3 = \frac{-f_\theta(q,\dot{q}) + \ddot{\theta}_d - \lambda_\theta^* \dot{e}_\theta u_{r_\theta}}{g_\theta(q,\dot{q})} \quad \text{where,}$$

$$g_\theta(q,\dot{q}) = \cos(\phi) \quad \text{and} \quad u_{r_\theta} = -K_\theta \cdot \text{sat}(\frac{s_\theta}{\phi_\theta})$$

where,

$$u_3 = \frac{-f_\theta(q,\dot{q}) + \theta\ddot{d} - \lambda_\theta * \dot{e}_\theta + u_{r,\theta}}{g_\theta(q,\dot{q})}$$

where,

$$g_\theta(q,\dot{q}) = \frac{1}{I_y}$$

$$u_{r,\theta} = -K_\theta * \text{sat}\left(\frac{s_\theta}{\phi_\theta}\right)$$

$$u_4 = \frac{-f_\psi(q,\dot{q}) + \psi\ddot{d} - \lambda_\psi * \dot{e}_\psi + u_{r,\psi}}{g_\psi(q,\dot{q})}$$

where,

$$f_\psi(q,\dot{q}) = \frac{\phi\dot{\theta}(I_x \dot{-} I_y)}{I_z}$$

$$g_\psi(q,\dot{q}) = \frac{1}{I_z}$$

$$u_{r,\psi} = -K_\psi * \text{sat}\left(\frac{s_\psi}{\phi_\psi}\right)$$

3.3 Controller parameters

- k_p and k_d are the tuning parameters for the forces acting on x and y direction.
- λ_z, K_z, and φ_z are the tuning parameters for z direction.
- λ_φ, K_φ, and φ_φ are the tuning parameters for roll angle φ.
- λ_θ, K_θ, and φ_θ are the tuning parameters for pitch angle θ.
- λ_ψ, K_ψ, and φ_ψ are the tuning parameters for yaw angle ψ.
- To simplify the tuning process, we picked the boundary layer parameter φ to be 1.0 for all control law.

4 ROS Code

The controller node for the quadrotor is referred to as the "quadrotor control" node. The `odom callback` function is utilized to receive up-to-date data from the quadrotor regarding its position in 3D space and its linear and angular velocities. This is achieved by subscribing to the `"/crazyflie2/ground truth/odometry"` topic. The trajectory that the quadrotor follows is saved for future use in plotting.

The `smc control` function uses the `traj evaluate` function to acquire the desired trajectory values for the specified time period. This function is crucial

to the implementation of the controller and includes the Sliding Mode Control Mechanism mentioned previously. The control law is employed to calculate all control inputs, which are then converted to motor speeds using the given allocation matrix. The resulting motor speeds are broadcasted to the `"/crazyflie2/command/motor speed"` topic to be consumed by the gazebo simulation.

$$\begin{bmatrix} \frac{1}{4k_F} & -\frac{\sqrt{2}}{4k_F l} & -\frac{\sqrt{2}}{4k_F l} & -\frac{1}{4k_M k_F} \\ \frac{1}{4k_F} & -\frac{\sqrt{2}}{4k_F l} & \frac{\sqrt{2}}{4k_F l} & \frac{1}{4k_M k_F} \\ \frac{1}{4k_F} & \frac{\sqrt{2}}{4k_F l} & \frac{\sqrt{2}}{4k_F l} & -\frac{1}{4k_M k_F} \\ \frac{1}{4k_F} & \frac{\sqrt{2}}{4k_F l} & -\frac{\sqrt{2}}{4k_F l} & \frac{1}{4k_M k_F} \end{bmatrix}$$

Figure 5: Allocation Matrix

5 Controller parameters tuning

Our tuning process began by selecting controller parameters as the average value of the suggested range.

$$\begin{aligned} k_p &= 100.0 \\ k_d &= 5.0 \\ \lambda_z &= 5.0 \\ K_z &= 10.0 \\ \lambda_\phi &= 12.5 \\ K_\phi &= 140.0 \\ \lambda_\theta &= 12.5 \\ K_\theta &= 140.0 \\ \lambda_\psi &= 5.0 \\ K_\psi &= 25.0 \end{aligned}$$

We first picked λ_ϕ and λ_θ, K_ϕ and K_θ being the same because the quad-rotor's inertia along the x axis and y axis were the same in the provided model parameters. However, after running the simulation, we can identify wobbling behaviour around y axis near the end of trajectory step 2 and step 4. Moreover, we didn't find the same wobbling behaviour around x axis. (This can be seen clearly from the recorded video and it's associated trajectory plot.) This might be the result of model mismatch between the provided inertia and actual inertia. Thus, we tuned λ_θ and K_θ to get rid of this wobbling behaviour. From experiment, we found that increasing λ_θ and decreasing K_θ at the same time reduces the

wobbling. We also noticed that tuning λ_z and K_z can help the quad-rotor to maintain its altitude while moving in the x and y direction. The final control parameters we used are as follows,

$$\begin{aligned} k_p &= 100.0 \\ k_d &= 5.0 \\ \lambda_z &= 10.0 \\ K_z &= 5.0 \\ \lambda_\phi &= 16.0 \\ K_\phi &= 141.0 \\ \lambda_\theta &= 20.0 \\ K_\theta &= 109.0 \\ \lambda_\psi &= 5.0 \\ K_\psi &= 25.0 \\ \text{boundary} &= 1.0 \end{aligned}$$

6 Results

The plot above shows the green desired trajectory and the blue trajectory obtained from the simulation. The controller was carefully tuned to enable accurate tracking of the desired trajectories with very little error. The quadrotor successfully visited all 5 waypoints with its velocity and acceleration being zero at each point. As there was no uncertainty in the physical parameters of the quadrotor, designing the controller was relatively easier. However, small errors in the trajectories could be attributed to two reasons. Firstly, we used the ϕ parameter to create a boundary region to prevent the chattering effect at the cost of slightly inaccurate convergence of trajectories. Secondly, the provided physical parameters may differ slightly from the actual physical parameters.

We also tested the hovering capability of the quadrotor after it reached the final point. We introduced small positional or rotational errors by dragging or rotating it in certain directions or orientations. The quadrotor was able to correct the introduced error and fly back to the original position and orientation, as seen in the attached videos. However, if the introduced error was too large, the quadrotor failed to stabilize and crashed.

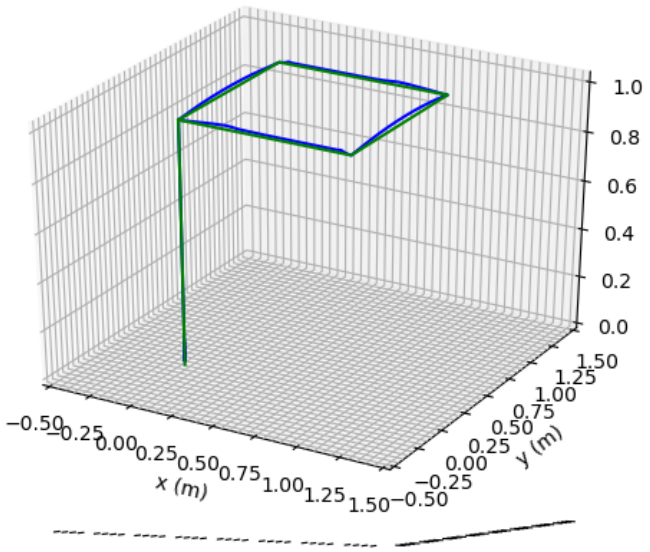

Figure 6: Actual Position Trajectory in 3D