Project: Icarus

Dynamical Modeling and Control of a QuadCopter

Abstract

This Project Consists of 3 phases:

1. Phase 1: Mathematical Modelling of the QuadCopter and Desiging of a Linear PID controller to control all 6 DoF of the QuadCopter Considering IMU Noise.
2. Phase 2: Simulating a flipping maneuver using the extracted Mathematical Model.
3. Phase 3: Designing a Non-Linear Sliding-Mode controller to track a desired trajectory under the influence of external disturbances and IMU noises.

Note: *The whole simulation process is done in MATLAB. And every algorithm is implemented from scratch.*

---

0. Introduction

The QuadCopter is a popular choice for aerial robotics research and applications due to its highly maneuverable capabilities, including vertical takeoff and landing, hovering, and multidirectional flight. In addition to its agility, the QuadCopter is relatively simple to construct and maintain, making it a cost-effective option for research and development in the field of aerial robotics. However, despite its advantages, the system is inherently unstable, necessitating the development of a robust mathematical model and controller. This project aims to derive a mathematical model of the system and design a control system using control theory to stabilize the QuadCopter and enable precise trajectory tracking.

The QuadCopter is a versatile aerial robotics platform that offers exceptional maneuverability in flight. It is capable of executing complex movements such as vertical takeoff and landing, hovering, and multi-directional flight. The construction and maintenance of the QuadCopter are relatively straightforward, making it an affordable option for research and development in the field of aerial robotics. However, the system's inherent instability requires the development of a robust mathematical model and controller to ensure stable flight and precise trajectory tracking.

---

1. Phase 1 - Dynamical Modelling and Implemetation of a Linear PID Controller

The QuadCopter is a rigid body with six degrees of freedom in space, defined by its center of mass position and Euler angles. It is propelled by four rotors, each modeled as a point mass with thrust force acting in the positive z-direction. The system is subject to

gravitational and aerodynamic forces, as well as environmental disturbances. The QuadCopter is assumed to be symmetric about the x and y axes for the sake of simplicity, and aerodynamic forces are assumed to be linearly proportional to the velocity of the QuadCopter. Additionally, the aerodynamic forces are negligible when compared to the thrust forces of the rotors. The system is assumed to be rigid, with rotors rigidly attached to the body of the QuadCopter. The QuadCopter is also assumed to be a point mass, with negligible rotor mass compared to the QuadCopter's mass.

This part consists of three main sections: kinematics, dynamics and control. The kinematics section describes the relationship between the angular velocity of the QuadCopter and the roll, pitch, and yaw angle rates. The dynamics section describes the forces and moments acting on the QuadCopter. Finally, the control section describes the design of a linear PID controller to control all six degrees of freedom of the QuadCopter.

1.1 Kinematics

The relation between the angular velocity of the quadcopter and the roll, pitch, and yaw angle rates can be expressed using the following equation (Diebel, 2006):

$$\omega = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\varphi & \cos\theta\sin\varphi \\ 0 & -\sin\varphi & \cos\theta\cos\varphi \end{bmatrix}^{-1} \begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\varphi\tan\theta & \cos\varphi\tan\theta \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \frac{\sin\varphi}{\cos\theta} & \frac{\cos\varphi}{\cos\theta} \end{bmatrix} \begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

where:

- ϕ, θ, and ψ are the roll, pitch, and yaw angles, respectively
- ω is the angular velocity of the quadcopter

  Note that the representation suffers from a sigularity when $ = $, which is known as gimbal lock. This can be avoided by using quaternions to represent the attitude of the quadcopter.

The relation between the translational velocity of the quadcopter and the roll, pitch, and yaw angles can be expressed using the following equation:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = R(\phi,\theta,\psi) \begin{bmatrix} \dot{\varphi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

where:

- $\dot{x}$, $\dot{y}$, and $\dot{z}$ are the translational velocities of the quadcopter in the inertial frame
- u, v, and w are the velocities of the quadcopter in the body frame
- $R(\phi,\theta,\psi)$ is the ZYX rotation matrix corresponding to the roll, pitch, and yaw angles

1.2 Dynamics

The motion of a quadcopter can be described using six degrees of freedom: three translational (surge, sway, heave) and three rotational (roll, pitch, yaw). The forces and moments acting on the quadcopter can be modeled using the following equations.

1.2.1 Translational Dynamics

There are three main forces acting on the quadcopter:

1. Thrust force generated by the rotors
2. Aerodynamic drag force
3. Gravitational force

The gravitational force is always directed downwards, while the thrust and drag forces are directed along the body axes of the quadcopter. The thrust and drag forces are functions of the angular velocity of the rotors, which is controlled by the flight controller. Using the Newton's equations, the translational motion of the quadcopter can be described by the following equations:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} + \frac{1}{m} R(\phi, \theta, \psi) \begin{bmatrix} 0 \\ 0 \\ T_\Sigma \end{bmatrix} + \frac{1}{m} F_D$$

where:

- m is the mass of the quadcopter
- g is the gravitational acceleration
- x, y, and z are the positions of the quadcopter in the inertial frame
- ϕ, θ, and ψ are the roll, pitch, and yaw angles, respectively
- $R(\phi, \theta, \psi)$ is the rotation matrix corresponding to the roll, pitch, and yaw angles
- T_Σ is the total thrust generated by the four rotors

The drag force is usually modelled as a function of the translational velocity of the quadcopter:

$$F_D = \begin{bmatrix} -D_x \dot{x} \\ -D_y \dot{y} \\ -D_z \dot{z} \end{bmatrix}$$

where D_x, D_y, and D_z are the drag coefficients in the x, y, and z directions, respectively.

1.2.2 Rotational Dynamics

Euler's equation for rotatinal motion:

$$I\dot{\omega} + \omega \times (I\omega) = \tau$$

$$\dot{\omega} = I^{-1}(\tau - \omega \times (I\omega)$$

where:

- I is the moment of inertia matrix
- ω is the angular velocity of the quadcopter
- τ is the total moment acting on the quadcopter

assuming that the quadcopter is symmetric about the x and y axes, the moment of inertia matrix will be diagonal and can be expressed as:

$$I = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}$$

where I_{xx}, I_{yy}, and I_{zz} are the moments of inertia about the x, y, and z axes, respectively. The quadcopter can be modelled as two thin rods of length l connected at the center.

> ***Note 1:*** For the sake of simplicity the angular motion of the quadcopter is expressed in the body frame, translational motion is expressed in the inertial frame. ***Note 2:*** All the system parameter are better be identified using system identification techniques for precision. And all the system parameters are assumed to be constant.

1.3.1 Thrusts and Torques

Each motor generates a thrust and a torque. The thrust is directed along the z axis of the body frame, while the torque is directed along the x and y axes of the body frame. Thrust force of each motor is modelled as a function of the square of the angular velocity of the motor:

$$T_i = k\omega_i^2$$

where k is a constant that depends on the propeller and motor characteristics.

Torque on each x, y and z axes are modelled as:

$$\tau = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} lk(\omega_2^2 - \omega_4^2) \\ lk(\omega_3^2 - \omega_1^2) \\ b(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{bmatrix}$$

The total thrust and torque generated by the rotors can be expressed as a single matrix equation:

$$\begin{bmatrix} T_\Sigma \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} k & k & k & k \\ 0 & -lk & 0 & lk \\ -lk & 0 & lk & 0 \\ b & -b & b & -b \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix}$$

where:

- k is a constant that depends on the propeller and motor characteristics
- l is the distance between the center of the quadcopter and the center of each rotor
- b is a constant that depends on the propeller and motor characteristics
- ω_1, ω_2, ω_3, and ω_4 are the angular velocities of the four rotors

---

1.3.2 Motor Dynamics

Each motor has a PWM pulse input which corresponds to the angular velocity of the motor. The relationship between the PWM pulse and the angular velocity can be expressed as quadratic function. Since, Icarus is a simulation project this relationship is not considered.

The Motors are modelled as first order systems with two time constants for the rise and fall in there speeds, since the motors are faster to speed up and slower to slow down. The motor dynamics can be expressed as:

$$\dot{\omega}_i = \begin{cases} P_{up}(\omega_{des} - \omega_i) & \text{for } \omega_{des} > \omega_i \\ P_{down}(\omega_{des} - \omega_i) & \text{otherwise.} \end{cases}$$

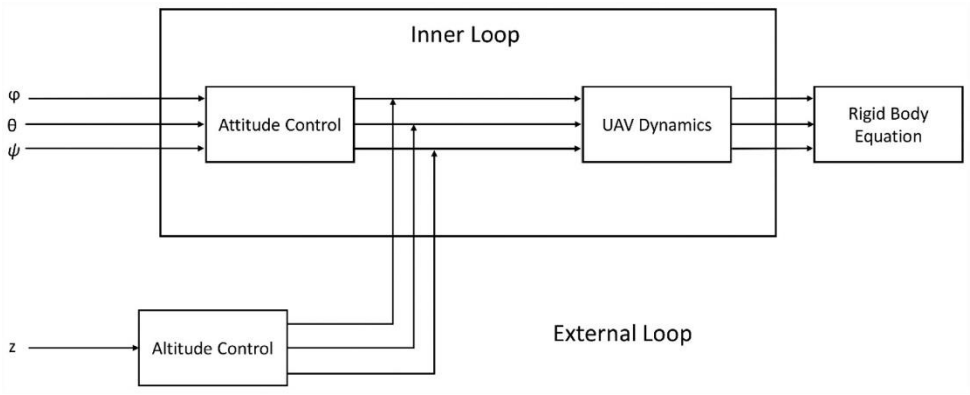
where:

- P_{up} and P_{down} are the motor time constants
- ω_{des} is the desired angular velocity of the motor

Motors are faster to speed up than to slow down. To model this, we can define a $P_{up} \geq P_{down}$. To bypass motor dynamics, we may set $\omega_{des_{1..4}} = \omega_{i_{1..4}}$. To ignore motor saturation, we may set $\omega_{max} = -\omega_{min} = \infty$.

1.4 Controller

1.4.1 Quadcopter Control Structure



At its most basic level, to be able to control and move the quadcopter, there must be a control system to control the thrust vector's amplitude and direction. To obtain this goal the problem is divided into two sub-problems:

1. **Attitude Control:** To control the orientation of the quadcopter in the 3D space.
2. **Altitude Control:** To control the altitude of the quadcopter in the 3D space.

The attitude control is basically controlling the moments acting on the quadcopter, in order to control the orientation of the thrust vector. In a similar manner the altitude control is controlling the thrust vector's amplitude, in order to control the altitude of the quadcopter.

The feedback needed for the attitude control system is the orientation of the quadcopter in the 3D space. This can be obtained by using an IMU sensor. The feedback needed for the altitude control system is the altitude of the quadcopter in the 3D space. However the height feedback can not be constructed simply and there are several ways that are usually used to obtain it.

1. Using Barometer
2. Odometry Sensors Containing a Laser Range Finder and a Camera
3. Using a Motion Capture System
4. Using a GPS
5. Using Visual Localization Systems Like SLAM
6. …

Although, as it is a simulation project, the altitude feedback is obtained by using the simulation environment and the sensor noise is simulated as a Gaussian noise $N(0, \sigma^2)$ with $\sigma = 1$.

As the system is underactuated, all the 6 degrees of freedom can not be controlled seperately. To be able to control the position of the quadcopter the altitude control system is cascaded with the attitude control system. The altitude control system is the outer loop and the attitude control system is the inner loop.

The desired attitude controller inputs are calculated as:

$$\begin{bmatrix} \phi_{des} \\ \theta_{des} \end{bmatrix} = mT_{\Sigma} \begin{bmatrix} \sin(\psi) & -\cos(\psi) \\ \cos(\psi) & \sin(\psi) \end{bmatrix} \begin{bmatrix} U_{ctrl}(x) \\ U_{ctrl}(y) \end{bmatrix}$$

1.4.2 Continuous PID Controller Theory

A Proportional-Integral-Derivative (PID) controller is a widely used control strategy in engineering. It works by continuously measuring the error between a desired setpoint and the actual process variable, and then adjusting the control output to minimize the tracking error.

The PID controller consists of three terms: the proportional term, the integral term, and the derivative term. The control output is the sum of these three terms:

$$u(t) = K_p e(t) + K_i \int_0^t e\,(\tau)d\tau + K_d \frac{de(t)}{dt}$$

where:

- $u(t)$ is the control output at time t
- $e(t)$ is the error at time t, which is the difference between the setpoint and the process variable
- K_p, K_i, and K_d are the proportional, integral, and derivative gains, respectively

1.4.3 Discrete PID Controller

The PID controller can be discretized using the zero-order hold method. The control output is calculated at a fixed sampling time T_s. The integral term is approximated using the trapezoidal rule, and the derivative term is approximated using the backward difference formula. The discrete PID controller can be expressed as:

$$u[k] = K_p e[k] + K_i T_s \frac{e[k] + e[k-1]}{2} + K_d \frac{e[k] - e[k-1]}{T_s}$$

where:

- $u[k]$ is the control output at time k
- $e[k]$ is the error at time k
- T_s is the sampling time
- K_p, K_i, and K_d are the proportional, integral, and derivative gains, respectively

PID Control from an Engineering Prespective

There are multiple approaches to design a PID controller for the system stability and control. When analysing the mathematical model of the systems, the most common way is using analytical methods like root-locus, bode-plot, etc, specially when the system is linear. However, when the system is non-linear, the analytical methods are not applicable. In this case, the most common way is to use the trial-and-error method. In this method, the PID gains are tuned manually until the desired response is achieved. This method is not efficient and can be time consuming. To overcome this problem, there are multiple methods to tune the PID gains automatically. The most common methods are:

1. **Proptional Term:** The proportional term is proportional to the current error. It provides an immediate response to any changes in the error and can help to reduce overshoot and settling time. However, it can also cause steady-state error if the gain is too low.
2. **Integral Term:** The integral term is proportional to the accumulated error over time. It helps to eliminate steady-state error by continuously adjusting the control output. However, it can also cause overshoot and instability if the gain is too high.
3. **Derivative Term:** The derivative term is proportional to the rate of change of the error. It helps to reduce overshoot and settling time by anticipating changes in the

error. However, it can also amplify noise and cause instability if the gain is too high. Inorder to overcome this problem, a low-pass filter is often used to smooth out the derivative term.

Notes to consider when tuning the PID gains

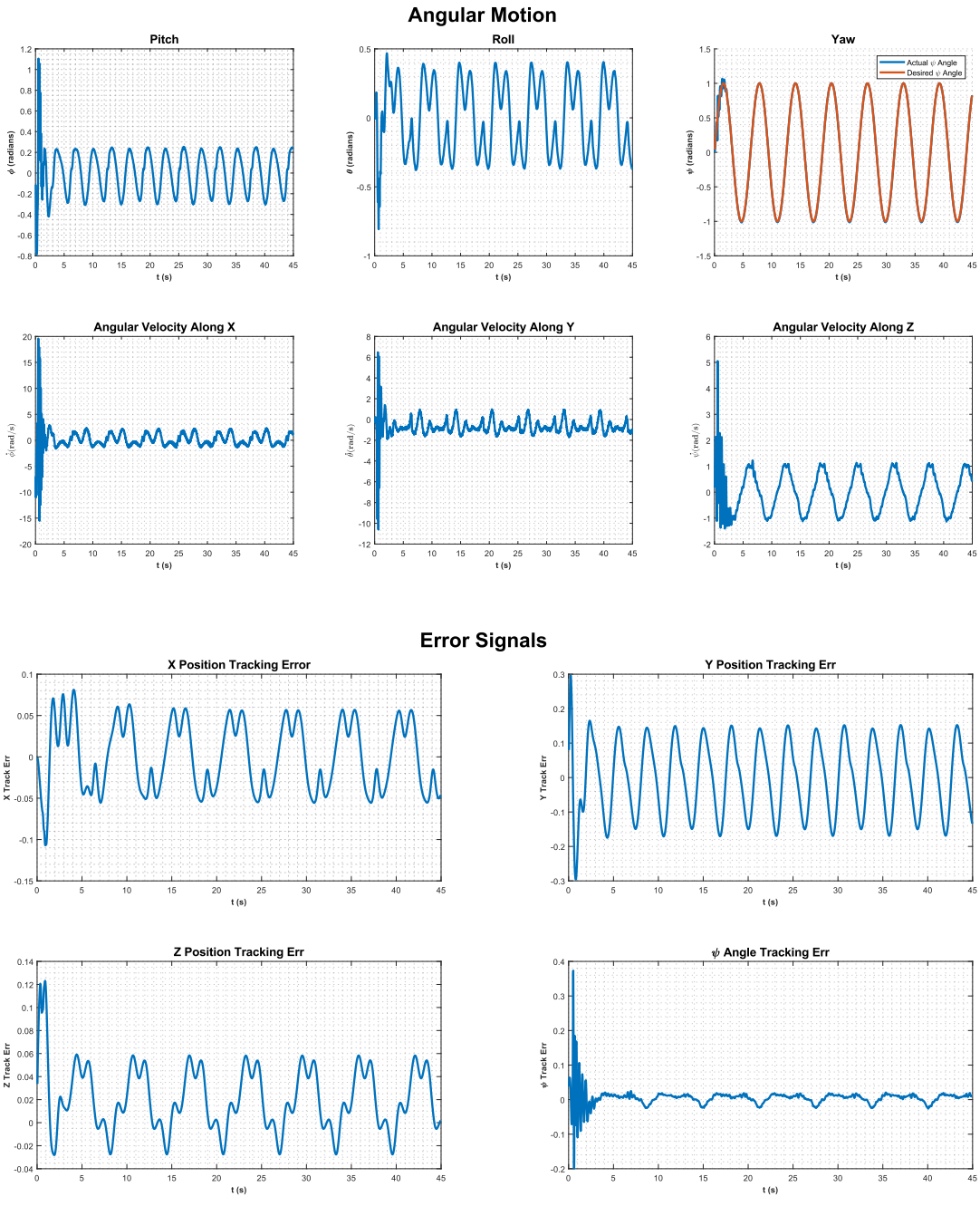
Note 1: The proportional gain affects the response speed and stability of the system. A higher gain results in a faster response but can also cause instability.
Note 2: The integral gain affects the steady-state error of the system. A higher gain results in a smaller steady-state error but can also cause instability.

Note 3: The derivative gain affects the overshoot and settling time of the system. A higher gain results in a smaller overshoot and settling time but can also cause instability.

1.4.2 AutoTune PID Controller

The quadcopter is a complex non-linear coupled system that cannot be analyzed using traditional methods like linearization over equilibrium point and linear controller design methods. As a result, the trial-and-error method is commonly employed to tune the initial PID gains manually until the desired response is achieved. In order to minimize control effort and tracking error, a Bio-Geography based optimization algorithm is then used to fine-tune the gains with the given initial guesses.

When tuning the PID gains for a quadcopter, there are several important factors to consider. Firstly, the dynamics of the quadcopter consist of two control loops: the attitude control loop, which controls the roll, pitch, and yaw angles, and the altitude control loop, which controls the altitude of the quadcopter. Secondly, there are six states that need to be controlled, including the roll, pitch, and yaw angles, as well as the roll, pitch, and yaw angular velocities. This means that there are six PID controllers that need to be tuned. Thirdly, the attitude dynamics are not independent of position dynamics, meaning that the attitude controllers can be affected by the position controllers and vice versa. Finally, it is important to note that the position controller calculates the desired roll and pitch setpoint angles.


Angular Motion
Pitch
Roll
Yaw
Actual ψ Angle
Desired ψ Angle
Angular Velocity Along X
Angular Velocity Along Y
Angular Velocity Along Z
t (s)
Error Signals
X Position Tracking Error
Y Position Tracking Err
Z Position Tracking Err
ψ Angle Tracking Err
X Track Err
Y Track Err
Z Track Err
ψ Track Err

1.5 Results

2. Phase 2 - A Simple Learning Strategy for QuadCopter MultiFlips

In this phase, a simple learning strategy is implemented to make the quadcopter perform a flip maneuver. First, the process of $ N $ flips is formulated on a low-order 2D model as a 5-step sequence with 5 adjustable parameters. An optimization algorithm is used to find the optimal parameters based on a cost function that minimizes the difference between the starting point and the end point of the flip maneuver.

2.1 Low-Order 2D Model

For the purpose of this phase, a low-order 2D model is used to formulate the propblem and out of the plane dynamics are ignored and assumed to be stable. The 2D model is:

$$\begin{cases} m\ddot{z} = T_\Sigma \cos\theta - mg \\ m\ddot{x} = T_\Sigma \sin\theta \\ I_{yy}\ddot{\theta} = l\tau_y \end{cases} \Rightarrow \begin{cases} \ddot{z} = \dfrac{T_\Sigma}{m}\cos\theta - g \\ \ddot{x} = \dfrac{T_\Sigma}{m}\sin\theta \\ \ddot{\theta} = \dfrac{l}{I_{yy}}\tau_y \end{cases}$$

where:

- m is the mass of the quadcopter
- I_{yy} is the moment of inertia of the quadcopter about the y axis
- l is the arm length of the quadcopter

2.2 Parameterization of Flip Maneuver

The QuadCopter needs to execute a flip that results in a rotation offset of a multiple of 2π while keeping all other states unaltered. All the out of plane dynamics are ignored and assumed to be stable.

The initial and final state conditions of the multi-flip maneuver are:

$$\begin{cases} x_0 = x_f = 0 \\ z_0 = z_f = 0 \\ \dot{x}_0 = \dot{x}_f = 0 \\ \dot{z}_0 = \dot{z}_f = 0 \\ \theta_f = \theta_0 + 2\pi N \end{cases}$$

where N is the number of flips.

To seek an almost time optimal control, a control envelope is defined. So, the control inputs can be seeked on the odge of the control envelope. A reduced order control envelope is defined as $\left[\underline{\beta}, \overline{\beta}\right]$ to account for modeling uncertainties and to reserve some control authority for the on-board feedback controllers. The desired T_Σ must be consistent with a slightly reduced range of accelerations. A convenient way to parameterize it is:

$$U_{min} \leq m\underline{\beta} \leq U \leq m\overline{\beta} \leq U_{max}$$

where U_{min} and U_{max} correspond to the minimum and maximum thrusts that the quadcopter can produce, respectively. If no control margin is reserved ($U_{min} = 0$ and $U_{max} = \infty$), $\overline{\beta}$ corresponds to the vehicle's maximum acceleration at full thrust.

The maximum and minumum collective thrust can be formulated as:

$$\begin{cases} T_{\Sigma,upper} = m\overline{\beta} - \dfrac{2|\ddot{\theta}_{max}|I * yy}{l} \\ T_{\Sigma,lower} = m\underline{\beta} + \dfrac{2|\ddot{\theta}_{max}|I * yy}{l} \\ |\ddot{\theta}_{max}| \leq lm\left(\overline{\beta} - \underline{\beta}\right)/4I * yy \end{cases}$$

Consider that there is a $\dot{\theta}_{max}$ which is the maximum angular velocity that the quadcopter can achieve. This is a design parameter.

The model inputs are the collective thrust U_{des} and the desired motion rates. Having the control envelope defined, the flip maneuver can be parameterized as a 5-step sequence where each step excepts a control action in the form of $\{U_{des}, \ddot{\theta}_{max}, t\}$ where t is the duration of the step.

The 5-step sequence is:

1. **Acceleration**: Accelerate up at near-maximum collective acceleration while rotating slightly away.
2. **Start Rotate**: Use maximum differential thrust to achieve $\dot{\theta}_{max}$.
3. **Coast**: Hold $\dot{\theta}_{max}$ (use a low collective thrust command to prevent accelerating into ground).
4. **Stop Rotate**: Maximum differential thrust to reach $\dot{\theta}$ slightly less than 0.
5. **Recovery**: Accelerate up with near-maximum collective thrust with a slight rotational acceleration to stop vertical descent and any remaining horizontal movement.

Of all the steps, there are 5 unknown parameters:

1. U_1 - collective acceleration during step 1.
2. T_1 - duration of step 1.
3. T_3 - duration of step 3 (coasting at $\dot{\theta}_{max}$).
4. U_5 - collective acceleration during step 5.
5. T_5 - duration of step 5.

At each step all the other parameters can be considered as known. For conciseness, a normalized mass distribution is variable is defined as:

$$\alpha = 2I_{yy}/(ml^2)$$

Now, the other parameters can be defined as:

$$\begin{aligned}
\ddot{\theta}_1 &= -(\overline{\beta} - U_1)/\alpha L \\
\ddot{\theta}_2 = -\ddot{\theta}_4 &= \left(\overline{\beta} - \bar{\beta}\right)/2\alpha L \\
U_2 = U_4 &= \left(\overline{\beta} + \bar{\beta}\right)/2 \\
T_2 &= (\dot{\theta}_{max} - \ddot{\theta}_1 T_1)/\ddot{\theta}_4 \\
\ddot{\theta}_3 &= 0 \\
U_3 &= \beta \\
T_4 &= -(\dot{\theta}_{max} - \ddot{\theta}_5 T_5)/\ddot{\theta}_4 \\
\ddot{\theta}_5 &= (\overline{\beta} - U_5)/\alpha L
\end{aligned}$$

Now, the whole problem is formulated as an optimization problem to finding values of the unknown parameters to satisfy the objective function below:

$$J = \sum_{i=1}^{6} T_i$$

$$where,\ T = \begin{bmatrix} x_0 \\ z_0 \\ \dot{x}_0 \\ \dot{z}_0 \\ \theta_0 \\ \dot{\theta}_0 \end{bmatrix} - \begin{bmatrix} x_f \\ z_f \\ \dot{x}_f \\ \dot{z}_f \\ \theta_f \\ \dot{\theta}_f \end{bmatrix}$$

2.3 3D Generalization

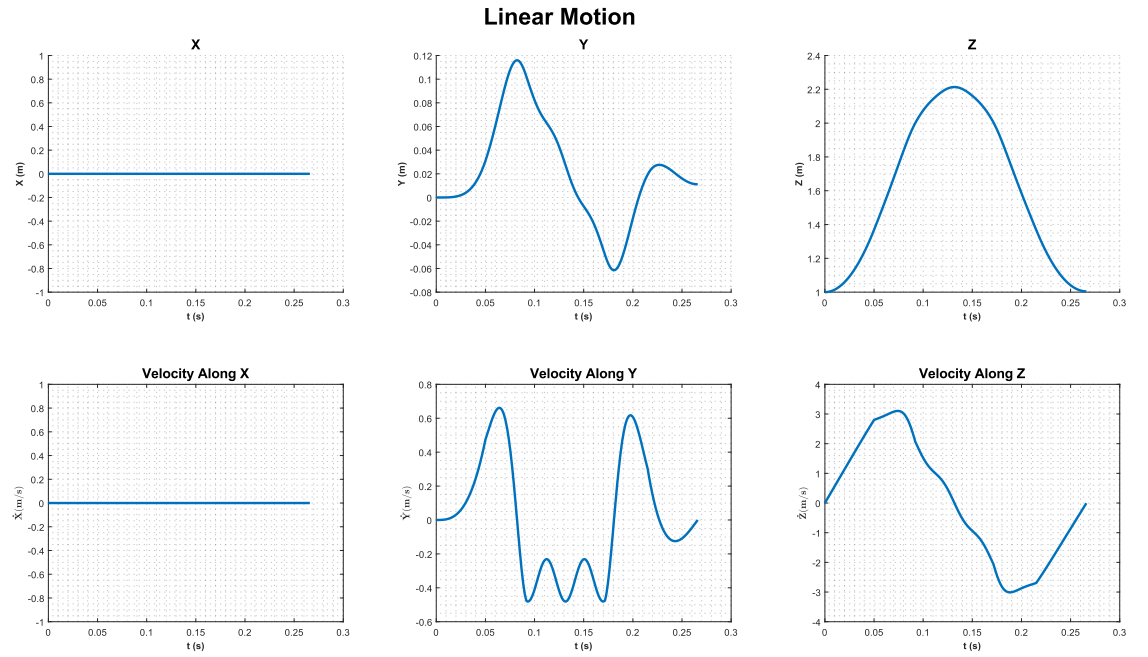
As it is just a simulation, the 3D genralization is just a simple change in the cost function and setting all the outer-plane desired accelerations to zero ($\ddot{\phi}_{des} = 0, \ddot{\psi}_{des} = 0$). However, in a real-world scenario, the 3D generalization would require a more complex control system.

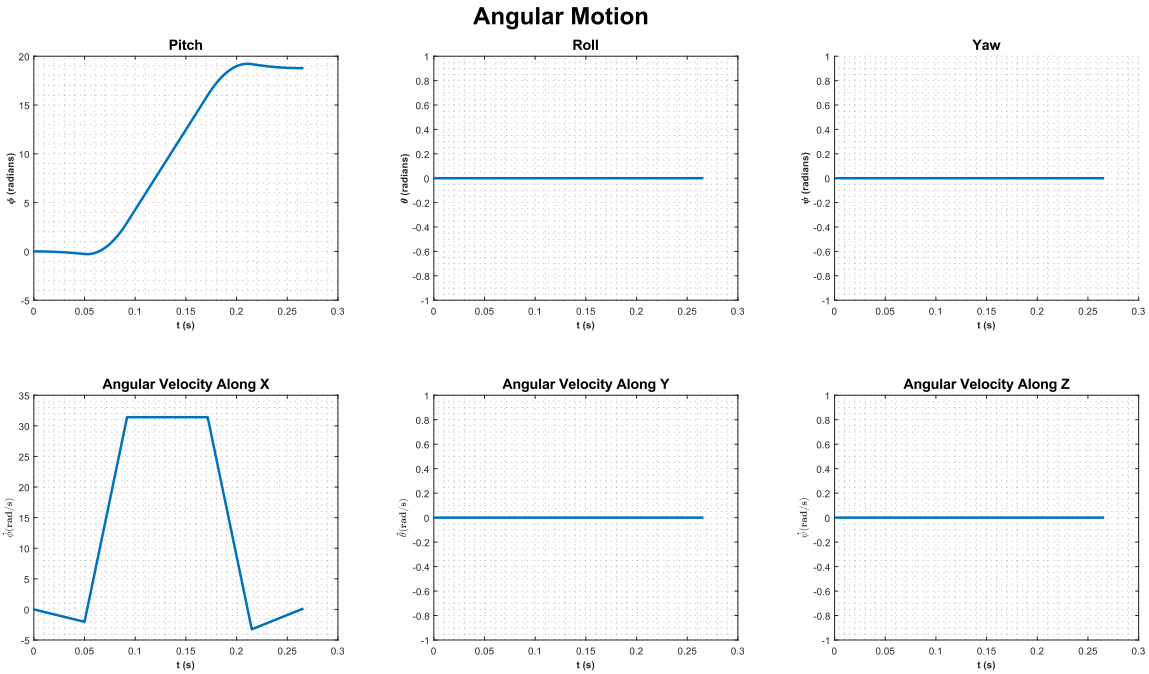
The cost function is changed to:

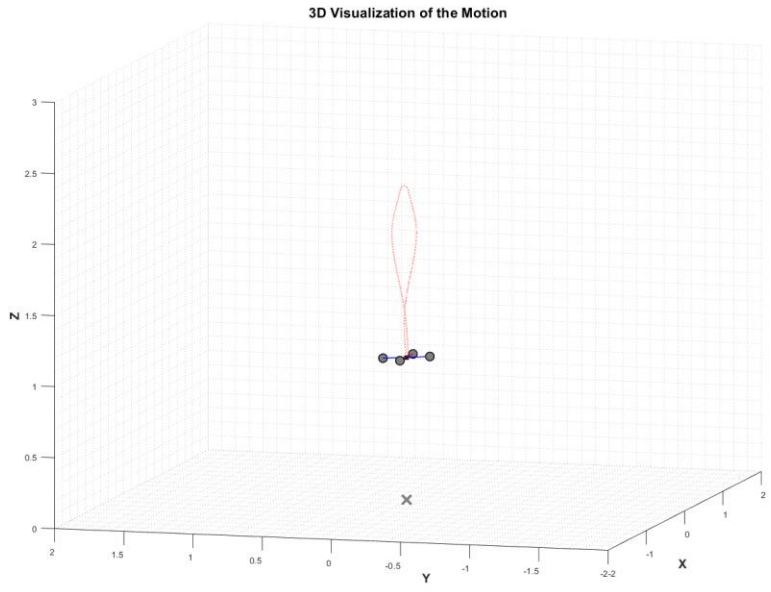
$$\begin{aligned}
X &= (x, y, z, \dot{x}, \dot{y}, \dot{z}, \phi, \theta, \psi, \dot{\phi}, \dot{\theta}, \dot{\psi}) \\
X_0 &= (x_0, y_0, z_0, \dot{x}_0, \dot{y}_0, \dot{z}_0, \phi_0, \theta_0, \psi_0, \dot{\phi}_0, \dot{\theta}_0, \dot{\psi}_0) \\
X_f &= (x_f, y_f, z_f, \dot{x}_f, \dot{y}_f, \dot{z}_f, \phi_f, \theta_f, \psi_f, \dot{\phi}_f, \dot{\theta}_f, \dot{\psi}_f)
\end{aligned}$$

$$T = X_0 - X_f \Rightarrow J = \sum_{i=1}^{12} T_i$$

2.4 Results


Linear Motion
X
Y
Z
X (m)
Y (m)
Z (m)
t (s)
Velocity Along X
Velocity Along Y
Velocity Along Z
$\dot{X}$(m/s)
$\dot{Y}$(m/s)
$\dot{Z}$(m/s)


Angular Motion
Pitch
Roll
Yaw
ϕ (radians)
θ (radians)
ψ (radians)
t (s)
Angular Velocity Along X
Angular Velocity Along Y
Angular Velocity Along Z
$\dot{\phi}$(rad/s)
$\dot{\theta}$(rad/s)
$\dot{\psi}$(rad/s)



3. Phase 3 - Dynamic Modelling and Implementation of a non-Linear Sliding Mode Controller

3.1 Sliding Mode Theory

Sliding mode control is a nonlinear control method that alters the dynamics of a nonlinear system by application of a discontinuous control signal that forces the system to "slide" along a cross-section of the system's normal behavior. This cross-section is called the "switching surface" or "switching hyperplane" or "sliding surface". The switching surface is a function of the system state and is defined such that the system trajectories always converge to it. The control signal is then designed to drive the system trajectories along the switching surface to the origin. The control signal is discontinuous because it is composed of two or more continuous signals, each active over a different region of the state space. The switching surface is designed such that the system trajectories are always driven toward the origin, regardless of the direction of the discontinuity in the control signal. This is the fundamental property of sliding mode control that makes it robust to system uncertainties and disturbances. The discontinuous control signal is also called a "chattering" control signal because it causes the system trajectories to "chatter" as they slide along the switching surface. The chattering is undesirable because it can cause high-frequency oscillations in the system states and control signal. The chattering can be reduced with several methods, including the use of a saturation function in the control signal, the use of a boundary layer around the switching surface, and the use of a high-gain control signal.

3.1.1 Sliding Mode Control Design

For a typical nonlinear system of the form:

$$\ddot{x} = f(x, \dot{x}, t) + g(x, \dot{x}, t)u + \omega_x$$

where:

- x is the state vector
- u is the control input
- $f(x, \dot{x}, t)$ is the nonlinear system dynamics
- $g(x, \dot{x}, t)$ is the control input gain matrix
- ω_x is the system uncertainties and disturbances ($|\omega_x| \leq \kappa$)

The sliding mode control design is as follows:

1. Define the system error as:

$$e = x_d - x$$

2. Define the sliding surface as:

$$\begin{aligned} s &= \dot{e} + \lambda e \\ \Rightarrow \dot{s} &= \ddot{e} + \lambda \dot{e} \\ \Rightarrow \dot{s} &= \ddot{x}_d - \ddot{x} + \lambda \dot{e} \\ \Rightarrow \dot{s} &= \ddot{x}_d - f(x, \dot{x}, t) - g(x, \dot{x}, t)u - \omega_x + \lambda \dot{e} \end{aligned}$$

   where λ is a positive constant.

3. Define a Lyapunov function as:

$$V = \frac{1}{2}s^2$$

   Taking the time derivative of V and extracting the conditions to make it negative definite:

$$\begin{aligned} \dot{V} &= s\dot{s} \leq -\eta|s| \\ \Rightarrow \dot{V} &= \dot{s} \leq -\eta \, sign(s) \end{aligned}$$

   where η is a positive constant.

4. Define the control input as:

$$u = \frac{1}{g(x, \dot{x}, t)}\left(-f(x, \dot{x}, t) - \omega_x + \lambda \dot{e} - \eta \, sign(s)\right)$$

5. Assuming that the system disturbances and uncertainties are bounded by κ, the control input can be written as:

$$d_L \leq |\omega_x| \leq d_U \Rightarrow \begin{cases} d_1 = \dfrac{d_U - d_L}{2} \\ d_2 = \dfrac{d_U + d_L}{2} \end{cases} \Rightarrow \omega_c = d_2 - d_1 sign(s)$$

$$u = \frac{1}{g(x, \dot{x}, t)}(-f(x, \dot{x}, t) - \omega_x + \lambda \dot{e} - \eta\, sign(s) - \omega_c)$$

3.2 Quadcopter Dynamic Model

The quadcopter dynamical model is discussed in the previous phases. But it is simpler to use the equations as second order differential equations and not in the state space representation. The equations are as follows:

$$\begin{cases} \ddot{x} &= \big(\cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi)\big)\dfrac{l}{m}U_1 + \omega_x \\ \ddot{y} &= \big(\cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi)\big)\dfrac{l}{m}U_1 + \omega_y \\ \ddot{z} &= -g + \big(\cos(\phi)\cos(\theta)\big)\dfrac{l}{m}U_1 + \omega_z \\ \ddot{\phi} &= \dot{\theta}\dot{\psi}\left(\dfrac{I_{yy} - I_{zz}}{I_{xx}}\right) - \dfrac{I_r}{I_{xx}}\dot{\theta}\Omega + \dfrac{l}{I_{xx}}U_2 + \omega_\phi \\ \ddot{\theta} &= \dot{\phi}\dot{\psi}\left(\dfrac{I_{zz} - I_{xx}}{I_{yy}}\right) - \dfrac{I_r}{I_{xx}}\dot{\phi}\Omega + \dfrac{l}{I_y}U_3 + \omega_\theta \\ \ddot{\psi} &= \dot{\theta}\dot{\phi}\left(\dfrac{I_{xx} - I_{yy}}{I_{zz}}\right) + \dfrac{l}{I_{zz}}U_4 + \omega_\psi \end{cases}$$

Note: Torques and forces are considered the same as discussed in the first phase.

For the sake of simplicity, the following constants are declared:

$$
\begin{aligned}
a_1 &= \frac{I_{yy} - I_{zz}}{I_{xx}} \\
a_2 &= -\frac{I_r}{I_{xx}} \\
a_3 &= \frac{I_{zz} - I_{xx}}{I_{yy}} \\
a_4 &= -\frac{I_r}{I_{yy}} \\
a_5 &= \frac{I_{xx} - I_{yy}}{I_{zz}} \\
b_1 &= \frac{l}{I_{xx}} \\
b_2 &= \frac{l}{I_{yy}} \\
b_3 &= \frac{l}{I_{zz}} \\
u_x &= \big(\cos(\phi)\sin(\theta)\cos(\psi) + \sin(\phi)\sin(\psi)\big) \\
u_y &= \big(\cos(\phi)\sin(\theta)\sin(\psi) - \sin(\phi)\cos(\psi)\big)
\end{aligned}
$$

Then the equations can be written as:

$$
\begin{cases}
\ddot{x} &= u_x \dfrac{l}{m} U_1 + \omega_x \\
\ddot{y} &= u_y \dfrac{l}{m} U_1 + \omega_y \\
\ddot{z} &= -g + \big(\cos(\phi)\cos(\theta)\big) \dfrac{l}{m} U_1 + \omega_z \\
\ddot{\phi} &= \dot{\theta}\dot{\psi} a_1 + \dot{\theta}\Omega a_2 + b_1 U_2 + \omega_\phi \\
\ddot{\theta} &= \dot{\phi}\dot{\psi} a_3 + \dot{\phi}\Omega a_4 + b_2 U_3 + \omega_\theta \\
\ddot{\psi} &= \dot{\theta}\dot{\phi} a_5 + b_3 U_4 + \omega_\psi
\end{cases}
$$

3.3 Sliding Mode Control Design

As said before, in quadcopter control system, trajectories are expressed for the motion of $[X, Y, Z, \psi]$. So, the control system is designed for these variables.

3.3.1 Translational Control

1. Z Control System:

$$\begin{aligned} e_z &= z_d - z \\ s_z &= \dot{e}_z + \lambda_z e_z \\ \rightarrow \dot{s}_z &= \ddot{e}_z + \lambda_z \dot{e}_z \\ &= \ddot{z}_d - \ddot{z} + \lambda_z \dot{e}_z \\ \rightarrow \dot{s}_z &= \ddot{z}_d + g - \big(\cos(\phi)\cos(\theta)\big)\frac{l}{m}U_1 - \omega_z + \lambda_z \dot{e}_z \\ &= -\eta_z \, sign(s_z) \\ \Rightarrow U_z &= \frac{m}{\cos(\phi)\cos(\theta)}\big(\ddot{z}_d + g - \omega_z + \lambda_z \dot{e}_z + \eta_z \, sign(s_z)\big) \end{aligned}$$

   U_z can be singular when $\cos(\phi)\cos(\theta) = 0$. So, we assume: $|\phi|\ ,\ |\theta| \leq \frac{\pi}{2}$.

2. As we have 2 control loops the U_x and U_y are controlled through virtual inputs u_x, u_y. Following the same principles:

$$\begin{aligned} e_x &= x_d - x \\ s_x &= \dot{e}_x + \lambda_x e_x \\ \rightarrow \dot{s}_x &= \ddot{e}_x + \lambda_x \dot{e}_x \\ &= \ddot{x}_d - \ddot{x} + \lambda_x \dot{e}_x \\ \rightarrow \dot{s}_x &= \ddot{x}_d - u_x \frac{l}{m}U_1 - \omega_x + \lambda_x \dot{e}_x \\ &= -\eta_x \, sign(s_x) \\ \Rightarrow U_x &= \frac{m}{U_1}\big(\ddot{x}_d - \omega_x + \lambda_x \dot{e}_x + \eta_x \, sign(s_x)\big) \end{aligned}$$

   In the same manner U_y can be extracted as:

$$U_y = \frac{m}{U_1}\Big(\ddot{y}_d - \omega_y + \lambda_y \dot{e}_y + \eta_y \, sign\big(s_y\big)\Big)$$

3. Stability check:

$$V\big(s_x, s_y, s_z\big) = \frac{1}{2}\sum_{i=x,y,z} s_i^2 \rightarrow \dot{V} = \sum_{i=x,y,z} s_i \, \dot{s} * i$$
$$= -\sum * i = x, y, z - \eta_i \, |s_i| < 0$$

This shows that the system approaches the sliding surface in finite time.

3.3.2 Rotational Control

To extract the desired && and θ from the desired x_d and y_d, u_x and u_y can be used by substituting the $\psi_d = 0$. So:

$$\begin{cases} u_x &= \big(\cos(\phi_d)\sin(\theta_d)\cos(0) + \sin(\phi_d)\sin(0)\big) \\ u_y &= \big(\cos(\phi_d)\sin(\theta_d)\sin(0) - \sin(\phi_d)\cos(0)\big) \end{cases} \Rightarrow \begin{cases} \phi_d &= -\arcsin(u_y) \\ \theta_d &= \arcsin\left(\dfrac{u_x}{\cos(\phi_d)}\right) \end{cases}$$

Also, for the simplification, $\dot{\phi}_d$ and $\dot{\phi}_d$ are also assumed to be 0.

Controller design:

1. design for ϕ:

$$\begin{aligned} e_\phi &= \phi_d - \phi \\ s_\phi &= \dot{e} * \phi + \lambda * \phi e_\phi \\ \rightarrow \dot{s} * \phi &= \ddot{e} * \phi + \lambda_\phi \dot{e} * \phi \\ &= \ddot{\phi} * d - \ddot{\phi} + \lambda * \phi \dot{e} * \phi \\ \rightarrow \dot{s} * \phi &= \ddot{\phi} * d - \dot{\theta}\dot{\psi}a_1 - \dot{\theta}\Omega a_2 - b_1 U_2 - \omega\phi + \lambda * \phi \dot{e} * \phi \\ &= -\eta_\phi \; sign(s_\phi) \\ \Rightarrow U_\phi &= \frac{1}{b_1}\Big(\ddot{\phi} * d - \dot{\theta}\dot{\psi}a_1 - \dot{\theta}\Omega a_2 - \omega * \phi + \lambda_\phi \dot{e} * \phi + \eta * \phi \; sign(s_\phi)\Big) \end{aligned}$$

2. design for θ. In the same manner:

$$U_\theta = \frac{1}{b_2}\Big(\ddot{\theta} * d + \dot{\phi}\dot{\psi}a_3 - \dot{\phi}\Omega a_4 - \omega * \theta + \lambda_\theta \dot{e} * \theta + \eta * \theta \; sign(s_\theta)\Big)$$

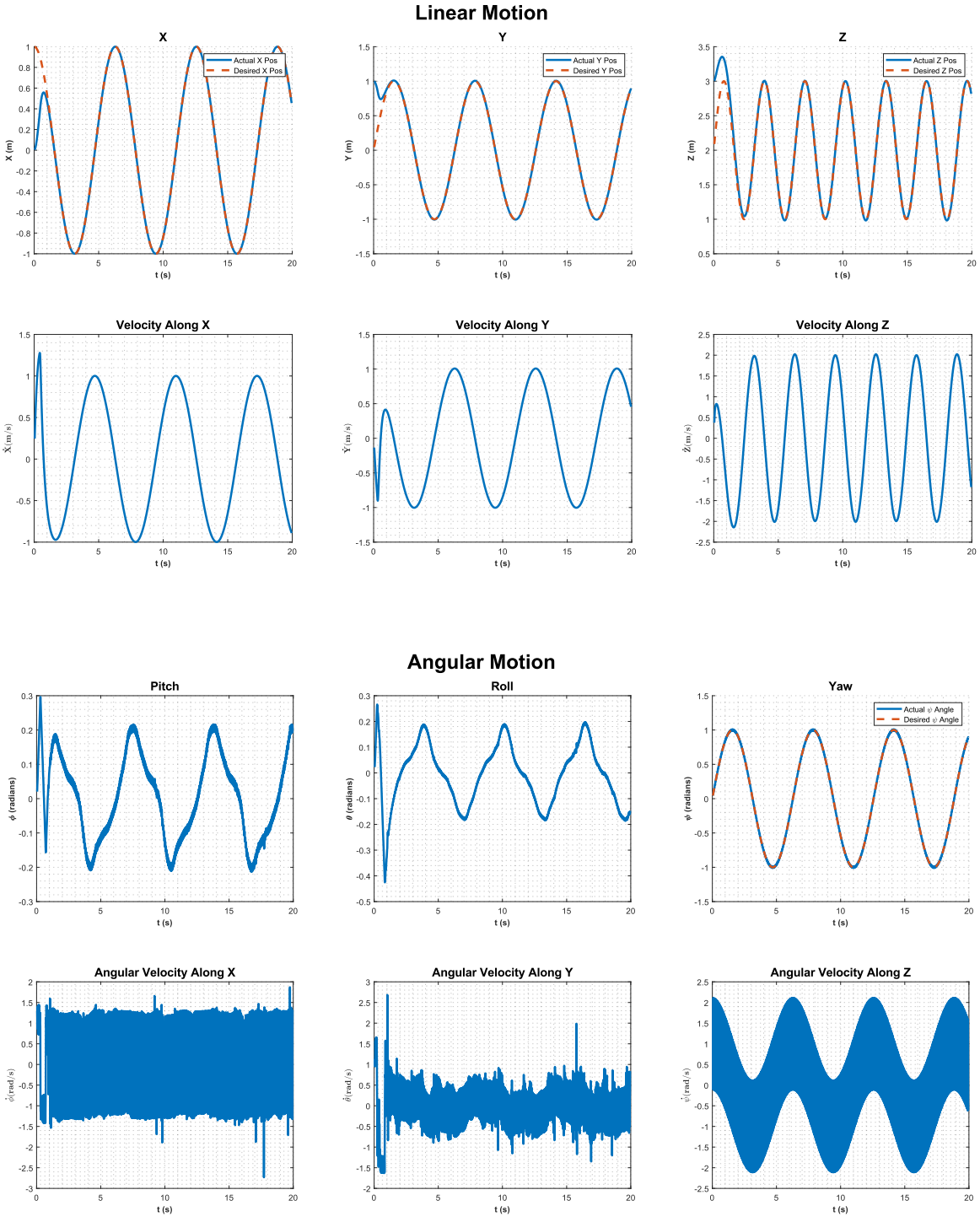
3. design for ψ. In the same manner:

$$U_\psi = \frac{1}{b_3}\Big(\ddot{\psi} * d + \dot{\phi}\dot{\theta}a_5 - \omega * \psi + \lambda_\psi \dot{e} * \psi + \eta * \psi \; sign(s_\psi)\Big)$$

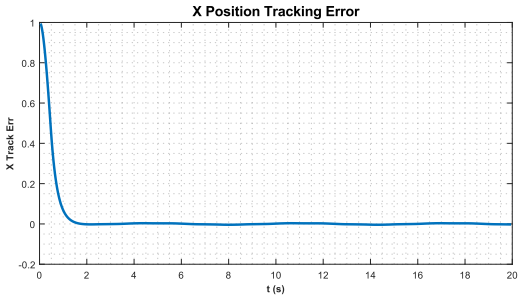
4. Stability is guaranteed in the exact same manner as the translational control.

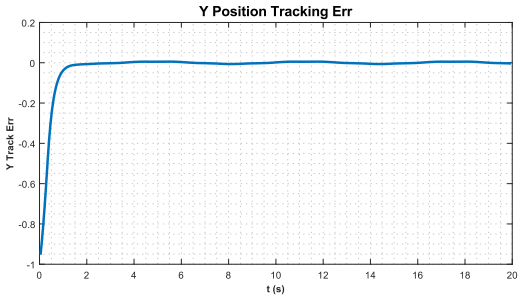
Note: The disturbance effect is modelled as $\omega = 0.5\sin(t/10) + 0.1\sin(t/5) + 0.02\sin(t)$ for all the inputs.

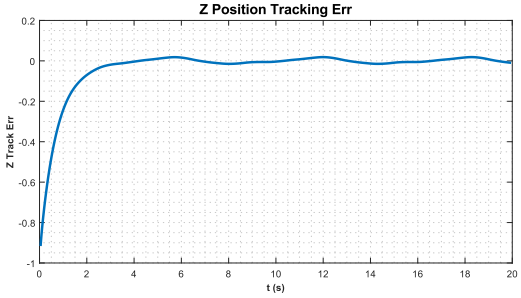
3.4 Results

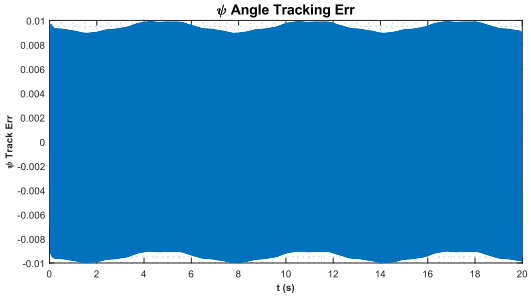

Linear Motion
X
Actual X Pos
Desired X Pos
X (m)
t (s)
Y
Actual Y Pos
Desired Y Pos
Y (m)
t (s)
Z
Actual Z Pos
Desired Z Pos
Z (m)
t (s)
Velocity Along X
t (s)
Velocity Along Y
t (s)
Velocity Along Z
t (s)
Angular Motion
Pitch
t (s)
Roll
t (s)
Yaw
Actual ψ Angle
Desired ψ Angle
t (s)
Angular Velocity Along X
t (s)
Angular Velocity Along Y
t (s)
Angular Velocity Along Z
t (s)

Error Signals

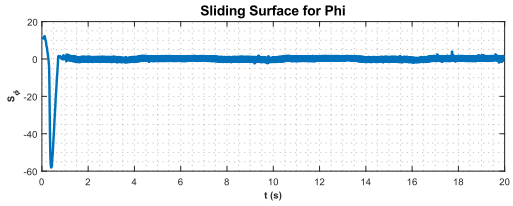


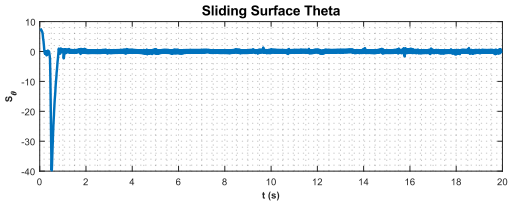


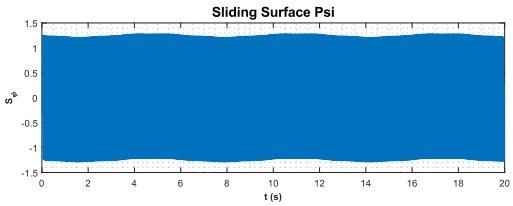


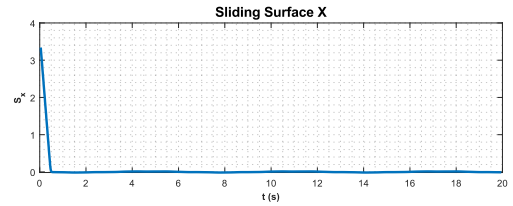


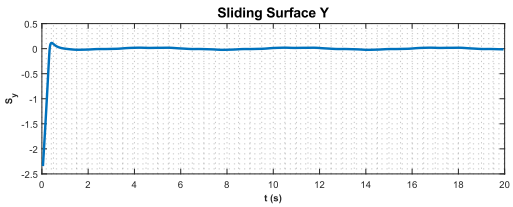
Sliding Surfaces

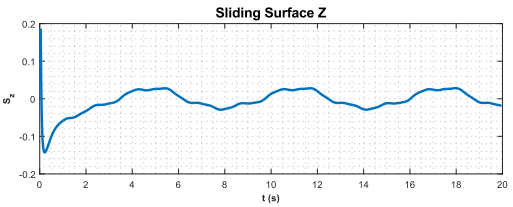












4. Simulation Environment

For all the phases of the project, the simulation environment is the same. All the simulation is done in MATLAB. For all the phases the ODE function of the system is created and solved using the 4th order Runge-Kutta method or Euler's method of ODE solving which will be discussed in the following sections.

4.1 4th Order Runge-Kutta Method

The 4th order Runge-Kutta algorithm, also known as RK4, is a numerical method used to solve ordinary differential equations (ODEs) of the form:

$$\frac{dy}{dt} = f(t, y)$$

where y is the dependent variable and t is the independent variable. The algorithm is based on the idea of approximating the solution at each time step by using a weighted average of four estimates.

The general form of the RK4 algorithm can be written as:

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

where y_n is the value of y at time t_n, and y_{n+1} is the value at time $t_{n+1} = t_n + h$, where h is the time step. The four estimates k_1, k_2, k_3, and k_4 are given by:

$$k_1 = hf(t_n, y_n)$$

$$k_2 = hf\left(t_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right)$$

$$k_3 = hf\left(t_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right)$$

$$k_4 = hf(t_n + h, y_n + k_3)$$

In words, the algorithm works as follows:

1. Evaluate the function $f(t, y)$ at the current time and state (t_n, y_n) to obtain k_1.
2. Use k_1 to estimate the value of y at the midpoint of the time step, $\left(t_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right)$, and evaluate $f(t, y)$ at this point to obtain k_2.
3. Use k_2 to estimate the value of y at the midpoint of the time step, $\left(t_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right)$, and evaluate $f(t, y)$ at this point to obtain k_3.
4. Use k_3 to estimate the value of y at the end of the time step, $(t_n + h, y_n + k_3)$, and evaluate $f(t, y)$ at this point to obtain k_4.

5. Combine the four estimates using a weighted average to obtain the final estimate for y_{n+1}.

The RK4 algorithm is widely used because it is relatively simple to implement and provides accurate results for a wide range of ODEs. However, it can be computationally expensive for very large systems of equations or for highly stiff ODEs.

4.2 Euler's Method

Euler's method is a simple numerical method used to solve ordinary differential equations (ODEs) of the form:

$$\frac{dy}{dt} = f(t, y)$$

where y is the dependent variable and t is the independent variable. The method involves approximating the solution at each time step by using the derivative of the solution at that point.

The general form of the Euler's method can be written as:

$$y_{n+1} = y_n + hf(t_n, y_n)$$

where y_n is the value of y at time t_n, and y_{n+1} is the value at time $t_{n+1} = t_n + h$, where h is the time step. The derivative $f(t_n, y_n)$ is evaluated at the current time and state (t_n, y_n).

In words, the algorithm works as follows:

1. Start with an initial value of y at time t_0.
2. Evaluate the derivative $f(t_0, y_0)$ at the initial time and state.
3. Use this derivative to estimate the value of y at the next time step, $t_1 = t_0 + h$, using the formula above.
4. Repeat steps 2-3 for each subsequent time step until the desired end time is reached.

Euler's method is easy to implement and computationally efficient, but it may not always provide accurate results for complex ODEs or for large time steps. It is often used as a starting point for more advanced numerical methods like the 4th order Runge-Kutta algorithm.

4.3 Comparison of the Two Methods

The 4th order Runge-Kutta algorithm is more accurate than Euler's method, especially for complex ODEs or for large time steps. However, it is also more computationally expensive and may not be necessary for simple ODEs or small-time steps. The choice between these methods depends on the complexity of the ODE and the desired level of accuracy.

---

5. References

[1] Robotics, Vision and Control: Fundamental Algorithms in MATLAB

[2] Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor

[3] A platform for aerial robotics research and demonstration: The Flying Machine Arena

[4] Backflipping with Miniature Quadcopters by Gaussian Process Based Control and Planning

[5] A Simple Learning Strategy for High-Speed Quadrocopter Multi-Flips

[6] Adaptive Open-Loop Aerobatic Maneuvers for Quadrocopters

[7] Adaptive fast open-loop maneuvers for quadrocopters

[8] MODELING, SIMULATION AND COMPLETE CONTROL OF A QUADCOPTER

[9] Modeling and Nonlinear Control of a Quadcopter for Stabilization and Trajectory Tracking

[10] Trajectory planning of quadrotor using sliding mode control with extended state observer

[11] Sliding Mode Control: An Approach to Control a Quadrotor

[12] Observer-Based Sliding Mode Control of a 6-DOF Quadrotor UAV

[13] Runge–Kutta methods

[14] Euler method

[15] Bio-Geography Based Optimization