

Chapter 1

Introduction

An Unmanned Aerial Vehicle (UVA), also known as a drone, is an aircraft that can fly remotely without any human pilot on board or can fly autonomously with an onboard flight controller that works in conjunction with sensors and a global positioning system (GPS). The last decades have seen the great technological advances in sensors, embedded control systems, aerodynamics, and control theories, which make unmanned aerial vehicles (UVAs) more widespread and gaining more and more attention in the field of science and technology.

A variety of UAVs have been developed and according to their configurations, they are grouped into three categories: fixed-wing UAVs, multi-rotor UAVs, and fixed-wing hybrid VTOL UAVs. Fixed-wing UAVs generate the lift through the wing itself, thanks to the forward airspeed provided by an internal engine or an electric motor propeller []. They can cover very long distances, map very large areas, and loiter for long-time monitoring because they are energy efficient. But the fixed-wing UAVs can't hover in the air. According to the shape of the wing, the fixed-wing UAVs can be categorized into the straight wing, swept wing, and delta wing UAVs. Multi-rotor UAVs generate lift by multiple electronic motor propellers, they are very maneuverable, can hover in the air, and achieve great control performance during flight. Multi-rotor UAVs can be categorized according to the number of motors, the most common configurations are Tricopters (3 motors), Quadcopters (4 motors), Hexacopters (6 motors), and Octocopters (8 rotors). By far, quadcopters are the most popular multi-rotor UAVs all over the world. Fixed-wing hybrid VTOL UAVs merge the benefits of both fixed-wing and rotor-based designs. UAVs of this type have rotors attached to the fixed wings, allowing them to hover and take off and land vertically, while at the same time retaining all the advantages of the fixed-wing type. Only a handful of fixed-wing hybrid VTOLs are currently on the market, and the technology used in these UAV types is still in the nascent stage.

In this work, attention is focused on the quadcopter, which is one type of helicopter lifted and propelled by four motors placed in a square format with equal distance from the center of mass. The motors are usually placed at the end points of the arms, which are in cross configuration and linked to the quadcopter's central body where the sensors, microcontrollers, and other electronic circuits are located. The quadcopter is an under-actuated system since only four motors are utilized to control its 6 degrees of freedom (6DoF) motion in the space (four control inputs, six variables to control). This means that the quadcopter's translational and rotational motions are coupled, leading to highly nonlinear dynamics in the system. Furthermore, quadcopters are inherently unstable and require constant corrections to be made by the onboard microcontroller hundreds or even thousands of times a second to maintain stability. Despite these disadvantages, quadcopters feature various advantages such as cheap, lightweight, strong maneuverability, vertical taking off and landing, and the capability of hovering.

This work is focused on the design of an autonomous flight controller for the quadcopter based on the GNSS positioning.

1.1 A Brief History of Quadcopter

The history of quadcopters dates back to the very beginning of rotary wing aviation in the early 20th century. The story starts from both the military development as well as the radio control technology. In 1898, Nikola Tesla tested his radio-controlled boat for the first time in a New York Pond in Madison Square Garden, which is the beginning of every radio-controlled aircraft as we know it today. Around 1907, the world's first quadcopter was created by inventor brothers Jacques and Louis Bréguet, working with controversial Nobel Prize winner Professor Charles Richet [1]. But it had big limitations and the design proved to be very unstable and hence impractical.

It was until 1924 that the world's first working quadcopter was invented by French engineer Etienne Ominichen, called the Ominichen 2 (Figure 1.1). The Ominichen 2 was a cross-shaped structure built of metal tubing and lifted by four two-bladed, counter-rotating main rotors. The pitch angle of these blades was controlled by warping. The quadcopter also had another five rotors positioned in the horizontal plane to provide lateral control, one of which at the front was used to steer the quadcopter [2]. In 1924, Ominichen flew this quadcopter a distance of 360 meters and in the same year, he flew a 1-kilometer closed circle in 7m and 40s, which established a record for helicopters. Around the same time, George de Bothezat built and tested his quadcopter for the US army, completing a number of test flights before the program was scrapped [3]. The de Bothezat quadcopter (Figure 1.2) was an X-shaped structure supported by four six-blade rotors at each end of the arms. At the ends of the lateral arms, two small propellers with variable pitch were used for thrusting and yaw control. The quadcopter made its first flight in October 1922, and about 100 flights were made by the end of 1923 [4].

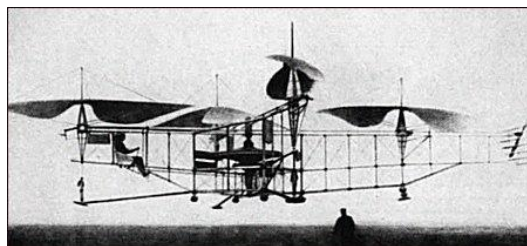


Figure 1.1 Ominichen 2 [2]

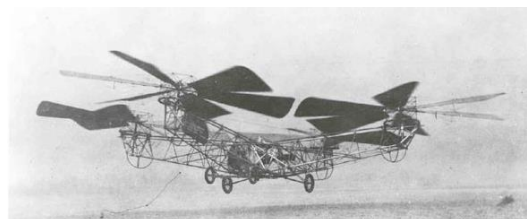


Figure 1.2 de Bothezat quadcopter [3]

In the middle of the 1950s, Convertawings Model A Quadcopter was intended to be the prototype for a line of much larger civil and military quadrotor helicopters []. The four rotors of this quadcopter are mounted laterally on outriggers in two tandem pairs. The control mechanism is extremely simple and obtained by differential change of thrust between the rotors []. It was one of the first quadcopters to use the varying thrusts of the four rotors to achieve control in flight. But it was very hard for the pilot to maneuver during flight because of the workload of trying to control the thrusts of all four rotors at once. Despite successful testing and development, military support for this quadcopter ceased after cutbacks in defense spending. However, the design, particularly its control system, was a precursor of the current experimental vertical-rising aircraft designs that incorporate tandem wings or a square configuration of four fans, ducts, or jets [].

In response to an Army Transportation Crops requirement, the Curtiss-Wright VZ-7 was developed in 1958. The VZ-7 essentially consists of a rectangular central airframe to which four vertically mounted propellers were attached in a square pattern. The VZ-7's control system was also very simple, the directional movement was controlled by varying the thrust of each propeller, with additional yaw control provided by moveable vanes fixed over the engine exhaust []. The VZ-7 performed very well when evaluated by the Army. The craft was capable of hovering and forward flight and proved relatively stable and easy to operate, but the altitude and speed requirement by the Army couldn't be fulfilled.



Figure 1.3 Convertawings model A quadcopter []

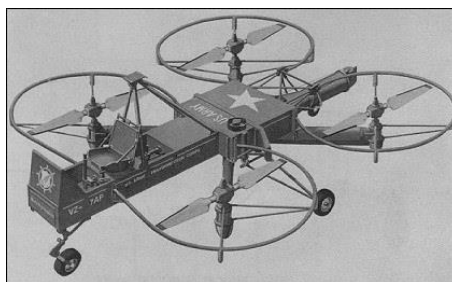


Figure 1.4 Curtiss-Wright VZ-7 []

Over the last few decades, with the development of electric motors, microelectronics, and micromechanical devices, it became possible to build small size, lightweight, reliable, and cheap modern quadcopters. The modern quadcopter first emerged in the late 1990s and early 2000s as

hobbyist kits. In 2006, the first commercial quadcopters and other UAV permits were released by the FAA, and in the same year, Frank Wang found DJI, whose idea was to revolutionize the use of quadcopters to the mainstream population. Up to now, DJI has released a series of quadcopter products such as the Flame Wheel series, the Phantom series, the Mavic series, the Spark series, and so on. Later in 2010, the French drone manufacturer Parrot unveiled its AR Drone (Figure 1.5 (a)), the first commercially successful ready-to-fly consumer drone, and the first able to be controlled solely by a Wi-Fi connection. In 2016, DJI released the Phantom 4 (Figure 1.5 (c)), which had computer vision and machine learning ability to track a person or an object on the ground without simply following a GPS track. In 2018, DJI and its competitors, like Autel and Parrot continued to release new models and grew both in the hobby sector as well as the professional sector. Then in 2019, in response to the new regulations of the US and Europe that any drone over 250 grams had to be registered, DJI released the Mavic Mini (Figure 1.5 (d)), a 4k stabilized drone under 250 grams that had unprecedented features for such a small quadcopter.



(a) Parrot AR drone



(b) Flame Wheel F450



(c) DJI Phantom 4



(d) DJI Mavic Mini

Figure 1.5 Modern quadcopters

1.2 Modern quadcopters applications

As one of the most popular UAVs at the moment, modern quadcopters feature the advantages of less power consumption, less risk to human life, ease of operation, simple structure, Vertical Take-off and Landing (VTOL), strong maneuverability, and stable hovering performance, which makes them more and more concerned and their applications more and more extensive. The potential application fields of modern quadcopters include civil, environmental, and defense sectors.

The civil application of modern quadcopter includes photography, construction, mining, delivery, agriculture, disaster management, surveillance, and so on (Figure 1.6). Modern quadcopters make aerial photography become very easy, cheap, and no longer just for professional photographers. With no need for helicopters, ground and onboard crews, just one quadcopter pilot and a camera operator, aerial photography that is affordable for ordinary people can be achieved. One of the applications of quadcopters in agriculture is precision farming. Precision farming is based on data collection and variability mapping of agricultural lands, and data analysis. Then farming management such as pesticide spray, irrigation, and fertilizer can be implemented based on the results of the collected and analyzed data. The traditional satellite-based mapping technology is costly, and the data collection can be easily influenced by weather conditions. Using a quadcopter equipped with special devices such as thermal sensors, RGB cameras, and LIDAR systems, the mapping can be implemented more accurately and cheaply. Also, a quadcopter can be equipped with spray, fertilizer, and irrigation devices to perform more precise crop spraying, fertilizing, and irrigating. In the field of aerial surveillance, quadcopters with high-resolution cameras and various security equipment will be widely used in the security system of families and companies. This will not only realize real-time monitoring from multiple precepts and a wide range but also make security work safer and easier. As for the application of quadcopters in delivery, it will greatly reduce the delivery time, increase the delivery radius by skipping roads, traffic, and buildings, and lower the human cost. But the power supply limitation due to the battery capacity will limit the delivery range and working time, and the load capacity of the quadcopter will also limit the payload requirements. The quadcopter delivery is still in development, researchers are working to overcome issues associated with the delivery scope limitation by proposing an autonomous battery-replacement infrastructure [].



(a) Aerial photography



(b) Agriculture application



(c) Aerial surveillance



(d) quadcopter used for delivery

Figure 1.6 Modern quadcopter applications in the civil sector

In the environmental field, applications of quadcopters include air quality monitoring, soil monitoring, mountain inspection, mapping & surveying, and so on. Future air quality monitoring can be implemented by using a remote-controlled quadcopter equipped with a telemetric device (Figure 1.7 (a)). And the measurements can be obtained almost in real-time and in different geometries – vertical and horizontal []. The applications of quadcopters to mountain inspection can effectively implement forest patrol, fire prevention, and wildlife protection. As for mapping & surveying, the quadcopter-based mapping technique can overcome the shortcomings of the traditional aircraft mapping operations. The remote-controlled quadcopters are the most powerful tools for safely capturing accurate aerial data quickly. Importantly, they can also create aerial maps and 360-degree virtual tours in real-time, even before they have landed (Figure 1.7 (b)).



(a) Micro station in UAV Air Quality Monitoring []



(b) image of drone mapping for the smart city []

Figure 1.7 Quadcopter applications in the environmental sector

As for the applications in defense, quadcopters can be utilized for anti-terror, border security monitoring, aerial reconnaissance, bomb-dropping, and so on. USA, UK, Russia, India, and Israel are the leading countries in the development and deployment of military drones []. The growing application of drones in the military and defense sector will propel the growth of the global drone market by 2027.

1.3 State-of-the-art control

As mentioned above, the quadcopter is an inherently unstable and under-actuated system, this brings nonlinear dynamics to the quadcopter system that makes the control very difficult. The flight controller design of the quadcopter is of crucial importance to achieve safe operation, reliable stabilization, autonomous flight, trajectory tracking, and robustness to unpredictable changes in the environment. Over the past years, various control methods have been implemented by researchers for quadcopters control. These control methods can be divided into three categories: linear control method, nonlinear control method, and learning-based control method.

Even though the quadcopter is a nonlinear system, linear control techniques are still useful and good control performance has been achieved now. The linear control is achieved by firstly

linearizing the non-linear quadcopter model around an equilibrium point, then deploying the linear controller. The mostly used linear control methods are the Proportional – Integrative – Derivative (PID) control, the Linear Quadratic Regulator (LQR) control, the H-infinity control, and the gain schedule. In [], a classic PID controller is implemented to control the attitude of the quadcopter and good control performance is achieved. In [], a cascade PID feedback controller is proposed to stabilize the attitude of the quadcopter, and the cascade PID controller is proved to be more effective and robust compared to the classic one. [] exploits the cascade PID controller to track the given trajectory, simulations considering the disturbances are implemented and the results are very satisfactory. The LQR-based control methods are also showing very good control performance. In [], the author utilizes an LQR controller to implement the attitude, position, and altitude control of the quadcopter. [] proposed two LQR controllers with integration action to control the position and yaw orientation of the quadcopter, the integration term is used to improve the quadcopter tracking performance. In [] the controller is in an inner-outer loop structure and the testing results on a real quadcopter platform show that the controller can provide good results in trajectory tracking and robustness to perturbations. In general, linear control methods can achieve the stability of the quadcopter around the chosen operating point, usually the hovering state, but when the quadcopter's state is far from the equilibrium, the control performance will be poor, leading to the instability of the quadcopter. To overcome this limitation, a group of linear controllers is designed based on several operating points, this is the known gain schedule approach. The gain schedule method can incorporate both linear and nonlinear controllers [] to improve the control performance. In [], a gain scheduled LQR controller is exploited taking into consideration the different yaw angles of the quadcopter during flight. The linearization is performed based on the current yaw angle, the designed control eliminates the limitation of the quadcopter during the trajectory tracking and improves the tracking performance. As for the H-infinity control, [] have shown great control performance of implementing this approach to the quadcopter.

Linear control methods are not able to handle all the system behaviors since the controller is designed based on the linearization of the model around an equilibrium point. Better control performances can be achieved by using nonlinear control methods that consider a more general form of the dynamics of the quadcopter in all flight zones []. Nonlinear control methods like backstepping, sliding mode (SMC), feedback linearization (FL), adaptive control, and model predictive control (MPC) have been proved to be very effective in quadcopter control. The state feedback linearization approach is implemented in [] to get a linear model of the quadcopter, the linearized system can be controlled with a linear control approach like PID, LQR, MPC, etc., and works of literature show that this configuration achieves very good performance in quadcopter trajectory tracking. [] present several cases where simple sliding mode control (SMC) has been successfully implemented. Even though the SMC has shown acceptable performance, it is often accompanied by the chattering issue. In [], the author attempts to avoid the undesired chattering effect by using 2-order sliding mode control (2-SMC), which acts directly on the second-derivative of the sliding surface, and the results demonstrate that the 2-SMC is superior in comparison with the simple SMC. Model predictive control shows the capability of working with constraints and disturbances. In [], the author has implemented a linear MPC to the quadcopter trajectory tracking under disturbances in a simulation environment. In [], the linear MPC is used in combination with the feedback linearization approach, and the simulation results demonstrate that the designed controller can effectively ensure the trajectory tracking under constraints and continuous

disturbances. Despite the linear MPC features above advantages, it can't handle the nonlinearity of the quadcopter, leading to a degree of performance degradation. In [], the author implemented the nonlinear MPC to the quadcopter trajectory tracking. The proposed controller can not only handle the nonlinearity of the quadcopter but also make use of the inherent capabilities of MPC. As for the adaptive control, it's usually coupled with other controllers like PID, LQR, back-stepping, and SMC. This kind of controller usually includes two loops where one is used for the normal feedback loop, and another one is used for parameter adjustment [].

learning-based control methods include fuzzy logic control and neural network control. The fuzzy logic control can be implemented on a quadcopter in a standalone approach or combined with other control. In [], the author proposes a Fuzzy-PID controller to study the roll and pitch angles stability on a circular trajectory, simulations with both the Fuzzy-PID controller and the PID controller are implemented, and the Fuzzy-PID controller has relatively smaller errors and better robustness than the PID one. In [] the Fuzzy-PID controller is implemented to the quadcopter trajectory tracking and shows better performance than the PID controller. Also, neural network control has been extensively used in quadcopter control. In [], the author proposes to use the neural network to continually adjust the PID parameters for minimizing the tracking error. Furthermore, in [] the reinforcement learning approach is implemented in the quadcopter's automatic landing.

1.4 outline

The rest of this thesis is organized as follows.

Chapter 2 first expounds on the quadcopter's flight mechanism, addressing its four basic motions. Then, the inertial coordinate frame and body coordinate frame that are required for describing the quadcopter's position and attitude are introduced. The concept of Euler angles is introduced to describe the attitude of the quadcopter in space. Finally, two nonlinear quadcopter mathematical models are derived based on Newton's law and the Euler equation. And the linearization of the nonlinear quadcopter mathematical model is implemented around the hovering state.

Chapter 3 demonstrates a cascade LQR controller and its control performance. The cascade controller structure is firstly introduced, then the altitude, attitude, and position controller design are implemented. Finally, a simulation is performed to tune and validate the parameters of the designed controller. The performance of the designed controller is evaluated considering the step signal response and trajectory tracking performance.

Chapter 4

Chapter 5

Chapter 6

Chapter 2

Mathematical model

To analyze the translational and rotational dynamics of the quadcopter and design the flight controller, the mathematical model of the quadcopter needs to be firstly derived. In this section, the preliminary quadcopter flight mechanism and its mathematical model are presented. Firstly, the configuration structure and motion mechanism of the quadcopter is expounded, then the nonlinear mathematical model of the quadcopter is derived and represented in state-space form, and finally, a linearization of the quadcopter mathematical mode is implemented.

2.1 Quadcopter flight mechanism

The quadcopter is a multi-rotor helicopter that is lifted and propelled by four motors which are driven by electronic speed controllers. Four motors with the same structure and radius are symmetrically located on the four edges of a cross formed by two arms and located on a plane at the same height. According to the position of the defined nose, two configurations are available for a quadcopter: the X (cross) configuration and the + (plus) configuration ([Figure 2.1](#)). The X (cross) configuration is mostly used, this type of configuration maximizes the moments generated by the motor thrust and is more flexible compared to the + (plus) configuration. Due to the above advantages, the X (cross) configuration is adopted in this project.

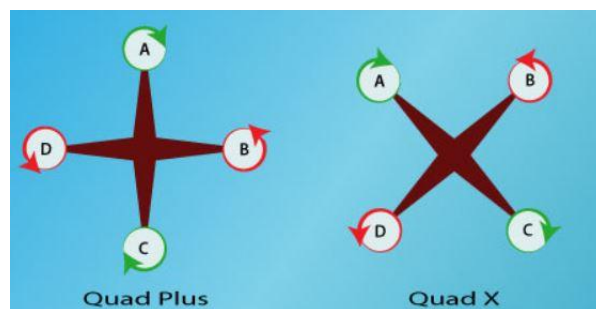


Figure 2.1 Two types of quadcopter configuration

For a quadcopter to fly, it must be capable of three different types of movement: vertical movement, lateral movement, and rotational movement. Based on Newton's third law, each of these can be achieved using the quadcopter's four propellers. The basic motions of a quadcopter are demonstrated in [Figure 2.3](#) below.

The vertical movement is achieved by thrust generated by the four motors, when the propellers spin, each of the propellers will create a thrust, and the total thrust for the quadcopter will be the sum of the four propellers' thrusts. As [Figure 2.3 \(c\)](#) shows when the power of each motor is increased at the same time, the increase of the rotor speed increases the total thrust. When the

total thrust is enough to overcome the gravity, the quadrotor will take off vertically from the ground; otherwise, the power of each motor will be reduced at the same time, and the quadrotor will descend vertically until it hits the ground. Once in the air, the quadcopter can hover with no vertical movement by having the thrust and gravity equal. The key to vertical movement is to ensure that the rotational speed of the four motors increases or decreases synchronously.

The rotational movements include roll motion, pitch motion, and yaw motion, both can be controlled to desired values by changing the speeds of the four motors. The rotational motions are shown in [Figure 2.3\(a\) \(b\)](#). The pitch motion and roll motion are caused by the unevenness of the motors' thrusts on the quadcopter's two sides, the aerodynamic torque effect, and the gyroscopic effect. The unbalanced thrusts on the quadcopter's two sides generate a moment, which pitches or rolls the quadcopter. The gyroscopic effect is due to the rotation of the rotors and is only taken into consideration in the lightweight construction quadcopter. The yaw motion can be achieved by the reactive torque produced by the motor. The magnitude of the reactive torque is relative to the motor speed. When a motor spins, a torque is produced, according to Newton's third law of motion, an equal and opposite torque is produced as well, which is the reactive torque. The sum of reactive torques generated by the four motors is the yaw moment, which causes the yaw motion of the quadcopter. As is shown in [Figure 2.3 \(d\)](#), to control the yaw motion, two motors locate on the same arm rotate in a clockwise direction, while the left ones rotate in a counterclockwise direction to cancel out the reactive torque. The yaw motion control can be achieved by increasing the speeds of the motors in one direction and/or decreasing the speeds of the motors in opposite direction. While having pairs of motors rotating in opposite directions effectively controls the yaw moment acting on the quadcopter, it presents a new problem. If motors spin in opposite directions, two of the propellers push air upward while two push air downward. When combining these forces, the total lift is zero and the quadcopter cannot take off. To overcome this, two different types of propellers are used for the quadcopter. In [Figure 2.2](#), the leftmost propeller has the left edge higher in the front while the other has the right edge higher. This design changes the direction the air is pushed by the two propellers and eventually makes sure that all the propellers push the air in the same direction.



Figure 2.2 Two Propellers with different blade angles

As discussed in chapter 1, the quadcopter is an under-actuated system. The lateral movement is strictly coupled with the rotational motion. This means that for the quadcopter to move forward or to move sideways, it has to tilt forward or sideways first. When the quadcopter tilts, the total thrust acts at an angle, in this case, part of the force of the thrust is upward and part of it is to the side, resulting in lateral movement that can be from side to side or forward and backward. As is shown in [Figure 2.3 \(a\) \(b\)](#), lateral movement occurs by varying the speed of the propellers, increasing the speed of the two propellers on one side of the quadcopter and/or decreasing the

speed of the two propellers on the other side creating uneven amounts of lift on the two sides, the lift created on the side with the faster spinning propellers is greater than the lift created on the opposite side. The result is that the quadcopter moves in the direction of the side where less lift is created.

The quadcopter is an inherently unstable system, when a disturbance occurs, it will lose its stability quickly. Hence, a flight controller is required to continuously control and adjust the four motors' speeds to maintain stability. Thanks to the quadcopter's configuration, the propeller's design, and the motor's spinning direction, each basic motion can be performed separately from the others. Engineers can design a corresponding controller for one basic motion individually, which makes the design and tuning of the flight controller very simple.

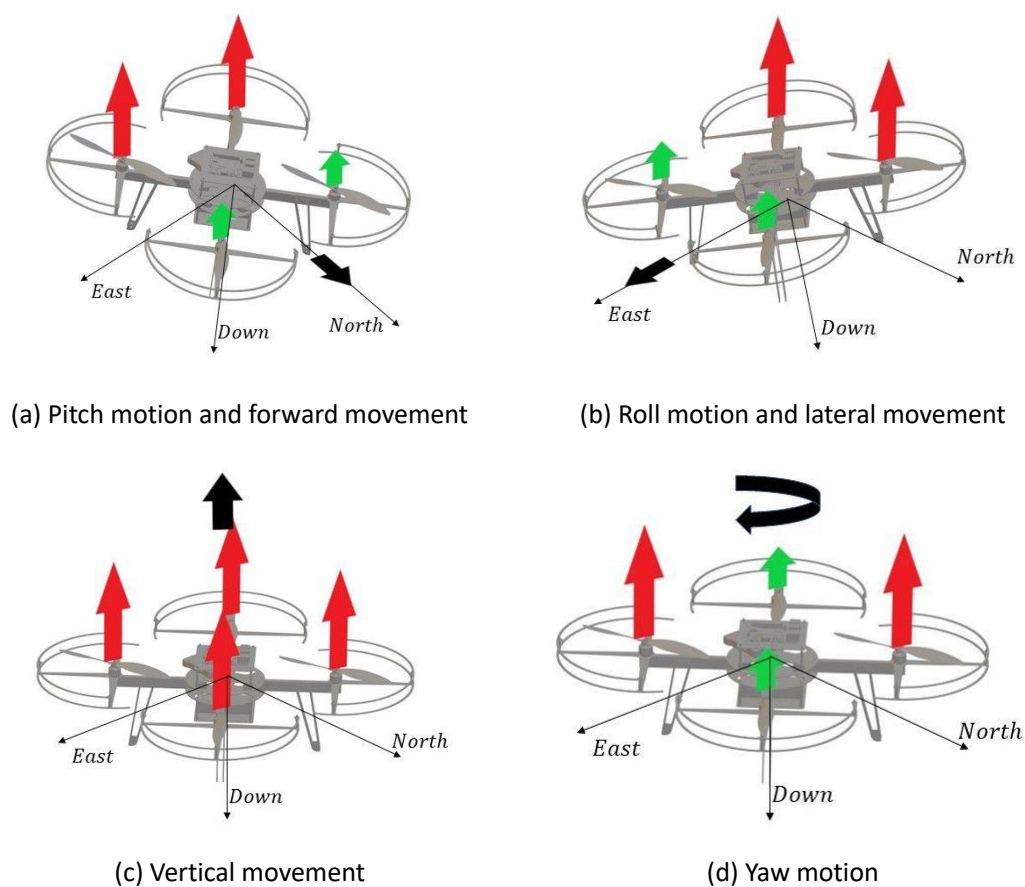


Figure 2.3 Basic maneuvers of a quadcopter []

2.2 Quadcopter mathematical model

Considering a quadcopter as a rigid body aircraft, its dynamics can be divided into two parts, one of which is the center of the mass movement, and the other is the rotational movement of the body around the center of mass. The movement of the body's center of mass is namely the translational dynamics and the rotational movement of the body is the rotational dynamics. In this

project, Newton's second law and Euler moment equation are used to derive the translational and rotational dynamics, respectively. To describe the position and attitude of the quadcopter in 3D space and the torque and thrust acting on it, an appropriate coordinate system should be defined firstly, then kinematics equations and dynamic equations are implemented based on the coordinate system. Finally, the mathematical model of the quadcopter kinematics and dynamics is developed, represented in the State-Space form and a linearization of the model is performed. The State-Space representation of the linearized model plays a key role in developing the control algorithm and making considerations about the stability and dynamics of the quadcopter.

2.2.1 Assumptions

Some assumptions must be done before developing the mathematical model:

1. The quadcopter and all its components are assumed to be a rigid body thus vibrations and deformations during the flight are neglected.
2. The quadcopter is supposed to be perfectly symmetric and the four motors of which are located on the same plane and perfectly equally spaced from the quadcopter's geometric center.
3. The Center of Gravity of the quadcopter coincides with its geometric center.
4. Forces and torques generated by the wind are neglected
5. The thrust provided by motors is supposed to be constant concerning the environmental conditions (air density, temperature, and altitude).
6. The Earth is considered as flat, and its rotation is negligible concerning body angular speeds.

2.2.2 Coordinate frames

Given an aircraft, the positive x direction in the body coordinate frame is usually defined as the moving direction, the positive y direction as the right side of the moving direction, and the positive z direction as the vertically downward direction; this is also termed forward-right-downward coordinate frame. The body coordinate frame follows the movement of the aircraft, and the origin coincides with the aircraft's center of mass.

To facilitate navigation and waypoint tracking, the axis directions of the inertial coordinate frame are chosen as North-East-Down (NED) navigation frame directions. As is shown in [Figure 2.4](#): the inertial coordinate frame is identified by $x - y - z$ axes, where the z axis is perpendicular to the ground and points toward the center of the earth, the x axis points toward the geometric north, and the y axis points toward the east, $\hat{e}_x, \hat{e}_y, \hat{e}_z$ are three unit vectors used to identify the direction of the three axes. The origin of the body coordinate frame is centered at the quadcopter's Center of Gravity and is identified by $x_b - y_b - z_b$ axes, $\hat{e}_1, \hat{e}_2, \hat{e}_3$ are three the unit vectors used to describe the direction of the three axes.

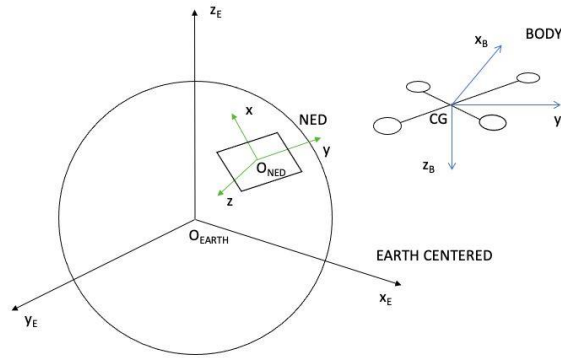


Figure 2.4 NED coordinate frame and body coordinate frame

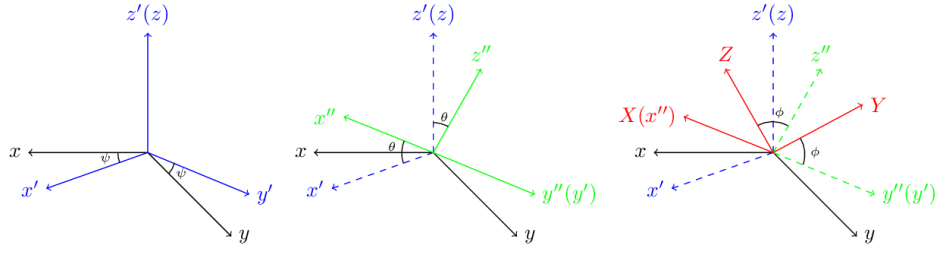
2.2.3 Rotation and Euler angles

Euler angles and Quaternion are mostly used to describe the rotation and orientation of a plane in a three-dimensional space. The quaternion uses four parameters to stand for a rotation, it's a very efficient rotation operator and is mostly used in the cases when the aircraft needs to do a full rotation. The Euler angles are simpler and more intuitive compared to the quaternion, but they are limited by a phenomenon called "Gimbal Lock". In this work, Euler angles are chosen to represent the rotation of the quadcopter since the quadcopter's pitch and roll motions are limited and the quadcopter doesn't need to do a full rotation.

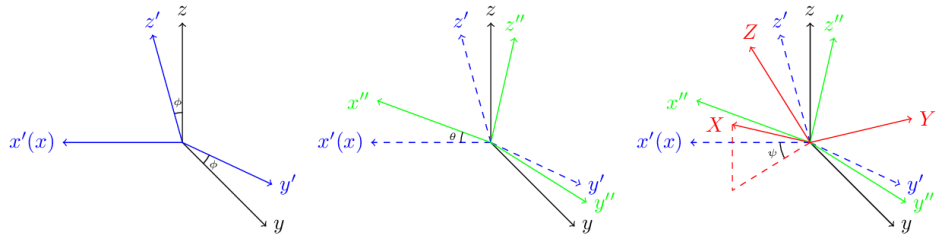
Euler angles are three angles introduced by Leonhard Euler to describe the rotation of a rigid body in three-dimensional space. Any three-dimensional rotation can be decomposed by a sequence of three elementary rotations around three axes (consecutive rotation must be around two different axes), and the rotation angles around three axes are Euler angles. Without considering the possibility of using two different conventions for the definition of the rotation axes (intrinsic or extrinsic), there exist twelve possible sequences of rotation axes, divided into two groups: Proper Euler angles and Tait-Bryan angles. The three elemental rotations may occur either about the axes of the original coordinate system, which remains motionless (extrinsic rotations), or about the axes of the rotating coordinate system, which changes its orientation after each element rotation (intrinsic rotation). Tait-Bryan 321 angles [Figure 2.5](#), following $z - y' - x''$ (intrinsic rotation) or $x - y - z$ (extrinsic rotation) convention, are also known as yaw, pitch, and roll angles, and are mostly used in aerospace to describe the orientation of a ship or aircraft. In this work, Tait-Bryan 321 angles are used to describe the rotation of the quadcopter body coordinate frame relative to the inertial coordinate frame. The rotation matrix of the quadcopter body coordinate frame can be derived by multiplying three elementary matrixes.

The roll pitch and yaw angles of an aircraft are demonstrated in [Figure 2.6](#). The roll angle values range from -90° to 90° where zero means horizontal, 90° means full roll right and -90° means full roll left. The pitch angle values also range from -90° to 90° where zero means horizontal, 90° means straight up and -90° means straight down. The yaw angle values range from 0 to 360° , which gives information about the quadcopter's direction in space. In the attitude and heading reference system (AHRS), the yaw angle represents the direction of the aircraft to the north, where 0° means the aircraft pointing North, 90° means pointing East, 180° means

pointing South, and 270° means pointing West.



(a) Intrinsic rotation axes in order $z - y' - x''$



(b) Extrinsic rotation axes in order $x - y - z$

Figure 2.5 Tait-Bryan angles

For better readability, the following notation is introduced:

$$\cos(*) = c_{(*)} \quad \sin(*) = s_{(*)} \quad \tan(*) = t_{(*)}$$

The basic elementary rotations around x , y , and z axes can be described as:

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi & c_\phi \end{bmatrix} \quad R_y = \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \quad R_z = \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2-1)$$

The notation ϕ , θ , ψ represents roll pitch and yaw angle around three axes respectively. By multiplying the three matrixes, the rotation matrix from the body coordinate frame to the inertial coordinate frame can be described as:

$$R_{body}^{NED} = R_z R_y R_x = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} \quad (2-2)$$

The rotation matrix R_{body}^{NED} is orthogonal, thus $R_{body}^{NED^{-1}} = R_{body}^{NED^T}$, which is the rotation matrix from the inertial coordinate frame to the body coordinate frame.

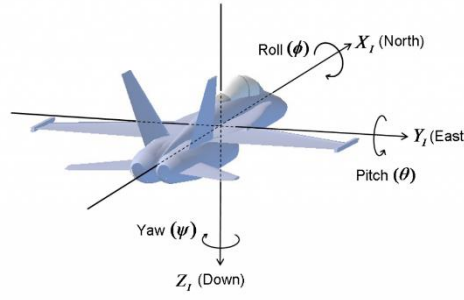


Figure 2.6 Roll pitch and yaw angles of an aircraft

2.2.4 Kinematics equations

Kinematics describes the motion of the quadcopter without considering the forces that cause it to move. Defining $V = [\dot{x} \ \dot{y} \ \dot{z}]^T$ the velocity of the quadcopter in the inertial coordinate frame and $V_b = [u \ v \ w]^T$ the velocity of it represented in the body frame, the rotation matrix transfers the velocity represented in the body coordinate frame to the inertial coordinate frame:

$$V = R_{body}^{NED} \cdot V_b \quad (2-3)$$

As in [], denoting the angular velocity of the quadcopter represented in inertial coordinate frame $\omega = [\dot{\phi} \ \dot{\theta} \ \dot{\psi}]^T$ and the angular velocity represented in the body coordinate frame $\omega_b = [p \ q \ r]^T$, it's possible to find these two angular velocities are linked by the following relations:

$$\omega_b = T \cdot \omega \quad (2-4)$$

Where T denotes the transformation matrix for angular velocities from the inertial coordinate frame to the body coordinate frame:

$$T = \begin{bmatrix} 1 & 0 & -s_\theta \\ 0 & c_\phi & s_\phi c_\theta \\ 0 & -s_\phi & c_\phi c_\theta \end{bmatrix} \quad (2-5)$$

In the case of $\theta \neq (2k-1)\pi/2, (k \in \mathbb{Z})$, T is invertible and T^{-1} denotes transformation matrix of angular velocities from the body coordinate frame to the inertial coordinate frame:

$$\omega = T^{-1} \cdot \omega_b \quad (2-6)$$

$$T^{-1} = \begin{bmatrix} 1 & s_\phi t_\theta & c_\phi t_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & \frac{s_\phi}{c_\theta} & \frac{c_\phi}{c_\theta} \end{bmatrix} \quad (2-7)$$

Combining [Equation 2-3](#) and [Equation 2-6](#), the kinematic equation of the quadcopter can be written as:

$$\begin{cases} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = R_{body}^{NED} \cdot \begin{bmatrix} u \\ v \\ w \end{bmatrix} \\ \dot{\phi} = p + r[\cos(\phi) \tan(\theta)] + q[\sin(\phi) \tan(\theta)] \\ \dot{\theta} = q[\cos(\phi)] - r[\sin(\phi)] \\ \dot{\psi} = r \frac{\cos(\phi)}{\cos(\theta)} + q \frac{\sin(\phi)}{\cos(\theta)} \end{cases} \quad (2-8)$$

2.2.5 Dynamics equations

Dynamics equations describe the movement of the quadcopter under the action of external forces and torques. The solution of these equations supplies a description of the position, the motion, and the acceleration of the quadcopter as a function of time. In this section, Newton's second law and the Euler moment equation are used to derive the translational and rotational dynamics equations of the quadcopter, respectively. The two equations are written as follows:

$$F_b = m(\omega_b \wedge V_b + \dot{V}_b) \quad (2-9)$$

$$M_b = I \cdot \dot{\omega}_b + \omega_b \wedge (I \cdot \omega_b) \quad (2-10)$$

In the work, F_b is the vector containing the total force applied to the quadcopter in the body frame, m is the total mass of the quadcopter, $\omega_b \wedge V_b$ represents the centrifugal acceleration, I is the inertial matrix of the quadcopter and $M_b = [M_x \ M_y \ M_z]^T$ is the vector containing the total torques applied to the quadcopter in the body frame. Notation \wedge represents cross-product. The inertial matrix I is defined as follows:

$$I = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix}$$

Where I_x, I_y and I_z are the moment of inertia along the axes of the body coordinate frame, the inertia matrix is assumed to be diagonal due to the quadcopter being assumed to be perfect symmetry to the body coordinate frame.

The dynamics equations of the quadcopter can be described as:

$$\begin{cases} F_x = m(\dot{u} + qw - rv) \\ F_y = m(\dot{v} + ru - pw) \\ F_z = m(\dot{w} + pv - qu) \\ M_x = I_x \dot{p} + (I_z - I_y)qr \\ M_y = I_y \dot{q} + (I_x - I_z)pr \\ M_z = I_z \dot{r} + (I_y - I_x)pq \end{cases} \quad (2-11)$$

It should be noted that all quantities in dynamics equations are expressed in the body coordinate frame, and kinematics equations in [Equation 2-8](#) must be used to derive the position and attitude of the quadcopter in the inertial coordinate frame.

The total force applied to the quadcopter is made up of two components: the gravity and the thrust generated by the four motors. The total force represented in the body coordinate frame can be given by:

$$F_b = mgR_{body}^{NED}{}^T \cdot \hat{e}_z + F_t \quad (2-12)$$

Where \hat{e}_z is the unit vector of the z axis of the inertial coordinate frame, g is the gravitational acceleration, and F_t is the total thrust provided by the four motors in the body coordinate frame. Note that in this work the forces due by the wind and other disturbances are neglected. The total thrust generated by the four motors can be written as:

$$F_t = \begin{bmatrix} 0 \\ 0 \\ -F_t \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -(T_1 + T_2 + T_3 + T_4) \end{bmatrix} \quad (2-13)$$

Where T_1, T_2, T_3, T_4 are thrusts generated by the four motors, respectively. It should be noted that the four motors are assumed to produce force only in the z axis direction of the body coordinate frame.

Grouping [Equation 2-11](#), [Equation 2-12](#), [Equation 2-13](#) and isolating $\dot{u}, \dot{v}, \dot{w}$ on the left side of the equation, the result is:

$$\dot{V}_b = \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix} = \begin{bmatrix} rv - qw - g\sin(\theta) \\ pw - ru + g\sin(\phi)\cos(\theta) \\ qu - pv + g\cos(\phi)\cos(\theta) - \frac{F_t}{m} \end{bmatrix} \quad (2-14)$$

Lastly, isolating $\dot{p}, \dot{q}, \dot{r}$ on the left side of [Equation 2-11](#), we can obtain:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{I_y - I_z}{I_x} qr + \frac{M_x}{I_x} \\ \frac{I_z - I_x}{I_y} pr + \frac{M_y}{I_y} \\ \frac{I_x - I_y}{I_z} pq + \frac{M_z}{I_z} \end{bmatrix} \quad (2-15)$$

The total torque $M_b = [M_x \ M_y \ M_z]^T$ applied to the quadcopter in the body coordinate frame is given by:

$$M_b = \tau_b + g_m \quad (2-16)$$

Where $\tau_b = [\tau_x \ \tau_y \ \tau_z]^T$ is the vector containing the control torques provided by the quadcopter's four motors, and $g_m = [g_{mx} \ g_{my} \ g_{mz}]^T$ is the vector containing the gyroscopic torque caused by the combined rotation of the four motors and the quadcopter body. The vector τ_b can be calculated using the following equation referring to [Figure 2.7](#):

$$\begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} (T_1 - T_2 - T_3 + T_4)L\sin(\alpha) \\ (T_1 + T_2 - T_3 - T_4)L\cos(\alpha) \\ -C_1 + C_2 - C_3 + C_4 \end{bmatrix} \quad (2-17)$$

Where L is the distance between the quadcopter's center of gravity and the motor, α is defined as in [Figure 2.7](#), and C_1, C_2, C_3, C_4 are the reactive torques generated by the four motors in z axis direction of the body coordinate frame, respectively. The gyroscopic torque g_m is given by the following relation:

$$g_m = \sum_{i=1}^4 J_p (\omega_b \wedge \hat{e}_3) (-1)^{i+1} \Omega_i \quad (2-18)$$

Where J_p is the moment of inertia of the propellers, \hat{e}_3 is the unit vector in z axis of the body coordinate frame, and Ω_i is the angular velocity of the i^{th} motor. The gyroscopic effect is only taken into consideration in the lightweight construction quadcopter. In this project, the torques due to the wind or any other disturbances and the gyroscopic effect are not taken into consideration, as a result, the total torque $[M_x \ M_y \ M_z]^T$ is equal to the control torques generated by the motor $[\tau_x \ \tau_y \ \tau_z]^T$.

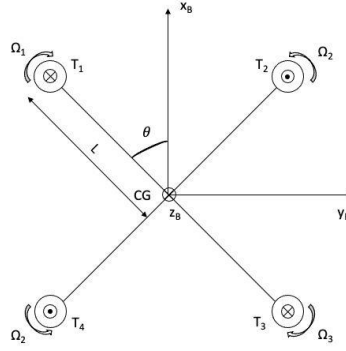


Figure 2.7 Motors configuration of the quadcopter

2.2.6 State-space representation and linearization

Organizing all the equations presented in the previous section, the state-space representation of the quadcopter mathematical model can be written as a nonlinear differential equation of this form:

$$\dot{x} = f(x, u) = f(x) + g(x) \cdot u \quad (2-19)$$

This kind of representation is convenient when dealing with control theories like feedback linearization (FL) and sliding mode control (SMC). x and u are the state variable and input of the system, respectively. Defining x and u as:

$$\begin{aligned} x &= [x \ y \ z \ \phi \ \theta \ \psi \ u \ v \ w \ p \ q \ r]^T \in \mathbb{R}^{12} \\ u &= [-F_t \ M_x \ M_y \ M_z]^T \in \mathbb{R}^4 \end{aligned} \quad (2-20)$$

Grouping [Equation 2-8](#), [Equation 2-14](#), and [Equation 2-15](#) together and rewriting the result following [Equation 2-19](#), it is possible to express the dynamical model in the state-space format like this:

$$\begin{aligned} \dot{x} &= f(x) + g(x) \cdot u \\ f(x) &= \begin{bmatrix} R_{body}^{NED} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \\ p + \tan(\theta) [q \sin(\phi) + r \cos(\phi)] \\ q \cos(\phi) - r \sin(\phi) \\ \frac{1}{\cos(\theta)} [q \sin(\phi) + r \cos(\phi)] \\ rv - qw - g \sin(\theta) \\ pw - ru + g \sin(\phi) \cos(\theta) \\ qu - pv + g \cos(\phi) \cos(\theta) \\ \frac{I_y - I_z}{I_x} r q \\ \frac{I_z - I_x}{I_y} p r \\ \frac{I_x - I_y}{I_z} p q \end{bmatrix} \\ g(x) &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 & 0 \\ 0 & \frac{1}{I_x} & 0 & 0 \\ 0 & 0 & \frac{1}{I_y} & 0 \\ 0 & 0 & 0 & \frac{1}{I_z} \end{bmatrix} \end{aligned} \quad (2-21)$$

Another kind of mathematical model of the quadcopter is presented below. The procedure of the Newton-Euler formalism modeling is to project thrust forces acting on the quadcopter to the inertial coordinate frame and analyze the translational dynamics in the inertial coordinate system and the rotational dynamics in the body coordinate systems. In the inertial coordinate frame, the

centrifugal force is nullified, only the gravitational force and the thrust are contributing to the acceleration of the quadcopter. The translational dynamics equations can be written as:

$$m\ddot{\xi} = mg + R_{body}^{NED}F_t$$

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = g \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} - \frac{F_t}{m} \begin{bmatrix} \cos(\phi) \sin(\theta) \cos(\psi) + \sin(\phi) \sin(\psi) \\ \cos(\phi) \sin(\theta) \sin(\psi) - \sin(\phi) \cos(\psi) \\ \cos(\phi) \cos(\theta) \end{bmatrix} \quad (2-22)$$

Grouping [Equation 2-8](#), [Equation 2-15](#), and [Equation 2-22](#) together, the dynamics equations can be written as:

$$\begin{cases} \ddot{x} = -\frac{F_t}{m} [\cos(\phi) \sin(\theta) \cos(\psi) + \sin(\phi) \sin(\psi)] \\ \ddot{y} = -\frac{F_t}{m} [\cos(\phi) \sin(\theta) \sin(\psi) - \sin(\phi) \cos(\psi)] \\ \ddot{z} = g - \frac{F_t}{m} \cos(\phi) \cos(\theta) \\ \dot{\phi} = p + r[\cos(\phi) \tan(\theta)] + q[\sin(\phi) \tan(\theta)] \\ \dot{\theta} = q[\cos(\phi)] - r[\sin(\phi)] \\ \dot{\psi} = r \frac{\cos(\phi)}{\cos(\theta)} + q \frac{\sin(\phi)}{\cos(\theta)} \\ \dot{p} = \frac{I_y - I_z}{I_x} qr + \frac{M_x}{I_x} \\ \dot{q} = \frac{I_z - I_x}{I_y} pr + \frac{M_y}{I_y} \\ \dot{r} = \frac{I_x - I_y}{I_z} pq + \frac{M_z}{I_z} \end{cases} \quad (2-23)$$

Defining the state vector and input as:

$$x = [x \ y \ z \ \phi \ \theta \ \psi \ \dot{x} \ \dot{y} \ \dot{z} \ p \ q \ r]^T \in \mathbb{R}^{12}$$

$$u = [-F_t \ M_x \ M_y \ M_z]^T \in \mathbb{R}^4$$

It is possible to write the equations in state-space form:

$$\dot{x} = f(x) + g(x) \cdot u$$

$$f(x) = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ p + \tan(\theta) [q \sin(\phi) + r \cos(\phi)] \\ q \cos(\phi) - r \sin(\phi) \\ \frac{1}{\cos(\theta)} [q \sin(\phi) + r \cos(\phi)] \\ 0 \\ 0 \\ g \\ \frac{I_y - I_z}{I_x} qr \\ \frac{I_z - I_x}{I_y} pr \\ \frac{I_x - I_y}{I_z} pq \end{bmatrix} \quad g(x) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ U_x & 0 & 0 & 0 \\ U_y & 0 & 0 & 0 \\ U_z & 0 & 0 & 0 \\ 0 & \frac{1}{I_x} & 0 & 0 \\ 0 & 0 & \frac{1}{I_y} & 0 \\ 0 & 0 & 0 & \frac{1}{I_z} \end{bmatrix} \quad (2-24)$$

Where:

$$U_x = \frac{1}{m} [\cos(\phi) \sin(\theta) \cos(\psi) + \sin(\phi) \sin(\psi)]$$

$$U_y = \frac{1}{m} [\cos(\phi) \sin(\theta) \sin(\psi) - \sin(\phi) \cos(\psi)]$$

$$U_z = \frac{1}{m} [\cos(\phi) \cos(\theta)]$$

The mathematical model of the quadcopter is a quite complex system due to its nonlinearity, which makes it difficult to gain an accurate insight into the quadcopter's behavior and to troubleshoot control systems in simulation. To develop basic control strategies like linear control, several assumptions that reduce the complexity of the non-linear equations are required. Based on these assumptions, the linearized mathematical model of the quadcopter is derived, and then its control strategy can be formulated. In this project, the Linear Quadratic Regulator (LQR) control strategy is adopted, and the controller parameters are tuned based on the linearized model. The designed controller can achieve acceptable control performance to some extent when deployed on the nonlinear model. In future works, the complexity of the system model can be increased, and then more advanced nonlinear control strategies such as Feedback Linearization Control, Sliding Mode Control (SMC), and Nonlinear Model Predictive Control (NMPC) can be deployed to implement much more precise control of the quadcopter.

The assumption made to simplify the model is that the quadcopter will be operated around a stable hover condition with small attitude angles and minimal rotational and translational velocities and accelerations. These assumptions are described by the mathematically functions below:

$$\begin{aligned}\dot{x} &= \dot{y} = \dot{z} = 0 \\ \dot{\phi} &= \dot{\theta} = \dot{\psi} = 0 \\ \phi &= \theta = 0 \\ \psi &= 0\end{aligned}\tag{2-25}$$

The linearization is made by approximating the sine function with its argument and the cosine function with unity. The resulting simplified and linearized equations are:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} \approx \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \approx \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}\tag{2-26}$$

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} -\frac{F_t}{m} [\theta \cos(\psi) + \phi \sin(\psi)] \\ -\frac{F_t}{m} [\theta \sin(\psi) - \phi \cos(\psi)] \\ g - \frac{F_t}{m} \end{bmatrix} = \begin{bmatrix} -\frac{F_t}{m} \theta \\ \frac{F_t}{m} \phi \\ g - \frac{F_t}{m} \end{bmatrix} \approx \begin{bmatrix} -g\theta \\ g\phi \\ g - \frac{F_t}{m} \end{bmatrix}\tag{2-27}$$

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \frac{1}{I_x} M_x \\ \frac{1}{I_y} M_y \\ \frac{1}{I_z} M_z \end{bmatrix}\tag{2-28}$$

It should be noted that in [Equation 2-27](#) the translational movement has been decoupled from the attitude assuming the yaw angle remains around zero degrees. With this assumption, movement along the global x and y axis can be controlled by independently controlling the pitch and roll angles, respectively. Also, the total thrust F_t is almost equal to the gravity for a quadcopter to maintain a stable altitude during hovering. The simplified equations of the quadcopter model by linearization are:

$$\begin{cases} \dot{x} = \dot{x} \\ \dot{y} = \dot{y} \\ \dot{z} = \dot{z} \\ \dot{\phi} = p \\ \dot{\theta} = q \\ \dot{\psi} = r \\ \ddot{x} = -g\theta \\ \ddot{y} = g\phi \\ \ddot{z} = g - \frac{F_t}{m} \\ \dot{p} = \frac{M_x}{I_x} \\ \dot{q} = \frac{M_y}{I_y} \\ \dot{r} = \frac{M_z}{I_z} \end{cases} \quad (2-29)$$

Once a linear model of the quadcopter dynamics is obtained, and letting matrix $C = I_{12}$ be the output matrix, the model can be written in the state-space form as follows:

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx \end{cases} \quad (2-30)$$

The matrices associated with the linear model are given by relations:

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -g & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 & 0 \\ 0 & \frac{1}{I_x} & 0 & 0 \\ 0 & 0 & \frac{1}{I_y} & 0 \\ 0 & 0 & 0 & \frac{1}{I_z} \end{bmatrix} \quad (2-31)$$

It should be noted that the input vector u for [Equation 2-30](#) should be defined as:

$$u = [mg - F_t \quad M_x \quad M_y \quad M_z]^T \in \mathbb{R}^4$$

Chapter 3

Controller strategy and simulation

In this chapter, the flight controller structure and the control strategy are discussed. The control strategy utilized to design the flight controller is the Linear Quadratic Regular (LQR) control. The linearized mathematical model derived above is used in the LQR controller design. From the mathematical model derived above, the rotational motions of the quadcopter are independent of translational motions and full actuated, while translational motions are underactuated and depend heavily on the rotational motions. Therefore, the cascade controller with an inner-outer loop structure can better control the quadcopter. The inner loop is related to the fast dynamics of the quadcopter, which generates control signals to control the attitude and altitude, while the outer loop is related to the slow dynamics of it, which controls the position in the $X - Y$ horizontal plane. The structure of this cascade controller is shown in [Figure 3.1](#) below.

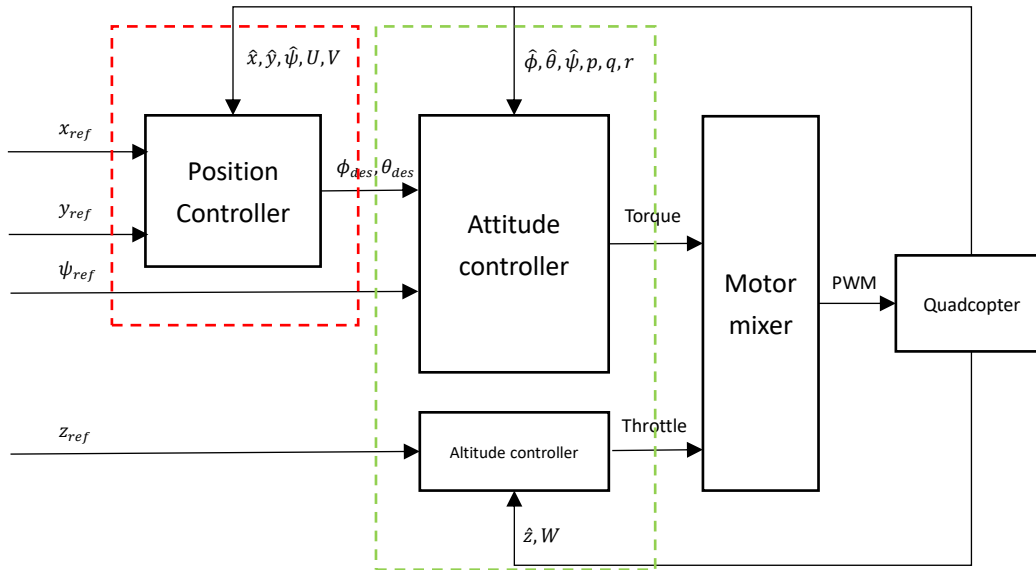


Figure 3.1 Cascade controller

3.1 Linear Quadratic Regulator control

The Linear Quadratic Regular (LQR) control, in the context of optimal control, uses full state feedback to determine the control signal to bring the system's state $x(t)$ to the desired reference $r(t)$ while at the same time minimizing some cost index. Furthermore, LQR control can minimize

the control inputs, thus reducing the use of actuators. For the Linear Time-Invariant system described in the state-space representation:

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx \end{cases} \quad (3-1)$$

The cost index is a function of the state variable $x(t)$ and controller signal $u(t)$.

$$J(x, u) = \frac{1}{2} \int_0^{\infty} (x(t)^T Q x(t) + u(t)^T R u(t)) dt \quad (3-2)$$

Where matrix Q weights the cost of the system state and matrix R weights the cost of actuators. These weighting matrices determine the relative importance of the existing state error as well as the energy expenditure of the system.

The full state feedback control law $u(t)$ which minimizes the cost index function is:

$$u(t) = -Kx(t) \quad (3-3)$$

The computation of the optimal gains is performed through the function:

$$K = R^{-1} B^T P \quad (3-4)$$

Where the positive-definite matrix P results from the steady-state Riccati equation:

$$A^T P + PA + Q - PBR^{-1}B^T P = 0 \quad (3-5)$$

The closed-loop system represented as $\dot{x} = (A - BK)x$ is asymptotically stable with the optimal gain computed by [Equation 3-4](#). Under MATLAB, the algebraic Riccati equation can be solved by the `lqr` function:

$$K = \text{lqr}(A, B, Q, R) \quad (3-6)$$

LQR requires a linear model to get an adequately controlled system, and it can handle multiple inputs and outputs simultaneously and offer a fast response, but unlike the PID control, LQR control often provides static error during tracking due to the lack of an integral part. To overcome this problem, an integrator can be included in the control structure to eliminate the static error and stabilize the system.

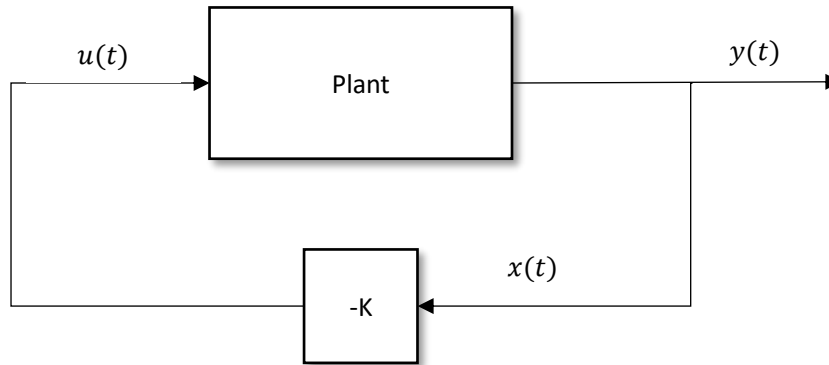


Figure 3.2 LQR control scheme

One of the key questions in LQR controller design is how to choose the weighting matrices Q and R . $Q \in \mathbb{R}^{n \times n}$ is a positive semi-definite matrix that penalizes the state variables and $R \in \mathbb{R}^{m \times m}$ is a positive definite matrix that penalizes the control signals. A simple choice is to use a diagonal weighting matrix:

$$Q = \begin{bmatrix} q_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & q_n \end{bmatrix} \quad R = \begin{bmatrix} \rho_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \rho_m \end{bmatrix} \quad (3-7)$$

For choosing Q and R , the individual diagonal elements describe how much each state and input should contribute to the overall cost index. The larger these values are, the more the state variables and input signals are penalized. Choosing a large value for R means stabilizing the system with less (weighted) energy. This is usually called an expensive control strategy. On the other hand, choosing a small value for R means the energy consumption is little considered (cheap control strategy). Similarly, choosing a large value for Q means that the state variables should remain small and a small value for Q implies less concern about the state variables. Hence, the tuning of weight matrices Q and R is a trade-off between the control performance and the energy consumption.

The simplest method for choosing the elements of Q and R matrix is to make $Q = I, R = \rho I \Rightarrow J = \|x\|^2 + \rho \|u\|^2$, and vary ρ to get something that has a good response. Another method for choosing the individual weights q_i and ρ_i can be done by deciding on a weighting of the errors from the individual terms. Bryson and Ho [] have suggested the following method for choosing the matrices Q and R : (1) choosing q_i and ρ_i as the inverse of the square of the maximum value for the corresponding x_i or u_i ; (2) modify the elements to obtain a compromise among response time, damping, and control effort. This second step can be performed by trial and error.

3.1.1 Altitude controller

The altitude controller belongs to the inner loop part of the flight controller as shown in [Figure 3.1](#). The altitude controller generates the thrust command to control the quadcopter's position along z axis of the inertial coordinate frame (altitude). The unit of the thrust command is Newton [N], in this way, the designing and the turning of the controller are more intuitive. LQR control strategy is used to design the altitude controller, and very good control performance is achieved. The linearized mathematical model ([Equation 2-29](#)) in chapter 2 is used for the LQR controller design. For the altitude subsystem, the related differential equation is:

$$\ddot{z} = g - \frac{F_t}{m} \quad (3-8)$$

To write the differential equation for altitude to the format $\dot{x} = Ax + Bu$, the state variables and the input of the subsystem are defined as:

$$x_z = [z \quad \dot{z}]^T \quad u_z = mg - F_t \quad (3-9)$$

Then the state-space representation of the altitude subsystem is:

$$\begin{aligned} \dot{x}_z &= \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x_z + \begin{bmatrix} 0 \\ 1/m \end{bmatrix} u_z \\ z &= [1 \quad 0] x_z \end{aligned} \quad (3-10)$$

After determining the weighting matrix $Q_z \in \mathbb{R}^{2 \times 2}$ and $R_z \in \mathbb{R}$, the feedback gain $K_z \in \mathbb{R}^{1 \times 2}$ for the altitude control can be computed by MATLAB command `lqr`, and the control law for the altitude can be expressed as:

$$F_t = -1 \cdot (K_z \cdot \begin{bmatrix} z_r - z \\ \dot{z}_r - \dot{z} \end{bmatrix} - mg) \quad (3-11)$$

The block diagram of the altitude controller is shown in [Figure 3.3](#) below:

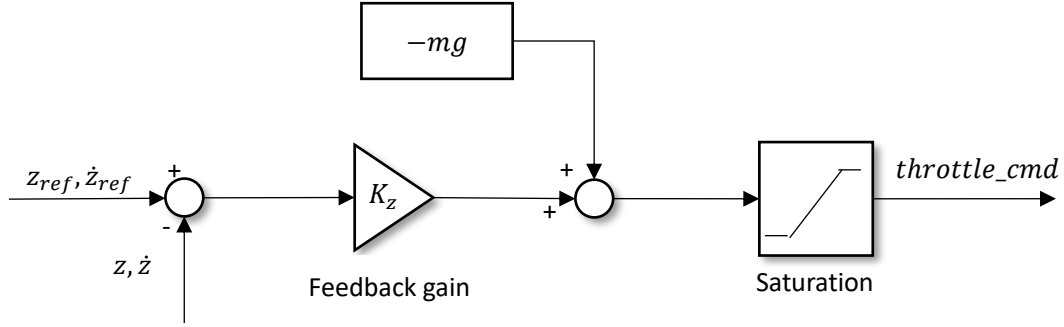


Figure 3.3 Block diagram of altitude controller

Where the $-mg$ is the feedforward gravity term, this is the amount of thrust needed to offset the weight of the quadcopter so that the LQR controller just generates a positive thrust to make the quadcopter go down and a negative thrust to make it go up (The sign is opposite here since we use NED coordinate frame, up is the opposite direction of the coordinate frame). To limit the control command to the feasible values the quadcopter's motor can generate, a saturation block is included in the controller. The limitations are set according to the characteristics of the actuator, and they are listed in below.

3.1.2 Attitude controller

The Attitude controller is related to the rotational dynamics of the quadcopter, which belongs to the inner loop part of the flight controller. The attitude controller computes the desired torques implied to the quadcopter in the body coordinate frame to control the roll, pitch, and yaw angles for tracking the desired references. The unit of the torque commands is Newton-meters [Nm], in this way the designing and tuning of the controller will be more intuitive. The LQR control strategy is used, and the designed controller can achieve zero static error. The linearized mathematical model [Equation 2-29](#) derived in chapter 2 is used for controller design. For the attitude subsystem, the related differential equations are:

$$\begin{cases} \dot{\phi} = p \\ \dot{p} = \frac{M_x}{I_x} \\ \dot{\theta} = q \\ \dot{q} = \frac{M_y}{I_y} \\ \dot{\psi} = r \\ \dot{r} = \frac{M_z}{I_z} \end{cases} \quad (3-12)$$

To write the equations in state-space format, the state vector x_a and input vector u_a for this subsystem are defined as:

$$x_a = [\phi \quad \theta \quad \psi \quad p \quad q \quad r]^T \quad u_a = [M_x \quad M_y \quad M_z]^T \quad (3-13)$$

The state-space representation of the subsystem is:

$$\dot{x}_a = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} x_a + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1/I_x & 0 & 0 \\ 0 & 1/I_y & 0 \\ 0 & 0 & 1/I_z \end{bmatrix} u_a \quad (3-14)$$

$$y_a = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} x_a \quad (3-15)$$

After determining the weighting matrix $Q_a \in \mathbb{R}^{6 \times 6}$ and $R_a \in \mathbb{R}^{3 \times 3}$, the feedback gain $K_a \in \mathbb{R}^{3 \times 6}$ can be computed by solving the algebraic Riccati equation with MATLAB command `lqr`. The feedback gain K_a is a 3×6 matrix, and the control law for the attitude subsystem can be expressed by:

$$u_a = \begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} = K_a \begin{bmatrix} \phi_r - \phi \\ \theta_r - \theta \\ \psi_r - \psi \\ -p \\ -q \\ -r \end{bmatrix} \quad (3-16)$$

The block diagram of the attitude controller is shown below:

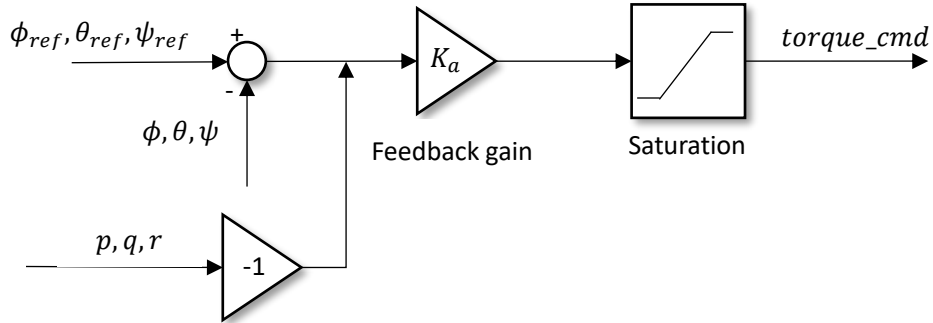


Figure 3.4 Block diagram of the attitude controller

A saturation block is included in the controller to limit the computed torque command to the feasible values the quadcopter's motor can generate. The maximum and minimum values are set according to the characteristics of the motors and the arm length of the quadcopter, and they are listed in below.

3.1.3 Position controller

The outer loop part of the controller refers to the position control of the quadcopter on the $X - Y$ horizontal plane. This loop takes the quadcopter's current heading angle ψ , current position, velocity, and the position reference signal x_r and y_r as inputs and computes the desired roll

angle ϕ and pitch angle θ for tracking the reference position. The unit of the position controller outputs is the radian. In this case, the turning of the controller is more intuitive. The linearized mathematical model derived in chapter 2 is used for designing the LQR controller and the differential equations related to the x and y positions are:

$$\begin{cases} \ddot{x} = -g\theta \\ \ddot{y} = g\phi \end{cases} \quad (3-17)$$

Defining the state variables and the input of the subsystem as:

$$x_p = [x \ y \ \dot{x} \ \dot{y}]^T \quad u_p = [\theta \ \phi]^T \quad (3-18)$$

The subsystem represented in state-space form is:

$$\dot{x}_p = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} x_p + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -g & 0 \\ 0 & g \end{bmatrix} u_p \quad (3-19)$$

$$y_p = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} x_p \quad (3-20)$$

After determining the weighting matrix Q_p and R_p , the full state feedback gain K_p for the position control can be computed by MATLAB `lqr` command. The control law for the position control can be expressed by:

$$\begin{bmatrix} \theta_r \\ \phi_r \end{bmatrix} = K_p \begin{bmatrix} x_r - x \\ y_r - y \\ \dot{x}_r - \dot{x} \\ \dot{y}_r - \dot{y} \end{bmatrix} \quad (3-21)$$

The output of the out-loop controller θ_r and ϕ_r are the desired pitch and roll angles for the quadcopter to track the desired reference, which are the input of the attitude controller part in the inner loop.

The position controller is derived from the model linearizing around a hover point where the yaw angle ψ is assumed to be zero. The movement of the quadcopter in the x axis is decoupled from the roll angle and the movement in the y axis is decoupled from the pitch angle under this assumption, which means giving a roll angle to the quadcopter, it only has a movement in y axis, and giving a pitch angle, it only has a movement in x axis. This simplifies the computation of the desired roll and pitch angles for tracking the reference position. In case when the yaw angle ψ is not zero, the movements of the quadcopter in the xy -plane are related to both roll and pitch angles. A roll angle will not only force the quadcopter to move in y axis but also x axis, and the same is for the pitch angle. To decouple the movement with the attitude as if the yaw angle ψ is zero, a rotation must be considered. The rotation matrix applied to the x and y axes of the body coordinate frame is defined as:

$$R_{body-hover} = \begin{bmatrix} \cos(\psi) & \sin(\psi) \\ -\sin(\psi) & \cos(\psi) \end{bmatrix} \quad (3-22)$$

The block diagram of the position controller is shown below:

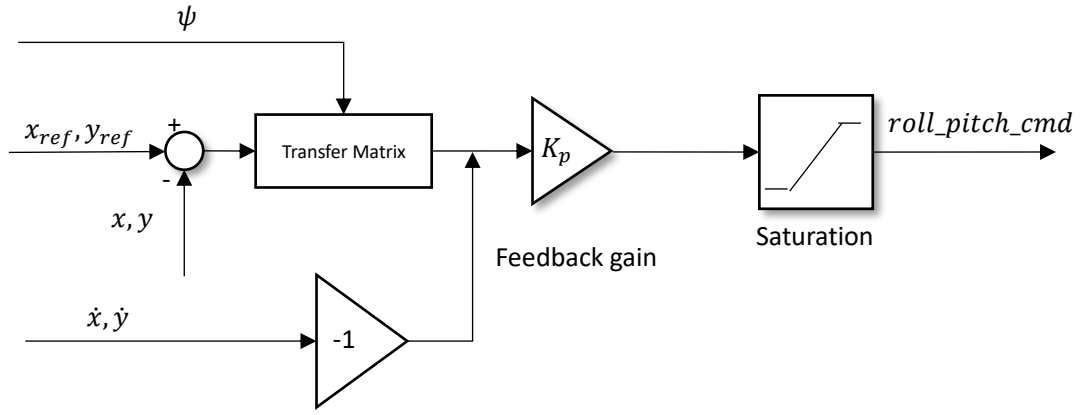


Figure 3.5 Block diagram of the position controller

A saturation is included in the controller to limit computed roll and pitch angle since too large angles make the quadcopter away from the equilibrium point at which the linearization is performed. The control performance of the linear controller will be less effective, leading to the instability of the quadcopter. The limit values are set to $\pm 30^\circ$ when performing the simulation.

3.2 Motor mixer

Grouping [Equation 3-11](#) and [Equation 3-16](#) together, we can get the physical control commands (thrust and moments). But such physical control commands are not enough for implementing the control action to a hardware platform since the real quadcopter platform takes the Power-Width Modulation (PWM) signal as input. The Electronic Speed Controllers of the quadcopter take the PWM signals as input to control and regulate the speeds of the motors, leading to the control action to the quadcopter. To generate the control command that can be used by motor ESC, the transformation between the physical values (thrust and moments) and PWM pulse width values must be implemented.

In this project, the gyroscopic effect of the motor and the torque generated by wind, or any other disturbance are not taken into consideration, so the total torques implied to the quadcopter are completely supplied by the motors. Considering the motor configuration as shown in [Figure 3.6](#), the relationship between physical control commands F_t, M_x, M_y, M_z and the variation of the thrust vector $\Delta T = [\Delta T_f \quad \Delta T_x \quad \Delta T_y]^T$ and torque ΔC can be expressed by:

$$\begin{cases} \Delta T_f = \frac{F_t}{4} \\ \Delta T_x = \frac{M_x}{4L\sin(\alpha)} \\ \Delta T_y = \frac{M_y}{4L\cos(\alpha)} \\ \Delta C = \frac{M_z}{4} \end{cases} \quad (3-23)$$

The first component of the variation vector ΔT_f represents the variation of the thrust required by each motor to provide the quadcopter with a control force along the z axis of the body coordinate

frame, the second component ΔT_x represents the variation of the thrust required by each motor to provide the quadcopter with a control torque along the x axis of the body coordinate frame, and the last component ΔT_y represents the variation of the thrust required by each motor to provide the quadcopter with a control torque along the y axis of the body coordinate frame. The torque variation ΔC represents the variation of the reactive torque required by each motor to provide the quadcopter with a control moment along the z axis of the body coordinate frame.

The relationship between the PWM signal pulse width variations which defined as $\Delta PWM = [\Delta PWM_f \ \Delta PWM_x \ \Delta PWM_y \ \Delta PWM_z]^T$ and the thrust and torque variations ΔT , ΔC need to be found. The first component ΔPWM_f represents the change of the PWM signal pulse width required by each motor to generate control thrust, the second component ΔPWM_x represents the change of the PWM signal pulse width required by each motor to generate the control torque around the x axis, the third component ΔPWM_y represents the PWM pulse width change of each motor to generate control torque around y axis, and the last component ΔPWM_z is the requested PWM signal pulse width change for each motor to generate control torque around z axis.

To find the relationship between ΔPWM , ΔT , and ΔC , the propeller test to find the thrust and torque generated by one motor with respect to the PWM signal pulse width is required. The propeller test can be deployed on a test stand where the torque and thrust generated by the motor can be measured. More detailed information about the propeller test can be found in Appendix A Experimental measurements. The propeller test shows that the thrust and torque generated by the motor have a linear relationship with the PWM pulse width value. Hence the relationship between ΔT , ΔC , and ΔPWM can be expressed by:

$$\begin{cases} \Delta T = n_F \cdot \Delta PWM \\ \Delta C = n_T \cdot \Delta PWM \end{cases} \quad (3-24)$$

Where n_F is the slope of the linear relationship between the thrust generated by each motor and the PWM pulse width value, and n_T is the slope of the linear relationship between the reactive torque of each motor and the PWM pulse width value. Thus, the relationship between the physical control command (thrust and torque) and the PWM pulse width value can be expressed as follows:

$$\begin{cases} F_t = 4n_F \Delta PWM_f \\ M_x = 4n_T L \sin(\alpha) \Delta PWM_x \\ M_y = 4n_T L \cos(\alpha) \Delta PWM_y \\ M_z = 4n_T \Delta PWM_z \end{cases} \quad (3-25)$$

The motor mixer block determines how the four motors of the quadcopter work together to generate the control action. [Figure 3.6](#) presents the motor configuration of the quadcopter considering the body coordinate frame. According to the motor configuration, the 1th motor contributes to the positive thrust, positive torque around x axis, positive torque around y axis, and negative torque around z axis. The motor mixer equation can be simply derived by a linear combination of the required thrust and torques. The same analytical method can be used to derive the motor mixer equations of the other motors. Denoting $PWM_{(i)}$ the PWM pulse width of the i^{th} motor, the motor mixer equations are:

$$\begin{aligned} PWM_{(1)} &= \Delta PWM_f + \Delta PWM_x + \Delta PWM_y - \Delta PWM_z + idel \\ PWM_{(2)} &= \Delta PWM_f - \Delta PWM_x + \Delta PWM_y + \Delta PWM_z + idel \\ PWM_{(3)} &= \Delta PWM_f - \Delta PWM_x - \Delta PWM_y - \Delta PWM_z + idel \\ PWM_{(4)} &= \Delta PWM_f + \Delta PWM_x - \Delta PWM_y + \Delta PWM_z + idel \end{aligned} \quad (3-26)$$

Where *idel* is the pulse width set when the quadcopter is armed (usually equal to 1000). The MATLAB Simulink block diagram of the motor mixer is shown in [Figure 3.7](#) below. The value of the PWM pulse width used by the ESC ranges between 1000 (minimum when armed) and 2000 (maximum), and a saturation block is added to limit the output of the motor mixer.

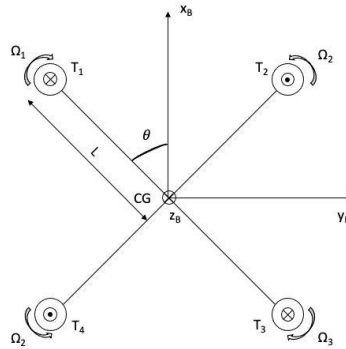


Figure 3.6 Quadcopter's motor configuration

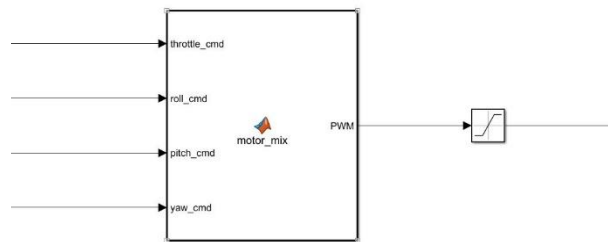


Figure 3.7 Simulink block diagram of motor mixer

3.3 Simulation results

In this section, the complete simulation model is implemented, and the simulation results are analyzed. Simulation is performed to turn and validate the LQR controllers without implementing them directly on the real quadcopter platform, leading to a significant saving on time and material. Firstly, the controller parameters are tuned, and the control performance is evaluated considering the rising time, the settling time, and the overshoot in response to the step input. Then, the trajectory tracking performance of the designed controller is validated and the results are reported and analyzed. In this project, the software used for deploying the simulation model is MATLAB/Simulink R2021b.

3.3.1 preliminary

To perform the simulation, some parameters such as the mass, and the moment of inertia of the quadcopter must be known. These mechanical parameters can be measured with the method proposed in [], and they are listed in [Table 3.1](#) below.

Quadcopter's mechanical parameters	
Parameters	Values
m	$1.7kg$
L	$0.15m$
I_x	$0.023kg \cdot m^2$
I_y	$0.028kg \cdot m^2$
I_z	$0.02kg \cdot m^2$

Table 3.1 Quadcopters parameters

The maximum torque and thrust that the quadcopter's motor can generate should be taken into consideration in the simulation. The computed torque and thrust by the controller must be limited to the maximum values that the quadcopter's motor can achieve. Referring to the propeller test in Appendix A Experimental measurements, as well as the quadcopter mechanical parameters, these maximum values are determined and listed in [Table 3.2](#) below.

	Limitation values
Thrust	30N
Torque around x axis	$\pm 1.5Nm$
Torque around y axis	$\pm 1.5Nm$
Torque around z axis	$\pm 0.02Nm$

Table 3.2 Limitations to the thrust and torque

3.3.2 Model implementation

Some assumptions have to be made to implement the mathematical model in the simulation environment:

1. All quadcopter states (attitude, position, linear velocity, and angular velocity) are assumed to be directly known from the mathematical model (the sensor block and the estimation block are not implemented).
2. The aerodynamics due to wind and any other disturbance are not taken into account (the environment simulation is not implemented and the impact of the environment on the quadcopter is not considered during simulation).
3. The control action is performed at 250Hz frequency, which is the same refresh frequency of the flight controller developed to implement on the quadcopter.

The structure of the Simulink model follows the cascade flight controller structure in [Figure 3.1](#), and it mainly consists of five blocks:

1. Position controller: this block takes the reference position, actual position, current velocity, and current yaw angle as input, and outputs the roll and pitch angle references, which are the desired roll and pitch angles required by the system for tracking the reference position.
2. Altitude controller: this block takes the reference, altitude, and vertical velocity as input, and the throttle command to control the system is given as output.
3. Attitude controller: this block takes the reference, attitude angles, and angular velocity as input and computes desired control torques to ensure that the system can track reference roll pitch and yaw angles.
4. Motor mixer: the motor mixer block transforms the physical control commands (thrust and torques) computed by the controllers into the Power-Width Modulation (PWM) Duty Cycle (DC).
5. Plant: the mathematical models of the actuator and the nonlinear quadcopter model are implemented in this block.

The controller and motor mixer blocks have been previously discussed in section 3.1 and section 3.2, respectively. The plant block includes the nonlinear mathematical model of the quadcopter represented by [Equation 2-24](#) and the actuator model. The block diagram of the plant block is presented in [Figure 3.8](#) below, which takes the PWM signal as input. The actuator block simulates the motors of the quadcopter, which transfers the PWM signal to physical control actions (thrust and torque). To model the actuator, an experimental test is performed on the test stand to find the equations describing the thrust and torque that one brushless motor can generate with respect to the PWM pulse width. More detailed information about the experimental test is presented in Appendix A Experimental measurements. Afterward, the actuator is simulated by a MATLAB function block, and the resulting model provides a relationship between the PWM signal computed by the motor mixer and the thrust and torques the actuator can generate.

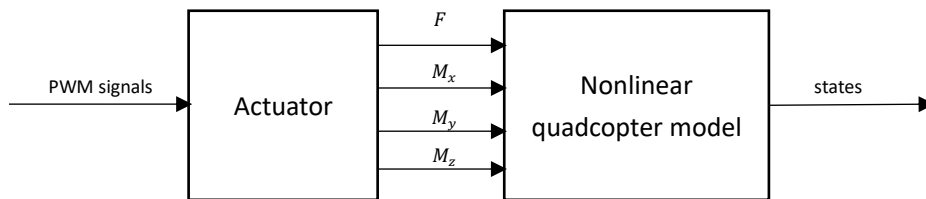


Figure 3.8 Block diagram of the plant

3.3.3 Step response performance

In this section, the controller performance in response to the step signal is simulated. The parameters of the LQR controller are tuned and determined by trial and error, and the designed controller achieves a fast response to commands with a low overshoot and a very small steady-

state error. The parameters used in the simulation are listed in [Table 3.3](#) below, and the x , y , z , roll, pitch, and yaw step signal responses are shown in Figure 3.9. The limitations to the minimum and maximum torque and thrust that the actuator can generate are required considering its capability. The limitation values are listed in [Table 3.2](#) above.

Subsystem	Q matrix	R matrix
Attitude control	$\text{diag}(0.05, 0.1, 0.1, 0.001, 0.001, 0.1)$	$\text{diag}(1, 1, 1) * 10^{-6}$
Altitude control	$\text{diag}(10, 0.5)$	0.01
Position control	$\text{diag}(15, 15, 1, 1)$	$\text{diag}(1, 1) * 500$

Table 3.3 Parameters of LQR controller

The control performance in response to the step signal using the parameters listed in [Table 3.3](#) are depicted in [Figure 3.9](#) below.

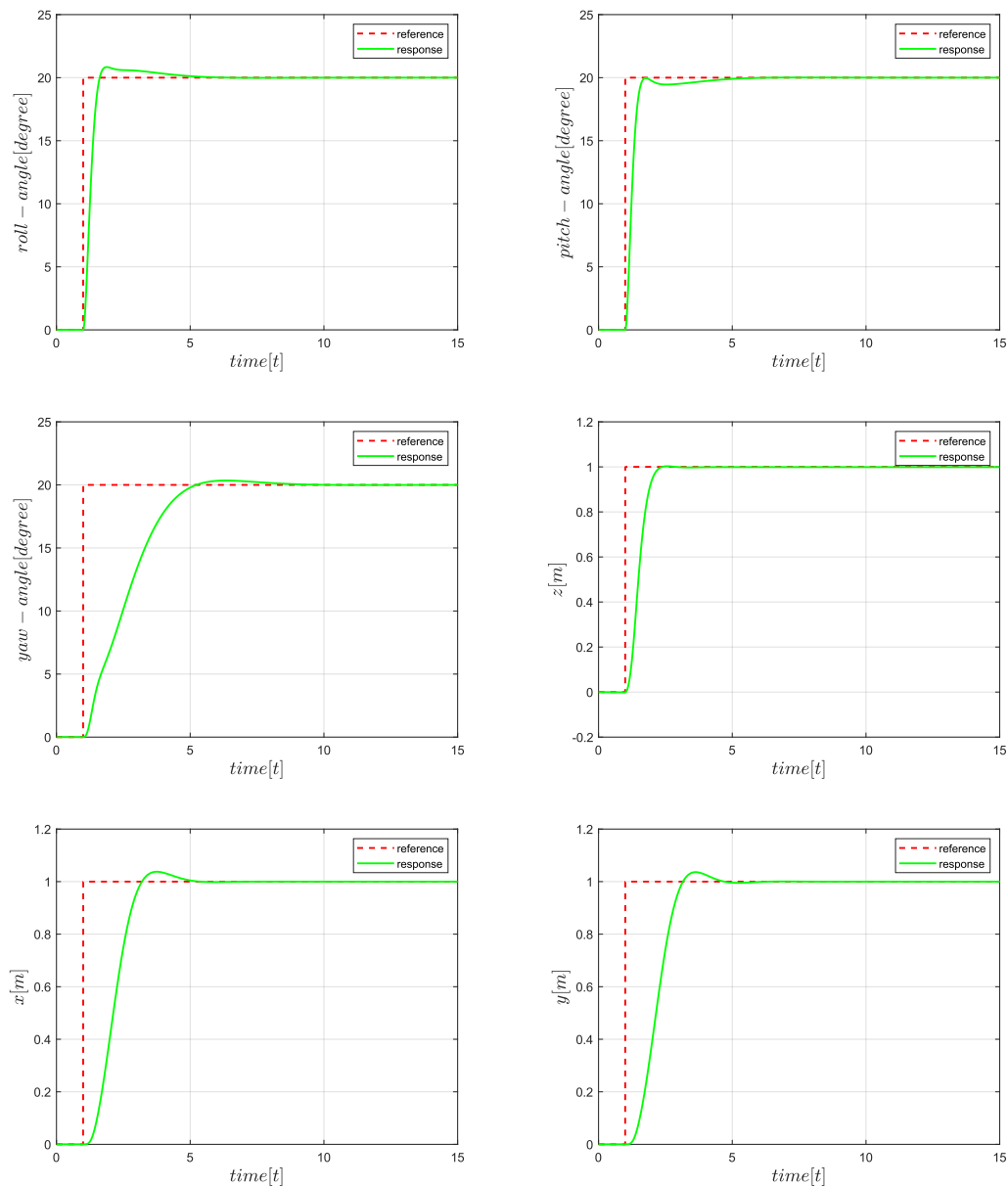


Figure 3.9 Step signal response of the designed controller

The performance of the designed controller is evaluated considering the rising time, settling time, overshoot, and steady-state error. The rising time is defined as the time to reach for the first time the steady-state value of the response. It reflects the responsiveness of the controller. The overshoot is the manifestation of the response that exceeds the reference. The settling time is defined as the time to reach and stay within $\pm\alpha\%$ of the steady-state value of the response. Typical values for α are 1, 2, and 5. In this project, we choose 2 for α . The steady-state value is defined as the difference between the reference and the steady-state value of the response. The response performance of the controller is shown in [Table 3.4](#) below.

	Roll	Pitch	Yaw	X	Y	Z
Rising time	0.62s	0.76s	4.23s	2.20s	2.16s	1.38s
Overshoot	3.90%	0%	1.72%	3.76%	3.62%	0%
Settling time	2.72s	2.43s	3.84s	3.42s	3.13s	1.16s
Steady-state error	0.0002°	0.0002°	0.0018°	0.00007m	0.000037m	0.0011m

Table 3.4 Controller performance in response to the step signal

The designed controller demonstrates a small rising time and settling time in response to roll and pitch commands, leading to a fast roll and pitch response speed, which benefits the position control. The rising time and settling time of the yaw response are much larger compared to the ones of roll and pitch response. The yaw response speed is much slower since the moment around the z axis generated by the actuator is smaller than the moment around the x or y axis. But it's still enough for autonomous flight. In general, the attitude controller shows a fast response speed, small overshoot, and small steady-state error. The altitude controller also shows a fast response with no overshoot. As for the position controller, it belongs to the out-loop controller and the response speed is slower than the one of the inner-loop controller. Also, it has an overshoot. But the overall control performance is acceptable.

Despite being very small, the designed LQR controller always presents a steady-state error due to the lack of integrator action. The LQR controller takes the state and the state derivative as input to compute the control command, which is similar to a classical PD controller. Unlike the PID controller, the lack of integrator action in the LQR controller leads to a non-zero steady-state error issue. Hence, under the influence of noise and environmental uncertainties, the control performance of the LQR will decrease and even lead to system instability.

3.3.4 Trajectory tracking performance

To evaluate the performance of the proposed controller, a trajectory is given as the reference signal for the controller. The trajectory

A series of waypoints and corresponding times of arrival are used to generate a trajectory for the quadcopter. Many methods can be used to generate a trajectory

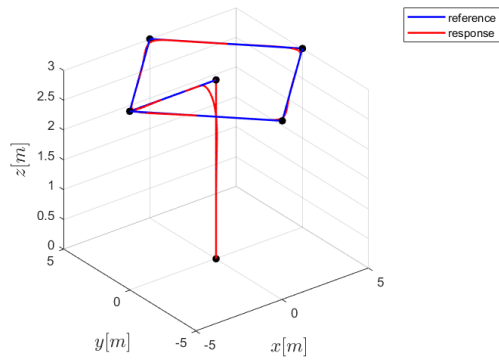
In this section, the trapezoidal speed profile is adopted as the trajectory generation method referring to the previous work in []. The trapezoidal velocity profile imposes a constant acceleration in the start phase, a cruise velocity, and a constant deceleration in the arrival phase []. The trapezoidal velocity profile can be described in Figure 3.10 below.

The square pattern trajectory is given as the reference and Table 3.5 below lists the coordinates of the waypoints in the NED coordinate frame and the corresponding times of arrival.

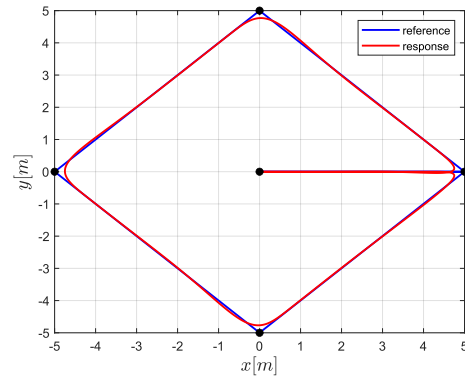
Sequence	Coordinates [m]	Time of arrival [s]
#1	(0, 0, 0)	0
#2	(0, 0, -3)	5
#3	(5, 0, -3)	10
#4	(0, -5, -3)	15
#5	(-5, 0, -3)	20
#6	(0, 5, -3)	25
#7	(5, 0, -3)	30
#8	(0, 0, -3)	35
#9	(0, 0, 0)	40

Table 3.5 Waypoint list of square pattern trajectory

The trajectory tracking performance of the designed LQR controller is shown in (Figure). The simulation results demonstrate that the designed controller allows an acceptable following of the trajectory. The $X - Y$ view of the trajectory in Figure (b) shows that the real response doesn't pass through the defined waypoint, it will go to the next waypoint when it is closest to the current waypoint. This is the drawback of the designed controller. Figure (a) and Figure (b) demonstrate a 1.1s delay and a tracking error in x, y direction between the reference and the response. As for the altitude control performance, Figure (c) shows a good tracking of the position in z axis with small error and delay. (Figure) below compares the velocity calculate by the trajectory and the real quadcopter velocity. Figure (a) and Figure (b) shows a large rising/settling time for tracking the velocity in x, y direction, which leads to the delay in position tracking. The velocity tracking in z direction performs much better than the ones in x, y direction. In general, the designed LQR controller shows excellent altitude control performance, but the x, y position control features steady-state error and delay.

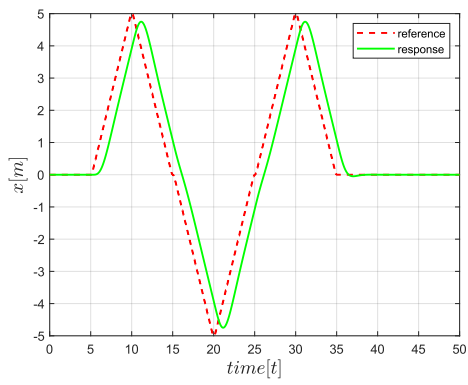


(a) Three-dimensional view

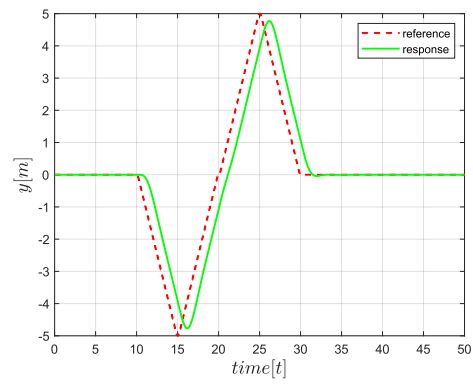


(b) $X - Y$ view

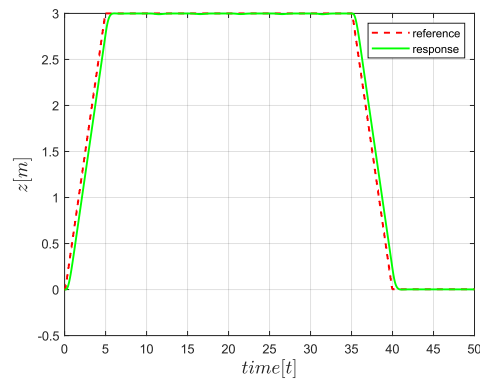
Figure Square pattern trajectory tracking performance



(a) Response and reference signal in x

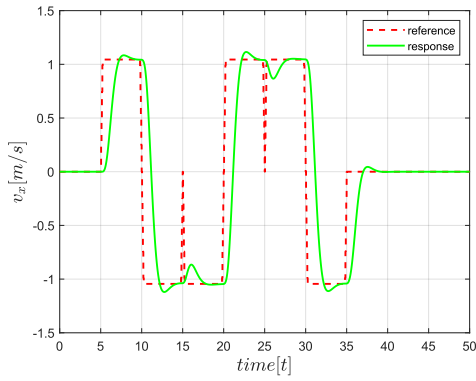


(b) Response and reference signal in y

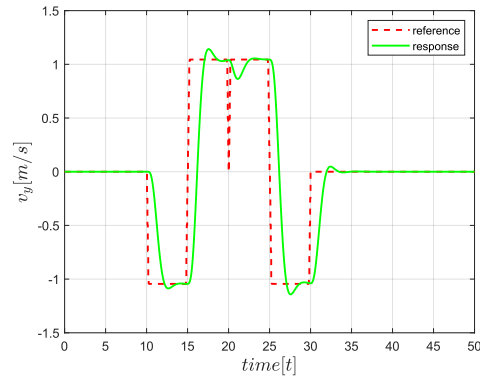


(c) Response and reference signal in z

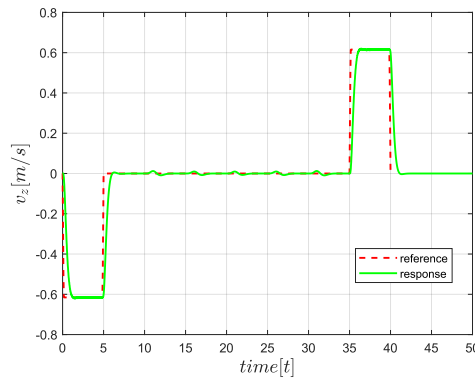
Figure x, y, z time response



(a) v_x reference and response



(b) v_y reference and response



(c) v_z reference and response

Chapter 4

Sensors and GNSS module

4.1 Inertial Measurement Unit

An inertial measurement unit or IMU for short is an electronic device that typically consists of accelerometers, gyroscopes, and magnetometers (optimal) which measure linear acceleration, angular velocity, and magnetic field strength, respectively. IMUs are often incorporated into Navigation Systems which utilize the raw measurements to calculate attitude, linear velocity, and position of an aircraft relative to a global reference frame. Besides navigational purposes, IMUs

also serve as orientation sensors in many consumers product such as smartphones and game equipment. No matter what, IMUs are essential tools for measuring the attitude of an object in space. To develop a flight controller for the quadcopter, the IMU is indispensable, it provides the microcontroller with the quadcopter's attitude, with which the control algorithm can generate the desired command to track an attitude signal. In this section, the method for computing attitude angles from the gyroscope and accelerometer respectively is expounded. Then, the complementary filter used for reducing noises and sensor fusion is introduced and implemented to get more accurate attitude angles.

4.2.1 Gyroscope

The gyroscope is an electronic sensor that measures the angular rate of the attached object, or how fast the object is turning relative to its body coordinate frame. By integrating the angular rate, it's possible to get the attitude of the object in space.

The gyroscope of MPU-6050 detects rotation about the X , Y , Z axis and outputs the angular rate in 16 bits two's complement format. The full-scale range of the gyroscope may be digitally programmed to ± 250 , ± 500 , ± 1000 , or ± 2000 . In this project, a ± 500 full-scale range is enough considering the quadcopter's rotational dynamics. By checking the MPU-6050 datasheet, it is found that in the case of a ± 500 full-scale range, the gyroscope will output 65.5 when it is rotating at 1 degree per second. Hence the measured angular rate in degrees per second can be expressed as:

$$angular\ rate = \frac{raw\ gyro\ output}{65.5}$$

Since the gyroscope measures angular rates, in ideal conditions, the gyroscope should output zero when it's at rest, and output a constant value when it's rotating at a constant speed. But in reality, the output of the gyroscope is not zero when it's static. As is shown in Figure, the output curve floats up and down around a certain value, which is the zero-rotation error of the gyroscope. The zero-rotation error should be subtracted from the output to center the gyroscope output to zero. The gyroscope calibration is a method to compute the zero-rotation error and it is implemented by collecting 2000 measurements and subtracting the mean value from the raw gyroscope output. The calibration starts every time the code runs on the microcontroller, and it lasts a couple of seconds. During calibration, the gyroscope should be static to avoid affecting the calculation of the mean value. After the calibration process is finished, the raw gyroscope output is centered to zero and can be used to calculate the attitude angles.

As mentioned above, the MPU-6050 is used to calculate the attitude angles of the quadcopter, which are the roll pitch and yaw angles introduced in section 2.2.3. The attitude angles can be obtained by integrating the angular rate. Denoting the roll pitch and yaw angles at n^{th} moment as $r(n)$, $p(n)$, $y(n)$ respectively, and the angular rates measured by the gyroscope as g_x , g_y , and g_z , the attitude at the next moment needs to be calculated at the current moment. To calculate the attitude at $(n + 1)^{th}$ moment, just add the corresponding attitude angle change based on the attitude at n^{th} moment. The amount of change in the attitude angle can be multiplied by the angular velocity and the adopted time interval. The renewal of the attitude angles

can be expressed as:

$$\begin{cases} r(n+1) = r(n) + \frac{dr}{dt} \Delta t \\ p(n+1) = p(n) + \frac{dp}{dt} \Delta t \\ y(n+1) = y(n) + \frac{dy}{dt} \Delta t \end{cases} \quad (4-1)$$

Here, $\left[\frac{dr}{dt} \quad \frac{dp}{dt} \quad \frac{dy}{dt} \right]^T$ is the angular velocity relative to the inertial coordinate frame, which is used for attitude update. The attitude update is based on the inertial coordinate frame, while the angular rate measured by the gyroscope at n^{th} moment is relative to its own IMU coordinate frame. Hence, the angular rate measured by the gyroscope needs to be transformed to the angular velocity relative to the inertial coordinate frame before updating the attitude. According to Equation 2-7, the transformation matrix is based on current attitude and can be expressed as:

$$T = \begin{bmatrix} 1 & \frac{\sin(p)\sin(r)}{\cos(p)} & \frac{\cos(r)\sin(p)}{\cos(p)} \\ 0 & \cos(r) & -\sin(r) \\ 0 & \frac{\sin(r)}{\cos(p)} & \frac{\cos(r)}{\cos(p)} \end{bmatrix} \quad (4-2)$$

The angular velocity relative to the inertial coordinate frame can be expressed as:

$$\begin{bmatrix} \frac{dr}{dt} \\ \frac{dp}{dt} \\ \frac{dy}{dt} \end{bmatrix} = T \begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix} = \begin{bmatrix} 1 & \frac{\sin(p)\sin(r)}{\cos(p)} & \frac{\cos(r)\sin(p)}{\cos(p)} \\ 0 & \cos(r) & -\sin(r) \\ 0 & \frac{\sin(r)}{\cos(p)} & \frac{\cos(r)}{\cos(p)} \end{bmatrix} \begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix} \quad (4-3)$$

The updating of the attitude angles can be implemented by Equation 4-3 and Equation 4-1. It should be noted that Δt is the integration interval. Since the main loop frequency of the developed program is 250Hz, the integration interval is 0.004s.

4.2.2 Accelerometer

Accelerometer sensors measure the linear acceleration and the gravitational field vector along the axes of its coordinate frame. The measurement of the rotated gravitational field vector can be used to determine the accelerometer roll and pitch attitude angles. The attitude angles are dependent on the order in which the rotations are applied, which are expounded in section 2.2.3.

When the accelerometer is at rest and placed horizontally, that is when the z axis is upright, the gravitational acceleration distribution along z axis is $1g$, and along both x axis and y axis, the distributed components are both 0. At this moment, the gravitational field vector relative to the accelerometer's coordinate frame can be recorded as:

$$G = [0 \quad 0 \quad g]^T \quad (4-4)$$

When the accelerometer rotates to a certain attitude, the gravitational acceleration will produce corresponding components along the three axes of the accelerometer's coordinate frame. it's essential the coordinates of the gravitational field vector relative to the new accelerometer's coordinate frame. The rotation sequence applied to the accelerometer's coordinate frame is ZYX (yaw, then pitch and finally roll), and according to section 2.2.3, the rotation matrix from the accelerometer's coordinate frame to the inertial coordinate frame can be expressed as:

$$R = \begin{bmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{bmatrix} \quad (4-5)$$

Denoting the coordinate of gravitational field vector relative to the accelerometer's coordinate frame as $[a_x \ a_y \ a_z]^T$, then:

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = R^T \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} = \begin{bmatrix} -\sin(\theta) g \\ \sin(\phi) \cos(\theta) g \\ \cos(\phi) \cos(\theta) g \end{bmatrix} \quad (4-6)$$

By solving Equation 4-6, the roll and pitch attitude angles are obtained and can be expressed as:

$$\begin{cases} \phi = \arctan\left(\frac{a_y}{a_x}\right) \\ \theta = -\arctan\left(\frac{a_x}{\sqrt{a_y^2 + a_z^2}}\right) \end{cases} \quad (4-7)$$

Since the gravitational acceleration along the z axis remains constant when the accelerometer rotates around the z axis, the yaw angle cannot be evaluated by the accelerometer. Another problem with the evaluated attitude angles is that the equations for the roll and pitch angles have mathematical instabilities when x or y axis happens to become aligned with the gravity and points upwards or downwards. In this project, the roll and pitch angles estimated by the accelerometer are limited considering the rotational dynamics of the quadcopter, hence this instability problem can be avoided.

The accelerometer of MPU-6050 outputs the measured accelerations in 16 bits two's complement format. And the full-scale range can be digitally programmed to $\pm 2g$, $\pm 4g$, $\pm 8g$, or $\pm 16g$. In this project, the full-scale range is set to $\pm 8g$ and by checking the MPU-6050 datasheet, an output of 4096 from the accelerometer will correspond to $1g$.

4.2.3 Complementary filter and sensor fusion

Although both the gyroscope and the accelerometer can be used individually to measure the roll and pitch angles of the quadcopter, neither of the measured angles can be directly used by the flight controller due to the characteristics of the sensors. Hence, it's necessary to integrate the results of the two sensor measurements to obtain more accurate and reliable results.

The gyroscope gives a good indication of the attitude angles in dynamic conditions. Due to the non-ideal offset compensation of the gyroscope, even with calibration, the raw outputs of the gyroscope are still not zero at rest. This makes the angles estimated by gyroscope drift. When the quadcopter is in flight, the vibrations generated by the quadcopter will speed up the drift, which makes the attitude angles computed by the gyroscope different from the real attitude angles. The accelerometer gives a good indication of the tilt angles in static conditions. The accelerometer doesn't have a drift problem, but the always presented vibrations and small accelerations during flight make the attitude angles unreliable and useless to the flight controller. Only the average of the angles computed by the accelerometer is usable to the flight controller.

To overcome these problems, a complementary filter is utilized in the project to combine the two measurements obtained from the gyroscope and the accelerometer. The complementary filter is a

convenient way to combine measurements from an accelerometer and a gyroscope into a better angle estimation than either could provide on its own. The idea behind the complementary filter is to take slow-moving signals from the accelerometer and fast-moving signals from the gyroscope and combine them.

The basic structure of the complementary filter is shown in Figure 4.4, where x_1 and x_2 are noisy measurements of the same signal that mainly contain low-frequency noise and high-frequency noise, respectively. The transfer function $G(s)$ can be made into a low-pass filter to filter out the high-frequency noise in x_2 and $[1 - G(s)]$ can be made into a high-pass filter that can filter out the low-frequency noise in x_1 . The output of the complementary filter can be expressed as:

$$x = x_1[1 - G(s)] + x_2G(s) \quad (4-8)$$

And the first-order low-pass filter (LPF) and first-order high-pass filter (HPF) can be expressed as:

$$LPF = G(s) = \frac{1}{\tau s + 1}$$

$$HPF = 1 - G(s) = \frac{\tau s}{\tau s + 1} \quad (4-9)$$

Where $s = \sigma + j\omega$ is the complex variable in the frequency domain used for Laplace transform, and τ is the time constant that determines the filter cut-off frequency.

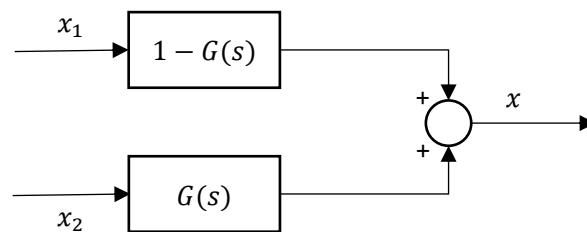


Figure 4.4 Basic complementary filter

In Figure 4.5, the bode diagrams of the first-order low-pass filter and high-pass filter are shown. The low-pass filter passes through signals with frequencies lower than the cut-off frequency and attenuates signals with frequencies higher than the cut-off frequency. Hence, the low-pass filter can be used to process the signals coming from the accelerometer which contain high-frequency noise (such as the accelerometer in the case of vibration). The high-pass filter behaves exactly in the opposite way. It allows short-duration signals to pass through while filtering out signals that are steady over time. This can be used to process signals coming from the gyroscope to cancel out the drift.

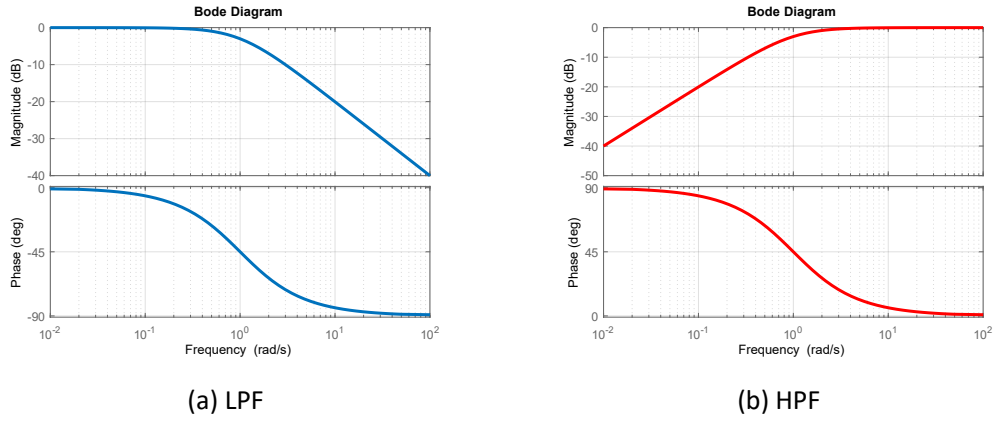


Figure 4.5 Bode diagram of LPF and HPF ($\tau = 1$)

To implement the complementary filter to the microcontroller, the complementary filter represented in transfer function format must be transformed into a difference equation format. A straightforward method to implement the complementary filter to a microcontroller is to make the complementary filter in the form like this:

$$x = \alpha \cdot x_1 + (1 - \alpha) \cdot x_2 \quad (4-10)$$

Where x_1 and x_2 represents signals coming from the gyroscope and accelerometer, the parameter α can be calculated by:

$$\alpha = \frac{\tau}{\tau + \Delta t} \quad (4-11)$$

Where $\alpha \in \mathbb{R}$ and $0 < \alpha < 1$, τ is the time constant of the low-pass filter and the high-pass filter, Δt is the sampling interval. Since the accelerometer doesn't estimate yaw angle, the complementary can only be used to integrate roll and pitch angles estimated from different sensors. The attitude roll and pitch angles estimated by the IMU can be expressed as:

$$\begin{cases} roll = \alpha \cdot roll_{gyro} + (1 - \alpha) \cdot roll_{acc} \\ pitch = \alpha \cdot pitch_{gyro} + (1 - \alpha) \cdot pitch_{acc} \end{cases} \quad (4-12)$$

And the block diagram of the complement filter utilized in this project is shown in the Figure below:

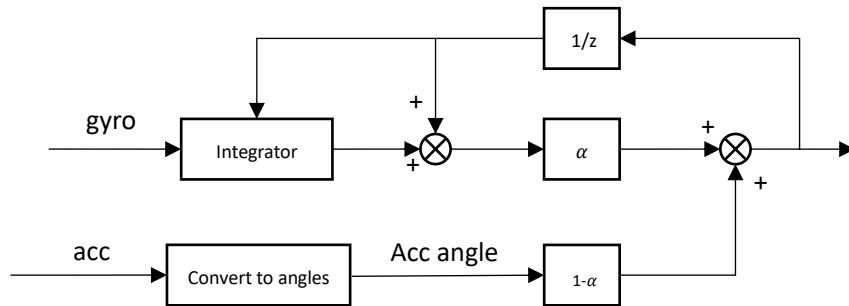


Figure Block diagram of the discrete complementary filter

4.3 Compass

Implementing a compass module to the quadcopter allows the quadcopter to know its heading angle or direction in space. With the direction information, it's possible

4.3.1 Earth's magnetic field

The Earth's magnetic field can be approximated with the dipole model shown in Figure 4.6. As the figure shows, the Earth's magnetic field points down toward the north in the northern hemisphere, is horizontal and points toward the north at the equator and points up toward the north in the southern hemisphere. In all cases, the Earth's magnetic field has a component parallel to the surface and points toward the magnetic north. This horizontal component can be measured by the magnetic sensor and the heading angle can be calculated by trigonometry. This is the basis for using the electronic compass to estimate the heading angle.

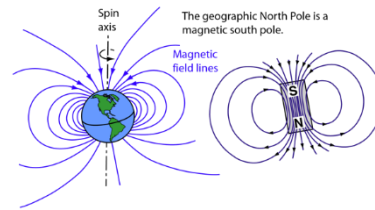


Figure 4.6 Earth's magnetic field

4.3.2 Heading angle calculation

When the compass is at a leveled position (horizontal to the Earth's surface), then the roll and pitch angles would be zero and the heading angle can be determined as shown in Figure 4.7. The local Earth's magnetic field H has a fixed component H_h on the horizontal plane pointing to the Earth's magnetic north. Denoting the horizontal components measured by the magnetic sensor as X_h and Y_h . Then the heading angle is calculated as:

$$heading = \arctan\left(\frac{Y_h}{X_h}\right) \quad (4-13)$$

To account for the tangent function being valid over 180° and not allowing the $Y_h = 0$ division calculation, the following equations can be used:

$$heading = \begin{cases} 180 - \arctan\left(\frac{Y_h}{X_h}\right), & \text{for}(X_h < 0) \\ -\arctan\left(\frac{Y_h}{X_h}\right), & \text{for}(X_h > 0, Y_h < 0) \\ 360 - \arctan\left(\frac{Y_h}{X_h}\right), & \text{for}(X_h > 0, Y_h > 0) \\ 90, & \text{for}(X_h = 0, Y_h < 0) \\ 270, & \text{for}(X_h = 0, Y_h > 0) \end{cases} \quad (4-14)$$

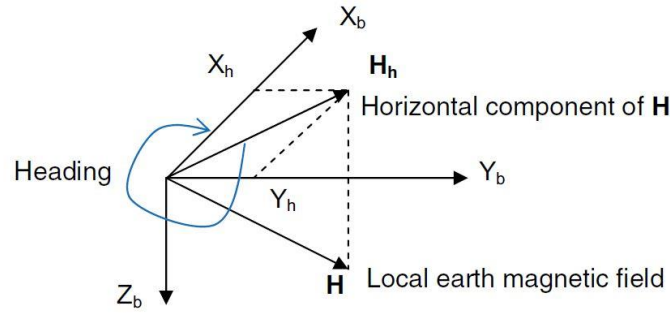


Figure4.7 Heading calculation

As is shown in Figure 4.7, when X_b (X axis of the compass coordinate frame) is parallel to H_h , which is the horizontal component pointing to the magnetic north, then the heading angle equals zero. Rotating the compass clockwise on the horizontal plane, the heading angle increases. After a full round 360° rotation, we can get a centered circle if plotting X_h and Y_h values coming from the magnetic sensor measurements. It should be noted that only the horizontal components X_h and Y_h are used when computing the heading angle, the vertical component Z_h is ignored.

In this project, the compass module is mounted on the quadcopter to measure the heading angle. Hence, in most cases the compass is not confined to a level plane (Figure 4.8), then the roll and pitch angles are not equal to zero, the magnetic sensor measurements X and Y are not the horizontal components and can't be directly used in Equation 4-14 for computing the heading angle. The conversion between the magnetic sensor measurements X , Y , Z , and the horizontal components X_h and Y_h is required. As mentioned in section 2.2.3 above, the rotation to be performed is first rolled then pitched, and the following relationship can be obtained:

$$\begin{bmatrix} X_h \\ Y_h \\ Z_h \end{bmatrix} = R(\theta) \cdot R(\phi) \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$\begin{bmatrix} X_h \\ Y_h \\ Z_h \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\phi) \sin(\theta) & \cos(\phi) \sin(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ -\sin(\theta) & \cos(\theta) \sin(\phi) & \cos(\phi) \cos(\theta) \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (4-15)$$

Where the roll and pitch angles can be estimated by the MPU-6050 as mentioned in section 4.2. the relationship between the horizontal components and the magnetic sensor measurements can be expressed as:

$$\begin{cases} X_h = X \cdot \cos(\theta) + Y \cdot \sin(\phi) \cos(\theta) + Z \cdot \cos(\phi) \sin(\theta) \\ Y_h = Y \cdot \cos(\phi) - Z \cdot \sin(\phi) \end{cases} \quad (4-16)$$

After the horizontal components are determined by Equation 4-16, then the heading angle can be

computed using Equation 4-14.

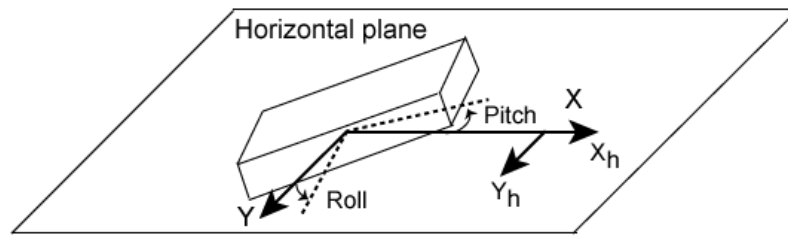


Figure 4.8 Tilt compensation of the compass

As is shown in Figure 4.9 below, the true north, also called the geographic north, is the intersection of the Earth's rotation axis and the Earth's surface, which is not at the same geographical location as the magnetic north (They are about 11.5° rotation from each other). The declination angle is used to describe the difference between them. By definition, the angle on the horizontal plane between magnetic north and geographic north is the declination angle. It's positive when the magnetic north is to the east of the geographic north, and negative when the magnetic north is to the west. The declination angle varies a lot depending on the position on the Earth's surface, and at some locations, the declination angle will even be as large as 25° . For a given location the declination angle can be found by using a geomagnetic declination map or by checking some official website. Equation 4-14 computes the heading angle relative to the magnetic north. To account for the difference between the magnetic north and the geographic north and get a more precise heading angle, the declination angle should be added or subtracted from the heading angle computed by Equation 4-14. The declination used in this project is $+2.85^\circ$.

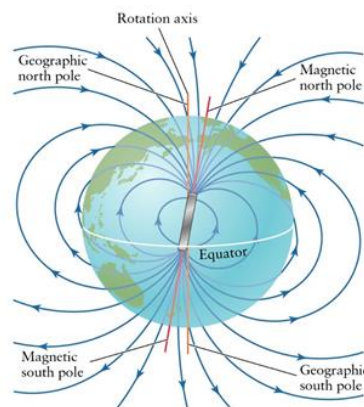


Figure 4.9 Magnetic north and Geographic north

Chapter 5

Hardware components

In this chapter, the hardware components requested for the real quadcopter platform and their specifications are expounded. The quadcopter platform consists of several different modules such as the sensor module, the flight controller, the actuator module, the radio controller module, the radio communication module, and the data storage module. The sensor module consists of an IMU, a compass, a GNSS module, and an altimeter that can be utilized to measure the crucial information of the quadcopter during flight, such as position, attitude, altitude, linear velocity, and angular velocity. The flight controller is the core of the quadcopter platform, which takes the sensor measurements and command signals as input and computes and generates the desired control signals. The actuator module consists of four DC brushless motors and four electronic speed controllers, and it takes the control signals from the flight controller as input and generates the corresponding control actions. The radio controller module is used by the pilot to send control commands to the quadcopter wirelessly. The radio communication module is used to monitor the status of the quadcopter while flying. And the data storage module is used to save the quadcopter's data and log information during flight, which can be used for subsequent analysis. The hardware components used to design the quadcopter platform include:

1. STM32 development board
2. MPU-6050 gyroscope/accelerometer
3. QMC5883L compass module
4. FlySky FS-i6X transmitter and FS-IA6B receiver
5. Electronic speed controller
6. SD card adapter
7. APC220 radio communication module
- 8.

5.1 STM32 development board

The STM32 development board used in this project is the STM-32F103C8T6 ([Figure 5.1](#)), which is also known as the blue pill board. This development board has very low power consumption and very strong computing ability, also its hardware is open source. Hence, it's ideal as the core of a flight control system. The microcontroller chip of this board is based on an ARM 32-bit Cortex-M3

CPU core, and the maximum CPU frequency is 72MHz, which makes this board very computational compared to an Arduino board (the CPU only runs at 8MHz or 16 MHz). Since the MCU works with 3.3V, the board also houses a 5V to 3.3V voltage regulator IC to power the MCU. Even though the MCU operates at 3.3V, most of its GPIO pins are 5V tolerant. There are also two onboard LEDs, one (red color) is used for power indication, and the other (green color) is connected to the GPIO pin PC13. It also has two header pins which can be used to toggle the MCU boot mode between programming mode and operating mode. In addition to the above features, this board also features 20 Kbytes of SRAM, up to 128Kbytes of Flash memory, and up to 9 communication interfaces. More detailed information about its specifications is shown in [Table 5.1](#) below.

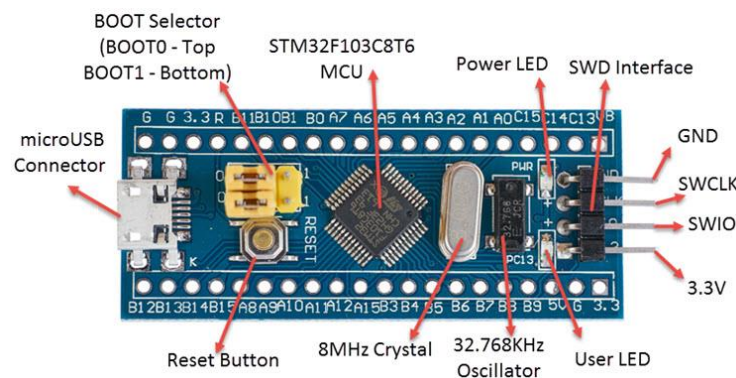


Figure 5.1 STM-32F103C8T6

As mentioned above, two header pins (boot 0 and boot 1) are used to select the memory from which the MCU boots. When both boot 0 and boot 1 pins are set to low (0), then the internal flash memory acts as the main boot space and when boot 0 is set to high (1) and boot 1 is set to low (0), the system memory acts as the main boot space.

To upload code to the flash memory of the MCU, the system memory must be selected as the main boot space. By booting into the system memory, the flash memory will be reprogrammable. This is called the programming mode of this board. In programming mode, every reset or power-off will clear the code uploaded to the board. Once the program is uploaded to the flash memory, switch back the boot 0 to low (0), so that from the next reset or power-up, the MCU will boot from the flash memory, and the code uploaded will be preserved. This is called the operating mode of this board. It should be noted that in both cases, the boot 1 pin is set to low, and only the boot 0 pin is toggled between low (flash memory) and high (system memory).

STM32F103C8T6

Operating Voltage	2.7V to 3.6V
CPU Frequency	72 MHz
Flash Memory	128 KB
RAM	20 KB
Number of GPIO pins	37
Number of PWM pins	12
Analog input pins	10 (12-bit)
USART peripherals	3
I2C peripherals	2
SPI peripherals	2
Can 2.0 peripheral	1
Timers	3 (16-bit), 1 (PWM)

Table 5.1 STM32F103C8T6 specifications

In this project, we use the Arduino IDE to program this STM32 board. The code written on the Arduino IDE can be uploaded to the STM32 board by two methods. One is through the interface headers on the board, for this, the st-link debugger is required. The other one is through UART, and for this, a USB to TTL module is required. In this project, the FT232RL FTDI adapter ([Figure 5.2](#)) is used as the USB to TTL module, with which the Arduino code can be uploaded to the board with the serial method.

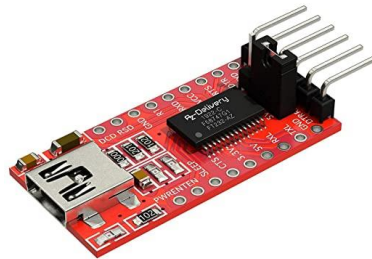


Figure 5.2 FT232RL FTDI adapter

5.2 MPU-6050 gyroscope/accelerometer

In this section, the IMU used for the estimation of the quadcopter's attitude and its specifications are introduced. The IMU used in this project is MPU-6050 ([Figure 5.3](#)), which is a 6-axis Motion Tracking device that combines a 3-axis gyroscope, a 3-axis accelerometer, and a Digital Motion Processor (DMP) all in a very small package. This IMU features a small size, low power consumption requirements, high accuracy, and high reputability. Also, it's pretty simple to interface with microcontrollers and other sensors such as magnetometers. More detailed information about its specifications can be found in [Table 5.3](#) below.

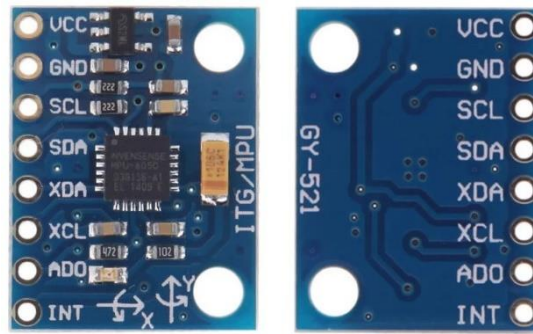


Figure 5.3 GY-521 IMU module

Pin	Function
VCC	Power supply
GND	Ground terminal
SCL	I2C serial clock
SDA	I2C serial data
XDA	Serial data for connection to external sensors
XCL	Serial clock for connection to external sensors
ADO	I2C LSB slave address (AD0)
INT	Interrupt digital output

Table 5.2 Pins description of GY-521 IMU module

Gyroscope Features	Accelerometer Features
3-Axis sensor with a user-programmable full-scale range of ± 250 , ± 500 , ± 1000 , and ± 2000 degrees per second	3-Axis sensor with a user-programmable full-scale range of $\pm 2g$, $\pm 4g$, $\pm 8g$, and $\pm 16g$
16-bit ADCs	16-bit ADCs
Digitally programmable low-pass filter	Orientation detection and signaling
Factory calibrated sensitivity scale factor	User-programmable interrupts
Up to 8000Hz output data rate	Up to 1000Hz output data rate
Electrical and other common specifications	
9-Axis motion fusion by the on-chip Digital Motion Processor (DMP)	
VDD supply voltage range of 2.375V-3.46V	
1024-byte FIFO buffer reduces power consumption	
Digital-output temperature sensor	
400kHz Fast Mode I2C for communication	
Internal clock oscillator	

Table 5.3 MPU-6050 specifications

5.3 Compass module

The compass module used in this project is QMC5883L ([Figure 5.4](#)), which is a multi-chip three-axis magnetic sensor. This sensor is based on state-of-the-art magneto-resistive technology and has the advantage of low noise, high accuracy, low power consumption, offset cancellation, and temperature compensation. The QMC5883L compass module can achieve 1° to 2° compass heading accuracy. More detailed information about its specifications can be found in ([Table 5.4](#)) below.



Figure 5.4 QMC5883L compass module

QMC5883L	
Supply voltage	2.16V to 3.6V
Full-scale range	± 8 Gauss
Resolution (ADC)	16 bits
Gauss resolution	$\pm 2\text{mG}$ to $\pm 8\text{G}$
Output data rate	10Hz, 50Hz, 100Hz, 200Hz
Interface	I2C
I2C rates (kHz)	100, 400
Operating temperature	-40°C to 85°C

Table 5.4 QMC5883L specifications

The output of the QMC5883L is in 16 bits two's complement format. In this project, the compass is set to continuous mode, the output data rate is set to 200Hz, and the full-scale range is set to ± 8 Gauss. By checking the datasheet of QMC5883L, the sensitivity is 3000 LSB/G when the full-scale range is set to ± 8 Gauss.

5.4 Transmitter and receiver

The transmitter and receiver make up the radio control system of the quadcopter. The transmitter is an electronic device that uses radio signals to transmit commands wirelessly over to the radio receiver that is connected to the quadcopter flight controller. The transmitter reads the stick inputs (control commands) and sends them through the air to the receiver in near real-time. The control commands are sent by channels. Each channel corresponds to an individual action being sent to the quadcopter. The basic channels are given as follows:

1. Roll. Moves the quadcopter left or right in the space.
2. Pitch. Tilts the quadcopter forward or backward.
3. Yaw. Rotates the quadcopter clockwise or counterclockwise, allowing the quadcopter to change its direction in space.
4. Throttle. Controls the amount of power sent to the quadcopter, which controls the altitude and also the speed of the quadcopter.

In addition to the above four basic channels, the transmitter also has some extra channels for any auxiliary switches it may have. These auxiliary switches can be used to arm the quadcopter or switch the quadcopter's flight mode.

The radio receiver is the device capable of receiving commands from the radio transmitter, once the receiver gets these commands it sends them to the flight controller and the flight controller converts them into specific actions for controlling the quadcopter. A transmitter uses many frequencies like 27MHz, 72MHz, 433MHz, 900MHz, 1.3GHz, and 2.4GHz. Most transmitters work on a 2.4GHz radio frequency. A receiver's frequency must be compatible with the transmitter one to establish a communication. For instance, a 2.4GHz transmitter can only work with the 2.4GHz radio receiver.

In this project, the FS-i6X transmitter with FS-iA6B receiver ([Figure 5.5](#)) is utilized as the radio control module of the quadcopter. The FS-i6X transmitter features 6 channels and works in the frequency of 2.4GHz with PPM output and the FS-iA6B is the matched 6 channels receiver that can receive PPM signal. The specifications of the transmitter and receiver are summarized and listed in [Table 5.5](#) and [Table 5.6](#), respectively.



Figure 5.5 Flysky FS-i6X transmitter with IA6B Receiver

FS-i6X transmitter	
Channels	6-10 (Default 6)
RF range	2.408-2.475GHz
RF power	Less than 20dBm
Bandwidth	500KHz
Range	500-1500m (in the air)
System	AFHDS 2A / AFDHS
Stick resolution	4096
Low voltage warning	Less than 4.2V
Power supply	6V DC 1.5AA*4
Size	174×89×190mm
Weight	392 g
Temperature range	-10°C to +60°C

Table 5.5 Specifications of FS-i6X transmitter

IA6B receiver	
Channels	6
RF range	2.408-2.475GHz
RF power	Less than 20dBm
RF receiver sensitivity	- 105dBm
Range	500-1500m (in the air)
Power supply	4.0-8.4V
Temperature range	-10°C to +60°C
Size	47×26.2×15mm
Weight	10 g

Table 5.6 Specifications of IA6B receiver

4.5 Electronic Speed Controller

The electronic speed controller or ESC is a fundamental component of a quadcopter, it takes the control signal (PWM signal) from the flight controller and controls the rotational speed of the motor. The ESCs take the control signal from the controller as input, then raise or lower the voltage to the motor according to the PWM duty cycle, thus changing the speed of the motor. The ESCs for quadcopters are typically rated by the maximum current. ESCs that can handle a large current draw will usually be larger and heavier, which may be an important consideration for small-size quadcopters. ESCs also have a refresh rate in Hertz, which is how many times a second the motor speed can be regulated. The ESCs for quadcopters and other multirotor drones may have higher refresh rates, as their stability and maneuverability depend entirely on the balance of motor

speeds, and as such, they require fine control over the motor speeds. In this project, four XXD HW30A Brushless Motor ESCs ([Figure 5.6](#)) are utilized to control and regulate the motors of the quadcopter. More detailed information about the ESC's specifications is available in [Table 5.7](#) below.

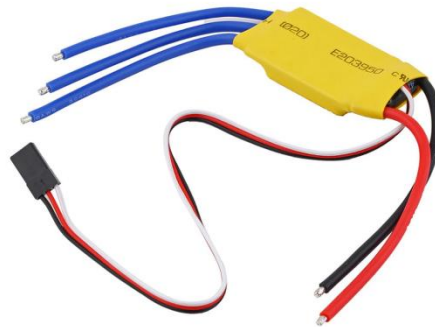


Figure 5.6 Electronic speed controller

HW30A Brushless Motor ESC	
BEC power	1.5A/5V
Cut off voltage	4V
Current	30A
Size	57mm×25mm×8mm
Net weight	27g

Table 5.7 ESC specifications

4.5.1 PWM generation

As mentioned above, the ESC takes the PWM signal generated by the flight controller to control and regulator the speed of the motor. This section describes how to generate the desired PWM signals using the STM32 board.

The STM32 timer can be used as a time-based generator to generate a PWM signal. The timer unit includes a counter register (CNT), a pre-scaler register (PSC), an auto-reload register (ARR), and a repetition counter register (RCR). The timer can generate a PWM signal in edge-aligned mode or center-aligned mode depending on the configuration of the timer register. When the STM32 timer module is set to the PWM generation mode, the counter register (CNT) gets the clock signal from the internal clock and counts up to the auto-reload register (ARR) value, then the output channel pin is driven high. And it remains until the counter register (CNT) value reaches the compare register (CCR_x) value, when the counter register (CNT) value is greater than the compare register

(CCR_x) value, the output channel pin is driven low. And it remains until the counter register (CNT) counts up to the auto-reload register (ARR) value, and so on. This is the edge-aligned PWM generation mode in an up-counting configuration. The frequency of the resulting PWM signal is determined by the internal clock, the pre-scalar (PSC) value, and the auto-reload register (ARR) value. And the duty cycle is determined by the compare register (CCR_x) value and the auto-reload register (ARR) value. The formula below can be used for calculating the PWM frequency for the output:

$$F_{PWM} = \frac{F_{CLK}}{(ARR+1) \times (PSC+1)} \quad (5-1)$$

Where F_{CLK} is the internal clock frequency. The PWM duty cycle percentage is controlled by changing the value of the compare register (CCPx), and the duty cycle can be expressed as:

$$DutyCycle_{PWM}[\%] = \frac{CCR_x}{ARR} [\%] \quad (5-2)$$

The following diagram shows how the auto-reload register (ARR) value affects the frequency (period) of the PMW signal, how the compare register (CCPx) value affects the corresponding PWM signal's duty cycle and illustrates the whole process of PWM signal generation in the edge-aligned up-counting configuration.

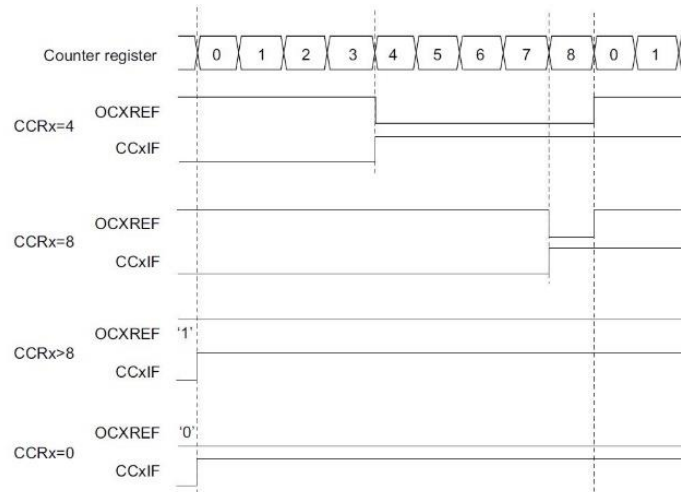


Figure 5.7 Edge-aligned PWM waveforms (ARR = 8)

One of the most important features of a PWM signal is the resolution, which is the number of discrete duty cycle levels that the PWM signal can get. The resolution determines how many steps the duty cycle can take until it reaches the maximum value. Hence, the PWM signal resolution can determine how fine the duty cycle can be changed to obtain a certain percentage. This can be extremely important for controlling the quadcopter's motors. The STM32 PWM resolution formula can be expressed as:

$$Resolution_{PWM} [Bits] = \frac{\log \left(\frac{F_{CLK}}{F_{PWM}} \right)}{\log (2)} [Bits] \quad (5-3)$$

$$Resolution_{PWM} [Bits] = \frac{\log (ARR+1)}{\log (2)} [Bits] \quad (5-4)$$

[Equation 5-3](#) can be used to calculate the resolution of the PWM signal given a specific frequency or the opposite. [Equation 5-4](#) describes the relationship between the auto-reload register (ARR)

value and the PWM signal resolution, it can be used to adjust the auto-reload register (ARR) value.

4.6 SD card and SD card adapter

To collect data on the quadcopter during flight such as position, attitude, and velocity for subsequent analysis, a data storage module is mandatory. The SD card is one of the most practical one among storage devices, hence, it is used in this project. With the corresponding SD card adapter, the flight controller can communicate with the SD card and write or read data on it. The SD card adapter interfaces with the flight controller in the Serial Peripheral Interface (SPI) protocol and the communication between them can be easily implemented with the Arduino SD library. The pinout of the SD card adapter is shown in (Figure) below. MISO is the Master In Slave Out line, this line transfers data from the SD card adapter (the slave device) to the flight controller (master device). MOSI is the Master Out Slave In line, this line transfers data from the flight controller to the SD card adapter. SCK is the clock line that synchronizes the data transmission between the master and slave. CS is the chip select line, which is used by the flight controller (master device) to enable and disable a specific device on the SPI bus.

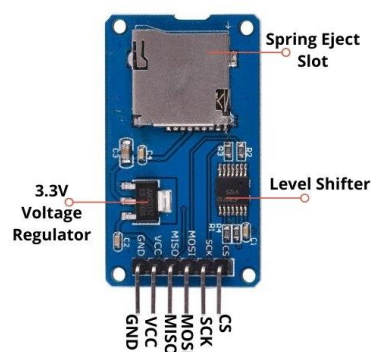


Figure SD card adapter

The SD card used in this project is shown in (Figure), the storage capacity of which is 2GB, which is more than enough for collecting data during flight.



Figure SD card

SD card adapter	
Operating voltage	4.5V to 5.5V DC
Current requirement	0.2mA to 200mA
3.3V on-board voltage regulator Supports FAT file system Supports micro SD up to 2GB Supports Micro SDHC up to 32GB	

Table Specifications of SD card adapter

4.7 Wireless communication module

To keep track of the quadcopter and have real-time information on key components of the quadcopter, such as position, altitude, and battery levels, the wireless communication module is required, which allows real-time communication between the quadcopter and the PC in the form of a data stream. In this project, the wireless communication module is utilized to send signals from the PC to the quadcopter to control it and also receive signals back from the quadcopter for monitoring important information about it. The wireless communication module used in this project is the APC220 radio communication module. Two APC220 modules are required for the data transmission, one is wired to the flight controller and the other is connected to the PC through a USB to TTL converter. More detailed information about the specifications of the APC220 module is in (Table) below.



Figure APC220 radio communication module and USB to TTL converter

APC220 radio communication module	
Working frequency	431 MHz to 478 MHz
Power supply	3.3V to 5.5V
Current	<25-35mA
Working temperature	-20°C to 70°C
Range	1200m line of sight (1200 bps)
Interface	UART/TTL
Baud rate	1200-19200 bps
Baud rate (air)	1200-19200 bps
Receiver buffer	256 bytes
Size	37mm × 17mm × 6.6mm
Weight	30g

Table APC220 specifications

4.8 Schematic of the flight controller

Appendix A

Experimental measurements

A.1 Propeller's thrust and torque measurements

To find the mathematical relationship between the PWM signal provided to the ESC from the flight controller and the thrust and torque generated by the motor, the propeller's thrust and torque tests are mandatory. This allows the conversion between the commands generated by the flight controller expressed as the PWM signals and the physical input vector u defined in chapter 2.

$$u = \begin{bmatrix} F_t \\ M_x \\ M_y \\ M_z \end{bmatrix} \quad (\text{A-1})$$

The measurements of the torque and thrust generated by the motor are implemented on the RCBenchmark Series 1580 test stand (Figure A.1), on which it is possible to measure up to 5kgf of thrust and 2 Nm of torque as well as voltage, current, power, motor rotation speed, vibration, and efficiency. The test stand has three measurement sensors, one of which is for measuring the thrust and the left two are for measuring the torque. The board on the test stand provides the PWM signal to the motor ESC and allows connecting the stand to a PC with a USB cable. Using the support software of this test stand, it's possible to manually control the pulse length of the PWM signal and collect the test data. To collect a sufficient number of samples, the software is set to continuous sample mode and the PWM pulse length is manually increased from 1000us to 1650us.



Figure A.1 RCBenchmark Series 1580 test stand



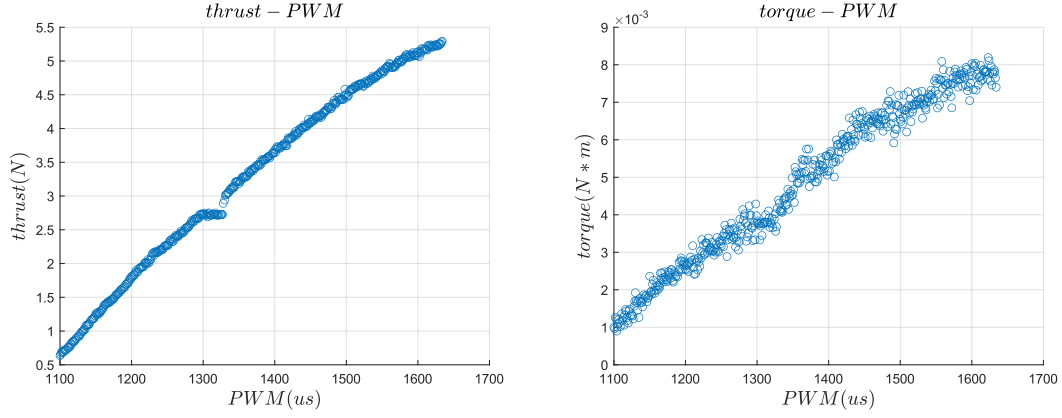
(a)



(b)

Figure A.2 Pictures during the experiment

The behavior of one motor is shown in Figure A.3 below, and the experimental data shows that the torque-PWM and thrust-PWM characteristics of the motor show a nearly linear behavior.



(a) Thrust behavior

(b) Torque behavior

Figure A.3 Behavior of single motor

Linear approximation of the torque-PWM and thrust-PWM characteristics can be derived using the Least Square method. The obtained data can be fitted using first-order functions, Take the measurement error e into consideration, the model equation can be suitably written in the following format:

$$\begin{aligned} F &= n_F \cdot PWM + q_F + e_F \\ T &= n_T \cdot PWM + q_T + e_T \end{aligned} \quad (A-2)$$

Where the gain n_F , n_T and the offset q_F , q_T are unknown parameters to be estimated. By considering the N measurements of the thrust collected in the experiment, the following system of linear equations is derived:

$$\begin{aligned} F_1 &= PWM_1 \cdot n_F + q_F + e_{F1} \\ F_2 &= PWM_2 \cdot n_F + q_F + e_{F2} \\ &\vdots \\ F_N &= PWM_N \cdot n_F + q_F + e_{FN} \end{aligned} \quad (A-3)$$

The previous equations can be written in matrix form:

$$\begin{bmatrix} F_1 \\ F_2 \\ \vdots \\ F_N \end{bmatrix} = \begin{bmatrix} PWM_1 & 1 \\ PWM_2 & 1 \\ \vdots & \vdots \\ PWM_N & 1 \end{bmatrix} \cdot \begin{bmatrix} n_F \\ q_F \end{bmatrix} + \begin{bmatrix} e_{F1} \\ e_{F2} \\ \vdots \\ e_{FN} \end{bmatrix} \quad (A-4)$$

The estimation problem is recast in the standard Least Square format:

$$y = \phi \cdot \vartheta + e \quad (A-5)$$

Where $y \in \mathbb{R}^N$, $\phi \in \mathbb{R}^{N \times 2}$, $\vartheta \in \mathbb{R}^2$ and $e \in \mathbb{R}^N$. Since the unknown is the vector ϑ , the problem to be solved is overdetermined, because the number of unknowns is smaller than the number of measurement equations taken into account and then, in general, the system of equations does not admit any solution, since the matrix ϕ cannot be inverted. Using the least square algorithm as an estimation method, the solution could be:

$$\hat{\vartheta} = (\phi^T \cdot \phi)^T \phi^T \cdot y \quad (A-6)$$

Under MATLAB, the least-squares solution is provided by the operator “\” or the command `mldivide`, and the result is:

$$\hat{\vartheta} = \begin{bmatrix} \hat{n}_F \\ \hat{q}_F \end{bmatrix} = \begin{bmatrix} 0.0088 \\ -8.8085 \end{bmatrix} \quad (A-7)$$

As a result, the thrust F in the function of the PWM signal is:

$$F_{(PWM)} = 0.0088 \cdot PWM - 8.8085 \quad (A-8)$$

Figure A.3 (a) shows the result of overlapping the equation to the experimental thrust curve, the equation can fit the linearity between thrust and PWM signal very nicely. Using the same method, the relationship between the torque T and the PWM signal is:

$$T_{(PWM)} = 0.0000136 \cdot PWM - 0.0134 \quad (A-9)$$

And Figure A.3 (b) shows the result of overlapping this equation to the experimental torque curve.

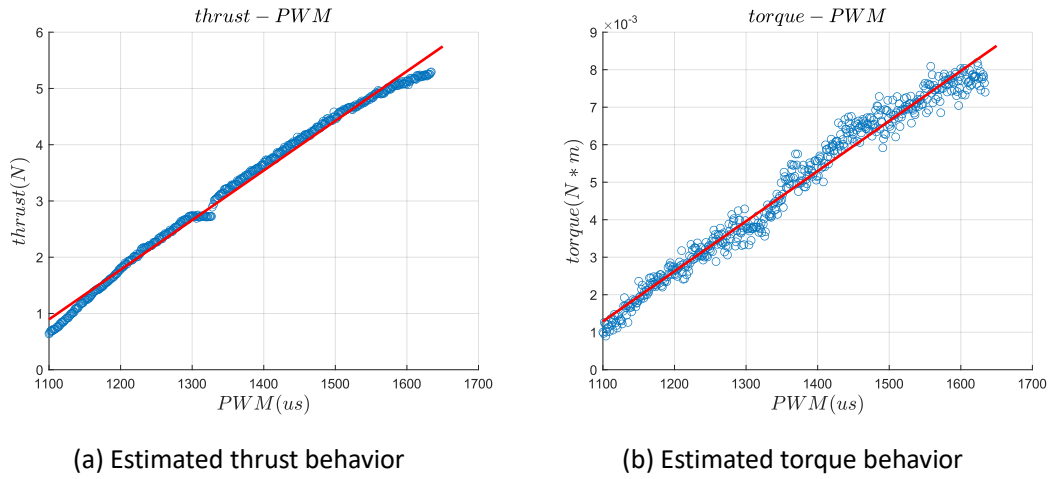


Figure A.4 Estimated single motor behavior

Lastly, the command input can be written in the function of the i^{th} motor PWM signal $PWM^{(i)}$:

$$u_{(PWM)} = \begin{bmatrix} -(F_{(PWM_1)} + F_{(PWM_2)} + F_{(PWM_3)} + F_{(PWM_4)}) \\ (F_{(PWM_1)} - F_{(PWM_2)} - F_{(PWM_3)} + F_{(PWM_4)})L\sin(\alpha) \\ (F_{(PWM_1)} + F_{(PWM_2)} - F_{(PWM_3)} - F_{(PWM_4)})L\cos(\alpha) \\ -T_{(PWM_1)} + T_{(PWM_2)} - T_{(PWM_3)} + T_{(PWM_4)} \end{bmatrix} \quad (A-10)$$

In this formula, PWM_i is the PMW signal to the i^{th} motor.

