UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

MASTER THESIS

Model Predictive Control (MPC) of Quadcopters using LPV techniques

CHANDRASHEKHAR ANAND IYER

SUPERVISED BY

Vicenç Puig

Universitat Politècnica de Catalunya
Master in Aerospace Science & Technology
July 2020

This Page Intentionally Left Blank

Model Predictive Control (MPC) of Quadrotors using LPV techniques

BY
Chandrashekhar Anand Iyer

DIPLOMA THESIS FOR DEGREE
Master in Aerospace Science and Technology

AT
Universitat Politècnica de Catalunya

SUPERVISED BY:
Vicenç Puig
Advanced control system group responsible

This Page Intentionally Left Blank

ABSTRACT

This Master Thesis proposes the study of the the Application of Model Predictive Control (MPC) of Quadrotors using LPV techniques.
First, the quadrotor control-oriented models existing in the literature is revised in order to select the one that best fits to MPC formulation including operative and physical limitations. Then, the MPC problem is formulated and discussion of which optimization solver is better adapted to the resulting optimization problem is carried out. Finally, the implementation of the MPC for the quadrotor in MATLAB with the selected solver is done and tested in simulation.
The primary step is to study control-oriented models for quadrotors that include physical/operational limitations. Secondly, the optimization algorithms that could be applied to solve the optimization problem associated to the MPC control of quadrotors are studied. Once this is done, the MPC algorithm for quadrotors is developed by selecting the most suitable model and optimization algorithm. In the end, the proposed deveoped algorithm is tested in simulation. To do all these steps, the thesis is divided into different chapters. In Chapter 1, the history of the quadcopter and the control systems are studied.
Chapter 2 begins with defining the coordinate frames used to control the quadcopter, in order to study the mathematical model and derive the quadcopter equations of motion. It is necessary that the mathematical model of the system is accurate. If it is not, then the obtained inputs from the model will not influence the real system as expected. Its response might become underdamped, overdamped or even unstable.

Chapter 3 is all about designing a suitable control strategy for the quadcopter using the LPV-MPC technique. The controller is split into two sub controllers. Position control with feedback linearization technique is responsible for governing the position variables *x, y, z* and the attitude controller controlled the angles using the LPV-MPC control strategy. The outputs generated by the position controller are *U1*, and the angles φ and θ. These angles together with *U1* and ψ, are necessary in order for the UAV to reach its *x, y, z* reference position. U1 is fed straight as an input into the open loop system. However, the angles φ and θ are fed into the attitude controller (which uses the LPV-MPC approach) as reference values. The reference angles φ and θ, together with the angle ψ from the reference generator, allows the LPV-MPC controller to find the remaining three control actions for the open loop system, which are *U2*, *U3* and *U4*. After integrating the open-loop system, the new angles φ, θ and ψ are fed back into the LPV-MPC controller together with the new Ω value at every one-fourth of the sample time. In addition, at every sample time, the open loop system sends the new *x*, *y* and *z* values back into the position controller.

Chapter 3 is about implementing the control strategy and validation of the controller. Several trajectories are tested and validated. A summary of the results is made which shows that the proposed control strategy has a very high success as all the Quadcopter's six degrees of freedom are tracked with very small errors.

Chapter 4 explains the Quadcopters impact on the Environment, society and economy. Finally, the last chapter provides the conclusions and results achieved during the Master Thesis project development. Additionally, in the Annexes Chapter, the MATLAB implementation script is attached.

This Page Intentionally Left Blank

Acknowledgement

I would like to thank my family, my girlfriend and my friends for constantly supporting me and motivating me to achieve success throughout this journey.

I am deeply grateful to my Thesis supervisor Prof. Vicenç Puig for providing me with his supervision and experience in the field of control systems. Also, his time and patience for providing me with honest remarks and feedbacks about my work.

Special thanks to my Master's coordinator Professor Ricard González Cinca for giving me the opportunity to do the Master thesis at Institut de Robòtica i Informàtica Industrial.

A big thank you to Carlos Trapiello Fernández and Mark Misin for helping me to understand the control system strategy.

This Page Intentionally Left Blank

Table of Contents

This Page Intentionally Left Blank

List of Figures

This Page Intentionally Left Blank

INTRODUCTION

Quadcopters are particular type of UAVs or flying Mini Robots or Miniature Pilotless Aircraft, drones that are rapidly growing in popularity. They have already broken rigid traditional barriers in industries that otherwise seemed quite impenetrable by similar technological innovations. Drones have managed to pierce through areas where certain industries were either stagnant or lagging behind. As we enter the decade in which drone technology enters more and more industries, it is imperative that they be reliable in terms of flight. They should be stable enough to withstand wind disturbances and must be robustly protected to reduce the risk of accidents in human populations. In addition, they need to be able to track given trajectories with very high precision if it is desired to use them in areas such as tight caves in the event of a search and rescue mission. In a nutshell, increasing work efficiency and producibility, improving accuracy and precision, decreasing workload and production costs, and resolving security issues on a big scale are some of the things drones offer industries globally. Drones possess the capability of reaching the most remote areas with the least required time, effort, and energy and manpower; one of the biggest reasons why they are being adopted Worldwide, especially in Military, Commercial, Personal and Future Technology. The UAV used in this thesis is a quadcopter - a drone with four rotors that are equally spaced from its centre.

This Master Thesis proposes the study of the the Application of Model Predictive Control (MPC) of Quadrotors using LPV techniques.
First, the quadrotor control-oriented models existing in the literature is revised in order to select the one that best fits to MPC formulation including operative and physical limitations. Then, the MPC problem is formulated and discussion of which optimization solver is better adapted to the resulting optimization problem is out. Finally, the implementation of the MPC for the quadrotor in MATLAB with the selected solver is done and tested in simulation.
The primary step is to study control-oriented models for quadrotors that includes physical/operational limitations. Secondly, the optimization algorithms that could be applied to solve the optimization problem associated to the MPC control of quadrotors are studied. Once this is done, the MPC algorithm for quadrotors is developed by selecting the most suitable model and optimization algorithm. In the end, the proposed deveoped algorithm is tested in simulation. To do all these steps, the thesis is divided into different chapters. In Chapter 1, the history of the quadcopters and their control systems are studied.
Chapter 2 begins with defining the coordinate frames used to control the quadcopter, in order to study the mathematical model and derive the quadcopter equations of motion. It is necessary that the mathematical model of the system is accurate. If it is not, then the obtained control inputs from the model will not influence the real system as expected. Its response might become underdamped, overdamped or even unstable.

Chapter 3 is all about designing a suitable control strategy for the quadcopter using the LPV-MPC technique. The controller is split into two sub controllers. Position control with feedback linearization technique is responsible for the position variables *x, y, z*

and the attitude controller controlled the angles using the LPV-MPC control strategy. The outputs generated by the position controller are *U1*, and the angles φ and θ. These angles together with *U1* and ψ, are necessary in order for the UAV to reach its *x, y, z* reference position. U1 is fed straight as an input into the open loop system. However, the angles φ and θ are fed into the attitude controller (which uses the LPV-MPC approach) as reference values. The reference angles φ and θ, together with the angle ψ from the reference generator, allows the LPV-MPC controller to find the remaining three control actions for the quadcopter, which are *U2*, *U3* and *U4*. After applying them the new angles φ, θ and ψ are fed back into the LPV-MPC controller together with the new Ω value at every one-fourth of the sample time. In addition, at every sample time, the available sensors in the UAV sends the new *x*, *y* and *z* values back into the position controller.

Chapter 3 is about implementing the control strategy and validation of the controller. The validation of the control strategy was performed in MATLAB® in the form of several simulations. First, different trajectories are given for the drone to follow. Then it is checked how well the quadcopter followed the given positions, velocities and angles. Results show that the proposed control strategy works satisfactorily as all the quadcopter's six degrees of freedom are tracked with very small errors.

Chapter 4 explains the Quadcopters impact on the Environment, society and economy. Finally, the last chapter provides the conclusions and results achieved during the Master Thesis project development. Additionally, in the Annexes, the MATLAB implementation script is attached.

Chapter 1

Quadcopters

This Chapter explains the history of the quadcopters and the current advancements. It includes the current advancements, Worldwide overview and applications of quadcopters.

1.1. History of Quadcopters

Quadcopters are Vertical Takeoff and Landing (VTOL) kind of rotorcraft with four propellers/rotors for drive. The beginning period pioneers in reality previously endeavored rotor flight utilizing quadcopter on the grounds that utilizing quadcopter appeared to be the common answer for the issue of VTOL flight.

The absolute first endeavors of a quadcopter were for the most part done in 1907 by Jacques and Louis Breguet [1] [2]. They constructed and tried Gyroplane No 1, a quadcopter. They oversaw take-off, despite the fact that the plan ends up being truly insecure and henceforth unreasonable [3]

In 1924, French designer Étienne Oehmichen flew his quadcopter a separation of 360m (1,181ft) establishing a world precedent. Around the same time, he flew a 1km (0.62miles) hover in 7m and 40s. Around a similar time George de Bothezat assembled and tried his quadcopter for the US armed force, finishing various dry runs before the program was rejected [4]

Early fashioners explored different avenues regarding quadcopters, in light of the fact that the other option, utilizing a solitary primary rotor with a tail rotor to balance the torque made by the single principle rotor appeared to be inefficient, perplexing and wasteful. The tail rotor on a solitary rotor helicopter configuration expends somewhere in the range of 10 and 15% of the motor force yet it makes no lift or forward push. Some portion of the principle rotor pivots over the fuselage, pushing down washed air against it, lessening compelling lift. Making enormous rotor cutting edges, 4 or 5 m long or significantly longer was a tremendous issue and bigger rotors right up 'til the present time are relatively a lot heavier than littler ones.

Not with standing, when PCs and great electric engines did not yet exist, the single fundamental rotor helicopter configuration had two immense points of interest [5]:

1. It was normally steady on the grounds that a weight hanging underneath a solitary connection point normally 'needs' to hang straight down thus normally adjusts undesirable inclining
2. Its motor just must be mated to just a single rotor, and could be mated to it pretty much legitimately over a short separation instead of with a belt over the few

meter/yard long expansions/arms associating the rotors of a multi-copter with its fuselage.

Early quadcopters would commonly have the motor sitting some place midway in the fuselage of the copter, driving the four rotors by means of belts or shafts. Belts and shafts anyway are substantial and significantly, subject to breakage. As the four rotors of a quadcopter are altogether marginally not quite the same as one another, a quadcopter is not normally steady, just running four rotors at a similar speed, while creating enough lift to float the copter, does not deliver stable flight [6]. Despite what might be expected, quadcopters must be continually settled. Without PCs, this implied an amazing outstanding task at hand for the pilot. Subsequently, multicopter plans were relinquished for single, or on uncommon events for exceptionally huge vehicle helicopters.

With the appearance of electric engines and particularly microelectronics and micromechanical gadgets, a couple of years prior it got conceivable to fabricate solid and effective multirotors. Present day multicopters have an electric engine mated to every rotor, sitting straightforwardly beneath or above it.
This plan has demonstrated to be enormously fruitful and most present-day VTOL automatons and side interest airplane are currently multicopters as opposed to single copters. The scaling up of this to airplane that can convey individuals has just barely started and Krossblade is a piece of this turn of events. The main quads were steered vehicles that date to the earliest reference point of revolving wing flight in the mid twentieth century: the Breguet-Richet Gyroplane, the Oehmichen No. 2, and the de Bothezat helicopter [7].

1.2. Current advancements

What we could call current quadcopter drones, which go from sub-palm-sized to just shy of two feet over, and which for the most part have locally available cameras and different highlights, first developed in the late 1990s and mid 2000s as specialist packs. The runup to omnipresence started with the 1999 arrival of the "Draganflyer quad helicopter." The first Draganflyer unit got famous among UAV scientists, and increased some open acknowledgment after its utilization in the film "Investigator Gadget." It developed into the principal valid, prepared to-fly law authorization drone.
Some current advancement includes [7]:

1. The Bell Boeing Quad Tilt Rotor idea takes the fixed quadcopter idea further by consolidating it with the tilt rotor idea for a proposed C-130 estimated military vehicle.
2. AeroQuad and ArduCopter are open-source equipment and programming ventures dependent on Arduino for the DIY development of quadcopters.
3. Nixie is a little camera-prepared automaton that can be worn as a wrist band
4. Airbus is building up a battery-fueled quadcopter to go about as a urban air taxi, from the outset with a pilot however possibly self-ruling later on.

During the mid-to-late 2000s, quads kept on developing in notoriety among specialists. In 2010, the French organization Parrot discharged their "AR.Drone", the principal economically fruitful prepared to-fly purchaser drone, and the primary ready to be

controlled exclusively by a Wi-Fi association. The automaton promptly made a sprinkle, and Parrot's replacement quads stay popular.

In April 2015, 3D mechanical technology discharged the "Solo" quadcopter, which just because permitted proficient quality video (with a connected GoPro). Not exactly a year later DJI came out with the Phantom 4, bragging "PC vision" capacity and AI to follow an individual, creature, or item on the ground without basically following a GPS track [7].

1.3. World overview of Quadcopters

Quadcopters are being incorporated in a broad range of applications now. These applications can range from wildlife conservation and protection to traffic monitoring, construction, building inspection, climate, geographical research, and so on. Being cost-efficient and immensely effective, drones are taking over mundane or dangerous works of mankind [8]. With the ability to vertically take off and land (VTOL) and hover over a point for hours to flying through small, congested spaces and be driven on gas, battery, or even solar energy, drone’s applications can range from light utility operations to rugged, long-endurance uses.

These drones can be used to employ a variety of sensor systems such as HD video cameras, IR imaging systems, radiation sensors, molecular sensors to detect hazardous substances and even ad-hoc mobile communications networks [9] that can be deployed in case of complete network failure or absence of it in an area. These sensors and devices will produce a lot of data that can be sent to the cloud storage system via a high bandwidth internet connection for storage and further processing [10].

In the 2015, Eu Aviation strategy as well as the 2016 SESAR Drones Outlook study with the Warsaw Declaration on drones, it was identified that there is a steady increase in the demand of the drone services and that this sector is going to generate significant economic growth and employment [11]. For this, there is a lot of effort being put into developing a concept of U-space, which basically will enable complex drone operations with a high degree of automation to take place in the varied environments along with the development of laws and regulations related with low-level airspace and drone flights.

1.4. New generation Quadcopters

There have been various generations of commercial drones over the past decade with every generation being smarter, more efficient, and cheaper than the previous one. This is also fuelling the use of these drones for civilian purposes with these smart drones fulfilling the immediate needs of various sectors and industries in a cost-effective manner without compromising with the efficiency of work.

The newest generation of quadcopters is considered to be:

- Commercially Suitable
- Fully Compliant to Safety & Regulatory Standards-Based Design,
- Platform & Payload Interchangeability,

- Automated Safety Modes,
- Enhanced Intelligent Piloting Models and Full Autonomy,
- Full Airspace Awareness
- Auto Action (take-off, land, mission execution).

With the drones getting smarter with the advancement of technology, they have already started to get inducted in daily life problems, like taking inventory of a store or a car factory, to using them for emergency services or surveying forests [12].

Moreover, technologies such as the batteries, drone software, sensors for detection and avoidance of obstacle, and 3D sensing for the drones are advancing at a fast pace making smart drones a reality [13].

1.5. Applications

Although the precursors of commercial drones were their military counterparts, their designs do not resemble them now. These commercial drones are usually built on a small platform which are cheap and are easily acquired. These drones are not the heavy lifter unlike the military ones, though they still are useful as they can carry various sensors and small cameras. The applications can range from surveillance and monitoring to filming high definition film shots or search and rescue. Few of the applications that are already in place discussed below:

1.5.1. Search and Rescue

In April of 2018, a drone was used by the Dalvik Search and Rescue team in Iceland to locate two stranded cousins who were stuck on a mountain. The drone was used to locate the cousins and helped the rescuers plot a rescue route to them as the terrain was very tough to negotiate through and having an aerial view of it made it easier to plot a walkable course to the victims [14]

1.5.2. Entertainment and Videography

This is one of the most popular uses of drones in the current times. There are plenty of examples of drones being used to film shots for various entertainment purposes. They are used in very high budget films to personal videography and also sports cinematography [8] [15]. This is possible due to the drop in the cost of both the drones and high-quality light weight cameras for the drones.

Figure 1.1 DJI MAVIC PRO is a very popular commercial drone used for Filmography **[15]**

1.5.3. Precision agriculture

In farming, examining crop conditions is a very cumbersome task, especially when the surface area is in the units of hectares. Here, the drones are being used to observe, measure and respond to crop conditions continuously helping the farmers to be more efficient with farming resources such as fertilizers and water. This helps in increasing the productivity of the farm with minimal effort. Also, this alerts the farmers of any disease or flooding taking place in the crops, giving the farmer enough to time react and minimize damages [8] [16].

1.5.4. Industrial Inspections

One of the very useful applications of drones is to use them to inspect huge industrial structures such as bridges, Windmills, Electrical towers and so on. Drones have made it easier and faster to fly to very high, inaccessible points and record or relay high quality videos to the engineers on the ground making the whole ordeal quite efficient and easy. For instance, San Diego Gas and Electric company inspects power lines and pipeline which in area that are difficult to reach. The send live images showing the conditions and any wear and tear to the structures [17].

Chapter 2

Quadcopter Model

In the first part of this chapter, a description of the used reference frames and the available control commands is carried out, and in the second part the quadrotor dynamic model is developed by defining the coordinate frames.
The MPC strategy uses the model of a system to predict its behaviour into the future based on the length of its horizon period. Based on the model prediction, the optimizer in the MPC minimizes the cost function. The inputs found are then applied to the system. Therefore, it is important that the mathematical model of the system is accurate. If it is not accurate enough, then the obtained inputs, that were obtained from the model will not influence the real system as expected. Its response might become underdamped, overdamped or even unstable. This chapter focuses on defining the coordinate frames used to control the drone and it establishes the mathematical model of a quadcopter.

2.1. Quadcopter

Multi-copters can be divided into quadcopters, hexacopters or even octacopters. The name indicates the number of rotor arms and propellers. The most common type among theses multicopter is the quadcopter.
A quadcopter is an aircraft lifted and propelled by four horizontal rotors located at the ends of a cross structure. They are classified as rotorcrafts or a rotary wing aircraft to distinguish them from fixed-wing aircraft. Quadcopter derived its source of lift from the rotor blades rotating around a mast.

2.2. Preliminary notions

Before studying the mathematical model of a quadcopter, it is really important to describe the reference coordinate frames. The two possible standards that can be followed are:

1. Aeronautical reference frame standard: Z-axis pointing downwards
2. Robotics reference frame standard: Z-axis pointing upwards

The robotics reference frame is used throughout the project. Accordingly, the coordinate frames for this project are defined by considering two reference frames.
A fixed ground reference frame (Earth, E-frame) in Figure 2.1 and a Body fixed frame (B-frame) in Figure 2.2.

E-frame (in green): East, North, Up (ENU). It has the axis N, that point towards the North, the axis E, that point towards the East, and the axis U that is perpendicular to its plane. This can be seen in Figure 2.1 [18].

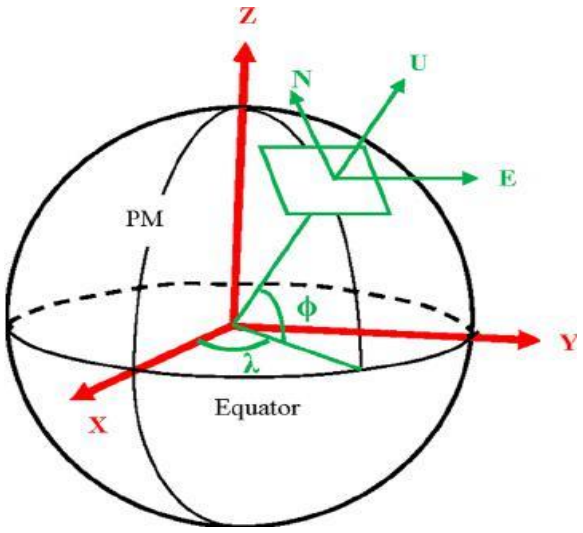


Figure 2.1 Fixed ground reference frame **[18]**

B-frame: Attached to the quadcopter, can be seen in Figure 2.2. It is more suitable to use E-frame for position measurement and the B-frame for the velocity measurement [18].

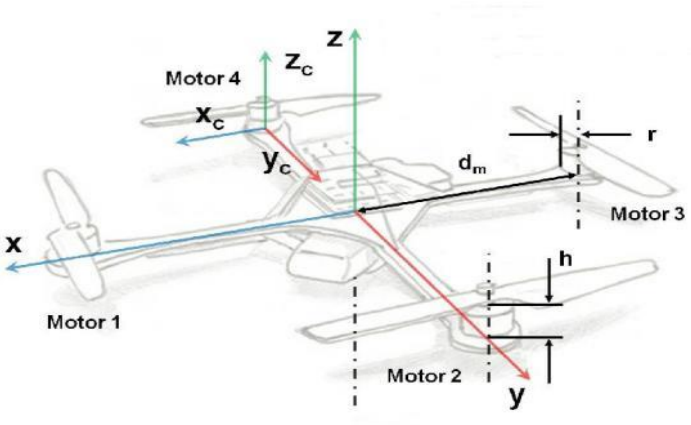


Figure 2.2 The body reference frame **[18]**

2.3. Quadcopter equations of motion

Describing Motion: This section explains how the motion of the quadcopter is described along with the necessary variables that are required to describe the motion.

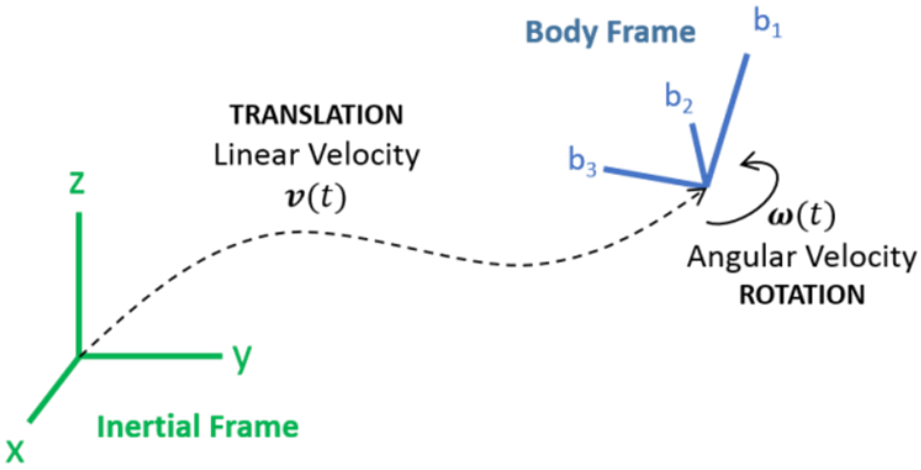


Figure 2.3 Modelling quadcopter's dynamics **[19]**

Calculation of variables necessary to describe the motion: Linear and angular velocity of the quadcopter.

The linear velocity of the body-frame of the quadcopter is defined as:

$$\mathbf{v}^b = \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} \text{longitudinal velocity} \\ \text{lateral velocity} \\ \text{normal velocity} \end{bmatrix}$$

The angular velocity of the rotating body-fixed reference frame is defined as:

$$\boldsymbol{\omega} = \begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \text{roll rate} \\ \text{pitch rate} \\ \text{yaw rate} \end{bmatrix}$$

Forces and Moments calculation which influence the motion of the quadcopter:
Linear Translation force:

$$\text{Force} = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix}$$

Angular Moment:

$$\text{Moment} = \begin{bmatrix} L \\ M \\ N \end{bmatrix}$$

Position and Orientation: A quadcopter can orientational and positional attributes i.e. any position (x, y, z coordinates) and angles (theta (θ), phi (φ) and psi (ψ)) with respect Inertial frame.

Euler angles must be tracked in order to convert between the inertial and body frames

$$\mathbf{\Phi} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \text{bank angle} \\ \text{pitch attitude} \\ \text{heading} \end{bmatrix}$$

The above variables are sufficient enough to describe all of the dynamics of the quadcopter vehicle. However, they do not specify the actual position in space of the vehicle because things that change with location such as aerodynamics and gravity will be considered constant, and only enter the equations through the force and moment variables. Here comes the need to define more variables for position in the inertial frame [19]

$$\text{Navigation Coordinates} = \begin{bmatrix} x \\ y \\ h \end{bmatrix} = \begin{bmatrix} \text{longitudinal position} \\ \text{lateral position} \\ \text{height} \end{bmatrix}$$

The motion for translation as well as rotation is expressed with respect to the inertial frame (e.g. the ground). Assumption: A rigid body, which means there can be no motion with respect to the body frame. The body frame is fixed to the Quadcopter, and any motion of the quadcopter with respect to that origin would imply flexible bending or movement of the quadcopter components. The body frame is very useful as the quadcopter's sensors and propellers are attached to it.

The equation of motion derived will be inertial motion but expressed within the coordinate system of our body axes.

A quadcopter has six degrees of freedom - three position and three attitude dimensions. These are *x, y, z and φ, θ, ψ*, respectively. The mathematical model of a quadcopter has to incorporate all the degrees of freedom. One way to express a

mathematical model of a drone is to write it in the B-frame. The system of equations describes the drone in the B-frame. The variables *u*, *v* and *w* are *x*, *y* and *z* velocities in the B-frame [ms^{-1}], respectively. The variables *p*, *q* and *r* are the angular velocities of *φ, θ, ψ* in the B-frame [$rad\ s^{-1}$], respectively. The Ω is the added rotation of all the rotors [20]. The constant g is the gravitational acceleration on the surface of the Earth, which is 9.81 ms^{-2}. Finally, the constant J_{TP} [Nms^2] is the total rotational moment of inertia around the propeller axis [18].

$$\begin{aligned}
\dot{u} &= (vr - wq) + g\sin\theta \\
\dot{v} &= (wp - ur) - g\cos\theta\sin\phi \\
\dot{w} &= (uq - vp) - g\cos\theta\cos\phi + \frac{U_1}{m} \\
\dot{p} &= qr\frac{I_y - I_z}{I_x} - \frac{J_{TP}}{I_x}q\Omega + \frac{U_2}{I_x} \\
\dot{q} &= pr\frac{I_z - I_x}{I_y} + \frac{J_{TP}}{I_y}p\Omega + \frac{U_3}{I_y} \\
\dot{r} &= pq\frac{I_x - I_y}{I_z} + \frac{U_4}{I_z}
\end{aligned} \tag{2.1}$$

The system of equations (2.1) is in a convenient form because it only contains first order differentiation. However, the problem with expressing everything in the B-frame is that now all six degrees of freedom states are velocities. However, the trajectory is given in the position values of *x*, *y* and *z* in the E-frame. Therefore, it is needed to have a system of equations in which the translational motion states are in the E-frame position format. The rotational motion states can stay in the B-frame as angular velocities. This Hybrid-frame (H-frame) can be seen in the set of equations (2.2) [18].

$$\begin{aligned}
\ddot{x} &= (\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi)\frac{U_1}{m} \\
\ddot{y} &= (\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi)\frac{U_1}{m} \\
\ddot{z} &= -g + \cos\phi\cos\theta\frac{U_1}{m} \\
\dot{p} &= qr\frac{I_y - I_z}{I_x} - \frac{J_{TP}}{I_x}q\Omega + \frac{U_2}{I_x} \\
\dot{q} &= pr\frac{I_z - I_x}{I_y} + \frac{J_{TP}}{I_y}p\Omega + \frac{U_3}{I_y} \\
\dot{r} &= pq\frac{I_x - I_y}{I_z} + \frac{U_4}{I_z}
\end{aligned} \tag{2.2}$$

The H-frame contains the position variables in the E-frame. However, now the problem is that it has second order differentiation in it. The MATLAB® integrator ode45 needs first order differential equations though. In addition, the H-frame does not contain the angles *φ, θ, ψ*, which are the orientation of a drone in the E-frame. To solve these two problems, the H-frame system of equations can be expanded. The relationship between the E and B-frame can be used to create one large system of equations that contains all the six states in the B and also in the E-frame. In total, 12 states, which are: *u*, *v*, *w*, *p*, *q*, *r*, *x*, *y*, *z*, *φ, θ, ψ*. This system of equations would be first order and therefore suitable for the ode45 integrator. The rotational matrix under the Z-Y'-X" Euler angles convention that relates the translational velocities in the B-frame (*u*, *v*, *w*) to the translational velocities in the E-frame, ($\dot{x}$, $\dot{y}$, $\dot{z}$) can be seen in the equation (2.3) [21]. The transformation matrix that relates the angular velocities in the B-frame (*p*, *q*, *r*) to the angular velocities in the E-frame ($\dot{\phi}$, $\dot{\psi}$, θ) can be seen in the equation (2.4) [18].

$$R = \begin{bmatrix} \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\, k\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \quad \textbf{(2.3)}$$

$$T = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix} \quad \textbf{(2.4)}$$

Now, there are all the tools needed to build a global system of equations with all the states from the both frames and that is also first order, suitable for the ode45 integrator. The global open loop system that will be integrated in this thesis while running the simulations can be seen in the equation (below). The system of equations in this configuration allows all the states (*u*, *v*, *w*, *p*, *q*, *r*, *x*, *y*, *z*, *φ, θ, ψ*) to be tracked. Once an initial value is given to them, the set of equations (2.5) computes their derivatives and then it is possible to know the state values in the next sample time period.

$$\dot{u} = (vr - wq) + g\,sin\,\theta$$

$$\dot{v} = (wp - ur) - g\,cos\,\theta\,sin\,\phi$$

$$\dot{w} = (uq - vp) - g\,cos\,\theta\,cos\,\phi + \frac{U_1}{m}$$

$$\dot{p} = qr\frac{I_y - I_z}{I_x} - \frac{J_{TP}}{I_x}q\Omega + \frac{U_2}{I_x}$$

$$\dot{q} = pr\frac{I_z - I_x}{I_y} + \frac{J_{TP}}{I_y}p\Omega + \frac{U_3}{I_y} \qquad \textbf{(2.5)}$$

$$\dot{r} = pq\frac{I_x - I_y}{I_z} + \frac{U_4}{I_z}$$

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{bmatrix} = R\begin{bmatrix}u\\ v\\ w\end{bmatrix}$$

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = T\begin{bmatrix}p\\ q\\ r\end{bmatrix}$$

2.4. Control Inputs

The attitude and position of the quadcopter can be controlled by modifying the speed of the four motors to obtain the desired values. To understand the input signals of the quadcopter, it is important to clarify how the four rotors move.

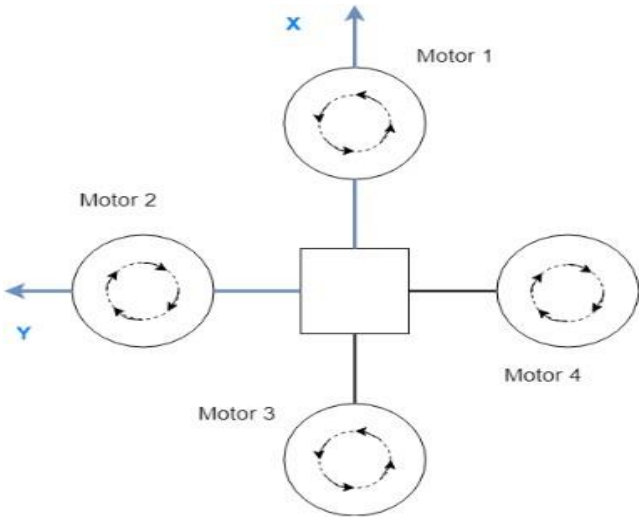


Figure 2.4 Direction of propellers rotation **[18]**

From the Figure 2.3, it can be seen that the motors 1 and 3 rotate anti-clockwise and the rotors 2 and 4 rotate clockwise. The body axes are positioned in such a way that the positive x-direction points towards motor 1 and the positive y-direction point towards motor 2. The following forces and moments can act in a quadcopter:

Thrust caused by the rotor's rotation, the pitching and rolling moment caused by the difference of the four rotors thrust, the gravity force, the gyroscopic and the yawing effect. The yaw moment of the quadcopter is generated by the reactive torque produced by each rotor. When the four rotor speeds are the same, the reactive torques will balance each other and the quadcopter will not rotate. Whereas, if the four rotor speeds are not same, the reactive torques will not be balanced and the quadcopter will start to rotate around the Z-axis.

The quadcopter has six degrees of freedom. It can move longitudinally (forward and backward), laterally (right and left), vertically (up and down) and also move rotationally among each axis to produce roll, pitch and yaw movements. See the figure below

Depending on the rotational speed of each propeller, it is possible to identify the four basic movements. There are 4 input signals that are introduced into the system: U1, U2, U3 and U4 [18].

Throttle U1 [N]

This command is provided by increasing, or decreasing, all the propeller speeds by the same amount. It leads to a vertical force with respect to the B-frame
This input signal is a force that points towards the z-axis of the B-frame. This force is generated by the angular rotation of all the rotors. It does not matter if the rotation of one rotor is faster than the spinning of the other three motors. The thrust force that all the rotors generate are summed up resulting in the global thrust force called U1. It can be clearly seen from the Figure 2.4 below that the added rotation of each of the rotors contributes to generating the thrust force U1 [18].

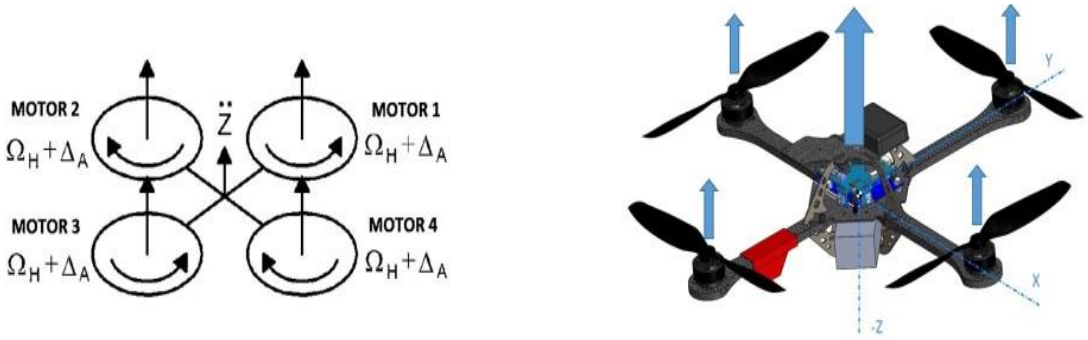


Figure 2.5 Thrust force U1 generated by the 4 rotors of the quadcopter **[18]**

Roll U2 [N m]

This command is provided by increasing (or decreasing) the left propeller speed and by decreasing (or increasing) the right one. It leads to a torque with respect to the X-ais in body frame, which makes the quadrotor turn. The input signal U2 is a torque signal, which is around the x-axis of the body frame. It can be seen visually in the Figure 2.5 [18]. In order to produce this input signal, the rotation of the rotors in motors 1 and 3 must be equal; however, the spinning of the motors 2 and 4 must be different. That creates an imbalance in the thrust force around the x-axis, which will create a moment around it, which is the control input signal U2. That moment, which also depends on how far the rotors are from the center of the drone, causes the UAV to rotate around the x-axis of the body frame.

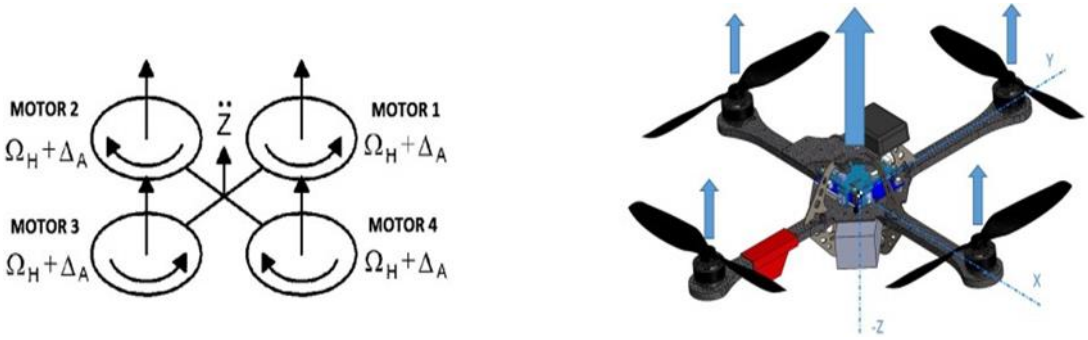


Figure 2.6 Moment U2 generated by the 4 rotors of the quadcopter **[18]**

Pitch U3 [N m]

This command is similar to the roll one and is provided by increasing (or decreasing) the rear propeller speed and by decreasing (or increasing) the front one. This led to a torque with respect to the Y-axis in the body frame which makes the quadrotor turn. The input signal U3 is a torque signal, which is around the y-axis of the body frame. It can be seen in the Figure 2.6 [18]. In order to produce this input signal, the rotation of the rotors in motors 2 and 4 must be equal; however, the spinning of the motors 1 and 3 must be different. That creates an imbalance in the thrust force around the y-axis, which will create a moment around it, which is the control input signal U3. That moment, which also depends on how far the rotors are from the centre of the drone, causes the quadcopter to rotate around the y-axis of the body frame.

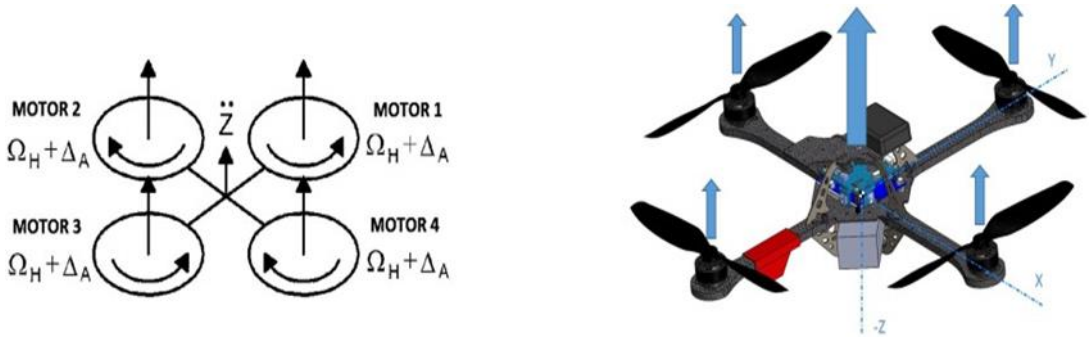


Figure 2.7 Moment U3 generated by the 4 rotors of the quadcopter **[18]**

Yaw U4 [N m]

This command is provided by increasing (or decreasing) the front-rear propellers speed and by decreasing (or increasing) that of the left-right couple. It leads to a torque with respect the Z-axis in body frame which makes the quadrotor turn
The input signal U4 is a torque signal, which is around the z-axis of the body frame. It can be seen in the Figure 2.7 [18]. In order to produce this input signal, the rotation of the rotors in motors 1 and 3 must be equal, and the rotation of the rotors 2 and 4 must be equal; however, the spinning of the motors 1 and 3 must be different from the spinning of the motors 2 and 4. Due to conservation of angular momentum, the Yaw moment, which is the U4 input signal will be generated. Due to that moment, the quadcopter starts rotating around the z-axis of its body frame.

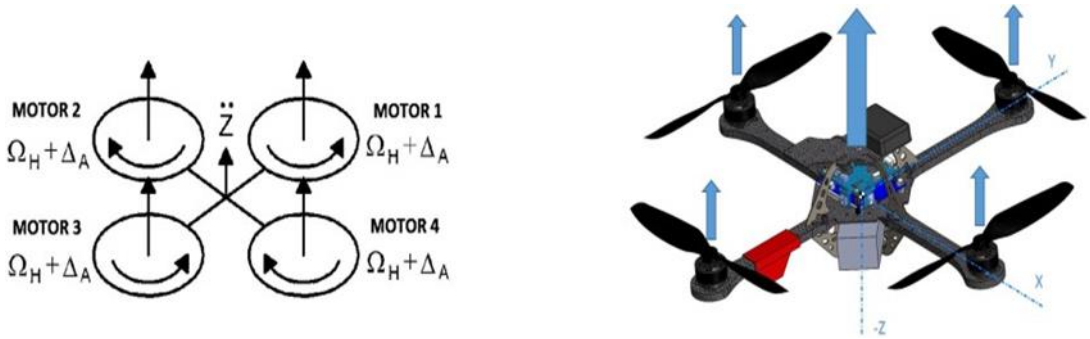


Figure 2.8 Moment U4 generated by the 4 rotors of the quadcopter **[18]**

$$U1 = m\ddot{z}$$

$$U2 = I_x\ddot{\phi}$$

$$U3 = I_y\ddot{\theta} \qquad \textbf{(2.6)}$$

$$U4 = I_z\ddot{\psi}$$

In the system of equations (2.6), it is shown how the force U1 and the moments U2, U3 and U4 are related to the acceleration in the z-axis and the angular accelerations of φ, θ and ψ, respectively. Here, *m, Ix, Iy, Iz* are the quadcopter's mass and the values of its angular momentum about the axes specified in their subscript, respectively. The double-dots mean the second time derivative of the variables, which in this case are their acceleration values.

In case of a reference tracking problem, the closed loop controller for the UAV gives the input signals U1, U2, U3 and U4 directly to the drone. Based on these inputs, the four rotors of the quadcopter rotate accordingly. That means that there must be a relationship between the input signals and the angular velocities of the rotors. In the system of equations (2.7) [18], it can be seen very clearly how the control input signals are related to the angular velocities of the rotors, which are denoted as Ω_1, Ω_2, Ω_3, Ω_4 [$rad \cdot s^{-1}$] for the motors 1, 2, 3 and 4, respectively. The values of c_T [Ns^2] *and* cQ [Nms^2] are aerodynamic coefficients of thrust and drag, respectively. The value of ℓ [*m*] is the distance between the center of the quadrotor and the center of a propeller. Finally, the equation (2.8) adds up the rotational velocities of all the rotors. Since the propellers 1 and 3 rotate counter-clockwise and the propellers 2 and 4 rotate clockwise, the motors 1 and 3 have the opposite sign compared to the motors 2 and 4

$$U1 = C_T \cdot (\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2)$$

$$U2 = C_T \cdot \ell \cdot (\Omega_4^2 + \Omega_2^2)$$

$$U3 = C_T \cdot \ell \cdot (\Omega_3^2 - \Omega_1^2) \quad \textbf{(2.7)}$$

$$U4 = C_Q \ .(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2)$$

such that,

$$\Omega_{total} = -\Omega_1 + \Omega_2 + \Omega_3 + \Omega_4 \quad \textbf{(2.8)}$$

Chapter 3

Control Strategy

3.1. Introduction

This chapter is about designing a suitable control strategy for the quadcopter.

3.1.1. Model Predictive Control (MPC)

MPC is an effective way of dealing with large multivariable constrainted control problems. The idea is to choose the control action by solving online an optimal control problem repeatedly, which aims to minimize a performance criterion over a future horizon. PID controllers do not perfrom well when applied to systems with significant time delays. MPC overcomes this problem of delayed feedback by using predictive future states of the output for control. MPC evaluates the future dynamics of the system by taking into account time delays, unstable systems, oscillatory systems among other and the possibility to introduce constraints in a natural form. Although MPC is computer expensive, it is highly effective to deal with the different types of nonlinearities and constraints [22].

Introduction to MPC frameworks:

Original MPC frameworks were grown autonomously during the 1970s by two spearheading mechanical exploration gatherings. Dynamic Matrix Control (DMC), contrived by Shell Oil (Cutler and Ramaker, 1980), and a related methodology created by ADRESA (Richalet et al., 1978) have very comparable abilities [23]. A versatile MPC strategy, Generalized Predictive Control (GPC), created by Clarke et al. (1987) has additionally gotten extensive consideration. Model prescient control has majorly affected modern practice. For instance, a MPC study by Qin and Badgwell (2003) revealed that there were more than 4,500 applications worldwide before the finish of 1999, essentially in petroleum treatment facilities and petrochemical plants [22]. In these businesses, MPC has become the strategy for decision for troublesome multivariable control issues that incorporate disparity limitations.

3.1.2. Linear parameter varying (LPV) techniques

LPV models a class of non linear systems into parametrized linear systems whose paramter change with state. It is well suited for control of dynamical systems with paramter variations.

Introduction to LPV Frameworks: LPV frameworks are an extremely extraordinary class of nonlinear frameworks which have all the earmarks of being appropriate for control of dynamical frameworks with boundary varieties. LPV methods give a methodical plan technique to increase booked multivariable controllers. This strategy permits execution, power, and data transmission impediments to be fused into a bound together system.

Boundary subordinate frameworks: In control designing, a state-space portrayal is a numerical model of a physical framework as a lot of information, yield, and state factors, related by first-order differential conditions.

In the event that the framework is time-variation:

The state factors portray the numerical "state" of a dynamical framework and in demonstrating enormous complex nonlinear frameworks if such state factors are picked to be conservative for reasonableness and straightforwardness, at that point portions of the dynamic advancement of a framework are absent. The state-space portrayal will include different factors considered exogenous factors whose advancement is not comprehended or is too convoluted to be in any way demonstrated however influence the state factors development in a known way and are quantifiable progressively utilizing sensors. At the point when an enormous number of sensors are utilized, a portion of these sensors measure yields in the framework hypothetical sense as known, unequivocal nonlinear elements of the displayed states and time, while different sensors are exact assessments of the exogenous factors. Consequently, the model will be a period differing, nonlinear framework, with the future time variety obscure, however, estimated by the sensors progressively [24]
Boundary subordinate frameworks are straight frameworks, whose state-space portrayals are known elements of time-changing boundaries. The time variety of every one of the boundaries is not known ahead of time however is thought to be quantifiable continuously. The controller is confined to be a direct framework, whose state-space passages rely causally upon the boundary's history. There exist three unique techniques to plan an LPV controller to be specific,

1. Linear fragmentary changes which depend on the little addition hypothesis for limits on execution and power.
2. Single Quadratic Lyapunov Function (SQLF)
3. Parameter Dependent Quadratic Lyapunov Function (PDQLF) to bound the attainable degree of execution [25] [26].

These issues are understood by reformulating the control structure into limited dimensional, curved achievability issues which can be tackled precisely, and unbounded dimensional arched attainability issues which can be illuminated around. This definition establishes a sort of addition planning issue and differentiations to old-style gain booking, this methodology tends with the impact of boundary varieties with guaranteed soundness and execution [26].

3.2. Proposed control strategy

This section describes the proposed control strategy i.e., using Position controller with an LPV-MPC controller. As using just LPV-MPC approach to control the entire system fails at controlling the quadcopter because of the hard non linearities in its mathematical model.

The schematic diagram can be seen in Figure 4.1, where the controller is split into two sub controllers. One controller is responsible for the position variables *x*, *y*, *z*. This

position controller uses the state feedback linearization method. The outputs it generates are U1, and the angles φ *and* θ. The aforementioned angles are the ones that, together with U1 and ψ, are necessary in order for the quadcopter to reach its *x*, *y*, *z* reference position. U1 is fed straight as an input into the open loop system. The angles φ *and* θ are then fed into the attitude controller as reference values which uses the LPV-MPC approach. The reference angles φ *and* θ, together with the angle ψ from the reference generator, allows the LPV-MPC controller to find the remaining three control actions for the open loop system, which are U2, U3 and U4. The inner loop (the loop for the attitude controller) works 4 times faster.

After integrating the open-loop system, the new angles φ, θ and ψ are fed back into the LPV-MPC controller together with the new Ω value at every one-fourth of the sample time. In addition, at every sample time, the open loop system sends the new *x*, *y* and *z* values back into the position controller which can also be seen in the schematic below.

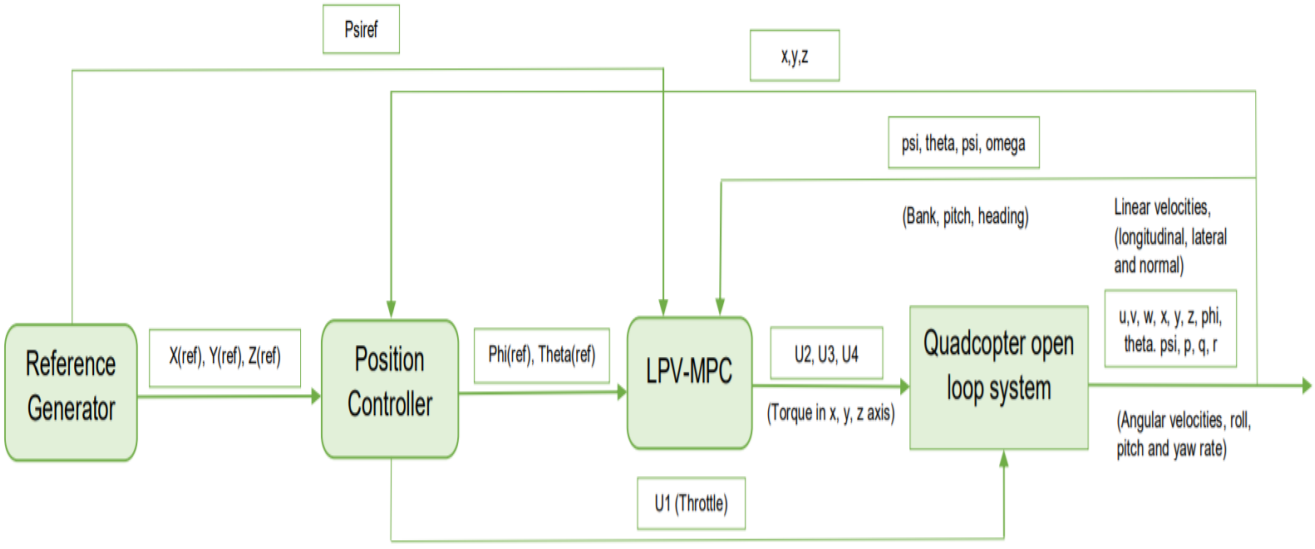


Figure 3.1 Control strategy: LPV-MPC applied in combination with a position controller

Keep in mind that the LPV-MPC controller needs time to push the state angles towards its reference values. Therefore, the attitude controller must work at a higher frequency compared to the position controller.

3.3. LPV model mathematical derivation

In this section, it is shown how the nonlinear quadcopter model is expressed in the linear parameter variance (LPV) format. This linear format is required so that the most basic MPC strategy made can be applied on it. All the nonlinearities are contained in A and B matrices of a state space equation.

To derive the LPV model of the system, the second order differential equations (the situation equations) needs to be transformed into linear state space format, the system of equations need to be expanded where the states are x, $\dot{x}$, y, $\dot{y}$, z, $\dot{z}$.

A state space system for linear systems is shown below

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{y} \\ \ddot{y} \\ \dot{z} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \\ z \\ \dot{z} \end{bmatrix} + \begin{bmatrix} 0 \\ (cos\,\phi\, sin\,\theta\, cos\,\psi + sin\,\phi\, sin\,\psi)\frac{1}{m} \\ 0 \\ (cos\,\phi\, sin\,\theta\, sin\,\psi - sin\,\phi\, cos\,\psi)\frac{1}{m} \\ 0 \\ -\frac{g}{U_1} + cos\,\phi\, cos\,\theta\,\frac{1}{m} \end{bmatrix} U_1$$

In order to have an LPV model for the angles, the last three equations of a system of equations (2.2) will be considered

However, these equations are in the B-frame. To control the quadcopter, the LPV model needs to contain the angles and their instantaneous changes in the E-frame. In case of angles, both of the frames are related to each other by the transformation matrix *T* in equation (2.4) [18]. The aim here is to design a controller that stabilizes the quadcopter close to the hovering position since the quadcopter cannot hover in one position if it is tilted in one direction all the time (it would start sliding down diagonally). The angles φ and θ are assumed to be zero, which makes the *T* matrix in the equation an identity matrix I. This converts the last three equations in the system of equations (2.2) into a system of equations, in which p, q, r become $\dot{\phi}$, $\dot{\theta}$, $\dot{\psi}$, respectively [18].

$$\begin{aligned} \ddot{\phi} &= \dot{\theta}\dot{\psi}\frac{I_y - I_z}{I_x} - \frac{J_{TP}}{I_x}\dot{\theta}\Omega + \frac{U_2}{I_x} \\ \ddot{\theta} &= \dot{\phi}\dot{\psi}\frac{I_z - I_x}{I_y} + \frac{J_{TP}}{I_y}\dot{\phi}\Omega + \frac{U_3}{I_y} \\ \ddot{\psi} &= \dot{\phi}\dot{\theta}\frac{I_x - I_y}{I_z} + \frac{U_4}{I_z} \end{aligned} \qquad \textbf{(3.1)}$$

These second order equations (3.1) are expanded to generate an LPV format for the angles [18]. The LPV format is shown in equation, in which it can be seen that all the nonlinearities are encapsulated in the *A* matrix. The *B* matrix only has constant values. The *A* matrix is multiplied by the states and the *B* matrix is multiplied by the inputs

$$\begin{bmatrix} \dot{\phi} \\ \ddot{\phi} \\ \dot{\theta} \\ \ddot{\theta} \\ \dot{\psi} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{\Omega \cdot J_{TP}}{I_x} & 0 & \dot{\theta} \cdot \frac{I_y - I_z}{I_x} \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & \frac{\Omega \cdot J_{TP}}{I_y} & 0 & 0 & 0 & \dot{\phi} \cdot \frac{I_z - I_x}{I_y} \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & \frac{\dot{\theta}}{2} \cdot \frac{I_x - I_y}{I_z} & 0 & \frac{\dot{\phi}}{2} \cdot \frac{I_x - I_y}{I_z} & 0 & 0 \end{bmatrix} \begin{bmatrix} \phi \\ \dot{\phi} \\ \theta \\ \dot{\theta} \\ \psi \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ \frac{1}{I_x} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \frac{1}{I_y} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{1}{I_z} \end{bmatrix} \begin{bmatrix} U_2 \\ U_3 \\ U_4 \end{bmatrix} \quad \textbf{(3.2)}$$

The position is handled by the feedback linearization strategy that does not require an LPV model of the system. Hence, in the proposed control strategy with the position controller, only the equation (3.2) is used in the MPC control [18].

The important point to note here is that the LPV model is continuous. However, the controller works discretely. That is why the LPV system needs to be discretized.

3.4. MPC Mathematical derivation

In this section, it is shown how the MPC technique was derived to match the MATLAB® quad-prog solver that allows to solve a problem with the following structure. The MPC strategy for linear systems is applied to the LPV model [27]

$$\min_x \frac{1}{2} x^T H x + f^T x \ such\ that \begin{cases} M \cdot x \leq b \\ Aeq \cdot x = beq \\ lb \leq x \leq ub \end{cases} \quad \textbf{(3.3)}$$

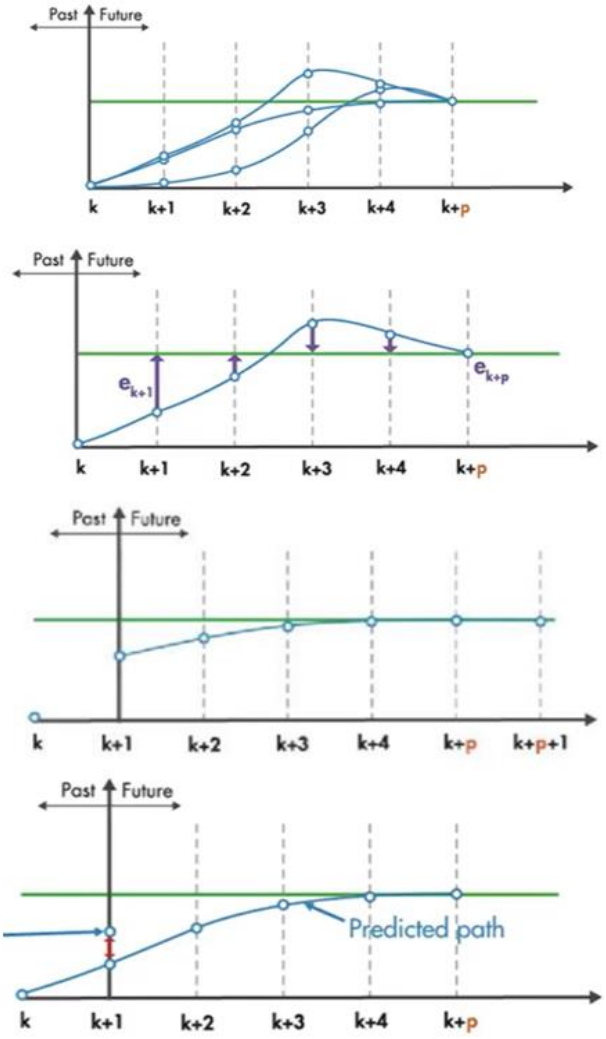


Figure 3.2 MPC Intuition

In Figure 3.2, a simple intuition into the MPC strategy is presented [28]. The sequence goes from up to down. The sample times go from the sample *k* to the sample *k* + *N*.

The parameter *N* is called the prediction horizon. The length of it depends on the system dynamics. The goal of the MPC is to stabilize itself around the reference during the prediction horizon. In the first sub-image, it can be seen that there are many ways to get there. The algorithm determines the error values in each sample time as it can be seen in the second sub-image. It then takes two main features into account - the squared sum of the errors (e) and the squared sum of the change of inputs (*δu*) as it can be seen in equation (3.4) [28]. The importance of errors and change of inputs can be regulated with weights (*w*). This entire equation is called a cost function (*J*)

$$J = \sum_{i=1}^{N} w_e e_{k+i}^2 + \sum_{i=0}^{N-1} w\Delta u \Delta u_{k+i}^2 \qquad \textbf{(3.4)}$$

MPC uses a solver that finds a set of change of inputs such that the cost function is minimized. In the third sub-image in Figure (3.2), one can see the predicted path that was generated by the inputs that the solver had previously chosen [28]. However, due to disturbances and uncertainties, the system might end up being slightly off from the prediction in the next sample period. In the third sub-image, it is slightly above the predicted value. Therefore, only the first element of the input vector is chosen - the rest are discarded. Then, the horizon period shifts and it goes from *k + 1* to *k + N + 1*. A new prediction is made from the most recent position as it can be seen in the fourth sub-image. That position might be measured or a combination of measured and predicted position that comes out of a filter such as the Kalman filter.

When dealing with MPC, a distinction must be made, which depends on the objectives. If MPC is used for the purpose of regulation, then that means that the goal is to bring the state values close to zero. In this case, the linear prediction model is

$$x_{k+1} = Ax_k + Bu_k \qquad \textbf{(3.5)}$$

$$yk = Cx_k + Du_k \qquad \textbf{(3.6)}$$

where *x* is a state variable, *u* is the input variable, and *y* is the output vector that is related to a state vector through the *C* matrix. The *D* matrix is assumed to be zero as it is with most systems. This model is discretized. This thesis makes us of both, the forward Euler method, and the zero-hold order (zoh) for discretization. In the case of regulation, the cost function of MPC is

$$J = min\frac{1}{2}x_{t+N}^{T}Sx_{t+N} + \frac{1}{2}\sum_{k=0}^{N-1}(x_{t+k}^{T}Qx_{t+k} + u_{t+k}^{T}Ru_{t+k}) \quad \textbf{(3.7)}$$

Interesting point to note here is that the cost function does not deal with input changes, in the case of regulation (the inputs are absolute values), because once the states are all zero, the inputs can also be zero by assuming the system to be stable.

In addition, the last element in the horizon period has a different weight matrix, which is called *S.* Also, the cost function is multiplied by a value of 0.5. That is for convenience. When a gradient of it is taken, then the constant in the cost function becomes 1[}. Since the cost function shifts in time, the symbol *t* is the present time and *t* + *k* is *k* samples from the current present. The aim here is not regulation but reference tracking. In a tracking problem, in the cost function, the state variable is replaced by the error variable [29] leading to

$$J = min\frac{1}{2}e_{t+N}^{T}Sx_{t+N} + \frac{1}{2}\sum_{k=0}^{N-1}(e_{t+k}^{T}Qe_{t+k} + u_{t+k}^{T}Ru_{t+k}) \quad \textbf{(3.8)}$$

The error is defined in equation [29] as follows

$$e_k = r_k - y_k = r_k - Cx_k \quad \textbf{(3.9)}$$

The inputs must be nonzero in a tracking problem to keep the tracking error zero and keep the quadcopter to follow the reference so that it could track the desired trajectory. Once it reaches the reference value, the change of input can be zero assuming that the system is stable. Therefore, in a tracking problem, the changes of input Δ_{uk} are used, which are defined as follows [29]

$$\Delta u_k = u_k - u_{k-1}$$
$$u_k = u_{k-1} + \Delta u_k \quad \textbf{(3.10)}$$

Now, the state space equations (3.5) and (3.6) can be rewritten in the following way as [29]

$$x_{k+1} = Ax_k + B(u_{k-1} + \Delta u_k)$$
$$y_k = Cx_k \quad \textbf{(3.11)}$$

Now, the system is augmented where the absolute input one sample in the past u_{k-1} becomes a state as well. It can be written as [29]

$$\begin{bmatrix} x_{k+1} \\ u_k \end{bmatrix} = \begin{bmatrix} A & B \\ 0 & I \end{bmatrix} \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix} + \begin{bmatrix} B \\ I \end{bmatrix} \Delta u = \tilde{A}\widetilde{x_k} + \tilde{B}\Delta u \quad \textbf{(3.12)}$$

$$y_k = \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix} = \tilde{C} \begin{bmatrix} x_k \\ u_{k-1.} \end{bmatrix} \quad \textbf{(3.13)}$$

Once the error term in the cost function is substituted with the equation () and the *C* matrix and the states are replaced with the augmented version from equations (3.12) and (3.13), the cost function will have Δu instead of an absolute of *u* [29]

$$J = min\frac{1}{2}(r_{t+N} - \tilde{C}\widetilde{x_{t+N}})^T S(r_{t+N} - \tilde{C}\widetilde{x_{t+N}}) + \\ + \frac{1}{2}\sum_{k=0}^{N-1}((r_{t+k} - \tilde{C}\widetilde{x_{t+k}})^T \ Q(r_{t+k} - \tilde{C}\widetilde{x_{t+k}}) + \Delta u_{t+k}^T R \Delta_{u_{t+k}}) \quad \textbf{(3.14)}$$

Here, constant terms do not affect the results of the optimizer (The terms in the two boxes). Therefore, to simplify the equation, they will be crossed out as follows [29]

$$J \boxed{= \frac{1}{2} r_{t+n}^T \, S r_{t+N}} - r_{t+n}^T \, S\tilde{C}\tilde{x}_{t+N} + \frac{1}{2}\tilde{x}_{t+N}^T \tilde{C}^T S \tilde{C}\tilde{x}_{t+N} + \\ + \sum_{k=0}^{n-1} \left[\boxed{\frac{1}{2} r_{t+k}^T \, Q r_{t+N}} - Q\tilde{C}\tilde{x}_{t+k} + \frac{1}{2}\tilde{x}_{t+k}\tilde{C}^T Q \tilde{C}\tilde{x}_{t+k} + \frac{1}{2}\Delta u_{t+k}^T R \Delta u_{t+k} \right] \tag{3.15}$$

This form is only valid if the weight matrices are diagonal because then they are equal to their transpose values and this form can be achieved.

The horizon period can also be described by stacking the future reference values, states and change of inputs in one big vector, where each element represents one sample time period. It can be seen in equation (3.16) along with a present state vector denoted as x_t, which is written separately and does not form part of the future state values [29].

$$r = \begin{bmatrix} r_{t+1} \\ r_{t+2} \\ . \\ . \\ r_{t+N} \end{bmatrix} \quad \tilde{x} = \begin{bmatrix} \tilde{x}_{t+1} \\ \tilde{x}_{t+2} \\ . \\ . \\ \tilde{x}_{t+N} \end{bmatrix} \quad \Delta u = \begin{bmatrix} \Delta u_t \\ \Delta u_{t+1} \\ . \\ . \\ \Delta u_{t+N-1} \end{bmatrix} \quad \tilde{x}_t = present \tag{3.16}$$

In the global vector for the horizon period, the reference and the state values start from the period *t + 1* and end at *t + N*; however, the change of inputs start at the period t and end at *t + N - 1*. That is because an input in one period affects a state and an output in the next period.

$$J' = min\frac{1}{2}\tilde{x}^T \begin{bmatrix} \tilde{C}^T Q\tilde{C} & & & \\ & \cdot & & \\ & & \tilde{C}^T Q\tilde{C} & \\ & & & \tilde{C}^T S\tilde{C} \end{bmatrix} \tilde{x} - r^T \begin{bmatrix} Q\tilde{C} & & & \\ & \cdot & & \\ & & Q\tilde{C} & \\ & & & S\tilde{C} \end{bmatrix} \tilde{x} + \\ + min\frac{1}{2}\Delta u^T \begin{bmatrix} R & & & \\ & \cdot & & \\ & & R & \\ & & & R \end{bmatrix} \Delta u = min\frac{1}{2}\tilde{x}^T \overline{\overline{Q}}\tilde{x} - r^T \overline{\overline{T}}\tilde{x} + min\frac{1}{2}\Delta u^T \overline{\overline{R}}\Delta u \quad \textbf{(3.17)}$$

Equation (3.17), shows how the entire cost function is written in a way, such that the weight matrices are stacked into big diagonal matrices, which describe the entire horizon period [29].

Future state values are removed with the help of a mathematical manipulation which is performed below [30], because the aim is to write the cost function in the form that only has change of input values and state values in the present. The state values are substituted with the previous state values that only consist of the A and B matrices, the current state value, and the current and future change of input values.

The state values are substituted with the previous state values that only consist of the A and B matrices, the current state value, and the current and future change of input values

$$\begin{gathered} \tilde{x}_1 = \tilde{A}\tilde{x}_0 + \tilde{B}\Delta u_0 \\ \tilde{x}_2 = \tilde{A}\tilde{x}_1 + \tilde{B}\Delta u_1 = \\ = \tilde{A}^2\tilde{x}_0 + \tilde{A}\tilde{B}\Delta u_0 + \tilde{B}\Delta u_1 \\ \cdots \\ \tilde{x}_k = \tilde{A}^k\tilde{x}_0 + [\tilde{A}^{k-1}\tilde{B} \quad \tilde{A}^{k-2}\tilde{B} \cdots \tilde{B}] \begin{bmatrix} \Delta u_0 \\ \Delta u_1 \\ \cdots \\ \Delta u_{N-1} \end{bmatrix} \end{gathered} \quad \textbf{(3.18)}$$

The entire state space system for the entire horizon period can be compactly represented as [29],

$$\tilde{x} = \begin{bmatrix} \tilde{B} & & & & \\ \tilde{A}\tilde{B} & \tilde{B} & & & \\ \tilde{A}^2\tilde{B} & \tilde{A}\tilde{B} & \tilde{B} & & \\ . & & & . & \\ \tilde{A}^{N-1}\tilde{B} & . & . & . & \tilde{B} \end{bmatrix} \Delta u + \begin{bmatrix} \tilde{A} \\ \tilde{A}^2 \\ . \\ . \\ \tilde{A}^N \end{bmatrix} \tilde{x}_t = \overline{\overline{C}}\Delta u + \hat{\hat{A}}\tilde{x}_t. \quad \textbf{(3.19)}$$

The future state variables have disappeared from the cost function which can be seen in the equation (3.20),

$$J' = \frac{1}{2}(\overline{\overline{C}}\Delta u + \hat{\hat{A}}\tilde{x}_t)^T\overline{\overline{Q}}(\overline{\overline{C}}\Delta u + \hat{\hat{A}}\tilde{x}_t) + \frac{1}{2}\Delta u^T\overline{\overline{R}}\Delta u - r^T\overline{\overline{T}}(\overline{\overline{C}}\Delta u + \hat{\hat{A}}\tilde{x}_t) \Rightarrow \quad \textbf{(3.20)}$$

$$\Rightarrow ignoring\ constant\ terms$$

The constant terms are ignored again due because they do not influence the optimizer results.

Hence, the final results of the derivation can be seen in the following. Shown (in bold) is how the entire cost function is put in a form that can be accepted by the MATLAB® quad-prog solver

$$J'' = \frac{1}{2}\Delta u^T(\overline{\overline{C}}^T\overline{\overline{QC}} + \overline{\overline{R}})\Delta u + \begin{bmatrix} \tilde{x}_t^T & r^T \end{bmatrix} \begin{bmatrix} \hat{\hat{A}}^T\overline{\overline{QC}} \\ -\overline{\overline{TC}} \end{bmatrix} \Delta u =$$

$$= \frac{1}{2}\Delta u^T\overline{\overline{H}}\Delta u + \begin{bmatrix} \tilde{x}_t^T & r^T \end{bmatrix}\overline{\overline{F}}^T\Delta u = \mathbf{\frac{1}{2}\Delta u^T\overline{\overline{H}}\Delta u + f^T\Delta u} \quad \textbf{(3.21)}$$

The first element of the set of Δu-s is then used to move the quadcopter. All the other elements will be discarded and in the next sample time period, the same process is repeated

3.5. Position controller

This section describes the feedback linearization method in the position controller that is used in proposed control strategy along with MPC-LPV controller. The system of the second order differential equations that govern the Quadcopter's position are [18],

$$\ddot{x} = (cos\,\phi\, sin\,\theta\, cos\,\psi + sin\,\phi\, sin\,\psi)\frac{U_1}{m}$$

$$\ddot{y} = (cos\,\phi\, sin\,\theta\, cos\,\psi - sin\,\phi\, cos\,\psi)\frac{U_1}{m} \quad \textbf{(3.22)}$$

$$\ddot{z} = -g + cos\,\phi\, cos\,\theta\frac{U_1}{m}$$

The system can be expressed in the state-space form as follows [18],

$$\dot{x}_1 = x_2$$

$$\dot{x}_2 = (cos\,\phi\, sin\,\theta\, cos\,\psi + sin\,\phi\, sin\,\psi)\frac{U_1}{m}$$

$$\dot{x}_3 = x_4$$

$$\dot{x}_4 = (cos\,\phi\, sin\,\theta\, sin\,\psi - sin\,\phi\, cos\,\psi)\frac{U_1}{m} \quad \textbf{(3.23)}$$

$$\dot{x}_5 = x_6$$

$$\dot{x}_6 = -g + cos\,\phi\, cos\,\theta\frac{U_1}{m}$$

The feedback-linearization control strategy is developed by dealing with the strong non-linearities that the translation subsystem presents and hence writing the system as a system of first order differential equations where [18],

$$x_1 = x, x_2 = \dot{x} = x_3 = y, \quad x_4 = \dot{y}, x_5 = z, \; x_6 = \dot{z}$$

Besides position reference values *x*, *y*, *z*, the planner also needs to provide the position controller with the reference velocities) $\dot{x}, \dot{y}, \dot{z}$ that are calculated as follows

$$\begin{aligned} x_{t+1}^{\dot{R}} &= \frac{x_{t+1}^{R} - x_t^{R}}{T_s} \\ y_{t+1}^{\dot{R}} &= \frac{y_{t+1}^{R} - y_t^{R}}{T_s} \\ z_{t+1}^{\dot{R}} &= \frac{z_{t+1}^{R} - z_t^{R}}{T_s} \end{aligned} \qquad \textbf{(3.24)}$$

Next, the errors of the position and velocity values are computed (the errors between reference states and the estimated states). The velocity errors are differentiated one more time to get the acceleration of the error values. The second error derivative is essentially the negative of the acceleration of a position variable, because the reference velocity values are constant during one sample time and so, they become zeros. These errors are calculated as [18],

$$\begin{aligned} e_x &= x_t^R - x_t \qquad \dot{e}_x = x_t^{\dot{R}} - \dot{x}_t \quad \ddot{e}_x = -\ddot{x}_t = v_x \\ e_y &= y_t^R - y_t \qquad \dot{e}_y = y_t^{\dot{R}} - \dot{y}_t \quad \ddot{e}_y = -\ddot{y}_t = v_y \\ e_z &= z_t^R - z_t \qquad \dot{e}_z = z_t^{\dot{R}} - \dot{z}_t \quad \ddot{e}_z = -\ddot{z}_t = v_z \end{aligned} \qquad \textbf{(3.25)}$$

The variables are then chosen to be a control action for the linearized state feedback control strategy as [18],

$$v_x = -k_1^x e_x - k_2^x \dot{e}_x$$

$$v_y = -k_1^y e_y - k_2^y \dot{e}_y \quad \textbf{(3.26)}$$

$$v_z = -k_1^z e_z - k_2^z \dot{e}_z$$

From equation (3.22), the variables v_x, v_y, v_z can also be substituted with the second order differential equations that govern the Quadcopter's position, as it can be seen in the following [18],

$$v_x = -(cos\,\phi\, sin\,\theta\, cos\,\psi + sin\,\phi\, sin\,\psi)\frac{U_1}{m}$$

$$v_y = -(cos\,\phi\, sin\,\theta\, sin\,\psi - sin\,\phi\, cos\,\psi)\frac{U_1}{m} \quad \textbf{(3.27)}$$

$$v_z = -(-g + cos\,\phi\, cos\,\theta\frac{U_1}{m})$$

By choosing negative real poles, the constants in equation (3.26) can be computed [18]. Then, the values v_x, v_y, v_z are determined. In order to find the angles θ and φ for the attitude controller, which it will then use as reference angles, the equations (3.28) and (3.29) and are used, respectively. The constants *a*, *b*, *c* and *d* are calculated as

$$\theta = tan^{-1}(\, ac + bd) \tag{3.28}$$

$$if \ \left(|\psi_{ref}| \ < \frac{\pi}{4} \ or \ | \ \psi_{ref}| \ > \frac{3\pi}{4}\right)$$

$$\phi = tan^{-1}\left(\frac{cos(\,\theta)(tan(\,\theta)d - b)}{c}\right)$$

$$else$$

$$\phi = tan^{-1}\left(\frac{cos(\,\theta)(a - tan(\,\theta)c)}{d}\right) \tag{3.29}$$

$$a = \frac{v_x}{v_z + g}, b = \frac{v_y}{v_z + g}, c = cos\,\psi_{ref}\,, d = sin\,\psi_{ref} \tag{3.30}$$

The total lifting force U_1 is computed as:

$$U_1 = \frac{(v_z + g)\, m}{cos\,\phi \ \ cos\,\theta} \tag{3.31}$$

These angles, along with the ψ^R angle from the planner, will serve as reference angles for the LPV-MPC controller. However, the position controller also finds the input U1, that is directly fed into the nonlinear model. The control action U1 can be computed in equation [18]

Chapter 4

Implementation and Validation

This Chapter is divided into two sections where in the first section, the structure of the control strategy code is explained and in the second section the proposed control strategy is validated.

4.1. Implementation of the LPV-MPC and the position controller

This section explains how the LPV-MPC controller along with the position control is implemented. Figure 4.1 describes the structure of the control strategy code. The code can be found in Appendix B. The script consists of one main file which is main.m and six supportive files which are Defining_initial_constants.m, position_controller.m, LPV_Discretization.m, Quadcopter_model.m, MPC_simplification.m and Trajectory_generator.m. The defining_initial_constants.m is a library type function in which one can find constants and certain initial values. It is shown with arrows in which location in the main file the functions are used.

The main.m file first loads the initial and the constants values. It gets the trajectory, the reference velocities and the reference yaw angle from the trajectory generator.
Then, the outer loop begins where the position controller function is used. After that, the LPV-MPC loop starts that uses a function to get the discrete LPV model. The inner loop loops through the code four times per one loop of the outer loop. In the MPC simplification function, the necessary matrices for the solver are generated. After that, the solver is called. The results are then fed into the nonlinear quadcopter model, where the integration of the open loop system happens. Finally. the results are plotted, which can be seen in the validation section.

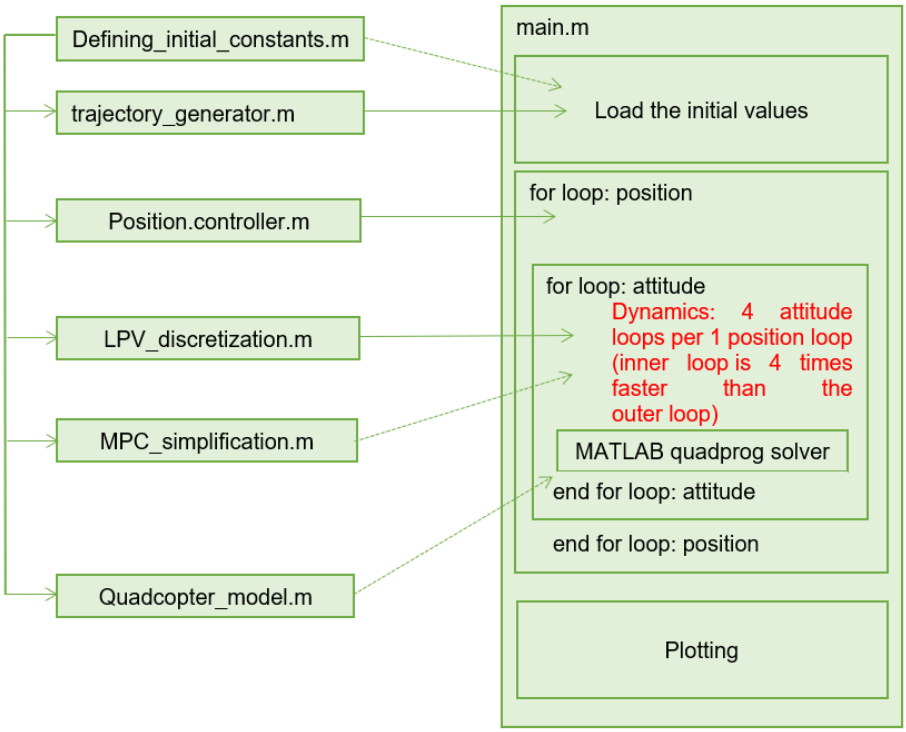


Figure 4.1 The Structure of the control strategy code

4.2. Validation of the controller

This section explains how the proposed control strategy is validated.

The initial rotational velocity of all the rotors is 3000 rad/s. At this rate, the quadcopter does not produce enough thrust in order to be able to leave the ground. The quadcopter is not tilted in terms of its roll and pitch angles. The initial yaw angle will be 90 degrees anti-clockwise from 3 o'clock. The initial x, y and z coordinates of the quadcopter are 0, -1, 0 meters, respectively. The weight matrices in MPC (Q, R, S) are all identity matrices. The parameters of the quadcopter in this thesis belong to *AscTec Hummingbird* [18].

The position and angular variables are decoupled. The position controller, which uses the feedback linearization methodology, computes the necessary *U1* input for the quadcopter. It also computes the angles φ and θ that are necessary for the quadcopter to be able to reach its target position. These angles along with the ψ angle from the reference generator are then fed into the LPV-MPC controller as reference values. In order to give the LPV-MPC controller time to adjust the quadcopter's angles and reach the target orientation, the inner control loop has to work faster. In this case, four times faster than the outer loop. The horizon period for MPC is also chosen to be four samples. Figures from 4.2 to 4.25, several trajectories are created to test how the

proposed position and LPV-MPC controller tracks different trajectories. The paths were created by trying to change the nature of each position (x, y, z) dimension.

Spiral trajectory Figure 4.2: After applying the MPC strategy, now it's time for the quadcopter to track this given trajectory. Smooth trajectory is preferable for this quadcopter. Function-Discrete-Continuous-Good approximation so less error %.

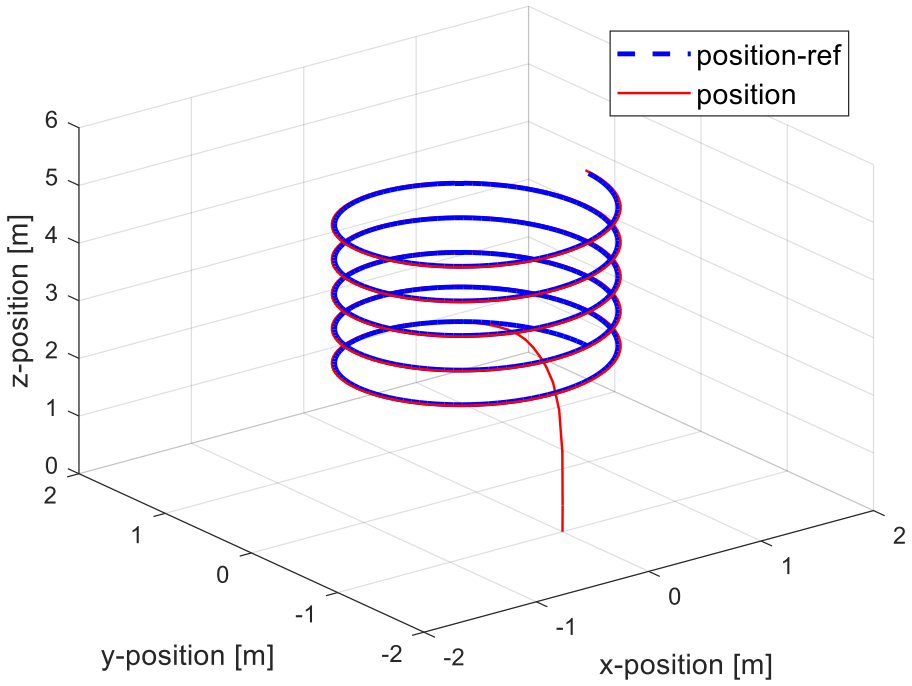


Figure 4.2 Spiral trajectory

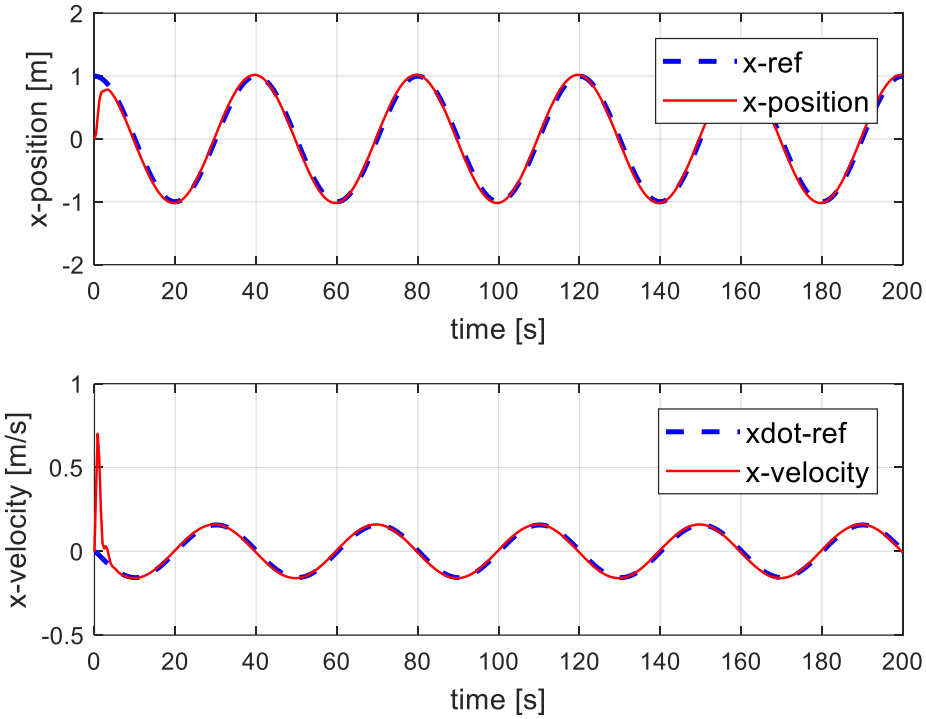


Figure 4.3 x and ẋ values as a function of time

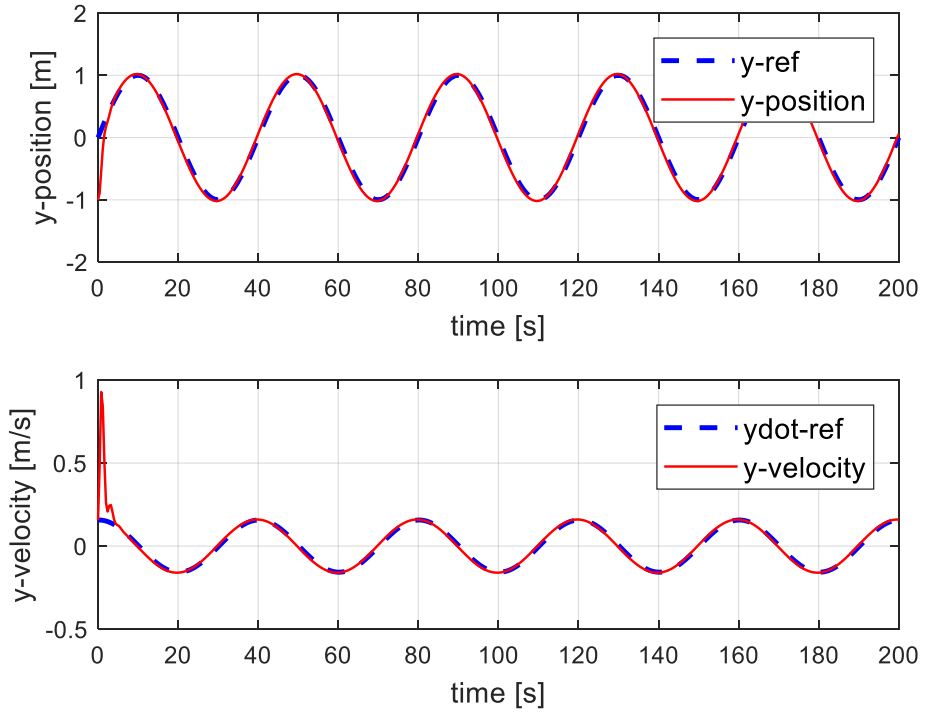


Figure 4.4 y and ẏ values as a function of time

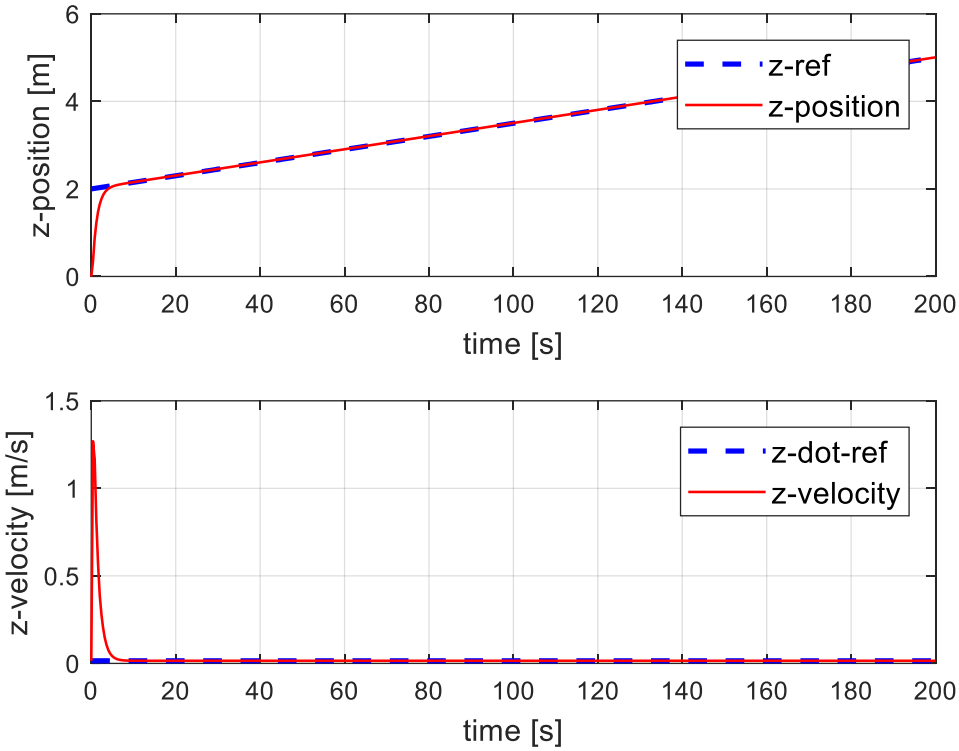


Figure 4.5 z and ż values as a function of time

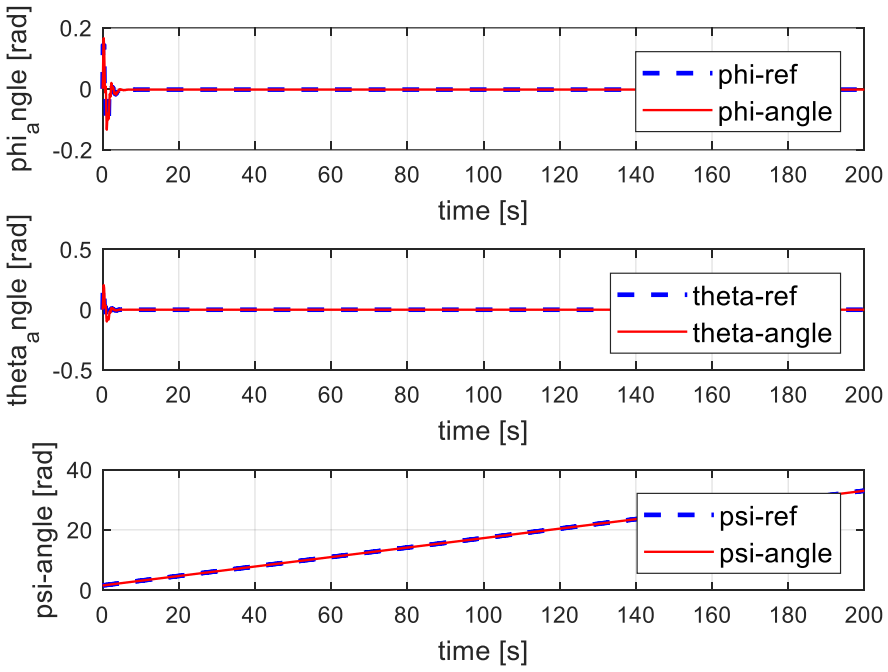


Figure 4.6 φ, θ, ψ values as a function of time

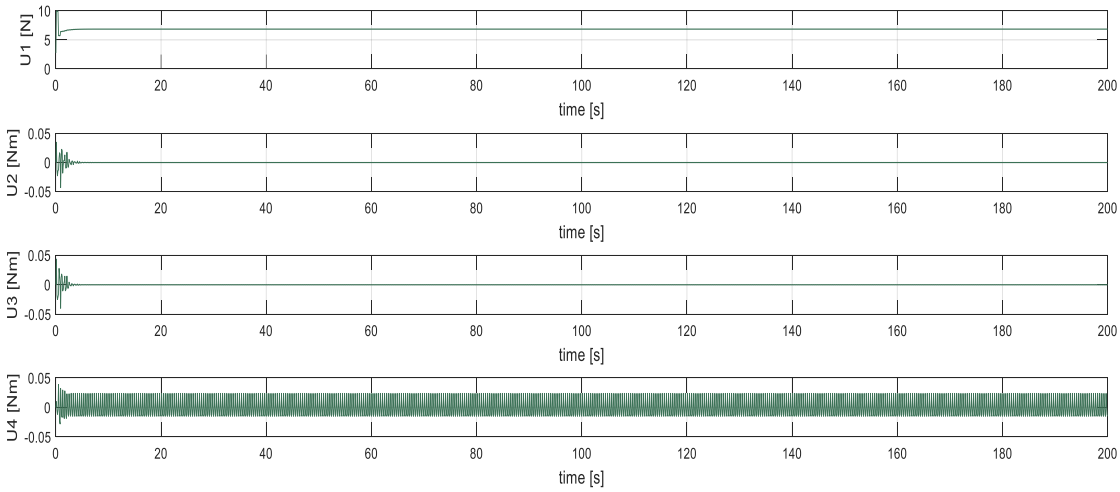


Figure 4.7 U1, U2, U3, U4 values as a function of time

It can be clearly seen from the Figures 4.2 to 4.7; the Quadcopter's six degrees of freedom are tracked with very small errors. In tracking the x, y, z reference velocity values, one can observe strong overshoot at the beginning of the test period. That can be explained by the fact that the quadcopter starts its journey from quite a long distance away from the trajectory. However, once it reaches the path that it needs to follow, the velocities of the quadcopter stabilize and track the reference values very smoothly.

Other trajectories considered in this thesis:

Straight line trajectory: After applying the MPC strategy, now it's time for the quadcopter to track the given straight-line trajectory.

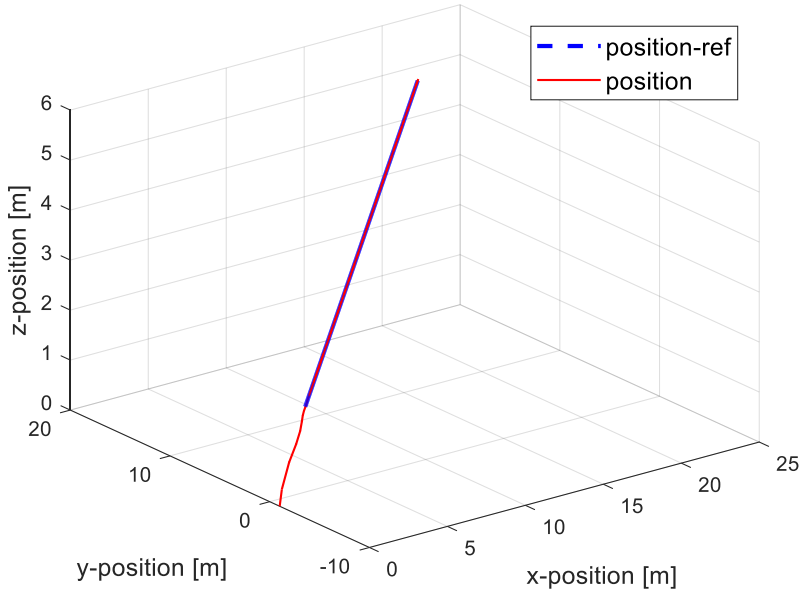


Figure 4.8 Straight line trajectory

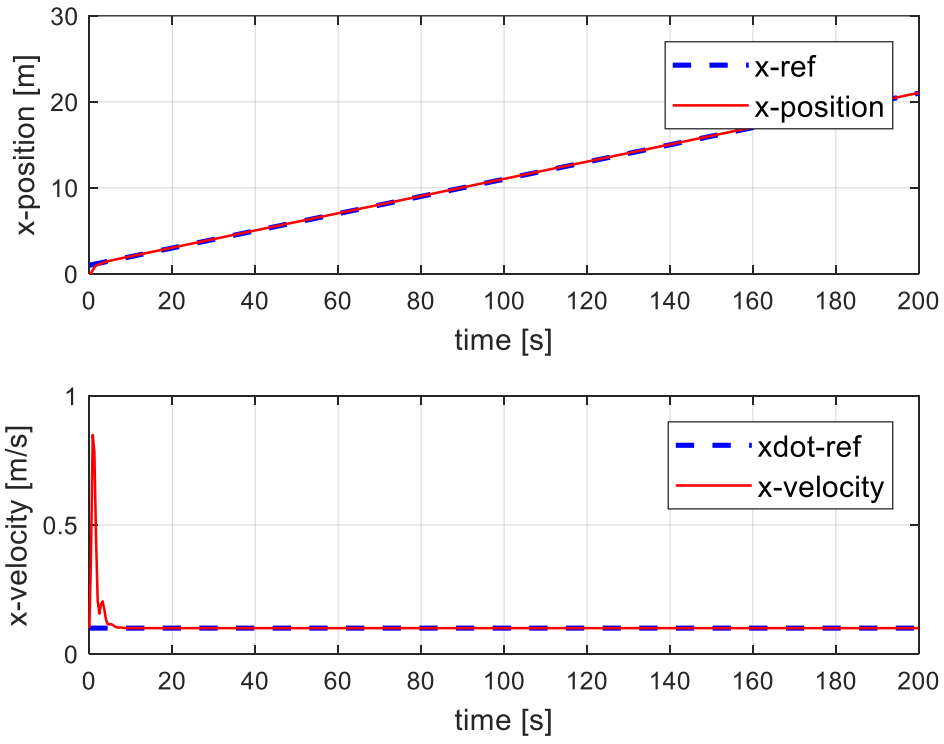


Figure 4.9 x and ẋ values as a function of time

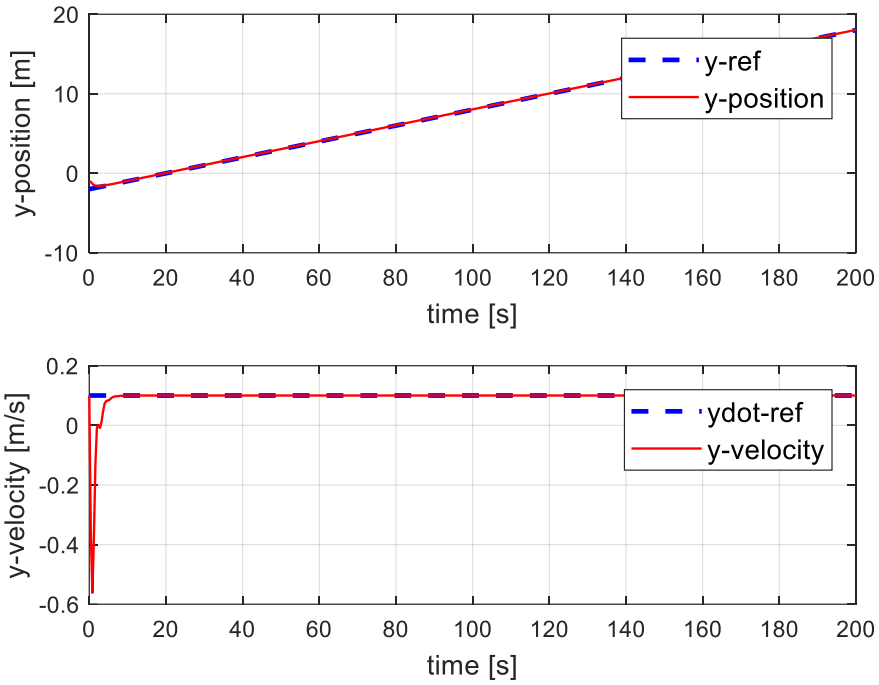


Figure 4.10 y and ẏ values as a function of time

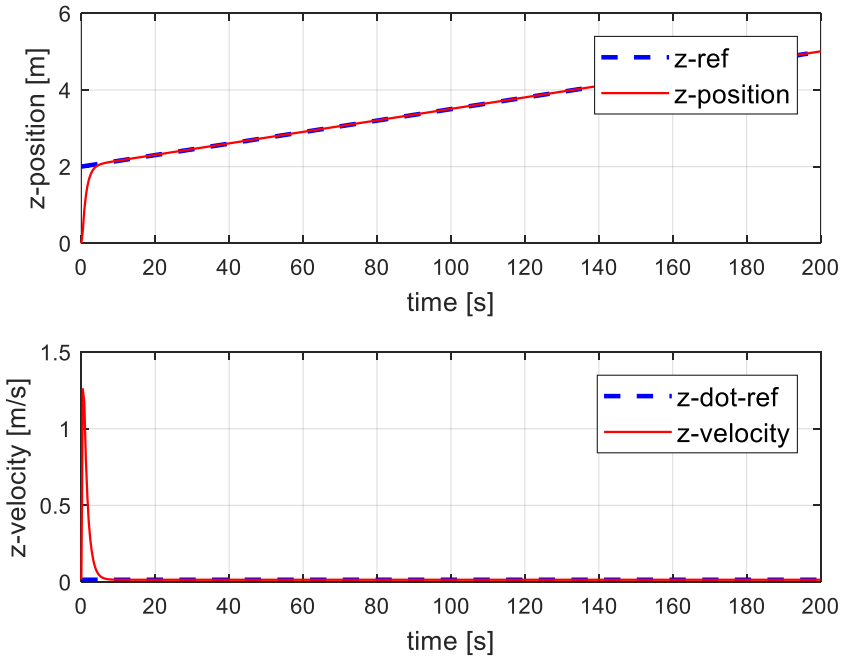


Figure 4.11 z and ż values as a function of time

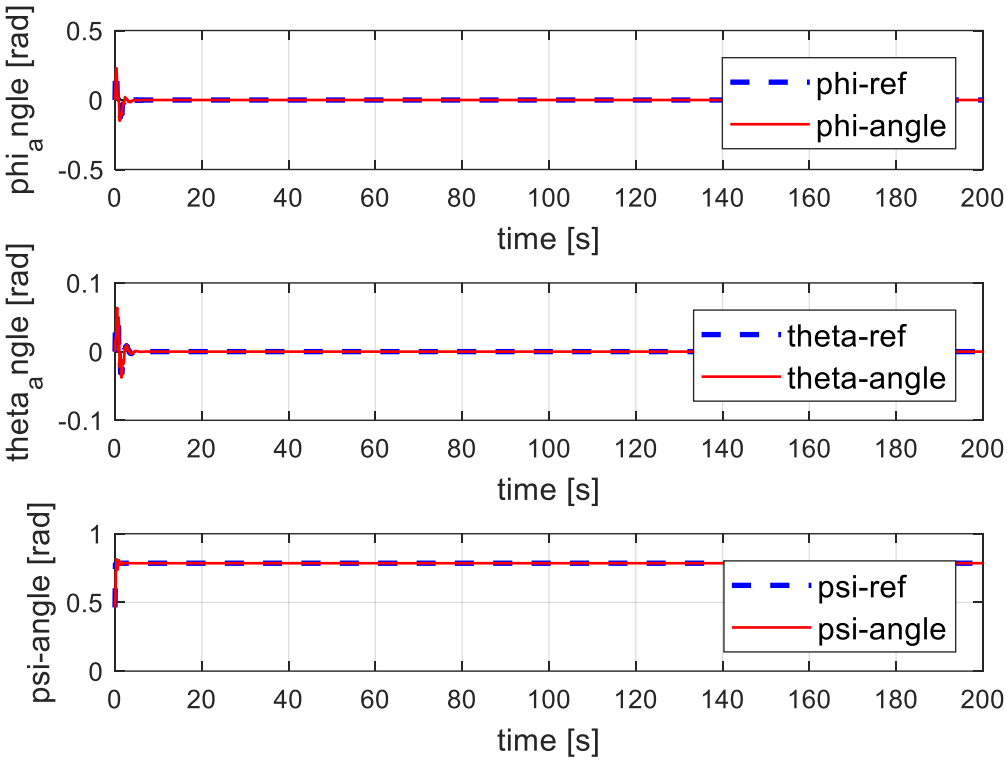


Figure 4.12 φ, θ, ψ values as a function of time

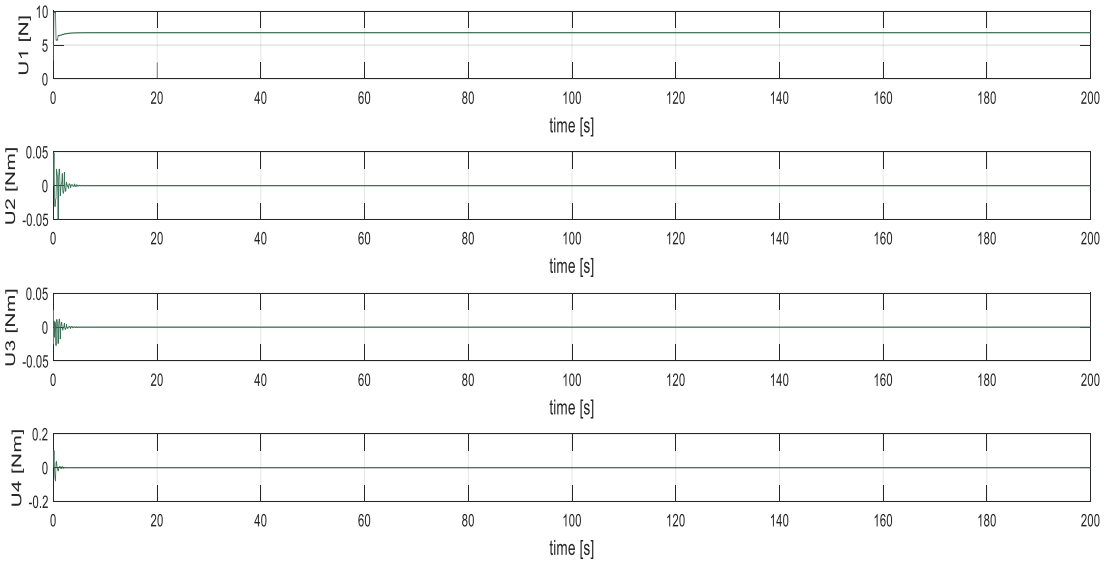


Figure 4.13 U1, U2, U3, U4 values as a function of time

Trajectory with z = height_i+50*d_height/t(end)*sin(t);

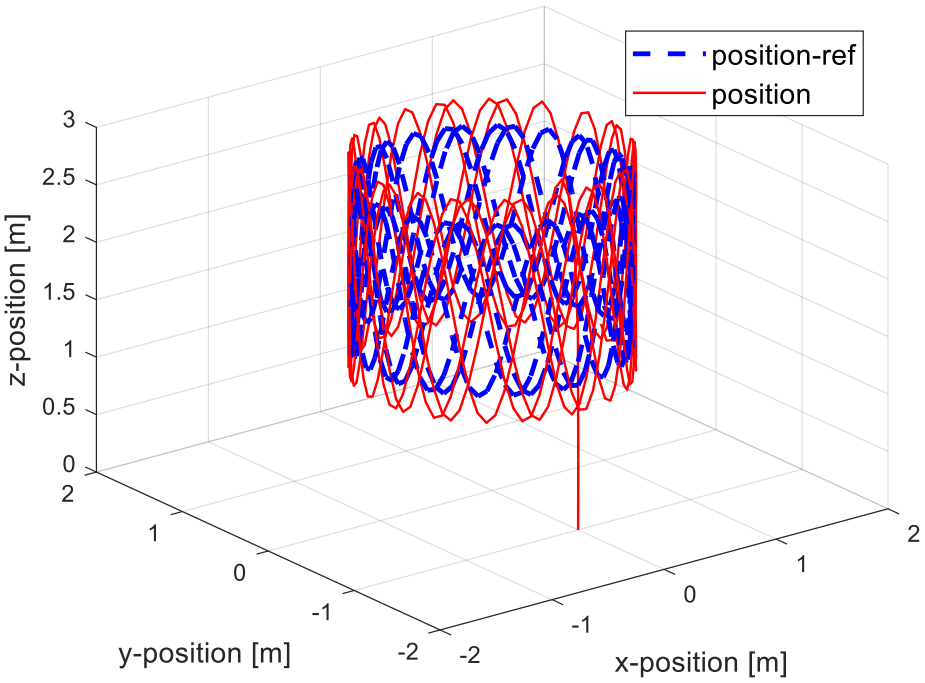


Figure 4.14 Quadcopter trajectory

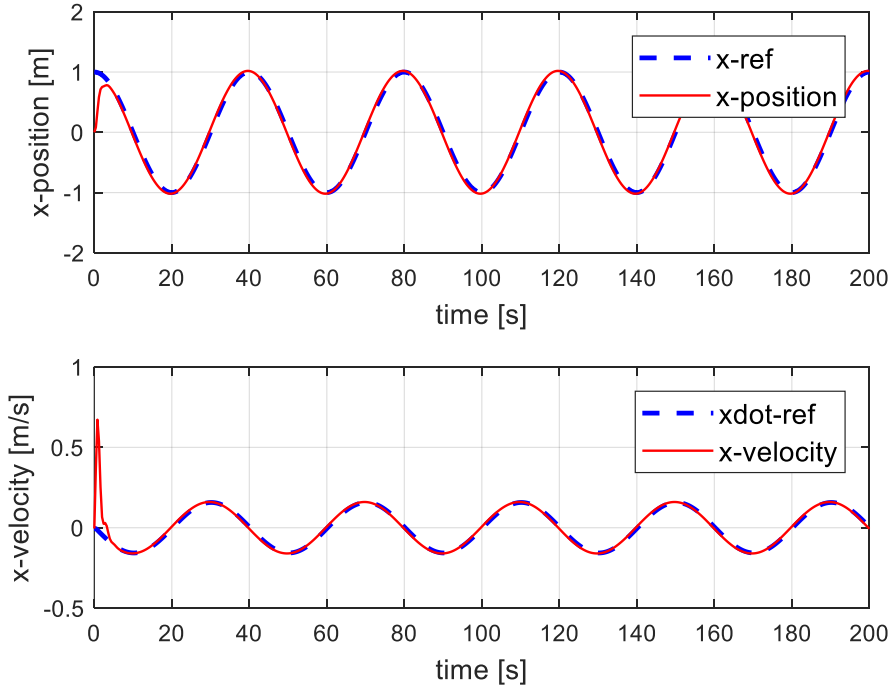


Figure 4.15 x and ẋ values as a function of time

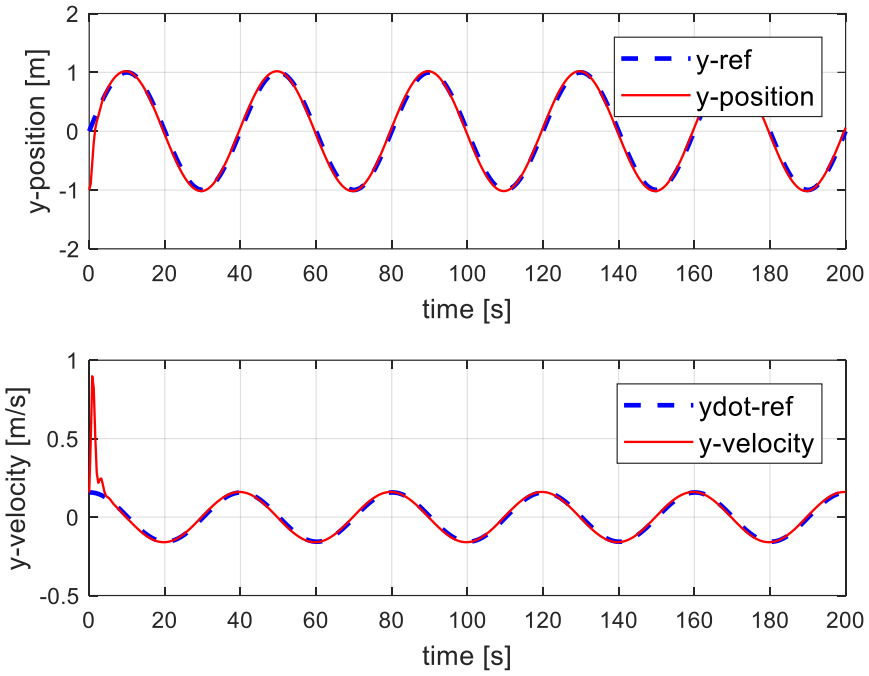


Figure 4.16 y and ẏ values as a function of time

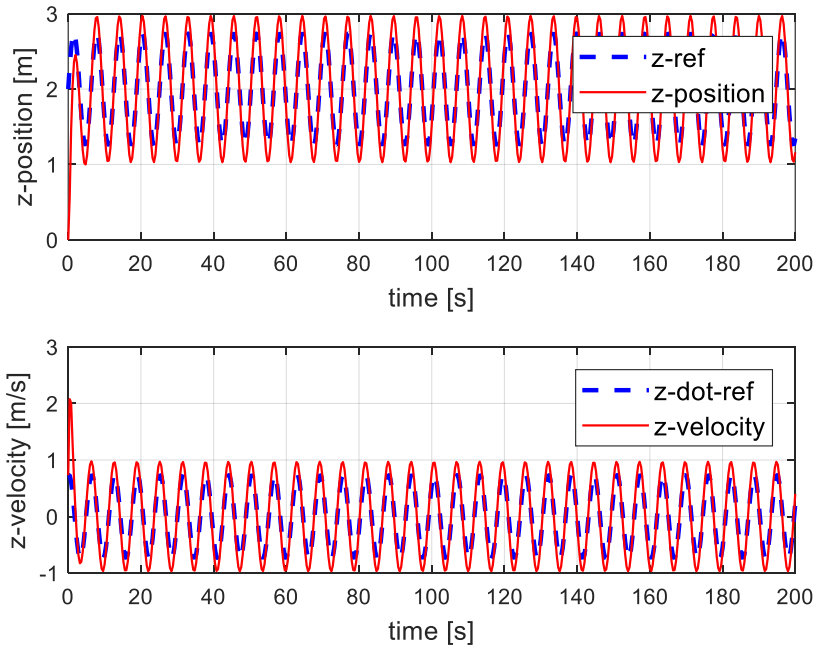


Figure 4.17 z and ż values as a function of time

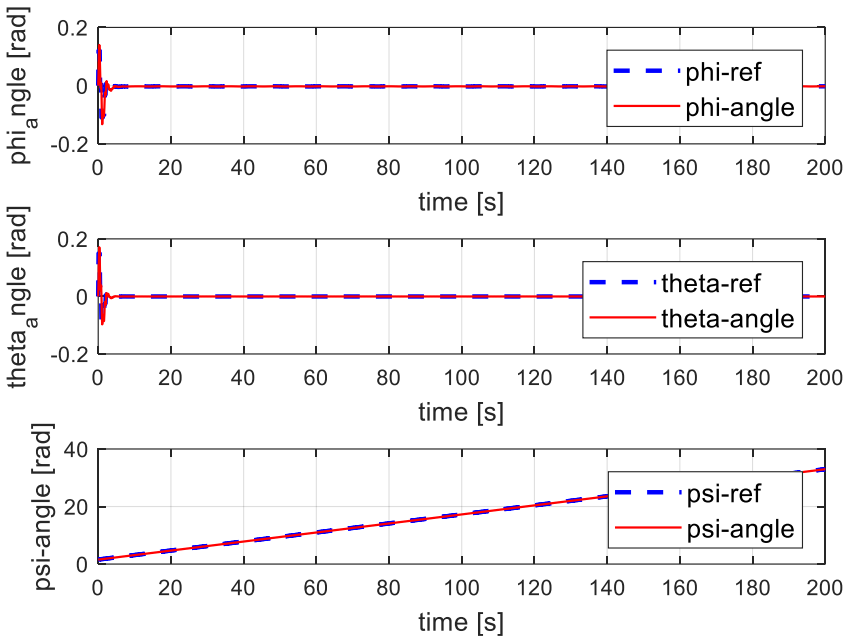


Figure 4.18 φ, θ, ψ values as a function of time

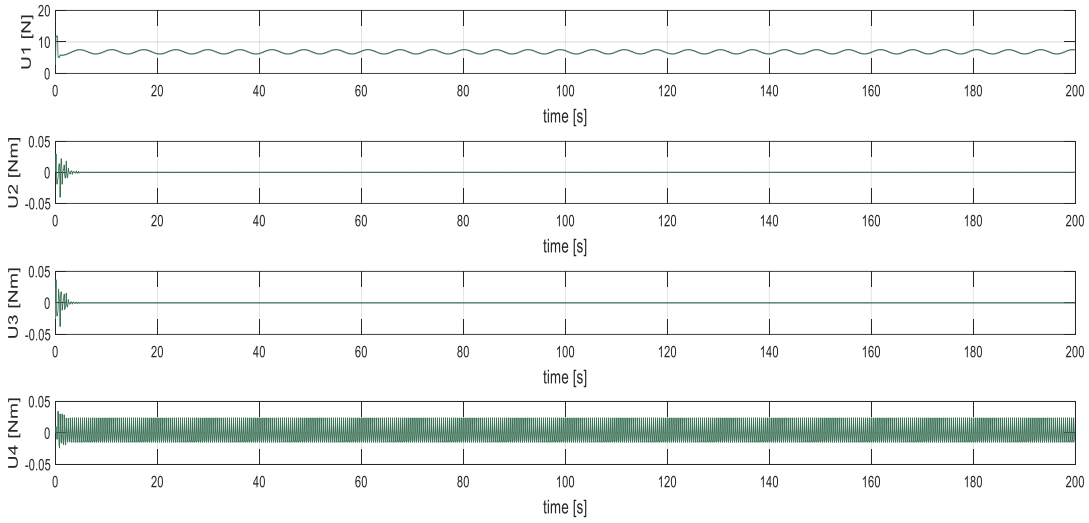


Figure 4.19 U1, U2, U3, U4 values as a function of time

Trajectory: Extended Spiral

```
x = r.*cos(alpha);
y = r.*sin(alpha);
z = height_i+50*d_height/t(end)*sin(t);
```

In tracking the x, y, z reference velocity values, strong overshoot at the beginning of the test period is observed. This is because the quadcopter starts its journey from quite a long distance away from the trajectory. However, once it reaches the path that it needs to follow, the velocities of the quadcopter stabilize and track the reference values very smoothly.

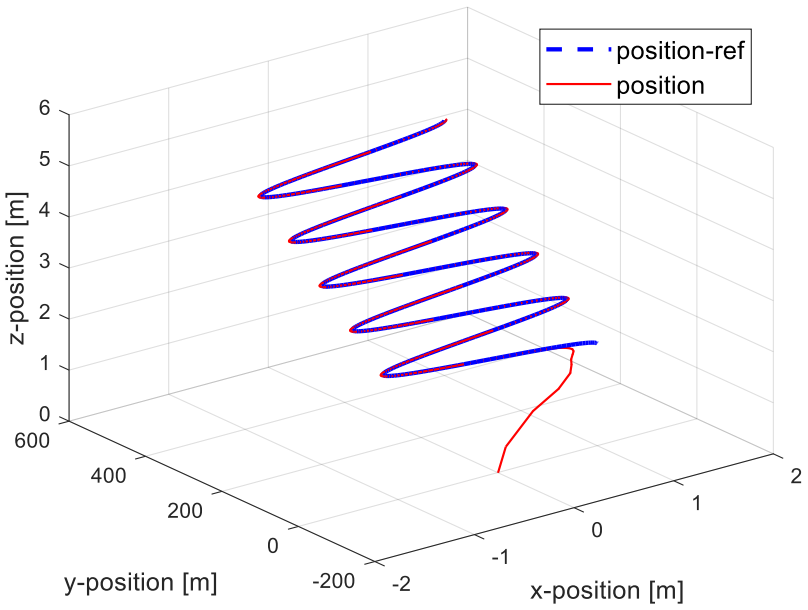


Figure 4.20 Quadcopter trajectory

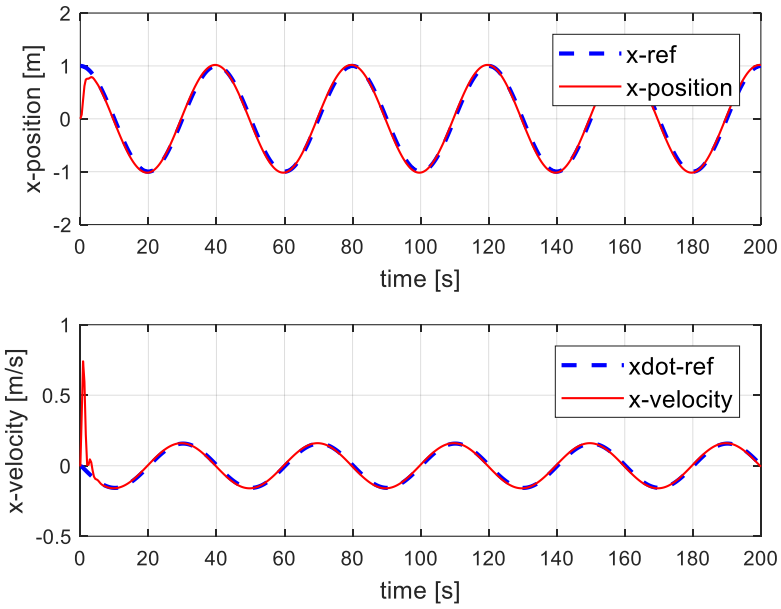


Figure 4.21 x and ẋ values as a function of time

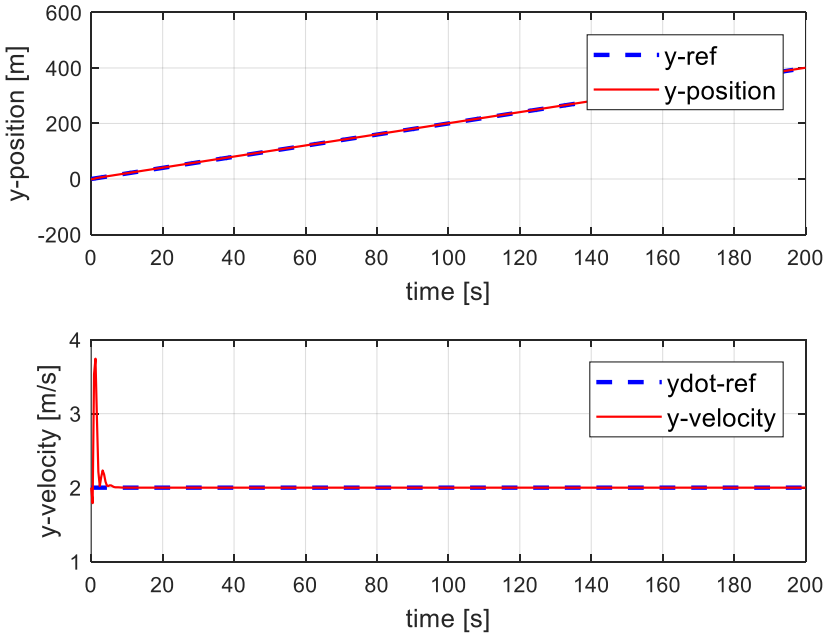


Figure 4.22 y and ẏ values as a function of time

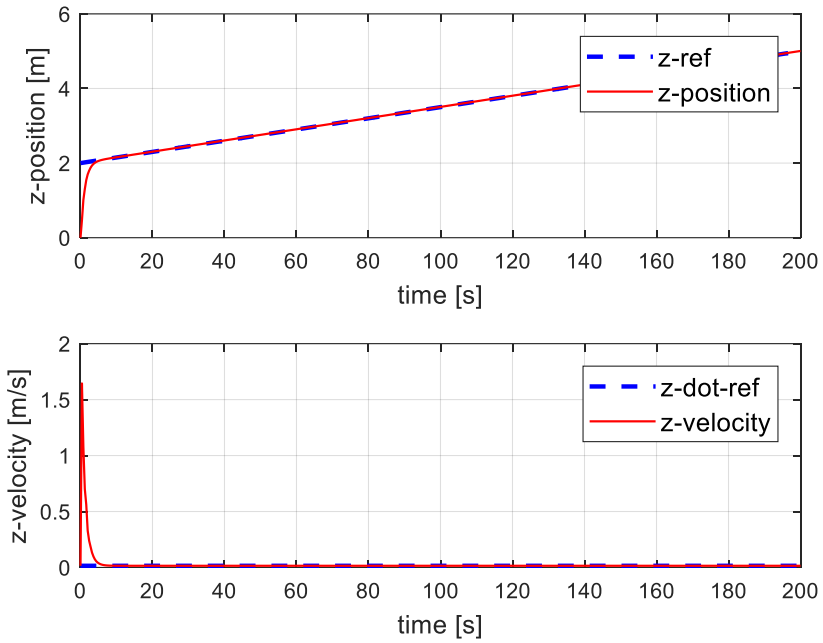


Figure 4.23 z and ż values as a function of time

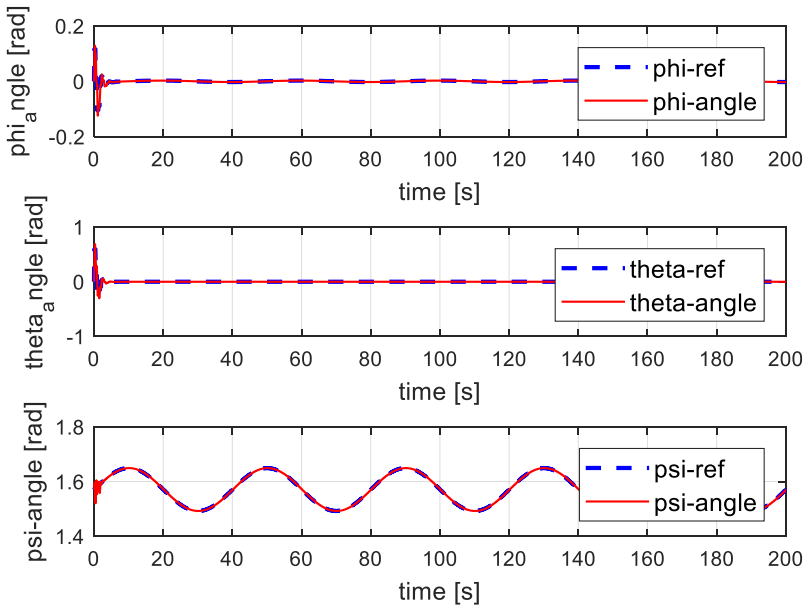
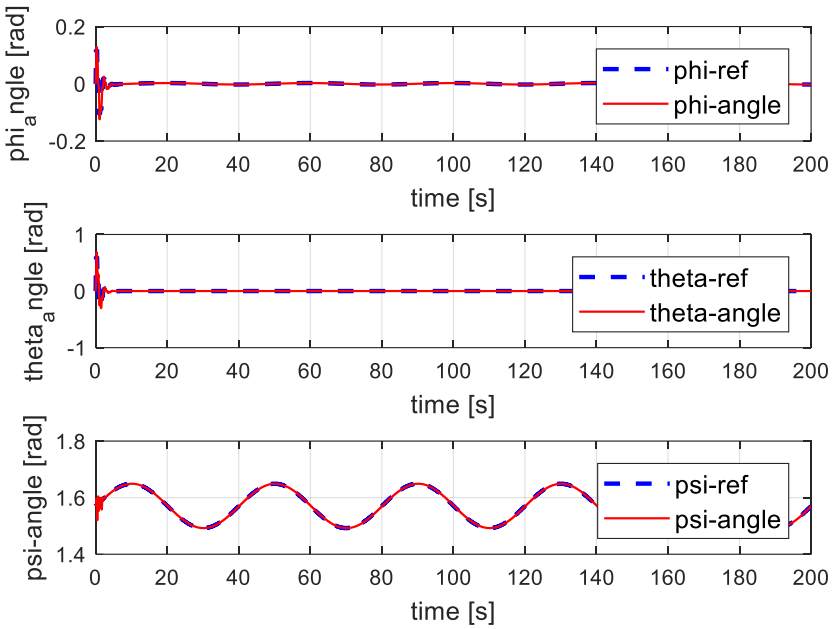


Figure 4.24 φ, θ, ψ values as a function of time



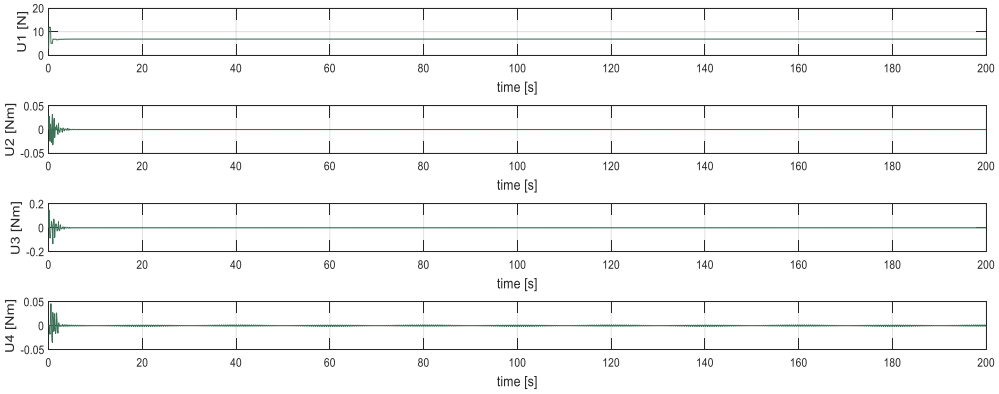


Figure 4.25 U1, U2, U3, U4 values as a function of time

4.3. Summary of the results

From the above figures, it can be seen that the proposed control strategy has a very high success as all the Quadcopter's six degrees of freedom are tracked with very small errors. In tracking the x, y, z reference velocity values, one can observe strong overshoot at the beginning of the test period. That can be explained by the fact that the quadcopter starts its journey from quite a long distance away from the trajectory. However, once it reaches the path that it needs to follow, the velocities of the quadcopter stabilize and track the reference values very smoothly.

This success would not be possible if only a global LPV-MPC control strategy was used because in that case the controller would fail to track the trajectory. The quadcopter will only oscillate up and down along the z-axis. This can be explained by the fact that the angles φ and θ might remain unaffected. Reason being, that the angles could not be separated from highly nonlinear equations of motion and represented as inputs in the LPV model. So, the control strategy would not be able to generate reference values for the angles. That is why the best way to do this would be to decouple the position and angular variables. Hence, a position controller with feedback linearization method was used for controlling the position variables, and the attitude controller controlled the angles using the LPV-MPC control strategy.

Important point to be noted here is, if the test period is very long and the structure of the expanding spiral remains the same i.e., the same initial and final height, and the continuously increasing radius along with the same sample time, then, there is a possibility that the quadcopter becomes unstable at some point.

Chapter 5

Quadcopters Impact on the World

5.1. Impact on the Environment

Automatons have just had a major effect in a few fields. They have been utilized for insight gathering, in filmmaking, and the Christmas season's must-have toy. In any case, automatons can do, and be, quite a lot more. Here are five significant ways drones are changing the standards with regard to ensuring the earth [31].

5.1.1. Transportation and Delivery

We've all found out about Amazon needing to enrol drones so as to convey its items directly to clients' entryways without the requirement for trucks and drivers. That seems like an extraordinary thought, particularly for those of us who had prefer to get our stuff as fast as could be expected under the circumstances, yet what amount will it truly eliminate the ecological costs we are taking a gander at the present moment?
That is a troublesome inquiry to answer totally. Since, as the Federal Highway Administration brings up in this investigation, the natural expenses of transportation are spread more than a few territories. The significant regions are rail, interstate, and air. While air transportation is the littlest zone as far as to request, the impacts of emanations from planes (particularly nitrogen oxides, generally alluded to as Nox) disproportionately affect the earth since they're discharged so high in the environment. So how precisely will drones change the transportation business? In the halfway term, automatons will be utilized to rapidly convey items that can be shipped between short separations. Consider the normal separation for pizza conveyance, for instance. These conveyances will before long be made not by fumes hacking vehicles and trucks, yet by battery-controlled automatons. This will eliminate the measure of littler street conveyances, and it will mean there will be fewer trucks out and about. In this way, as opposed to wiping out the contamination from air load, automatons will cut into the contamination brought about by trucks. Automatons do have moderately little payloads, however, which confines their conveying limit. It makes them perfect for little conveyances, or for dropping off crisis supplies, yet not for the mass vehicle.
Obviously, as the innovation develops, automatons will have the option to convey bigger burdens over more prominent separations. This could have an enormous effect, given that automatons will in general sudden spike in demand for battery power. On the off chance that those batteries are accused of efficient power vitality, the considerable measure of carbon produced from business and mechanical transportation will be fundamentally diminished. This decrease will go far toward battling the quick development of environmental change [32].

5.1.2. Natural life Conservation

Automatons can do a ton to help save our regular spaces. Automatons can screen huge areas of domain easily, which implies there's less effect on the region since no individuals need to make the trek. Far and away superior, drones give an elevated bird's-eye see, which means gives that would not be noticeable from the beginning be promptly obvious.
Automatons can be utilized to follow creatures, especially hazardous creatures, without putting anybody in danger. They can likewise be utilized to look for poachers and trespassers, expanding security in territories where there is essentially an excessive amount of ground to cover. Furthermore, automatons can be utilized to give help when catastrophic events strike. Regardless of whether it is to brush a zone after a seismic tremor or flood to search for survivors, or battling fires by conveying payloads, drones are an integral asset in the battle to keep natural life, and wild territories, safe [32].

5.1.3. Observation and Inspection of Solar Panels, Wind Turbines and Oil Pipelines

It is insufficient to simply assemble things like breeze turbines, oil pipelines, sunlight-based boards, or high-voltage electrical cables. When those things have been fabricated, they must be observed and assessed with a reasonable piece of consistency. In any event, on the off chance that you need to stay away from major issues, and breakdowns that can have deplorable results.
The issue we have needed to manage up to this point is that checking and examining these structures requires significant investment and labour. Getting somebody to move to the highest point of a breeze turbine, and afterward to examine it, is a risky undertaking. It is additionally something that can take a great deal of time. The equivalent is genuine with regards to strolling the full length of an oil pipeline or getting up close to electrical cables to review them for fraying or harm. The activity despite everything should be done, however, with drones, it tends to be done securely, and in a small amount of the time.
There is no compelling reason to send overwhelming trucks, and to spend incalculable worker hours in the audit when an automaton can skip along a line, or fly right dependent upon an instrument, to see whether it needs work. At the point when it does, real individuals will even now need to come to do it, yet the sheer measure of time and exertion automatons can spare is somewhat stunning [32].

5.1.4. Supportable Agriculture and Crop Monitoring

As Gizmodo called attention to, drones are a major help to ranchers who need to attempt to take a gander at the master plan with regards to their homesteads. Since customary ranches will in general spread so much space, it is difficult to get a feeling of what is moving on from the beginning. Aeronautical looking over administrations can assist ranchers with making sense of what's new with their fields, and it can give maps that would then be able to be utilized when making arrangements for the following developing season.
There are numerous new programming instruments that go with automatons to help with NDVI mapping. NDVI (Normalized Difference Vegetation Index) is a straightforward graphical marker that can be utilized to dissect remote detecting estimations to evaluate whether the objective being watched contains live green

vegetation or not. Solid vegetation (or chlorophyll) reflects progressively close infrared (NIR) and green light contrasted with different frequencies. This information is caught utilizing drone cameras and brought into an application, for example, Drone Deploy to make graphical portrayals. Ranchers and agrarian laborers can then basically get to the guide to decide the general wellbeing of vegetation on a given homestead and direct specialists to the yields requiring the most consideration and care.
Also, automatons can be utilized to convey significant materials to crops. Rather than utilizing planes for crop cleaning, for example, automatons can make short rushes to desert the essential added substances. They are little, quick, and adaptable, which permits them to satisfy various occupations in any farming arrangement [31] [32].

5.1.5. Land Management

Perhaps the greatest battle individuals have needed to manage is being restricted in our viewpoint with regards to land. Airborne studying is sensitive craftsmanship, and it is hard to envision the completed item with regards to building or to modify the land. Automatons can make this whole procedure that a lot simpler. When seen from the air, a completed item gets simpler to picture. Issue territories can be immediately recognized, and an automaton can be utilized to make 3D maps and to record GPS facilitates for simple following. There is no compelling reason to send a group nearby when a significant part of the work should be possible from the air. Particularly when automatons can deal with huge numbers of the assignments all the more rapidly, on account of onboard PCs and programming [32].

5.2. Impact on the society

Ventures have to a great extent been utilizing drones in their activities – from oil organizations that need to screen their pipelines to realtors who need to make elevated efforts of a property, and individuals in media outlets.

Automaton coach and pilot Mahmood Hussein says that the prizes are incredible for individuals seeking after a profession as an automaton pilot. We are seeing six-figure pay rates that can go as high as $150,000 every year.
While it is normal to consider flight the area where the vast majority of the automaton exercises occur, there are different fields going to be upset by drones.
The huge issue of poaching perseveres, with 30,000 elephants butchered each year for their tusks. Rhinoceros experience the ill effects of a similar destiny – all to flexibly elements for deceptive malignant growth medicines. There are different innovations used to endeavour to stop poachers, however, none of them has been effective up until now. Automatons are insulted to be one of the innovations that will help control poachers as they can be utilized to float in dim wilderness territories utilizing an infrared camera [33] [34].

Wellbeing out and about can be troublesome and exorbitant to oversee. The better comprehension of streets and expressways will spare numerous lives, that is the reason state and government divisions of transportation (DOTs) presently use automatons to assist them with observing. Automatons are utilized to report the state of regions influenced by rock slides and flooding that may put drivers and travellers in danger. This will assist them with sending earlier alerts, so mishaps can stay away from [33].

5.3. Economic Impact

Fast mechanical advancement has furnished purchasers with front line items at moderate costs. Generally, drones had been constrained to military use because of significant expenses and specialized advancement. Be that as it may, because of economies of scale, shoppers can buy drones for under $100. With across the board get to, customer organizations, for example, Amazon has investigated the utilization of unmanned airborne vehicles for business purposes. Amazon Prime Air has guaranteed a 30-minute conveyance administration for bundles of up to 5 lbs. Google (GOOG), in benevolent complexity to Amazon, has created airborne automatons for natural protection and conveyance of medication to remote locations. The natural effect is additionally tremendous. Fuelled by batteries, drones are more ecologically agreeable than conveyance trucks. On the off chance that conveyance drones increase far-reaching use, this would decrease the dependence on vehicles for some organizations. This would adversely affect vehicle makers, yet the effect on the earth would be a shelter and would enable numerous nations to lessen discharges, helping meet emanations targets set in different worldwide understandings. The financial ramifications of business drone utilization are evident. The market for drones is assessed to be $127 billion over an assortment of enterprises. Quiet, the business utilization of automatons will transcendently influence horticulture and foundation more so than trade. Because of the capacity to cover enormous territories, drone use in agribusiness is foreseen to successfully take care of and hydrate plants while likewise constraining introduction to maladies [31] [34].

On a macroeconomic scale, the coordination of UAVs could make in excess of 100,000 occupations. Over a 10-year length, work creation from business drone use will comprise fundamentally of assembling occupations and automaton administrators.

The suggestions plainly positively affect organizations and purchasers. Purchasers straightforwardly advantage from work creation, bringing about extra income. Business automatons will likewise permit ventures to acknowledge investment funds from practical methods for the stock, transportation, and dissemination. These cost investment funds can be passed down to the shopper through a decrease in costs [31].

Chapter 6

Summary and Conclusions

The main objective of this Master thesis was to apply Model predictive control (MPC) on a quadcopter using linear parameter varying (LPV) techniques. In order to do so the non-linear mathematical model of the quadcopter was put in the Linear Parameter Varying (LPV) form in order to be able to use the most basic Model Predictive Control (MPC) strategy, developed for linear systems. After applying the MPC strategy, the aim was to make the quadcopter track a given trajectory. Different trajectories were tested and validated. Smooth trajectory is preferable for this quadcopter. The thesis was divided into several chapters to achieve the final objective. Initially, the most important step was to define the coordinate frames that are used to control the quadcopter and to establish the mathematical model of a quadcopter. Once the mathematical model of the quadcopter is developed, the next step was to design the controller. The controller was split into two sub controllers. One controller is responsible for the position variables x, y, z. This position controller controls the position variables by using the state feedback linearization method. Moreover, the attitude controller was used to control the angles using the LPV-MPC control strategy.

The proposed strategy turned out to be a success in controlling the quadcopter. All the Quadcopter's six degrees of freedom are tracked with very small errors. In tracking the x, y, z reference velocity values, one can observe strong overshoot at the beginning of the test period. That can be explained by the fact that the quadcopter starts its journey from quite a long distance away from the trajectory. However, once it reaches the path that it needs to follow, the velocities of the quadcopter stabilize and track the reference values very smoothly. Everything was done keeping in mind that the LPV-MPC controller needs time to push the state angles towards its reference values. Therefore, the attitude controller must work at a higher frequency compared to the position controller

Chapter 7

Proposed future work

Robust control can be done for this quadcopter as a proposed future work. The simplest and most effective way to do robust control for the quadcopter is by introducing uncertainty inside the prediction horizon and then generating the updated bounds.

7.1. Zonotope-Tube-based LPV-MPC

The results of this thesis are very promising. Though, a lot of more work is needed as this thesis work did not apply any constraints on the inputs, outputs, nor on the states of the quadcopter. The parameters of the drone used in this thesis belong to AscTec Hummingbird. This thesis does not deal with any unexpected disturbance or uncertainty. Also, it does not take into account the noise that occurs in the sensors which in the end adds uncertainty in the states or outputs.

7.1.1. LQR Vs H_∞:

Linear Quadratic Regulator (LQR) is one of the most used techniques when there is a need to determine a local control structure for robustifying the MPC strategy using the tube-based approach. But it has been noticed, when dealing with systems subject to external disturbances, the LQR method becomes less efficient against such system variations than the proposed H_∞ strategy.

7.1.2. Online T-LPV-MPC using zonotopes

It is done by computing a polytopic state feedback controller offline using a H_∞-LMI based problem and computing the state feedback gain online, as a linear function of the scheduling vector.

The primary aim could be to design a local controller to reduce the mismatch between the states of the system and the nominal state vectors, even under the presence of exogeneous disturbances and model uncertainty.
A polytopic LPV H_∞ controller could be designed offline by means of minimizing the infinity norm of the transfer function between the disturbance signal and the control variables. Primarily, the proposed problem could be to find a polytopic state feedback gain.
Choosing H_∞ method because more degrees of freedom to include additional performance weights and better attenuate unknown inputs (disturbance and noise).
Online computation: At each control iteration K, the state feedback LPV control gain should be updated based on the current value of the scheduling vector. Also, terminal cost and terminal constraint should be incorporated.

Important point to note here is that when considering larger disturbances acting over the quadcopter implies bounding the differences between the real and the nominal system in a large zonotope which will lead to a more conservative scenario and also to the reduction of the maximum prediction horizon in the MPC design.

7.2. State Initialization problem

State initialization approach can also be used where Kalman filter could be added to the control loop to make the tracking more robust. In this thesis, weight matrices Q, S, R in MPC are taken as identity matrices. They can be properly tuned. Results could be further tested on a real quadcopter.

Bibliography

[1] L. Dormehl, “Urbanismnext.org,” University of Oregon, 2018. [Online]. Available: https://www.urbanismnext.org/resources/the-history-of-drones-in-10-milestones.

[2] Simanaitis, “simanaitissays.com,” 27 may 2020. [Online]. Available: https://simanaitissays.com/2020/05/27/louis-breguet-a-man-ahead-of-his-time-part-1/.

[3] “KROSSBLADE AEROSPACE,” [Online]. Available: https://www.krossblade.com/history-of-quadcopters-and-multirotors.

[4] C. P. P. Shana Priwer, “Later Copters,” in *The Everything Da Vinci Book*, Adams Media, 2006.

[5] W. R. Young, “The Helicopters,” in *The Epic of Flight*, Chicago, Time-Life Books, 1982.

[6] “Helicopter rotor system”. Patent US3261407, 13 March 2015.

[7] E. Darack, “airspacemag.com,” 19 May 2019. [Online]. Available: https://www.airspacemag.com/daily-planet/brief-history-quadrotors-180963372/.

[8] A. G. G. a. R. M. B. Rao, “The societal impact of commercial drones,” *Technol. Soc., NYU Scholars,* vol. 45, pp. 83-90, 2016.

[9] T. T. a. O. A. N. Hossein Motlagh, “Low-Altitude Unmanned Aerial Vehicle-Based Internet of Thing Services: Comprehensive Survey and Future Perspectives,” *IEEE Internet Things J,* vol. 3, pp. 899-922, 2016.

[10] X. Lin, “Mobile Netwroks Connected Drones: Field Trials, Simulations, and Design Insights,” 2018.

[11] E. Commission, “Press release - Aviation: Commission is taking the European drone sector to new heights,” 2017.

[12] D. Joshi, “Drone Technology and Usuage: Current Uses and Future Drone Technology,” Business Insider, 2017. [Online]. [Accessed 30 April 2019].

[13] D. T. a. t. I. o. s. drones. [Online]. Available: https://airdronecraze.com/drone-yech/. [Accessed 30 April 2019].

[14] P. Kremonas, “EENA Awards Heroic Icelandic Drone Rescue - EENA,” 2018.

[15] V. M. a. N. N. I. Mademlis, “Overview of Drone Cinematography for Sports Filming,” *Researchgate,* pp. 3-4, 2018.

[16] B. P. L. P. a. P. L. Dakkak-Arnoux, “Digital Transformation Monitor Drones inagriculture,” January 2018.

[17] R. Smith, “Utilities Turn to Drone to Inspect Power Lines and Pipelines,” [Online]. Available: https://www.wsj.com/articles/utilities-turn-to-drones-to-inspect-power-lines-and-pipelines-1430881491.

[18] C. T. Fernandez, “CONTROL OF AN UAV USING LPV TECHNIQUES,” The Universitat Politecnica de Catalunya - ETSEIB, 2018.

[19] Charlie, “charlestyler.com,” [Online]. Available: https://charlestytler.com/quadcopter-equations-motion/.

[20] T. Bresciani, “Modelling, Identification and Control of a Quadcopter Helicopter,” Lund University, 2008.

[21] J. J. Craig, Introduction to Robotics. Mechanics and Control, PEARSON.

[22] T. A. a. S. Q. Badgwell, “A review of Nonlinear Model Predictive Control Applications, in Nonlinear Predictive Control: Theroy and Practice,” London, 2001.

[23] C. R. a. B. L. .. R. Cutler, “Dynamic Matrix Control - A Computer Control Algorithm,” Proc. Joint Auto. Control Conf., Paper WP5-B, San Francisco, 1980.

[24] U. A. V. Vsilyev A. S, “Modeling of dynamic systems with modulation by means of Kronecker vector-matrix representation,” *Scientific and Technical Journal of Information Technologies, Mechanics and Optics,* pp. 839-848, 2015.

[25] G. J. Balas, “Linear Parameter-Varying Control and its Application to Aerospace,” *ICAS 2002 CONGRESS.*

[26] F. Wu, “Control of linear Parameter Varying systems,” Uni. of California Berkeley, 1995.

[27] MATLAB®, “Quadratic programming @ONLINE,” [Online]. Available: https://es.mathworks.com/help/optim/ug/quadprog.html?lang=en.

[28] MATLAB®, “Understanding Model Predictive Control, Part 2: What is MPC?,” [Online]. Available: https://www.youtube.com/watch?v=cEWnixjNds$vl=en.

[29] T. C. T. U. i. P. (CTU), “Model predictive control (MPC) - regulation,” [Online]. Available: https://moodle.fel.cvut.cz/mod/page/view.php?id=72476.

[30] T. C. T. U. i. P. (CTU), “Discrete-time optimal control over a finite,” [Online]. Available: https://moodle.fel.cvut.cz/mod/page/view.php?id=72476.

[31] J. Drugan, “Drones a Source od Debate - and Economic Impact,” 5 February 2015. [Online]. Available: https://www.uschamberfoundation.org/blog/post/drones-source-debate-and-economic-impact/42600.

[32] Drone genuity, “Drones Will Help Save The Environment,” [Online]. Available: https://www.dronegenuity.com/ways-drones-help-save-the-environment/.

[33] F. Beldia, “Impact of Drones on Society - Business, Connections, Privacy,” 23 April 2018. [Online]. Available: https://www.boldbusiness.com/society/impact-of-drones-on-society/.

[34] M. J. B. a. M. W. N. M. Zimmermann J. F. Peters, “The environmental impact of li-ion batteries and the role of key parameters - a review,” Renewable and Sustainable Energy Reviews, 2017.

Annexes

The MATLAB Implementation script

A. Main file: main.m

```
clear;
close all;clc

%% Main file for controling the quadcopter

% The proposed control strategy has a position controller
with
% state feeback linearlization and an MPC-LPV controller
% The relevant function files to this main file are the
following:
% Defining_initial_constants.m
% position_controller.m
% Quadcopter_model.m
% LPV_Discretization.m
% MPC_simplification.m
% trajectory_generator.m

%% Load the constant values from the
Defining_initial_constants file
constants= Defining_initial_constants();
Ts=constants{7};
controlled_states=constants{14};
% number of controlled states
innerDyn_length=constants{18}; % Number of inner control
loop iterations

%% Generate the reference signals
t = 0:Ts*innerDyn_length:200;
t_angles=(0:Ts:t(end))';
r = 1;
f=0.025;
height_i=2;
height_f=5;
```

```
[X_ref  ,X_dot_ref  ,Y_ref  ,Y_dot_ref  ,Z_ref  ,Z_dot_ref
,psi_ref]=    trajectory_generator(t    ,r,f    ,height_i
,height_f);  %smooth  trajectories  are  good  for  this
quadcopter
plotl=length(t); % No. of outer control loop iterations

%% Load the initial state vector
ut=0;
vt=0;
wt=0;
pt=0;
qt=0;
rt=0;
xt=0;
%X_ref(1,2); % Initial translational position
yt=-1;
%Y_ref(1,2); % Initial translational position
zt=0;%Z_ref(1,2); % Initial translational position
phit=0; % Initial angular position

thetat=0; % Initial angular position
psit=psi_ref(1,2); % Initial angular position
states=[ut,vt  ,wt,pt,qt  ,rt  ,xt  ,yt  ,zt  ,phit  ,thetat  ,
psit];
states_total=states;
% Assume that first Phi_ref , Theta_ref , Psi_ref are equal
to the first
% phit , thetat , psit
ref_angles_total=[phit ,thetat , psit ];
velocityXYZ_total=[X_dot_ref(1,2)          ,Y_dot_ref(1,2)
,Z_dot_ref(1,2)];


%% Initial Quadcopter state
omega1=3000; % rad/s at t = -1s %enough angular velocity to
rotate
omega2=3000; % rad/s at t = -1s
omega3=3000; % rad/s at t = -1s
omega4=3000; % rad/s at t = -1s
ct = constants{11};
cq = constants{12};
l = constants{13};
U1=ct*(omega1^2+omega2^2+omega3^2+omega4^2); % Input at t
= -1s
U2=ct*l*(omega4^2-omega2^2); % Input at t = -1s
U3=ct*l*(omega3^2-omega1^2); % Input at t = -1s
```

```
U4=cq*(-omega1^2+omega2^2-omega3^2+omega4^2); % Input at t
= -1s

UTotal=[U1,U2,U3,U4];% 4 inputs


global omega_total
omega_total=-omega1+omega2-omega3+omega4;


%% Start the global controller
for i_global = 1:plotl-1
    %% Implementation of the position controller (state
feedback linearization)(outer loop)
    [phi_ref            ,            theta_ref           ,
U1]=position_controller(X_ref(i_global+1,2),
X_dot_ref(i_global+1,2),Y_ref(i_global+1,2),Y_dot_ref(i_gl
obal
+1,2),Z_ref(i_global+1,2),Z_dot_ref(i_global+1,2),psi_ref(
i_global+1,2),states);

    Phi_ref=phi_ref*ones(innerDyn_length+1,1);
    Theta_ref=theta_ref*ones(innerDyn_length+1,1);

Psi_ref=psi_ref(i_global+1,2)*ones(innerDyn_length+1,1);
    ref_angles_total=[ref_angles_total;Phi_ref(2:end)
Theta_ref(2:end ) Psi_ref(2:end)];
    %% Create the reference vector
    % Starting with the inner loop once the reference vector
is created

refSignals=zeros(length(Phi_ref(:,1))*controlled_states,1)
;
    %                                                Format:
refSignals=[Phi_ref;Theta_ref;Psi_ref;Phi_ref;  ...  etc]  x
inner
    % loop frequency per one set of position controller
outputs
    k_ref_local=1;
    for i = 1:controlled_states:length(refSignals)
        refSignals(i)=Phi_ref(k_ref_local ,1);
        refSignals(i+1)=Theta_ref(k_ref_local ,1);
        refSignals(i+2)=Psi_ref(k_ref_local ,1);
        k_ref_local=k_ref_local+1;
    end

    k_ref_local=1; % for reading reference signals
    hz = constants{15}; % horizon period
```

```
    for i =1:innerDyn_length
        %% Generate discrete LPV Ad, Bd, Cd, Dd matrices
        % Discretization of the system
        % Transforming continuos varables and functions
into discrete counterparts so that it is more suitable and
        % convinient for numerical evaluation and
implementation.
        [Ad, Bd, Cd, Dd, x_dot, y_dot, z_dot, phit , phi_dot
,      thetat      ,      theta_dot      ,      psit      ,
psi_dot]=LPV_Discretization(states);
        velocityXYZ_total=[velocityXYZ_total;[x_dot,
y_dot, z_dot]];
        %% Generating the current state and the reference
vector
        x_aug_t=[phit;phi_dot;thetat    ;theta_dot;    psit
;psi_dot;U2;U3;U4 ];
        k_ref_local=k_ref_local+controlled_states;
        % Start counting from the second sample period:
        %
r=refSignals(Phi_ref_2;Theta_ref_2;Psi_ref_2;Phi_ref_3
...) etc.
        if      k_ref_local+controlled_states*hz-1      <=
length(refSignals)

r=refSignals(k_ref_local:k_ref_local+controlled_states*hz-
1);
        else
            r=refSignals(k_ref_local:length(refSignals));
            hz=hz-1;
        end
        %% Generate simplification matrices for the cost
function
        [Hdb,Fdbt,Cdb,Adc]                                  =
MPC_simplification(Ad,Bd,Cd,Dd,hz);
        %Need to find a compromise (minimizing errors vs
controlling
        %inputs). In this case, minimizing inputs is more
important
        %The goal is to transform the generic cost
function(states and
        %input changes, ref values) to only having input
changes. Basically
        %getting rid of states
        %% Calling the optimizer (quadprog)
        % Using quadprog to use (du)s ie,. change of inputs.
Computing it
```

```
        % into linear set of equations and then computing
roll, pitch, yaw
        %      Cost      function      in      quadprog:
min(du)*1/2*du'Hdb*du+f 'du
        % f'=[x_t ' , r']*Fdbt
        ft=[x_aug_t' ,r']*Fdbt;
        % Hdb must be positive definite for the problem to
have finite minimum.
        % Check if matrix Hdb in the cost function is
positive definite .
        [~,p] = chol(Hdb);
        if p~=0
            disp('Hdb is NOT positive definite ');
        end



        % Call the solver
        options = optimset('Display' , 'off ');
        lb=constants{16};
        ub=constants{17};

        if ~issymmetric(Hdb)
        Hdb=(Hdb+Hdb')/2;
        end
        [du,fval]=quadprog(Hdb,ft ,[] ,[] ,[] ,[] ,[] ,[]
,[] ,options);
        % Update the real inputs
        U2=U2+du(1);
        U3=U3+du(2);
        U4=U4+du(3);
        UTotal=[UTotal;U1,U2,U3,U4];
        % Computing the new omegas based on the new U-s.
        U1C=U1/ct;
        U2C=U2/(ct*l);
        U3C=U3/(ct*l);
        U4C=U4/cq;


        omega4P2=(U1C+2*U2C+U4C)/4;
        omega3P2=(-U4C+2*omega4P2-U2C+U3C)/2;
        omega2P2=omega4P2-U2C;
        omega1P2=omega3P2-U3C;


        omega1=sqrt(omega1P2);
        omega2=sqrt(omega2P2);
        omega3=sqrt(omega3P2);
```

```
        omega4=sqrt(omega4P2);


        % Compute the total omega
        omega_total=-omega1+omega2-omega3+omega4;


        % Simulate the new states
        T = (Ts)*(i-1):(Ts)/30:Ts*(i-1)+(Ts);
        %Using Ode45 to integrate the non linear quadcopter
model
        [T,x]=ode45(@(t       ,x)       Quadcopter_model(t
,x,[U1,U2,U3,U4]) ,T , states);
        states=x(end,:) ;
        states_total=[states_total ; states ];
        imaginary_check=imag(states)~=0;
        imaginary_check_sum=sum(imaginary_check);
        if imaginary_check_sum~=0
            disp('Imaginary part exists , Looks like
something is wrong');
        end
    end
end


%% Plot the trajectory



% Trajectory
figure;
plot3(X_ref(:,2)   ,Y_ref(:,2)   ,Z_ref(:,2)   ,   '--b'   ,
'LineWidth' ,2)
hold on
plot3(states_total(1:innerDyn_length:end,7)
,states_total(1:                    innerDyn_length:end,8)
,states_total(1:innerDyn_length:end,9) , 'r' , 'LineWidth'
,1)
grid on;
xlabel('x-position [m]' , 'FontSize' ,12)
ylabel('y-position [m]' , 'FontSize' ,12)
zlabel('z-position [m]' , 'FontSize' ,12)
legend({'position-ref'   ,   'position'},   'Location'   ,
'northeast' , 'FontSize' ,12)


%% Plot the positions and velocities individually
```

```
% X and X_dot
figure;
subplot(2,1,1)
plot(t(1:plotl),X_ref(1:plotl ,2) , '--b' , 'LineWidth' ,2)
hold on
subplot(2,1,1)
plot(t(1:plotl),states_total(1:innerDyn_length:end,7)    ,
'r' , 'LineWidth' ,1)
grid on
xlabel('time [s]' , 'FontSize' ,12)
ylabel('x-position [m]' , 'FontSize' ,12)
legend({'x-ref' , 'x-position '}, 'Location' , 'northeast'
, 'FontSize' ,12)
subplot(2,1,2)
plot(t(1:plotl),X_dot_ref(1:plotl ,2) , '--b' , 'LineWidth'
,2)
hold on
subplot(2,1,2)
plot(t(1:plotl),velocityXYZ_total(1:innerDyn_length:end,1)
, 'r' , 'LineWidth' ,1)
grid on
xlabel('time [s]' , 'FontSize' ,12)
ylabel('x-velocity [m/s]' , 'FontSize' ,12)
legend({'xdot-ref  '  ,  'x-velocity  '},  'Location'  ,
'northeast' , 'FontSize' ,12)



% Y and Y_dot
figure;
subplot(2,1,1)
plot(t(1:plotl),Y_ref(1:plotl ,2) , '--b' , 'LineWidth' ,2)
hold on
subplot(2,1,1)
plot(t(1:plotl),states_total(1:innerDyn_length:end,8)    ,
'r' , 'LineWidth' ,1)
grid on
xlabel('time [s]' , 'FontSize' ,12)
ylabel('y-position [m]' , 'FontSize' ,12)
legend({'y-ref ' , 'y-position '}, 'Location' , 'northeast'
, 'FontSize' ,12)
subplot(2,1,2)
plot(t(1:plotl),Y_dot_ref(1:plotl ,2) , '--b' , 'LineWidth'
,2)
hold on
```

```
subplot(2,1,2)
plot(t(1:plotl),velocityXYZ_total(1:innerDyn_length:end,2)
, 'r' , 'LineWidth' ,1)
grid on
xlabel('time [s]' , 'FontSize' ,12)
ylabel('y-velocity [m/s]' , 'FontSize' ,12)
legend({'ydot-ref  '  ,  'y-velocity  '},  'Location'  ,
'northeast' , 'FontSize' ,12)



% Z and Z_dot
figure;
subplot(2,1,1)
plot(t(1:plotl),Z_ref(1:plotl ,2) , '--b' , 'LineWidth' ,2)
hold on
subplot(2,1,1)
plot(t(1:plotl),states_total(1:innerDyn_length:end,9)    ,
'r' , 'LineWidth' ,1)
grid on
xlabel('time [s]' , 'FontSize' ,12)
ylabel('z-position [m]' , 'FontSize' ,12)
legend({'z-ref ' , 'z-position '}, 'Location' , 'northeast'
, 'FontSize' ,12)
subplot(2,1,2)
plot(t(1:plotl),Z_dot_ref(1:plotl ,2) , '--b' , 'LineWidth'
,2)
hold on
subplot(2,1,2)
plot(t(1:plotl),velocityXYZ_total(1:innerDyn_length:end,3)
, 'r' , 'LineWidth' ,1)
grid on
xlabel('time [s]' , 'FontSize' ,12)
ylabel('z-velocity [m/s]' , 'FontSize' ,12)
legend({'z-dot-ref  '  ,  'z-velocity  '},  'Location'  ,
'northeast' , 'FontSize' ,12)



%% Plot angle Phi
% Phi
figure;
subplot(3,1,1)
plot(t_angles(1:length(ref_angles_total(:,1))),ref_angles_
total(:,1) , '--b' , 'LineWidth' ,2)
hold on
```

```
subplot(3,1,1)
plot(t_angles(1:length(states_total(:,10))),states_total(:
,10) , 'r' , 'LineWidth' ,1)
grid on
xlabel('time [s]' , 'FontSize' ,12)
ylabel('phi_angle [rad]' , 'FontSize' ,12)
legend({'phi-ref ' , 'phi-angle'}, 'Location' , 'northeast'
, 'FontSize' ,12)

%% Plot angle Theta
% Theta
subplot(3,1,2)
plot(t_angles(1:length(ref_angles_total(:,2))),ref_angles_
total(:,2) , '--b' , 'LineWidth' ,2)
hold on
subplot(3,1,2)
plot(t_angles(1:length(states_total(:,11))),states_total(:
,11) , 'r' , 'LineWidth' ,1)
grid on
xlabel('time [s]' , 'FontSize' ,12)
ylabel('theta_angle [rad]' , 'FontSize' ,12)
legend({'theta-ref  '  ,  'theta-angle'},  'Location'  ,
'northeast' , 'FontSize' ,12)


%% Plot angle psi
% Psi
subplot(3,1,3)
plot(t_angles(1:length(ref_angles_total(:,3))),ref_angles_
total(:,3) , '--b' , 'LineWidth' ,2)
hold on
subplot(3,1,3)
plot(t_angles(1:length(states_total(:,12))),states_total(:
,12) , 'r' , 'LineWidth' ,1)
grid on
xlabel('time [s]' , 'FontSize' ,12)
ylabel('psi-angle [rad]' , 'FontSize' ,12)
legend({'psi-ref ' , 'psi-angle'}, 'Location' , 'northeast'
, 'FontSize' ,12)



%% Plot the inputs

figure
subplot(4,1,1)
plot(t_angles(1:length(states_total(:,10))),UTotal(:,1))
grid on
```

```
xlabel('time [s]' , 'FontSize' ,12)
ylabel('U1 [N]' , 'FontSize' ,12)
subplot(4,1,2)
plot(t_angles(1:length(states_total(:,10))),UTotal(:,2))
grid on
xlabel('time [s]' , 'FontSize' ,12)
ylabel('U2 [Nm]' , 'FontSize' ,12)
subplot(4,1,3)
plot(t_angles(1:length(states_total(:,10))),UTotal(:,3))
grid on
xlabel('time [s]' , 'FontSize' ,12)
ylabel('U3 [Nm]' , 'FontSize' ,12)
subplot(4,1,4)
plot(t_angles(1:length(states_total(:,10))),UTotal(:,4))
grid on
xlabel('time [s]' , 'FontSize' ,12)
ylabel('U4 [Nm]' , 'FontSize' ,12)
%\\
%\\
```

A. Supporting file: Defining_initial_constants.m

```
%Defining constant values of the Quadcopter which is then
loaded in the main
%file

function constants= Defining_initial_constants()
% Constants
Ix = 0.0034; %kg*m^2
Iy = 0.0034; %kg*m^2
Iz = 0.006; %kg*m^2
m = 0.698; %kg
g = 9.81; %m/s^2
Jtp=1.302*10^(-6); %N*m*s^2=kg*m^2
Ts=0.1; %s

% Matrix weights for the cost function (They must be
diagonal)
Q=[1 0 0;0 1 0;0 0 1]; %Errors % weights for outputs (output
x output)/Sample time
S=[1 0 0;0 1 0;0 0 1]; % weights for the final horizon
outputs ( output x output)
R=[1 0 0;0 1 0;0 0 1]; %Weight matrix for the inputs
ct = 7.6184*10^(-8); %N*s^2
cq = 2.6839*10^(-9); %N*m^2
l = 0.171; %m;
```

```
controlled_states=3; %output
hz = 4;  % horizon period
% Input bounds:
lb=[-0.5; -0.5; -0.5];
ub=[0.5; 0.5; 0.5];
%Inner loop has to be faster than the outer loop
innerDyn_length=4;    % Number of inner control loop
iterations (which means the inner loop is 4 times faster
than the outer loop)
 px=[-1+0j -2+0j ];
 py=[-1+0j -2+0j ];
 pz=[-1+0j -2+0j ];
 constants={Ix  Iy  Iz  m  g  Jtp  Ts  Q  S  R  ct  cq  l
controlled_states hz lb ub innerDyn_length px py pz};
end
```

B. Supporting file: Position_controller.m

```
function        [Phi_ref        ,        Theta_ref        ,
U1]=position_controller(X_ref ,X_dot_ref , Y_ref ,Y_dot_ref
,Z_ref ,Z_dot_ref ,Psi_ref , states)
%% Load the constants from the Defining_initial_constants
file
constants= Defining_initial_constants();
m = constants{4}; %kg
g = constants{5}; %m/s^2

%% Assign the states
% States: [u,v,w,p,q,r,x,y,z,phi,theta ,psi] % 12 states (6
body frame and
% 6 inertial frame)

u = states(1);
v = states(2);
w = states(3);
x = states(7);
y = states(8);
z = states(9);
phi = states(10);
theta = states(11);
psi = states(12);
%    Rotational    matrix    that    relates    u,v,w    with
x_dot,y_dot,z_dot
R_matrix=[cos(theta)*cos(psi),
sin(phi)*sin(theta)*cos(psi)-cos(phi)* sin(psi), ...
```

```
cos(phi)*sin(theta)*cos(psi)+sin(phi)*sin(psi); ...
cos(theta)*sin(psi),
sin(phi)*sin(theta)*sin(psi)+cos(phi)*cos( psi), ...
cos(phi)*sin(theta)*sin(psi)-sin(phi)*cos(psi); ...
-sin(theta), sin(phi)*cos(theta), cos(phi)*cos(theta)];
x_dot=R_matrix(1,:)*[u;v;w];  %x_dot
y_dot=R_matrix(2,:)*[u;v;w];  %y_dot
z_dot=R_matrix(3,:)*[u;v;w];  %z_dot
%% Compute the errors
ex=X_ref-x;
ex_dot=X_dot_ref-x_dot;
ey=Y_ref-y;
ey_dot=Y_dot_ref-y_dot;
ez=Z_ref-z;
ez_dot=Z_dot_ref-z_dot;
%% Compute the the constants K1, K2, and the values vx, vy,
vz to stabilize the position subsystem
Ax=[0 1;0 0];
Bx=[0;1];
px=constants{19};
Kx=place(Ax,Bx,px);
ux=-Kx*[ex;ex_dot];
vx=-ux;
Ay=[0 1;0 0];
By=[0;1];
py=constants{20};
Ky=place(Ay,By,py);
uy=-Ky*[ey;ey_dot];
vy=-uy;
Az=[0 1;0 0];
Bz=[0;1];
pz=constants{21};
Kz=place(Az,Bz,pz);
uz=-Kz*[ez;ez_dot];
vz=-uz;
%% Compute phi, theta , U1
a=vx/(vz+g);
b=vy/(vz+g);
c=cos(Psi_ref);
d=sin(Psi_ref);

tan_theta=a*c+b*d;
Theta_ref=atan(tan_theta);

if or(abs(Psi_ref)<pi/4,abs(Psi_ref)>3*pi/4)
    tan_phi=cos(Theta_ref)*(tan(Theta_ref)*d-b)/c;
else
```

```
    tan_phi=cos(Theta_ref)*(a-tan(Theta_ref)*c)/d;
end

Phi_ref=atan(tan_phi);
U1=(vz+g)*m/(cos(Phi_ref)*cos(Theta_ref));
%% Check
%(cos(Phi_ref)*sin(Theta_ref)*cos(Psi_ref)+sin(Phi_ref)*si
n(Psi_ref))/ m*U1;
%(cos(Phi_ref)*sin(Theta_ref)*sin(Psi_ref)-
sin(Phi_ref)*cos(Psi_ref))/ m*U1;
%-g+(cos(Phi_ref)*cos(Theta_ref))/m*U1;

% vx;
% vy;
% vz;
 end
%\\
%\\
```

C. Supporting file: Quadcopter_model.m

```
function dx = Quadcopter_model(t , states , U)
 % The B-frame and its transformation is used
 % instead of a hybrid frame because for the solver ode45
, it
 % is important to have the nonlinear system of equations
in the first
 % order form.
 % Constants
constants = Defining_initial_constants();
Ix = constants{1}; %kg*m^2
Iy = constants{2}; %kg*m^2
Iz = constants{3}; %kg*m^2
m = constants{4}; %kg
g = constants{5}; %m/s^2
Jtp=constants{6}; %N*m*s^2=kg*m^2
% States: [u,v,w,p,q,r,x,y,z,phi,theta ,psi]
u = states(1);
v = states(2);
w = states(3);
p = states(4);
q = states(5);
r = states(6);
x = states(7);
y = states(8);
```

```
z = states(9);
phi = states(10);
theta = states(11);
psi = states(12);

% Inputs:
U1 = U(1);
U2 = U(2);
U3 = U(3);
U4 = U(4);

% Rotational matrix that relates u,v,w with
x_dot,y_dot,z_dot
R_matrix=[cos(theta)*cos(psi),
sin(phi)*sin(theta)*cos(psi)-cos(           phi)*sin(psi),
cos(phi)*sin(theta)*cos(psi)+sin(phi)*sin(psi);
cos(theta)*sin(psi),
sin(phi)*sin(theta)*sin(psi)+cos(phi)*           cos(psi),
cos(phi)*sin(theta)*sin(psi)-sin(phi)*cos(psi);
-sin(theta), sin(phi)*cos(theta), cos(phi)*cos(theta)];

% Transformation matrix that relates p,q,r with phi_dot
,theta_dot ,psi_dot
T_matrix=[ 1, sin(phi)*tan(theta), cos(phi)*tan(theta);
           0, cos(phi), -sin(phi);
           0, sin(phi)*sec(theta), cos(phi)*sec(theta)
         ];

global omega_total

% The nonlinear equations describing the dynamics of the
Quadcopter
dx(1,1)=(v*r-w*q)+g*sin(theta); %u_dot
dx(2,1)=(w*p-u*r)-g*cos(theta)*sin(phi); %v_dot
dx(3,1)=(u*q-v*p)-g*cos(theta)*cos(phi)+U1/m; %w_dot

dx(4,1)=q*r*(Iy-Iz)/Ix-Jtp/Ix*q*omega_total+U2/Ix; %p_dot
dx(5,1)=p*r*(Iz-Ix)/Iy+Jtp/Iy*p*omega_total+U3/Iy; %q_dot
dx(6,1)=p*q*(Ix-Iy)/Iz+U4/Iz; %r_dot

dx(7,1)=R_matrix(1,:)*[u;v;w]; %x_dot
dx(8,1)=R_matrix(2,:)*[u;v;w]; %y_dot
dx(9,1)=R_matrix(3,:)*[u;v;w]; %z_dot

dx(10,1)=T_matrix(1,:)*[p;q;r]; %phi_dot
dx(11,1)=T_matrix(2,:)*[p;q;r]; %theta_dot
dx(12,1)=T_matrix(3,:)*[p;q;r]; %psi_dot
 end
```

D. Supporting file: LPV_Discretization.m

```
% A Linear paramter varying model concerning the three
rotational axis.

function [Ad, Bd, Cd, Dd, x_dot, y_dot, z_dot, phi, phi_dot
,   theta   ,   theta_dot   ,   psi   ,   psi_dot]   =
LPV_Discretization(states)
% Import values from Defining_initial_constants file
constants = Defining_initial_constants();
Ix = constants{1}; %kg*m^2
Iy = constants{2}; %kg*m^2
Iz = constants{3}; %kg*m^2
Jtp=constants{6}; %N*m*s^2=kg*m^2
Ts=constants{7}; %s



 % Assign the states
 % States: [u,v,w,p,q,r,x,y,z,phi,theta ,psi]
 u = states(1);
 v = states(2);
 w = states(3);
 p = states(4);
 q = states(5);
 r = states(6);
 phi = states(10);
 theta = states(11);
 psi = states(12);
 global omega_total;


 %%


 %   Rotational   matrix   that   relates   u,v,w   with
x_dot,y_dot,z_dot (Rotational
 % body in the body frame with the inertial frame)
 R_matrix=[cos(theta)*cos(psi),
sin(phi)*sin(theta)*cos(psi)-cos( phi)*sin(psi), ...
 cos(phi)*sin(theta)*cos(psi)+sin(phi)*sin(psi); ...
cos(theta)*sin(psi),
sin(phi)*sin(theta)*sin(psi)+cos(phi)* cos(psi), ...
cos(phi)*sin(theta)*sin(psi)-sin(phi)*cos(psi); ...
-sin(theta), sin(phi)*cos(theta), cos(phi)*cos(theta)];
```

```
x_dot=R_matrix(1,:)*[u;v;w]; %x_dot
y_dot=R_matrix(2,:)*[u;v;w]; %y_dot
z_dot=R_matrix(3,:)*[u;v;w]; %z_dot

% To get phi_dot , theta_dot , psi_dot , you need the T
matrix
% Transformation matrix that relates p,q,r with phi_dot
,theta_dot ,psi_dot
T_matrix=[1, sin(phi)*tan(theta), cos(phi)*tan(theta); ...
0, cos(phi), -sin(phi); ...
0, sin(phi)*sec(theta), cos(phi)*sec(theta)];

phi_dot=T_matrix(1,:)*[p;q;r]; %phi_dot
theta_dot=T_matrix(2,:)*[p;q;r]; %theta_dot
psi_dot=T_matrix(3,:)*[p;q;r]; %psi_dot

A12=1;
A24=-omega_total*Jtp/Ix;
A26=theta_dot*(Iy-Iz)/Ix;
A34=1;
A42=omega_total*Jtp/Iy;
A46=phi_dot*(Iz-Ix)/Iy;
A56=1;
A62=(theta_dot/2)*(Ix-Iy)/Iz;
A64=(phi_dot/2)*(Ix-Iy)/Iz;

A = [0 A12 0 0 0 0;
0 0 0 A24 0 A26;
0 0 0 A34 0 0;
0 A42 0 0 0 A46;
0 0 0 0 0 A56;
0 A62 0 A64 0 0];

B = [ 0 0 0 ;
1/Ix 0 0 ;
0 0 0 ;0 1/Iy 0 ;
0 0 0 ;
0 0 1/Iz];

C = [1 0 0 0 0 0;0 0 1 0 0 0;0 0 0 0 1 0];

D=0;

% Discretization of the system to obtain an appropriate
solution.
% Transforming continuos varables and functions into
discrete counterparts so that it is more suitable and
```

```
% convinient for numerical evaluation and implementation.
% % Forward Euler method
% Ad=eye(length(A(1,:)))+Ts*A;
% Bd=Ts*B;
% C=Cd;
% D=Dd; (which will be 0)

% Zero-Order Hold

% Create state-space
sysc=ss(A,B,C,D);
sysd=c2d(sysc ,Ts, 'zoh');
Ad=sysd.A;
Bd=sysd.B;
Cd=sysd.C;
Dd=sysd.D;


 end
 %\
```

E. Supporting file: MPC_simplification.m

```
 function                [Hdb,Fdbt,Cdb,Adc]                =
MPC_simplification(Ad,Bd,Cd,Dd,hz) %hz=4

% db - double bar
% dbt - double bar transpose
% dc - double circumflex
% Augmenting (change of input ie,. change of pitch, yaw,
roll)
A_aug=[Ad,Bd;zeros(length(Bd(1,:)),length(Ad(1,:))),eye(le
ngth(Bd (1,:)))];
B_aug=[Bd;eye(length(Bd(1,:)))];
C_aug=[Cd,zeros(length(Cd(:,1)),length(Bd(1,:)))];
D_aug=Dd; % zero matrix

constants = Defining_initial_constants();
Q = constants{8};
S = constants{9};
R = constants{10};

CQC=C_aug'*Q*C_aug;
CSC=C_aug'*S*C_aug;
QC=Q*C_aug;
```

```
SC=S*C_aug;

Qdb=zeros(length(CQC(:,1))*hz,length(CQC(1,:))*hz);
Tdb=zeros(length(QC(:,1))*hz,length(QC(1,:))*hz);
Rdb=zeros(length(R(:,1))*hz,length(R(1,:))*hz);
Cdb=zeros(length(B_aug(:,1))*hz,length(B_aug(1,:))*hz);
Adc=zeros(length(A_aug(:,1))*hz,length(A_aug(1,:)));

for i = 1:hz
 if i == hz
    Qdb(1+length(CSC(:,1))*(i-
1):length(CSC(:,1))*i,1+length(              CSC(1,:))*(i-
1):length(CSC(1,:))*i)=CSC;
     Tdb(1+length(SC(:,1))*(i-
1):length(SC(:,1))*i,1+length(SC                 (1,:))*(i-
1):length(SC(1,:))*i)=SC;
 else
    Qdb(1+length(CQC(:,1))*(i-
1):length(CQC(:,1))*i,1+length(              CQC(1,:))*(i-
1):length(CQC(1,:))*i)=CQC;
 Tdb(1+length(QC(:,1))*(i-1):length(QC(:,1))*i,1+length(QC
(1,:))*(i-1):length(QC(1,:))*i)=QC;
end

Rdb(1+length(R(:,1))*(i-
1):length(R(:,1))*i,1+length(R(1,:))*(                   i-
1):length(R(1,:))*i)=R;
for j = 1:hz
    if j<=i
        Cdb(1+length(B_aug(:,1))*(i-
1):length(B_aug(:,1))*i,1+          length(B_aug(1,:))*(j-
1):length(B_aug(1,:))*j)= A_aug^(i-j)*B_aug;
    end
end
Adc(1+length(A_aug(:,1))*(i-1):length(A_aug(:,1))*i
,1:length( A_aug(1,:)))=A_aug^(i);
end
Hdb=Cdb'*Qdb*Cdb+Rdb;
Fdbt=[Adc'*Qdb*Cdb;-Tdb*Cdb];
end
%\\
%\\
```

F. Supporting file: trajectory_generator.m

```
%% Position trajectory generation
```

```
 function  [X_ref  ,X_dot_ref  ,Y_ref  ,Y_dot_ref  ,Z_ref
,Z_dot_ref ,psi_ref]= trajectory_generator(t ,r,f ,height_i
,height_f)

constants = Defining_initial_constants();
Ts=constants{7};
%s
innerDyn_length=constants{18};
% Number of inner control loop iterations

alpha=2*pi*f.*t;
d_height=height_f-height_i;

 x = r.*cos(alpha);
 y = r.*sin(alpha);
 z = height_i+d_height/t(end)*t;

% x = (r/10.*t+2).*cos(alpha);
% y = (r/10.*t+2).*sin(alpha);
% z = height_i+d_height/t(end)*t;

% x = r.*cos(alpha);
% y = r.*sin(alpha);
% z = height_i+50*d_height/t(end)*sin(t);

% x = r.*cos(alpha);
% y = 2.*t;
% z = height_i+d_height/t(end)*t;

% x = 2.*t/20+1;
% y = 2.*t/20-2;
% z = height_i+d_height/t(end)*t;

% x = r.*cos(alpha);
% y = r.*sin(alpha);
% z = height_i+50*d_height/t(end)*(t);



dx=[x(2)-x(1) ,x(2:end)-x(1:end-1)];
dy=[y(2)-y(1) ,y(2:end)-y(1:end-1)];
dz=[z(2)-z(1) ,z(2:end)-z(1:end-1)];

x_dot=dx.*(1/(Ts*innerDyn_length));
y_dot=dy.*(1/(Ts*innerDyn_length));
z_dot=round(dz.*(1/(Ts*innerDyn_length)),8);

psi=zeros(1,length(x));
```

```
psi(1)=atan2(y(1) ,x(1))+pi/2;
psi(2:end)=atan2(dy(2:end),dx(2:end));

for i = 1:length(psi)
    if psi(i)<0
        psi(i)=2*pi-abs(psi(i));
    end
end

for i = 1:length(psi)
 if i>1
     if abs(psi(i)-psi(i-1))>pi
    psi(i :end)=psi(i :end)+2*pi;
    end
 end
end

X_ref = [t' x'];
X_dot_ref = [t' x_dot'];
Y_ref = [t' y'];
Y_dot_ref = [t' y_dot'];
Z_ref = [t' z'];
Z_dot_ref = [t' z_dot'];
psi_ref = [t' psi'];
end
%\\
 %\\
```