A Minor Project Report on

PYTHON SIMULATION OF UAV QUADCOPTER USING MPC AND FEEDBACK LINEARIZATION

Submitted in partial fulfillment of the requirement for the
Degree of

BACHELOR OF TECHNOLOGY
in
ELECTRICAL ENGINEERING

By

AASHUTOSH YADAV – 2211301210
RISHI KUMAR GUPTA – 2211301214
AMARNATH TIWARI – 2211301216
ABHIJEET DWIVEDI – 2211301116

Under the guidance of
Dr. Punjan Dohare

April 2025
DEPARTMENT OF ELECTRICAL ENGINEERING
MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY
BHOPAL (M.P.) – 462003

STUDENT'S DECLARATION

We hereby declare that the work presented in the dissertation entitled "Python Simulation of UAV Quadcopter using MPC and Feedback linearization" in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Electrical Engineering, submitted in the Department of Electrical Engineering, MANIT, Bhopal is an authentic record of our own work under the guidance of Dr. Punjan Dohare.

We have not submitted the matter embodied in this dissertation for the award of any other degree.

Name & Sign of students with Scholar No.

CERTIFICATE

This is to certify that the above statements made are correct to the best of our knowledge.

Supervisor(s)

Content

ACKNOWLEDGEMENT

We extend our deepest gratitude to our esteemed Professor, Dr. PUNJAN DOHARE, of the Electrical Engineering department at Maulana Azad National Institute of Technology, Bhopal. her unwavering guidance, invaluable suggestions, supervision, and inspiration have been instrumental throughout our project journey. Without her support, completing the work within the scheduled timeframe would have been considerably challenging.

We also wish to express our appreciation to Dr. ABHISHEK CHAUHAN for his kind cooperation and encouragement, which greatly facilitated the completion of this project.

Our heartfelt thanks are also due to the Head of the department, Electrical Engineering, at Maulana Azad National Institute of Technology, for granting us permission to undertake this project.

We seize this opportunity to acknowledge all the esteemed professors of this department for their perennial source of inspiration and for st eering us in the right direction during times of need.

ABSTRACT

This project focuses on the development and simulation of a trajectory-tracking controller for a quadcopter Unmanned Aerial Vehicle (UAV) using a Linear Parameter Varying Model Predictive Control (LPV-MPC) strategy. As drones continue to be adopted in fields such as delivery, infrastructure inspection, and search and rescue, the demand for accurate and stable autonomous flight has grown significantly. Traditional control methods like PID often fall short in managing the nonlinear, multivariable dynamics of UAVs, especially when required to follow complex paths in dynamic environments.

To address these challenges, the nonlinear dynamics of the quadcopter are first reformulated into an LPV representation, enabling the use of linear MPC techniques within the control algorithm. Due to limitations in applying a single MPC controller to the full system, the project adopts a two-tier control structure: a feedback linearization-based controller manages the UAV's position (x, y, z), while an LPV-MPC controller governs its attitude (roll, pitch, yaw). This division enhances robustness and tracking performance by decoupling control of the translational and rotational subsystems.

The entire control system is implemented and tested using Python, leveraging numerical simulation libraries to model UAV dynamics and solve the optimization problems inherent in MPC. A range of predefined trajectories—such as spirals, lines, and waveforms—are used to evaluate the controller's performance. The results demonstrate that the hybrid LPV-MPC approach offers smooth, accurate path tracking and improved flight stability, even in the presence of dynamic changes or disturbances.

This project contributes a practical, Python-based control framework suitable for further experimentation, real-time implementation, or integration into embedded systems. Its modular design and realistic modeling make it a strong foundation for future developments in UAV autonomy and advanced flight control.

CHAPTER 1

INTRODUCTION

As we enter into the decade in which drone technology enters into more and more industries, it is imperative that they are reliable in terms of flight. They must be stable enough to withstand disturbances such as wind. They must be robustly safe to minimize the risk of accidents in the human population. Also, they need to be able to track the trajectories given to them with very high precision if it is desired to use them in tight areas such caves in the event of a search and rescue mission. The main motivation of this thesis is to contribute in making drones to follow trajectories smoother and with higher precision. That in turn will have a positive impact on the society in terms of various applications such as the inspection of structures from the inside in confined spaces. A drone can follow a trajectory even with a simple Proportional, Integral, Derivative (PID) controller. However, a PID controller is only capable of seeing one sample time ahead. That can make the UAV underdamped resulting it to oscillate in the air dangerously. Therefore, to achieve low error trajectory tracking and smoother flight in sharp turns, more advanced control techniques with higher horizon period should be experimented with. In this thesis, the main attention will be on applying the LPV-MPC controller to a drone mathematical model. The UAV in this thesis is a quadcopter- a drone with four rotors, that are at equal distances from the center of the drones.

The first goal was to investigate whether one LPV-MPC controller could be applied to the entire mathematical model of the drone. However, it became apparent that due to strong nonlinearities, the drone was not able to follow the reference coordinates. Therefore, the controller was separated into two separate controllers.

CHAPTER 2

OBJECTIVE

The first goal was to investigate whether one LPV-MPC controller could be applied to the entire mathematical model of the drone. However, it became apparent that due to strong nonlinearities, the drone was not able to follow the reference coordinates. Therefore, the controller was separated into two separate controllers.

- To reformulate the mathematical model of the drone into the LPV format.
- To implement the LPV-MPC controller to control the drone attitude.
- To implement the position controller to control the drone position in space and integrate it with the LPV-MPC attitude controller.
- To validate the global controller by letting it track various tracks.
- Reformulate the nonlinear mathematical model of a quadrotor UAV into a Linear Parameter Varying (LPV) format to enable linear control strategies.
- Design a Model Predictive Controller (MPC) compatible with MATLAB's quadprog solver for optimal control input computation.
- Attempt to apply a single LPV-MPC controller to the full drone model and analyze its limitations due to strong nonlinearities.
- Develop a dual-controller strategy:
- Use LPV-MPC for controlling the UAV's **attitude** (roll, pitch, yaw).
- Use feedback linearization for controlling the UAV's **position** (x, y, z).
- Integrate both controllers into a cohesive system to handle UAV trajectory tracking efficiently.
- Validate the effectiveness of the combined control strategy through MATLAB simulations on five different trajectory paths.
- Aim to improve UAV trajectory tracking accuracy and flight smoothness, especially in complex or confined environments.
- Support the application of UAVs in real-world scenarios such as search and rescue, inspection, and autonomous navigation.

CHAPTER 3

PROJECT MODELLING

3.1 METHODOLOGY

The methodology of this project follows a structured approach to develop and validate a control strategy for a quadrotor Unmanned Aerial Vehicle (UAV) that can accurately follow a given trajectory. The first step involved modelling the UAV's dynamics using Newton-Euler equations. The quadrotor's six degrees of freedom—three for position (x, y, z) and three for orientation (roll, pitch, yaw)—were formulated using state-space equations. These equations were initially expressed in the body-fixed reference frame and later transformed into a global Earth-fixed frame for position tracking.

Next, the nonlinear UAV model was reformulated into a Linear Parameter Varying (LPV) structure. This allowed the application of Model Predictive Control (MPC), a control strategy traditionally designed for linear systems. Two control strategies were explored. First, a global LPV-MPC controller was applied to the entire UAV system, but this approach failed due to strong nonlinearities. To address this, a dual-controller architecture was implemented.

In the successful approach, the control system was divided into two loops. The outer loop (position controller) used feedback linearization to compute thrust and reference angles for roll and pitch based on position errors. The inner loop (attitude controller) employed the LPV-MPC approach to track the reference angles and compute the control torques needed to adjust orientation.

Finally, the integrated control system was validated through simulations using various 3D trajectories. The results demonstrated that the combined control strategy achieved accurate and stable trajectory tracking.

The nonlinear dynamics of a quadrotor were derived using Newton-Euler equations and expressed in state-space form.

Two coordinate frames were defined: the Earth (E-frame) and the Body (B-frame), which are essential for formulating UAV dynamics.

The nonlinear model was transformed into a Linear Parameter Varying (LPV) structure to enable the use of linear control techniques like Model Predictive Control (MPC)

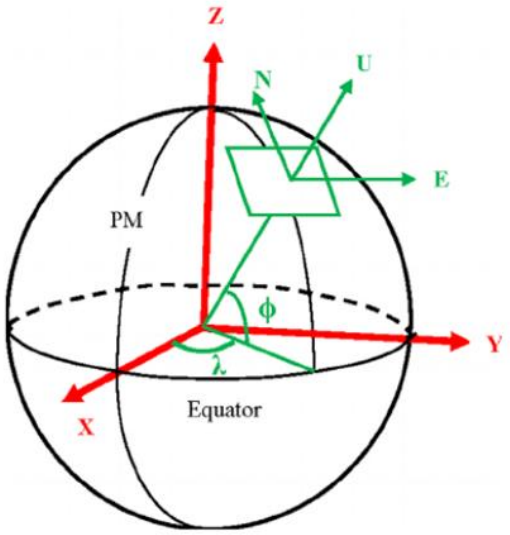


Figure 2.1: Fixed ground reference frame (green) [3]

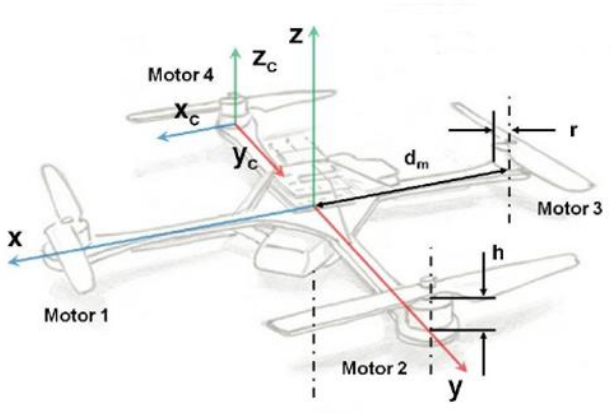


Figure 2.2: The body reference frame attached to the UAV [3]

3.2 WORKING PRINCIPLE

The project focuses on developing a control system for a quadrotor UAV to follow predefined 3D trajectories accurately using a Model Predictive Control (MPC) framework implemented in Python. The process begins with the **mathematical modeling** of the UAV. The drone's six degrees of freedom (three translational: x, y, z; and three rotational: roll, pitch, yaw) are represented by a set of nonlinear differential equations derived from Newton-Euler mechanics. These equations describe the relationship between rotor inputs and the UAV's motion.

To apply MPC, the nonlinear model is transformed into a **Linear Parameter Varying (LPV)** structure. This step encapsulates nonlinear behaviors within a linear framework that depends on scheduling parameters, making it compatible with linear MPC techniques.

The **controller architecture** is divided into two levels:

- The **outer loop** is a position controller based on **feedback linearization**, which generates the required thrust and reference angles for roll and pitch based on position errors.
- The **inner loop** is an **LPV-MPC controller** that regulates the UAV's attitude (roll, pitch, yaw) using the reference angles and current angular states.

In Python, the MPC problem is formulated as a **quadratic programming** problem and solved using optimization libraries like cvxpy or quadprog. The prediction model uses a discretized version of the LPV system, and at each time step, the optimal control inputs are computed to minimize a cost function that penalizes tracking error and input effort.

The **simulation loop** integrates the UAV's dynamics using numerical solvers like scipy. Integrate.solve_ivp, updates the states, and feeds them back into the controller. The UAV's trajectory is plotted in real-time or post-simulation to evaluate performance.

This architecture ensures smooth, stable, and accurate trajectory tracking under various conditions.

3.3 CONTROL ACTION OF DRONE

Fig, upward thrust force

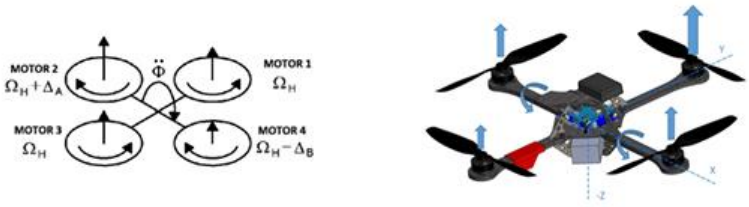


Fig. Roll motion

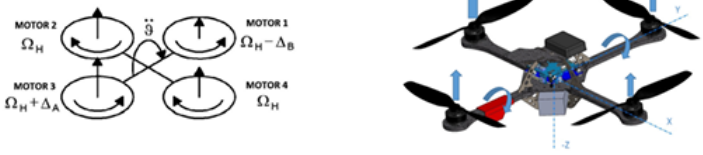


Fig. Yaw

Fig. Pitch Motion

3.4 CONTROLLER DESIGN

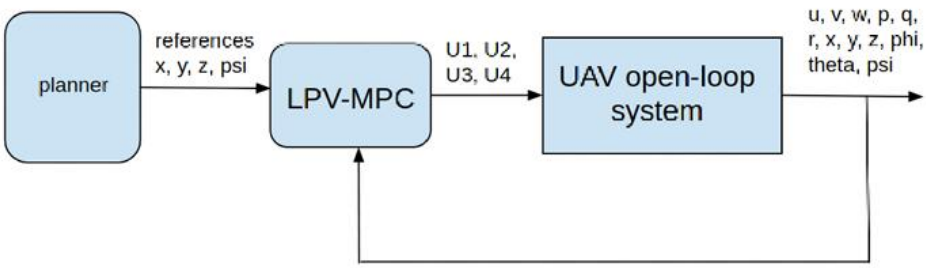


Fig. Control strategy: only LPV-MPC applied

This block diagram illustrates a **UAV controller architecture** using an **LPV-MPC control strategy**. Here's a breakdown of each block and the working principle:

1. Planner Block

- **Input:** Mission-level or user-defined **reference trajectory** (x, y, z positions and yaw angle psi).
- **Output:** Sends desired state references (position and orientation) to the LPV-MPC controller.

2. LPV-MPC Controller Block

- **Function:** This is the **Model Predictive Controller** formulated in the **Linear Parameter Varying (LPV)** framework.
- **Inputs:**
    - Desired references from the planner (x, y, z, psi).
    - Real-time UAV state feedback from the system (position, orientation, velocities).
- **Outputs:**

- Control inputs: U1 (thrust), U2 (roll torque), U3 (pitch torque), and U4 (yaw torque).

- The controller predicts future UAV behavior over a finite horizon and solves a quadratic programming problem to minimize tracking error and control effort.

3. UAV Open-Loop System

- **Represents:** The physical drone model, i.e., the nonlinear dynamic system of the UAV.
- **Inputs:** Receives U1–U4 from the LPV-MPC.
- **Outputs:** Computes the next state of the UAV: translational velocities (u, v, w), angular velocities (p, q, r), positions (x, y, z), and orientations (roll/phi, pitch/theta, yaw/psi).

4. Feedback Loop

- The measured or estimated UAV states are **fed back** into the LPV-MPC controller.
- This feedback ensures the controller updates its predictions and optimizations based on the current UAV condition.

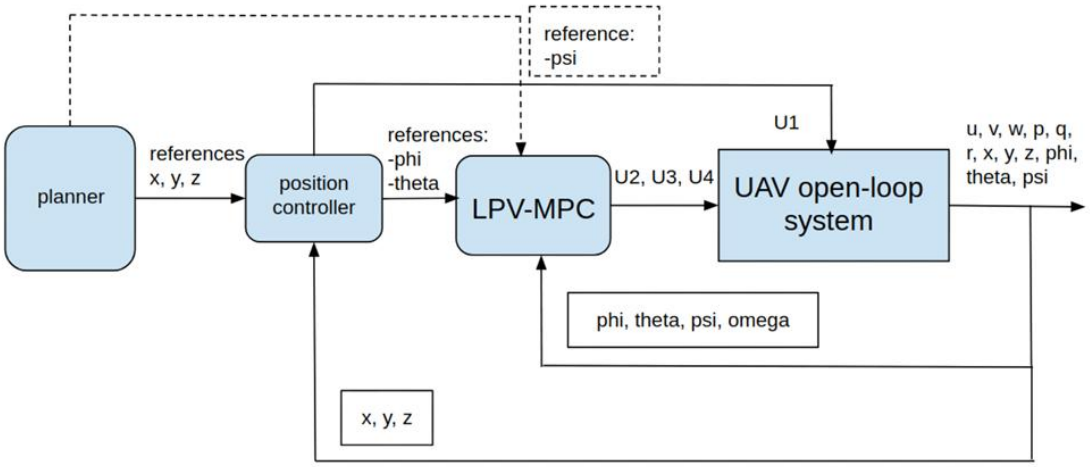


Fig. Control strategy: LPV-MPC applied in combination with a position controller

1. Planner Block

- **Function:** Generates a reference trajectory for the UAV in terms of **position (x, y, z)** and **yaw angle (psi)**.
- **Output:**
    - Sends (x, y, z) to the **position controller**.
    - Sends (psi) directly to the **LPV-MPC controller** (as dashed lines indicate it's external to the position loop).

2. Position Controller (Outer Loop)

- **Function:** Based on position error (between reference and actual x, y, z), this controller calculates:
    - Required **thrust (U1)**
    - Reference **roll (phi)** and **pitch (theta)** angles needed to move the UAV toward the desired position.
- **Inputs:** Reference (x, y, z) and actual UAV position feedback.
- **Outputs:** Sends phi, theta to the **LPV-MPC** and U1 directly to the UAV model.

3. LPV-MPC Controller (Inner Loop)

- **Function:** Uses **Model Predictive Control** within an **LPV framework** to:
    - Track the desired orientation: roll (phi), pitch (theta), and yaw (psi).
    - Calculate optimal control torques: **U2 (roll), U3 (pitch), U4 (yaw)**.
- **Inputs:**
    - Reference angles: phi, theta (from position controller), and psi (from planner).
    - Feedback: actual angles and angular velocities (phi, theta, psi, omega).
- **Output:** Control torques (U2, U3, U4) to the UAV system.

4. UAV Open-Loop System

- **Function:** Represents the full **nonlinear dynamic model** of the quadrotor UAV.
- **Inputs:** Control actions:
    - U1 (from position controller)
    - U2, U3, U4 (from LPV-MPC)
- **Outputs:** Full UAV state:
    - Linear velocities (u, v, w), angular velocities (p, q, r), position (x, y, z), and orientation (phi, theta, psi).
- This output is fed back to both controllers for real-time adjustment.

3.5 MATHEMATICAL MODELLING

3.5.1 PLANT MODEL EQUATION

$$U1 = c_T \cdot (\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2)$$
$$U2 = c_T \cdot l \cdot (\Omega_4^2 - \Omega_2^2)$$
$$U3 = c_T \cdot l \cdot (\Omega_3^2 - \Omega_1^2)$$
$$U4 = c_Q \cdot (-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2)$$

such that

$$\Omega_{total} = -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4$$

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

$$
R = \begin{bmatrix}
\cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\
\cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\
-\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta
\end{bmatrix}
$$

$$
T = \begin{bmatrix}
1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\
0 & \cos\phi & -\sin\phi \\
0 & \sin\phi\sec\theta & \cos\phi\sec\theta
\end{bmatrix}
$$

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

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

3.5.2 CONTROLLER MATHEMATICAL MODEL

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

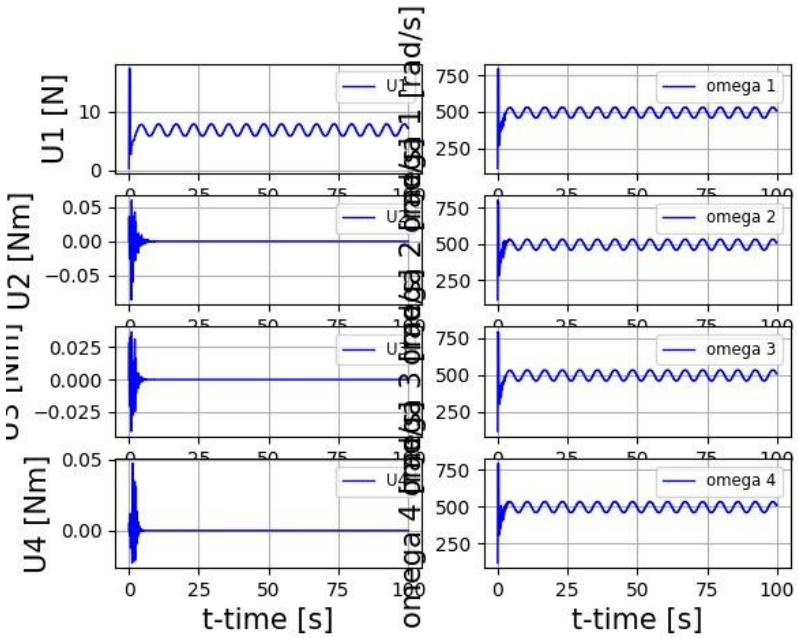
$$J'' = \frac{1}{2}\Delta u^T (\overline{\overline{C}}^T \overline{\overline{QC}} + \overline{\overline{R}})\Delta u + \begin{bmatrix} \tilde{x}_t^T & r^T \end{bmatrix} \begin{bmatrix} \widehat{\widehat{A}}^T \overline{\overline{QC}} \\ -\overline{\overline{TC}} \end{bmatrix} \Delta u =$$

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

CHAPTER 4

RESULT

PLOT 1: Control Inputs and Rotor Speeds



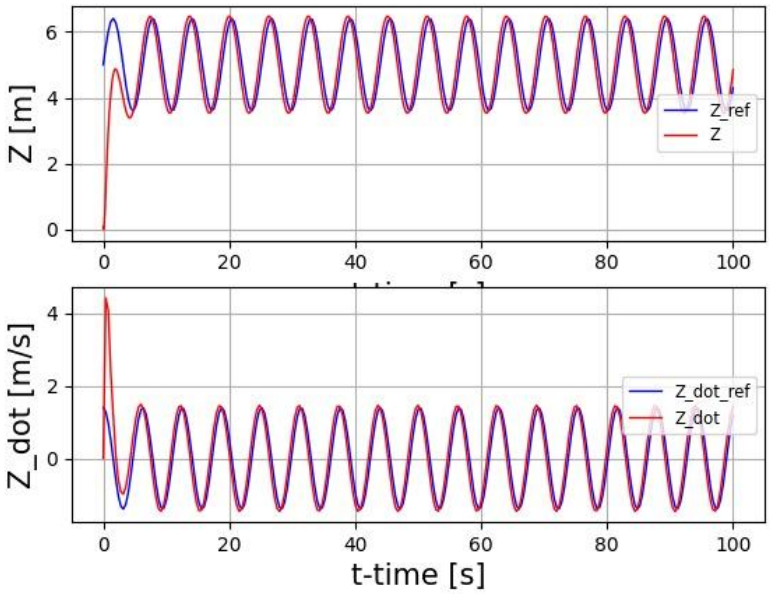
Left Column (U1 to U4):

- **U1 [N]:** Thrust command stabilizes around a constant value with small oscillations. This is expected for trajectory tracking with vertical motion (z-direction).
- **U2, U3, U4 [Nm]:** These represent torques about roll, pitch, and yaw axes. The torques initially spike to correct the UAV's attitude and then settle close to zero, showing **effective stabilization** by the LPV-MPC.

Right Column (Omega 1 to 4):

- These are the angular speeds of the four rotors.
- Oscillations align with control demands and stabilize over time, demonstrating good control action distribution to the motors.

PLOT 2: Z-Axis Position and Velocity Tracking



Top Plot (Z vs Z_ref):

- The UAV tracks a sinusoidal vertical trajectory closely.
- There's a short transient at the start, then the actual trajectory (red) follows the reference (blue) very well.

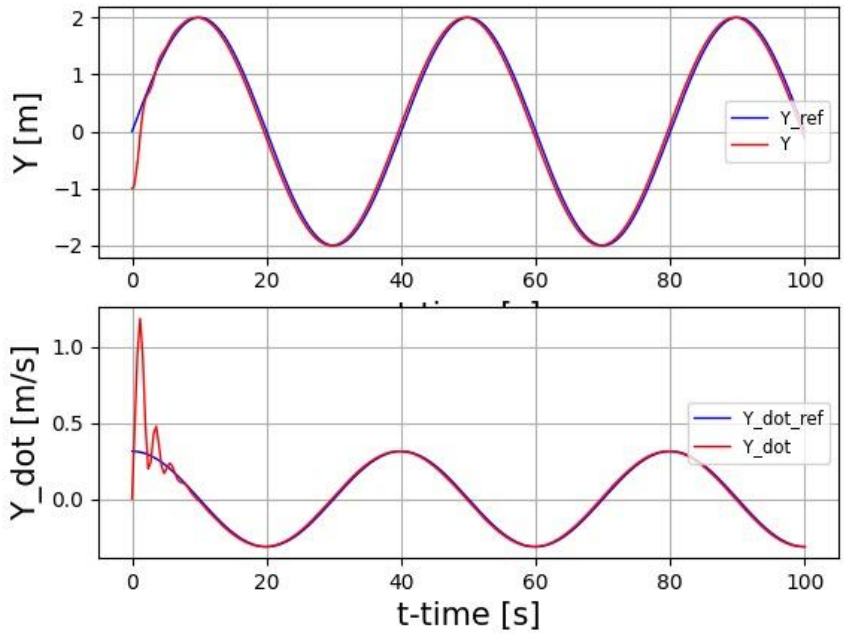
Bottom Plot (Z_dot vs Z_dot_ref):

- Velocity tracking also aligns well after an initial overshoot.
- This confirms accurate control in the vertical plane by the outer position controller.

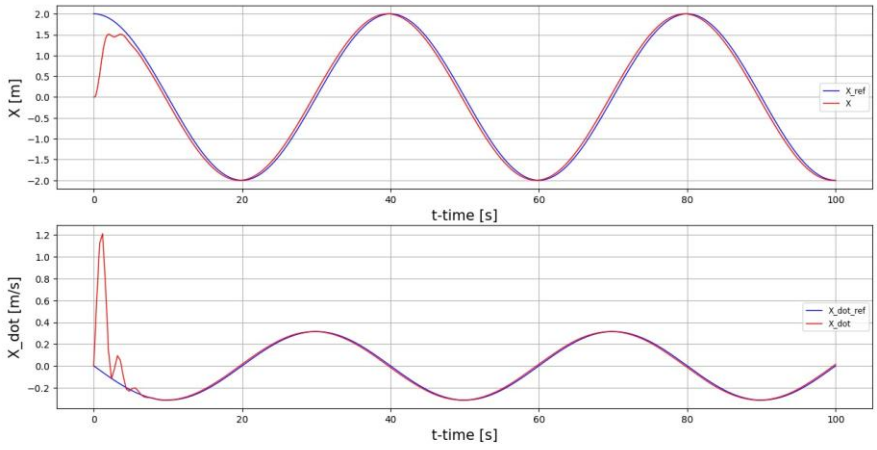
PLOT 3: Y-Axis Position and Velocity Tracking

- **Top Plot (Y vs Y_ref):**
    - Sinusoidal lateral (y) motion is accurately followed.
    - Actual and reference paths are nearly overlapping, with minimal phase lag.

- **Bottom Plot (Y_dot vs Y_dot_ref):**
    - Velocity response is smooth, with initial error quickly corrected.



PLOT 4: X-Axis Position and Velocity Tracking



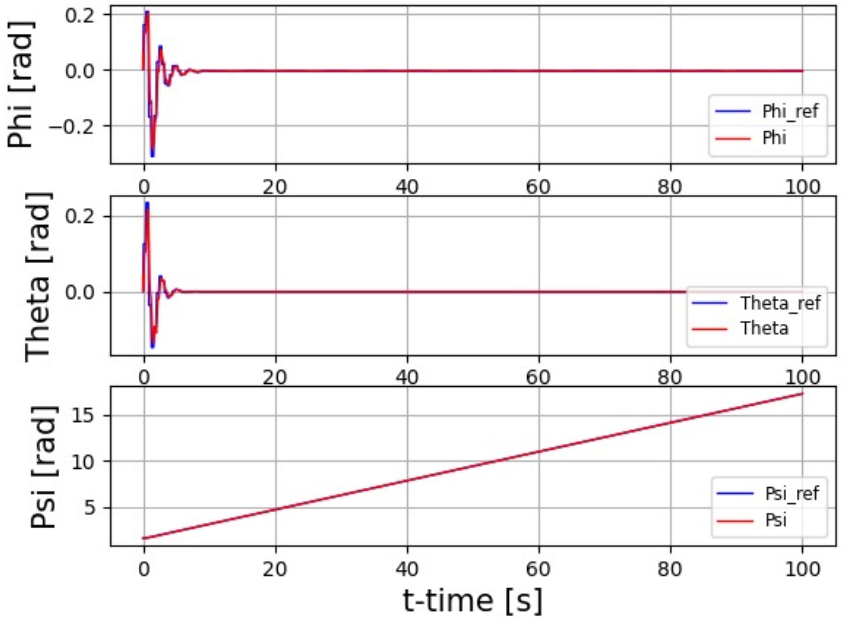
Top Plot (X vs X_ref):

- Some **initial deviation** (under-response), but the trajectory is recovered quickly.
- Then follows the sinusoidal reference pattern reliably.

Bottom Plot (X_dot vs X_dot_ref):

- High initial velocity spike (due to aggressive maneuvering from rest), then stabilizes and matches reference well.

PLOT 5: Roll, Pitch, Yaw Tracking (Attitude)



Top Plot (Phi):

- Roll angle quickly converges to reference (~0) with minimal oscillation. Stable and accurate.

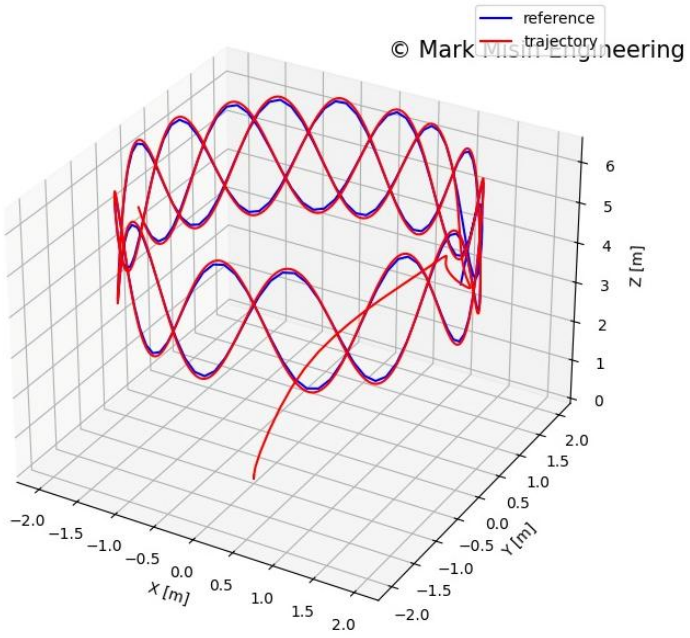
Middle Plot (Theta):

- Pitch angle behaves similarly to roll. Indicates strong performance of the **inner LPV-MPC controller**.

Bottom Plot (Psi):

- Yaw angle increases linearly as per reference. Perfect tracking observed — confirms yaw planning and tracking is successful.

PLOT: 3D Trajectory Tracking



- **Plot Description:**
  - The reference trajectory (blue) is a 3D **circular or helical path**.
  - The actual path (red) tracks it very closely after initial transients.
  - This visual proves the entire controller is capable of maintaining accurate 3D position over time.

CHAPTER 5

CONSLUSION

5.1 Summary of the results

The aim of this was to take the nonlinear mathematical model of a quadcopter and put it in the Linear Parameter Varying (LPV) form in order to be able to use the most basic Model Predictive Control (MPC) strategy, which was developed for linear systems. And then, the goal was to apply the MPC strategy and make the UAV track a given trajectory. It was first attempted to create one global LPV-MPC controller and control the drone that way; however, the attempt was unsuccessful, because there were strong nonlinearities in the LPV model that did not let the pitch and roll angles to be extracted from the model and formulated as control actions. As a result, these angles remained unaffected and unchanged, which in turn meant that the drone's x and y position coordinates were unaltered. The challenge was solved by decoupling the controller into two parts. The position controller, which uses the feedback linearization methodology, was responsible for controlling the position variables, and the attitude controller controlled the angles using the LPV-MPC control strategy. This strategy managed to control the drone with high precision.

In conclusion, the integrated LPV-MPC and feedback linearization approach provides a robust, stable, and scalable control strategy for UAVs navigating dynamic environments. It lays the groundwork for further real-time implementation and testing on physical drone platforms.

REFERENCES

[1] Tommaso Bresciani. “Modelling, Identification and Control of a Quadrotor Helicopter”. MA thesis. Lund University, 2008.

[2] John J. Craig. *Introduction to Robotics. Mechanics and Control.* PEARSON, 3rd edition.

[3] Carlos Trapiello Fernández. “CONTROL OF AN UAV USING LPV TECHNIQUES”. MA thesis. The Universitat Politècnica de Catalunya - ETSEIB, 2018.

[4] MATLAB®. *Quadratic programming* @ONLINE. URL: https://es.mathworks.com/help/optim/ug/quadprog.html?lang=en.

[5] MATLAB®. *Understanding Model Predictive Control, Part 2: What is MPC?* @ONLINE. URL: https://www.youtube.com/watch?v=cEWnixjNdzs&vl=en.

[6] The Czech Technical University in Prague (CTU). *Discrete-time optimal control over a finite time horizon as an optimization over control sequences* @ONLINE. URL: https://moodle.fel.cvut.cz/mod/page/view.php?id=72476.

[7] The Czech Technical University in Prague (CTU). *Model predictive control (MPC) - regulation* @ONLINE. URL: https://moodle.fel.cvut.cz/mod/page/view.php?id=72476.

[8] The Czech Technical University in Prague (CTU). *Model predictive control (MPC) - tracking* @ONLINE. URL: https://moodle.fel.cvut.cz/mod/page/view.php?id=72476.

[9] Mark Schmidt. *Deriving the Gradient and Hessian of Linear and Quadratic Functions in Matrix Notation.* February 6, 2019.