Optimal Aggressive Constrained Trajectory Synthesis and Control for Multi-Copters

by

TSUNG-LIANG LIU

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2023

To

my father Chin-Shen Liu, my mother Chiu-Kuei Hung,

my elder sisters Alice and Grace Liu, my elder brother Jimmy Liu,

and my beloved wife Ashley Ting

This work would not be possible without all your love and support.

Acknowledgements

First and foremost, I would like to express my deepest appreciation to my advisor, Dr. Kamesh Subbarao, for allowing me to be a part of his dedicated group of engineers, guiding me throughout my Ph.D. studies, and believing in my ability to conduct research at my own pace. It is a privilege to learn from him, as he possesses excellent knowledge and a great passion for engineering research and teaching. I am also extremely grateful to my committee members, Dr. Alan P. Bowling, Dr. Animesh Chakravarthy, Dr. Bo P Wang, and Dr. Yijing Xie, for generously providing their knowledge and expertise to enrich this dissertation.

I am deeply indebted to the National Chung-Shan Institute of Science and Technology in Taiwan for supporting my Ph.D. program. Additionally, this research has been supported by the Office of Naval Research under grant number N00014-18-1-2215 and the National Science Foundation (S&AS) under grant number 1724248.

Special thanks go to Dr. Animesh Chakravarthy for kindly providing access to the Vicon motion capture system in the Guidance and Controls of Autonomous Systems Laboratory (GCASL) for quadcopter flight experiments, Abhishek Kashyap for helping with the Vicon system setup and integration, Uluhan Cem Kaya for sharing extensive knowledge and working experience in PX4 and ROS programming, Kashish Dhal for getting me started with RotorS simulation, Suguru Sato for assisting with quadcopter 3D modeling and mass property analysis in Solidworks, Abel Martinez Martinez for aiding in the rotor system tests, and Baris Taner for helping with subsystem integration and tests. This research would not have been successful without their generous support.

I would like to extend my sincere thanks to all my teachers at the University of Texas at Arlington for their guidance and advice. Moreover, I am grateful to the administrative staff of the Mechanical and Aerospace Engineering Department, especially Lanie Gordon, Wendy Ryan, and Ayesha Fatima, for their support and dedication. I would also like to thank my colleagues, Baris, Cem, Kashish, Saina, Suguru, Jinay, Rajnish, Diganta, Kamal, and all my past and present lab-mates at the Aerospace Systems Laboratory (ASL) for their friendship, constant support, and knowledge/culture/food sharing.

Words cannot express my gratitude to my incredible parents, Chin-Shen Liu and Chiu-Kuei Hung, and my supportive siblings, Alice, Grace, and Jimmy Liu. Their unconditional support, encouragement, and belief in me have been instrumental in completing this demanding endeavor. Lastly and most importantly, I am extremely grateful to my loving wife, Ashley Ting, and our wonderful children, Howard and Ellie Liu. Throughout this demanding journey of completing my dissertation, their unwavering love, understanding, and patience have been invaluable. Ashley has been my best life partner and the foundation that helped me persevere. She provides endless encouragement, belief in my abilities, and a comforting presence during both the triumphs and challenges I faced. Howard and Ellie have been a constant source of inspiration and motivation. Their laughter, hugs, and moments of respite have provided the necessary life balance during my Ph.D. studies, reminding me of what truly matters.

July 17, 2023

Abstract

Optimal Aggressive Constrained Trajectory Synthesis and Control for Multi-Copters

Tsung-Liang Liu, Ph.D.

The University of Texas at Arlington, 2023

Supervising Professor: Kamesh Subbarao

Multi-copters have become increasingly popular in recent years due to their availability, mobility, agility, and flexibility. They serve as excellent platforms for control experiments and various applications. Their characteristics also make them an attractive choice for high-speed aerial navigation in complex environments. Numerous studies have been conducted on aggressive trajectory generation and maneuver tracking. *However, the level of aggressiveness achievable depends on the optimality of the trajectory plan and the dynamic capabilities of the vehicle.* It would be valuable if one can generate a trajectory that fully utilizes the dynamic capabilities of the multi-copter while accommodating specific maneuvers at predetermined locations. Therefore, in this research we assume that the multi-copter's dynamic capability is limited by its maximum rotor thrust and address the problem as "**Given waypoint coordinates, heading angles, and certain vehicle velocity and attitude constraints at these waypoints, how to find a feasible trajectory that fully utilizes the dynamic capabilities of a multi-copter to achieve optimal aggressiveness and perform precise trajectory tracking control?**". We divide this research into three phases. In the first phase, we adopt a simplified quadcopter dynamic model and develop a synthesis framework that fully utilizes the available dynamic capability of the quadcopter

when performing the most aggressive maneuver. Next to be more realistic, we further consider rotor aerodynamic, gyroscopic, and rolling effects in phase two and perform high fidelity Gazebo/RotorS simulation. In the final phase, we perform real quadcopter flight experiments to verify all the theories we developed. The summary of the three research phases is provided below.

1. First phase of the research focuses on developing a synthetic architecture to find the most aggressive trajectory based on the waypoints, the requirement on vehicle heading and maneuver at waypoints, and the full usage of rotor thrust for a multi-coper. We use multi-segment polynomials to generate the trajectory and ensure its smoothness by maintaining continuity in trajectory derivatives. To optimize the polynomial coefficients, we form the quadratic problem with the cost on the trajectory derivative which is related to the actuator input of the vehicle. We incorporate waypoint velocity and attitude requirements as equality constraints in the optimization problem. To optimize the segment time between waypoints, we introduce an augmented cost on both trajectory derivative and total time. We connect the polynomial trajectory with the actual quadcopter dynamics and rotor inputs through inverse dynamics analysis. We finally define aggressiveness and find the optimal trajectory with the desired aggressiveness by tracking the maximum rotor thrust required for the optimized trajectory. As the optimized aggressive trajectory is obtained, a geometric controller is used to perform the polynomial trajectory tracking and verify the feasibility of the trajectory and the accuracy of the inverse dynamics analysis. Further, a novel method for yaw trajectory optimization is also proposed to further improve the aggressive performance in the case when there are no specific requirements on the vehicle heading.
2. Many aggressive trajectory studies have employed a simplified multi-copter dynamic model that neglects factors such as aerodynamic, gyroscopic, and rotor rolling ef-

fects. However, these effects could be significant, especially during aggressive flights. Therefore, in the second phase of the research we further incorporate these additional effects into the dynamic model, examine how they could affect the vehicle dynamics, and propose compensatory measures in both optimal constrained trajectory planning and geometric control trajectory tracking. We integrate the compensations for these effects into the optimal aggressive constrained trajectory synthesis proposed in the phase one. Firstly we integrate all these effects into the inverse dynamics analysis and solve for the vehicle states, control inputs, and rotor speeds in an iterative fashion. Then, we integrate the aerodynamic drag into the trajectory derivative constraints for window passing to compensate for attitude changes caused by drag forces. This is also an iterative process, and the modified inverse dynamics analysis is utilized to solve for rotor speeds at each window waypoint iteratively. In trajectory planning, we observe how the compensation for aerodynamic drag could significantly change the resultant optimized trajectory. In trajectory tracking simulations, we observe how the compensations for the additional effects could improve the tracking performance.

3. In the final phase, we verify the aggressive trajectory optimization and the trajectory tracking control through real flight experiments. The DJI F450 quadcopter is used to conduct the flight tests in the Vicon motion capture environment. The SITL (Software in the Loop) test environment has been built up as well as the actual flight system. We obtain inertia properties of the vehicle through detailed SolidWorks CAD modeling. A series of static, and flight tests are conducted to identify all the parameters in our quadcopter dynamic model through maximum likelihood estimation. The trajectory optimization framework is modified according to the actual quadcopter actuator configuration, the identified parameters, and additional dynamic model adjustments found necessary during flight tests. The trajectory tracking control is based on the native PX4 flight controller and one companion trajectory controller running

on the on-board computer to offer trajectory information and additional feedforward and compensation terms that were proposed in the second phase of this research to the flight controller. We tuned the rate and attitude controller gains for fast tracking. We overcome the issue of conservative attitude set-point mapping and attitude tracking delay caused by rotor time constant by pre-sending scaled feedforward acceleration commands with a fixed lead time. Finally, the precise trajectory tracking is achieved, and the feasibility of the optimal aggressive constrained trajectory is confirmed through actual narrow window passing flight tests.

Table of Contents

List of Illustrations

List of Tables

Chapter 1

Introduction

1.1 Motivation and Background

Phase 1 – Aggressive Constrained Trajectory Optimization

Multi-copters, such as quadcopters, hexacopters, octocopters, and others have been very popular in recent years mainly because of their availability, mobility, agility, and flexibility. They are a great platform for control experiments and various applications. Their characteristics also make them an attractive choice for high-speed aerial navigation through complex environments. Various researches related to aggressive trajectory generation and aggressive maneuver tracking have been conducted. Some recent examples include [1], in which the authors proposed a novel control law for accurate tracking of aggressive quadcopter trajectories. In [2] the authors presented a framework to do optimal time allocation for quadcopter trajectory generation. In [3] the authors addressed the problem of performing aggressive quadcopter maneuvers that are attitude-constrained. We pursue aggressiveness because time is a critical issue in the given scenario. We adopt aggressive maneuvers because the environment is complicated and sometimes very specific maneuvers are needed in order to satisfy path and vehicle state constraints, given the environment model. However, the level of aggressiveness one can achieve depends on the optimality of the trajectory plan and the dynamic capability of the vehicle. It will be valuable if one can generate a trajectory that fully utilizes the dynamic capability of the vehicle while accommodating the maneuvers required at specific locations.

Firstly regarding trajectory optimization, in [4] the authors proposed an effective and efficient method to jointly optimizing spatial and temporal parameters of a trajectory. In

this method, the actuator constraints are only enforced by restricting norms on trajectory velocity and acceleration. However, these variables only have loosely connection to the maximum rotor thrust needed in the trajectory especially in aggressive maneuvers. In [5] the authors proposed a multi-fidelity Bayesian optimization framework to find time-optimal quadrotor trajectory. This method involves machine learning and simulation/experiment samples which are scenario dependent. Moreover, the maximum rotor thrust can hardly be specified with this method. In [6] the authors found time-optimal trajectory through discrete state space model optimization which is highly computational demanding. The result discrete non-analytical state trajectory could make it difficult to be tracked accurately. In [7] the authors proposed an optimization-based framework to find minimum control effort trajectory subject to geometrical spatial constraints and user-defined dynamic constraints. This method is not trivial because all the objective function, user-defined constraints and vehicle dynamics must be expressed in differential flat output space with spatial and temporal gradients available. In this method, the rotational dynamics is neglected, the dynamic constraint is on collective thrust rather than individual rotor thrust, and the full usage of thrust capability is not pursued. For methods in [4–6], constraints on vehicle maneuver at waypoints can hardly be implemented. For methods in [6] and [7], it could be difficult to adopt realistic vehicle dynamics. Secondly regarding vehicle maneuver constraint at waypoints, in [8] the authors proposed a trajectory planning method for quadrotor to fly through narrow gaps. This method is to approach an initial point and stay on the plane orthogonal to the gap with constant collective thrust forming a ballistic trajectory passing through the center of the gap until reaching the final point. It would be hard to incorporate this approach into a trajectory optimization framework. In [9] and [10], the authors proposed a safe-passage cone-based guidance strategy for robots to pass through fixed and moving narrow orifices. They did not use this guidance strategy in trajectory optimization. In [11] the authors posed the narrow gap passing requirement as a constraint on trajectory

acceleration at the waypoint. The proposed method is yaw angle dependent and the vertical acceleration is user specified. To optimize the trajectory as a whole, the narrow gap passing method should be independent of yaw angle and the vertical acceleration should also be a variable which is optimized in the framework.

For multi-copter trajectory generation and motion control, ideas such as multi-segment polynomial, minimum derivative optimization, and differential flatness inverse dynamics analysis are widely adopted [12–14]. To achieve an aggressive trajectory, methods such as segment time allocation [15] and spatial-temporal trajectory optimization [2,4,7] have been proposed. However, we do not yet have a synthetic architecture to find the most aggressive trajectory based on the waypoints, the requirement for vehicle heading and maneuvering at waypoints, and the full usage of rotor thrust for a multi-coper. Therefore in the first phase of this research, we would like to develop a complete synthesis to fill this gap. We adopt a simplified quadcopter dynamic model and develop a optimization framework that fully utilizes the available rotor thrust of the quadcopter when performing the most aggressive maneuver. The constraint on vehicle maneuvering at waypoints is posed as a window passing through problem and solved through the trajectory optimization as trajectory derivative constraints. The maximum rotor thrust for the trajectory is obtained through the inverse dynamics analysis. As the optimized aggressive trajectory is obtained, a geometric controller [16] is used to perform the polynomial trajectory tracking and verify the feasibility of the trajectory and the accuracy of the inverse dynamics analysis.

Phase 2 – Compensation for Additional Effects

Over the past decade multi-copter aggressive trajectory planning and accurate tracking control has been a popular research topic. For example in [17] the authors proposed the polynomial trajectory planning for quadrotors to fly aggressively through cluttered indoor environments, in [18] the authors proposed an optimization-based framework to find minimum

control effort trajectory subject to geometrical spatial constraints and user-defined dynamic constraints, and in [19] the authors proposed a novel control law for accurate tracking of aggressive quadcopter trajectories. In the majority of these works, a standard simplified multi-copter dynamic model had been used in which many factors such as aerodynamic, gyroscopic and rotor rolling effects are simply neglected. However, these effects could be significant in aggressive flights while it is reasonable to neglect them in mild flights. Recently, some works also took aerodynamic drag into account for example in [20] the authors proposed a drag-utilization scheme to improve the tracking performance, in [21] the authors developed a reachability control strategy with drag force considered, and in [22] the authors derived an accurate mathematical model for quadrotor UAVs based on the Euler-Lagrange formulation which also includes gyroscopic effects and aerodynamic drag. In the first phase of this research, we adopted the simplified dynamic model and proposed an optimal aggressive constrained trajectory synthesis to address the problem of how to generate an optimal aggressive trajectory subject to waypoint maneuver constraint and individual rotor thrust limitation. Therefore, in this next phase we would like to further incorporate these additional effects into the dynamic model, examine how they could affect the vehicle dynamics, and propose the way to compensate these effects both in the optimal constrained trajectory planning and in the geometric control trajectory tracking.

Regarding those additional effects, in [23] the authors proposed a way to prove that the dynamical model of a quadrotor subject to linear rotor drag effects is differentially flat in its position and heading. However, while it makes the inverse dynamics derivation easier, the simple linear drag model is somewhat impractical because the variation in rotor speed is not taken into account. As a result, they have to experimentally find different approximate value of the drag coefficient for different flight trajectory. For aerodynamic effects, the rotor drag model based on momentum and blade element theories is widely discussed for example in [24] and [25]. In [26], the authors proposed aerodynamic effect models

for induced drag and refined thrust, accurately identified the parameters experimentally, and improved the flight tracking error by compensating these effects. However, they only compensated the induced drag for the collective thrust and desired attitude in the trajectory tracking control, not further considering its effect on the rotational dynamics. Gyroscopic and rotor rolling effects are commonly considered secondary effects and often neglected as discussed in [27] and [28]. In general multi-copter configurations, there is usually equal number of clockwise and counter clockwise spinning identical rotors, and the gyroscopic and rotor rolling torques from one group of rotors tend to cancel out those from the other group especially in near-hover situation. However, in aggressive flight when the vehicle is in high dynamic motion (edgewise velocity and rolling/pitching rate) with significant yawing control effort applied, these two effects could become significant. For gyroscopic effect, the standard model is widely used as in [29] and [30]. For rotor rolling effect, in [31] and [32] the authors proposed a similar model which is proportional to the rotor speed and the edgewise velocity, and the experimental data in [33] is found supportive to this model. Some studies indicate that the flexibility of the propeller could partially turns the gyroscopic and rotor rolling effects into the blade flapping phenomenon [32,34]. In this research, we only consider relatively stiff propellers for small multi-copters and develop the compensation framework accordingly.

We adopted the models for aerodynamic effects proposed in [26], the standard gyroscopic torque model, and the rotor rolling model in [31] into our previous framework of optimal aggressive constrained trajectory synthesis [35]. Besides the full inverse dynamics derivation, the challenge also lies in the dependence of all the additional effects on individual rotor speeds. However, the resultant rotor speeds can only be obtained in the end of the analysis when all the additional effects are compensated and the individual rotor thrusts are determined. Likewise, in the attempt to compensate rotor drag in the trajectory derivative constraint for waypoint maneuver, the rotor speeds and vehicle velocity that decide the

rotor drag can only be determined when the trajectory polynomials are obtained after the optimization process. To address this problem, we utilize the refined thrust model in [26] and propose an iterative method which was proved practical and efficient in all of our test cases.

1.2 Objectives

In this research we assume that the dynamic capability of the multi-copter is limited by its maximum rotor thrust and address the problem as "**Given waypoint coordinates, heading and vehicle velocity and attitude constraints at these waypoints, how to find a feasible trajectory that fully utilizes the dynamic capability of a multi-copter to achieve the optimal aggressiveness and perform precise trajectory tracking control?**". The research objectives are listed as:

1. Synthesize optimal aggressive trajectories for a multi-copter with state, path and control constraints.
2. Synthesize inverse dynamics based trajectories that accommodate aerodynamic, gyroscopic, and rolling effects.
3. Estimate key vehicle mass-inertia-drag-rotor parameters using controlled flight experiments.
4. Verify synthesis methodology using high fidelity RotorS/PX4 SITL/Gazebo simulations with estimated vehicle parameters.
5. Verify synthesis methodology and perform trajectory and maneuver tracking using quadcopter flights.

1.3 Summary of Contributions

Phase 1 – Aggressive Constrained Trajectory Optimization

Our main contribution in this phase is the development of a complete synthesis framework to find the most aggressive trajectory that fully utilizes the available dynamic capability (maximum thrust) of the multi-copter and satisfies the constraints on the waypoints and the requirement for vehicle heading and maneuver at waypoints. This is achieved by integrating the constrained minimum snap quadratic program and the differential flatness based inverse dynamics analysis into a whole optimization framework. With this framework we can find the optimal polynomial trajectory and the corresponding segment time allocation for a given multi-copter model, a specified scenario, and a desired aggressiveness. *Unlike other methods, we actually track the maximum individual rotor thrust required during trajectory optimization.* This approach allows us to precisely manage the level of aggressiveness, ensuring it is not overly risky while still pursuing aggressiveness and, at the same time, avoiding excessive conservatism. We can also tell if the given scenario is beyond the capability of the given multi-copter by examining whether the solution exists. Additionally, a yaw trajectory optimization method based on our framework is proposed to improve the aggressive performance in the case of no requirement on heading angle. In this research we also compared and confirmed that minimum snap optimization is indeed the optimal choice for this approach of multi-copter aggressive trajectory optimization. This has not previously shown in the literature, though minimum snap or jerk trajectories are so commonly used by researchers.

Phase 2 – Compensation for Additional Effects

The main contribution in this phase is in the development of the complete framework to compensate the aerodynamic, gyroscopic and rotor rolling effects in the inverse dynamics analysis. *This framework shows how the differential flatness property can be preserved while introducing all these additional effects.* This method benefits both the precise es-

timation of the vehicle state and control input in the trajectory planning phase and the accurate computation of feedforward terms in the trajectory tracking phase. For trajectory planning, this method can be adopted for realistic flight estimation regardless the planning or optimization method used as long as the resultant timed trajectory is analytical. Based on this capability, we are also able to compensate the aerodynamic drag in the waypoint maneuver constraint and find optimal trajectories that are more precise and realistic. And for trajectory tracking, the compensation in feedforward terms improves the tracking performance. Without the compensation, the tracking accuracy and the controller flexibility could be sacrificed.

Phase 3 – High Fidelity Simulation, System Identification and Flight Test

In the high fidelity simulation, we verify our aggressive trajectory synthesis in a more realistic virtual environment. We also share our experience in establishing the comprehensive Software-in-the-Loop (SITL) simulation environment. Such a simulation is valuable because it allows for thorough testing of system integration, data communication, controller implementation, and even the detailed flight test procedure before conducting actual flights. Additionally, we present a comparison between the simulation results and the data obtained from actual flight tests to validate the degree of similarity.

During the system identification process, we have successfully developed a comprehensive approach to obtain all the parameters necessary for our quadcopter's dynamic model. The accurate identification of these parameters holds significant value for our current research and serves as a valuable reference for further study. Additionally, the flight test procedure used for identification, along with the data analysis and parameter estimation techniques employed, provides a helpful guideline for researchers undertaking similar investigations in the field.

During actual flight tests, we successfully achieved precise trajectory tracking by integrating our proposed trajectory tracking control and effects' compensation with the PX4

flight controller. This provides a hybrid approach to benefit from the robustness and stability of the PX4 flight controller while incorporating the inverse dynamics based trajectory tracking control. The feasibility of the optimal aggressive constrained trajectory synthesis is confirmed through actual narrow window passing flight tests. The actual RPM data in the flight tests also verifies the specification of maximum rotor thrust in our framework and proves the practical value of our trajectory optimization framework and inverse dynamics analysis.

List of Publications

- Published: Tsung-Liang Liu and Kamesh Subbarao, "Optimal aggressive constrained trajectory synthesis and control for multi-copters," Aerospace, vol. 9, no. 6, 2022, https://www.mdpi.com/2226-4310/9/6/281
- Under review: Tsung-Liang Liu and Kamesh Subbarao, "Inverse Dynamics based Aerodynamic, Gyroscopic, and Rotor Effects' Compensation in Constrained Trajectory Synthesis for Multi-Copters," Part G: Journal of Aerospace Engineering, 2023
- To be submitted: Tsung-Liang Liu and Kamesh Subbarao, "Parameter Identification, Modeling, and Control of Multicopters: Simulation and Experiments," ASME Journal of Autonomous Vehicles and Systems, 2023

1.4 Dissertation Outline

This dissertation is divided into 8 chapters. Chapter 2 presents the development of the complete aggressive constrained trajectory optimization framework which is based on a simplified quadcopter dynamic model. The geometric controller is used in MATLAB/Simulink simulations to verify the resultant optimized trajectory and the estimated vehicle dynamics. In Chapter 3, aerodynamic, gyroscopic and rotor rolling effects are incorporated into the dynamic model and the trajectory synthesis. The compensations for

these effects are proposed both in the trajectory optimization and the trajectory tracking control. Then we introduce the flight test environment. Key components/devices in the flight test system and important hardware/software integration setup are described in Chapter 4. Chapter 5 introduces two high fidelity simulations used in this research. Trajectory tracking control test is described, and comparison between simulation and actual flight data is provided. Chapter 6 discusses the quadcopter dynamic model parameter identification through CAD modeling, static rotor tests, and a series of flight tests. Chapter 7 carries out the actual aggressive trajectory tracking flight. We introduce the actual trajectory tracking control implementation, the controller tuning, the trajectory aggressiveness verification, and finally the narrow window passing flight verification. Lastly, the concluding remarks of this dissertation are presented in Chapter 8.

Chapter 2

Multi-Copter Aggressive Constrained Trajectory Optimization Framework and Tracking Control

While aggressive flight using multi-copters is widely discussed/mentioned, there is not much effort put toward defining aggressiveness and finding the most aggressive trajectory with complex maneuver requirements for a given vehicle model. Therefore, in this research, the problem is addressed as: Given waypoint coordinates, heading angles, and some vehicle velocity and attitude constraints at these waypoints, how can we find a feasible trajectory that fully utilizes the dynamic capability of a multi-copter to achieve the optimal aggressiveness? In this chapter we adopt a simplified quadcopter dynamic model and develop a synthesis framework that fully utilizes the available dynamic capability of the quadcopter when performing the most aggressive maneuver. A formal definition of aggressiveness will be provided in the latter sections. As the optimized aggressive trajectory is obtained, a geometric controller is used to perform the polynomial trajectory tracking and verify the feasibility of the trajectory and the accuracy of the inverse dynamics analysis.

2.1 Mathematical Model Description

A generic model for quadcopters is used in this phase. Note that the proposed approach can be applied to different configurations of multi-copters by modifying the vehicle force and moment model according to the desired rotor configuration and control allocation method. The coordinate systems and forces and moments generated by the rotors are shown in Figure 2.1.

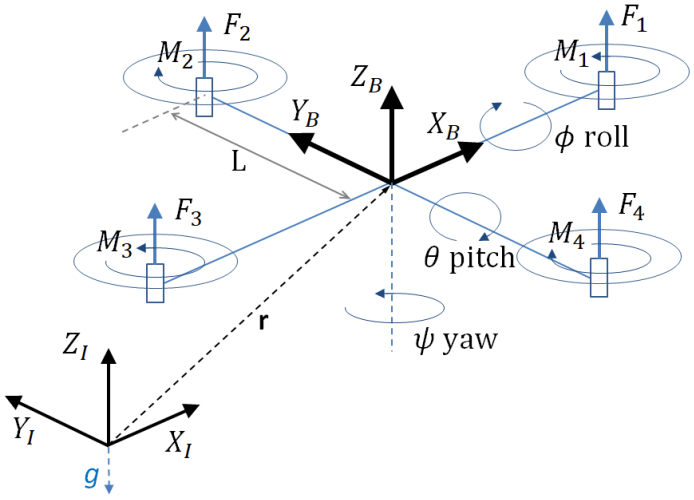

Figure 2.1. Reference frames and quadcopter forces/moments.

The body frame, B, is attached to the center of mass of the quadcopter and rotor 1 is on the positive X_B-axis. The gravitational acceleration g is in the $-Z_I$ direction of the inertial frame, I. Euler angles roll ϕ, pitch θ, and yaw ψ are used to define orientation from inertial frame to body frame. Note, $Z-X-Y$ rotation order is used here, and therefore the rotation matrix for transforming coordinates from B to I is given by:

$$\boldsymbol{R}_{IB} = \begin{bmatrix} \cos\psi\cos\theta - \sin\phi\sin\psi\sin\theta & -\cos\phi\sin\psi & \cos\psi\sin\theta + \cos\theta\sin\phi\sin\psi \\ \cos\theta\sin\psi + \cos\psi\sin\phi\sin\theta & \cos\phi\cos\psi & \sin\psi\sin\theta - \cos\psi\cos\theta\sin\phi \\ -\cos\phi\sin\theta & \sin\phi & \cos\phi\cos\theta \end{bmatrix} \tag{2.1}$$

The position vector of the quadcopter in the inertial frame is denoted by $\mathbf{r}$. With the gravity force acting in the $-Z_I$ direction and the forces of the rotors acting in the Z_B

direction, the equation governing the acceleration of the quadcopter with respect to inertial frame is given by:

$$m\ddot{\mathbf{r}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + \boldsymbol{R}_{IB} \begin{bmatrix} 0 \\ 0 \\ F_1 + F_2 + F_3 + F_4 \end{bmatrix} \tag{2.2}$$

With p, q, and r denoting the components of angular velocity of the quadcopter in the body frame, the rotational kinematics equation is given by:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & -\cos\phi\sin\theta \\ 0 & 1 & \sin\phi \\ \sin\theta & 0 & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{2.3}$$

Assuming rotors 1 and 3 rotate in the $-Z_B$ direction while 2 and 4 rotate in the Z_B direction, M_1 and M_3 act in the Z_B direction while M_2 and M_4 act in the $-Z_B$ direction since the moment produced by the rotor is opposite to the direction of rotation of the blade. With $\boldsymbol{I}$ denoting the moment of inertia of the quadcopter referenced to the center of mass and L denoting the distance from the axis of rotation of the rotors to the center of the quadcopter, the rotational dynamics equation is given by:

$$\boldsymbol{I} \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} L(F_2 - F_4) \\ L(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \boldsymbol{I} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.4}$$

Define the input $u = [u_1 \quad u_2 \quad u_3 \quad u_4]^T$ wherein u_1 is the total force from the rotors and u_2, u_3 and u_4 are the moments about X_B, Y_B and Z_B axes. Following [14], we assume that the force and moment produced by the ith rotor are proportional to the square of its rotational speed as:

$$F_i = k_f \omega_i^2 \ , \quad M_i = k_m \omega_i^2 \tag{2.5}$$

The relationship between the input and the angular speed of the rotors can be represented as:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} F_1 + F_2 + F_3 + F_4 \\ L(F_2 - F_4) \\ L(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix} = \begin{bmatrix} k_f & k_f & k_f & k_f \\ 0 & k_f L & 0 & -k_f L \\ -k_f L & 0 & k_f L & 0 \\ k_m & -k_m & k_m & -k_m \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \tag{2.6}$$

2.2 Solution Methodology

We use multi-segment polynomials to generate the trajectory and assure its smoothness by having trajectory derivative continuities. To optimize the polynomial coefficients, we form the quadratic problem with the cost on the trajectory derivative which is related to the actuator input of the vehicle. We accommodate waypoint velocity and attitude requirements in the optimization problem as equality constraints. To optimize the segment time between waypoints, we introduce an augmented cost on both trajectory derivative and total time. We connect the polynomial trajectory and actual quadcopter dynamics and rotor inputs by performing inverse dynamics analysis. We finally define aggressiveness and find the optimal trajectory with desired aggressiveness by tracking the desired maximum rotor force needed for the optimized trajectory. The whole methodology and the solution procedure are illustrated below in Figure 2.2.

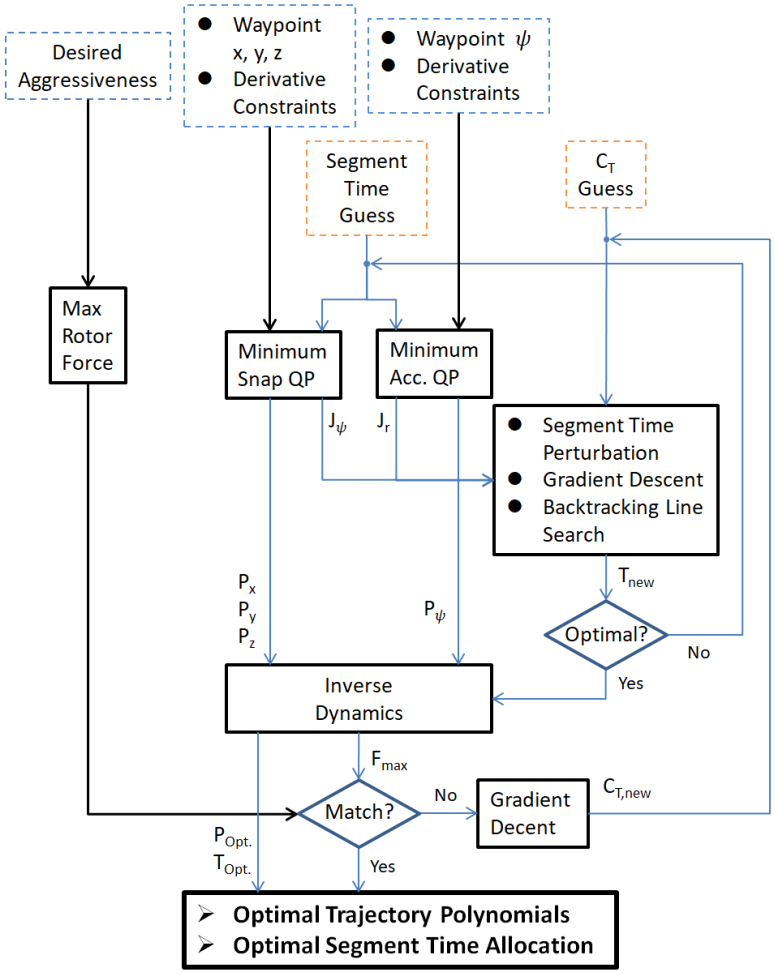

Figure 2.2. Methodology of the aggressive trajectory optimization.

2.2.1 Multi-Segment Polynomial Trajectory Optimization

Multi-segment polynomials are used to generate the trajectory. For each segment connecting one and the subsequent waypoint, independent polynomials $P_x(t)$, $P_y(t)$, $P_z(t)$, and $P_\psi(t)$ are used to represent the quadcopter states x, y, z, and ψ (yaw angle). Since the

polynomial framework is identical to the four vehicle states, we show the general procedure here. Each polynomial segment is represented as:

$$P(t) = p_0 + p_1 t + \cdots + p_{N-1} t^{N-1} + p_N t^N = \sum_{i=0}^{N} p_i t^i \tag{2.7}$$

Similarly, four independent costs J_x, J_y, J_z, and J_ψ are used to represent the cost on each vehicle states, and we show the general procedure here. The cost function on the integral of the quadratic of the trajectory derivatives is:

$$J = \int_0^T [c_0 P(t)^2 + c_1 P'(t)^2 + c_2 P''(t)^2 + \cdots + c_N P^{(N)}(t)^2] dt \tag{2.8}$$

where T is the flight time for the trajectory segment and c_r is user-specified penalty on the r^{th} derivative of the trajectory. This function can be written in matrix form as:

$$J = \bar{p}^T \boldsymbol{Q} \bar{p} \tag{2.9}$$

where $\bar{p} = [p_0 \quad p_1 \quad \cdots \quad p_N]^T$ and $\boldsymbol{Q}$ is the cost matrix. Following the formulation in [12], $\boldsymbol{Q}$ can be constructed as:

$$\text{for } r = 0, 1, \cdots N \ , \ i = 0, 1, \cdots N \ , \ l = 0, 1, \cdots N$$

$$\boldsymbol{Q}_r(i+1, l+1) = \begin{cases} [\prod_{m=0}^{r-1} (i-m)(l-m)] \frac{T^{i+l-2r+1}}{i+l-2r+1} & \text{if } i \geq r \wedge l \geq r \\ 0 & \text{if } i < r \vee l < r \end{cases} \tag{2.10}$$

$$\boldsymbol{Q} = \sum_{r=0}^{N} c_r \boldsymbol{Q}_r \tag{2.11}$$

In this research, to minimize the input needed to achieve optimal aggressiveness, we adopted the choice in [13] which is to minimize the snap (4th derivative) in trajectory x, y, and z while minimizing the acceleration (2nd derivative) in trajectory ψ. Therefore, we use $c_4 = 1$ while all other coefficients are set to zero for x, y, and z polynomial, and only $c_2 = 1$

for yaw polynomial. The constraints on the derivatives on the endpoints of a polynomial segment can be imposed as a linear function of the coefficients:

$$\begin{gathered} \boldsymbol{A}\bar{p} = \boldsymbol{b} \\ \boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_0 \\ \boldsymbol{A}_T \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} \boldsymbol{b}_0 \\ \boldsymbol{b}_T \end{bmatrix} \end{gathered} \tag{2.12}$$

where $\boldsymbol{A}$ is constructed by evaluating the components in the derivative formulations of the polynomial at $t = 0$ and $t = T$ corresponding to the appropriate coefficients as:

$$\boldsymbol{A}_{0_{rn}} = \begin{cases} \prod_{m=0}^{r-1}(n-m) & \text{if } r = n \\ 0 & \text{if } r \neq n \end{cases} \tag{2.13}$$

$$\boldsymbol{A}_{T_{rn}} = \begin{cases} [\prod_{m=0}^{r-1}(n-m)]T^{n-r} & \text{if } r \leq n \\ 0 & \text{if } r > n \end{cases} \tag{2.14}$$

We use the 0th order derivative constraint to specify the waypoint position. Higher order derivatives can be used to specify desired waypoint velocity, acceleration, etc., e.g., to enforce that the quadcopter starts from rest at the beginning of a trajectory. If not specified, these derivatives are subject to minimization of the cost function. Having assembled $\boldsymbol{Q}$, $\boldsymbol{A}$, and $\boldsymbol{b}$, the quadratic problem can be written below. There are methods to solve such a standard equality constrained QP [36]. In this research, the MATLAB solver "quadprog" is used.

$$\begin{aligned} \min_{\bar{p}} \quad & \bar{p}^T \boldsymbol{Q} \bar{p} \\ \text{s.t.} \quad & \boldsymbol{A}\bar{p} - \boldsymbol{b} = 0 \end{aligned} \tag{2.15}$$

For M polynomial segments, the joint optimization can be composed by concatenating their cost matrices in a block-diagonal fashion as:

$$J_{\text{joint}} = \begin{bmatrix} \bar{p}_1 \\ \vdots \\ \bar{p}_M \end{bmatrix}^T \begin{bmatrix} \boldsymbol{Q}_1(T_1) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \boldsymbol{Q}_M(T_M) \end{bmatrix} \begin{bmatrix} \bar{p}_1 \\ \vdots \\ \bar{p}_M \end{bmatrix} \tag{2.16}$$

The derivative constraints can also be concatenated in a block-diagonal fashion as:

$$\begin{bmatrix} \boldsymbol{A}_1(T_1) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \boldsymbol{A}_M(T_M) \end{bmatrix} \begin{bmatrix} \bar{p}_1 \\ \vdots \\ \bar{p}_M \end{bmatrix} = \boldsymbol{A}_{\text{der}} \begin{bmatrix} \bar{p}_1 \\ \vdots \\ \bar{p}_M \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_1 \\ \vdots \\ \boldsymbol{b}_M \end{bmatrix} \tag{2.17}$$

To have a feasible smooth trajectory, the derivatives should be continuous between segments. The continuity constraints must be imposed to ensure that the derivatives at the end of the ith segment match the derivatives at the beginning of the $(i+1)$th segment:

$$\boldsymbol{A}_{T,i}\bar{p}_i = \boldsymbol{A}_{0,i+1}\bar{p}_{i+1} \tag{2.18}$$

$$\begin{bmatrix} \boldsymbol{A}_{T,1} & -\boldsymbol{A}_{0,2} & 0 & 0 & \cdots & 0 & 0 \\ 0 & \boldsymbol{A}_{T,2} & -\boldsymbol{A}_{0,3} & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \boldsymbol{A}_{T,M-1} & -\boldsymbol{A}_{0,M} \end{bmatrix} \begin{bmatrix} \bar{p}_1 \\ \bar{p}_2 \\ \bar{p}_3 \\ \vdots \\ \bar{p}_{M-1} \\ \bar{p}_M \end{bmatrix} = A_{\text{con}} \begin{bmatrix} \bar{p}_1 \\ \bar{p}_2 \\ \bar{p}_3 \\ \vdots \\ \bar{p}_{M-1} \\ \bar{p}_M \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \tag{2.19}$$

These constraints can be compiled into a single set of linear equality constraints for the joint optimization problem:

$$\begin{bmatrix} \boldsymbol{A}_{\text{der}} \\ \boldsymbol{A}_{\text{con}} \end{bmatrix} \begin{bmatrix} \bar{p}_1 \\ \vdots \\ \bar{p}_M \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_1 \\ \vdots \\ \boldsymbol{b}_M \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{2.20}$$

2.2.2 Constraints on Vehicle Velocity and Attitude at Waypoints

In scenarios such that the quadcopter needs to pass through some narrow gap (e.g., a window) as shown in Figure 2.3, we can utilize the derivative constraints to achieve the objective.

Assume a waypoint at the center of the window and the window orientation is defined with $Z - X - Y$ Euler angles roll ϕ, pitch θ, and yaw ψ. The window forward vector in inertial frame can be obtained as:

$$\boldsymbol{w}_{\text{F}} = \boldsymbol{R}_{IB} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \cos\psi\cos\theta - \sin\phi\sin\psi\sin\theta \\ \cos\theta\sin\psi + \cos\psi\sin\phi\sin\theta \\ -\cos\phi\sin\theta \end{bmatrix} = \begin{bmatrix} W_{\text{F}_x} \\ W_{\text{F}_y} \\ W_{\text{F}_z} \end{bmatrix} \tag{2.21}$$

The window upward vector in inertial frame can be obtained as:

$$\boldsymbol{w}_{\text{U}} = \boldsymbol{R}_{IB} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\psi\sin\theta + \cos\theta\sin\phi\sin\psi \\ \sin\psi\sin\theta - \cos\psi\cos\theta\sin\phi \\ \cos\phi\cos\theta \end{bmatrix} = \begin{bmatrix} W_{\text{U}_x} \\ W_{\text{U}_y} \\ W_{\text{U}_z} \end{bmatrix} \tag{2.22}$$

To fly through the window safely, the velocity vector at the waypoint should be aligned with the window forward vector. Therefore, the cross product of the vectors should be zero, and the constraints for the waypoint 1st order derivatives can be composed as:

$$\boldsymbol{v} \times \boldsymbol{w}_{\mathrm{F}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \times \begin{bmatrix} W_{\mathrm{F}_x} \\ W_{\mathrm{F}_y} \\ W_{\mathrm{F}_z} \end{bmatrix} = \begin{bmatrix} \dot{y}W_{\mathrm{F}_z} - \dot{z}W_{\mathrm{F}_y} \\ \dot{z}W_{\mathrm{F}_x} - \dot{x}W_{\mathrm{F}_z} \\ \dot{x}W_{\mathrm{F}_y} - \dot{y}W_{\mathrm{F}_x} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.23}$$

From the equation of acceleration, we can deduce that $\boldsymbol{z}_B$ in inertial frame is in the direction of the vector $[\ddot{x} \quad \ddot{y} \quad \ddot{z} + g]^T$:

$$m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + \boldsymbol{R}_{IB} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \sum F \Rightarrow m \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} + g \end{bmatrix} = \boldsymbol{z}_B \cdot \sum F \tag{2.24}$$

For safely flying through the window, the $\boldsymbol{z}_B$ vector of the quadcopter should be aligned with the window upward vector. Therefore, the constraints for the waypoint 2nd order derivatives can be composed as:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} + g \end{bmatrix} \times \begin{bmatrix} W_{\mathrm{U}_x} \\ W_{\mathrm{U}_y} \\ W_{\mathrm{U}_z} \end{bmatrix} = \begin{bmatrix} \ddot{y}W_{\mathrm{U}_z} - (\ddot{z} + g)W_{\mathrm{U}_y} \\ (\ddot{z} + g)W_{\mathrm{U}_x} - \ddot{x}W_{\mathrm{U}_z} \\ \ddot{x}W_{\mathrm{U}_y} - \ddot{y}W_{\mathrm{U}_x} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.25}$$

$$\Rightarrow \begin{bmatrix} \ddot{y}W_{\mathrm{U}_z} - \ddot{z}W_{\mathrm{U}_y} \\ \ddot{z}W_{\mathrm{U}_x} - \ddot{x}W_{\mathrm{U}_z} \\ \ddot{x}W_{\mathrm{U}_y} - \ddot{y}W_{\mathrm{U}_x} \end{bmatrix} = \begin{bmatrix} gW_{\mathrm{U}_y} \\ -gW_{\mathrm{U}_x} \\ 0 \end{bmatrix} \tag{2.26}$$

For these constraints, the relationship between x, y, and z derivatives needs to be specified. Therefore, the joint optimization of x, y, and z should be composed by concatenating their cost and constraint matrices in a block-diagonal fashion as:

$$J_{xyz} = \begin{bmatrix} \bar{\bar{p}}_x \\ \bar{\bar{p}}_y \\ \bar{\bar{p}}_z \end{bmatrix}^T \begin{bmatrix} \boldsymbol{Q}_x & 0 & 0 \\ 0 & \boldsymbol{Q}_y & 0 \\ 0 & 0 & \boldsymbol{Q}_z \end{bmatrix} \begin{bmatrix} \bar{\bar{p}}_x \\ \bar{\bar{p}}_y \\ \bar{\bar{p}}_z \end{bmatrix} \tag{2.27}$$

$$\begin{bmatrix} \boldsymbol{A}_x & 0 & 0 \\ 0 & \boldsymbol{A}_y & 0 \\ 0 & 0 & \boldsymbol{A}_z \end{bmatrix} \begin{bmatrix} \bar{\bar{p}}_x \\ \bar{\bar{p}}_y \\ \bar{\bar{p}}_z \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_x \\ \boldsymbol{b}_y \\ \boldsymbol{b}_z \end{bmatrix} \tag{2.28}$$

where $\bar{\bar{p}}_x = [\bar{p}_{x_1} \ \cdots \ \bar{p}_{x_M}]^T$, $\bar{\bar{p}}_y = [\bar{p}_{y_1} \ \cdots \ \bar{p}_{y_M}]^T$ and $\bar{\bar{p}}_z = [\bar{p}_{z_1} \ \cdots \ \bar{p}_{z_M}]^T$. With this joint optimization setup, the constraints for a window passage on waypoint s in an M-segment trajectory can be implemented as:

$$\begin{bmatrix} 0 & W_{\mathrm{F}_z}\boldsymbol{v}_{0,s} & -W_{\mathrm{F}_y}\boldsymbol{v}_{0,s} \\ -W_{\mathrm{F}_z}\boldsymbol{v}_{0,s} & 0 & W_{\mathrm{F}_x}\boldsymbol{v}_{0,s} \\ W_{\mathrm{F}_y}\boldsymbol{v}_{0,s} & -W_{\mathrm{F}_x}\boldsymbol{v}_{0,s} & 0 \\ 0 & W_{\mathrm{U}_z}\boldsymbol{a}_{0,s} & -W_{\mathrm{U}_y}\boldsymbol{a}_{0,s} \\ -W_{\mathrm{U}_z}\boldsymbol{a}_{0,s} & 0 & W_{\mathrm{U}_x}\boldsymbol{a}_{0,s} \\ W_{\mathrm{U}_y}\boldsymbol{a}_{0,s} & -W_{\mathrm{U}_x}\boldsymbol{a}_{0,s} & 0 \end{bmatrix} \begin{bmatrix} \bar{\bar{p}}_x \\ \bar{\bar{p}}_y \\ \bar{\bar{p}}_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ gW_{\mathrm{U}_y} \\ -gW_{\mathrm{U}_x} \\ 0 \end{bmatrix} \tag{2.29}$$

$$\begin{cases} \boldsymbol{v}_{0,s} = \begin{bmatrix} [0_{\times(n+1)}]_{\times(s-1)} & \boldsymbol{v}_0 & [0_{\times(n+1)}]_{\times(M-s)} \end{bmatrix} \\ \boldsymbol{a}_{0,s} = \begin{bmatrix} [0_{\times(n+1)}]_{\times(s-1)} & \boldsymbol{a}_0 & [0_{\times(n+1)}]_{\times(M-s)} \end{bmatrix} \end{cases} \tag{2.30}$$

For a segment polynomial $P(t) = p_0 + p_1 t^1 + p_2 t^2 + p_3 t^3 + \cdots + p_N t^N$, the 1st order derivative (velocity) at $t = 0$ is $\boldsymbol{v}_0 \bar{p}$ and $\boldsymbol{v}_0 = [0 \ 1 \ 0 \ 0 \ \cdots \ 0]$. The 2nd order derivative (acceleration) at $t = 0$ is $\boldsymbol{a}_0 \bar{p}$ and $\boldsymbol{a}_0 = [0 \ 0 \ 2 \ 0 \ \cdots \ 0]$.

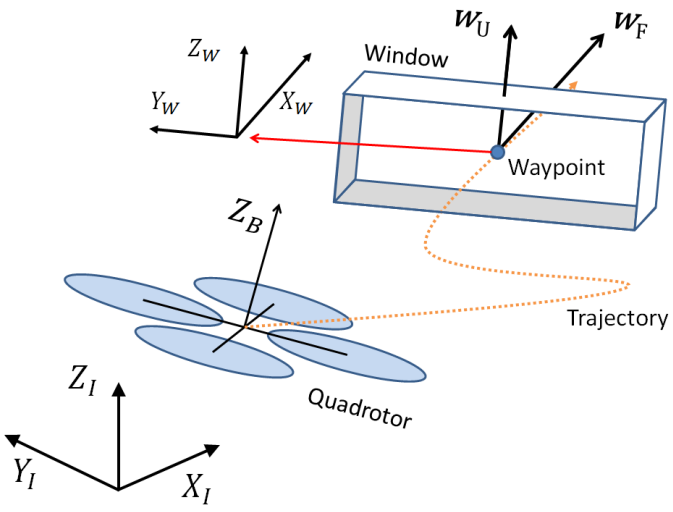

Figure 2.3. Relevant frames for quadcopter and window constrains.

2.2.3 Segment Time Optimization

The above optimization finds the optimal trajectory polynomials for the given segment times. However, in general cases, we do not specify segment times, and instead, we would like to also optimize them to achieve better aggressiveness. To optimize segment times, we modify the cost function in the form:

$$J_T = J_{org} + c_T \sum_{i=1}^{M} T_i \tag{2.31}$$

where J_{org} combines the original cost on x, y, z, and ψ while c_T is a penalty on total time. The cost on x, y, z is the integral of square of derivatives on distance while the cost on ψ is on angle. To combine the costs, two coefficients μ_r and μ_ψ are introduced to non-dimensionalize the costs.

$$J_{org} = \mu_r J_{xyz} + \mu_\psi J_\psi \tag{2.32}$$

In this research, we minimize the 4th derivative in trajectory x, y, and z while minimizing the 2nd derivative in trajectory yaw angle. Therefore, the above equation can be represented as:

$$\begin{aligned} J_{org} =& \mu_r \int_0^{T_f} \left[\left(\frac{d^4x}{dt^4} \right)^2 + \left(\frac{d^4y}{dt^4} \right)^2 + \left(\frac{d^4z}{dt^4} \right)^2 \right] dt \\ &+ \mu_\psi \int_0^{T_f} \left(\frac{d^2\psi}{dt^2} \right)^2 dt \end{aligned} \tag{2.33}$$

where μ_r and μ_ψ are defined as:

$$\mu_r = \left[\frac{1}{\max\left(\left| \frac{d^4x}{dt^4} \right|, \left| \frac{d^4y}{dt^4} \right|, \left| \frac{d^4z}{dt^4} \right| \right)} \right]^2 \tag{2.34}$$

$$\mu_\psi = \left[\frac{1}{\max\left(\left| \frac{d^2\psi}{dt^2} \right| \right)} \right]^2 \tag{2.35}$$

We find μ_r and μ_ψ in the first iteration with initial segment times and then use them as constants throughout the optimization process. Otherwise, the variation of these two parameters will affect the convergence of the optimization. With the cost function defined, we perturb each segment time by some δt to obtain the gradient of the cost function with respect to each segment time. This is then used in a gradient descent method to find the time allocation for the minimum cost iteratively.

$$\bar{T} = \begin{bmatrix} T_1 & \cdots & T_M \end{bmatrix}^T \tag{2.36}$$

$$\nabla_i J_T = \frac{J_T \begin{bmatrix} T_1 & \cdots & T_i + \delta t & \cdots & T_M \end{bmatrix}^T - J_T(\bar{T})}{\delta t} \tag{2.37}$$

$$T_i^{new} = T_i + \alpha \Delta T_i \ , \quad \Delta T_i = -\frac{\nabla_i J_T}{\| \nabla_i J_T \|} \tag{2.38}$$

Since this is a high-dimensional problem with a complex cost function, the step size α can easily become too small or too large during the iterations and lead to slow conver-

gence or even divergence. For numerical efficiency, stability, and convergence, we use the backtracking line search method [37] to find a suitable step size α in every iteration.

$$\text{while } J(\bar{T} + \alpha\Delta\bar{T}) > J(\bar{T}) + \epsilon\alpha\nabla J_T^T \Delta\bar{T} \ , \quad \alpha \coloneqq \beta\alpha \tag{2.39}$$

With $\beta = 0.5$ and $\epsilon = 0.0001$, we have fast and stable convergence in our test cases. Figure 2.4 shows the result of segment time optimization for a simple 2D 4-waypoint scenario. The initial segment time is $\bar{T} = [1 \quad 3 \quad 1]^T$ s, and we optimize it with two different c_T values, 50 and 5000. The result trajectories look identical because they have similar segment time distribution while the total time turned out to be 6.28 and 3.53 seconds, respectively. It is observed that, with this segment time optimization process, the optimal segment time distribution will be found to minimize the cost on integral of the square of trajectory derivatives, while the optimal total time is found based on the value of c_T. The larger c_T used, the smaller the total time will be, which makes the trajectory more aggressive.

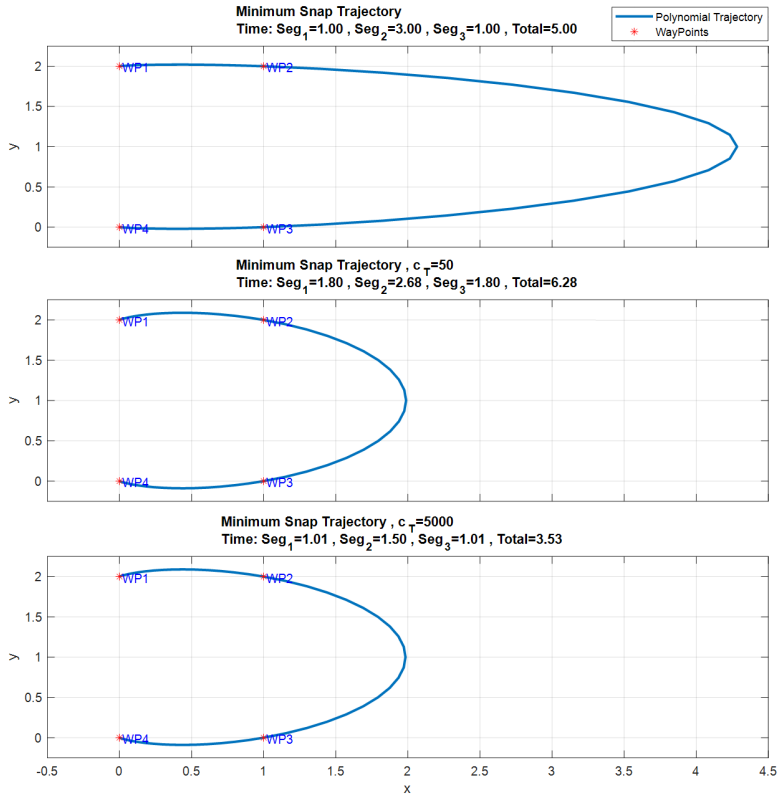

Figure 2.4. Comparison of trajectories optimized with different c_T.

2.2.4 Inverse Dynamics

To connect the polynomial trajectory optimization framework and actual quadcopter dynamics, we perform the inverse dynamics analysis. The differential flatness method is widely adopted in inverse dynamics analysis for multi-copters [14]. In this process, we will find all the states and inputs of the quadcopter according to the trajectory x, y, z, ψ, and their derivatives. For orientation ϕ and θ, first from the equation of acceleration we have:

$$\boldsymbol{z}_B = \frac{\begin{bmatrix} \ddot{x} & \ddot{y} & \ddot{z}+g \end{bmatrix}^T}{\| \begin{bmatrix} \ddot{x} & \ddot{y} & \ddot{z}+g \end{bmatrix} \|}, \quad u_1 = m \| \begin{bmatrix} \ddot{x} & \ddot{y} & \ddot{z}+g \end{bmatrix} \| \tag{2.40}$$

Assume $\boldsymbol{x}_C$ is the vector obtained by rotating $\boldsymbol{x}_I$ around $\boldsymbol{z}_I$ by yaw angle ψ:

$$\boldsymbol{x}_C = \begin{bmatrix} \cos\psi & \sin\psi & 0 \end{bmatrix}^T \tag{2.41}$$

We can determine $\boldsymbol{x}_B$ and $\boldsymbol{y}_B$ by:

$$\boldsymbol{y}_B = \frac{\boldsymbol{z}_B \times \boldsymbol{x}_C}{\|\boldsymbol{z}_B \times \boldsymbol{x}_C\|} , \quad \boldsymbol{x}_B = \boldsymbol{y}_B \times \boldsymbol{z}_B \tag{2.42}$$

With vehicle body frame defined, we can determine the rotation matrix and roll and pitch angles by:

$$\boldsymbol{R}_{IB} = \begin{bmatrix} \boldsymbol{x}_B & \boldsymbol{y}_B & \boldsymbol{z}_B \end{bmatrix} \tag{2.43}$$

$$\phi = \arcsin\left[\boldsymbol{R}_{IB}(3,2)\right] , \quad \theta = \arctan\left[\frac{-\boldsymbol{R}_{IB}(3,1)}{\boldsymbol{R}_{IB}(3,3)}\right] \tag{2.44}$$

For angular velocity p, q, and r, first we take the 1st derivative of the equation of acceleration:

$$m\dot{\boldsymbol{a}} = \dot{u}_1 \boldsymbol{z}_B + \boldsymbol{\omega}_B \times u_1 \boldsymbol{z}_B \tag{2.45}$$

Substituting $\dot{u}_1 = \boldsymbol{z}_B \cdot m\dot{\boldsymbol{a}}$ we have:

$$\boldsymbol{\omega}_B \times \boldsymbol{z}_B = \frac{m}{u_1}[\dot{\boldsymbol{a}} - (\boldsymbol{z}_B \cdot \dot{\boldsymbol{a}})\boldsymbol{z}_B] \tag{2.46}$$

With $\dot{\boldsymbol{a}} = [\,\dddot{x} \quad \dddot{y} \quad \dddot{z}\,]$, the RHS is known. With $\boldsymbol{x}_B$, $\boldsymbol{y}_B$, and $\boldsymbol{z}_B$ being unit vectors, $\boldsymbol{\omega}_B \times \boldsymbol{z}_B$ can be considered as the projection of $\boldsymbol{\omega}_B$ onto the $\boldsymbol{x}_B - \boldsymbol{y}_B$ plane with 90° shift. Therefore, the body angular velocities p and q can be determined as:

$$p = -(\boldsymbol{\omega}_B \times \boldsymbol{z}_B) \cdot \boldsymbol{y}_B , \quad q = (\boldsymbol{\omega}_B \times \boldsymbol{z}_B) \cdot \boldsymbol{x}_B \tag{2.47}$$

From the rotational kinematics equation, we have:

$$\begin{aligned} \dot{\psi} &= -\frac{\sin\theta}{\cos\phi}p + \frac{\cos\theta}{\cos\phi}r \\ \Rightarrow r &= \left(\dot{\psi} + \frac{\sin\theta}{\cos\phi}p\right)\frac{\cos\phi}{\cos\theta} \end{aligned} \tag{2.48}$$

With p, q, and r solved, we have $\boldsymbol{\omega}_B = \boldsymbol{R}_{IB}[p\ q\ r]^T$, and $\dot{\theta}$ and $\dot{\phi}$ can also be obtained from the inversion of the rotational kinematics equation. For angular acceleration $\dot{p}$, $\dot{q}$ and $\dot{r}$, first we take the 2nd derivative of the equation of acceleration:

$$\begin{aligned} m\ddot{\boldsymbol{a}} &= \ddot{u}_1\boldsymbol{z}_B + 2\boldsymbol{\omega}_B \times \dot{u}_1\boldsymbol{z}_B + \boldsymbol{\omega}_B \times \boldsymbol{\omega}_B \times u_1\boldsymbol{z}_B + \dot{\boldsymbol{\omega}}_B \times u_1\boldsymbol{z}_B \\ \Rightarrow \dot{\boldsymbol{\omega}}_B \times \boldsymbol{z}_B &= (m\ddot{\boldsymbol{a}} - \ddot{u}_1\boldsymbol{z}_B - 2\boldsymbol{\omega}_B \times \dot{u}_1\boldsymbol{z}_B - \boldsymbol{\omega}_B \times \boldsymbol{\omega}_B \times u_1\boldsymbol{z}_B)/u_1 \end{aligned} \tag{2.49}$$

With $\ddot{\boldsymbol{a}} = [\dddot{x} \quad \dddot{y} \quad \dddot{z}]$, $\ddot{u}_1 = \boldsymbol{z}_B \cdot (m\ddot{\boldsymbol{a}} - \boldsymbol{\omega}_B \times \boldsymbol{\omega}_B \times u_1\boldsymbol{z}_B)$ and $\dot{u}_1 = \boldsymbol{z}_B \cdot m\dot{\boldsymbol{a}}$, the RHS is known and the body angular accelerations $\dot{p}$ and $\dot{q}$ can be determined as:

$$\dot{p} = -(\dot{\boldsymbol{\omega}}_B \times \boldsymbol{z}_B) \cdot \boldsymbol{y}_B\ , \quad \dot{q} = (\dot{\boldsymbol{\omega}}_B \times \boldsymbol{z}_B) \cdot \boldsymbol{x}_B \tag{2.50}$$

Taking derivative of previous $\dot{\psi}$ equation, we have:

$$\begin{aligned} \ddot{\psi} &= -\frac{\sin\theta}{\cos\phi}\dot{p} + \frac{\cos\theta}{\cos\phi}\dot{r} - p\frac{d}{dt}\left(\frac{\sin\theta}{\cos\phi}\right) + r\frac{d}{dt}\left(\frac{\cos\theta}{\cos\phi}\right) \\ \Rightarrow \dot{r} &= \frac{\cos\phi}{\cos\theta}\left[\ddot{\psi} + \frac{\sin\theta}{\cos\phi}\dot{p} + p\left(\frac{\cos\theta\cos\phi\dot{\theta} + \sin\theta\sin\phi\dot{\phi}}{\cos^2\phi}\right)\right. \\ &\qquad \left. -r\left(\frac{\cos\theta\sin\phi\dot{\phi} - \sin\theta\cos\phi\dot{\theta}}{\cos^2\phi}\right)\right] \end{aligned} \tag{2.51}$$

Next we determine the moment inputs from obtained angular velocity and acceleration by the rotational dynamics equation:

$$\begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix} = \boldsymbol{I}\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} + \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \boldsymbol{I}\begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.52}$$

Now we have found all the states and inputs of the quadcopter derived from the trajectory x, y, z, ψ, and their derivatives. We can further determine the angular speed and force produced of each rotor by:

$$\begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} = \begin{bmatrix} k_f & k_f & k_f & k_f \\ 0 & k_f L & 0 & -k_f L \\ -k_f L & 0 & k_f L & 0 \\ k_m & -k_m & k_m & -k_m \end{bmatrix}^{-1} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}, \quad F_i = k_f \omega_i^2 \tag{2.53}$$

2.2.5 Max Force Tracking and Aggressiveness Defined

We find the maximum rotor force needed for the trajectory by the inverse dynamics analysis throughout the whole trajectory and observe how the maximum force and the trajectory total time vary with different c_T values. Figure 2.5 shows the analysis result for a 3D 4-waypoint scenario. It is observed that, with the increment in c_T, the total time decreases smoothly while the maximum force rises smoothly with increasing slope. With this relationship, we can track a particular total time or maximum force for a given scenario by finding a corresponding c_T. This is done with gradient decent method from an initial c_T by making a small perturbation on c_T and finding the gradient. Because the rotor cannot generate negative thrust, we should also track the minimum thrust during the process. If the minimum thrust reaches zero before the maximum thrust reaches desired value, we should stop the process and call this maximum thrust (corresponding to zero minimum thrust) a limitation due to the scenario setting.

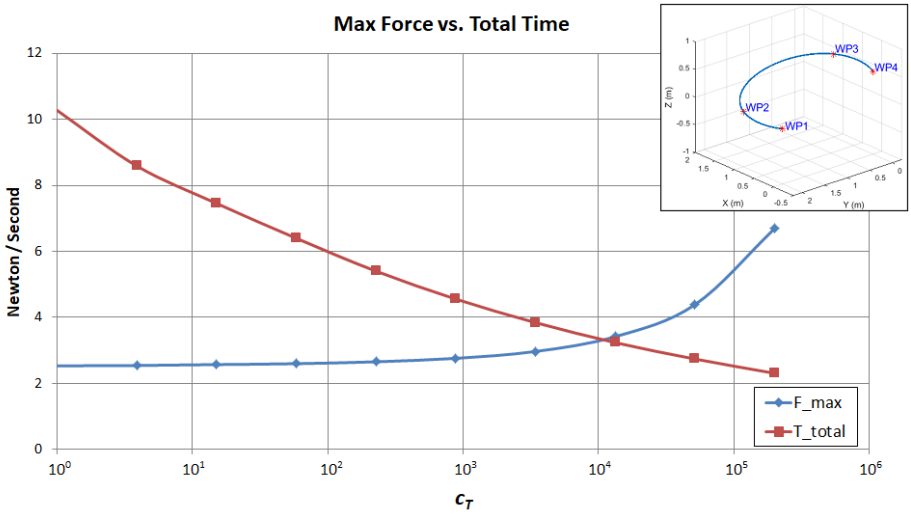

Figure 2.5. Comparison of maximum rotor force and total time for trajectories optimized with different c_T.

With optimal aggressiveness, the quadcopter should finish the tasks in the shortest time possible by means of its dynamic capability. Generally speaking, the dynamic capability of a quadcopter is restricted by the maximum thrust of its rotors. Therefore, we define aggressiveness as the percentage of excess thrust required for the time optimized minimum snap trajectory.

$$\text{Aggressiveness} = \frac{F_{max\_req} - F_{hover}}{F_{max} - F_{hover}} \cdot 100\% \tag{2.54}$$

where F_{max_req} is the maximum rotor force required for the trajectory, F_{hover} is the rotor force for steady hover, and F_{max} is the maximum thrust of the rotor. One might conserve the aggressiveness for safety reasons. Given waypoints x, y, z, ψ, and the desired aggressiveness, the whole methodology to find the optimal trajectory and segment time allocation for a given quadcopter model is summarized in Figure 2.2.

2.3 Numerical Results

The quadcopter parameters we use in this research are from [38] and tabulated below in Table 2.1.

Table 2.1. Quadcopter parameters

m	1.023 kg	g	$9.81\ \mathrm{m/s^2}$
L	0.2223 m	I_{xx}	$0.0095\ \mathrm{kg \cdot m^2}$
k_f	$1.4865 \cdot 10^{-7}\ \mathrm{N/RPM^2}$	I_{yy}	$0.0095\ \mathrm{kg \cdot m^2}$
k_m	$2.9250 \cdot 10^{-9}\ \mathrm{N \cdot m/RPM^2}$	I_{zz}	$0.0186\ \mathrm{kg \cdot m^2}$

We test the optimization framework with a four-waypoint scenario. The waypoint settings are tabulated below (yaw angles at waypoints 2 and 3 are not specified) in Table 2.2.

Table 2.2. Waypoint settings in the scenario.

WPT Setting	$\boldsymbol{x}$ (m)	$\boldsymbol{y}$ (m)	$\boldsymbol{z}$ (m)	$\boldsymbol{\psi}$
Waypoint 1	0	2	0	0
Waypoint 2	1	2	0	-
Waypoint 3	1	0	0.5	-
Waypoint 4	0	0	0.5	-180°

We specify the mission to be from rest to rest, therefore the velocity and acceleration at the first and the last waypoint are constrained to be zero. There are two narrow windows to pass through in this scenario. The window settings are tabulated below in Table 2.3.

Table 2.3. Window (WDW) settings in the scenario.

WDW Setting	ϕ	θ	ψ	**Location**
Window 1	$0°$	$15°$	$0°$	Waypoint 2
Window 2	$-30°$	$0°$	$-20°$	Waypoint 3

We specify the aggressiveness to be 80%. For this quadcopter model the rotor force for steady hover is $F_{hover} = 2.5$ N. Assuming $F_{max} = 3.75$ N, we have the maximum rotor force we can use for the trajectory as $F_{max\_req} = 3.5$ N. With initial guess of $\bar{T} = [5 \quad 5 \quad 5]^T$ s and $c_T = 100$, we use 10th order segment polynomials and have derivative continuity constraints on up to 6th order derivative (Pop). Figure 2.6 shows the scenario and the optimal trajectory.

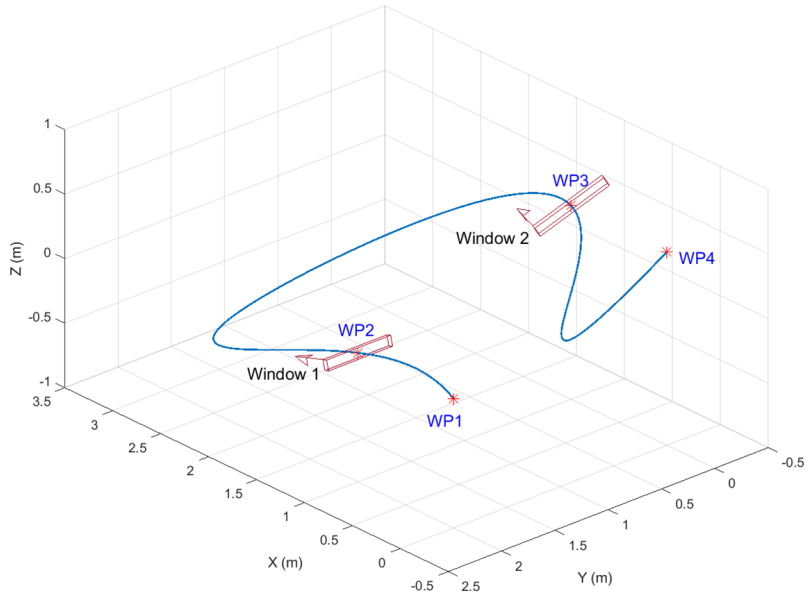

Figure 2.6. Plot of the test scenario and the optimal trajectory.

Figures 2.7–2.9 show the result of the inverse dynamics analysis from the trajectory with respect to the particular quadcopter model. The magenta dashed lines mark the times of waypoint passages. It can be noticed that the quadcopter had a spike in angular acceleration and angular rate around waypoint 3 because it was making the maneuver to pass through the highly tilted window 2.

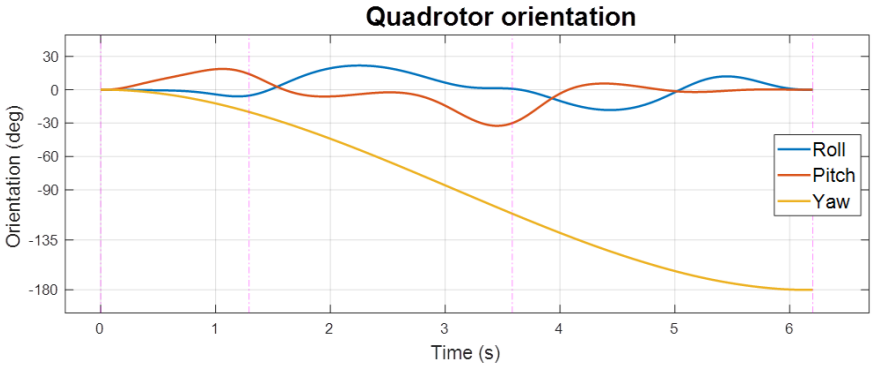

Figure 2.7. Vehicle Euler angles along the trajectory flight.

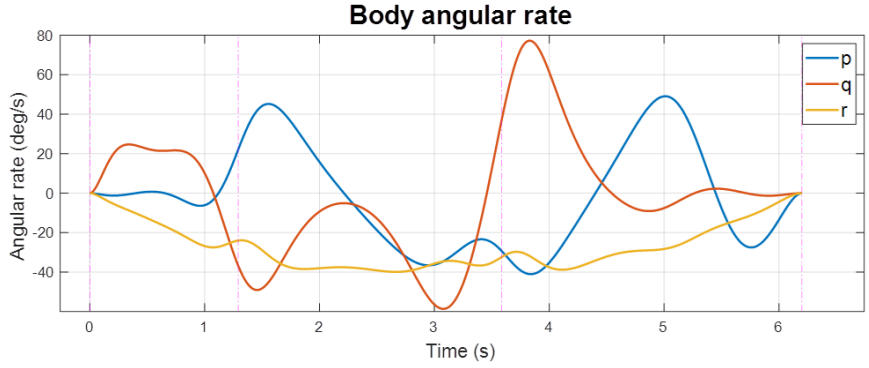

Figure 2.8. Vehicle body angular rate along the trajectory flight.

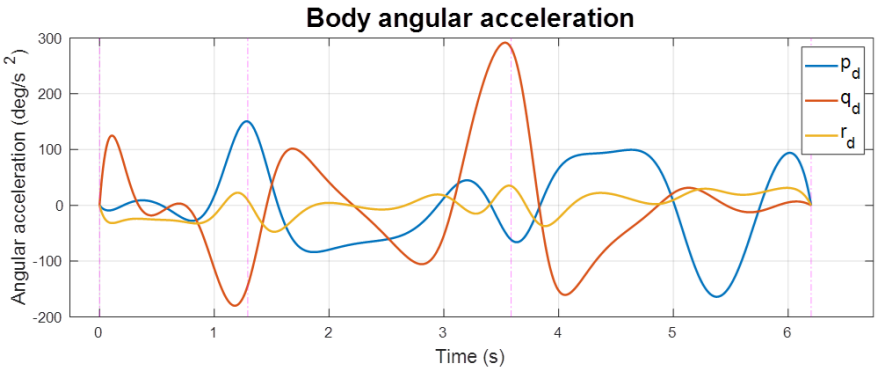

Figure 2.9. Vehicle body angular acceleration along the trajectory flight.

Figures 2.10 and 2.11 show the control input and the rotor thrust along the trajectory. From the plot, we confirmed that the maximum rotor thrust used is 3.5 N as expected.

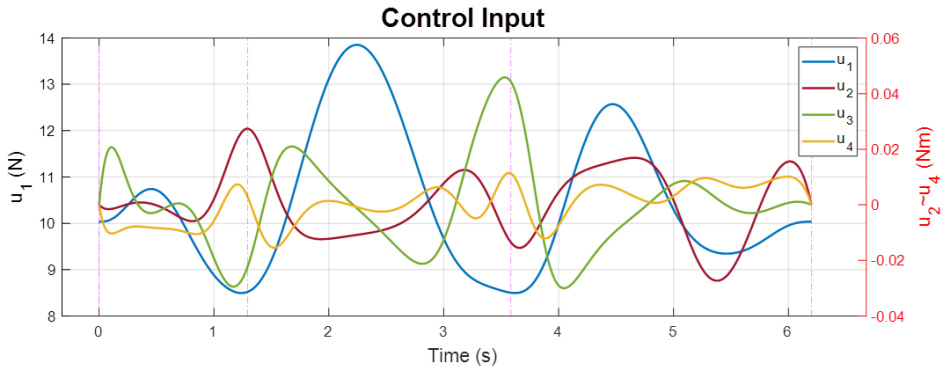

Figure 2.10. Control input along the trajectory flight.

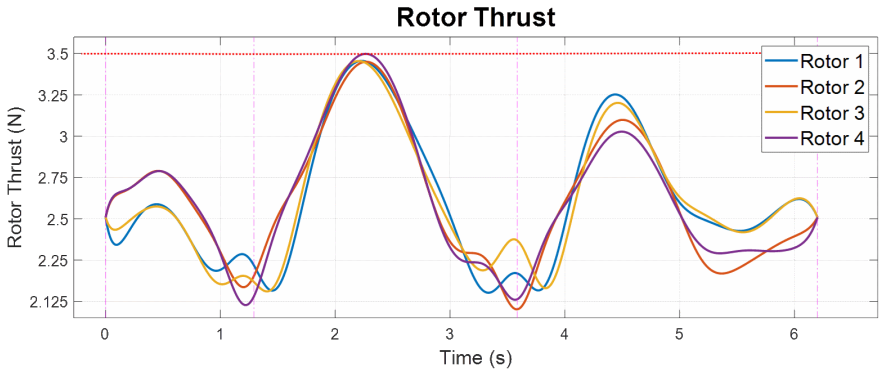

Figure 2.11. Rotor thrust along the trajectory flight.

Figure 2.12 shows the quadcopter on the trajectory with a fixed time interval of 0.33 s. It can be observed that the quadcopter had the highest speed in segment 2 so it can maneuver through the highly tilted window 2. The successful passage through the narrow windows can be visually confirmed. If we further check the data at waypoint 3, we have the window forward and upward vectors as $\boldsymbol{w}_F = [0.939 \quad -0.342 \quad 0]^T$ and $\boldsymbol{w}_F = [0.171 \quad 0.469 \quad 0.866]^T$. Additionally, the quadcopter velocity and thrust vectors (aligned with $\boldsymbol{z}_B$) are $\boldsymbol{v} = [-1.777 \quad 0.6470 \quad 0]^T$ and $\boldsymbol{f} = [1.454 \quad 3.995 \quad 7.364]^T$. The velocity and thrust vectors are confirmed to be aligned with the window forward and upward vectors. Finally, the optimal time allocation was found as $\bar{T}_{opt} = [1.29 \quad 2.29 \quad 2.62]^T$ s.

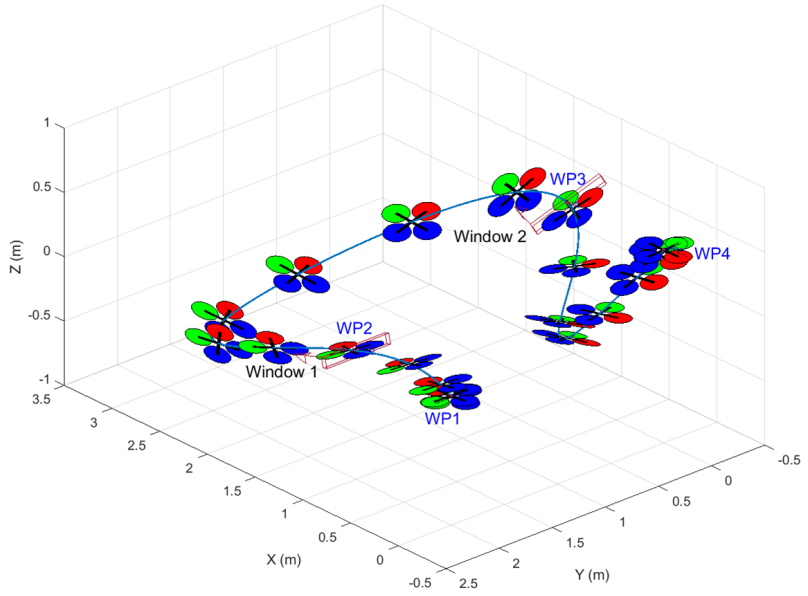

Figure 2.12. Plot of the quadcopter along the trajectory flight.

2.4 Geometric Control and Simulation Result

To perform the polynomial trajectory tracking and verify the result of the aggressive trajectory optimization, we adopted the geometric controller proposed by Lee et al. in [16] and implemented it in MATLAB/Simulink. The controller is constructed in two parts, trajectory tracking and attitude tracking. The control input of total force f is obtained by the trajectory tracking part with:

$$f = -(-k_x \boldsymbol{e}_x - k_v \boldsymbol{e}_v - mg\boldsymbol{e}_3 + m\ddot{\boldsymbol{x}}_d) \cdot \boldsymbol{R}\boldsymbol{e}_3 \tag{2.55}$$

The control input of the moments is obtained by the attitude tracking part with:

$$\boldsymbol{M} = -k_R \boldsymbol{e}_R - k_\omega \boldsymbol{e}_\omega + \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega} - I(\hat{\boldsymbol{\omega}}\boldsymbol{R}^T\boldsymbol{R}_d\boldsymbol{\omega}_d - \boldsymbol{R}^T\boldsymbol{R}_d\dot{\boldsymbol{\omega}}_d) \tag{2.56}$$

where vector $\boldsymbol{e}_3$ defines the direction of gravity and $\boldsymbol{R}$ is the rotation matrix from body to inertial frame. The tracking errors for position $\boldsymbol{e}_x$, velocity $\boldsymbol{e}_v$, attitude $\boldsymbol{e}_R$, and angular velocity $\boldsymbol{e}_\omega$ are defined as:

$$\begin{aligned} \boldsymbol{e}_x &= \boldsymbol{x} - \boldsymbol{x}_d \\ \boldsymbol{e}_v &= \boldsymbol{v} - \boldsymbol{v}_d \end{aligned} \tag{2.57}$$

$$\begin{aligned} \boldsymbol{e}_R &= \frac{1}{2}(\boldsymbol{R}_d^T \boldsymbol{R} - \boldsymbol{R}^T \boldsymbol{R}_d)^\vee \\ \boldsymbol{e}_\omega &= \boldsymbol{\omega} - \boldsymbol{R}^T \boldsymbol{R}_d \boldsymbol{\omega}_d \end{aligned} \tag{2.58}$$

where the $hat\ map\ ^\wedge : \mathbb{R}^3 \to \mathrm{SO}(3)$ is defined by the condition that $\hat{\boldsymbol{x}}\boldsymbol{y} = \boldsymbol{x} \times \boldsymbol{y}$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^3$. Additionally, the $vee\ map\ ^\vee : \mathrm{SO}(3) \to \mathbb{R}^3$ is the inverse of the hat map. At a given moment, $\boldsymbol{x}$, $\boldsymbol{v}$, $\boldsymbol{\omega}$, and $\boldsymbol{R}$ represent the position, linear velocity, body angular velocity, and rotation matrix of the vehicle. The corresponding desired position $\boldsymbol{x}_d$ and velocity $\boldsymbol{v}_d$ are captured from the polynomial trajectory. The desired body z axis can be obtained as:

$$\boldsymbol{z}_d = -\frac{-k_x \boldsymbol{e}_x - k_v \boldsymbol{e}_v - mg\boldsymbol{e}_3 + m\ddot{\boldsymbol{x}}_d}{\| - k_x \boldsymbol{e}_x - k_v \boldsymbol{e}_v - mg\boldsymbol{e}_3 + m\ddot{\boldsymbol{x}}_d \|} \tag{2.59}$$

With the desired body z axis, yaw angle, and derivatives of the polynomial trajectory available, we can find the desired rotation matrix $\boldsymbol{R}_d$, angular velocity $\boldsymbol{\omega}_d$, and angular acceleration $\dot{\boldsymbol{\omega}}_d$ by following the process discussed previously in inverse dynamics. The simulation is implemented with the following equations of motion:

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{v} \\ m\dot{\boldsymbol{v}} &= mg\boldsymbol{e}_3 - f\boldsymbol{R}\boldsymbol{e}_3 \end{aligned} \tag{2.60}$$

$$\begin{aligned} \dot{\boldsymbol{R}} &= \boldsymbol{R}\hat{\boldsymbol{\omega}} \\ \boldsymbol{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega} &= \boldsymbol{M} \end{aligned} \tag{2.61}$$

With 10% error in estimated inertia, ± 1 N noise added to f and ± 0.003 Nm noise added to M (around 10% of the maximum input used) as control input disturbance, we have the simulation result shown in figures below.

From the simulation result we can see that, despite the existence of inertia estimation error and control input disturbance, the geometric control had a good performance in tracking the polynomial trajectory. Position and yaw angle are well tracked and roll and pitch angles turned out to be very close to the inverse dynamics estimation as shown in Figures 2.13 and 2.14. Therefore, the successful passage through the narrow windows is confirmed. Figures 2.15 and 2.16 show that the linear acceleration is well tracked and the angular acceleration is very close to the estimation. From Figure 2.17, the control input used to track this polynomial trajectory is verified to be very close to the inverse dynamics estimation. The maximum rotor thrust used in the simulation is also confirmed to be around the maximum rotor force (3.5 N) requirement we specified in the trajectory generation as shown in Figure 2.18.

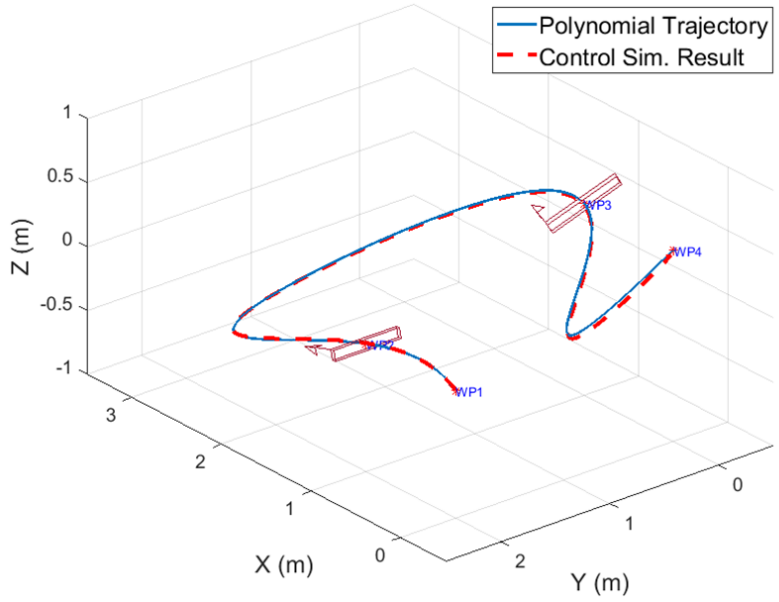

Figure 2.13. Trajectory tracking result.

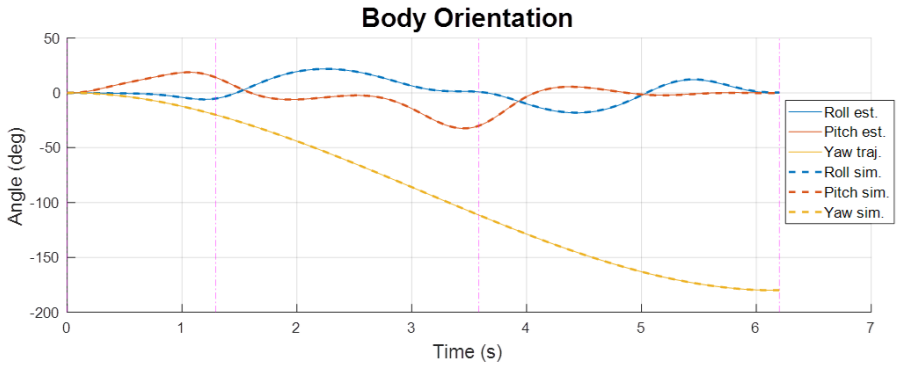

Figure 2.14. Vehicle Euler angle comparison.

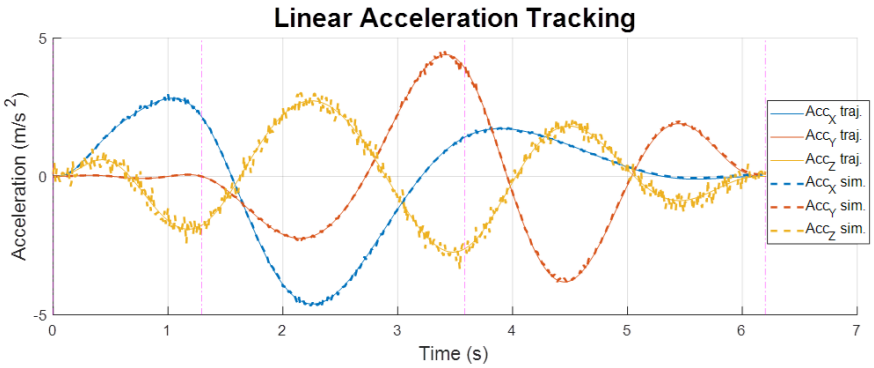

Figure 2.15. Vehicle linear acceleration tracking.

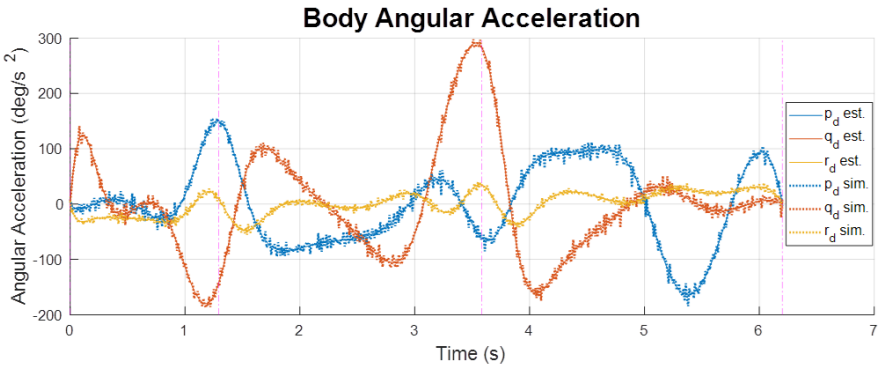

Figure 2.16. Vehicle angular acceleration comparison.

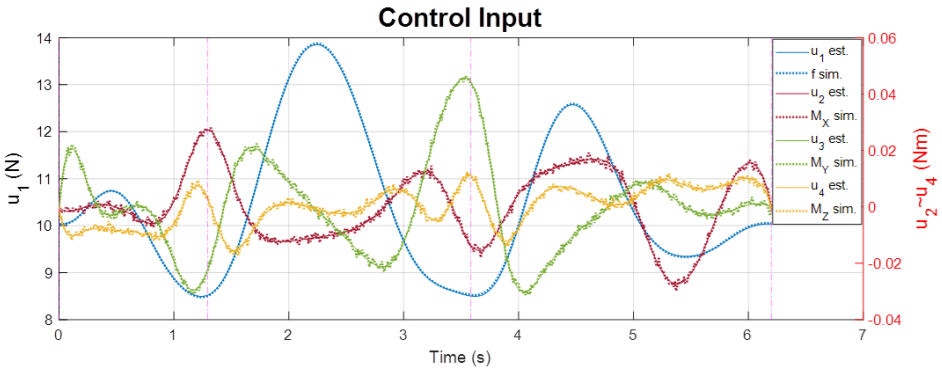

Figure 2.17. Control input comparison.

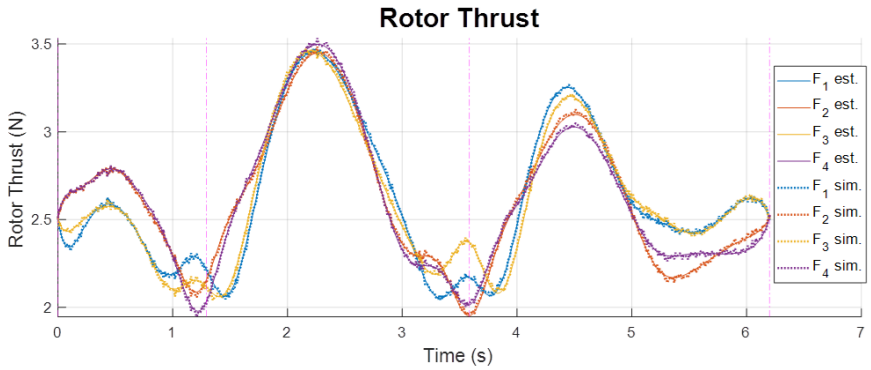

Figure 2.18. Rotor force comparison.

2.5 Yaw Trajectory Optimization

If we do not have a specific requirement on yaw angle, can we optimize the yaw trajectory to achieve better aggressive performance? In other words, by finding the yaw trajectory such that no M_z control moment input is required to track the aggressive polynomial trajectory, we can further reduce the maximum rotor force needed and thereby track

the aggressive trajectory with less motor force or achieve a faster trajectory with the same aggressiveness specified. From the equations of motion:

$$\begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} = \boldsymbol{I} \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} + \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \boldsymbol{I} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.62}$$

Because of the symmetry, the moment of inertia of the quadcopter is assumed as:

$$\boldsymbol{I} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}, \text{ and } I_{xx} = I_{yy} \tag{2.63}$$

$$M_z = I_{zz}\dot{r} + pq(I_{yy} - I_{xx}) = I_{zz}\dot{r} \tag{2.64}$$

Therefore, to make $M_z = 0$, we need $\dot{r} = 0$. Assuming the initial state $r_0 = 0$, this means $\dot{r} = r = 0$ for the whole trajectory. Based on this, we can modify the inverse dynamics analysis to solve for $\dot{\psi}$ and $\ddot{\psi}$ as:

$$\begin{aligned} r &= \left(\dot{\psi} + \frac{\sin\theta}{\cos\phi}p\right)\frac{\cos\phi}{\cos\theta} = 0 \\ \Rightarrow \dot{\psi} &= -\frac{\sin\theta}{\cos\phi}p \end{aligned} \tag{2.65}$$

$$\begin{aligned} \dot{r} = \frac{\cos\phi}{\cos\theta}\Bigg[\ddot{\psi} + \frac{\sin\theta}{\cos\phi}\dot{p} + p\left(\frac{\cos\theta\cos\phi\dot{\theta} + \sin\theta\sin\phi\dot{\phi}}{\cos^2\phi}\right) \\ -r\left(\frac{\cos\theta\sin\phi\dot{\phi} - \sin\theta\cos\phi\dot{\theta}}{\cos^2\phi}\right)\Bigg] = 0 \\ \Rightarrow \ddot{\psi} = -\frac{\sin\theta}{\cos\phi}\dot{p} - p\left(\frac{\cos\theta\cos\phi\dot{\theta} + \sin\theta\sin\phi\dot{\phi}}{\cos^2\phi}\right) \end{aligned} \tag{2.66}$$

Having $\dot{\psi}$ and $\ddot{\psi}$ at each moment in the trajectory, we can integrate ψ iteratively throughout the inverse dynamics analysis by $\psi_{n+1} = \psi_n + \dot{\psi}\Delta t + \frac{1}{2}\ddot{\psi}\Delta t^2$. Figures 2.19 and 2.20 show a simple four-waypoint aggressive trajectory and the corresponding optimal yaw trajectory obtained by this method.

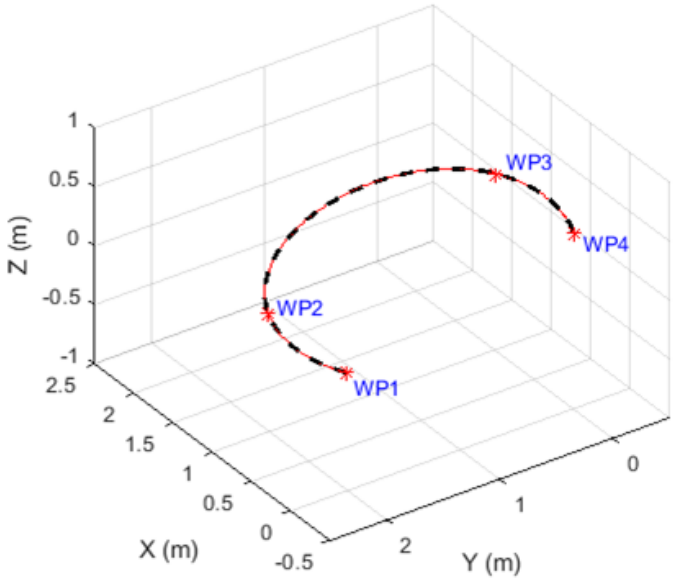

Figure 2.19. Simple 4-waypoint aggressive trajectory.

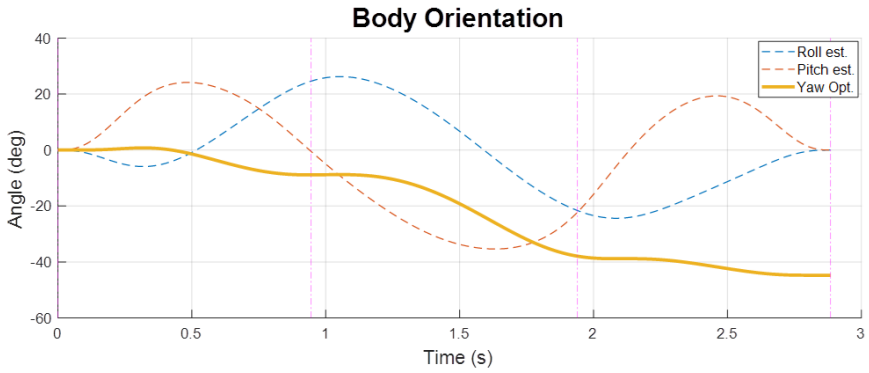

Figure 2.20. Optimal yaw trajectory for the trajectory.

We converted this optimal yaw trajectory into a polynomial form and used the geometric controller to track this aggressive polynomial trajectory. Figure 2.21 shows that, by tracking this optimal yaw trajectory, the M_z control input will be indeed close to zero.

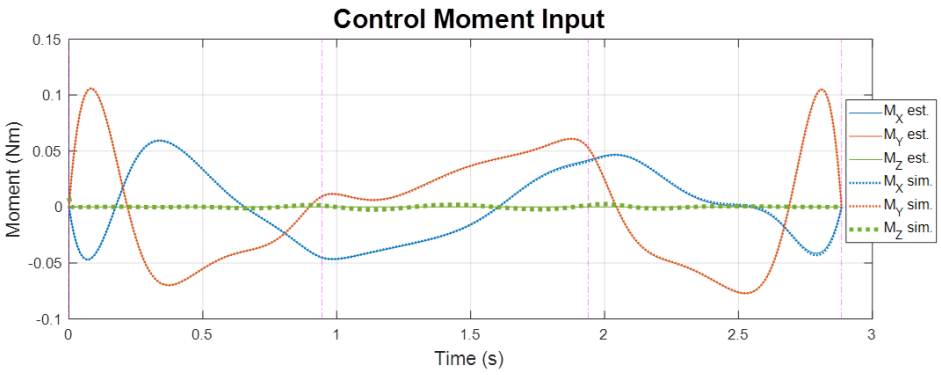

Figure 2.21. Simulation result of control input with geometric controller.

Figure 2.22 shows the comparison of rotor thrust used to track the aggressive trajectory between constant zero yaw and the optimal yaw trajectory. The additional yaw control effort M_z in the constant zero yaw case pulls the F_1 and F_3 rotor forces away from the F_2 and F_4 rotor forces and thereby increases the maximum rotor force needed. From the result, we can see that, by tracking the optimal yaw trajectory, the maximum rotor force is reduced significantly by 17.5% (from 4 N to 3.3 N).

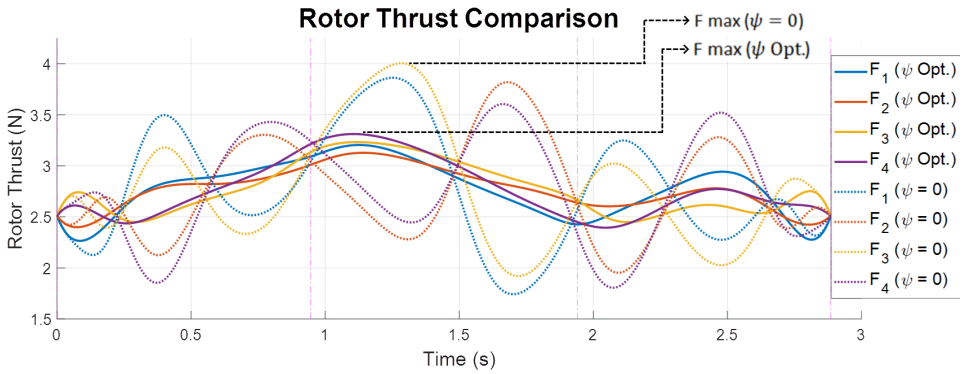

Figure 2.22. Rotor force comparison.

Remark:

With the optimization and analysis framework developed, we review the choice of min-

imum snap trajectory. For the same four-waypoint scenario and the same total time, we compare the optimal trajectory and the maximum rotor force required with the cost of different orders of trajectory derivatives to minimize. Figure 2.24 shows the geometry of the trajectories. As shown in Figure 2.23, the trajectory that minimizes the cost on the fourth derivative has the minimum force required. Therefore, we conclude that minimum snap is indeed the optimal choice for this approach of multi-copter aggressive trajectory optimization. This kind of analysis is rarely seen in literatures though minimum snap or jerk trajectories are so commonly used by researchers.

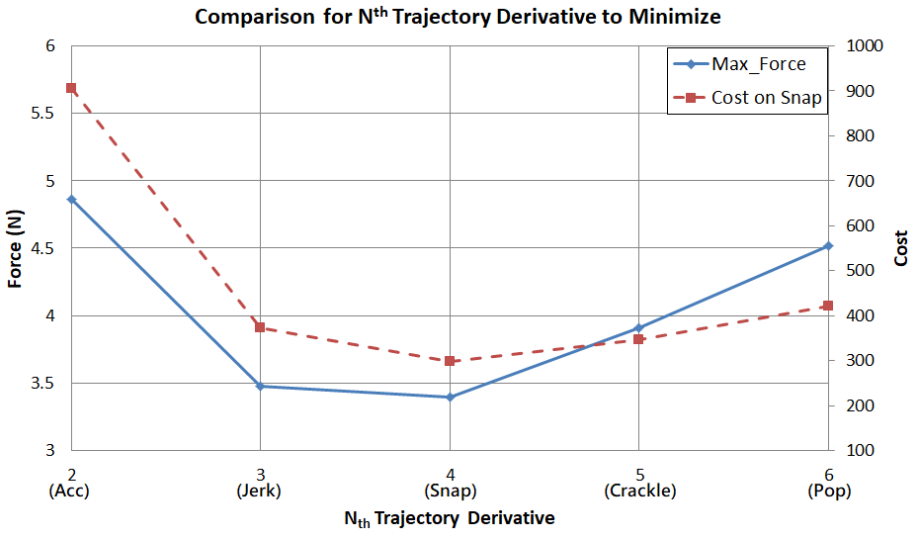

Figure 2.23. Maximum rotor force required for trajectories optimized with cost on different order of derivative.

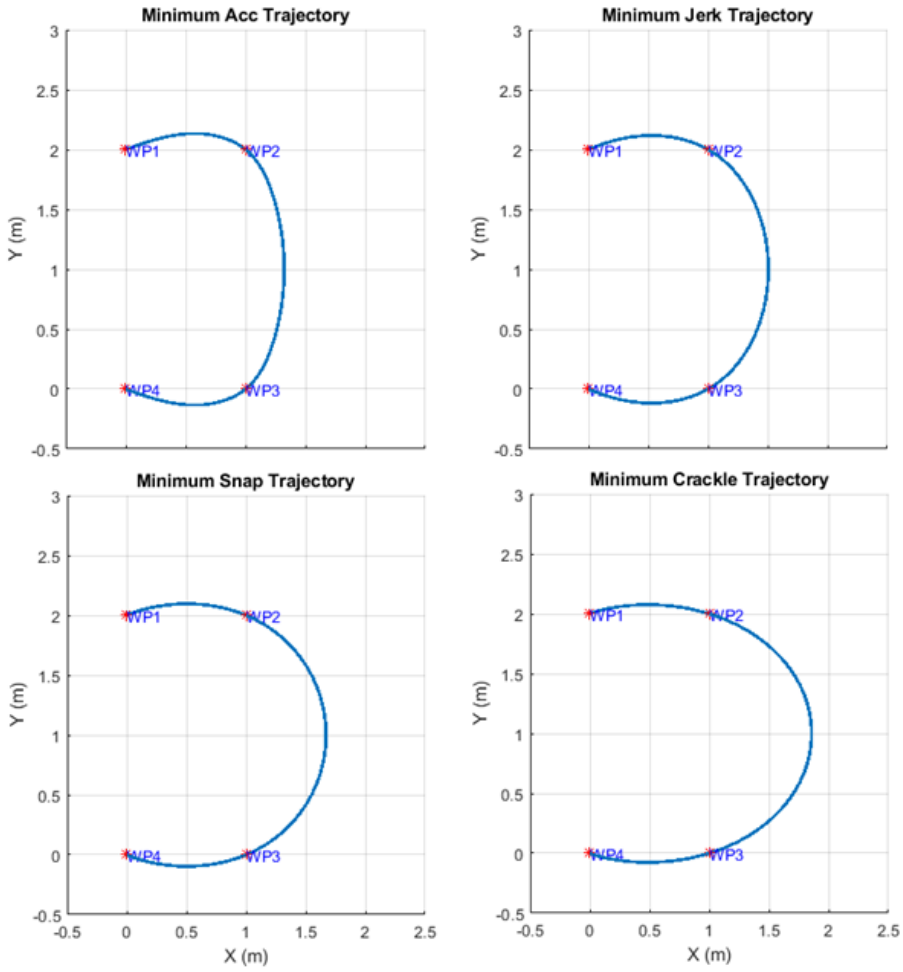

Figure 2.24. Comparison of trajectories optimized with cost on different order of derivative.

2.6 Conclusions

With the framework developed in this research, we can find the optimal polynomial trajectory and the corresponding segment time allocation for a given quadcopter model, a specified scenario, and a desired aggressiveness. We can also tell if the given scenario is beyond the capability of the given quadcopter by examining whether the solution exists. Furthermore, we can use the algorithm to evaluate the minimum required rotor thrust to

achieve the scenario. Instead of tracking the maximum force, we can also use the algorithm to track the total time. That is, given a desired flight time, we can find the optimal trajectory that has the minimum required rotor thrust. Though the quadcopter model is used in this paper, the inverse dynamics analysis for control input (collective force and three-axis moment) applies to generic multi-copters. Therefore, the method developed in this research can be applied to multi-copters with minor modifications to the vehicle control allocation model. The geometric controller used in this research showed the capability of tracking the optimized aggressive trajectory. The tracking result verified the feasibility of the optimized trajectory and the credibility of the inverse dynamics analysis. In addition, the yaw trajectory optimization method is capable of improving the aggressive performance in the case of no requirement on heading angle.

Chapter 3

Compensation for Aerodynamic, Gyroscopic and Rotor Rolling Effects in the Synthesis

In Chapter 2, a complete framework had been developed to find the optimal aggressive trajectory that fully utilizes the thrust capability of the rotor for given waypoint coordinates, corresponding heading angles, and vehicle velocity and attitude constraints at the waypoints. For clarity and simplicity, we adopted a commonly used simplified quadcopter dynamic model as in [39]. However, in aggressive flight some ignored effects could become significant, and it could be unrealistic to find the optimal trajectory based on the simplified dynamic model. Therefore, in this chapter the problem is addressed as: how to incorporate aerodynamic, gyroscopic and rotor rolling effects into the multi-copter aggressive constrained trajectory synthesis and the tracking control for more realistic trajectory planning and more accurate trajectory tracking.

3.1 Mathematical Model Description

We start from the general model for quadcopters used in Chapter 2. The coordinate systems and forces and moments generated by the rotors are shown in figure 3.1.

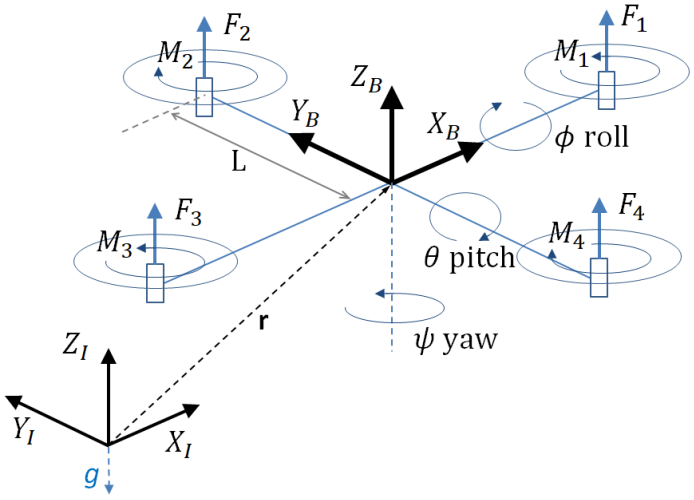

Figure 3.1. Reference frames and quadcopter forces/moments.

The body frame, B, is attached to the center of mass of the quadcopter and rotor 1 is on the positive X_B-axis. The gravitational acceleration g is in the $-Z_I$ direction of the inertial frame, I. Euler angles roll ϕ, pitch θ and yaw ψ are used to define orientation from inertial frame to body frame. Note, $Z-X-Y$ rotation order is used here and therefore the rotation matrix for transforming coordinates from B to I is given by:

$$\boldsymbol{R}_{IB} = \begin{bmatrix} \cos\psi\cos\theta - \sin\phi\sin\psi\sin\theta & -\cos\phi\sin\psi & \cos\psi\sin\theta + \cos\theta\sin\phi\sin\psi \\ \cos\theta\sin\psi + \cos\psi\sin\phi\sin\theta & \cos\phi\cos\psi & \sin\psi\sin\theta - \cos\psi\cos\theta\sin\phi \\ -\cos\phi\sin\theta & \sin\phi & \cos\phi\cos\theta \end{bmatrix} \tag{3.1}$$

With p, q and r denoting the components of angular velocity of the quadcopter in the body frame, the rotational kinematics equation is given by:

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & -\cos\phi\sin\theta \\ 0 & 1 & \sin\phi \\ \sin\theta & 0 & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{3.2}$$

Assuming rotor 1 and 3 rotate in the $-Z_B$ direction while 2 and 4 rotate in the Z_B direction, M_1 and M_3 act in the Z_B direction while M_2 and M_4 act in the $-Z_B$ direction since the moment produced by the rotor is opposite to the direction of rotation of the blade. Define the input $u = [u_1 \quad u_2 \quad u_3 \quad u_4]^T$ wherein u_1 is the total force from the rotors and u_2, u_3 and u_4 are the moments about X_B, Y_B and Z_B axes.

From here, we modify the model to add additional effects. Following [26], we assume that the counter-moment M_i of the i^{th} rotor is proportional to the thrust F_i by constant k_M:

$$M_i = k_M F_i \tag{3.3}$$

The relationship between the input and the rotor thrust can be represented as:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} F_1 + F_2 + F_3 + F_4 \\ L(F_2 - F_4) \\ L(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & L & 0 & -L \\ -L & 0 & L & 0 \\ k_M & -k_M & k_M & -k_M \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} \tag{3.4}$$

For the aerodynamic effects, we adopted the well-defined models used in [26]. It is found reasonable and practical because the authors showed in [26] how the models can be verified and the coefficients can be accurately obtained through experiments. Assuming the minor parasitic drag negligible in the flight regime, the aerodynamic effects are modelled into three parts, a drag force $\boldsymbol{D}$ on the propeller plane, a torque $\boldsymbol{\tau}_D$ induced by the drag force, and a refined thrust model to accommodate the vertical effect along the body z-axis.

$$\boldsymbol{D} = -k_d \omega_s \boldsymbol{v}_h = -k_d \omega_s \boldsymbol{R}_{IB} P \boldsymbol{R}_{IB}^T \boldsymbol{v} \tag{3.5}$$

$$\boldsymbol{\tau}_D = -(k_d \omega_s \boldsymbol{R}_{IB} P \boldsymbol{R}_{IB}^T \boldsymbol{v}) \times h \boldsymbol{e}_3 \tag{3.6}$$

$$F_i = k_\omega \omega_i^2 - k_z v_z \omega_i + k_h v_h^2 \tag{3.7}$$

Where k_d is the drag constant, $\omega_s = \sum_{i=1}^{4} \omega_i$ is the sum of the rotor speeds ω_i, P is the projection matrix $P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$, $\boldsymbol{e}_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$ is the third standard basis vector, h is the height of the propeller plane above the center of mass, and k_ω, k_z and k_h are thrust constants for rotor speed, vertical and horizontal vehicle velocity. Figure 3.2 shows the drag force the induced torque acting on a quadcopter.

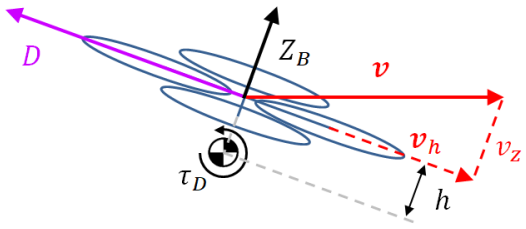

Figure 3.2. Drag and drag induced torque acting on a quadcopter.

Because the rotor is spinning fast about the z-axis, the existence of vehicle pitching or rolling rate will cause the gyro effect. This gyroscopic torque for an individual rotor is modeled as:

$$\boldsymbol{\tau}_{Gi} = \boldsymbol{I}_r \begin{bmatrix} 0 \\ 0 \\ \epsilon_i \omega_i \end{bmatrix} \times \boldsymbol{\omega}_c \tag{3.8}$$

Where $\boldsymbol{I}_r$ is the moment of inertia of the rotor, ω_i is the rotary speed of the i^{th} rotor, $\boldsymbol{\omega}_c$ is the angular rate of the quadcopter, and ϵ_i denotes the turning direction of the

rotor, namely +1 (clockwise) or -1 (counter clockwise). The total gyroscopic torque for the quadcopter is accumulated as:

$$\boldsymbol{\tau}_G = \sum_{i=1}^{4} \boldsymbol{I}_r \begin{bmatrix} 0 \\ 0 \\ \epsilon_i \omega_i \end{bmatrix} \times \boldsymbol{\omega}_c = \boldsymbol{I}_r \begin{bmatrix} 0 \\ 0 \\ \omega_p \end{bmatrix} \times \boldsymbol{\omega}_c \tag{3.9}$$

Where the difference of the motor speeds $\omega_p = (\omega_1 + \omega_3 - \omega_2 - \omega_4)$. For a rotor during forward flight, the advancing blade encounters higher wind speed and generates more lift than the retreating blade does, and thus a rolling torque is generated as shown in figure 3.3. Following [31] with additional ϵ_i for rotor turning direction, this rotor rolling torque for an individual rotor is modelled as:

$$\boldsymbol{\tau}_{Ri} = \epsilon_i \omega_i C_R \boldsymbol{v}_h \tag{3.10}$$

Where C_R is the rolling torque constant. The total rotor rolling torque for the quadcopter is accumulated as:

$$\boldsymbol{\tau_R} = \sum_{i=1}^{4} \epsilon_i \omega_i C_R \boldsymbol{v}_h = \omega_p C_R \boldsymbol{v}_h \tag{3.11}$$

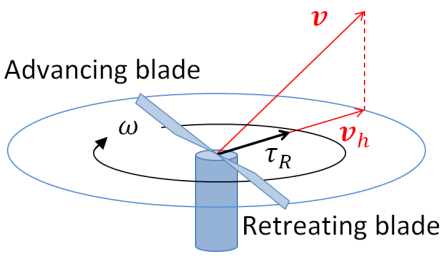

Figure 3.3. Rotor rolling torque acting on a forward moving rotor.

3.2 Solution Methodology

We integrate the compensations for aerodynamic, gyroscopic and rotor rolling effects into the optimal aggressive constrained trajectory synthesis proposed in Chapter 2. Figure 3.4 shows the whole methodology with the modified parts highlighted in red. First we integrate all those effects into the inverse dynamics analysis and solve for the vehicle states, control inputs and rotor speeds in an iterative fashion. Then, we integrate the aerodynamic drag into the trajectory derivative constraints for window passing to compensate the attitude change due to the drag force. This is also an iterative process and the modified inverse dynamics analysis is utilized to solve ω_s at each window waypoint iteratively.

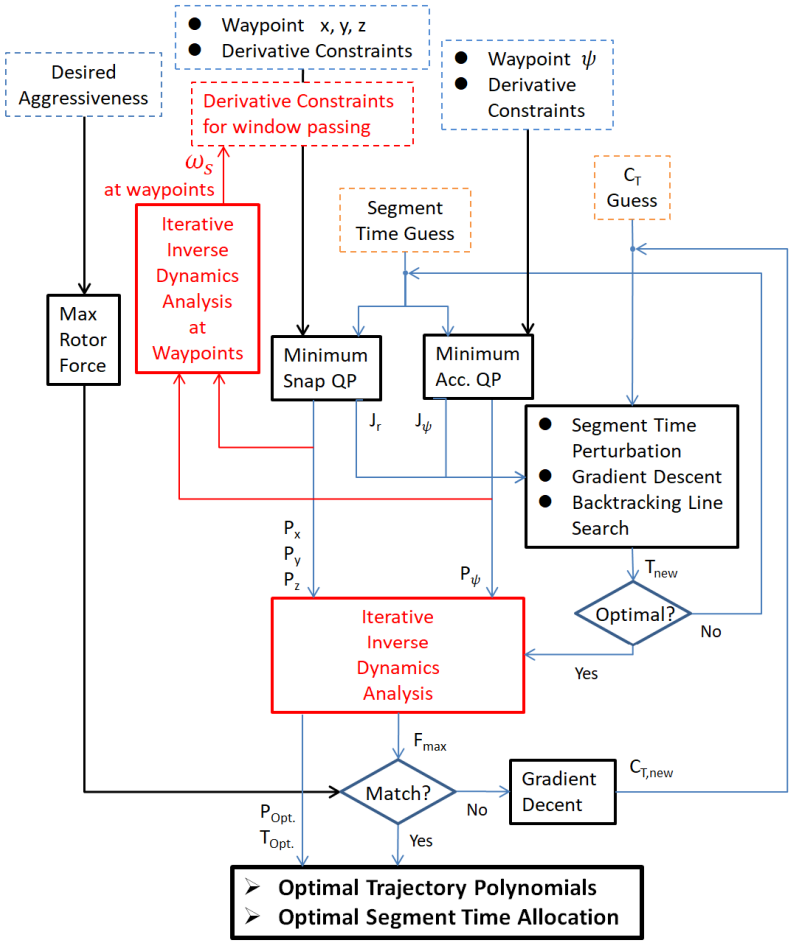

Figure 3.4. Solution methodology and procedure of the aggressive trajectory optimization.

3.2.1 Inverse Dynamics Analysis

The differential flatness property [39] is an essential part in our constrained trajectory optimization to connect the polynomial trajectory to the actual vehicle dynamics and control inputs. In this section we show how the differential flatness can be preserved in the inverse dynamics analysis while introducing all the additional effects. One dilemma we have here is that we need ω_s and ω_p in the additional effect models to start the process but the actual rotor speeds can only be obtained in the end of the analysis when all the additional effects are compensated and the individual rotor thrusts are determined. To address

this problem, we propose an iterative method to solve ω_s and ω_p as well as the vehicle state and control inputs for a given moment t through the trajectory position and its derivatives. Firstly, we make the initial guess on ω_s and ω_p as in steady hovering condition as:

$$\omega_s = 4\sqrt{\frac{mg}{4k_F}}\ ,\ \omega_p = 0 \tag{3.12}$$

3.2.1.1 Drag Compensation in Attitude and Collective Force

Let $\boldsymbol{X} = [x \quad y \quad z]^T$ denote the trajectory position at a given moment t, we have the following relationship among desired collective rotor force $\boldsymbol{F}_d$, aerodynamic drag $\boldsymbol{D}$, trajectory velocity $\dot{\boldsymbol{X}}$ and acceleration $\ddot{\boldsymbol{X}}$, and gravity force:

$$\boldsymbol{F}_d + \boldsymbol{D} = u_1 \boldsymbol{R}_{IB} \boldsymbol{e}_3 - k_d \omega_s \boldsymbol{R}_{IB} P \boldsymbol{R}_{IB}^T \dot{\boldsymbol{X}} = m\ddot{\boldsymbol{X}} + mg\boldsymbol{e}_3 \tag{3.13}$$

To solve for vehicle attitude (rotation matrix $\boldsymbol{R}_{IB}$) and collective rotor force input u_1, we project the equation onto body frame by left-multiply $\boldsymbol{R}_{IB}^T$:

$$\begin{bmatrix} 0 \\ 0 \\ u_1 \end{bmatrix} - k_d \omega_s P \boldsymbol{R}_{IB}^T \dot{\boldsymbol{X}} = \boldsymbol{R}_{IB}^T (m\ddot{\boldsymbol{X}} + mg\boldsymbol{e}_3) \tag{3.14}$$

With $\boldsymbol{R}_{IB}^T = [\boldsymbol{x}_B \quad \boldsymbol{y}_B \quad \boldsymbol{z}_B]^T$, we have:

$$\begin{bmatrix} 0 \\ 0 \\ u_1 \end{bmatrix} - k_d \omega_s \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_B^T \\ \boldsymbol{y}_B^T \\ \boldsymbol{z}_B^T \end{bmatrix} \dot{\boldsymbol{X}} = \begin{bmatrix} \boldsymbol{x}_B^T \\ \boldsymbol{y}_B^T \\ \boldsymbol{z}_B^T \end{bmatrix} (m\ddot{\boldsymbol{X}} + mg\boldsymbol{e}_3) \tag{3.15}$$

$$\begin{cases} -k_d\omega_s \boldsymbol{x}_B^T \dot{\boldsymbol{X}} = \boldsymbol{x}_B^T m(\ddot{\boldsymbol{X}} + g\boldsymbol{e}_3) \Rightarrow \boldsymbol{x}_B^T[k_d\omega_s \dot{\boldsymbol{X}} + m(\ddot{\boldsymbol{X}} + g\boldsymbol{e}_3)] = 0 \\ -k_d\omega_s \boldsymbol{y}_B^T \dot{\boldsymbol{X}} = \boldsymbol{y}_B^T m(\ddot{\boldsymbol{X}} + g\boldsymbol{e}_3) \Rightarrow \boldsymbol{y}_B^T[k_d\omega_s \dot{\boldsymbol{X}} + m(\ddot{\boldsymbol{X}} + g\boldsymbol{e}_3)] = 0 \\ u_1 = \boldsymbol{z}_B^T m(\ddot{\boldsymbol{X}} + g\boldsymbol{e}_3) \end{cases} \tag{3.16}$$

Since the rotor cannot generate negative thrust, F_d is always in the direction of $\boldsymbol{z}_B$ in our case, we have the solution:

$$\begin{cases} \boldsymbol{z}_B = \frac{k_d\omega_s \dot{\boldsymbol{X}} + m(\ddot{\boldsymbol{X}} + g\boldsymbol{e}_3)}{\|k_d\omega_s \dot{\boldsymbol{X}} + m(\ddot{\boldsymbol{X}} + g\boldsymbol{e}_3)\|} \\ u_1 = m(\ddot{\boldsymbol{X}} + g\boldsymbol{e}_3) \cdot \boldsymbol{z}_B \end{cases} \tag{3.17}$$

Figure 3.5 illustrates the physical meaning of this solution and how we can find the unique $\boldsymbol{z}_B$ from $k_d\omega_s \dot{\boldsymbol{X}}$ and $m(\ddot{\boldsymbol{X}} + g\boldsymbol{e}_3)$ vectors.

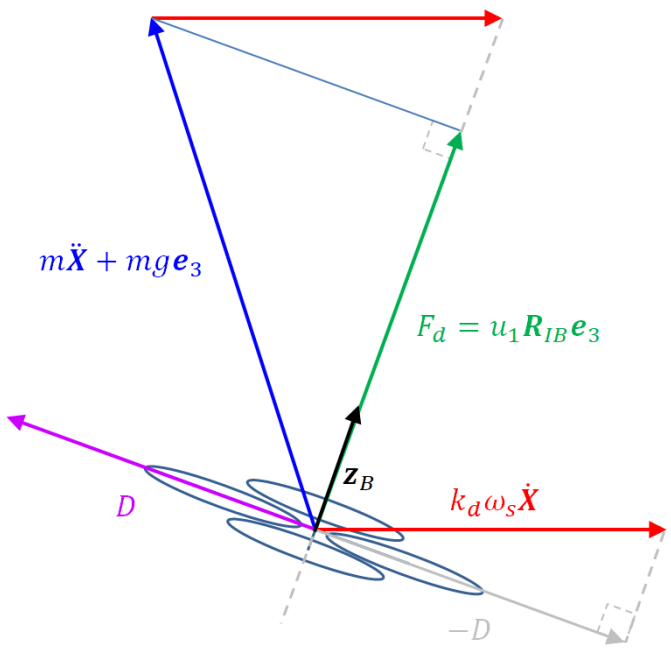

Figure 3.5. Schematic of drag force compensation in u_1 and $\boldsymbol{z}_B$ direction.

With $\boldsymbol{z}_B$ obtained, we can then solve $\boldsymbol{x}_B$ and $\boldsymbol{y}_B$ by assuming $\boldsymbol{x}_c$ as the vector obtained by rotating $\boldsymbol{x}_W$ around $\boldsymbol{z}_W$ by trajectory yaw angle ψ:

$$\boldsymbol{x}_C = \begin{bmatrix} \cos\psi & \sin\psi & 0 \end{bmatrix}^T \tag{3.18}$$

Assuming $\boldsymbol{z}_B$ not parallel to $\boldsymbol{x}_c$, we can determine $\boldsymbol{x}_B$ and $\boldsymbol{y}_B$ by:

$$\boldsymbol{y}_B = \frac{\boldsymbol{z}_B \times \boldsymbol{x}_C}{\|\boldsymbol{z}_B \times \boldsymbol{x}_C\|}\ ,\ \boldsymbol{x}_B = \boldsymbol{y}_B \times \boldsymbol{z}_B \tag{3.19}$$

Notice that in some extreme cases when $\theta = \pm\pi/2$ and $\phi = 0$ or $\pm\ \pi$, $\boldsymbol{z}_B$ could be parallel to $\boldsymbol{x}_c$ and a singularity would happen in this method. Some further fix will be needed if such extreme cases are expected in the flight. Finally, the rotation matrix $\boldsymbol{R}_{IB}$ is then obtained as:

$$\boldsymbol{R}_{IB} = \begin{bmatrix} \boldsymbol{x}_B & \boldsymbol{y}_B & \boldsymbol{z}_B \end{bmatrix} \tag{3.20}$$

3.2.1.2 Drag Compensation in Angular Velocity

To solve for vehicle body angular velocity p, q and r, we start from the equation of acceleration with drag force added as:

$$m\ddot{\boldsymbol{X}} = -mg\boldsymbol{e}_3 + u_1\boldsymbol{R}_{IB}\boldsymbol{e}_3 - k_d\omega_s\boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T\dot{\boldsymbol{X}} \tag{3.21}$$

Take the time derivative of the equation we have:

$$\begin{aligned} m\dddot{\boldsymbol{X}} =& \dot{u}_1\boldsymbol{R}_{IB}\boldsymbol{e}_3 + u_1\dot{\boldsymbol{R}}_{IB}\boldsymbol{e}_3 \\ & - k_d\omega_s[\dot{\boldsymbol{R}}_{IB}P\boldsymbol{R}_{IB}^T\dot{\boldsymbol{X}} + \boldsymbol{R}_{IB}P\dot{\boldsymbol{R}}_{IB}^T\dot{\boldsymbol{X}} + \boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T\ddot{\boldsymbol{X}}] \end{aligned} \tag{3.22}$$

Express the time derivative of the rotation matrix as $\dot{\boldsymbol{R}}_{IB} = \hat{\boldsymbol{\omega}}_B\boldsymbol{R}_{IB}$:

$$\begin{cases} \dot{\boldsymbol{R}}_{IB}\boldsymbol{e}_3 = \boldsymbol{\omega}_B \times \boldsymbol{R}_{IB}\boldsymbol{e}_3 = \boldsymbol{\omega}_B \times \boldsymbol{z}_B \\ \dot{\boldsymbol{R}}_{IB}P\boldsymbol{R}_{IB}^T\dot{\boldsymbol{X}} = \hat{\omega}_B\boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T\dot{\boldsymbol{X}} = \boldsymbol{\omega}_B \times \boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T\dot{\boldsymbol{X}} \\ \boldsymbol{R}_{IB}P\dot{\boldsymbol{R}}_{IB}^T\dot{\boldsymbol{X}} = \boldsymbol{R}_{IB}P(\hat{\boldsymbol{\omega}}_B\boldsymbol{R}_{IB})^T\dot{\boldsymbol{X}} \\ \qquad\qquad = \boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T(-\hat{\boldsymbol{\omega}}_B)\dot{\boldsymbol{X}} = -\boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T(\boldsymbol{\omega}_B \times \dot{\boldsymbol{X}}) \end{cases} \tag{3.23}$$

We can rewrite the above equation (3.22) as:

$$\begin{aligned} m\dddot{\boldsymbol{X}} =& \dot{u}_1\boldsymbol{z}_B + \boldsymbol{\omega}_B \times u_1\boldsymbol{z}_B - k_d\omega_s[\boldsymbol{\omega}_B \times \boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T\dot{\boldsymbol{X}} \\ & - \boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T(\boldsymbol{\omega}_B \times \dot{\boldsymbol{X}}) + \boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T\ddot{\boldsymbol{X}}] \end{aligned} \tag{3.24}$$

Neglecting terms perpendicular to $\boldsymbol{z}_B$, we have $\dot{u}_1 = \boldsymbol{z}_B \cdot m\dddot{\boldsymbol{X}} + k_d\omega_s[\boldsymbol{z}_B \cdot (\boldsymbol{\omega}_B \times \boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T\dot{\boldsymbol{X}})]$. Substituting $\dot{u}_1$ back we can rewrite the equation as:

$$\begin{aligned} m\dddot{\boldsymbol{X}} =& (\boldsymbol{z}_B \cdot m\dddot{\boldsymbol{X}})\boldsymbol{z}_B + \boldsymbol{\omega}_B \times u_1\boldsymbol{z}_B - k_d\omega_s\{\boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T\ddot{\boldsymbol{X}} \\ & + \boldsymbol{\omega}_B \times \boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T\dot{\boldsymbol{X}} - [\boldsymbol{z}_B \cdot (\boldsymbol{\omega}_B \times \boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T\dot{\boldsymbol{X}})]\boldsymbol{z}_B \\ & - \boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T(\boldsymbol{\omega}_B \times \dot{\boldsymbol{X}})\} \end{aligned} \tag{3.25}$$

Now we further express the rotation matrix $\boldsymbol{R}_{IB}$ and angular rate $\boldsymbol{\omega}_B$ terms with body axes unit vectors $[\boldsymbol{x}_B \quad \boldsymbol{y}_B \quad \boldsymbol{z}_B]$ and body angular velocity p, q and r:

$$
\begin{cases}
\boldsymbol{R}_{IB} P \boldsymbol{R}_{IB}^T \dot{\boldsymbol{X}} = (\dot{\boldsymbol{X}} \cdot \boldsymbol{x}_B)\boldsymbol{x}_B + (\dot{\boldsymbol{X}} \cdot \boldsymbol{y}_B)\boldsymbol{y}_B \\
\boldsymbol{\omega}_B \times \boldsymbol{x}_B = (\boldsymbol{R}_{IB}{}^B \boldsymbol{\omega}_B) \times \boldsymbol{x}_B \\
\qquad\qquad = \left(\begin{bmatrix} \boldsymbol{x}_B & \boldsymbol{y}_B & \boldsymbol{z}_B \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \right) \times \boldsymbol{x}_B = r\boldsymbol{y}_B - q\boldsymbol{z}_B \\
\boldsymbol{\omega}_B \times \boldsymbol{y}_B = -r\boldsymbol{x}_B + p\boldsymbol{z}_B \\
\boldsymbol{\omega}_B \times \boldsymbol{z}_B = q\boldsymbol{x}_B - p\boldsymbol{y}_B
\end{cases} \tag{3.26}
$$

We can rewrite $\boldsymbol{\omega}_B \times \boldsymbol{R}_{IB} P \boldsymbol{R}_{IB}^T \dot{\boldsymbol{X}} - [\boldsymbol{z}_B \cdot (\boldsymbol{\omega}_B \times \boldsymbol{R}_{IB} P \boldsymbol{R}_{IB}^T \dot{\boldsymbol{X}})]\boldsymbol{z}_B$ as:

$$
\begin{aligned}
&(\dot{\boldsymbol{X}} \cdot \boldsymbol{x}_B)[\boldsymbol{\omega}_B \times \boldsymbol{x}_B]_{\boldsymbol{x}_B \boldsymbol{y}_B} + (\dot{\boldsymbol{X}} \cdot \boldsymbol{y}_B)[\boldsymbol{\omega}_B \times \boldsymbol{y}_B]_{\boldsymbol{x}_B \boldsymbol{y}_B} \\
&\qquad = (\dot{\boldsymbol{X}} \cdot \boldsymbol{x}_B) r \boldsymbol{y}_B - (\dot{\boldsymbol{X}} \cdot \boldsymbol{y}_B) r \boldsymbol{x}_B
\end{aligned} \tag{3.27}
$$

And rewrite $\boldsymbol{R}_{IB} P \boldsymbol{R}_{IB}^T (\boldsymbol{\omega}_B \times \dot{\boldsymbol{X}})$ as:

$$
\begin{aligned}
&[\boldsymbol{x}_B \cdot (\boldsymbol{\omega}_B \times \dot{\boldsymbol{X}})]\boldsymbol{x}_B + [\boldsymbol{y}_B \cdot (\boldsymbol{\omega}_B \times \dot{\boldsymbol{X}})]\boldsymbol{y}_B \\
&= \{\dot{\boldsymbol{X}} \cdot [\boldsymbol{x}_B \times (p\boldsymbol{x}_B + q\boldsymbol{y}_B + r\boldsymbol{z}_B)]\}\boldsymbol{x}_B + \{\dot{\boldsymbol{X}} \cdot [\boldsymbol{y}_B \times (p\boldsymbol{x}_B + q\boldsymbol{y}_B + r\boldsymbol{z}_B)]\}\boldsymbol{y}_B \\
&= [q(\dot{\boldsymbol{X}} \cdot \boldsymbol{z}_B) - r(\dot{\boldsymbol{X}} \cdot \boldsymbol{y}_B)]\boldsymbol{x}_B + [-p(\dot{\boldsymbol{X}} \cdot \boldsymbol{z}_B) + r(\dot{\boldsymbol{X}} \cdot \boldsymbol{x}_B)]\boldsymbol{y}_B
\end{aligned} \tag{3.28}
$$

Now the previous equation (3.25) can be rewritten and simplified as:

$$
\begin{aligned}
m\ddot{\dot{\boldsymbol{X}}} =& (\boldsymbol{z}_B \cdot m\ddot{\dot{\boldsymbol{X}}})\boldsymbol{z}_B + u_1(q\boldsymbol{x}_B - p\boldsymbol{y}_B) \\
&- k_d \omega_s [\boldsymbol{R}_{IB} P \boldsymbol{R}_{IB}^T \ddot{\dot{\boldsymbol{X}}} - q(\dot{\boldsymbol{X}} \cdot \boldsymbol{z}_B)\boldsymbol{x}_B + p(\dot{\boldsymbol{X}} \cdot \boldsymbol{z}_B)\boldsymbol{y}_B]
\end{aligned} \tag{3.29}
$$

Collect known terms and let $A = m\ddot{\dot{\boldsymbol{X}}} - (\boldsymbol{z}_B \cdot m\ddot{\dot{\boldsymbol{X}}})\boldsymbol{z}_B + k_d \omega_s \boldsymbol{R}_{IB} P \boldsymbol{R}_{IB}^T \ddot{\dot{\boldsymbol{X}}}$, we have:

$$u_1(q\boldsymbol{x}_B - p\boldsymbol{y}_B) + k_d\omega_s q(\dot{\boldsymbol{X}} \cdot \boldsymbol{z}_B)\boldsymbol{x}_B - k_d\omega_s p(\dot{\boldsymbol{X}} \cdot \boldsymbol{z}_B)\boldsymbol{y}_B = A \tag{3.30}$$

Project this equation on $\boldsymbol{x}_B$ and $\boldsymbol{y}_B$, we have the equations to solve p and q:

$$\begin{cases} u_1 q + k_d\omega_s q(\dot{\boldsymbol{X}} \cdot \boldsymbol{z}_B) = A \cdot \boldsymbol{x}_B \\ -u_1 p - k_d\omega_s p(\dot{\boldsymbol{X}} \cdot \boldsymbol{z}_B) = A \cdot \boldsymbol{y}_B \end{cases} \tag{3.31}$$

The solution for p and q is:

$$\begin{cases} q = \frac{A \cdot \boldsymbol{x}_B}{u_1 + k_d\omega_s(\dot{\boldsymbol{X}} \cdot \boldsymbol{z}_B)} \\ p = \frac{-A \cdot \boldsymbol{y}_B}{u_1 + k_d\omega_s(\dot{\boldsymbol{X}} \cdot \boldsymbol{z}_B)} \end{cases} \tag{3.32}$$

Finally we can solve r through the rotational kinematics equation:

$$\begin{aligned} \dot{\psi} &= -\frac{\sin\theta}{\cos\phi}p + \frac{\cos\theta}{\cos\phi}r \\ \Rightarrow r &= \left(\dot{\psi} + \frac{\sin\theta}{\cos\phi}p\right)\frac{\cos\phi}{\cos\theta} \end{aligned} \tag{3.33}$$

Where Euler angles $\phi = \arcsin\left[\boldsymbol{R}_{IB}(3,2)\right]$, $\theta = \arctan\left[\frac{\boldsymbol{R}_{IB}(3,1)}{-cos\phi}\right]$, and $\dot{\psi}$ is trajectory yaw rate.

3.2.1.3 Drag Compensation in Angular Acceleration

With p, q and r solved, we have $\boldsymbol{\omega}_B = \boldsymbol{R}_{IB}[p \quad q \quad r)]^T$. $\dot{\phi}$ and $\dot{\theta}$ can be obtained from the inversion of the rotational kinematics equation (3.2). We also have $\dot{u}_1 = \boldsymbol{z}_B \cdot m\ddot{\boldsymbol{X}} + k_d\omega_s[\boldsymbol{z}_B \cdot (\boldsymbol{\omega}_B \times \boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T\dot{\boldsymbol{X}})]$. Now take the 2nd time derivative of equation (3.21):

$$
\begin{aligned}
m\dddot{\boldsymbol{X}} =&(\ddot{u}_1\boldsymbol{z}_B + \boldsymbol{\omega}_B \times \dot{u}_1\boldsymbol{z}_B + \boldsymbol{\omega}_B \times \boldsymbol{\omega}_B \times u_1\boldsymbol{z}_B + \dot{\boldsymbol{\omega}}_B \times u_1\boldsymbol{z}_B + \boldsymbol{\omega}_B \times \dot{u}_1\boldsymbol{z}_B) - k_d\omega_s\{\boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T\ddot{\boldsymbol{X}} \\
&+ \boldsymbol{\omega}_B \times \boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T\ddot{\boldsymbol{X}} - \boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T(\boldsymbol{\omega}_B \times \ddot{\boldsymbol{X}}) + \dot{\boldsymbol{\omega}}_B \times \boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T\dot{\boldsymbol{X}} + \boldsymbol{\omega}_B \times [\boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T\ddot{\boldsymbol{X}} \\
&+ \boldsymbol{\omega}_B \times \boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T\dot{\boldsymbol{X}} - \boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T(\boldsymbol{\omega}_B \times \dot{\boldsymbol{X}})] - [\boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T(\dot{\boldsymbol{\omega}}_B \times \dot{\boldsymbol{X}}) + \boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T(\boldsymbol{\omega}_B \times \ddot{\boldsymbol{X}}) \\
&+ \boldsymbol{\omega}_B \times \boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T(\boldsymbol{\omega}_B \times \dot{\boldsymbol{X}}) - \boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T(\boldsymbol{\omega}_B \times \boldsymbol{\omega}_B \times \dot{\boldsymbol{X}})]\} \\
=&(\ddot{u}_1\boldsymbol{z}_B + 2\boldsymbol{\omega}_B \times \dot{u}_1\boldsymbol{z}_B + \boldsymbol{\omega}_B \times \boldsymbol{\omega}_B \times u_1\boldsymbol{z}_B + \dot{\boldsymbol{\omega}}_B \times u_1\boldsymbol{z}_B) - k_d\omega_s\{\boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T\ddot{\boldsymbol{X}} \\
&- 2\boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T(\boldsymbol{\omega}_B \times \ddot{\boldsymbol{X}}) + \dot{\boldsymbol{\omega}}_B \times \boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T\dot{\boldsymbol{X}} - \boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T(\dot{\boldsymbol{\omega}}_B \times \dot{\boldsymbol{X}}) + \boldsymbol{\omega}_B \times [2\boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T\ddot{\boldsymbol{X}} \\
&+ \boldsymbol{\omega}_B \times \boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T\dot{\boldsymbol{X}} - 2\boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T(\boldsymbol{\omega}_B \times \dot{\boldsymbol{X}})] + \boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T(\boldsymbol{\omega}_B \times \boldsymbol{\omega}_B \times \dot{\boldsymbol{X}})\}
\end{aligned} \tag{3.34}
$$

Collect known terms perpendicular to $\boldsymbol{z}_B$:

$$
\begin{aligned}
T_1 =&2\boldsymbol{\omega}_B \times \dot{u}_1\boldsymbol{z}_B - k_d\omega_s[\boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T\dddot{\boldsymbol{X}} \\
&- 2\boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T(\boldsymbol{\omega}_B \times \ddot{\boldsymbol{X}}) + \boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T(\boldsymbol{\omega}_B \times \boldsymbol{\omega}_B \times \dot{\boldsymbol{X}})]
\end{aligned} \tag{3.35}
$$

Collect known terms with $\boldsymbol{z}_B$ component:

$$
\begin{aligned}
T_2 =&\boldsymbol{\omega}_B \times \boldsymbol{\omega}_B \times u_1\boldsymbol{z}_B - k_d\omega_s\boldsymbol{\omega}_B \times [2\boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T\ddot{\boldsymbol{X}} \\
&+ \boldsymbol{\omega}_B \times \boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T\dot{\boldsymbol{X}} - 2\boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T(\boldsymbol{\omega}_B \times \dot{\boldsymbol{X}})]
\end{aligned} \tag{3.36}
$$

Neglecting terms perpendicular to $\boldsymbol{z}_B$, we have:

$$
\ddot{u}_1 = \boldsymbol{z}_B \cdot [m\dddot{\boldsymbol{X}} - T_2 + k_d\omega_s\dot{\boldsymbol{\omega}}_B \times \boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T\dot{\boldsymbol{X}}] \tag{3.37}
$$

Substituting $\ddot{u}_1$ back we have:

$$
\begin{aligned}
m\dddot{\boldsymbol{X}} =&(\boldsymbol{z}_B \cdot m\dddot{\boldsymbol{X}})\boldsymbol{z}_B + T_1 + T_2 - (\boldsymbol{z}_B \cdot T_2)\boldsymbol{z}_B + \dot{\boldsymbol{\omega}}_B \times u_1\boldsymbol{z}_B \\
&- k_d\omega_s\{\dot{\boldsymbol{\omega}}_B \times \boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T\dot{\boldsymbol{X}} - [\boldsymbol{z}_B \cdot (\dot{\boldsymbol{\omega}}_B \times \boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T\dot{\boldsymbol{X}})]\boldsymbol{z}_B \\
&- \boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T(\dot{\boldsymbol{\omega}}_B \times \dot{\boldsymbol{X}})\}
\end{aligned} \tag{3.38}
$$

Now collect known terms again and define B as:

$$
\begin{aligned}
B =& m\dddot{\boldsymbol{X}} - (\boldsymbol{z}_B \cdot m\dddot{\boldsymbol{X}})\boldsymbol{z}_B - T_1 - T_2 + (\boldsymbol{z}_B \cdot T_2)\boldsymbol{z}_B \\
=& \dot{\boldsymbol{\omega}}_B \times u_1 \boldsymbol{z}_B - k_d \omega_s \{\dot{\boldsymbol{\omega}}_B \times \boldsymbol{R}_{IB} P \boldsymbol{R}_{IB}^T \dot{\boldsymbol{X}} \\
& - [\boldsymbol{z}_B \cdot (\dot{\boldsymbol{\omega}}_B \times \boldsymbol{R}_{IB} P \boldsymbol{R}_{IB}^T \dot{\boldsymbol{X}})]\boldsymbol{z}_B - \boldsymbol{R}_{IB} P \boldsymbol{R}_{IB}^T (\dot{\boldsymbol{\omega}}_B \times \dot{\boldsymbol{X}})\}
\end{aligned} \tag{3.39}
$$

Similar to the process in previous section, we express some rotation matrix $\boldsymbol{R}_{IB}$ and angular acceleration $\dot{\boldsymbol{\omega}}_B$ terms with body axis unit vectors $[\boldsymbol{x}_B \quad \boldsymbol{y}_B \quad \boldsymbol{z}_B]$ and body angular acceleration $\dot{p}$, $\dot{q}$ and $\dot{r}$:

$$
\begin{cases}
\dot{\boldsymbol{\omega}}_B \times \boldsymbol{x}_B = (\boldsymbol{R}_{IB}{}^B \dot{\boldsymbol{\omega}}_B) \times \boldsymbol{x}_B \\
\qquad\qquad = \left(\begin{bmatrix} \boldsymbol{x}_B & \boldsymbol{y}_B & \boldsymbol{z}_B \end{bmatrix} \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} \right) \times \boldsymbol{x}_B = \dot{r}\boldsymbol{y}_B - \dot{q}\boldsymbol{z}_B \\
\dot{\boldsymbol{\omega}}_B \times \boldsymbol{y}_B = -\dot{r}\boldsymbol{x}_B + \dot{p}\boldsymbol{z}_B \\
\dot{\boldsymbol{\omega}}_B \times \boldsymbol{z}_B = \dot{q}\boldsymbol{x}_B - \dot{p}\boldsymbol{y}_B
\end{cases} \tag{3.40}
$$

Now we have:

$$
B = u_1(\dot{q}\boldsymbol{x}_B - \dot{p}\boldsymbol{y}_B) + k_d\omega_s\dot{q}(\dot{\boldsymbol{X}} \cdot \boldsymbol{z}_B)\boldsymbol{x}_B - k_d\omega_s\dot{p}(\dot{\boldsymbol{X}} \cdot \boldsymbol{z}_B)\boldsymbol{y}_B \tag{3.41}
$$

Project this equation on $\boldsymbol{x}_B$ and $\boldsymbol{y}_B$, we have the equations to solve $\dot{p}$ and $\dot{q}$:

$$
\begin{cases}
u_1\dot{q} + k_d\omega_s\dot{q}(\dot{\boldsymbol{X}} \cdot \boldsymbol{z}_B) = B \cdot \boldsymbol{x}_B \\
-u_1\dot{p} - k_d\omega_s\dot{p}(\dot{\boldsymbol{X}} \cdot \boldsymbol{z}_B) = B \cdot \boldsymbol{y}_B
\end{cases} \tag{3.42}
$$

And we have the solution as:

$$
\begin{cases}
\dot{q} = \frac{B \cdot \boldsymbol{x}_B}{u_1 + k_d\omega_s(\dot{\boldsymbol{X}} \cdot \boldsymbol{z}_B)} \\
\dot{p} = \frac{-B \cdot \boldsymbol{y}_B}{u_1 + k_d\omega_s(\dot{\boldsymbol{X}} \cdot \boldsymbol{z}_B)}
\end{cases} \tag{3.43}
$$

Finally we solve $\dot{r}$ by taking the time derivative of equation (3.33):

$$\begin{aligned}\ddot{\psi} =& -\frac{\sin\theta}{\cos\phi}\dot{p} + \frac{\cos\theta}{\cos\phi}\dot{r} - p\frac{d}{dt}\left(\frac{\sin\theta}{\cos\phi}\right) + r\frac{d}{dt}\left(\frac{\cos\theta}{\cos\phi}\right) \\ \Rightarrow \dot{r} =& \frac{\cos\phi}{\cos\theta}\left[\ddot{\psi} + \frac{\sin\theta}{\cos\phi}\dot{p} + p\left(\frac{\cos\theta\cos\phi\dot{\theta} + \sin\theta\sin\phi\dot{\phi}}{\cos^2\phi}\right)\right. \\ & \left. -r\left(\frac{\cos\theta\sin\phi\dot{\phi} - \sin\theta\cos\phi\dot{\theta}}{\cos^2\phi}\right)\right]\end{aligned} \tag{3.44}$$

3.2.1.4 Torque Compensation in Control Moment Input

With vehicle angular velocity and acceleration obtained, next we determine the control moment input $\boldsymbol{M}_c$ through the rotational dynamics equation and compensate the drag induced, gyroscopic, and rotor rolling torques here. We write the rotational dynamics equation as:

$$\boldsymbol{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega} = \boldsymbol{M}_c + \boldsymbol{\tau}_D + \boldsymbol{\tau}_G + \boldsymbol{\tau}_R \tag{3.45}$$

Solving for the control moment input and expanding the additional torques we have:

$$\begin{aligned}\boldsymbol{M}_c = \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix} =& \boldsymbol{I}\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} + \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \boldsymbol{I}\begin{bmatrix} p \\ q \\ r \end{bmatrix} + (k_d\omega_s\boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T\dot{\boldsymbol{X}}) \times h\boldsymbol{e}_3 \\ & - \boldsymbol{I}_r\begin{bmatrix} 0 \\ 0 \\ \omega_p \end{bmatrix} \times \begin{bmatrix} p \\ q \\ r \end{bmatrix} - \omega_p C_R\boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T\dot{\boldsymbol{X}}\end{aligned} \tag{3.46}$$

3.2.1.5 Thrust Compensation and Rotor Speeds Update

Now we have found all the states and control inputs of the quadcopter derived from the trajectory x, y, z, ψ and their derivatives. We can further determine the individual rotor thrusts by:

$$\begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & L & 0 & -L \\ -L & 0 & L & 0 \\ k_M & -k_M & k_M & -k_M \end{bmatrix}^{-1} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \tag{3.47}$$

We obtain each rotor speed by solving the refined thrust equation. Because k_ω, k_z and k_h are small positive constants [26] and rotor speed is never negative, we have the solution:

$$\begin{aligned} F_i =& k_\omega \omega_i^2 - k_z v_z \omega_i + k_h v_h^2 \\ \Rightarrow \omega_i =& \frac{k_z v_z + \sqrt{(k_z v_z)^2 - 4k_\omega (k_h v_h^2 - F_i)}}{2k_\omega} \end{aligned} \tag{3.48}$$

Where $v_z = (\boldsymbol{R}_{IB}^T \dot{\boldsymbol{X}}) \cdot \boldsymbol{e}_3$ and $v_h = \|\boldsymbol{R}_{IB} P \boldsymbol{R}_{IB}^T \dot{\boldsymbol{X}}\|$. Then we update $\omega_{s,new} = (\omega_1 + \omega_2 + \omega_3 + \omega_4)$ and $\omega_{p,new} = (\omega_1 - \omega_2 + \omega_3 - \omega_4)$ and compare them with the initial guess. If $|\omega_{s,new} - \omega_s| > \sigma$ or $|\omega_{p,new} - \omega_p| > \sigma$, we update the sum and difference of rotor speeds as $\omega_s = \omega_{s,new}$ and $\omega_p = \omega_{p,new}$ and go through the inverse dynamics analysis process again until the tolerance is satisfied. Here σ is a user defined tolerance and in this research we use $\sigma = 1(\mathrm{rpm})$. From figure 3.6 and figure 3.7 we can see that this iterative method is practical and only takes 2 to 3 iterations to converge for this example 4-waypoit aggressive trajectory.

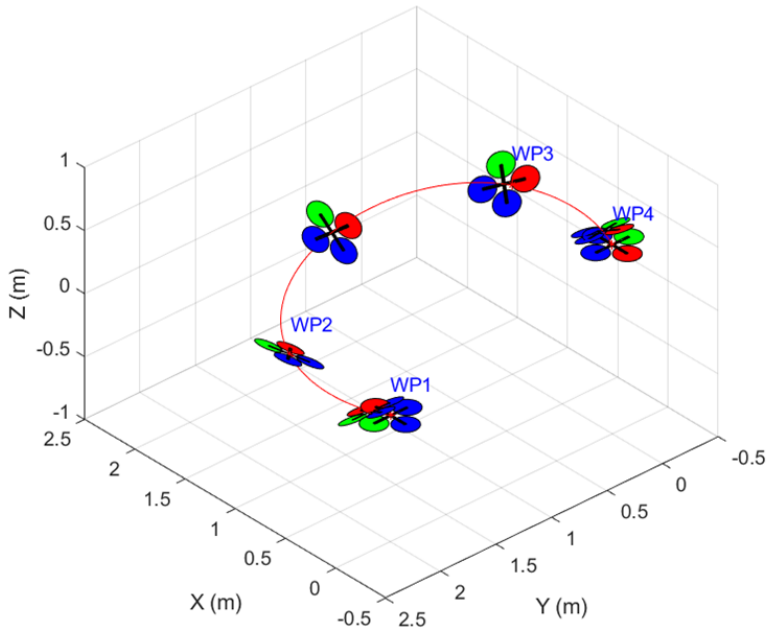

Figure 3.6. Simple 4-waypoint aggressive trajectory.

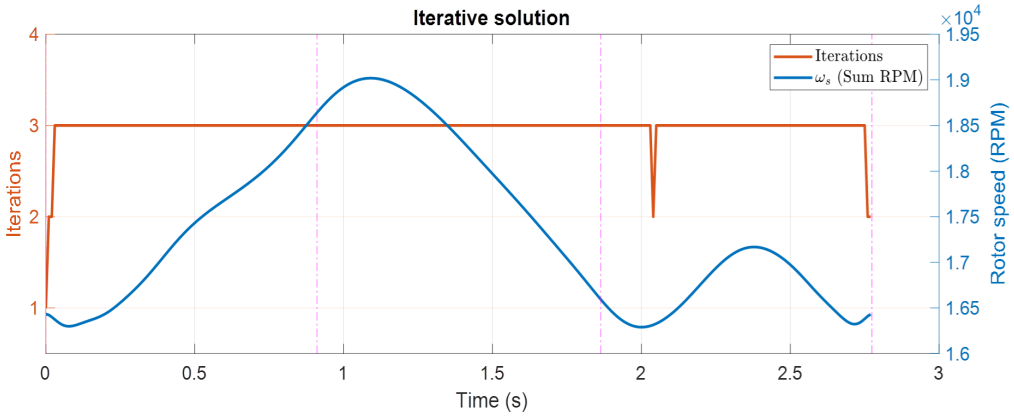

Figure 3.7. Iterations taken and resultant ω_s in the inverse dynamics analysis.

3.2.2 Drag Compensation in Waypoint Attitude Constraint

In our previous work [35], we posed the requirement to pass through narrow windows as trajectory derivative constraints in the polynomial trajectory optimization problem. However, the presence of aerodynamic drag force changes the vehicle attitude along the trajectory. We take the resultant window passing trajectory in [35] for example. Figure 3.8 shows the vehicle attitude and body z-axis ($\boldsymbol{z}_B$) at each waypoint in the case of accurately tracking the trajectory with drag force simulated. We can see that, at waypoint 2 and 3, the $\boldsymbol{z}_B$ vector is no longer aligned with the window upward vector W_U. Therefore, the safely passage through the narrow window is no longer guaranteed.

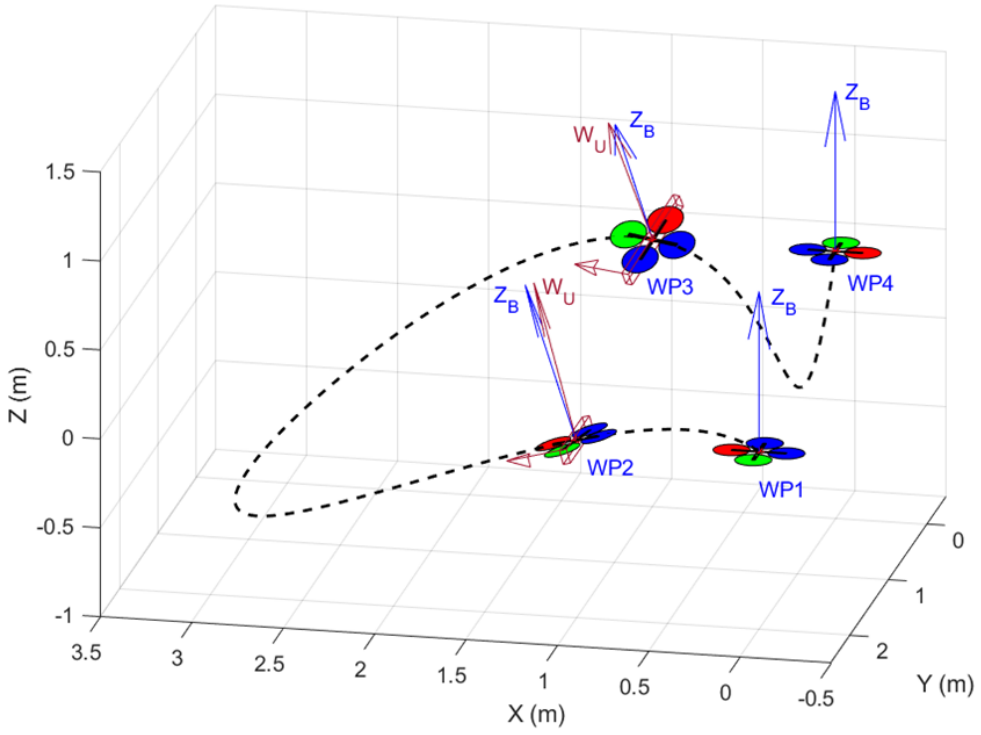

Figure 3.8. Plot of vehicle attitude at waypoints – drag force not considered in planning.

To solve this problem, we used iterative method again. First we make the initial guess on the sum of rotor speeds:

$$\omega_s = 4\sqrt{\frac{mg}{4k_F}} \tag{3.49}$$

Add the drag term into the equation of acceleration:

$$m\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = \begin{bmatrix}0\\ 0\\ -mg\end{bmatrix} + \boldsymbol{R}_{IB}\begin{bmatrix}0\\ 0\\ 1\end{bmatrix}\sum F - k_d\omega_s\boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T\boldsymbol{v} \tag{3.50}$$

Since we specify that, at the specific waypoint, the velocity is along the window forward vector W_F and the quadcopter body z-axis $\boldsymbol{z}_B$ is aligned with the window upward vector W_U [35] as shown in figure 3.9, the drag term can be simplified as:

$$k_d\omega_s\boldsymbol{R}_{IB}P\boldsymbol{R}_{IB}^T\boldsymbol{v} = k_d\omega_s\boldsymbol{v} \tag{3.51}$$

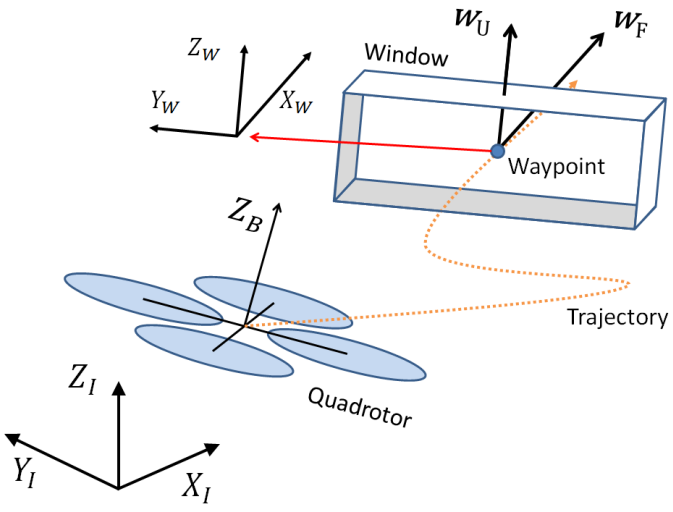

Figure 3.9. Schematic of a quadcopter flying through a narrow window.

The equation of acceleration can be rewritten as:

$$m\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix}+\begin{bmatrix}0\\ 0\\ mg\end{bmatrix}+k_d\omega_s\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{bmatrix}=\begin{bmatrix}m\ddot{x}+k_d\omega_s\dot{x}\\ m\ddot{y}+k_d\omega_s\dot{y}\\ m(\ddot{z}+g)+k_d\omega_s\dot{z}\end{bmatrix}=\boldsymbol{z}_B\cdot\sum F \tag{3.52}$$

To have $\boldsymbol{z}_B$ aligned with W_U, the constraints for the waypoint 1st and 2nd order derivatives can be composed as:

$$\begin{bmatrix}m\ddot{x}+k_d\omega_s\dot{x}\\ m\ddot{y}+k_d\omega_s\dot{y}\\ m(\ddot{z}+g)+k_d\omega_s\dot{z}\end{bmatrix}\times\begin{bmatrix}W_{\mathrm{U}_x}\\ W_{\mathrm{U}_y}\\ W_{\mathrm{U}_z}\end{bmatrix}=\begin{bmatrix}0\\ 0\\ 0\end{bmatrix} \tag{3.53}$$

$$\Rightarrow\begin{bmatrix}(m\ddot{y}+k_d\omega_s\dot{y})W_{\mathrm{U}_z}-(m\ddot{z}+k_d\omega_s\dot{z})W_{\mathrm{U}_y}\\ (m\ddot{z}+k_d\omega_s\dot{z})W_{\mathrm{U}_x}-(m\ddot{x}+k_d\omega_s\dot{x})W_{\mathrm{U}_z}\\ (m\ddot{x}+k_d\omega_s\dot{x})W_{\mathrm{U}_y}-(m\ddot{y}+k_d\omega_s\dot{y})W_{\mathrm{U}_x}\end{bmatrix}=\begin{bmatrix}mgW_{\mathrm{U}_y}\\ -mgW_{\mathrm{U}_x}\\ 0\end{bmatrix} \tag{3.54}$$

Notice that in our previous method proposed in [35] with simple dynamic model, this part only involves the trajectory acceleration and the window upward vector. However, here the trajectory velocity and the vehicle mass are also involved because additional aerodynamic drag is introduced. For these constraints, the relationship between x, y and z derivatives needs to be specified. Therefore, the joint optimization of x, y and z should be composed by concatenating their cost and constraint matrices in a block-diagonal fashion as:

$$J_{xyz}=\begin{bmatrix}\bar{\bar{p}}_x\\ \bar{\bar{p}}_y\\ \bar{\bar{p}}_z\end{bmatrix}^T\begin{bmatrix}\boldsymbol{Q}_x & 0 & 0\\ 0 & \boldsymbol{Q}_y & 0\\ 0 & 0 & \boldsymbol{Q}_z\end{bmatrix}\begin{bmatrix}\bar{\bar{p}}_x\\ \bar{\bar{p}}_y\\ \bar{\bar{p}}_z\end{bmatrix} \tag{3.55}$$

$$\begin{bmatrix} \boldsymbol{A}_x & 0 & 0 \\ 0 & \boldsymbol{A}_y & 0 \\ 0 & 0 & \boldsymbol{A}_z \end{bmatrix} \begin{bmatrix} \bar{\bar{p}}_x \\ \bar{\bar{p}}_y \\ \bar{\bar{p}}_z \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_x \\ \boldsymbol{b}_y \\ \boldsymbol{b}_z \end{bmatrix} \tag{3.56}$$

Where $\bar{\bar{p}}_x = [\bar{p}_{x_1} \ \cdots \ \bar{p}_{x_M}]^T$, $\bar{\bar{p}}_y = [\bar{p}_{y_1} \ \cdots \ \bar{p}_{y_M}]^T$ and $\bar{\bar{p}}_z = [\bar{p}_{z_1} \ \cdots \ \bar{p}_{z_M}]^T$. With this joint optimization setup, the constraints for a window passage on waypoint s in an M-segment trajectory can be implemented as:

$$\begin{bmatrix} 0 & W_{\mathrm{F}_z}\boldsymbol{v}_{0,s} & -W_{\mathrm{F}_y}\boldsymbol{v}_{0,s} \\ -W_{\mathrm{F}_z}\boldsymbol{v}_{0,s} & 0 & W_{\mathrm{F}_x}\boldsymbol{v}_{0,s} \\ W_{\mathrm{F}_y}\boldsymbol{v}_{0,s} & -W_{\mathrm{F}_x}\boldsymbol{v}_{0,s} & 0 \\ 0 & W_{\mathrm{U}_z}(m\boldsymbol{a}_{0,s} + k_d\omega_s\boldsymbol{v}_{0,s}) & -W_{\mathrm{U}_y}(m\boldsymbol{a}_{0,s} + k_d\omega_s\boldsymbol{v}_{0,s}) \\ -W_{\mathrm{U}_z}(m\boldsymbol{a}_{0,s} + k_d\omega_s\boldsymbol{v}_{0,s}) & 0 & W_{\mathrm{U}_x}(m\boldsymbol{a}_{0,s} + k_d\omega_s\boldsymbol{v}_{0,s}) \\ W_{\mathrm{U}_y}(m\boldsymbol{a}_{0,s} + k_d\omega_s\boldsymbol{v}_{0,s}) & -W_{\mathrm{U}_x}(m\boldsymbol{a}_{0,s} + k_d\omega_s\boldsymbol{v}_{0,s}) & 0 \end{bmatrix} \begin{bmatrix} \bar{\bar{p}}_x \\ \bar{\bar{p}}_y \\ \bar{\bar{p}}_z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ mgW_{\mathrm{U}_y} \\ -mgW_{\mathrm{U}_x} \\ 0 \end{bmatrix} \tag{3.57}$$

$$\begin{cases} \boldsymbol{v}_{0,s} = \begin{bmatrix} [0_{\times(n+1)}]_{\times(s-1)} & \boldsymbol{v}_0 & [0_{\times(n+1)}]_{\times(M-s)} \end{bmatrix} \\ \boldsymbol{a}_{0,s} = \begin{bmatrix} [0_{\times(n+1)}]_{\times(s-1)} & \boldsymbol{a}_0 & [0_{\times(n+1)}]_{\times(M-s)} \end{bmatrix} \end{cases} \tag{3.58}$$

For a segment polynomial $P(t) = p_0 + p_1t^1 + p_2t^2 + p_3t^3 + \cdots + p_nt^n$, the 1st order derivative (velocity) at $t = 0$ is $\boldsymbol{v}_0\bar{p}$ and $\boldsymbol{v}_0 = [0 \ 1 \ 0 \ 0 \ \cdots \ 0]$. The 2nd order derivative (acceleration) at $t = 0$ is $\boldsymbol{a}_0\bar{p}$ and $\boldsymbol{a}_0 = [0 \ 0 \ 2 \ 0 \ \cdots \ 0]$.

We obtain the minimum snap polynomial trajectory $\bar{\bar{p}}$ through the optimization process as in [35] with these additional constraints. Then we conduct the inverse dynamics analysis on each of the window waypoints with $\bar{\bar{p}}$ and find the actual sum of rotor speeds $\omega_{s,act} = (\omega_1 + \omega_2 + \omega_3 + \omega_4)$. If $|\omega_{s,act} - \omega_s| > \delta$, we update the sum of rotor speeds as $\omega_s = \omega_{s,act}$ and go through the whole process again until the tolerance is satisfied. Here δ is a user defined tolerance and in this research we use $\delta = 1(\mathrm{rpm})$.

3.3 Numerical Results

The quadcopter parameters we use in this research are from [40], [26] and Gazebo RotorS module [31] and tabulated in Table 3.1.

Table 3.1. Quadcopter parameters.

m	1.023 kg	g	$9.81\ \mathrm{m/s^2}$
L	0.2223 m	I_{xx}	$0.0095\ \mathrm{kg \cdot m^2}$
h	0.023 m	I_{yy}	$0.0095\ \mathrm{kg \cdot m^2}$
k_M	0.0197 m	I_{zz}	$0.0186\ \mathrm{kg \cdot m^2}$
k_ω	$1.4865 \times 10^{-7}\ \mathrm{N/RPM^2}$	k_d	$1.314 \times 10^{-5}\ \mathrm{N \cdot s/m/RPM}$
k_z	$2.55 \times 10^{-5}\ \mathrm{N \cdot s/m/RPM}$	I_r	$5.0 \times 10^{-5}\ \mathrm{kg \cdot m^2}$
k_h	$3.39 \times 10^{-3}\ \mathrm{N \cdot s^2/m^2}$	C_R	$1 \times 10^{-6}\ \mathrm{N \cdot s/RPM}$

3.3.1 Drag Induced, Gyroscopic and Rotor Rolling Torques

We simulated the drag induced, gyroscopic and rotor rolling torques in a simple 4-waypoint aggressive trajectory as shown in figure 3.10. The waypoint coordinates are as those listed in Table 3.2 and we specified a counter clockwise yaw trajectory from 0° to $+180^\circ$. This is not an extreme case but the counter clockwise turning increased the yawing control effort and thereby enhanced the gyroscopic and rotor rolling effects.

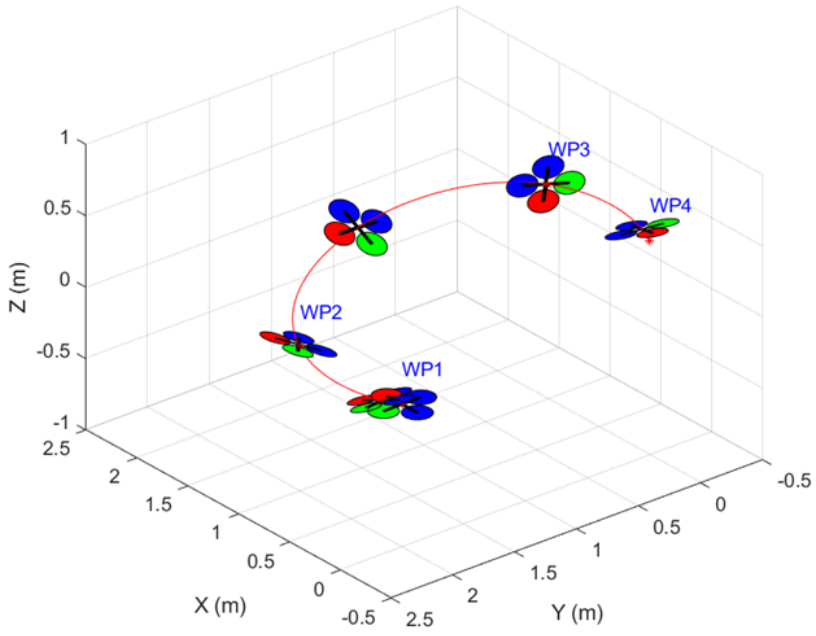

Figure 3.10. Simple 4-waypoint aggressive trajectory with counter-clockwise turning.

Figure 3.11∼3.13 show the magnitude of drag induced, gyroscopic and rotor rolling torque comparing to the control moment inputs in this scenario. From this result we learned that these effects could be significant in quadcopter aggressive flight and should not be neglected. They should be estimated and compensated in both trajectory planning and flight control for accurate flight condition prediction and trajectory tracking. From the figures we can also find that gyroscopic and rotor rolling effects take place when there is effort in yawing control. Besides the compensation, one alternative way to avoid these two effects is to take the optimal yaw trajectory proposed in [35] which eliminates yawing control effort.

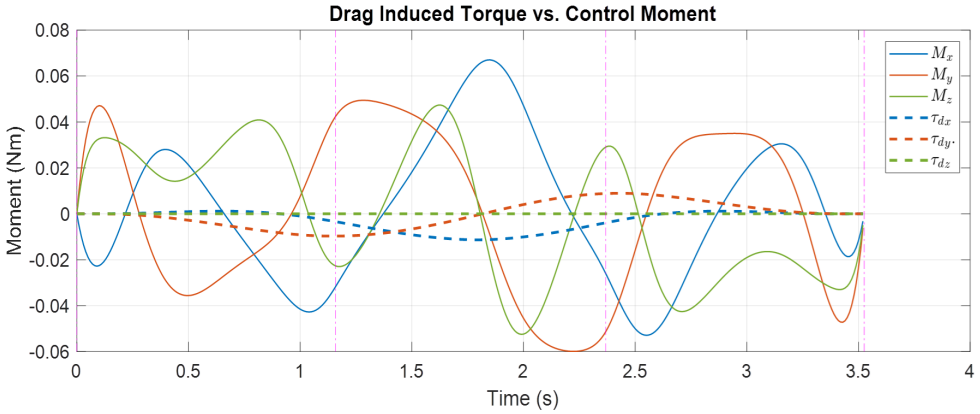

Figure 3.11. Drag induced torque and control moment along the trajectory flight.

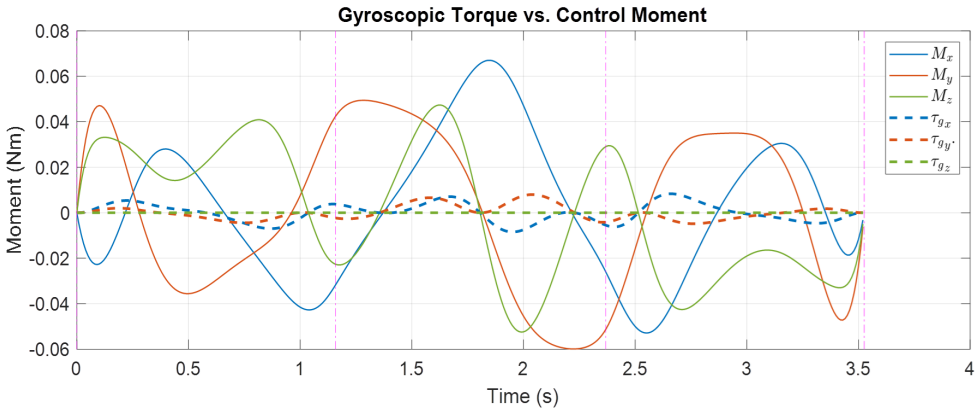

Figure 3.12. Gyroscopic torque and control moment along the trajectory flight.

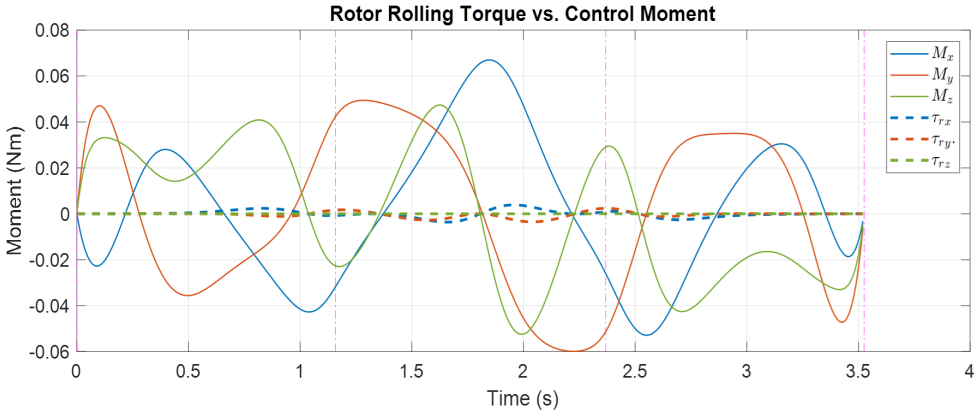

Figure 3.13. Rotor rolling torque and control moment along the trajectory flight.

Figure 3.14 shows the combined effect of all the three torques. In this test case, the combined torque can be as high as 22.7% about the x-axis and 19.9% about the y-axis comparing to the control moment.

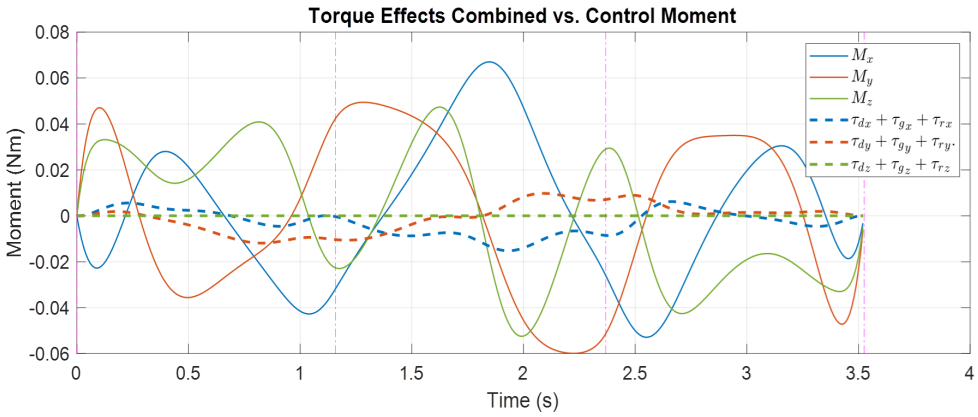

Figure 3.14. Combined torque effect and control moment along the trajectory flight.

3.3.2 Window Passing Trajectory Planning

We test the optimization framework with a 4-waypoint scenario as in Section 2.3. The waypoint settings are tabulated in Table 3.2 (yaw angle at waypoint 2 and 3 are not specified).

Table 3.2. Waypoint settings in the scenario.

WPT Setting	x (m)	y (m)	z (m)	ψ (deg)
Waypoint 1	0	2	0	0
Waypoint 2	1	2	0	-
Waypoint 3	1	0	0.5	-
Waypoint 4	0	0	0.5	180

We specify the mission to be from rest to rest, therefore the velocity and acceleration at the first and the last waypoint are constrained to be zero. There are two narrow windows to pass through in this scenario. The window settings are tabulated in Table 3.3. We also specify the $\mathrm{Aggressiveness}$ to be 80%. For this quadcopter model the rotor force for steady hover is $F_{hover} = 2.5$(N). Assuming $F_{max} = 3.75$(N), we have the maximum rotor force we can use for the trajectory as $F_{max\_req} = 3.5$(N).

Table 3.3. Window (WDW) settings in the scenario.

WDW Setting	ϕ	θ	ψ	Location
Window 1	0°	15°	0°	Waypoint 2
Window 2	-30°	0°	-20°	Waypoint 3

Figure 3.15 shows the comparison of the original trajectory with simple model used in [35] and the new trajectory with aerodynamic, gyroscopic and rotor rolling effects estimated and compensated. We can see that the presence of aerodynamic drag changes the

optimal trajectory to a considerable extent. It is also interesting to see that, for the same maximum rotor thrust, the case with aerodynamic drag leads to a shorter and faster trajectory. The original trajectory takes 6.2 seconds while the new one only takes 5.74 seconds. This counterintuitive result indicates that the aerodynamic drag somehow benefits the aggressive maneuver. The main reason we found behind this is that the consideration of drag force changes the trajectory derivative constraints at the window waypoints and thus affects the trajectory acceleration direction on those waypoints. And in this particular scenario setting, this change happens to be in favor of the aggressive maneuver and makes the resultant trajectory more "natural" with less maximum rotor thrust required at the sharp turn between waypoint 2 and waypoint 3. As a result, the new case allows a faster trajectory with the same maximum thrust though the aerodynamic drag is applied.

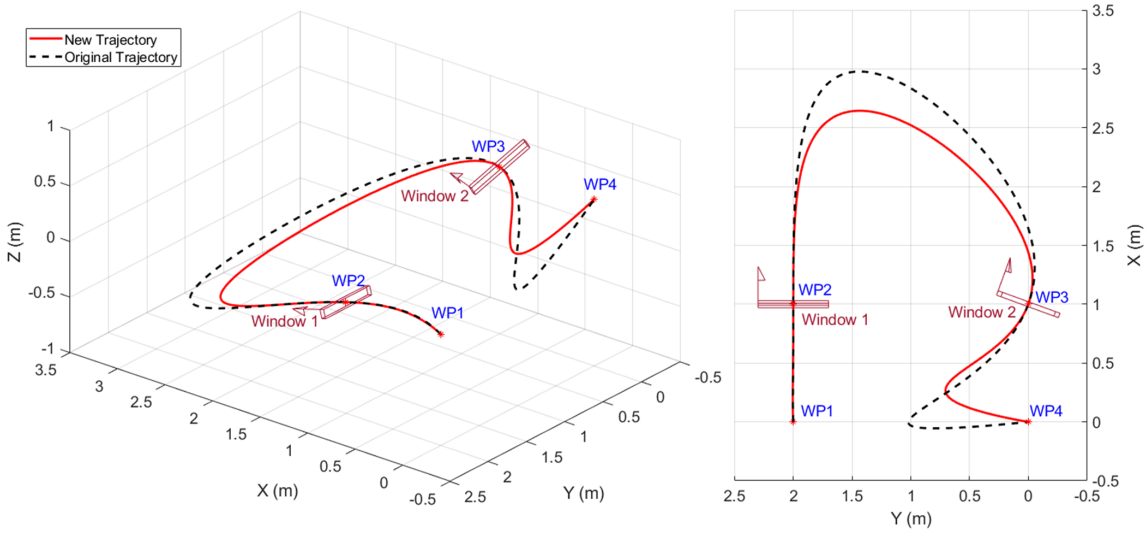

Figure 3.15. Comparison between the new trajectory and the original one.

To further explain this phenomenon, we change the scenario slightly by modifying the pitch angle of window 1 and the roll angle of window 2 both to 0°. Figure 3.16 shows

the comparison of the trajectories generated with and without the consideration of aerodynamic drag for this modified scenario. In this case, the two optimal trajectories have a similar traverse time as 4.1 seconds. We can see that the shapes of the two trajectories are still different. However, the change of the trajectory derivative constraints at window waypoints did not make obvious advantage or disadvantage for the aggressive maneuver in this modified scenario.

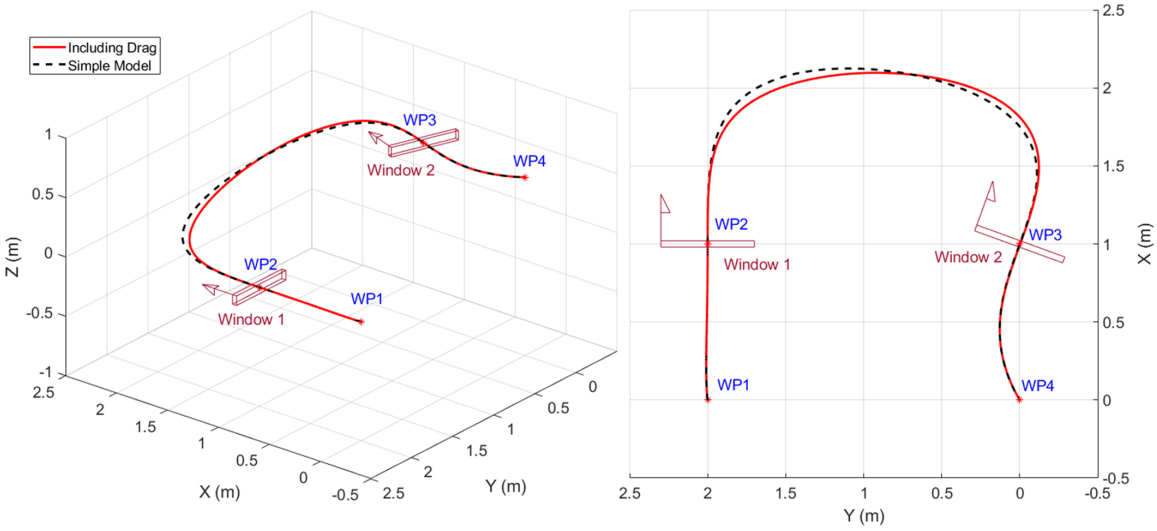

Figure 3.16. Comparison between the trajectories for the modified scenario.

Next we focus on the inverse dynamics analysis result for the new trajectory shown in figure 3.15. After all the compensations for aerodynamic drag, refined thrust model, gyroscopic and rotor rolling effects, we check whether the scenario requirements are all fulfilled. From Figure 3.17 we can see that the estimated maximum rotor thrust is indeed 3.5 N as specified and this maximum thrust happens at the sharp turn between waypoint 2 and waypoint 3.

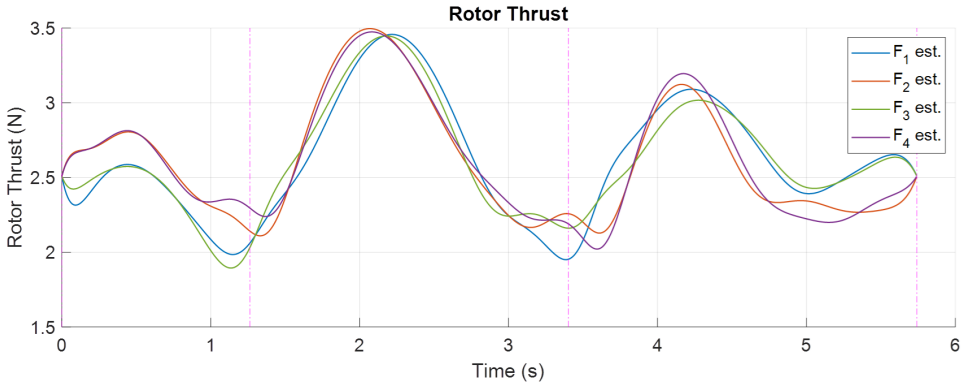

Figure 3.17. Individual rotor thrusts along the trajectory flight.

Figure 3.18 shows the track of vehicle orientation along the trajectory. And from figure 3.19 we can see that, at waypoint 2 and 3, the vehicle z_B vector is indeed aligned with the window upward vector W_U. This ensures the safely passage through the windows with aerodynamic drag considered. More detailed inverse dynamics analysis estimations can be seen in the trajectory tracking flight simulation comparison plots in the next section.

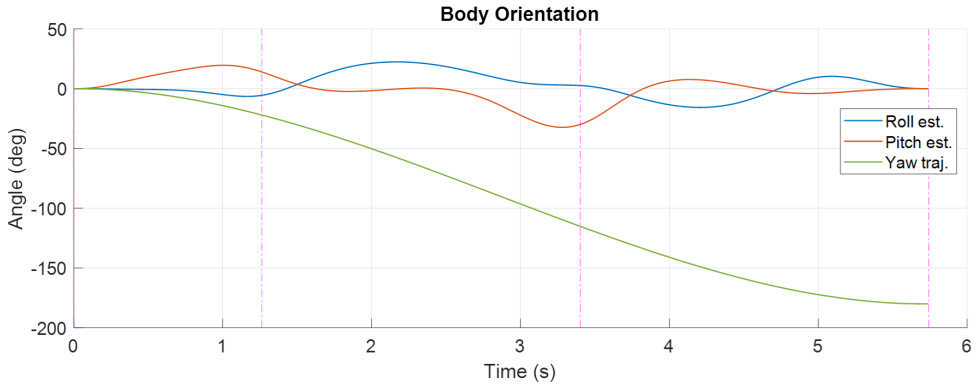

Figure 3.18. Vehicle Euler angles along the trajectory flight.

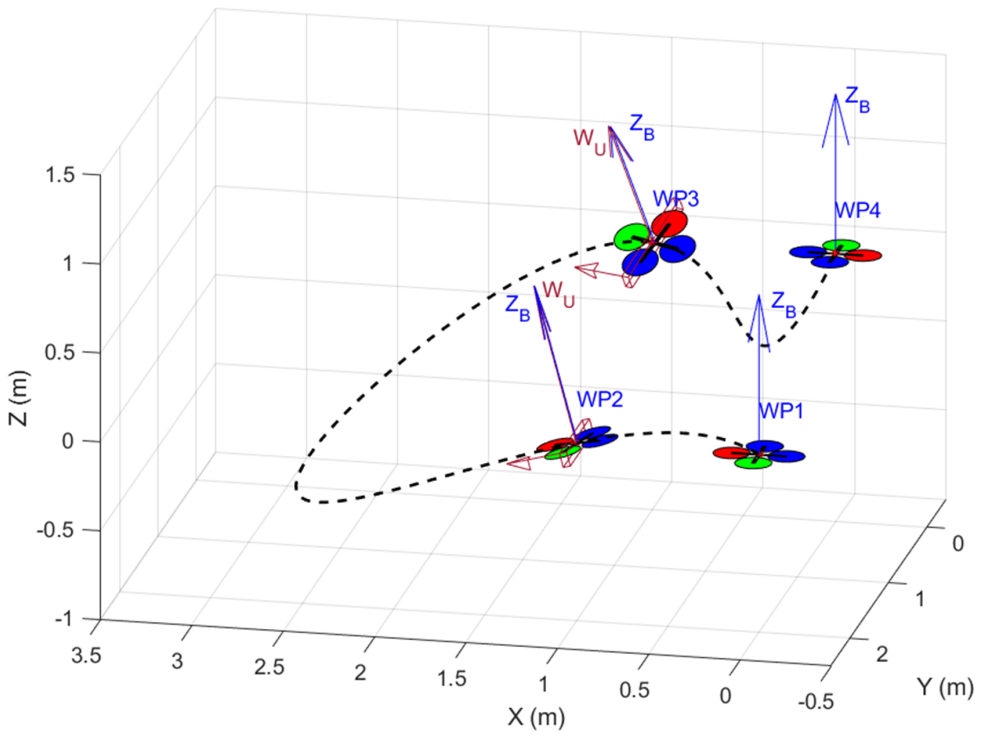

Figure 3.19. Plot of vehicle attitude at waypoints – drag force compensated in planning.

3.4 Trajectory Tracking Control and Simulation Result

To perform the polynomial trajectory tracking and verify the result of the aggressive trajectory optimization with aerodynamic, gyroscopic and rotor rolling effects, we adopted the geometric controller proposed by Lee et al. in [41] , modified it with our compensations and implemented it in MATLAB/Simulink. Utilizing the derivations in Section 3.2.1.1, we have the control input of total force f and desired rotation matrix $\boldsymbol{R}_d$ obtained for the trajectory tracking part with position error $\boldsymbol{e}_x$ and velocity $\boldsymbol{e}_v$:

$$\begin{aligned} \boldsymbol{e}_x &= \boldsymbol{x} - \boldsymbol{X} \\ \boldsymbol{e}_v &= \boldsymbol{v} - \dot{\boldsymbol{X}} \end{aligned} \tag{3.59}$$

$$\boldsymbol{F}_d + \boldsymbol{D} = f\boldsymbol{R}_d\boldsymbol{e}_3 - k_d\omega_s\boldsymbol{R}_d P \boldsymbol{R}_d^T \dot{\boldsymbol{X}} = m\ddot{\boldsymbol{X}} + mg\boldsymbol{e}_3 - k_x\boldsymbol{e}_x - k_v\boldsymbol{e}_v \tag{3.60}$$

$$\Rightarrow \begin{cases} \boldsymbol{z}_B = \frac{k_d\omega_s\dot{\boldsymbol{X}} + m(\ddot{\boldsymbol{X}} + g\boldsymbol{e}_3) - k_x\boldsymbol{e}_x - k_v\boldsymbol{e}_v}{\|k_d\omega_s\dot{\boldsymbol{X}} + m(\ddot{\boldsymbol{X}} + g\boldsymbol{e}_3) - k_x\boldsymbol{e}_x - k_v\boldsymbol{e}_v\|} \\ f = m\boldsymbol{z}_B^T(\ddot{\boldsymbol{X}} + g\boldsymbol{e}_3) \end{cases} \tag{3.61}$$

$$\begin{cases} \boldsymbol{x}_C = \begin{bmatrix} \cos\psi & \sin\psi & 0 \end{bmatrix}^T \\ \boldsymbol{y}_B = \frac{\boldsymbol{z}_B \times \boldsymbol{x}_C}{\|\boldsymbol{z}_B \times \boldsymbol{x}_C\|} \\ \boldsymbol{x}_B = \boldsymbol{y}_B \times \boldsymbol{z}_B \\ \boldsymbol{R}_d = \begin{bmatrix} \boldsymbol{x}_B & \boldsymbol{y}_B & \boldsymbol{z}_B \end{bmatrix} \end{cases} \tag{3.62}$$

With the desired rotation matrix $\boldsymbol{R}_d$ and derivatives of the polynomial trajectory available, we can find the desired angular velocity $\boldsymbol{\omega}_d = [p \quad q \quad r]^T$ by following Section 3.2.1.2, and find the desired angular acceleration $\dot{\boldsymbol{\omega}}_d = [\dot{p} \quad \dot{q} \quad \dot{r}]^T$ by following Section 3.2.1.3. Then we find attitude error $\boldsymbol{e}_R$ and angular velocity error $\boldsymbol{e}_\omega$ and add torque compensations to the control input of the moments M for the attitude tracking part as:

$$\begin{aligned} \boldsymbol{e}_R &= \frac{1}{2}(\boldsymbol{R}_d^T\boldsymbol{R} - \boldsymbol{R}^T\boldsymbol{R}_d)^\vee \\ \boldsymbol{e}_\omega &= \boldsymbol{\omega} - \boldsymbol{R}^T\boldsymbol{R}_d\boldsymbol{\omega}_d \end{aligned} \tag{3.63}$$

$$
\begin{aligned}
\boldsymbol{M} =& -k_R\boldsymbol{e}_R - k_\omega\boldsymbol{e}_\omega + \boldsymbol{\omega}\times\boldsymbol{I}\boldsymbol{\omega} - \boldsymbol{I}(\hat{\boldsymbol{\omega}}\boldsymbol{R}^T\boldsymbol{R}_d\boldsymbol{\omega}_d - \boldsymbol{R}^T\boldsymbol{R}_d\dot{\boldsymbol{\omega}}_d) - \boldsymbol{\tau}_D - \boldsymbol{\tau}_G + \boldsymbol{\tau}_R \\
=& -k_R\boldsymbol{e}_R - k_\omega\boldsymbol{e}_\omega + \boldsymbol{\omega}\times\boldsymbol{I}\boldsymbol{\omega} - \boldsymbol{I}(\hat{\boldsymbol{\omega}}\boldsymbol{R}^T\boldsymbol{R}_d\boldsymbol{\omega}_d - \boldsymbol{R}^T\boldsymbol{R}_d\dot{\boldsymbol{\omega}}_d) \\
&+ (k_d\omega_s\boldsymbol{R}P\boldsymbol{R}^T\boldsymbol{v})\times h\boldsymbol{e}_3 - \boldsymbol{I}_r\begin{bmatrix}0\\0\\\omega_p\end{bmatrix}\times\boldsymbol{\omega} - \omega_p C_R\boldsymbol{R}P\boldsymbol{R}^T\boldsymbol{v}
\end{aligned}
\tag{3.64}
$$

The simulation is implemented in MATLAB/Simulink with the following equations of motion:

$$
\begin{aligned}
\dot{\boldsymbol{x}} &= \boldsymbol{v} \\
m\dot{\boldsymbol{v}} &= -mg\boldsymbol{e}_3 + f\boldsymbol{R}\boldsymbol{e}_3 - k_d\omega_s\boldsymbol{R}P\boldsymbol{R}^T\boldsymbol{v}
\end{aligned}
\tag{3.65}
$$

$$
\begin{aligned}
&\dot{\boldsymbol{R}} = \boldsymbol{R}\hat{\boldsymbol{\omega}} \\
&\boldsymbol{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega}\times\boldsymbol{I}\boldsymbol{\omega} = \boldsymbol{M} - (k_d\omega_s\boldsymbol{R}P\boldsymbol{R}^T\boldsymbol{v})\times h\boldsymbol{e}_3 + \boldsymbol{I}_r\begin{bmatrix}0\\0\\\omega_p\end{bmatrix}\times\boldsymbol{\omega} + \omega_p C_R\boldsymbol{R}P\boldsymbol{R}^T\boldsymbol{v}
\end{aligned}
\tag{3.66}
$$

And we simulate rotor speeds with the model below:

$$
\begin{aligned}
\begin{bmatrix}F_1\\F_2\\F_3\\F_4\end{bmatrix} &= \begin{bmatrix}1 & 1 & 1 & 1\\0 & L & 0 & -L\\-L & 0 & L & 0\\k_M & -k_M & k_M & -k_M\end{bmatrix}^{-1}\begin{bmatrix}f\\M_x\\M_y\\M_z\end{bmatrix} \\
\omega_i &= \frac{k_z v_z + \sqrt{(k_z v_z)^2 - 4k_\omega(k_h v_h^2 - F_i)}}{2k_\omega}
\end{aligned}
\tag{3.67}
$$

Simulation results for tracking the new trajectory obtained in Section 3.3.2 are shown in figure 3.20. First we compare the effect of the compensations by using the same controller gains and adding the compensations successively. When using the original geometric

controller, we had significant tracking error as shown in the green line due to the presence of aerodynamic, gyroscopic and rotor rolling effects. Then we added the compensation of aerodynamic drag to the total force f and the desired rotation matrix $\boldsymbol{R}_d$ in the trajectory tracking part and the result turned out to be the blue line. The tracking performance is significantly improved but the tracking error is still quite noticeable. Then we further added the drag force compensation to the desired angular velocity $\boldsymbol{\omega}_d$ and angular acceleration $\dot{\boldsymbol{\omega}}_d$ in the attitude tracking part. The result is shown in the purple line which is already very close to the desired trajectory. Finally we added the gyroscopic, rotor rolling and drag induced torque compensations to the control moment input and this further improved the trajectory tracking as shown in the red line.

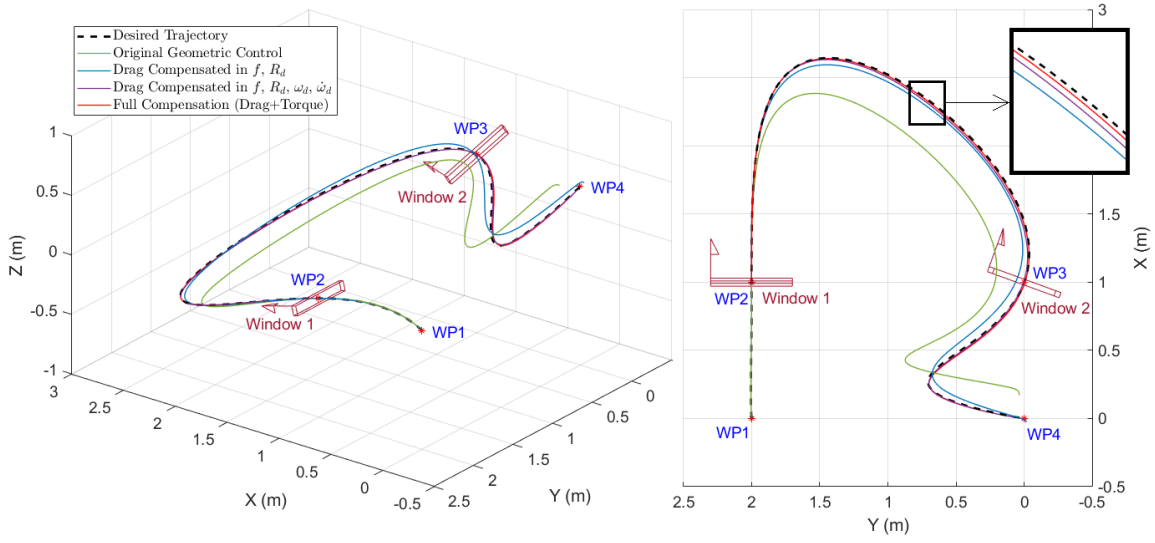

Figure 3.20. Trajectory tracking comparison for compensations added step by step.

From the 3 axes tracking error comparison in figure 3.21∼3.23 we can see more clearly how the compensations improved the tracking performance step by step. The cor-

responding RMS tracking errors are tabulated in Table 3.4. The drag compensation in f and $\boldsymbol{R}_d$ improved the x-axis and y-axis peak tracking error by around 78% but made the altitude tracking worse. This phenomenon is very close to the experimental result in [26] (x error improved by 75% from 26.0 to 6.6 cm, but z error got worse from 3.0 to 4.7 cm). Although the x and y tracking errors are much improved with this method, the overall tracking accuracy is not enough for our objective of precise narrow window passing and inverse dynamics analysis verification. Next coupled with our method of the drag compensation in $\boldsymbol{\omega}_d$ and $\dot{\boldsymbol{\omega}}_d$, the tracking error is further improved by 75% in x-axis, 63% in y-axis, and 91% in z-axis. Finally the torque compensations gave additional improvement of 6.5% in x-axis, 46% in y-axis and 65% in z-axis. With the full compensation, we had the RMS tracking error as 0.85 cm in x-axis, 0.53 cm in y-axis, and 0.12 cm in z-axis.

Table 3.4. The RMS Tracking Errors

	X-axis RMS Error (cm)	Y-axis RMS Error (cm)	Z-axis RMS Error (cm)
Original Geometric Control	16.83	12.21	2.22
Drag Compensation in f and $\boldsymbol{R}_d$ (Method in [8])	3.63	2.65	3.99
Drag Compensation in f, $\boldsymbol{R}_d$, $\boldsymbol{\omega}_d$ and $\dot{\boldsymbol{\omega}}_d$	0.91	0.99	0.34
Full Compensation (Drag + Torque)	0.85	0.53	0.12

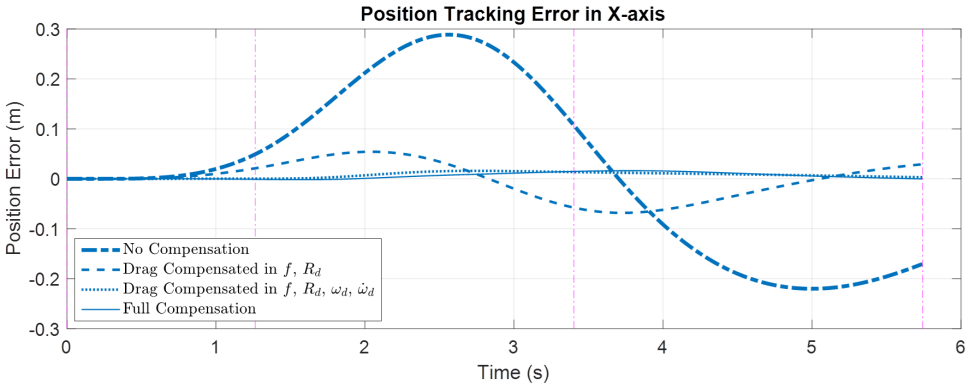

Figure 3.21. X-axis tracking error comparison.

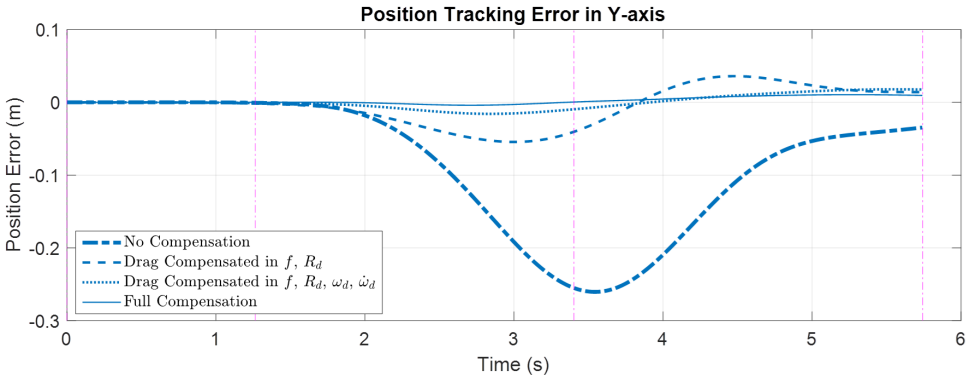

Figure 3.22. Y-axis tracking error comparison.

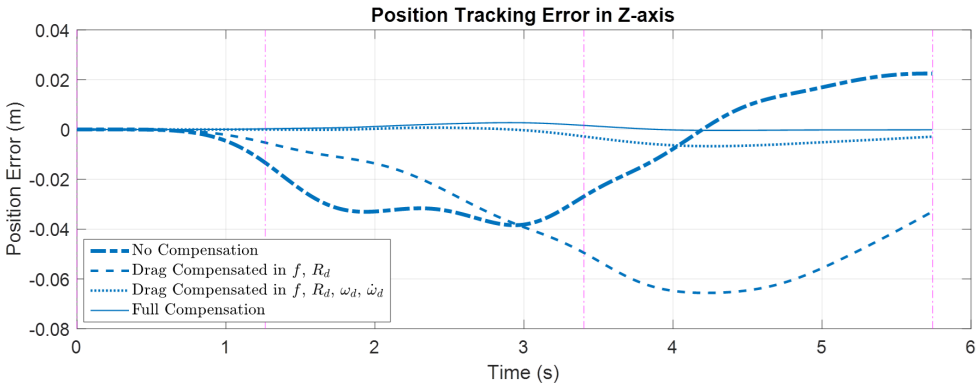

Figure 3.23. Z-axis tracking error comparison.

Having a good trajectory tracking performance with all the compensations, we then verify the inverse dynamics analysis estimation with the fully compensated controller tracking simulation result. First we check the rotational dynamics in figure 3.24 and 3.25. It can be seen that the angular velocity and angular acceleration being used to track the trajectory match our estimation very well. As a result, the body orientation in the simulation also turned out to be very close to the estimation. Having the trajectory well tracked and the vehicle orientation as expected, the requirement for narrow window passing is verified to be satisfied.

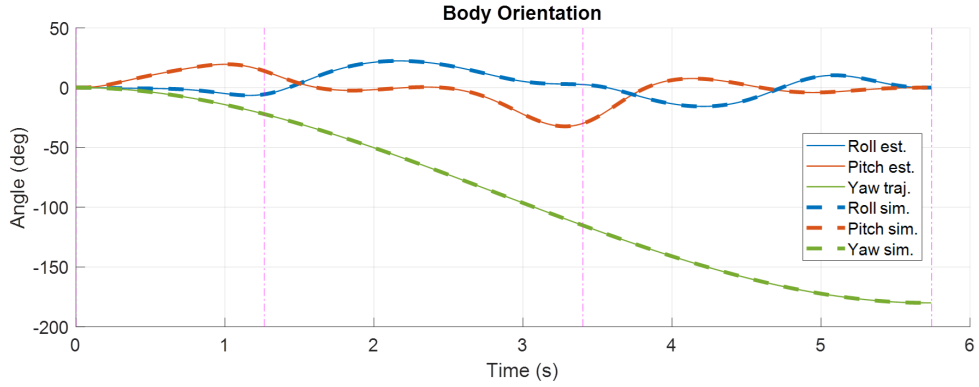

Figure 3.24. Comparison between simulation and estimation – vehicle attitude.

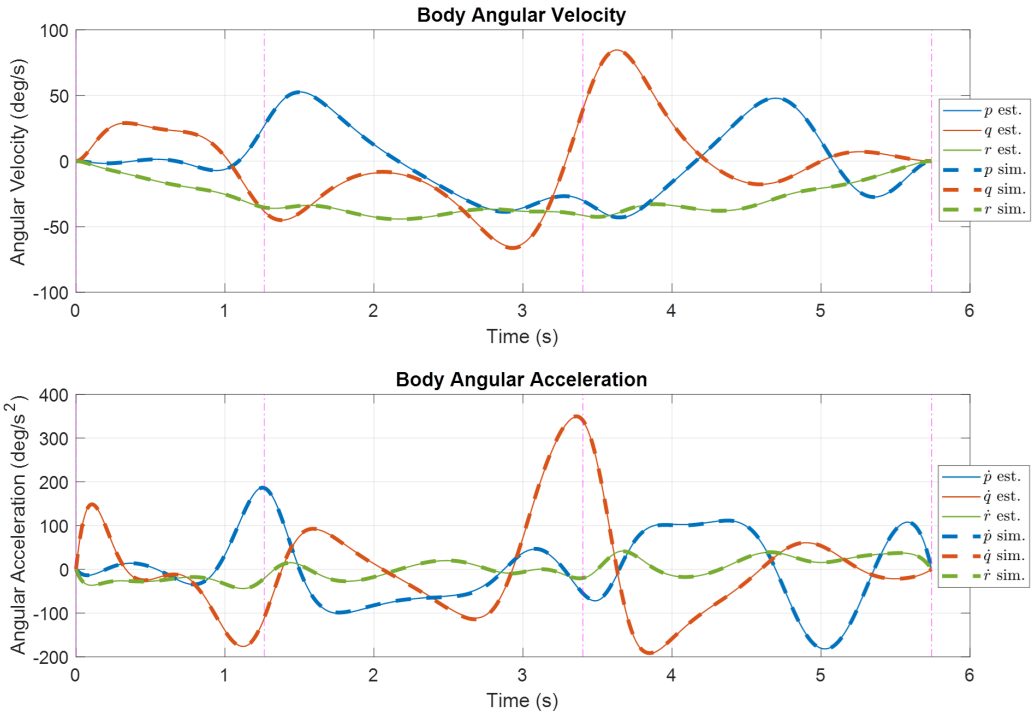

Figure 3.25. Comparison between simulation and estimation – angular velocity and acceleration.

Then we check the control input and actual rotor thrust in figure 3.26 and 3.27. From figure 3.27 we can see that the collective thrust and the 3-axis control moment being used to track the trajectory match our estimation very well. As a result, the individual rotor thrust in the simulation also turned out to be very close to our estimation and the requirement for maximum rotor thrust is satisfied.

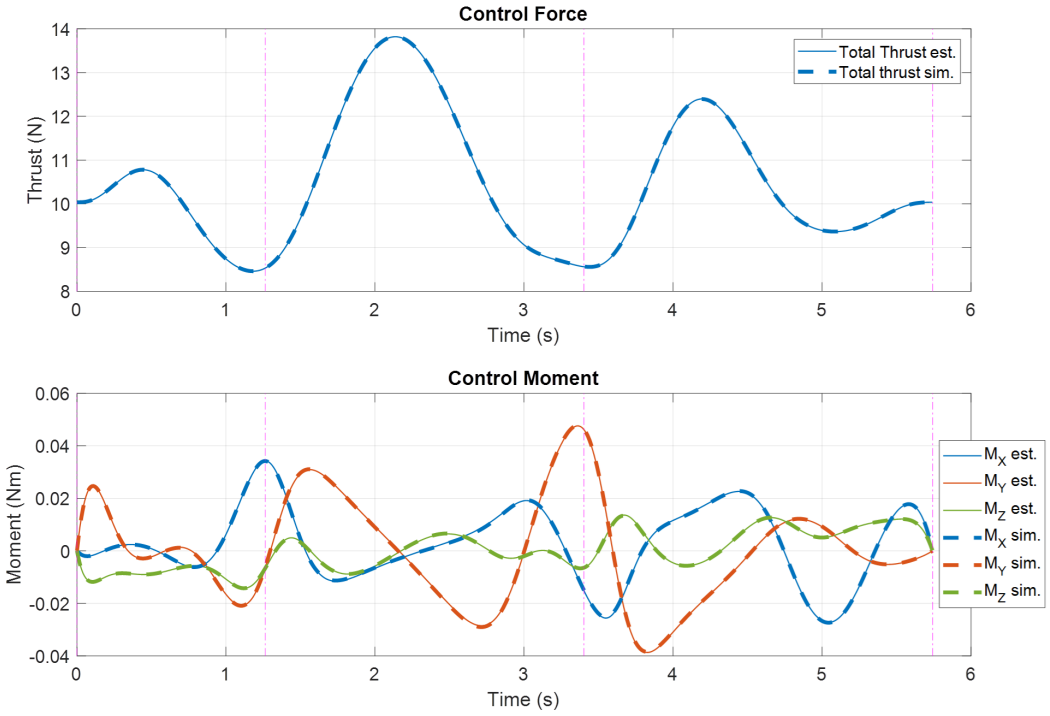

Figure 3.26. Comparison between simulation and estimation – control input.

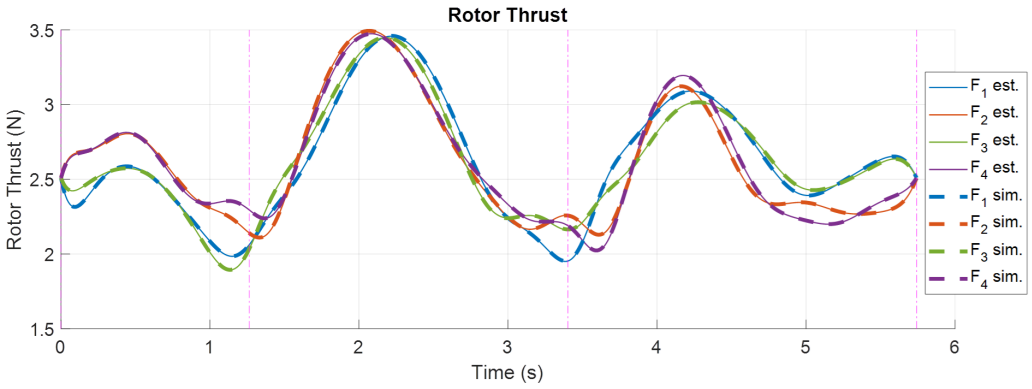

Figure 3.27. Comparison between simulation and estimation – rotor thrust.

3.5 Conclusions

The full compensation for aerodynamic, gyroscopic and rotor rolling effects has been achieved in this research both in optimal constrained trajectory planning and in geometric control trajectory tracking. In all of our test cases, the iterative method practically solved the rotor speeds which are essential for the estimation of all the additional effects. The case study provided in this chapter shows how the drag induced, gyroscopic and rotor rolling torques could be significant in a general aggressive trajectory flight and should not simply be neglected. In the window passing trajectory planning, we observed how the compensation for aerodynamic drag could significantly change the resultant aggressive trajectory and this change does not necessarily lead to disadvantage in constrained aggressive flight. In the geometric control trajectory tracking simulations, we observed how the compensations improved the tracking performance step by step and that these compensations are all important for the accurate aggressive trajectory tracking. From the full compensation tracking result, the resultant aggressive trajectory is verified to be feasible; the requirement for maximum rotor thrust and waypoint maneuver is verified to be reached; and the inverse dynamics analysis estimation is verified to fit the simulation result.

Chapter 4

Flight Test Environment Setup

The DJI F450 quadcopter equipped with Pixhawk autopilot and Raspberry Pi on-board computer shown in Figure 4.1 is used to conduct flight tests in this research. For accurate and real-time vehicle position/orientation information, we fly the quadcopter in a Vicon motion capture system equipped environment. To monitor the vehicle states and give control commands, we use a laptop running GCS (ground control station) software and a RC (remote control) transmitter. The overall system architecture and the data/signal flow are illustrated in Figure 4.2. Next in this chapter we divide the flight test environment into three parts, aerial system, ground control system, and Vicon motion capture system and discuss their major components and integration setup.

Figure 4.1. DJI F-450 quadcopter.

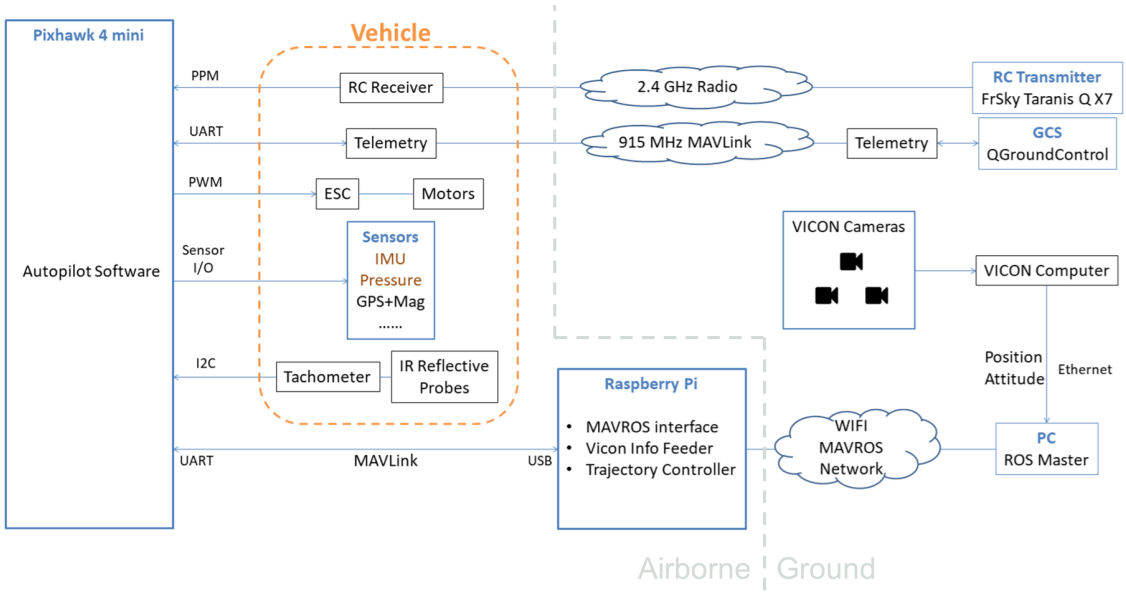

Figure 4.2. Flight test system architecture.

4.1 Aerial System

The main components used in the aerial system which is the quadcopter are listed and introduced below:

- Airframe – DJI F450:

  The DJI F450 shown in Figure 4.3 is a popular drone frame designed for aerial photography, FPV and other aero-modeling activities. It has a 450mm scale diagonal wheelbase. The F450 is part of DJI's FlameWheel series [42], which is known for its sturdy build and versatility.

Figure 4.3. DJI F450 Airframe.

- Propulsion Suit – DJI E305:

  The DJI E305 [43] propulsion suit shown in Figure 4.4 is combined with the DJI-430 Lite electronic speed controller (ESC) with maximum allowable continuous current as 30A, the DJI-2312E brushless motors with velocity constant as 800 Kv, and the DJI-9450 ABS self-locking propellers.

Figure 4.4. DJI E305 propulsion suit.

- RC Receiver – FrSky X4R-SB

  The FrSky X4R-SB [44] shown in Figure 4.5 is a 2.4G RC receiver which supports FrSky ACCST protocol and features 3 PWM outputs and 1 SBUS port that carries all 16 channels. We use the SBUS to send control commands to the autopilot device.

Figure 4.5. FrSky X4R-SB RC Receiver.

- Telemetry Radio – Holybro SiK 915MHz

  The Holybro SiK telemetry radio [45] shown in Figure 4.6 is a small, light and open source radio platform that typically allows ranges of better than 300m. This radio is plug-n-play all Pixhawk Standard and other flight controllers. We use this 915MHz 100mW telemetry radio to setup the telemetry connection between the autopilot device and the ground station.

Figure 4.6. Holybro SiK 915MHz telemetry radio.

- RPM Tachometer – ThunderFly TFRPM01

  The ThunderFly TFRPM01 [46] shown in Figure 4.7 is an open-source frequency sensor tachometer intended for the measurement of rotational actuators on drones. We use this tachometer with IR reflective optical probe to measure the rotor RPM and send out the readings through I2C. Because four tachometers with the same I2C address are used, we also add a TFI2CADT01 I2C address translator to collect the four signals and send all the readings to the autopilot device. The ThunderFly TFI2CADT01 [47] is a device translating addresses of I2C devices on a bus. As a result, multiple I2C slave devices with the same address can be connected to one master device.

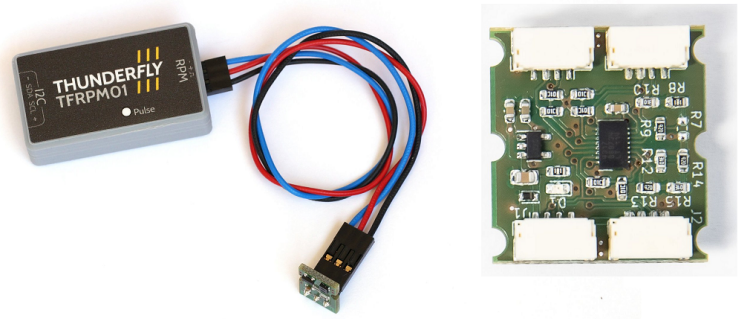

Figure 4.7. ThunderFly TFRPM01 RPM sensor (left) and TFI2CADT01 address translator (right).

Figure 4.8 shows our installation of the RPM Tachometer. Note that we attached 20 stripes on the motor shell to be detected by the IR reflective probe. This way, we can improve the high sampling rate RPM measurement resolution to 30 RPM at 10 Hz. The corresponding PX4 parameter settings are: PCF8583_MAGNET=20 and PCF8583_POOL=100000. Also note that the PX4 driver for this RPM tachometer is

not started automatically. We add the driver to the startup script on the SD card as suggested in [48].

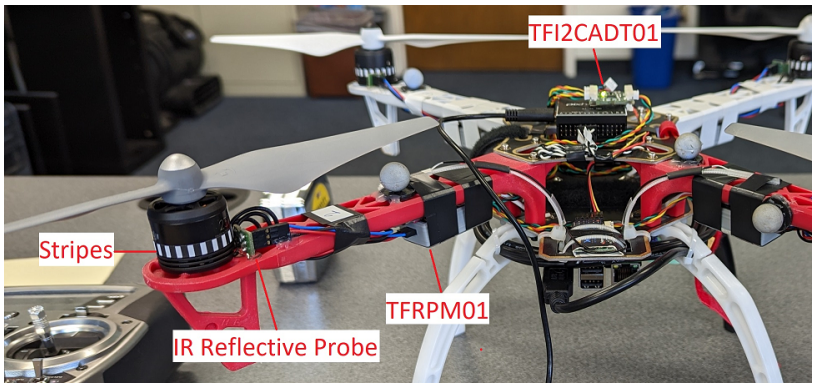

Figure 4.8. RPM tachometer installation.

- Autopilot Device – Holybro Pixhawk 4 Mini

  The Holybro Pixhawk 4 Mini [49] shown in Figure 4.9 is a small and versatile autopilot system designed for unmanned aerial vehicles (UAVs) or drones. It is a compact version of the popular Pixhawk 4 flight controller and is developed by Holybro, in collaboration with the Pixhawk team. The Pixhawk series is widely used in the drone industry for its robustness and reliability. Pixhawk 4 Mini features a powerful 32-bit ARM Cortex-M4F processor running at 168 MHz, along with a floating-point unit (FPU). It is equipped with on-board sensors such as two accelerometer and gyroscope sets, one magnetometer, and one barometer. It supports various communication interfaces, including UART, I2C, SPI, and CAN, allowing it to connect with a wide range of peripherals and sensors. It also has a microSD card slot for data logging.

  We run PX4 flight stack on the Pixhawk 4 Mini. The PX4 flight stack [50] is an open-source flight control software stack specifically designed for autonomous unmanned aerial vehicles (UAVs) and drones. It provides the core software components and

algorithms necessary for controlling the flight of a drone, including stabilization, navigation, and mission execution. It also provides flexible set of tools for drone developers to create tailored solutions for drone applications. For detailed description about our PX4 settings regarding system integration please refer to [51].

Figure 4.9. Holybro Pixhawk 4 Mini autopilot device.

- On-board Computer – Raspberry Pi 3B

  The Raspberry Pi 3B [52] shown in Figure 4.10 is a single-board computer developed by the Raspberry Pi Foundation. It is powered by a 1.2 GHz 64-bit quad-core ARM Cortex-A53 processor and includes 1GB of RAM, allowing for smoother multitasking and performance. The board features built-in Wi-Fi (802.11n) and Bluetooth 4.2, enabling wireless connectivity for internet access, wireless communication, and peripheral devices.

  We run Ubuntu MATE 18.04 operating system on the Raspberry Pi 3B and install ROS Melodic robotic system and MAVROS package. We use this on-board computer to make Wi-Fi connection to the Vicon MAVROS network. We also make serial connection to the autopilot device through an FTDI-cable. The MAVLink communication with PX4 flight stack is established by executing the launch file "px4.launch" in the MAVROS package [53]. We run programs to relay the vehicle position/attitude

information from Vicon system to the autopilot, receive vehicle information from the autopilot, and send control commands to the autopilot through MAVROS interface.

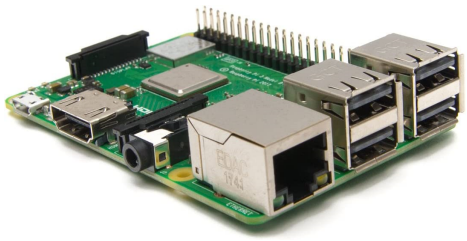

Figure 4.10. Raspberry Pi 3B on-board computer.

4.2 Ground Control System

The main purpose of the ground control system is to monitor the vehicle states and give control commands during the flight tests. Two major components are listed and introduced below:

- Ground Station – Acer Laptop

  One Acer laptop running QGroundControl software shown in Figure 4.11 serves as a ground station to monitor all the quadcopter state information during flight tests through the Holybro SiK 915MHz telemetry radio which is paired to the other one on the quadcopter. Besides flight tests, we also use this ground station to connect to the autopilot device to perform system integration test and parameter setting.

  QGroundControl [54] is an open-source ground control station (GCS) software designed for autonomous vehicles, particularly unmanned aerial vehicles (UAVs) and drones. It provides a user-friendly interface to control and monitor the flight of drones, plan missions, and access telemetry and flight data. It also offers firmware management capabilities, allowing users to update the firmware of their drones and connected flight controllers. QGroundControl also allows users to configure various

parameters and settings of their drones and flight controllers. This includes configuring flight modes, adjusting control parameters, setting up communication protocols, and calibrating sensors.

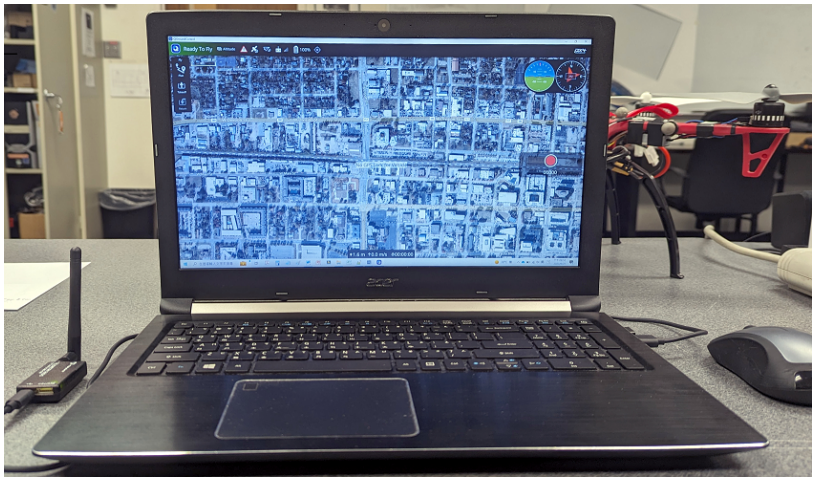

Figure 4.11. The ground station laptop.

- RC Transmitter – FrSky Taranis Q X7

  The FrSky Taranis Q X7 [55] shown in Figure 4.12 is a popular radio transmitter designed for remote control of RC (radio-controlled) models. The transmitter supports up to 16 channels, allowing control over various functions and features of RC models. It operates on the 2.4GHz frequency band and is compatible with FrSky's ACCST (Advanced Continuous Channel Shifting Technology) protocol, ensuring reliable and interference-free communication.

  We use this RC transmitter to send manual commands to the autopilot to perform takeoff and landing, check vehicle flight response, switch between control modes, and execute the kill switch to stop the rotors immediately in any dangerous case to ensure safety. We also use some additional switches and knobs on the transmitter to give commands to our vehicle control program running on the on-board computer to

help the proceeding of some flight experiments. This is done by coding the program to subscribe to MAVROS topic "/mavros/rc/in".

Figure 4.12. FrSky Taranis Q X7 RC transmitter.

4.3 Vicon Motion Capture System

Accurate and real-time vehicle position and orientation information is crucial to our flight experiments because we are performing precise aggressive trajectory tracking in this research. To acquire the crucial information in our flight tests, we utilize the Vicon motion capture system in the Guidance & Controls of Autonomous Systems Laboratory (GCASL) at The University of Texas at Arlington. This lab group also has some great endeavors on robot guidance control through fixed and moving orifices such as in [9] and [10].

Figure 4.13 shows the Vicon motion capture system setup. The Vicon system [56] utilizes a combination of specialized cameras and reflective markers to track the position and orientation of objects in real-time. The cameras are strategically placed around the capture area to provide optimal coverage. The markers, typically small retro-reflective spheres, are attached to the object being tracked. Vicon uses high-resolution cameras with high-speed image capture capabilities. These cameras capture the movement of the reflective markers in the capture volume at a high frame rate. The Vicon software analyzes the captured images from multiple cameras to reconstruct the 3D position and orientation of the

markers. By triangulating the position of each marker in 3D space using the known camera geometry, the system can precisely track the movement of objects. Figure 4.14 shows the Vicon computer screen displaying the captured marker locations and the quadcopter with retro-reflective markers flying under the Vicon infrared cameras.

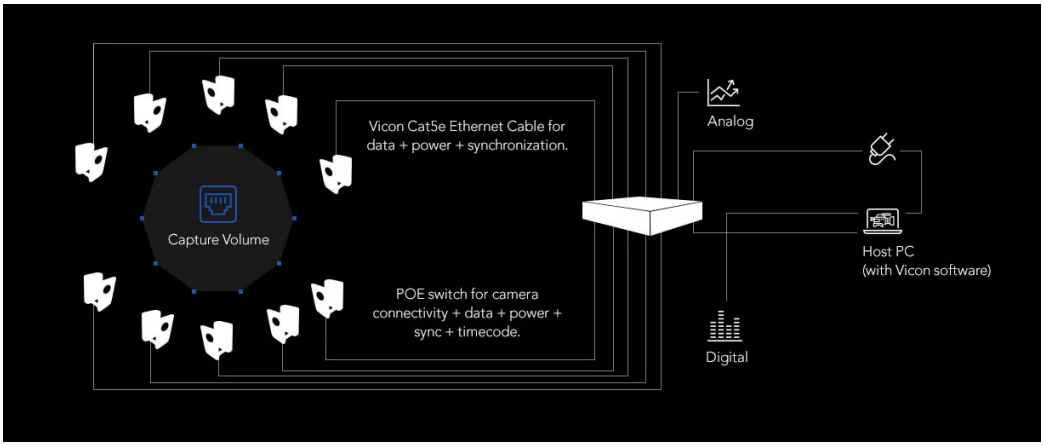

Figure 4.13. Vicon Motion Capture System.

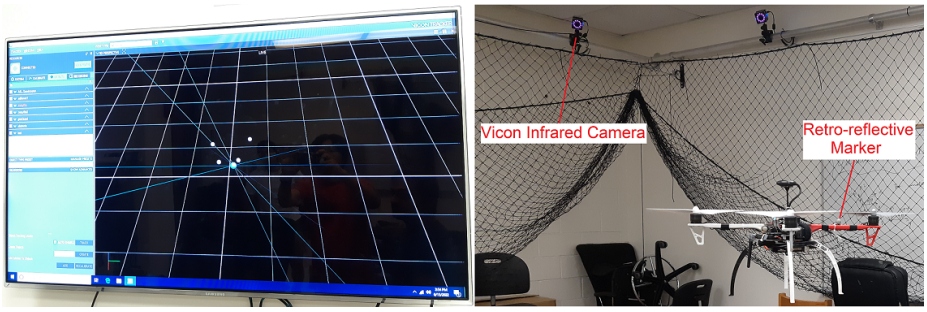

Figure 4.14. Vicon computer screen and the flight test.

In our application, one additional computer runs the "vicon_bridge" driver [57] through launch file "vicon.launch" to obtain vehicle position and orientation information from the Vicon computer and publish the information to MAVROS network. Then on the quadcopter

on-board computer, one program is coded to subscribe the vehicle information and publish it to MAVROS topic "mavros/vision_pose/pose". Once the MAVLink communication between the on-board computer and PX4 flight stack is established, PX4 will obtain the Vicon vehicle information automatically. We run this Vicon feed at 50 Hz.

To add this Vicon information to PX4 sensor fusion, we need to set up EKF2 parameters. EKF2_AID_MASK is set as vision position fusion and vision yaw fusion. EKF2_HGT_MODE is set to use the vision as a primary source for altitude estimation. And we set EKF2_EV_DELAY=5. Throughout this research, all the vehicle dynamic states are obtained from the PX4 EKF2 estimator. For detail parameter settings of the EKF2 estimator we use in this research, please refer to APPENDIX A.

Chapter 5

High Fidelity Simulations

To conduct the polynomial trajectory tracking and verify the result of the aggressive trajectory optimization, we perform the simulation in MATLAB/Simulink with the proposed quadcopter dynamic model and trajectory tracking controller in Chapter 2 and 3. But to be more realistic and prepare for the actual flight test, we set up two high fidelity simulation environments. In RotorS/Gazebo simulation, realistic sensor, actuator and controller implementations are included. Experience on working in the ROS (Robot Operating System) environment is also gained in this practice and found very helpful for subsequent works towards the actual flight test. In PX4 SITL simulation, we simulate the whole flight test system. We can test the planned flight experimental procedure as well as the programs we develop in Pixhawk autopilot and Raspberry Pi on-board computer.

5.1 RotorS/Gazebo Simulation

RotorS is a MAV (Micro Aerial Vehicle) gazebo simulator developed by the Autonomous Systems Lab at ETH Zurich [31,58]. It provides some multirotor models such as the AscTec Hummingbird, the AscTec Pelican, or the AscTec Firefly, but the simulator is not limited for the use with these multi-copters. There are simulated sensors coming with the simulator such as an IMU, a generic odometry sensor, and the VI-Sensor, which can be mounted on the multirotor. We chose RotorS because it is open-source, instantly ready for high fidelity and comprehensive MAV flight simulation, and widely used by researchers in the field of MAV control. We installed RotorS on Ubuntu 18.04 operating system with ROS Melodic robotic system environment and Gazebo 9.19.0 multi-robot simulator. Figure 5.1

shows a Gazebo screen displaying an AscTec Firefly hexacopter automatic flight simulated with RotorS.

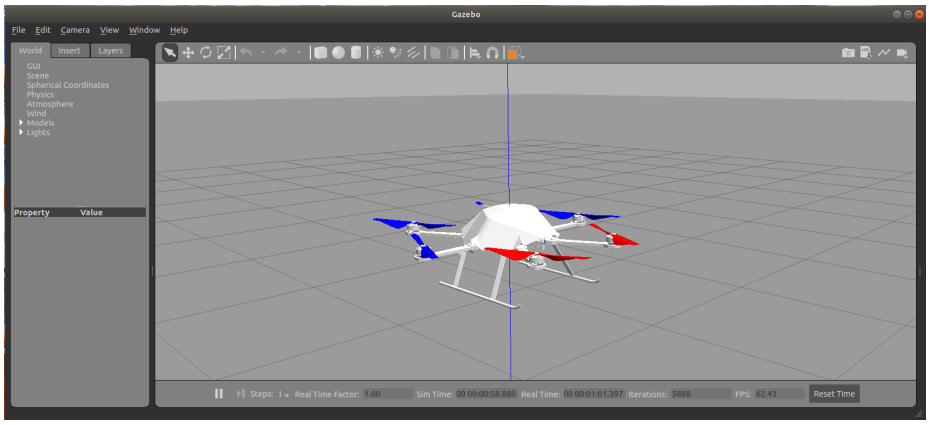

Figure 5.1. Gazebo screen of an AscTec Firefly hexacopter flight simulated with RotorS.

We use the built-in AscTec Firefly hexacopter model to verify the modified geometric controller proposed in Chapter 3 and also to show that our trajectory synthesis can be applied to generic multi-copters with minor modifications to the vehicle control allocation model. In this case, we consider six rotor outputs as shown in Figure 5.2 and modify equation (3.4) as:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ L\sin\frac{\pi}{6} & L & L\sin\frac{\pi}{6} & -L\sin\frac{\pi}{6} & -L & -L\sin\frac{\pi}{6} \\ -L\cos\frac{\pi}{6} & 0 & L\cos\frac{\pi}{6} & L\cos\frac{\pi}{6} & 0 & -L\cos\frac{\pi}{6} \\ -k_M & k_M & -k_M & k_M & -k_M & k_M \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \end{bmatrix} \tag{5.1}$$

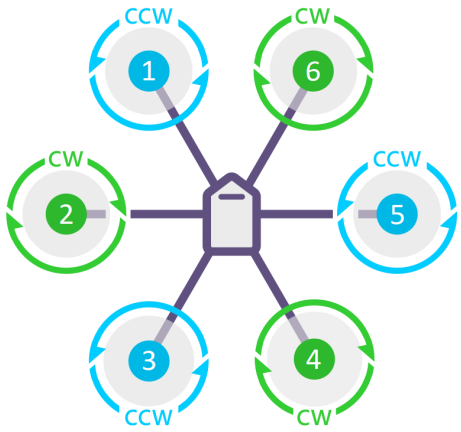

Figure 5.2. Rotor configuration of the hexacopter.

The parameters of the AscTec Firefly model in RotorS we use in the aggressive trajectory synthesis are tabulated in Table 5.1.

Table 5.1. Quadcopter parameters.

m	1.5 kg	I_{xx}	$0.0347563\ \mathrm{kg}\cdot\mathrm{m}^2$
L	0.215 m	I_{yy}	$0.0458929\ \mathrm{kg}\cdot\mathrm{m}^2$
h	0.037 m	I_{zz}	$0.0977\ \mathrm{kg}\cdot\mathrm{m}^2$
k_M	0.016 m	k_d	$8.06428\times10^{-5}\ \mathrm{N}\cdot\mathrm{s/m/(rad/s)}$
k_f	$1.4865\times10^{-6}\ \mathrm{N/(rad/s)}^2$		

For trajectory tracking control, the built-in "lee_position_controller.cpp" in rotors_control module already has the basic structure of the geometric controller [16] except for the rotational feedforward terms $\boldsymbol{\omega}_d$ and $\boldsymbol{I}(\hat{\boldsymbol{\omega}}\boldsymbol{R}^T\boldsymbol{R}_d\boldsymbol{\omega}_d - \boldsymbol{R}^T\boldsymbol{R}_d\dot{\boldsymbol{\omega}}_d)$ in equation (3.64). We modified this controller to fully implement the trajectory tracking controller with rotor drag compensation as proposed in Chapter 3.

For trajectory command assignment, we modified the built-in "waypoint_publisher.cpp" in rotors_gazebo module so that it reads multi-segment trajectory polynomial coefficients

from a data file and publishes trajectory position, velocity, acceleration, jerk, and snap commands through mav_msgs topic COMMAND_TRAJECTORY at 50 Hz.

5.1.1 1-D Minimum Snap Trajectory Tracking Test

We start the trajectory tracking test with a simple 1-segment 1-D minimum snap trajectory that goes from $x = 0$ to $x = 2$ in 3 seconds. Figure 5.3 shows the sequence stock image of the flight.

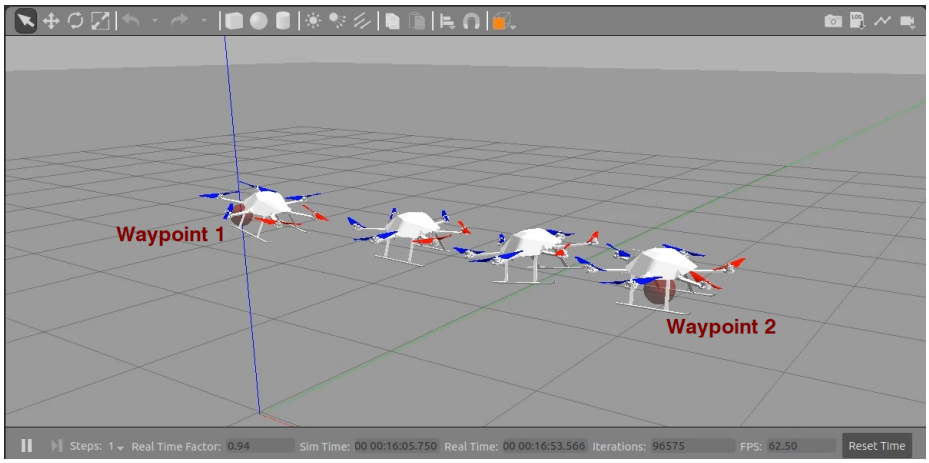

Figure 5.3. 1-D minimum snap trajectory tracking in RotorS simulation.

Using the same geometric controller gains, we conduct a series of tests. With the built-in position controller, we first test the trajectory tracking with only trajectory position command. Figure 5.4 shows the X-axis position and velocity tracking result. The blue line shows the odometry data actually used by the controller which is the simulated state measurement with measuring noise. The red line is the actual data from Gazebo, and the yellow line is the desired trajectory. In this case, we can see a significant tracking lag because only trajectory position has been tracked.

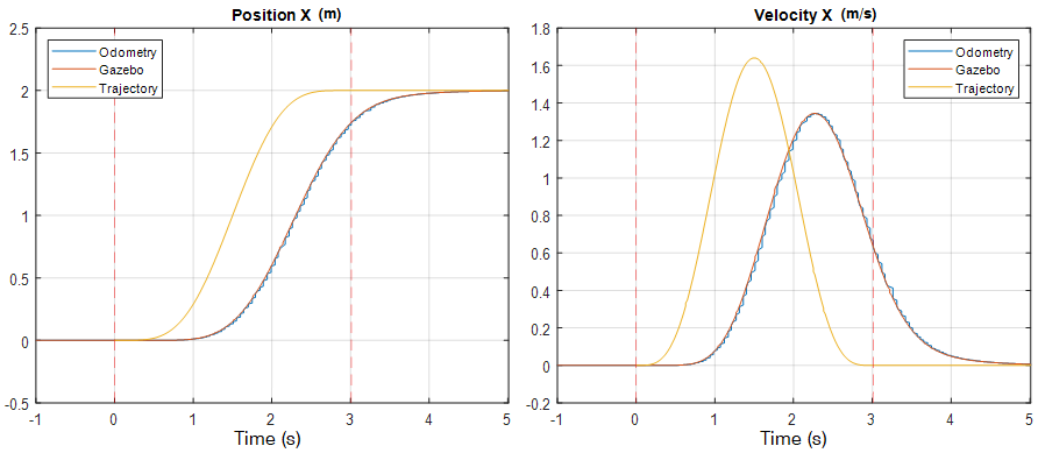

Figure 5.4. 1-D trajectory tracking with only position commands.

Next we add the trajectory velocity and acceleration commands to the controller and obtain the tracking result as shown in Figure 5.5. The trajectory tracking is much improved, but the lag in the beginning and the subsequent overshoot is obvious.

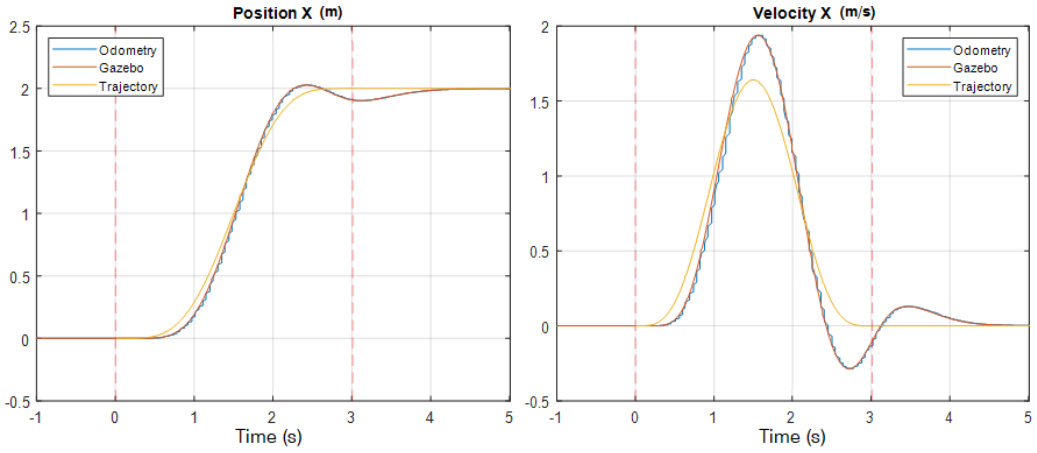

Figure 5.5. 1-D trajectory tracking with up to acceleration commands.

Next we add the trajectory jerk and snap commands, the inverse dynamics calculation proposed in Section 2.2.4, and the feedforward terms $\boldsymbol{\omega}_d$ and $\boldsymbol{I}(\hat{\boldsymbol{\omega}}\boldsymbol{R}^T\boldsymbol{R}_d\boldsymbol{\omega}_d - \boldsymbol{R}^T\boldsymbol{R}_d\dot{\boldsymbol{\omega}}_d)$

to the controller. From the tracking result shown in Figure 5.6, we can see that the trajectory tracking in the first half is already good, but in the second half the sustained velocity tracking error still cause position error.

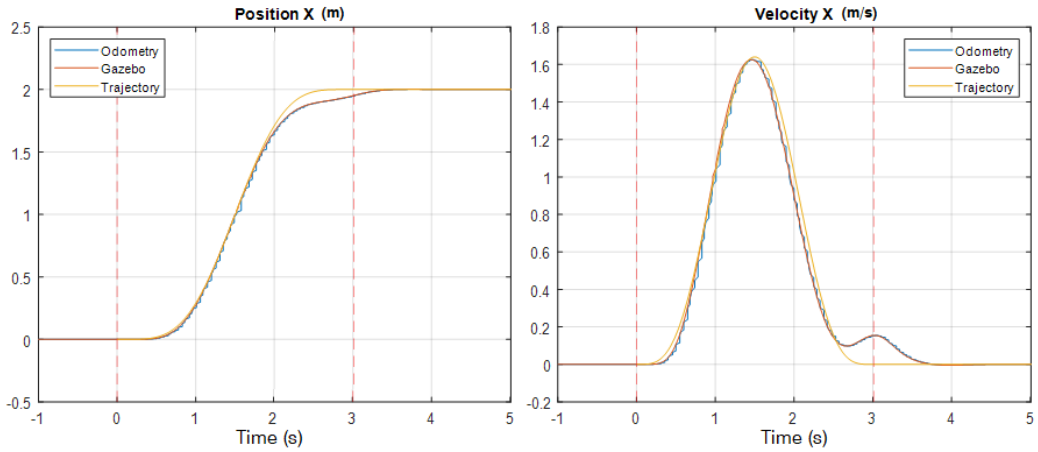

Figure 5.6. 1-D trajectory tracking with rotational feedforward terms.

The rotor drag is included in the RotorS simulation. Therefore, next we add the rotor drag compensation proposed in Chapter 3 to the controller. From the tracking result shown in Figure 5.7, we can see that the precise trajectory tracking is finally achieved. From this series of tests, we observe how the rotational feedforward terms and the full compensation for rotor drag plays their part to improve the trajectory tracking performance.

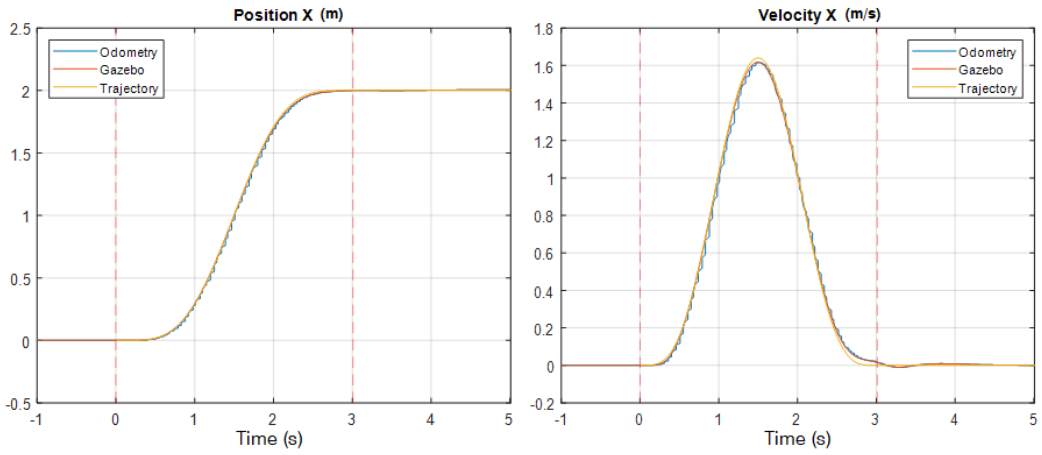

Figure 5.7. 1-D trajectory tracking with rotor drag compensation.

5.1.2 Window Passing Trajectory Tracking Test

With the precise trajectory tracking, we then test the narrow window passing trajectory. We use the same 2-window scenario settings as in Section 3.3.2 and also specify the maximum rotor thrust as 3.5 N. The optimized trajectory in this case is different from the one in Section 3.3.2 because the vehicle model parameters are different. Figure 5.8 shows the sequence stock image of the successful 2-window passing flight.

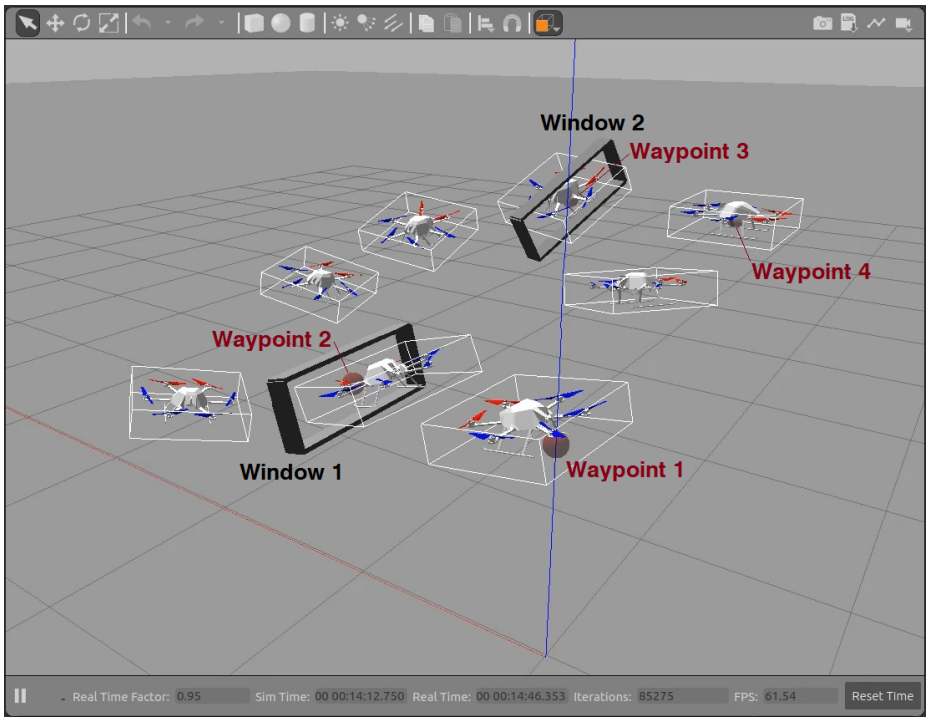

Figure 5.8. 2-window passing trajectory tracking in RotorS simulation.

Figure 5.9 and Figure 5.10 show the trajectory position and velocity tracking result in this simulation. From these 3-axis tracking comparisons, we can see that the trajectory tracking controller still works well with this complicated aggressive multi-segment trajectory. The hexacopter passed the two narrow windows at expected times with very small position and velocity tracking error.

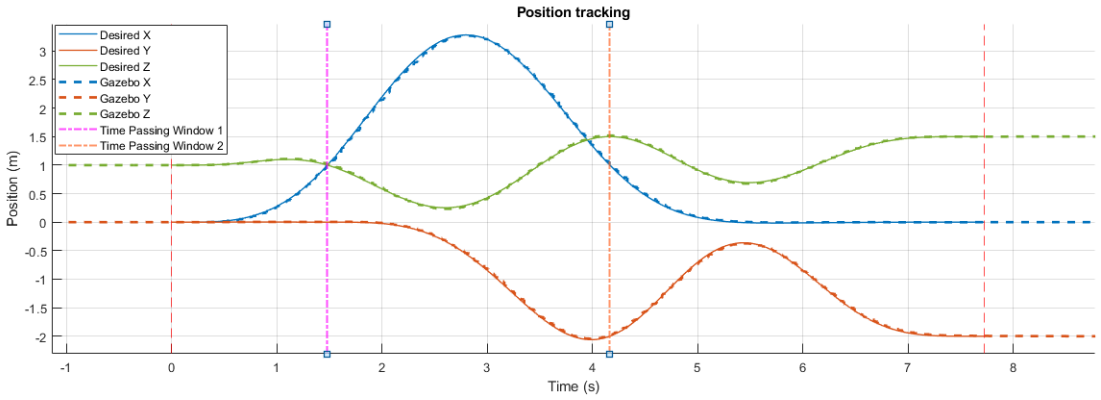

Figure 5.9. Position tracking in the 2-window passing flight.

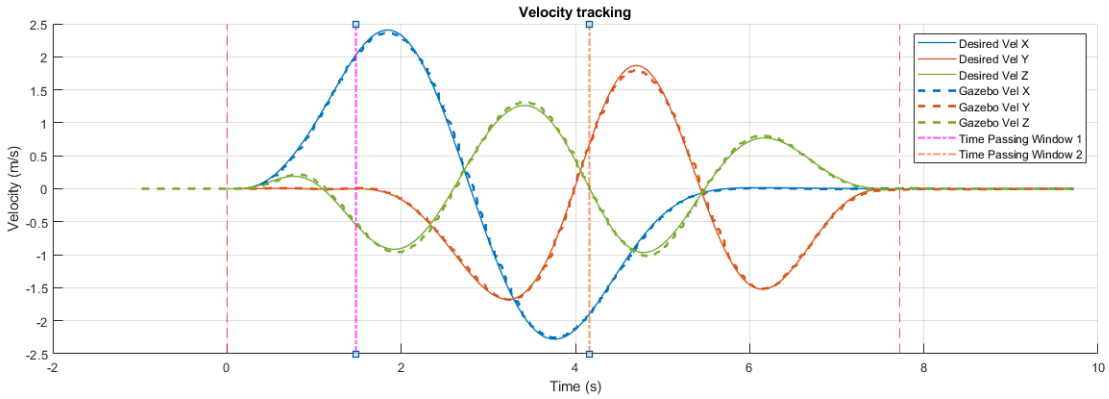

Figure 5.10. Velocity tracking in the 2-window passing flight.

Figure 5.11 shows the comparison of the actual vehicle attitude and the estimation from the inverse dynamics analysis in the trajectory planning. Note that the trajectory tracking controller does not track the estimated roll and pitch angle directly. The good match comes from the precise trajectory tracking and vehicle dynamic model estimating. From Figure 5.11 we can see that, at each window passing time, the vehicle attitude is as expected. Thus the successful narrow window passing is achieved.

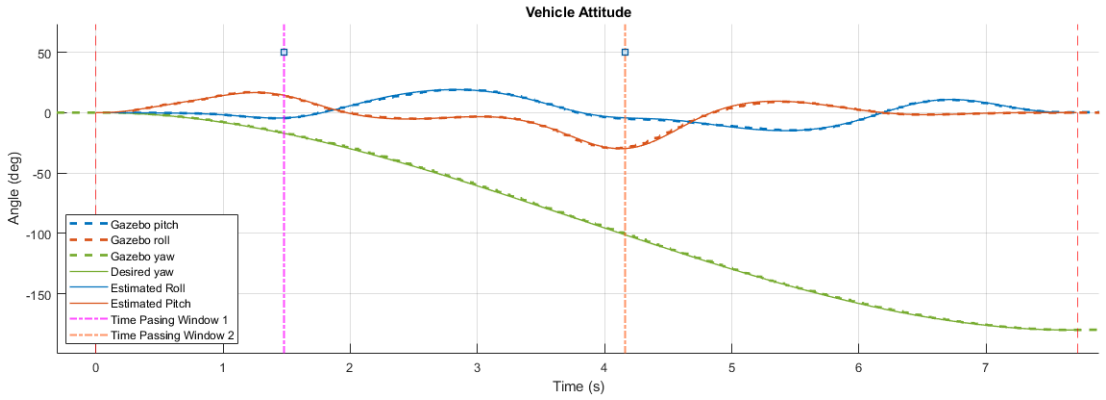

Figure 5.11. Vehicle attitude in the 2-window passing flight.

The video of this narrow window passing flight simulation is available at https://youtu.be/4g_2kYad4cA. The window passing would be failing if the rotor drag compensation is not applied. Another video showing this comparison is available at https://youtu.be/5-DZhND3Eq4.

5.2 PX4 SITL Simulation

It is important at this stage to have a high fidelity simulation environment to test the system integration, the aggressive polynomial trajectory implementation, the algorithms and controller code, and even the flight test procedures before the real flight. PX4 is the open source autopilot platform we use on the Pixhawk flight control hardware to implement the quadcopter control. One of the great features of PX4 is that it supports SITL (Software in the Loop) simulation. This allows PX4 flight code to control a computer modeled vehicle in a simulated "world". PX4 SITL supports several popular simulators such as Gazebo, FlightGear and JSBSim [59]. We choose Gazebo because it is widely used by researchers in this field and also highly recommended by PX4.

5.2.1 Simulation System Architecture

We set up this SITL simulation on Ubuntu 18.04 operating system and ROS Melodic robotic system environment. PX4 SITL runs the on-board autopilot program in this simulation environment, and Gazebo 9.19.0 is used as the 3D dynamic simulation and display engine. As in actual flight experiments, we use the QGroundControl as GCS software. One computer joystick is used to replace the RC transmitter to control the quadcopter manually during takeoff and landing.

We create a gazebo world file to represent the Vicon lab where we conduct actual flight tests. We make a detailed DJI F450 quadcopter model for visualization and collision detection. To simulate the rotors and sensors such as magnetometers, barometers, accelerometers, and gyros on the quadcopter, the simulation model is equipped with gazebo plug-ins including "libgazebo_motor_model.so", "libgazebo_magnetometer_plugin.so", "libgazebo_barometer_plugin.so", and "libgazebo_imu_plugin.so".

Figure 5.12 shows the Gazebo and GCS display during the actual SITL simulation. The "world" in the simulation represents the environment of the Vicon lab. During the simulation, the QGroundControl GCS software functions identically as it would do in the actual flight test.

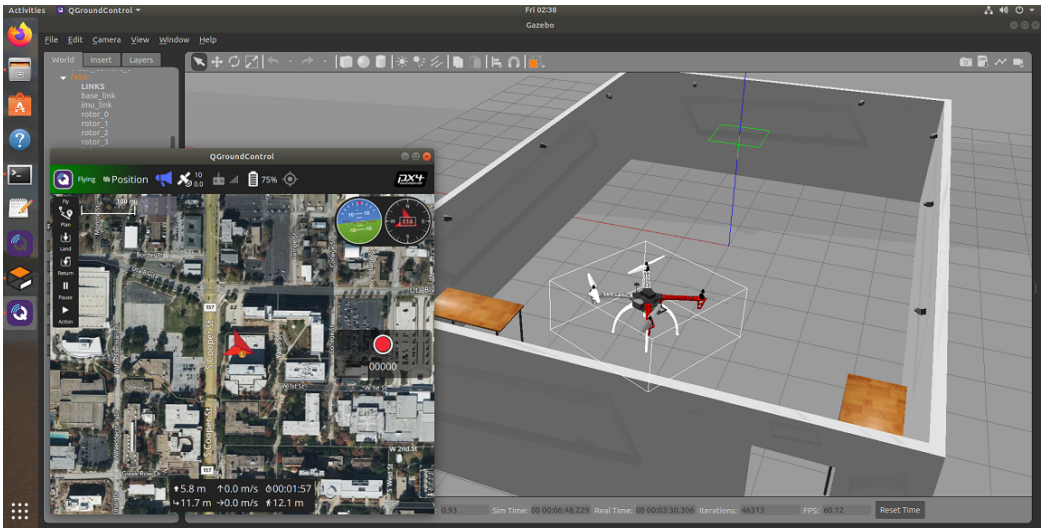

Figure 5.12. Gazebo and GCS display in SITL simulation.

The actual I/O connections for the autopilot program are replaced in this SITL simulation. The serial connection to GCS software through telemetry device is replaced by UDP connection with MAVlink communication. The I/O interface to on-board sensors is replaced by program API. The PWM output to motor ESC is also replace by program API. The serial connection to on-board computer is replaced by UDP connection with MAVROS (MAVlink based) communication. The PPM (or SBUS) control input from RC transmitter through RC radio receiver is replaced by joystick input through QGroundControl GCS software MAVlink communication.

We also run our flight control programs on the on-board computer here. The MAVROS communication to the autopilot program is established by executing the launch file "px4.launch" provided in the MAVROS package with local host port 14540 designated as "fcu_url". One additional program is created to simulate the Vicon system by obtaining the quadcopter position and attitude from Gazebo through Gazebo transport API and feeding the information to autopilot software through MAVROS interface. The overall SITL simulation system architecture and the data flow are illustrated in Figure 5.13.

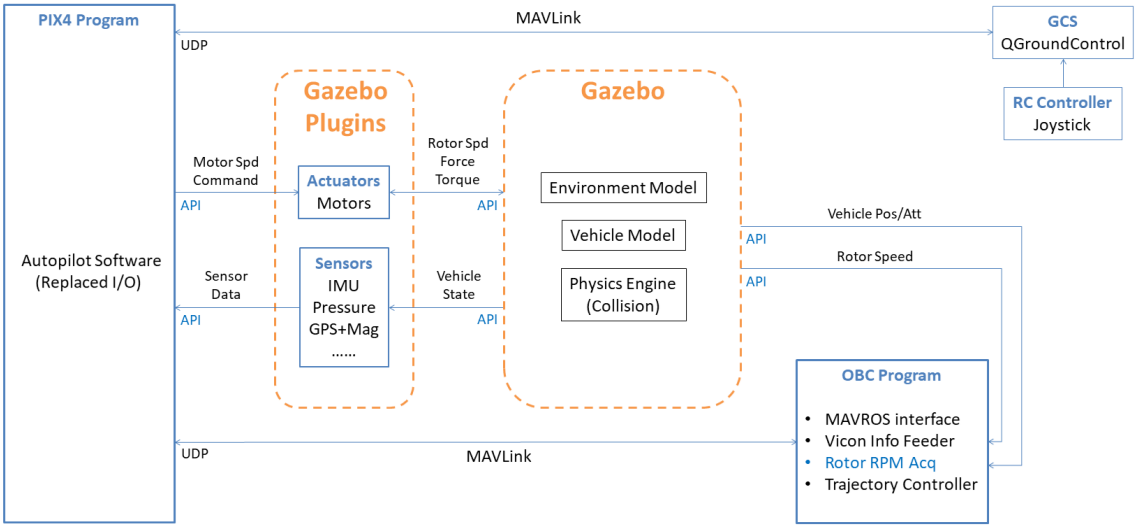

Figure 5.13. SITL simulation system architecture.

5.2.2 Flight Test Data Comparison

We apply the quadcopter parameters identified in Chapter 6 to the simulation model and test the simulation system with a simple 1-segment 1-D minimum snap trajectory that goes from $y = -1$ to $y = 1$ in 5 seconds. We conduct the tracking with the native PX4 controller (firmware version: 1.13.0) and compare the simulation result with actual flight test data. Figure 5.14 and Figure 5.15 show the sequence stock image of the trajectory tracking in actual flight and in SITL simulation. The quadcopter flies from right to left. From the sequence stock image comparison, we can already see the similarity between the flight test and the simulation.

Figure 5.14. 1-D minimum snap trajectory tracking in actual flight.

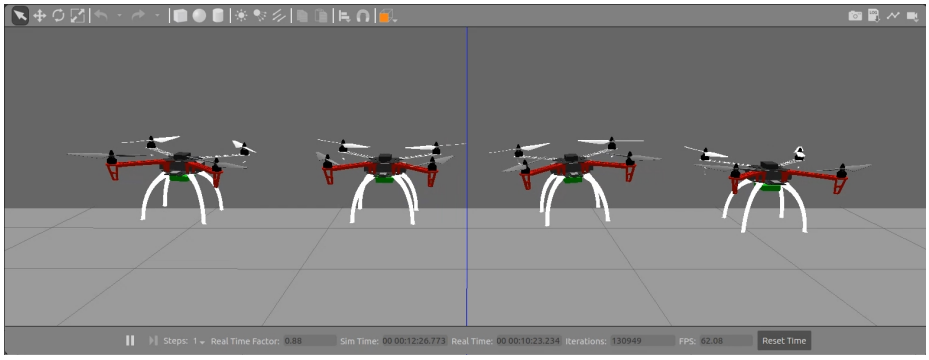

Figure 5.15. 1-D minimum snap trajectory tracking in SITL simulation.

We conduct the trajectory tracking by sending "PositionTarget" message to MAVROS "/mavros/setpoint_raw/local" topic at 50 Hz. In the first test, we only send the position tracking command. Figure 5.16 shows the comparison of desired trajectory, SITL simulation result, and actual flight test data. The tracking is not good with a significant lag because only position has been tracked, but the quadcopter response in flight test and simulation is quite similar.

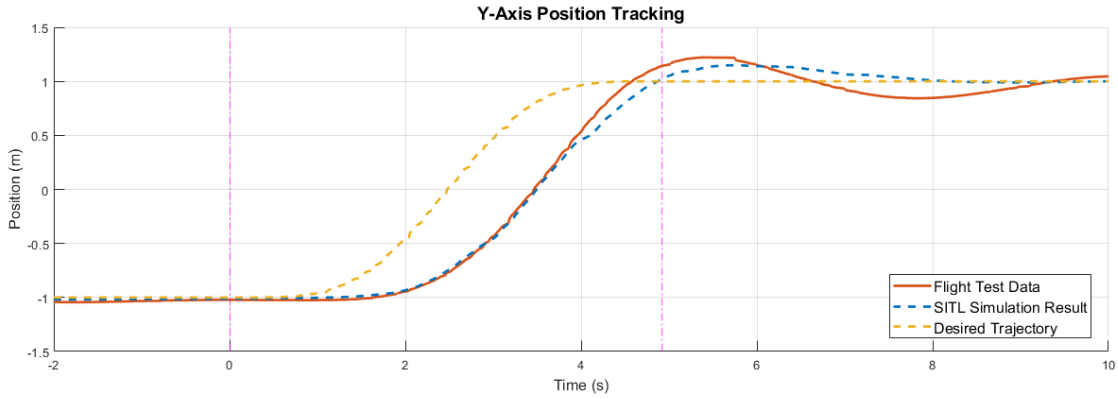

Figure 5.16. Trajectory tracking comparison with only position commands.

Next we send both position and velocity commands. From Figure 5.17 we can see that the tracking lag is improved because the trajectory velocity is also tracked, but the overshoot is increased because of the built-up velocity tracking error. However, the similarity in the quadcopter response in the flight test and in the simulation is still observed.

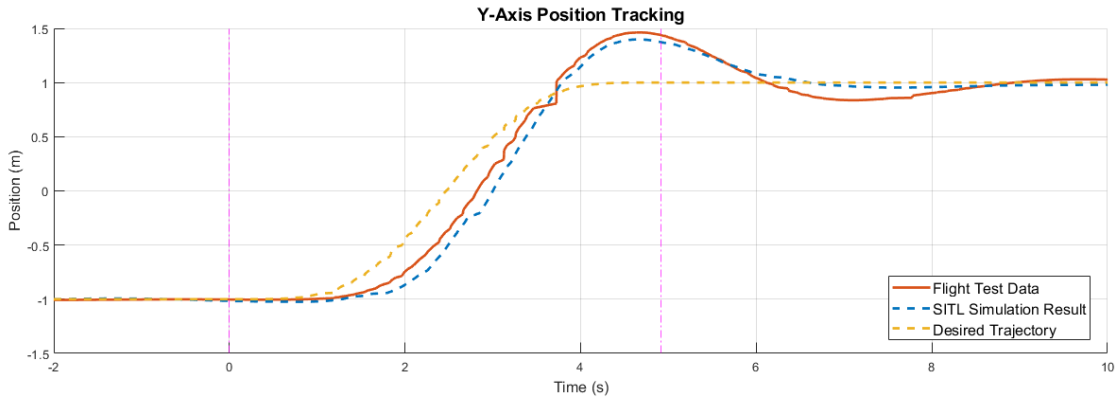

Figure 5.17. Trajectory tracking comparison with up to velocity commands.

Next we send position, velocity and acceleration commands. From Figure 5.18 we can see that the trajectory tracking is much improved because the desired attitude is now

also generated directly from the trajectory acceleration. In this case, we can still see that the quadcopter response in flight test is very close to that in the SITL simulation.

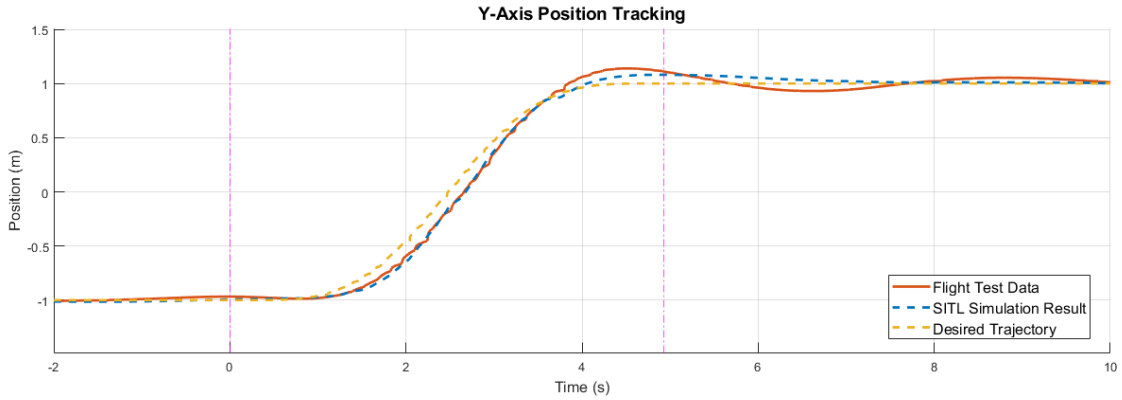

Figure 5.18. Trajectory tracking comparison with up to acceleration commands.

From this series of test we conclude that, although the trajectory tracking is not precise yet, the SITL simulation can already fairly reflect the actual quadcopter dynamic response in different control modes and effectively support our following quadcopter control development and flight test. Also, because of the entirety of this simulation environment, we can simulate the flight test with exactly the same test procedure and on-board/off-board programming.

Chapter 6

System Model Parameter Identification

Precise identification of system model parameters is crucial for both the trajectory optimization and the trajectory tracking control. Main parameters we need to identify in the quadcopter model are tabulated in Table 6.1 below.

Table 6.1. Main Parameters to be identified

Symbol	Meaning	Unit
m	Mass of the quadcopter	kg
L	Rotor arm length	m
h	Height of propeller plane above cg.	m
I_{xx}	Moment of inertia about x-axis	$\mathrm{kg} \cdot \mathrm{m}^2$
I_{yy}	Moment of inertia about y-axis	$\mathrm{kg} \cdot \mathrm{m}^2$
I_{zz}	Moment of inertia about z-axis	$\mathrm{kg} \cdot \mathrm{m}^2$
I_r	Moment of inertia of the rotor	$\mathrm{kg} \cdot \mathrm{m}^2$
k_M	Rotor counter-moment constant	m
k_ω	Thrust constant for rotor speed	$\mathrm{N/RPM}^2$
k_z	Thrust constant for vertical velocity	$\mathrm{N} \cdot \mathrm{s/m/RPM}$
k_h	Thrust constant for horizontal velocity	$\mathrm{N} \cdot \mathrm{s}^2/\mathrm{m}^2$
k_d	Rotor drag constant	$\mathrm{N} \cdot \mathrm{s/m/RPM}$
C_R	Rolling torque constant	$\mathrm{N} \cdot \mathrm{s/RPM}$

The mass and the rotor arm length are measured directly as $m = 1.412$ kg and $L = 0.2275$ m. Firstly we obtain precise mass properties of the vehicle through detailed SolidWorks CAD modeling. Then we find rotor properties through static rotor tests. Lastly, a series of flight test is conducted to identify the quadcopter dynamic model coefficients through the maximum likelihood estimation technique.

6.1 SolidWorks CAD modeling

When developing dynamic model of the quadrotor, fairly accurate estimation of moment of inertia becomes very important. To obtain accurate moment of inertia of the quadrotor, we utilized SolidWorks and developed CAD model. Firstly, the baseline model of the vehicle was created by using CAD components available online [60]. After the baseline model was developed, the actual quadrotor was disassembled into components to measure their individual masses. Then, the component masses in the CAD model were updated to the actual measurements. Dimensions as well as masses of additional components such as Pixhawk, Raspberry pi, LiPo battery, RPM sensor and etc. were also measured, and 3D models of these components were created and added into the baseline model assembly. This process was repeated as components are added or removed from the actual quadrotor. After the CAD model was developed, there was a slight total mass difference between the 3D model and the actual vehicle due to miscellaneous parts such as wiring, fastening and gluing which are hard to be 3D modeled. These additional masses were distributed equally to the components which have the majority of these elements. As a result, the following CAD model shown in Figure 6.1 was developed and finalized.

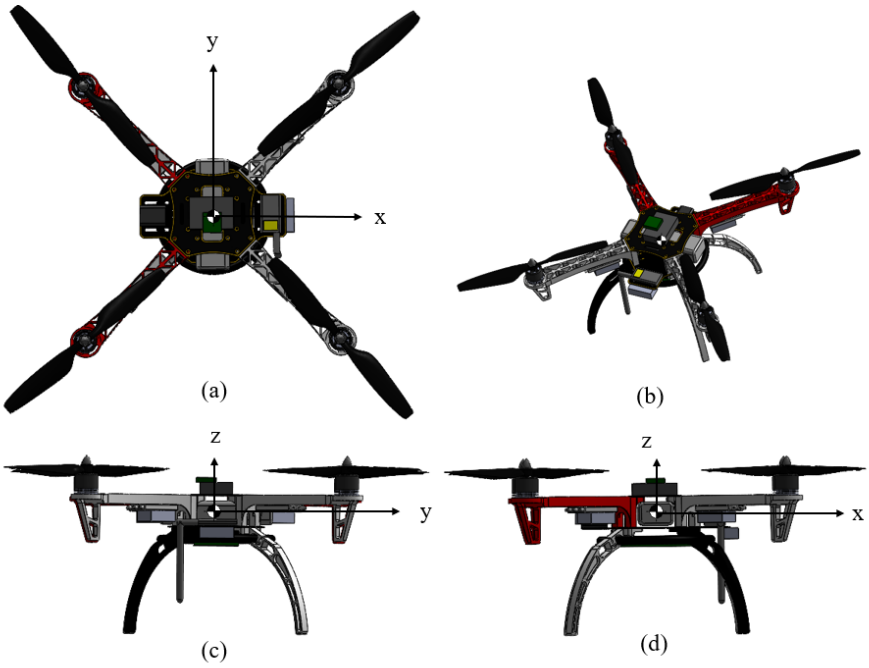

Figure 6.1. Finalized quadrotor CAD model; (a): Top view, (b): Isometric View, (c): Front view, (d): Side view.

By using mass property measuring feature in SolidWorks, the following inertia tensor was obtained.

$$\boldsymbol{I} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} = \begin{bmatrix} 0.013020313 & -0.000087965 & -0.000065616 \\ -0.000087965 & 0.012897730 & 0.000039814 \\ -0.000065616 & 0.000039814 & 0.02362329 \end{bmatrix} \text{kg} \cdot \text{m}^2 \tag{6.1}$$

From the CAD model assembly, location of the vehicle center of gravity was measured with respect to the center of the top plate (reference point) which is the location where the Pixhawk autopilot is attached. Figure 6.2 shows the CG location and 3-axis measurements.

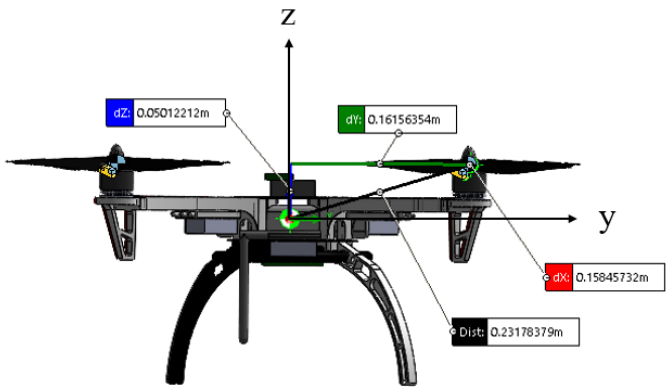

Figure 6.2. Vehicle CG location and measurements from the reference point.

The position of vehicle CG with respect to the reference point is found as follows. Note that the measurement was done in the body fixed reference frame.

$$\boldsymbol{r}_{cg,r} = \begin{bmatrix} 0.00064165 & -0.00246455 & -0.01828377 \end{bmatrix}^T \text{m} \tag{6.2}$$

To estimate the gyroscopic effect generated by the fast spinning rotor, we also need the estimation of the moment of inertia of the rotating parts about the spinning axis. Figure 6.3 shows the power unit disassembled into rotating and stationary parts.

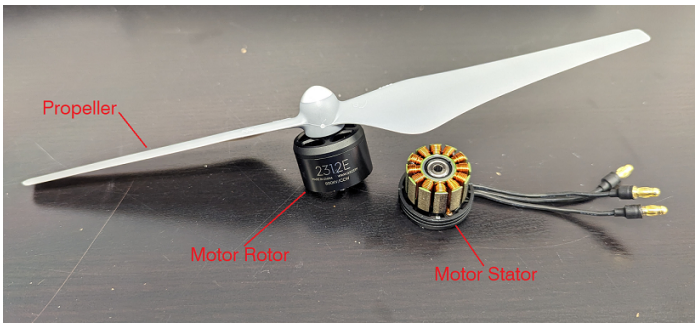

Figure 6.3. Power unit disassembly.

We create a CAD model assembly of the power unit rotating part which is composed of the propeller, the motor rotor, and the motor shaft as shown in Figure 6.4

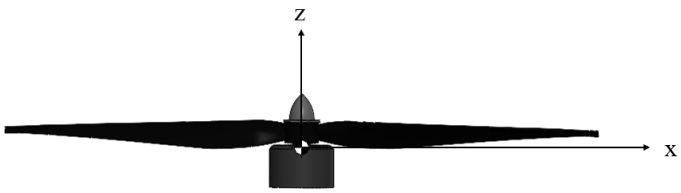

Figure 6.4. CAD model assembly of the power unit rotating part.

After carefully measuring and updating the mass of each component, the moment of inertia of the rotor about its spinning axis was estimated by utilizing SolidWorks "Mass Properties" feature as:

$$I_r = 0.00004958668 \text{ kg} \cdot \text{m}^2 \tag{6.3}$$

To estimate the rotor drag induced torque, we also need the geometry from vehicle CG to the propellers. Figure 6.5 shows the measurement in the CAD model, and the height of propeller plane above CG is estimated as:

$$h = 0.05012212 \text{ m} \tag{6.4}$$

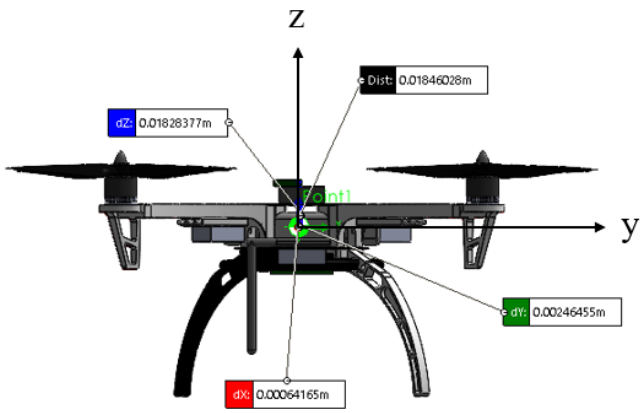

Figure 6.5. Measurement of the propeller from vehicle CG.

6.2 Static Rotor Test

To obtain the rotor properties, a rotor test bench shown in Figure 6.6 developed at the Aerospace Systems Laboratory at The University of Texas at Arlington was used [61]. The test bench consists of a motor mount with a load cell which measures the thrust produced by the propeller. Along with the load cell there are additional sensors to record various quantities like the battery voltage, the current drawn by the motor, infra-red temperature sensor to measure the temperature of the motor and an optical sensor to measure the RPM of the motor. These sensors are connected to an Arduino microcontroller which sends the data to MATLAB through serial communication.

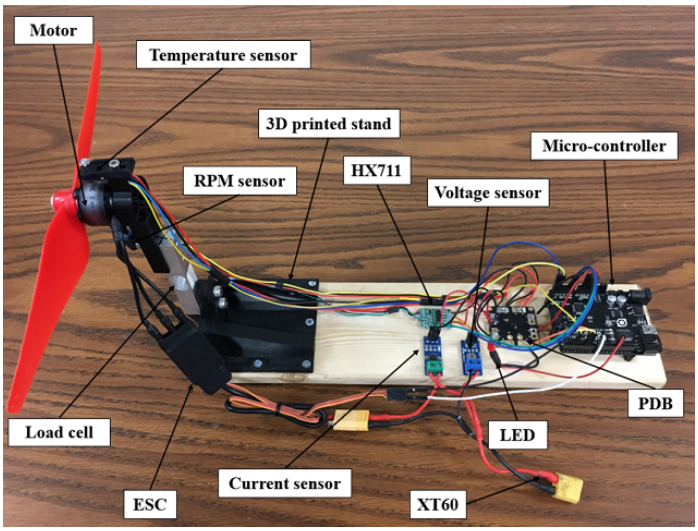

Figure 6.6. Rotor test bench.

The rotor system to be tested is combined with the DJI-430 Lite ESC, the DJI-2312E brushless motor and the DJI-9450 propeller. We test the rotor system with various power supply voltage within the actual flight test usage range from 10.5 V to 12.6 V with 0.3 V intervals. Under each voltage, we send various PPM commands from 1200 μs to 1900 μs with 5 μs intervals and collect measurement including thrust, RPM, current and etc. With the same PPM command, the rotor would reach different RPM and thrust under different power supply voltage. Figure 6.7 and Figure 6.8 show the thrust and RPM curve comparisons between 10.5 V and 12.6 V.

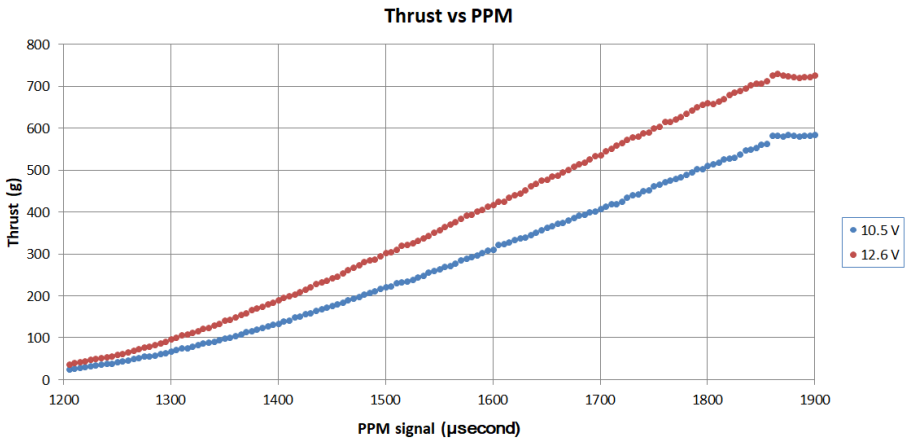

Figure 6.7. Thrust vs PPM test data plot.

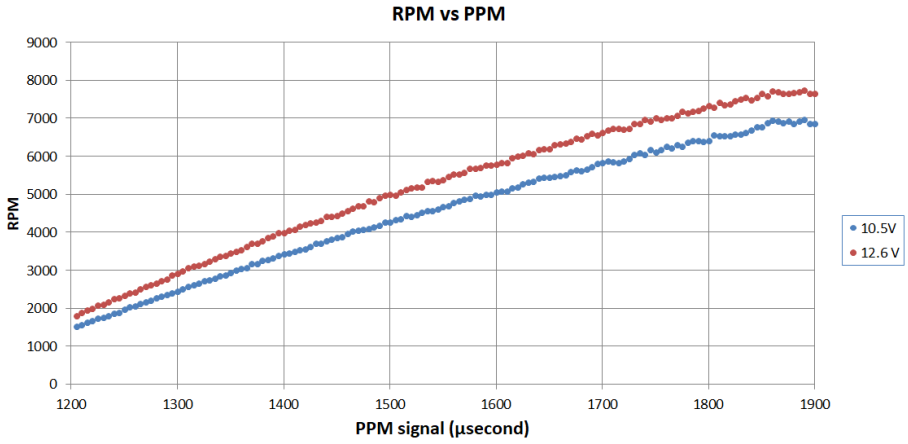

Figure 6.8. RPM vs PPM test data plot.

In the data analysis, first we combine all the thrust and RPM test data and identify the relationship between them. In Chapter 2 and Chapter 3, we assume that the thrust is proportional to the square of RPM. By making the thrust vs RPM^2 plot as shown in Figure 6.9, this assumption is verified to be very reasonable in our usage range. The coefficient k_f in Chapter 2, equivalent to k_ω in Chapter 3, is identified as 1.2×10^{-7} N/RPM^2.

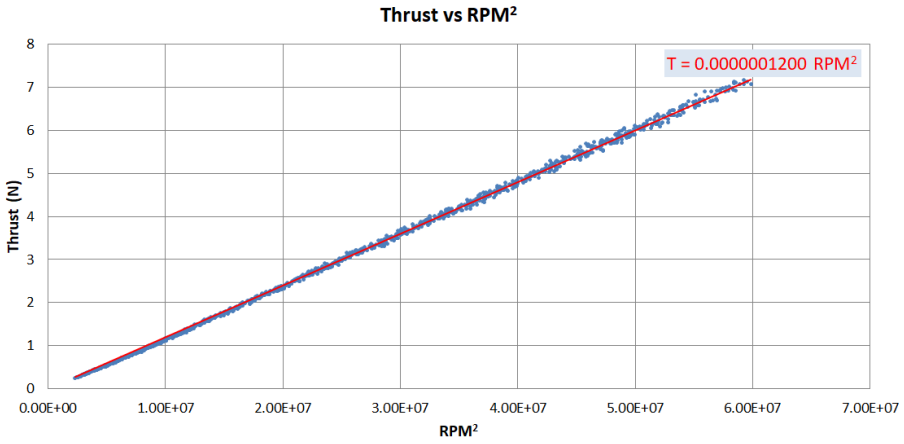

Figure 6.9. Thrust vs RPM^2 test data plot.

Then for precise rotor thrust control, we identify the thrust as a function of the battery voltage and the PPM signal by a 3rd order surface fitting as shown in Figure 6.10 in the form:

$$T(s,V) = p_{T00} + p_{T10}s + p_{T01}V + p_{T20}s^2 + p_{T11}sV + p_{T02}V^2 + p_{T30}s^3 + p_{T21}s^2V + p_{T12}sV^2 \tag{6.5}$$

where T is the thrust measured in grams, s is the PPM signal in μs, and V is the battery voltage in volts. The coefficients are tabulated in Table 6.2.

Table 6.2. Coefficients for the thrust function

Coefficient	Value	Coefficient	Value
p_{T00}	6771	p_{T02}	-1.075
p_{T10}	-11.65	p_{T30}	-1.176$\times 10^{-6}$
p_{T01}	-242.2	p_{T21}	-6.111$\times 10^{-5}$
p_{T20}	0.006503	p_{T12}	0.00152
p_{T11}	0.2608		

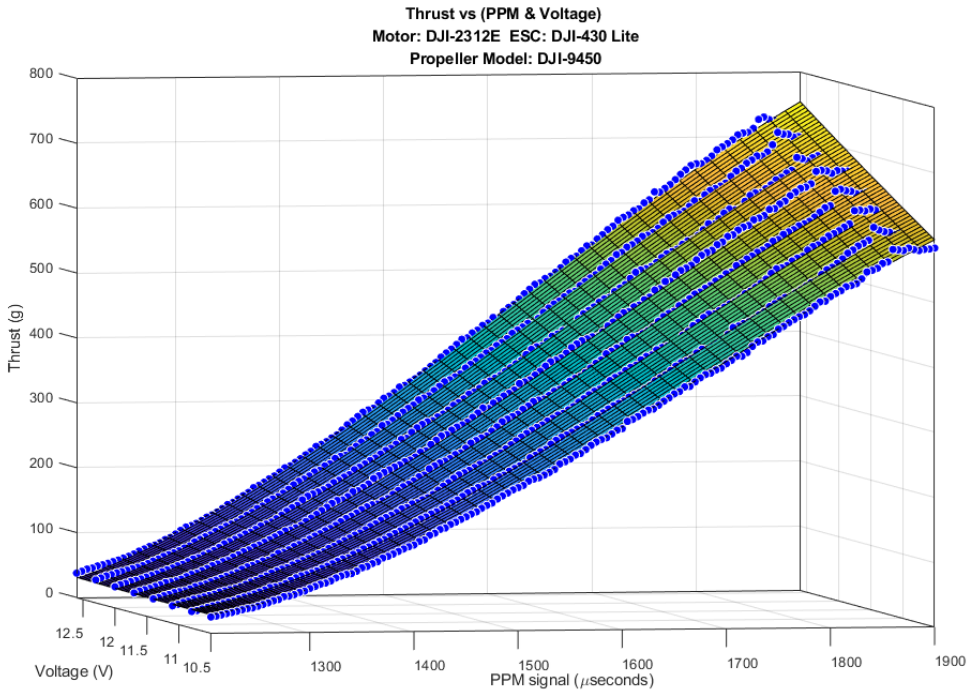

Figure 6.10. Thrust vs PPM and voltage surface fitting.

Lastly for rotor RPM estimation in the on-board computer trajectory controller, we also identify the RPM as a function of the battery voltage and the PPM signal by a 3rd order surface fitting as shown in Figure 6.11 in the form:

$$\Omega(s,V) = p_{R00} + p_{R10}s + p_{R01}V + p_{R20}s^2 + p_{R11}sV + p_{R02}V^2 + p_{R30}s^3 + p_{R21}s^2V + p_{R12}sV^2 \tag{6.6}$$

where Ω is the rotor RPM, s is the PPM signal in μs, and V is the battery voltage in volts. The coefficients are tabulated in Table 6.3.

Table 6.3. Coefficients for the RPM function

Coefficient	Value	Coefficient	Value
p_{R00}	1.486×10^{4}	p_{R02}	52.65
p_{R10}	-20.31	p_{R30}	-1.034×10^{-6}
p_{R01}	-3150	p_{R21}	-0.0007157
p_{R20}	0.008793	p_{R12}	-0.03583
p_{R11}	3.408		

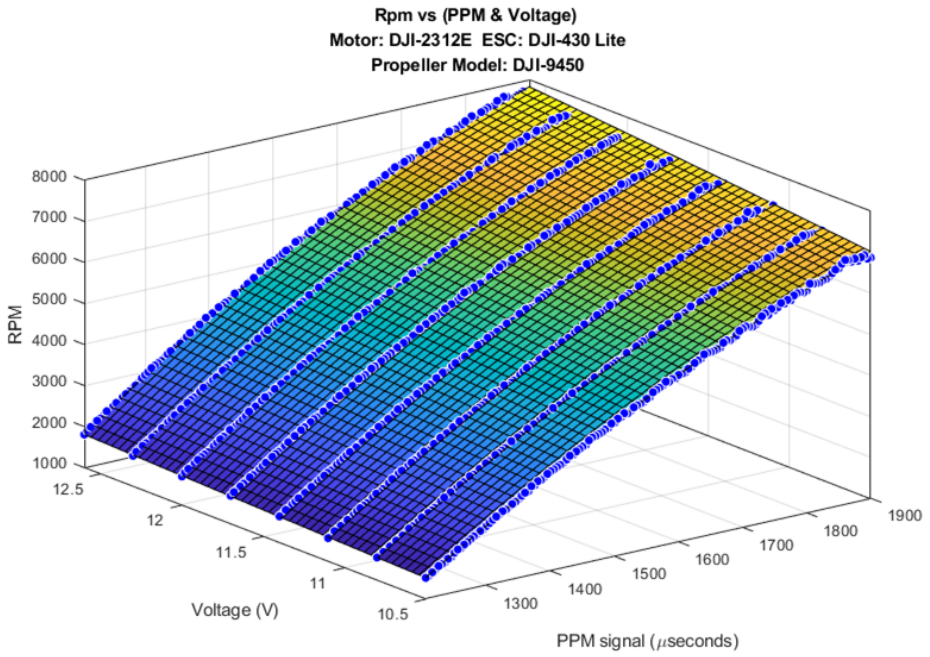

Figure 6.11. RPM vs PPM and voltage surface fitting.

6.3 Flight Experiments

With those parameters already estimated, next we try to identify the rest dynamic model parameters through flight experiments and data analysis technics such as maximum likelihood estimation.

6.3.1 Experimental Design

The goal of these flight experiments is to identify the rest model parameters as tabulated in Table 6.4. k_ω, k_z and k_h are used in the rotor thrust estimation as in equation (3.7). k_d is used in the rotor drag estimation as in equation (3.5). C_R is used in the rotor rolling effect estimation as in equation (3.11). And k_M is used in rotor counter-moment estimation as in equation (3.3).

Table 6.4. Parameters to be identified in flight experiments

Symbol	Meaning	Unit
k_M	Rotor counter-moment constant	m
k_ω	Thrust constant for rotor speed	$\mathrm{N/RPM^2}$
k_z	Thrust constant for vertical velocity	$\mathrm{N \cdot s/m/RPM}$
k_h	Thrust constant for horizontal velocity	$\mathrm{N \cdot s^2/m^2}$
k_d	Rotor drag constant	$\mathrm{N \cdot s/m/RPM}$
C_R	Rolling torque constant	$\mathrm{N \cdot s/RPM}$

The main idea in the experimental design is to start from identifying the minimal number of parameters through simple flight motion to eliminate the influence from other dynamic coupling. The Ideal case is to perform one simple flight motion in which only one dynamic equation is involved and only one parameter in the dynamic equation is unknown and to be identified. When some parameters are isolated and identified through simple experiments, we then conduct more complex experiments to identify other parameters in more involved dynamic equations.

First we conduct fixed position hovering test. The objective is to identify the thrust constant for rotor speed k_ω. Although k_ω has already been identified previously in the static rotor test, we still need to verify its value again in actual flight because the situation is different when the rotor is installed on the quadcopter than on the test bench.

Then we conduct climbing and descending test. The objective in this experiment is to identify the thrust constant for vertical velocity k_z in different Z-axis only vertical motions given known k_ω.

Next we conduct fixed roll angle level flight test. In different horizontal motions, we identify the rotor drag constant k_d mainly in Y-axis response, identify the rolling torque constant C_R mainly in pitching response, and identify the thrust constant for horizontal velocity k_h mainly in Z-axis response given known k_ω and k_z.

Lastly we conduct fixed position turning test. The objective in this experiment is to identify the rotor counter-moment constant k_M in yawing response given known k_d.

6.3.2 Maximum Likelihood Estimation

Maximum likelihood yields estimates for the unknown quantities which maximize the probability of obtaining the observed set of data. The desirable attributes of this technique, such as asymptotically unbiased and consistent estimates, are especially useful for the estimation of aircraft coefficients in the presence of measurement errors associated with flight data. The maximum likelihood estimation approach introduced in [62] is used here to estimate the unknown dynamic model parameters. Assume that $\mathbf{p}$ is the $q \times 1$ vector of quadcopter coefficients to be determined, $\mathbf{y}$ is the $m \times 1$ measurement vector, and R is the measurement error covariance. The goal is to find the best estimation to minimize the loss function:

$$J(\hat{\mathbf{p}}) = \frac{1}{2}\sum_{k=1}^{N}(\tilde{\mathbf{y}}_k - \hat{\mathbf{y}}_k)^T R^{-1}(\tilde{\mathbf{y}}_k - \hat{\mathbf{y}}_k) \tag{6.7}$$

where $\tilde{\mathbf{y}}_k$ is the actual measurement at time t_k, $\hat{\mathbf{y}}_k$ is the estimated response of $\mathbf{y}$ at time t_k for a given value of the unknown parameter vector $\mathbf{p}$, and N is the total number of measurements.

Newton-Raphson method is used to update the estimation of $\mathbf{p}$ iteratively. Assume that i is the iteration number, then the $i+1$ estimate of $\mathbf{p}$, denoted by $\hat{\mathbf{p}}$, is obtained from the ith estimate as [63]:

$$\hat{\mathbf{p}}_{i+1} = \hat{\mathbf{p}}_i - [\nabla^2_{\hat{\mathbf{p}}} J(\hat{\mathbf{p}})]^{-1} [\nabla_{\hat{\mathbf{p}}} J(\hat{\mathbf{p}})] \tag{6.8}$$

where the first and second gradients are defined as:

$$[\nabla_{\hat{\mathbf{p}}} J(\hat{\mathbf{p}})] = -\sum_{k=1}^{N} [\nabla_{\hat{\mathbf{p}}} \hat{\mathbf{y}}_k]^T R^{-1} (\tilde{\mathbf{y}}_k - \hat{\mathbf{y}}_k) \tag{6.9}$$

$$[\nabla^2_{\hat{\mathbf{p}}} J(\hat{\mathbf{p}})] = \sum_{k=1}^{N} [\nabla_{\hat{\mathbf{p}}} \hat{\mathbf{y}}_k]^T R^{-1} [\nabla_{\hat{\mathbf{p}}} \hat{\mathbf{y}}_k] - \sum_{k=1}^{N} [\nabla^2_{\hat{\mathbf{p}}} \hat{\mathbf{y}}_k]^T R^{-1} (\tilde{\mathbf{y}}_k - \hat{\mathbf{y}}_k) \tag{6.10}$$

The Gauss-Newton approximation to the second gradient is given by:

$$[\nabla^2_{\hat{\mathbf{p}}} J(\hat{\mathbf{p}})] \approx \sum_{k=1}^{N} [\nabla_{\hat{\mathbf{p}}} \hat{\mathbf{y}}_k]^T R^{-1} [\nabla_{\hat{\mathbf{p}}} \hat{\mathbf{y}}_k] \tag{6.11}$$

where the estimation gradient is defined as:

$$[\nabla_{\hat{\mathbf{p}}} \hat{\mathbf{y}}] = \begin{bmatrix} \frac{\partial \hat{\mathrm{y}}_1}{\partial \hat{\mathrm{p}}_1} & \frac{\partial \hat{\mathrm{y}}_1}{\partial \hat{\mathrm{p}}_2} & \cdots & \frac{\partial \hat{\mathrm{y}}_1}{\partial \hat{\mathrm{p}}_q} \\ \frac{\partial \hat{\mathrm{y}}_2}{\partial \hat{\mathrm{p}}_1} & \ddots & \cdots & \frac{\partial \hat{\mathrm{y}}_2}{\partial \hat{\mathrm{p}}_q} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \hat{\mathrm{y}}_m}{\partial \hat{\mathrm{p}}_1} & \frac{\partial \hat{\mathrm{y}}_m}{\partial \hat{\mathrm{p}}_2} & \cdots & \frac{\partial \hat{\mathrm{y}}_m}{\partial \hat{\mathrm{p}}_q} \end{bmatrix} \tag{6.12}$$

And the partial derivatives used in the Gauss-Newton algorithms are computed using the small perturbation method and a simple first-order numerical derivative, for example:

$$\frac{\partial \hat{\mathrm{y}}_1}{\partial \hat{\mathrm{p}}_1} = \frac{\hat{\mathrm{y}}_1|_{\hat{\mathrm{p}}_1 + \delta \mathrm{p}_1} - \hat{\mathrm{y}}_1|_{\hat{\mathrm{p}}_1}}{\delta \mathrm{p}_1} \tag{6.13}$$

6.3.3 Fixed Position Hovering Test

In this test, we simply send fixed position command (0, 0, 1) to Pixhawk in "Off-board" mode through MAVROS "/mavros/setpoint_raw/local" topic via "PositionTarget" message and collect the flight data during stable hovering. The position and RPM data in this 2-minute test are shown in 6.12 and 6.13. The position holding is good, but a certain degree of air disturbance is noticed during the flight test and caused the small fluctuation in rotor RPMs. From the RPM data, we also noticed that there is a constant speed difference between rotors 1, 2 and rotors 3, 4. From the rotor configuration shown in Figure 6.14, we can see that the constant rotor speed difference is between the counter-clockwise rotating rotors and the clockwise rotating rotors. This indicates a constant yawing control effort, and we will address this phenomenon later in Section 6.3.6.

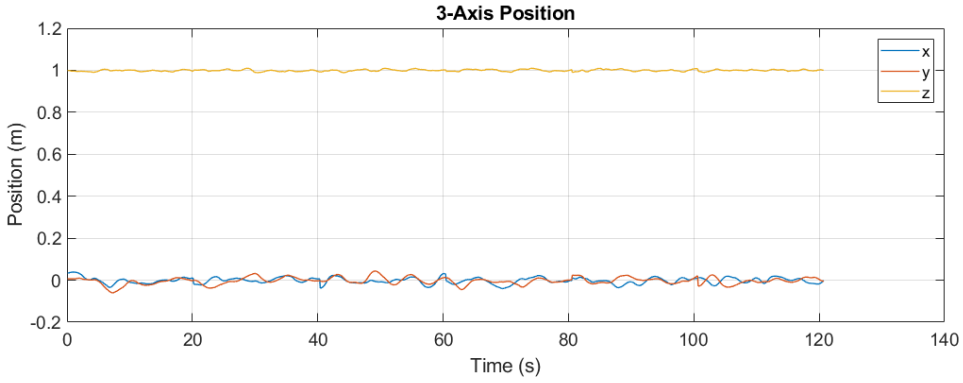

Figure 6.12. Position data in hovering test.

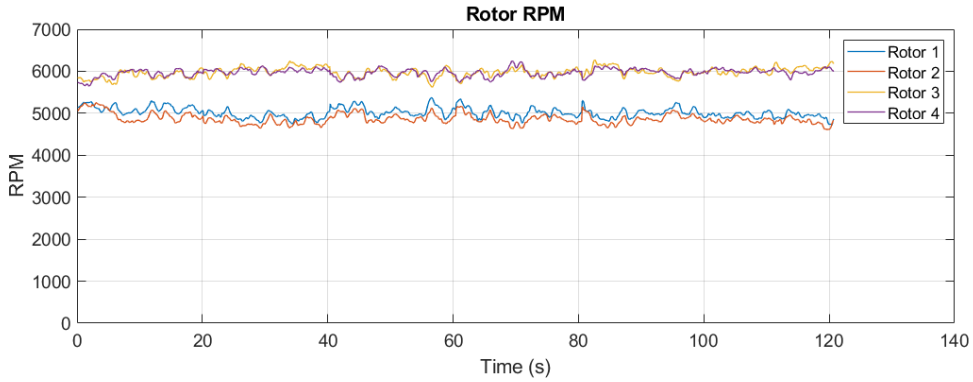

Figure 6.13. RPM data in hovering test.

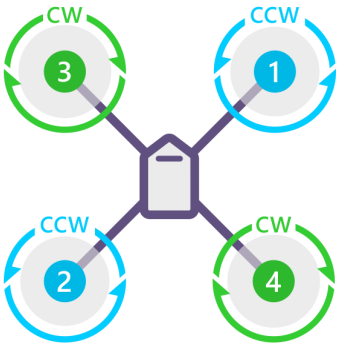

Figure 6.14. Rotor configuration of the quadcopter.

Because there is no vertical or horizontal velocity, we simplify equation (3.7) and calculated the thrust constant for rotor speed k_ω using the RPM data as:

$$k_\omega = \frac{mg}{\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2} \tag{6.14}$$

From the calculation result shown in Figure 6.15, we can observe the **small fluctuation on k_ω of around ±3% in magnitude caused by air disturbance**. And the thrust constant for rotor speed k_ω is identified as the average value:

$$k_\omega = 1.1513 \times 10^{-7}\ \mathrm{N/RPM^2} \tag{6.15}$$

Compare this value to the result from the static rotor bench test as shown in the cyan line in Figure 6.15, this equates to a **4.1% thrust loss**. This is reasonable because, like the "download penalty" in a helicopter, the structure situated in the downwash field below the rotor suffers an additional vertical drag, and thereby the net thrust is reduced.

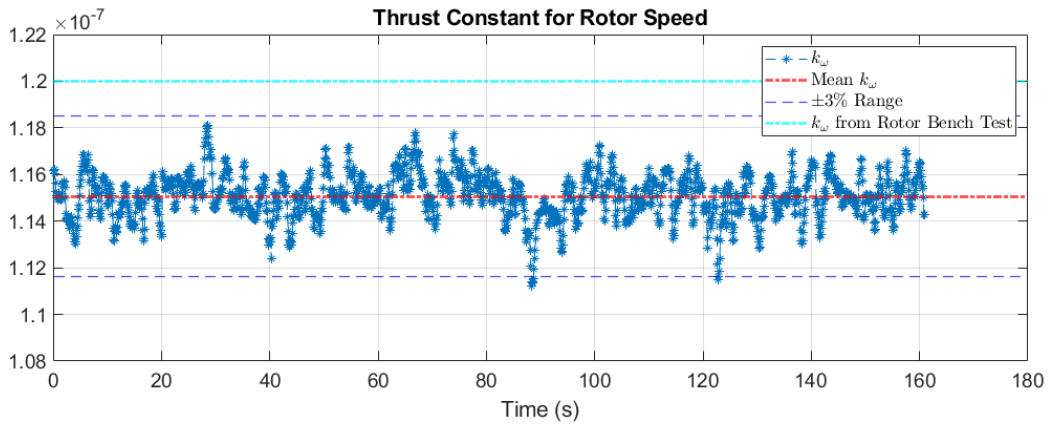

Figure 6.15. k_ω calculation result from hovering test data.

6.3.4 Climbing and Descending Test

In this test, we perform vehicle climbing and descending by sending (0, 0, 1) and (0, 0, 2) position commands back and forth to Pixhawk. The Z-axis response and RPM data in one test are shown in Figure 6.16 and Figure 6.17.

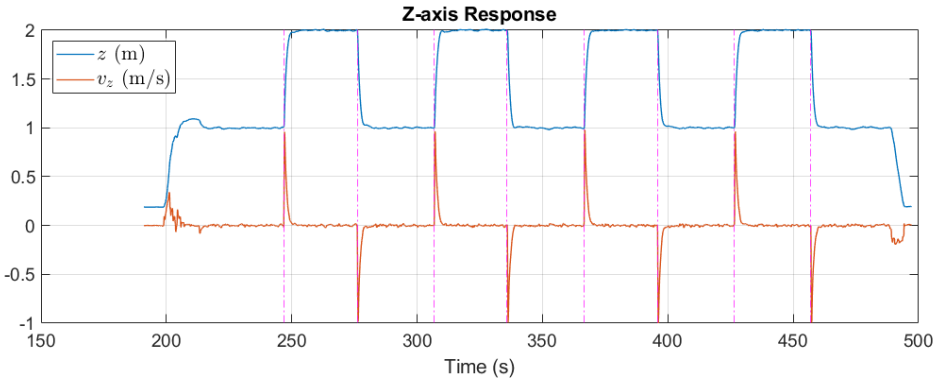

Figure 6.16. Z-axis response in climbing and descending test.

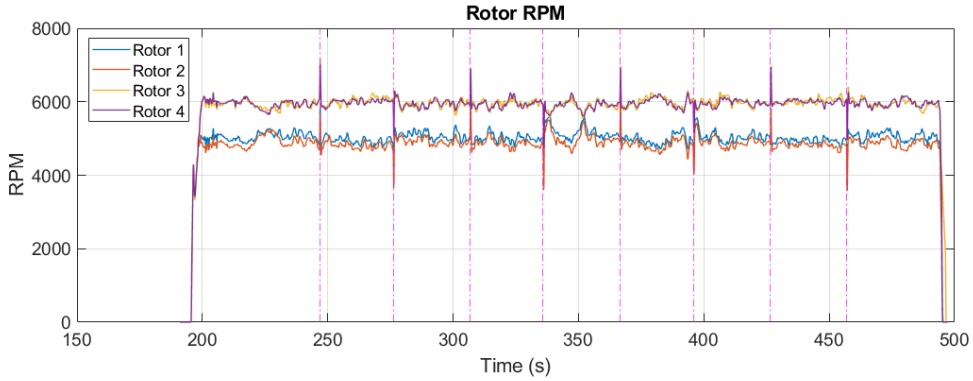

Figure 6.17. RPM data in climbing and descending test.

For maximum likelihood estimation, we use Z-axis position z and velocity $\dot{z}$ as measurement, and the thrust constant for vertical velocity k_z is the main parameter to be identified. Additionally, knowing that k_ω fluctuates during the flight due to air disturbance, we also identify k_ω in each relatively short test section and use previously identified average value as the initial guess. The identification and measurement vectors are:

$$\mathbf{p} = \begin{bmatrix} k_z & k_\omega \end{bmatrix}, \ \mathbf{y} = \begin{bmatrix} z & \dot{z} \end{bmatrix}^T \tag{6.16}$$

The forces acting on the quadcopter in this Z-axis motion are illustrated in Figure 6.18. We use the measured states as system states, and the equations of motion for the states are:

$$\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y}, k_z, k_\omega, \boldsymbol{\omega}_k) \tag{6.17}$$

where $\boldsymbol{\omega}_k$ is the RPM measurement $[\omega_1 \;\; \omega_2 \;\; \omega_3 \;\; \omega_4]$ at time t_k.

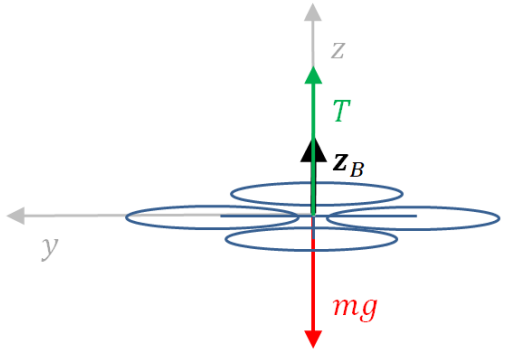

Figure 6.18. Quadcopter force diagram in climbing and descending test.

Without horizontal velocity, the individual rotor thrust is estimated as:

$$T_i = k_\omega \omega_i^2 - k_z v_z \omega_i \tag{6.18}$$

The final ODE is formed below and the 4th-order Runge-Kutta method is used for the integration with initial condition $\hat{\mathbf{y}}_0 = \tilde{\mathbf{y}}_0 = [\tilde{z}_0 \;\; \dot{\tilde{z}}_0]^T$.

$$\dot{\mathbf{y}} = \begin{bmatrix} \dot{z} \\ \ddot{z} = \frac{T - mg}{m} \end{bmatrix} \tag{6.19}$$

We take a 2.5-second section of the flight data for analysis in each climbing and descending action. Figure 6.19 and Figure 6.20 show example test data and estimation results for climbing and descending tests. In both cases, k_z converged very fast, and the final

estimation of the states fitted the measurement well. Comparing to the state estimations without k_z as shown in blue dashed lines, we can clearly see the effect of this parameter on the system response.

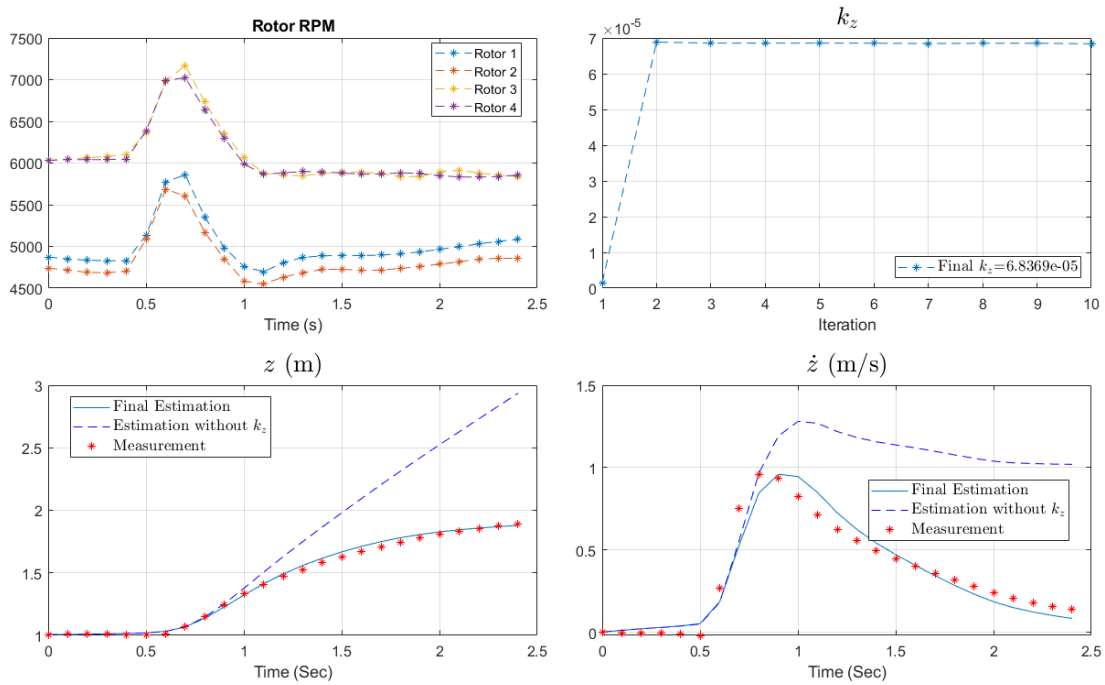

Figure 6.19. Climbing test data and estimation result.

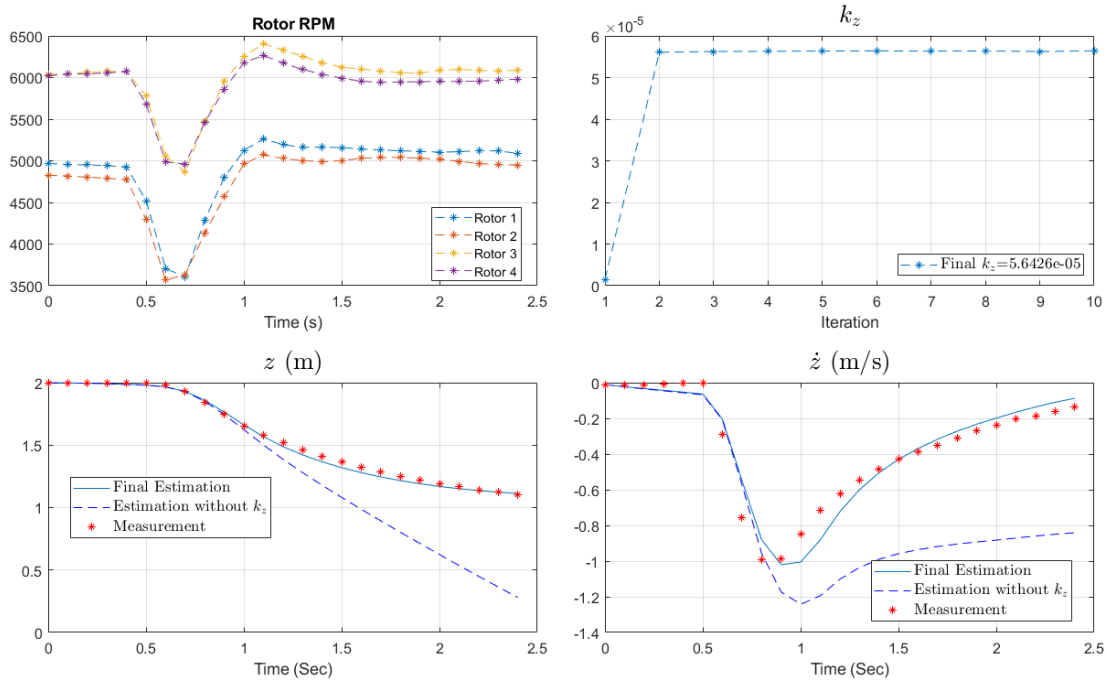

Figure 6.20. Descending test data and estimation result.

The analysis results for 10 individual tests are summarized in Table 6.5. The identified k_ω values for the short test sections are within a reasonable range as we observed in the previous hover test. And the thrust constant for vertical velocity k_z is identified as the average value:

$$k_z = 6.2062 \times 10^{-5}\ \mathrm{N \cdot s/m/RPM^2} \tag{6.20}$$

6.3.5 Fixed Roll Angle Level Flight Test

In this test, we send fixed roll angle command to Pixhawk in "Offboard" mode through MAVROS "/mavros/setpoint_raw/attitude" topic via "AttitudeTarget" message. In order to keep the altitude, the "thrust" command (normalized value between 0 and 1) in

Table 6.5. Estimation results in climbing and descending tests

Test No.	Action	k_ω	k_ω Diff (%)	k_z
1	Climb	1.1759×10^{-7}	2.1	6.8369×10^{-5}
2	Descend	1.1103×10^{-7}	-3.6	5.3987×10^{-5}
3	Climb	1.1682×10^{-7}	1.5	6.1010×10^{-5}
4	Descend	1.1298×10^{-7}	-1.9	6.5358×10^{-5}
5	Climb	1.1659×10^{-7}	1.3	6.3535×10^{-5}
6	Descend	1.1324×10^{-7}	-1.6	5.6426×10^{-5}
7	Climb	1.1467×10^{-7}	-0.4	5.9170×10^{-5}
8	Descend	1.1217×10^{-7}	-2.6	5.6462×10^{-5}
9	Climb	1.1806×10^{-7}	2.5	6.8423×10^{-5}
10	Descend	1.1190×10^{-7}	-2.8	6.7876×10^{-5}

the AttitudeTarget message also has to be specified. We obtain 20-second average value of the "thrust" setpoint from "/mavros/setpoint_raw/target_attitude" topic during fixed position hovering before every test. Using this hover average value as a baseline T_0, we adjust the thrust command $\mathrm{T}_{\mathrm{cmd}}$ during the fixed roll flight test according to the actual roll angle ϕ as:

$$\mathrm{T}_{\mathrm{cmd}} = \frac{\mathrm{T}_0}{\cos\phi} \tag{6.21}$$

The system response and RPM data in one -4-degree test are shown in Figure 6.21 and Figure 6.22.

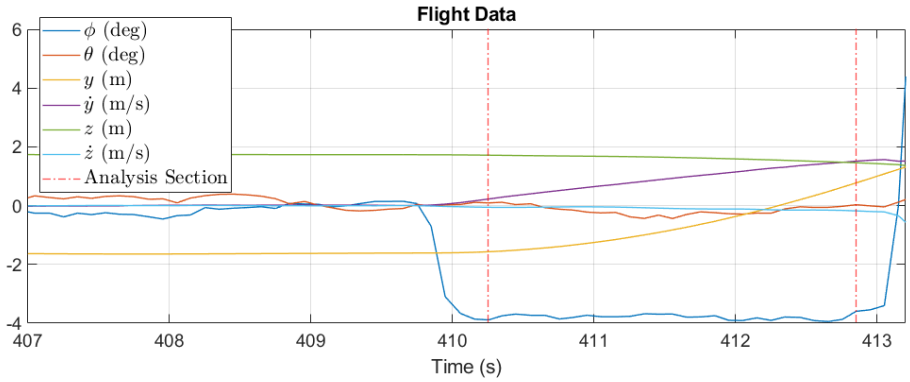

Figure 6.21. Flight data in -4 degree roll test.

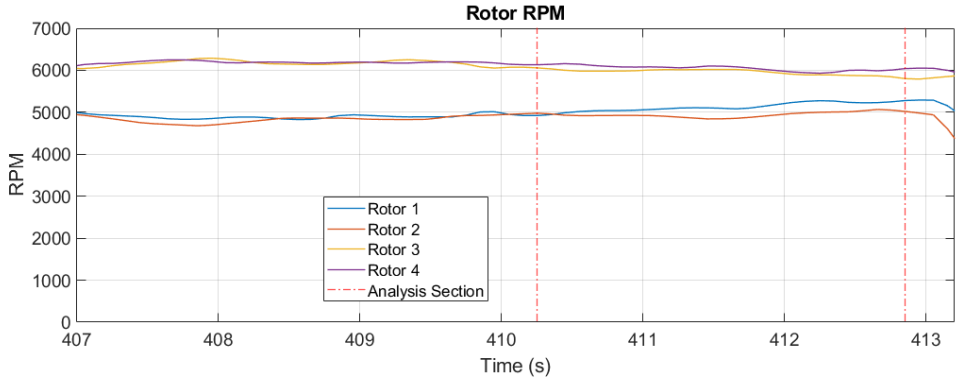

Figure 6.22. RPM data in -4 degree roll test.

For maximum likelihood estimation, we use Y-axis position y and velocity $\dot{y}$, Z-axis position z and velocity $\dot{z}$, and pitch angle θ and rate $\dot{\theta}$ as measurement, and the rotor drag constant k_d, the thrust constant for horizontal velocity k_h, and the rolling torque constant C_R are the main parameters to be identified. As in last test, we also identify k_ω in each relatively short test section and use previously identified average value as the initial guess. The identification and measurement vectors are:

$$\mathbf{p} = \begin{bmatrix} k_d & k_h & C_R & k_\omega \end{bmatrix}^T, \ \mathbf{y} = \begin{bmatrix} y & \dot{y} & z & \dot{z} & \theta & \dot{\theta} \end{bmatrix}^T \tag{6.22}$$

The forces acting on the quadcopter in this Y-Z plane motion are illustrated in Figure 6.23. We use the measured states as system states, and the equations of motion for the states are:

$$\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y}, k_d, k_h, C_R, k_\omega, \boldsymbol{\omega}_k, \phi_k) \tag{6.23}$$

where ϕ_k is the roll angle measurement at time t_k.

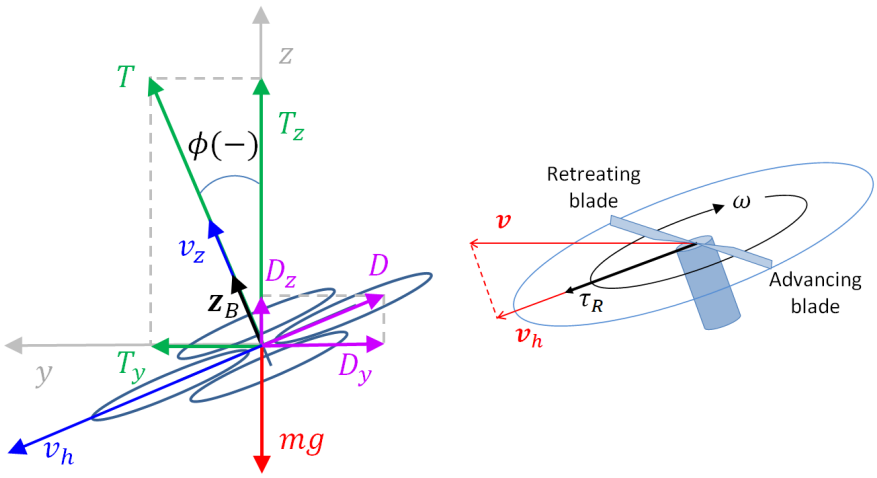

Figure 6.23. Quadcopter force diagram in fixed roll test.

Given known k_z, the individual rotor thrust is estimated as:

$$T_i = k_\omega \omega_i^2 - k_z v_z \omega_i + k_h v_h^2 \tag{6.24}$$

$$v_h = \dot{y} \cos \phi_k + \dot{z} \sin \phi_k \tag{6.25}$$

$$v_z = -\dot{y}\sin\phi_k + \dot{z}\cos\phi_k \tag{6.26}$$

The individual rotor drag is estimated as:

$$D_i = k_d \omega_i v_h \tag{6.27}$$

And the individual rotor torque caused by rolling effect is estimated as:

$$\tau_{Ri} = \epsilon_i \omega_i C_R v_h \tag{6.28}$$

where ϵ_i denotes the turning direction of the rotor, namely +1 (clockwise) or -1 (counter clockwise). We also calculate the body Y-axis torque generated by rotor thrust as:

$$\tau_T = (-T_1 + T_2 - T_3 + T_4)\cos\frac{\pi}{4}L \tag{6.29}$$

The final ODE is formed below and the 4th-order Runge-Kutta method is used for the integration with initial condition $\hat{\mathbf{y}}_0 = \tilde{\mathbf{y}}_0 = [\tilde{y}_0 \;\; \dot{\tilde{y}}_0 \;\; \tilde{z}_0 \;\; \dot{\tilde{z}}_0 \;\; \tilde{\theta}_0 \;\; \dot{\tilde{\theta}}_0]^T$.

$$\dot{\mathbf{y}} = \begin{bmatrix} \dot{y} \\ \ddot{y} = \frac{T_y - D_y}{m} = \frac{-T\sin\phi_k - D\cos\phi_k}{m} \\ \dot{z} \\ \ddot{z} = \frac{T_z + D_z - mg}{m} = \frac{T\cos\phi_k - D\sin\phi_k - mg}{m} \\ \dot{\theta} \\ \ddot{\theta} = \frac{\tau_T + \tau_R}{I_y} \end{bmatrix} \tag{6.30}$$

We take stable region of the flight data for analysis as the "Analysis Section" shown in Figure 6.21. To avoid interference, we identify k_d, k_h and k_ω from dominant Y-axis and Z-axis response first, and then identify C_R from pitching response using those identified

parameters. Figure 6.24 and Figure 6.25 show example test data and estimation results for one -4-degree roll test. In this analysis, all three parameters to be identified converged very fast, and the final estimation of the states fitted the measurement well. Comparing to the state estimations without the parameters as shown in blue dashed lines, we can clearly see the effect of k_d on Y-axis response, k_h on Z-axis response, and C_R on pitching response.

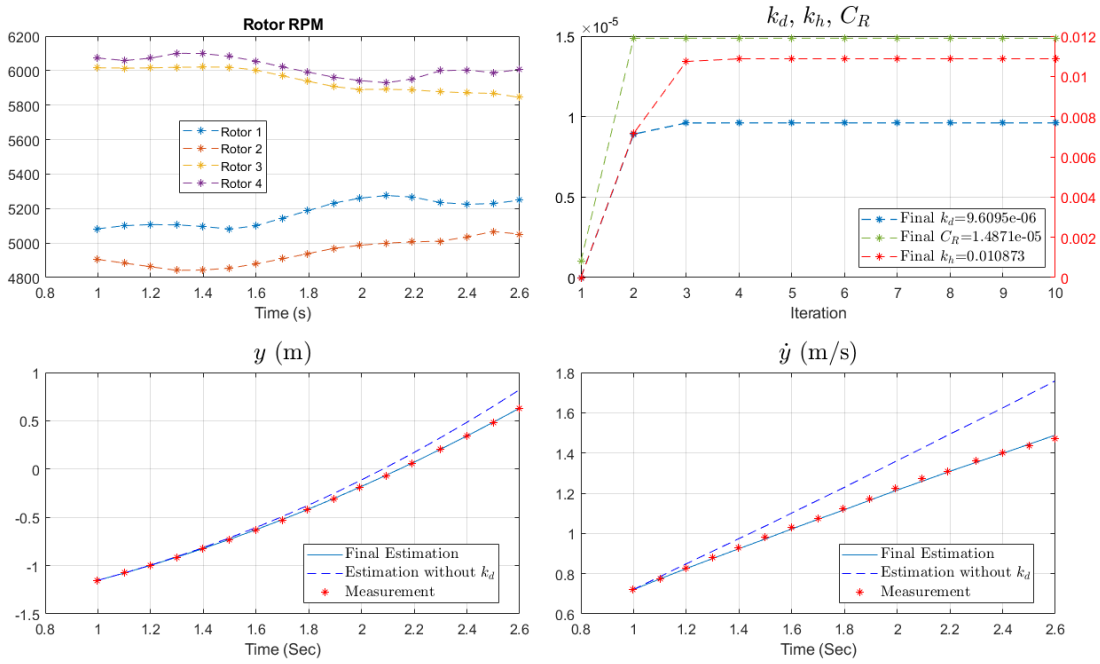

Figure 6.24. -4-degree roll test data and Y-axis estimation result.

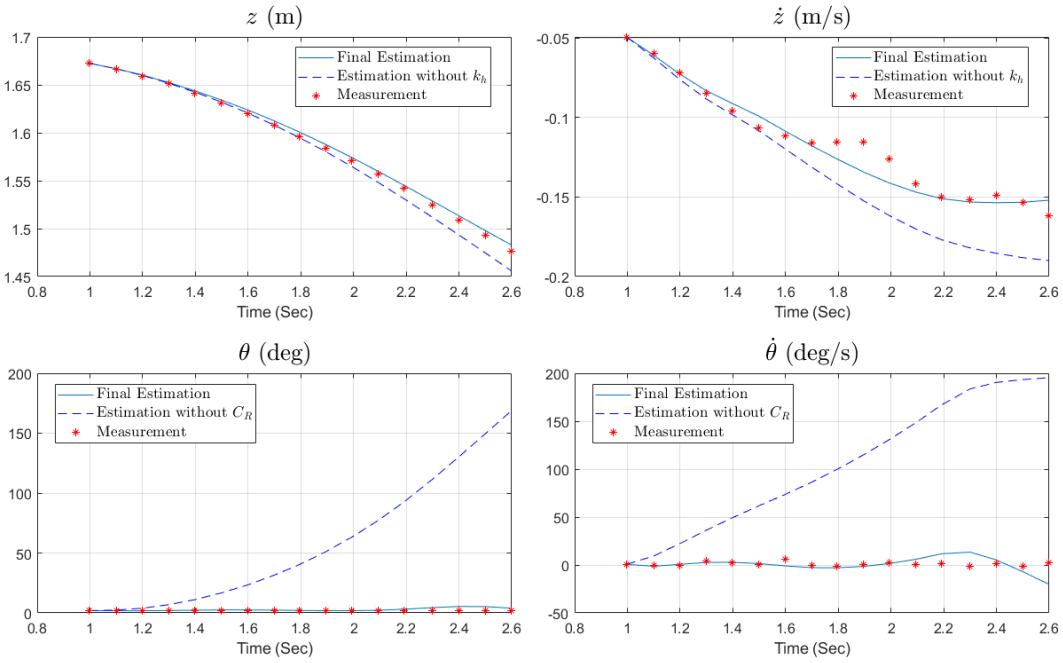

Figure 6.25. -4-degree roll test data and Z-axis and pitching estimation result.

The analysis results for 9 individual tests are summarized in Table 6.6. The identified k_ω values for the short test sections are within a reasonable range as we observed in the previous hover test. The rotor drag constant k_d, the thrust constant for horizontal velocity k_h, and the rolling torque constant C_R are identified as the average values:

$$k_d = 1.0587 \times 10^{-5} \mathrm{N \cdot s/m/RPM} \tag{6.31}$$

$$k_h = 2.1366 \times 10^{-2} \mathrm{N \cdot s^2/m^2} \tag{6.32}$$

$$C_R = 1.1689 \times 10^{-5} \mathrm{N \cdot s/RPM} \tag{6.33}$$

Table 6.6. Estimation results in fixed roll angle tests

Test No.	ϕ (deg)	k_ω	k_ω Diff (%)	k_d	k_h	C_R
1	-4	1.1710×10^{-7}	1.7	9.3210×10^{-6}	5.5964×10^{-2}	1.4180×10^{-5}
2	-4	1.1187×10^{-7}	-2.8	9.6136×10^{-6}	9.7912×10^{-3}	6.8488×10^{-6}
3	-4	1.1124×10^{-7}	-3.4	9.6095×10^{-6}	1.0873×10^{-2}	1.4871×10^{-5}
4	-8	1.1437×10^{-7}	-0.7	1.2812×10^{-5}	2.1167×10^{-2}	1.1145×10^{-5}
5	-8	1.1156×10^{-7}	-3.1	1.1639×10^{-5}	9.5773×10^{-3}	1.5832×10^{-5}
6	-8	1.1289×10^{-7}	-1.9	1.0434×10^{-5}	4.6570×10^{-3}	1.1538×10^{-5}
7	-12	1.1515×10^{-7}	0.0	1.1068×10^{-5}	1.4041×10^{-2}	1.4128×10^{-5}
8	-12	1.1290×10^{-7}	-1.9	1.2405×10^{-5}	3.6166×10^{-2}	6.2805×10^{-6}
9	-12	1.1607×10^{-7}	0.8	8.3839×10^{-6}	3.0062×10^{-2}	1.0382×10^{-5}

6.3.6 Fixed Position Turning Test

In this test, we send various yaw rate commands to Pixhawk in "Offboard" mode through MAVROS "/mavros/setpoint_raw/local" topic via "PositionTarget" message during fixed position hovering. The yawing response and RPM data in this test are shown in Figure 6.26 and Figure 6.27. From the RPM data, we can observe a constant RPM difference of around 1000 RPM between CW and CCW rotors which indicates a constant yawing control effort. The configuration of the quadcopter is quite symmetric, and we do not observe obvious misalignment of the quadcopter frame or rotors visually. Therefore, we assume that there is a slight rotor misalignment which causes the yawing moment and try to estimate the misalignment angle in the data analysis.

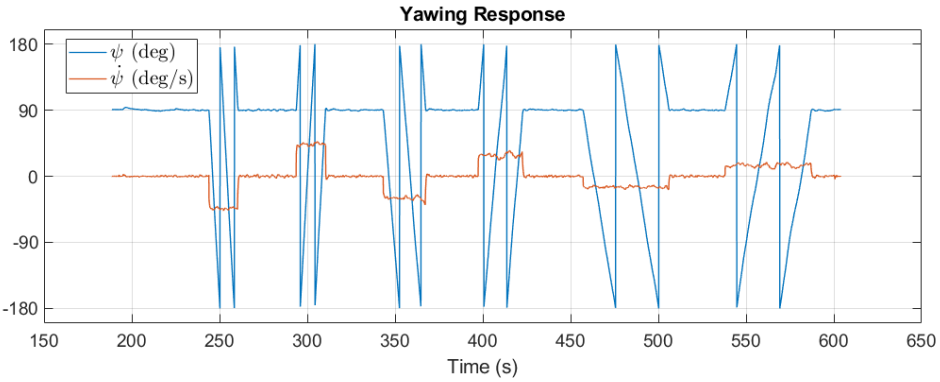

Figure 6.26. Yawing response in fixed position turning test.

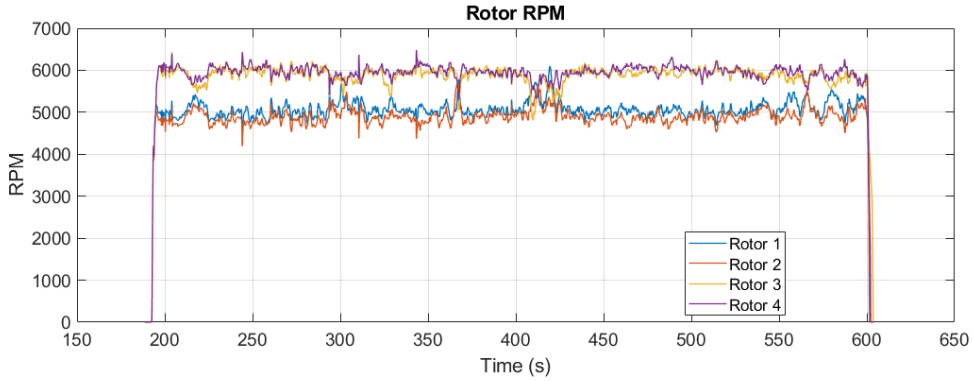

Figure 6.27. RPM data in fixed position turning test.

For maximum likelihood estimation, we use yaw angle ψ and yaw rate $\dot{\psi}$ as measurement, and the rotor counter-moment constant k_M and the rotor misalignment angle δ_r are the main parameters to be identified. As in last test, we also identify k_ω with additional measurement z and $\dot{z}$ in each relatively short test section and use previously identified average value as the initial guess. The identification and measurement vectors are:

$$\mathbf{p} = \begin{bmatrix} k_M & \delta_r & k_\omega \end{bmatrix}^T, \ \mathbf{y} = \begin{bmatrix} z & \dot{z} & \psi & \dot{\psi} \end{bmatrix}^T \tag{6.34}$$

The forces and moments acting on the quadcopter in this turning motion are illustrated in Figure 6.28. We use the measured states as system states, and the equations of motion for the states are:

$$\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y}, k_M, \delta_r, k_\omega, \boldsymbol{\omega}_k) \tag{6.35}$$

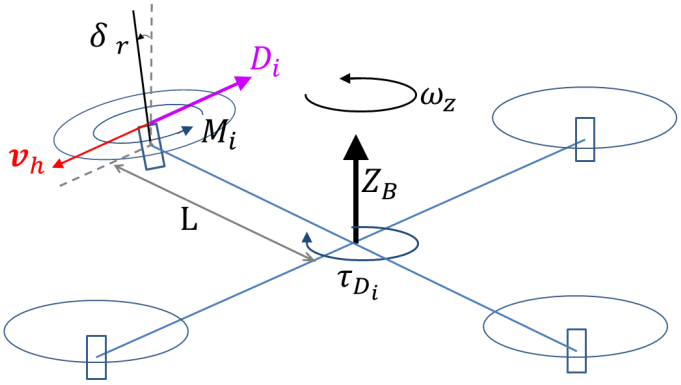

Figure 6.28. Quadcopter force and moment diagram in fixed position turning test.

Given known k_ω and k_h, the individual rotor thrust is estimated as:

$$T_i = k_\omega \omega_i^2 - k_z v_z \omega_i + k_h v_h^2 \tag{6.36}$$

$$v_h = L\omega_z \quad , \quad v_z = \dot{z} \tag{6.37}$$

Because the quadcopter is flying horizontally in this test, we use yaw rate as Z-axis angular velocity:

$$\omega_z = \dot{\psi} \tag{6.38}$$

According to equation (3.3), individual rotor counter moment is estimated as:

$$M_i = \epsilon_i k_M T_i \tag{6.39}$$

Given known k_d, the individual rotor drag in the yawing motion is estimated as:

$$D_i = k_d \omega_i v_h \tag{6.40}$$

And the torque caused by individual rotor drag is estimated as:

$$\tau_{Di} = -D_i L \tag{6.41}$$

The torque caused by individual rotor thrust because of the rotor misalignment is estimated as:

$$M_{Ti} = T_i \sin \delta_r L \tag{6.42}$$

The final ODE is formed below and the 4th-order Runge-Kutta method is used for the integration with initial condition $\hat{\mathbf{y}}_0 = \tilde{\mathbf{y}}_0 = [\tilde{z}_0 \;\; \dot{\tilde{z}}_0 \;\; \tilde{\psi}_0 \;\; \dot{\tilde{\psi}}_0]^T$.

$$\dot{\mathbf{y}} = \begin{bmatrix} \dot{z} \\ \ddot{z} = \frac{T - mg}{m} \\ \dot{\psi} \\ \ddot{\psi} = \frac{M + M_T + \tau_D}{I_z} \end{bmatrix} \tag{6.43}$$

We take a 3-second section of the flight data for analysis in each yaw rate command execution. Figure 6.29shows example test data and estimation results for one -45 deg/s yaw rate test. In this analysis, both parameters to be identified converged very fast. Comparing to the state estimations without the rotor drag induced torque τ_D as shown in blue dashed lines, we can clearly see the effect of this damping term on the yawing response.

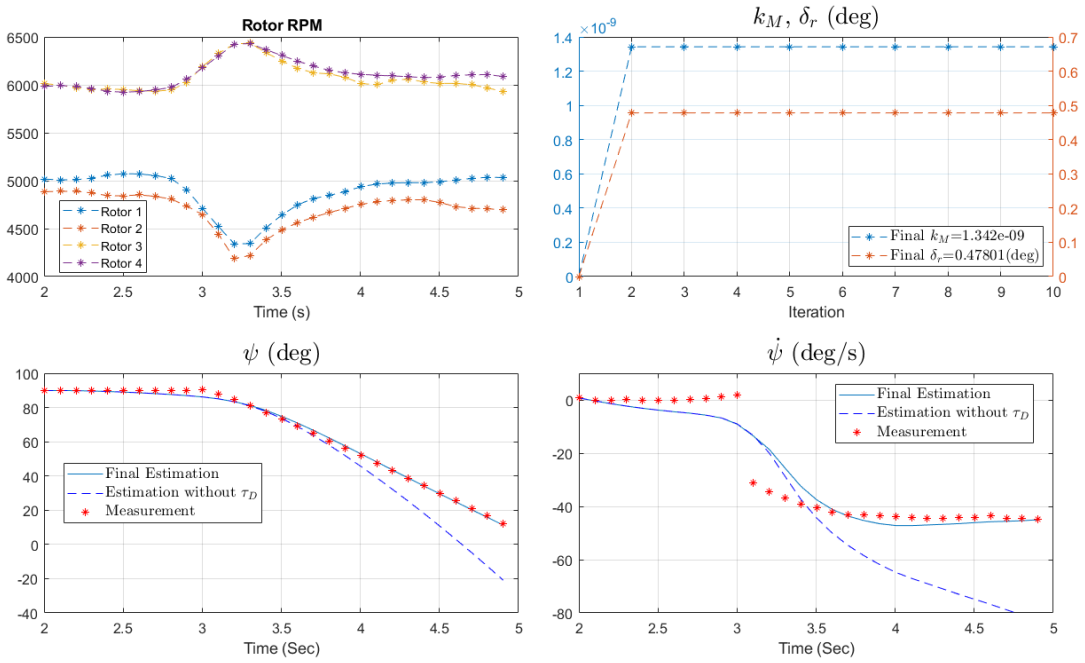

Figure 6.29. Fixed position turning test data and estimation result.

The analysis results for 8 individual tests are summarized in Table 6.7. The identified k_ω values for the short test sections are within a reasonable range as we observed in the previous hover test. The rotor counter-moment constant k_M and the rotor misalignment angle δ_r are identified as the average values:

$$k_M = 0.0112 \text{ m} \tag{6.44}$$

$$\delta_r = 0.51 \text{ deg} \tag{6.45}$$

The estimation result of the rotor misalignment angle turned out to be **as small as 0.51 deg**. This is consistent with our hypothesis of a slight rotor misalignment which is

Table 6.7. Estimation results in fixed position turning tests

Test No.	Yaw Rate (deg/s)	k_ω	k_ω Diff (%)	k_M	δ_r (deg)
1	-15	1.1305×10^{-7}	-1.8	0.01145	0.36
2	-30	1.1406×10^{-7}	-0.9	0.011797	0.55
3	+30	1.1524×10^{-7}	0.1	0.011833	0.68
4	-30	1.1369×10^{-7}	-1.3	0.010172	0.29
5	+30	1.1286×10^{-7}	-2.0	0.009425	0.55
6	-45	1.1539×10^{-7}	0.2	0.011462	0.47
7	+45	1.1477×10^{-7}	-0.3	0.013041	0.67
8	-45	1.1348×10^{-7}	-1.4	0.010377	0.53

hard to notice in visual inspection. We will add this additional parameter and its effect into the quadcopter dynamic model in following research.

In these tests from Section 6.3.4 to Section 6.3.6, the additional identification of k_ω in the relatively short test sections is very important because k_ω is a key factor affecting all the vehicle response. And as shown in Section 6.3.3, k_ω fluctuates during the flight tests due to air disturbance. With a biased k_ω, all other estimations would be inaccurate and divergent.

All the vehicle dynamic states used in the system identification flight tests are obtained from the PX4 EKF2 estimator. For detail parameter settings of the EKF2 estimator please refer to APPENDIX A.

Chapter 7

Trajectory Tracking Flight Test

After the flight test environment setup, the high fidelity SITL simulation setup and the model parameter identification, we move on to the actual trajectory tracking flight test. We start from the dynamic model modification to ensure that the model in out framework matches the actual quadcopter. Then we find the way to apply our control strategy based on the PX4 flight controller architecture. Next we conduct the flight controller tuning and various effect compensation tests. Finally we verify the trajectory tracking though various aggressive trajectories and conduct actual narrow window passing flight tests.

7.1 Dynamic Model Modification

We apply all the identified parameters to our dynamic model and SITL simulation model. Previously in our trajectory optimization framework and trajectory tracking controller development, we used the "plus" quadcopter configuration. However, the PX4 software uses the "cross" configuration as the default setting for DJI F450 quadcopter. Figure 7.1 shows a comparison of the two configurations.

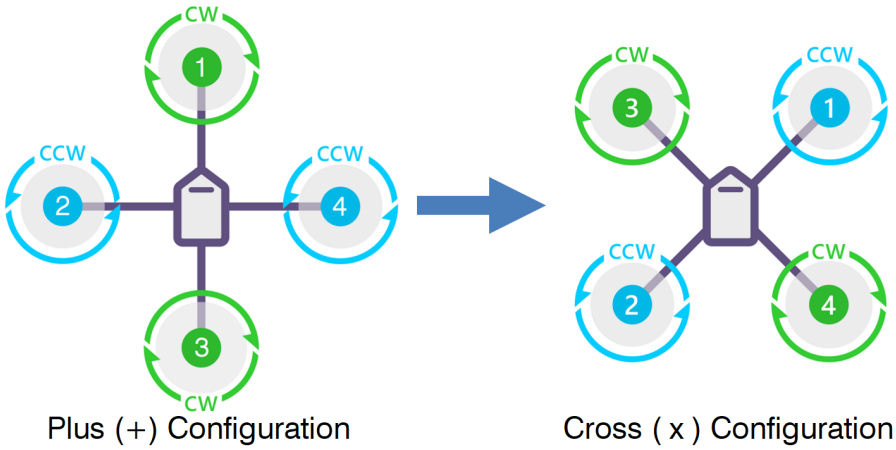

Figure 7.1. Plus and cross configuration for quadcopters.

While modifying the control allocation model, we also incorporate the slight rotor misalignment identified in Section 6.3.6 and its effect on Z-axis moment into the dynamic model. We consider the rotor misalignment angle δ_r and modify equation (3.4) as:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -L\cos\frac{\pi}{4} & L\cos\frac{\pi}{4} & L\cos\frac{\pi}{4} & -L\cos\frac{\pi}{4} \\ -L\cos\frac{\pi}{4} & L\cos\frac{\pi}{4} & -L\cos\frac{\pi}{4} & L\cos\frac{\pi}{4} \\ -k_M + L\sin\delta_r & -k_M + L\sin\delta_r & k_M + L\sin\delta_r & k_M + L\sin\delta_r \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} \tag{7.1}$$

7.2 Trajectory Tracking Control

So far we can use the native PX4 flight controller on the Pixhawk 4 mini autopilot module to perform the polynomial trajectory tracking via its offboard mode by feeding position, velocity, acceleration, yaw and yaw rate set-point from the on-board computer (Raspberry Pi) through"/mavros/setpoint_raw/local" topic via "PositionTarget" message at 50 Hz. However, the trajectory tracking will not be accurate this way. Figure 7.2 shows an

example Y-axis minimum snap trajectory tracking result. The tracking lag and overshoot are obvious even with such a simple and slow trajectory.

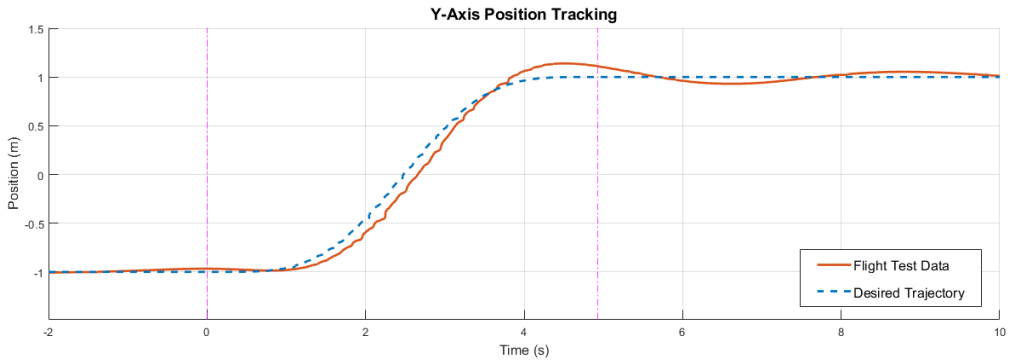

Figure 7.2. 1-D minimum snap trajectory tracking with native PX4 controller.

To achieve precise trajectory tracking, we need to tune the controller, check the mapping from acceleration commands to attitude set-points, and add the nonlinear feedforward terms in the quadcopter rotational dynamics and the compensations for rotor drag and other effects into the controller. Therefore, we plan to both utilize the on-board computer and modify the PX4 flight controller code to implement the full geometric controller with effects' compensation as proposed in Chapter 3 on top of the original PX4 flight controller architecture.

For trajectory tracking part in the modified geometric controller proposed in equation (3.61), we leave the position and velocity tracking ($k_x \boldsymbol{e}_x$ and $k_v \boldsymbol{e}_v$) and the acceleration to desired attitude mapping part to original PX4 controller. To deal with the rotor drag compensation, we implement the drag estimation in the trajectory control program in on-board computer. We obtain required vehicle states and rotor speed commands (PPM) through MAVROS interface. The actual rotor RPM is estimated by the identified function in equation (6.6). Finally, we convert the estimated rotor drag into an acceleration compensation

term and add it to the trajectory acceleration commands to be sent to Pixhawk. The acceleration compensation term is formulated as:

$$\boldsymbol{A}_C = \frac{k_d \omega_s \boldsymbol{R} P \boldsymbol{R}^T \boldsymbol{v}}{m} \tag{7.2}$$

For attitude tracking part in the modified geometric controller proposed in equation (3.64), we leave the angle and angular rate tracking ($k_R \boldsymbol{e}_R$ and $k_\omega \boldsymbol{e}_\omega$) part to original PX4 controller. We obtain required vehicle states through MAVROS interface and implement the inverse dynamics analysis in the trajectory control program in on-board computer. We obtain the trajectory desired angular rate $\boldsymbol{\omega}_d$ from the inverse dynamics analysis and send it to Pixhawk to be used as a feedforward angular rate compensation. Then we calculate the rotational dynamic feedforward terms and effect compensation terms and send it to Pixhawk to be used as a feedforward control moment compensation. The moment compensation term is formulated as:

$$\boldsymbol{M}_C = \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega} - \boldsymbol{I}(\hat{\boldsymbol{\omega}} \boldsymbol{R}^T \boldsymbol{R}_d \boldsymbol{\omega}_d - \boldsymbol{R}^T \boldsymbol{R}_d \dot{\boldsymbol{\omega}}_d) - \boldsymbol{\tau}_D - \boldsymbol{\tau}_G + \boldsymbol{\tau}_R \tag{7.3}$$

We utilize available MAVROS topic "/mavros/setpoint_raw/attitude" to send these information and create a unique type_mask code for this purpose.

On PX4 side, we create a uORB message that contains global variables for the additional compensation terms. We modify MAVLink code in "mavlink/mavlink_receiver.cpp" to assign the compensating variable values when the unique type_mask is received. We modify attitude control code in "mc_att_control/mc_att_control_main.cpp" to add the compensating angular rates to original vehicle rate set-point "vehicle_rates_setpoint" before it is published. We also modify rate control code in "mc_rate_control/MulticopterRateControl.cpp" to add the compensating moments to original vehicle torque set-point "vehicle_torque_setpoint"

before it is published. Note that this vehicle torque set-point only works when the dynamic control allocation is enabled through the "SYS_CTRL_ALLOC" setting.

The trajectory tracking control implementation is illustrated in Figure 7.3.

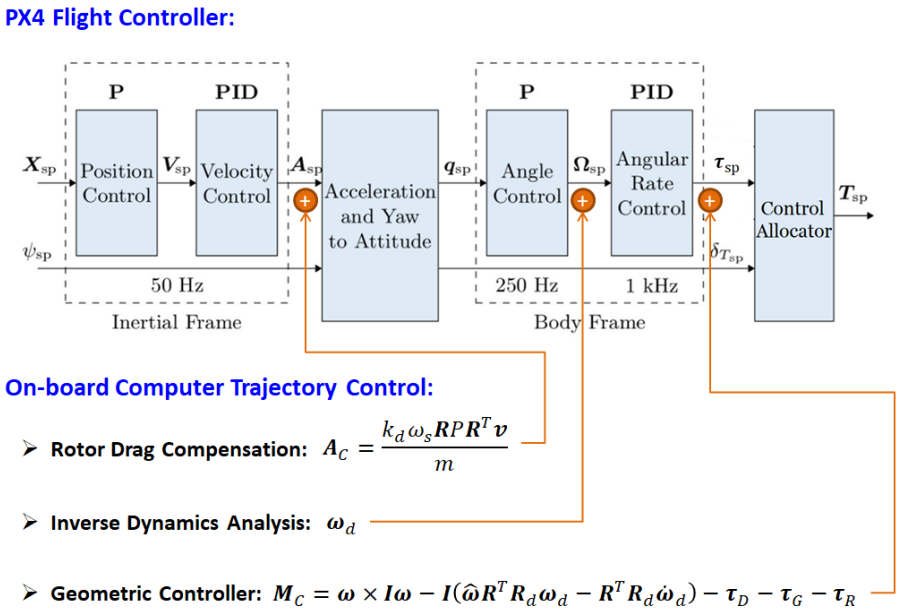

Figure 7.3. Trajectory tracking control architecture.

7.3 Controller Tuning

Attitude control is the fundamental part in this flight controller. For fast attitude tracking, we start the tuning from the rate controller which is the inner-most loop with three independent PID controllers to control the body rates (yaw, pitch, and roll). Figure 7.4 shows the tuning of the roll rate controller with a -5 deg/s step input. Note that the roll rate data is filtered for clear comparison because the actual flight data is noisy with high frequency oscillations. Figure 7.5 shows the actual flight data and the filtered result for the $k_P = 0.25$ test. At $k_P = 0.3$, the system starts to show signs of instability during flight.

Therefore, we end up this tuning and choose to set the gains as $k_{p_{\dot{\phi}}} = 0.25$, $k_{D_{\dot{\phi}}} = 0.004$, and $k_{I_{\dot{\phi}}} = 0.15$.

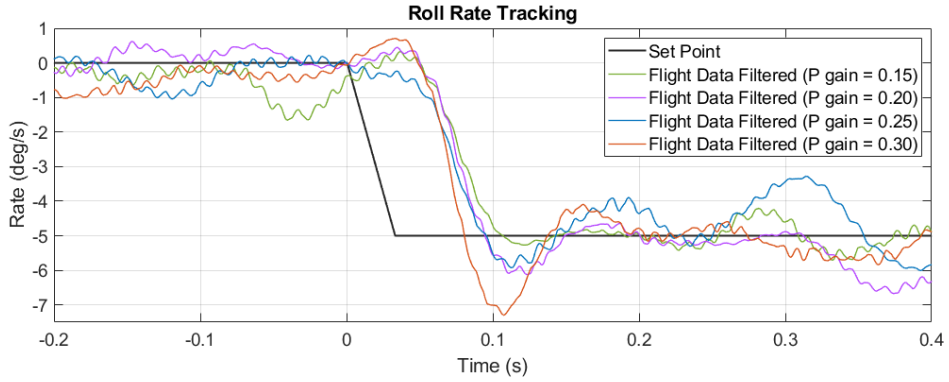

Figure 7.4. Roll rate controller gain tuning.

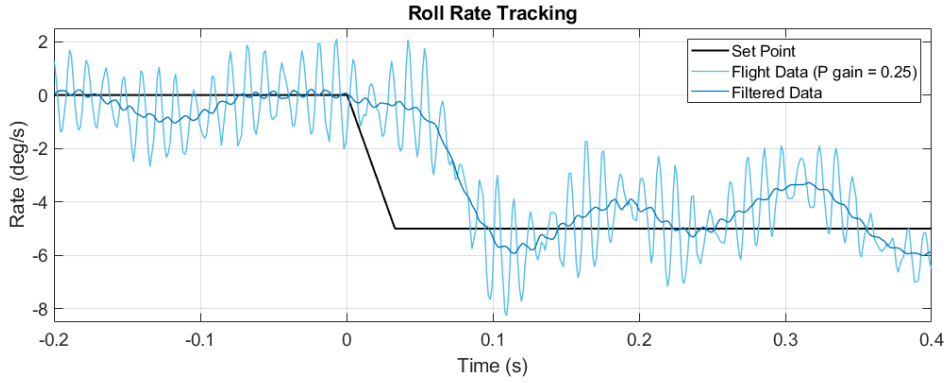

Figure 7.5. Actual data and filtered data in roll rate tracking.

Then we move on to roll controller tuning. The outer loop of the attitude control in PX4 is a simple proportional control. Figure 7.6 shows the tuning of the roll controller with a -5 deg step input. We finally choose to set the gain as $k_{p_{\phi}} = 10$ because this is already very close to the upper limit recommended by PX4.

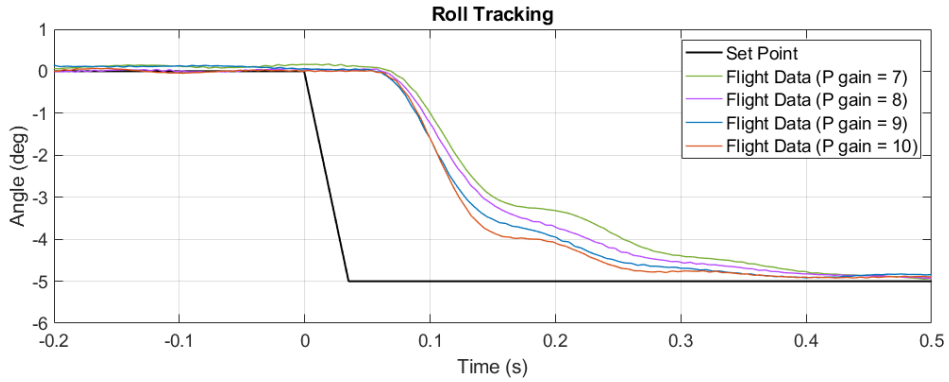

Figure 7.6. Roll controller gain tuning.

Although we try to tune the controller for faster response, an obvious delay from the command to the actual action in both roll and roll rate tracking is observed. This has to do with the rotor effect time constant. Figure 7.7 shows an example of rotor response during flight. Although the sampling rate is not high enough to reveal dynamic details, we can still observe the delay from the PPM command peak to the actual RPM peak.

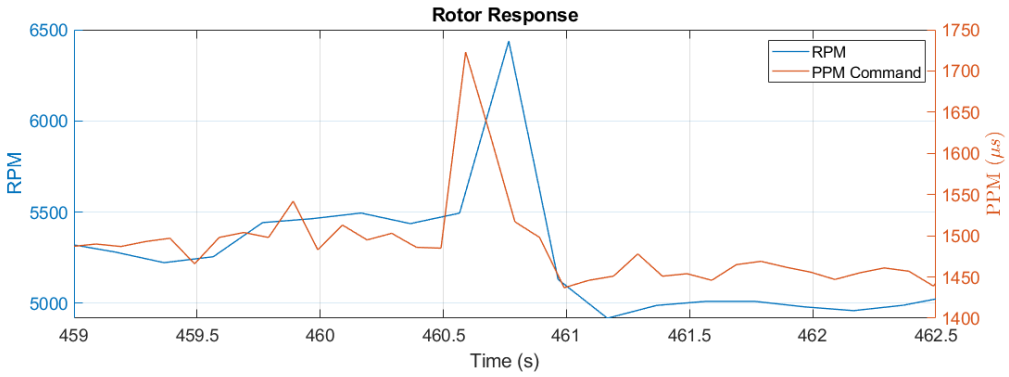

Figure 7.7. Rotor response example.

As the attitude controller is tuned up, we test the trajectory tracking with a simple Y-axis 1-segment minimum snap trajectory which goes from $y = -1$ to $y = 1$ in 3 seconds.

From the comparison shown in Figure 7.8, we can see that the position tracking is slightly better but not significantly improved after the gain tuning.

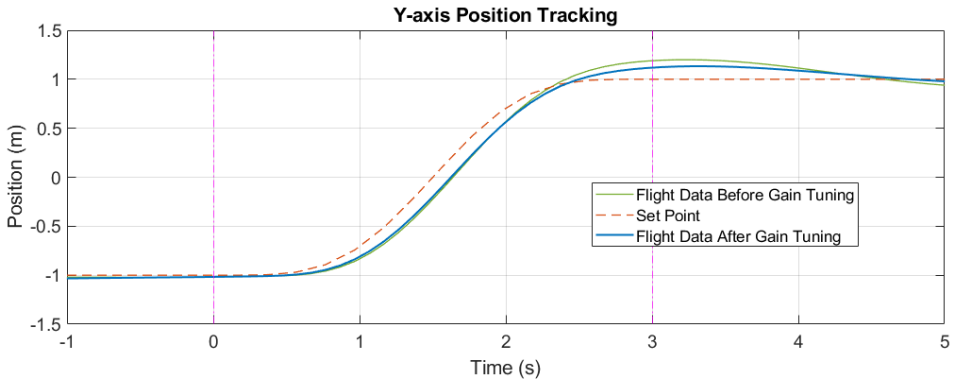

Figure 7.8. Trajectory tracking comparison in attitude control gain tuning.

However, the improvement in the roll angle tracking during the trajectory tracking is obvious. Figure 7.9 shows the roll angle tracking result before the attitude control gain tuning. The red dashed line shows the roll set-point which is the desired roll angle generated by the flight controller. From the figure, we can see that the tracking lag is around 0.16 seconds, and the lag time varies with flight condition. Figure 7.10 shows the tracking result with new control gains. The tracking lag is now around 0.11 seconds which indicates a 31% improvement. More importantly, the lag time in this case is almost like a constant. It is inferred that, because the attitude controller has been tuned fast enough, the attitude tracking lag time is now mainly caused by the rotor effect time constant and therefore does not change much with flight condition. This gives us an edge on the next trial.

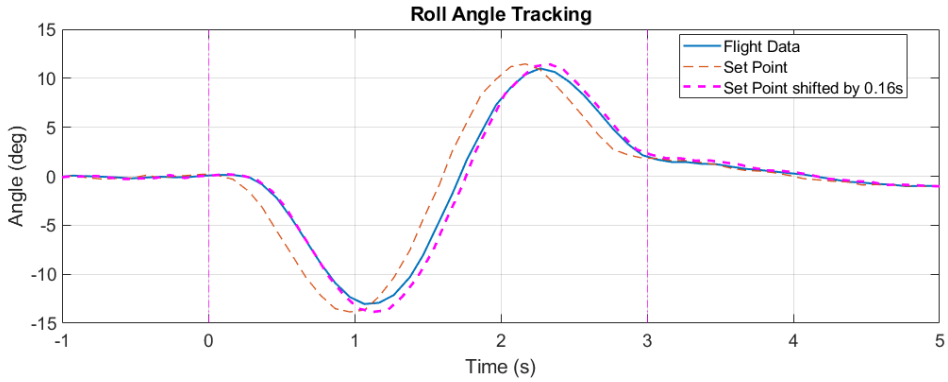

Figure 7.9. Roll tracking before the attitude control gain tuning.

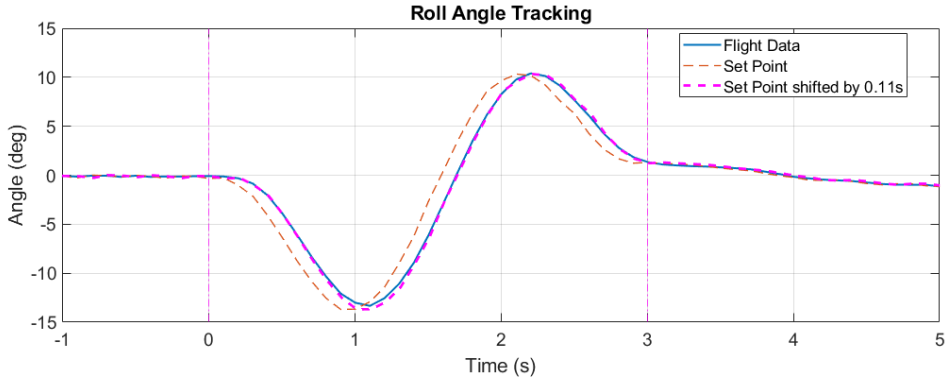

Figure 7.10. Roll tracking after the attitude control gain tuning.

Knowing that the attitude tracking lag time is almost a constant and the nominal desired attitude comes from the acceleration commands, we next try to pre-send the trajectory acceleration commands by a fixed lead time equal to the attitude tracking lag time, 0.11 seconds. We also compare the roll set-point generated by the flight controller with the roll angle estimation obtained from our inverse dynamics analysis. From the position tracking comparison shown in Figure 7.11, we can see that, by pre-sending acceleration commands, the tracking lag in the beginning of the trajectory is improved, but the tracking error accumulated gradually and exceeded the previous case in the late part of the trajec-

tory. From the roll tracking comparison shown in Figure 7.12, we can see that, unlike the previous case, the new roll response matched the estimation very well in the first 0.3 seconds due to the pre-sending of acceleration commands. However, after 0.3 seconds the roll trajectory gradually deviated from the estimation. It is observed that the slope of the set-point curve is obviously flatter than that of the estimation throughout the trajectory. In PX4 controller, there are only three factors affect the desired attitude, position tracking error, velocity tracking error, and acceleration command. In the first 0.5 seconds, the position and velocity tracking are good. All these observations imply that the scaling from acceleration command to desired attitude in PX4 controller is being conservative.

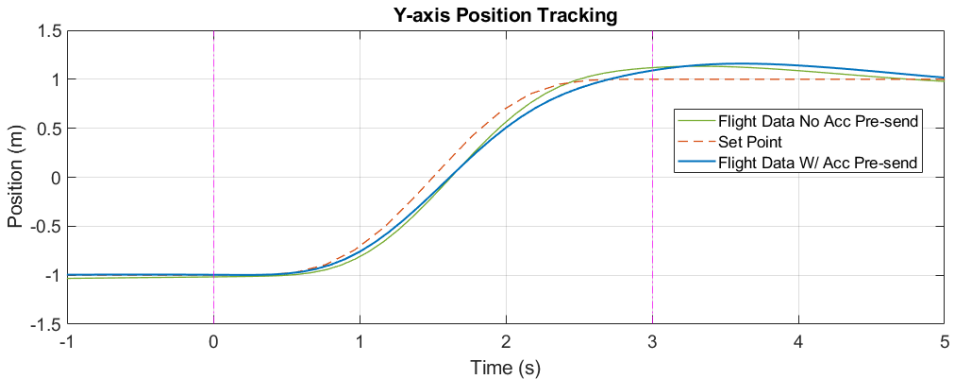

Figure 7.11. Trajectory tracking comparison in acceleration command pre-sending.

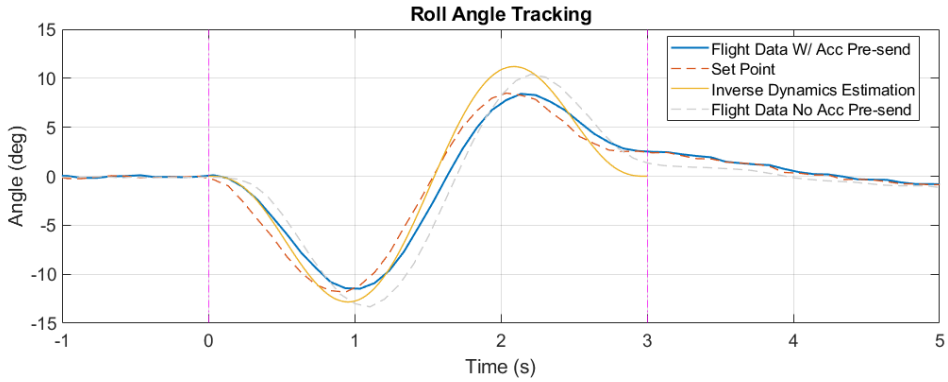

Figure 7.12. Roll tracking comparison in acceleration command pre-sending.

Next we try to reasonably match the attitude set-point curve slope in low velocity region by amplifying the acceleration commands to be sent to PX4 controller by a scaling factor k_a. From the roll tracking comparison shown in Figure 7.13, we can see that, unlike the previous case shown in the gray dashed line, the new roll response with $k_a = 1.3$ matched the estimation very well in the first 0.9 seconds. In fact, the whole roll trajectory got significantly closer to the estimation not only due to the acceleration command scaling but also because of the reduction of position and velocity tracking error. From the position and velocity comparison shown in Figure 7.14 and Figure 7.15, we can see how the tracking result got improved gradually as k_a being tuned step by step. Now the remaining issue is that the vehicle velocity cannot catch up with the velocity command in high velocity region, so that the position tracking also got lagged from the middle of the trajectory. This has to do with the rotor drag.

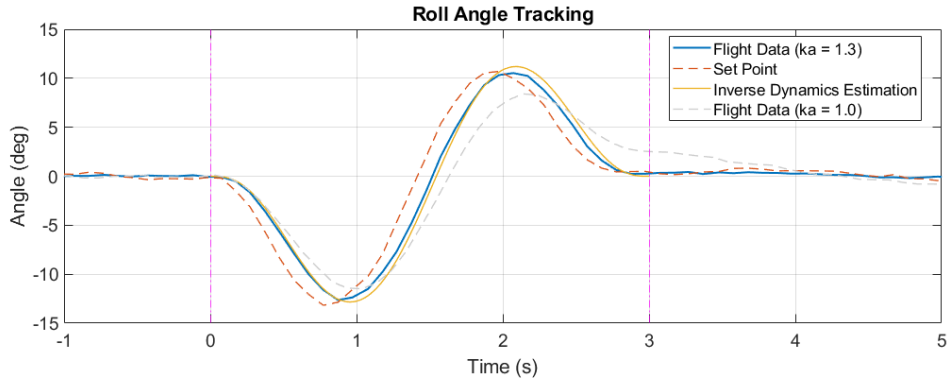

Figure 7.13. Roll tracking comparison in acceleration command scaling.

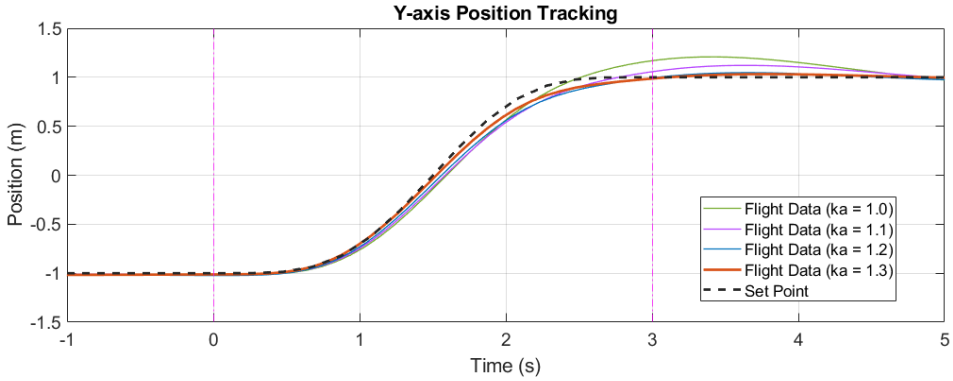

Figure 7.14. Position tracking comparison in acceleration command scaling.

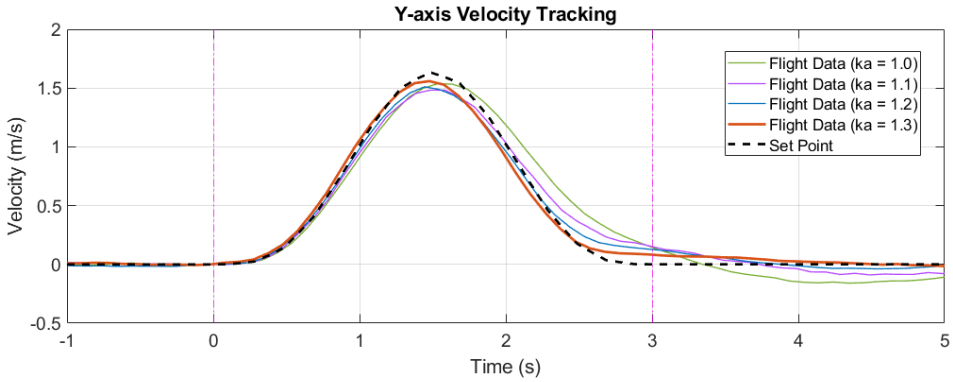

Figure 7.15. Velocity tracking comparison in acceleration command scaling.

Finally we added the compensations for rotor drag and other effects and achieved the precise trajectory tracking. Figure 7.16-7.18 shows the final position, velocity, and roll angle tracking result. The position and velocity tracking are good except for the slight tracking error in the final phase of the trajectory. The actual roll trajectory to achieve this precise tracking exactly matches the inverse dynamics estimation. This also confirms the precision of the inverse dynamics analysis and the rotor drag model. It is learned here that the test/tuning order is important, and the inverse dynamics estimation is very helpful in this controller tuning process.

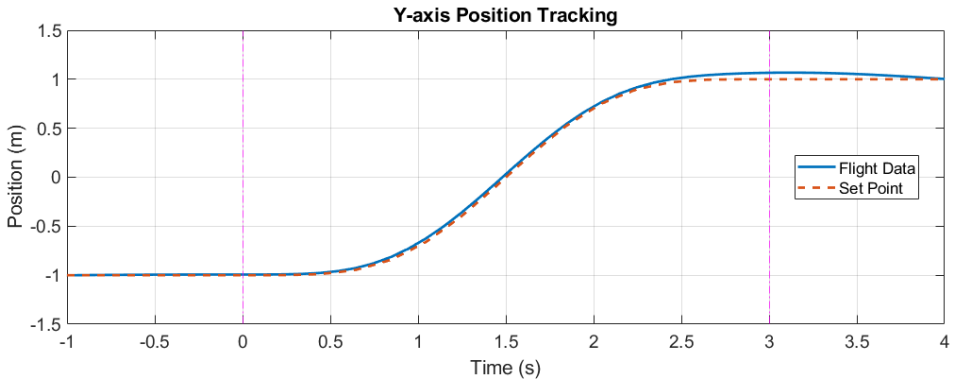

Figure 7.16. Final position tracking result.

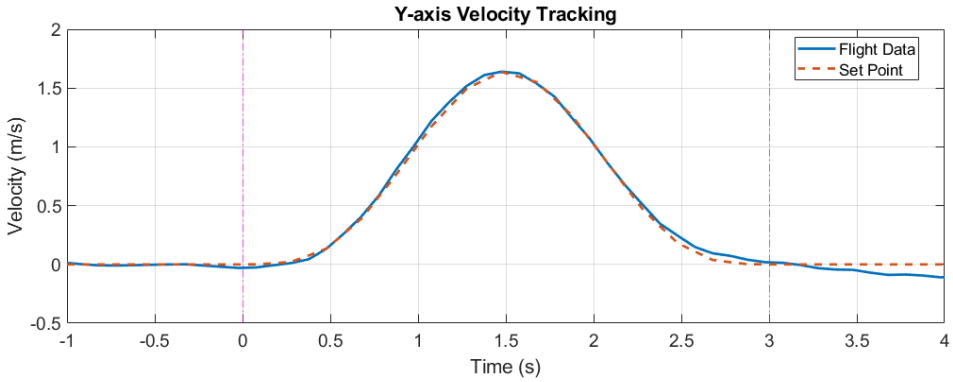

Figure 7.17. Final velocity tracking result.

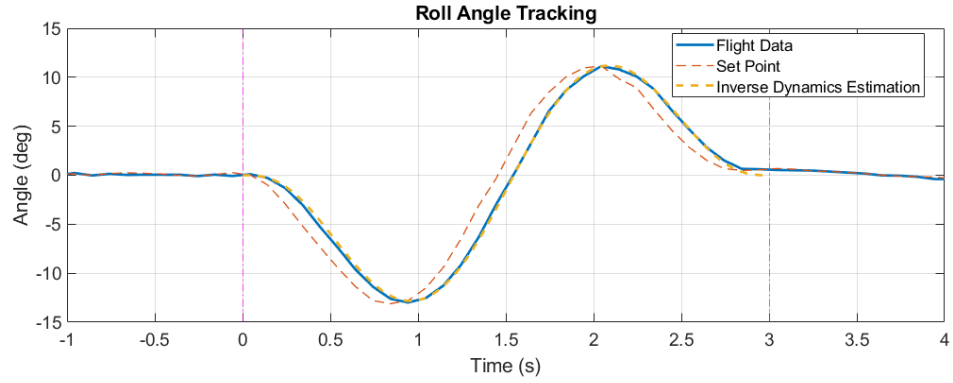

Figure 7.18. Final roll angle tracking result and comparison.

7.4 Multi-segment Aggressive Trajectory Tracking and Trajectory Aggressiveness Verification

After the controller tuning, we applied the finalized roll controller gains to the pitch controller and verified the tracking performance in X-axis 1-segment minimum snap trajectory tracking tests. The next step is to test actual 3D multi-segment aggressive trajectory tracking. We conduct the test with a 3-waypoint scenario. The waypoint settings are tabulated in Table 7.1.

Table 7.1. Waypoint settings in the scenario

WPT Setting	x (m)	y (m)	z (m)	ψ (deg)
Waypoint 1	0.5	-1.5	1	0
Waypoint 2	-0.5	0	1	0
Waypoint 3	0.5	1.5	1	0

First we specify the maximum rotor thrust as 4.8 N in the trajectory planning and test the 2-segment trajectory tracking. Figure 7.19 shows the 3D plot of the trajectory and the flight sequence obtained from MATLAB/Simulink simulation. Figure 7.20 shows the sequence stock image of the actual trajectory tracking flight.

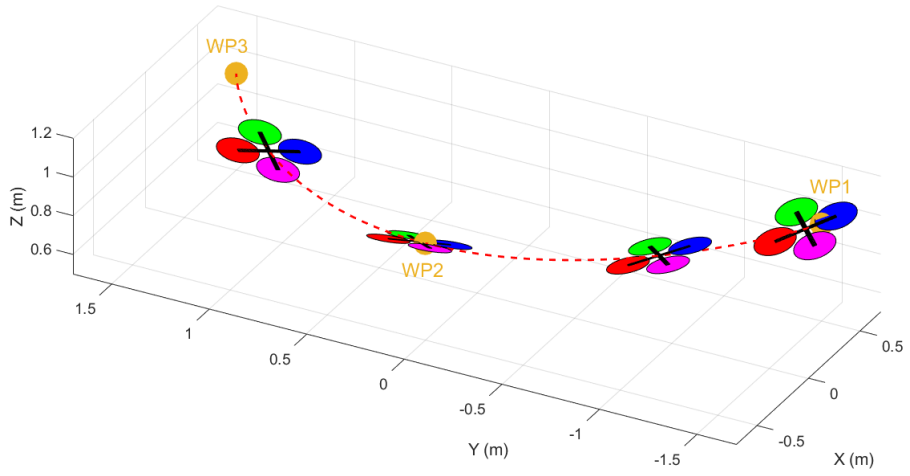

Figure 7.19. The 2-segment trajectory and MATLAB/Simulink simulation.

Figure 7.20. The 2-segment trajectory tracking flight.

Figure 7.21-7.23 shows the position and velocity tracking and the vehicle attitude comparison in this trajectory tracking flight. From the flight data, we can see that the trajectory tracking is still quite precise, though slightly higher tracking error is observed in this more complicated and more aggressive trajectory tracking.

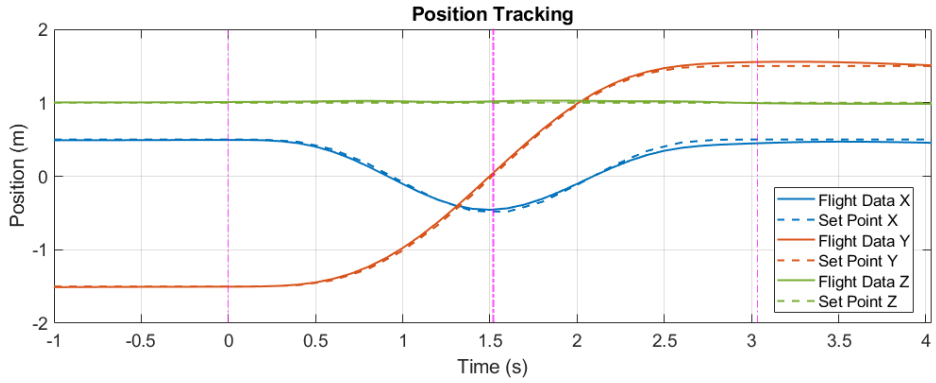

Figure 7.21. Position tracking result in 4.8N max-thrust test.

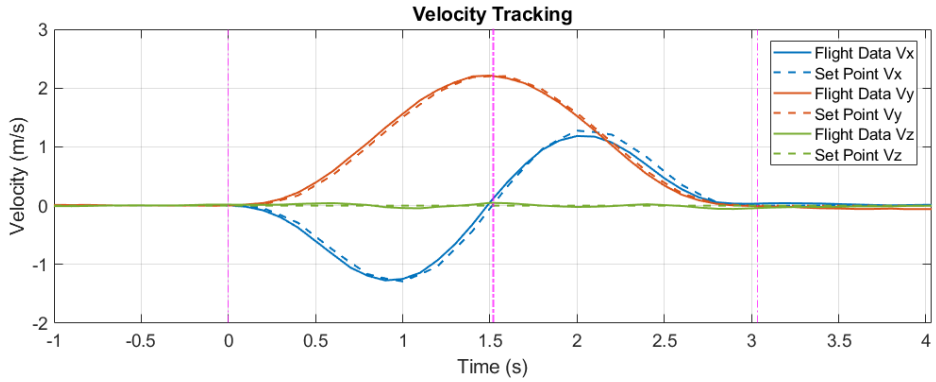

Figure 7.22. Velocity tracking result in 4.8N max-thrust test.

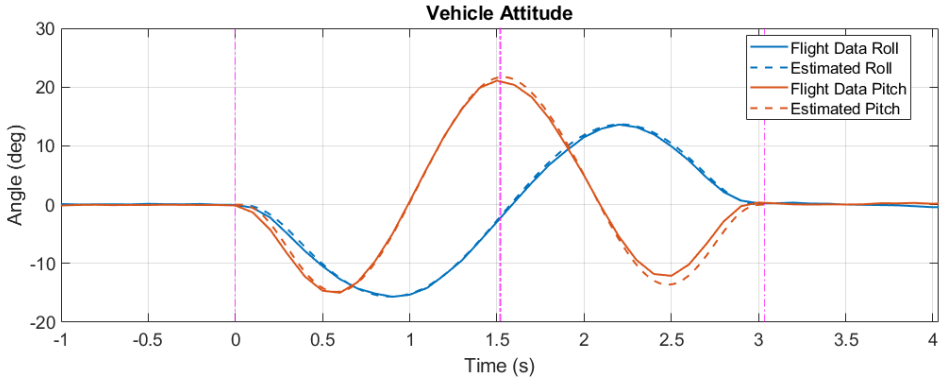

Figure 7.23. Attitude data comparison in 4.8N max-thrust test.

Next we use the same waypoint settings and test different aggressive levels by specifying different maximum rotor thrust. We generate optimized trajectories for 4.2N, 4.5N, 4.8N, and 5.1N maximum thrust and conduct three flight tests for each trajectory. The test results are tabulated in Table 7.2.

The maximum RPM estimation in our trajectory planning comes from the maximum thrust required in the trajectory and its corresponding flight condition. The actual maximum RPM used in a trajectory tracking flight will be affected by trajectory tracking error, air disturbance, and other factors. However, from the test results we can see that, in our

Table 7.2. Trajectory aggressiveness test results

Test No.	Specified Max Thrust (N)	Flight Time (s)	Estimated Max RPM	Actual Max RPM	Difference (%)
1-1	4.2	4.41	6065	6378.2	5.16
1-2	4.2	4.41	6065	6329.7	4.36
1-3	4.2	4.41	6065	6338.4	4.51
2-1	4.5	3.40	6302	6564.0	4.16
2-2	4.5	3.40	6302	6633.8	5.27
2-3	4.5	3.40	6302	6444.7	2.26
3-1	4.8	3.03	6538	6629.7	1.40
3-2	4.8	3.03	6538	6671.9	2.05
3-3	4.8	3.03	6538	6638.6	1.54
4-1	5.1	2.83	6756	6871.3	1.71
4-2	5.1	2.83	6756	6697.2	-0.87
4-3	5.1	2.83	6756	6814.7	0.87

test region, the differences between the actual maximum RPM and the estimation are all **within 5.3%**. The estimation would not be as close to actual value if the additional **rotor misalignment** identified in Section 6.3.6 is not considered in our dynamic model. **This result verifies the specification of maximum rotor thrust in our aggressive trajectory synthesis and proves the practical value of our trajectory optimization framework and inverse dynamics analysis**. In fact, based on our framework, we can modify the trajectory optimization to trace maximum individual rotor RPM instead of maximum thrust. This could be more practical in real application.

7.5 Narrow Window Passing Verification

Having the precise tracking of multi-segment aggressive trajectory achieved, we move on to the actual narrow window passing test. We still use the 3-waypoint scenario as in the previous section. One narrow window is added at waypoint 2. The window is sized to have 10 cm of clearance each side from the quadcopter at the center as shown in Figure

7.24. And Figure 7.25 shows the 3D plot of the trajectory and the flight sequence obtained from MATLAB/Simulink simulation.

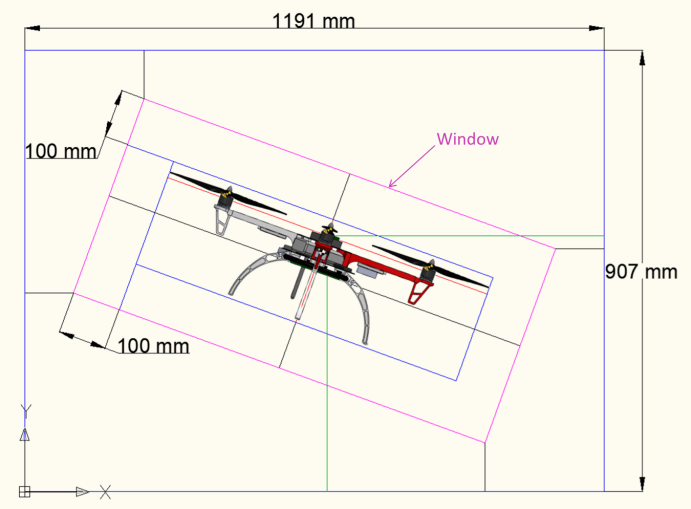

Figure 7.24. 20-degree tilted narrow window geometry.

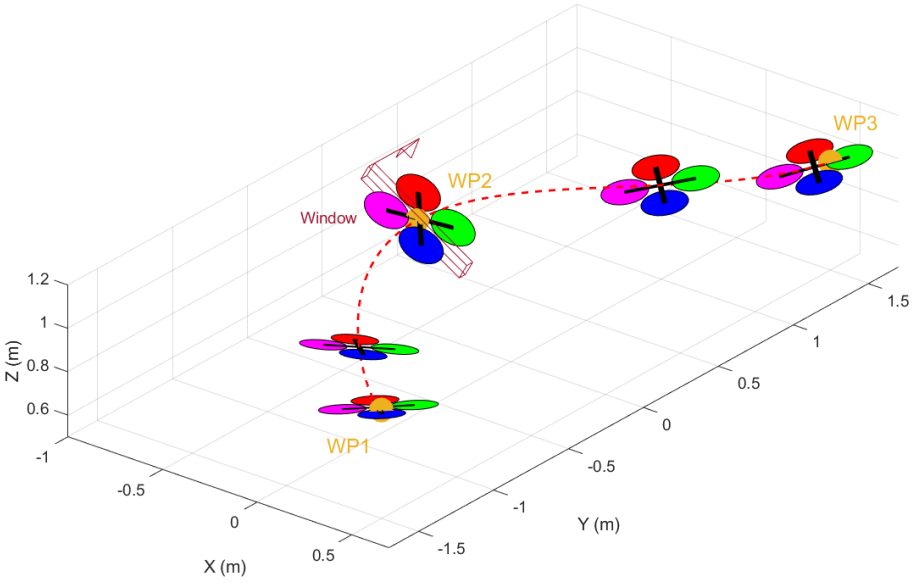

Figure 7.25. The window passing trajectory and MATLAB/Simulink simulation.

We test two different window tilted angle, 20 and 30 degrees. Figure 7.26 shows the sequence stock image of the actual 20-degree tilted narrow window passing flight. The maximum rotor thrust in this trajectory is specified as 5.0N. Figure 7.27 and Figure 7.28 show the position tracking and the attitude comparison in this flight. The tracking error at the window position (marked with thick vertical purple dash-dotted line) is very small; therefore the successful window passing is achieved.

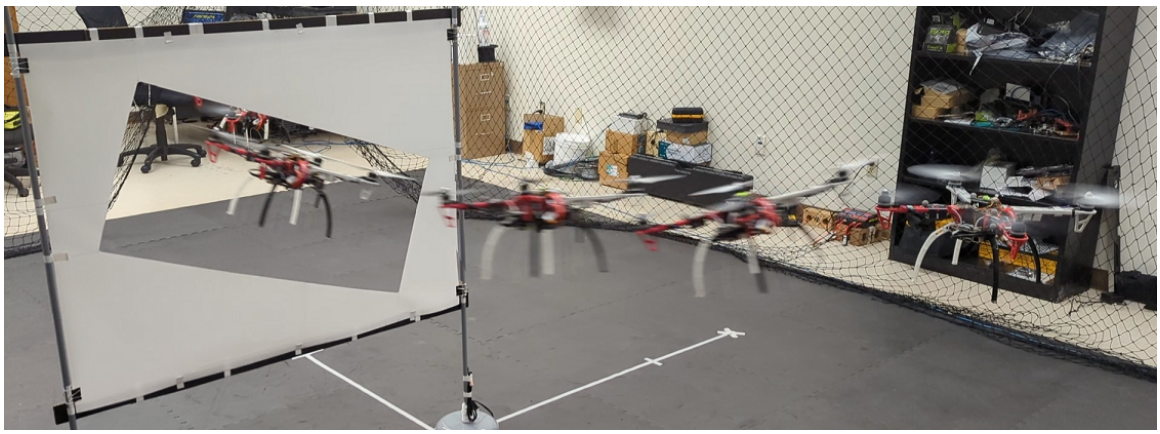

Figure 7.26. The 20-degree window passing flight.

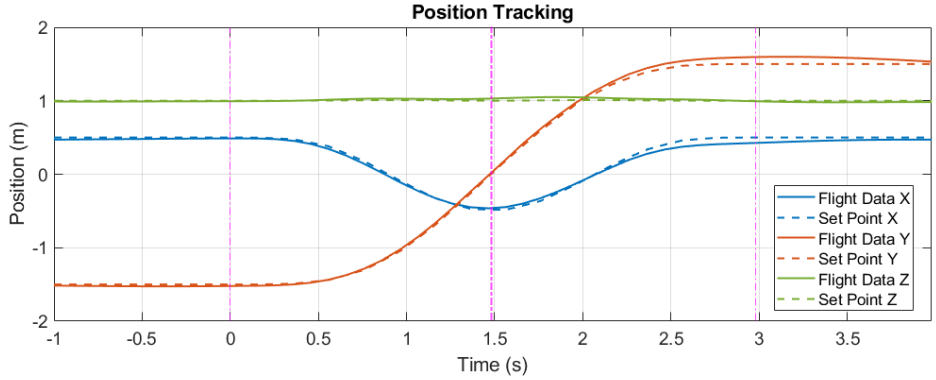

Figure 7.27. Position tracking result in 20-degree window passing flight.

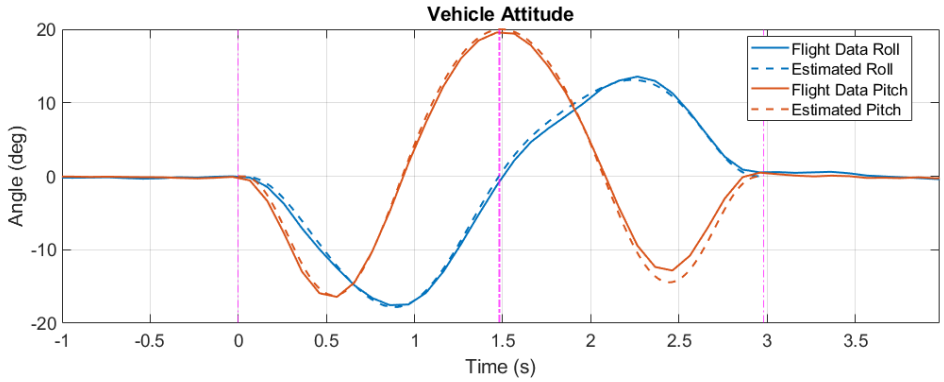

Figure 7.28. Attitude data comparison in 20-degree window passing flight.

Figure 7.29 shows the sequence stock image of the 30-degree tilted narrow window passing flight. The maximum rotor thrust in this trajectory is also specified as 5.0N. From the position tracking and the attitude comparison shown in Figure 7.30 and Figure 7.31, we can see that the tracking error and attitude difference are more obvious. However, the position error at the window passing moment is still within 3 cm, and the angular difference is with 2.5 deg. Therefore, the window passing is still successful.

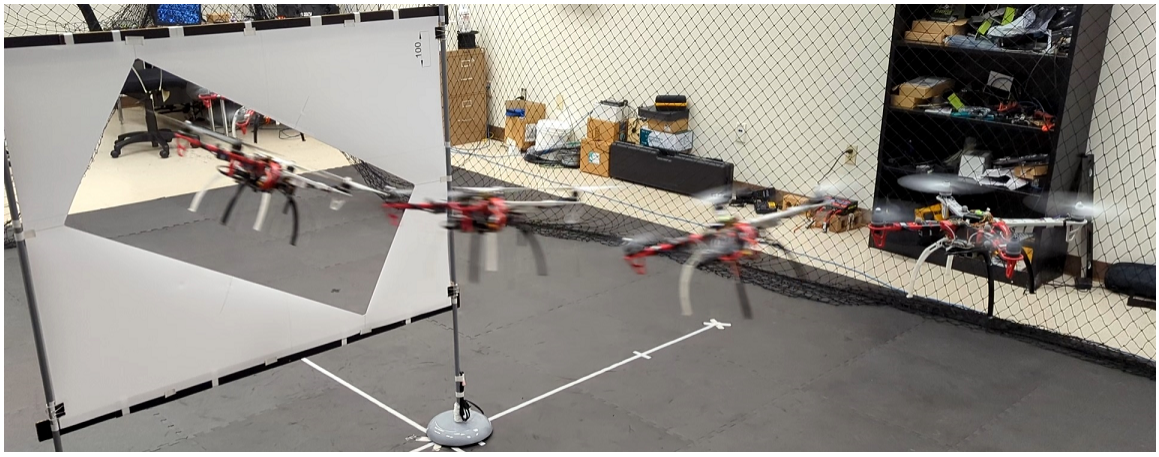

Figure 7.29. The 20-degree window passing flight.

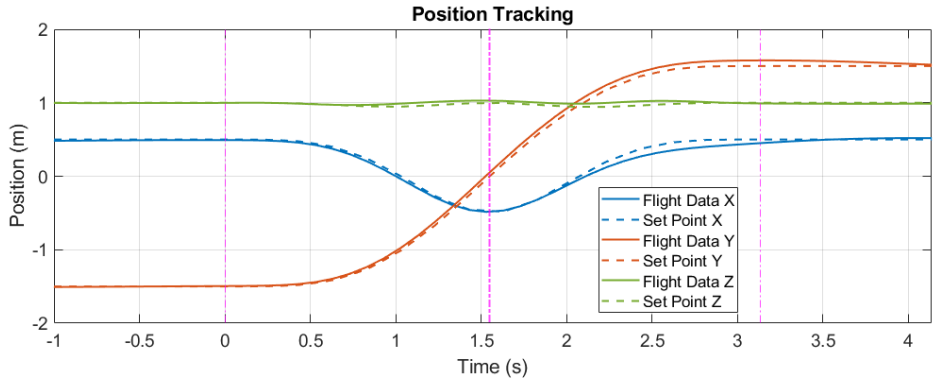

Figure 7.30. Position tracking result in 20-degree window passing flight.

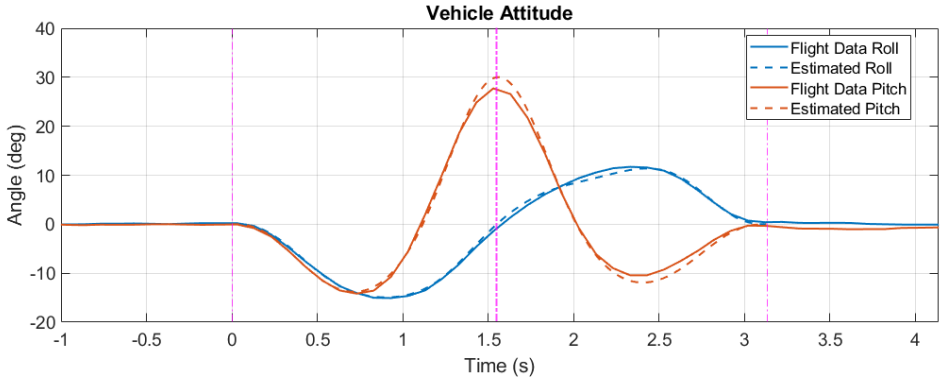

Figure 7.31. Attitude data comparison in 20-degree window passing flight.

We also generate and test another trajectory for 20-degree window and 4.5N maximum thrust for comparison. From Figure 7.32 we can see that, comparing between trajectory 1 and trajectory 2, trajectory 2 makes a lower curvature at the window position. This is because they need the same centrifugal acceleration for the same window angle, and trajectory 2 is faster (more aggressive) because of higher specified maximum thrust. And by comparing between trajectory 2 and trajectory 3, we can see that trajectory 3 makes higher curvature at the window position to generate higher centrifugal acceleration to match the

higher tilted window given they having similar speed because of the same specified maximum thrust.

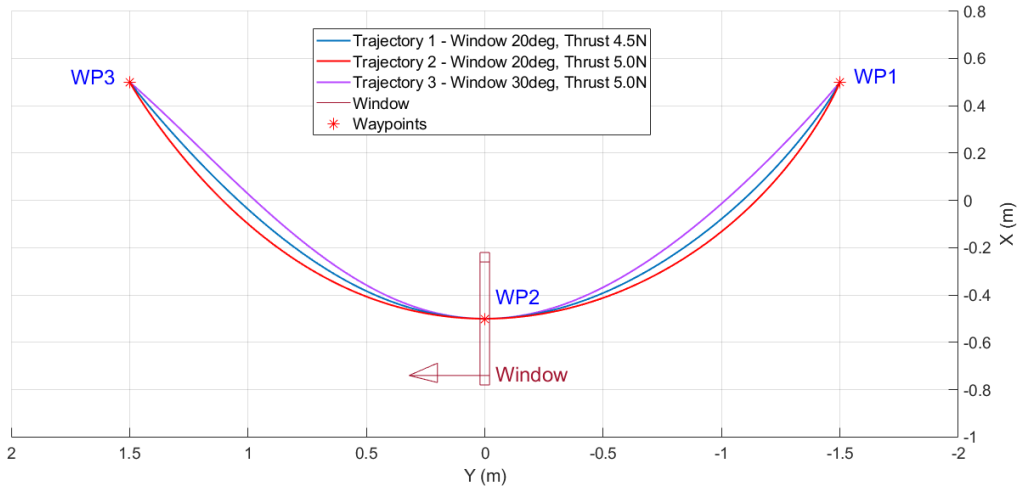

Figure 7.32. Comparison of window passing trajectories.

We conduct three flight tests for each trajectory. The test results are tabulated in Table 7.3.

Table 7.3. Window passing test results

Test No.	Window Angle (deg)	Specified Max Thrust (N)	Seg. 1 Time (s)	Seg. 2 Time (s)	Total Time (s)	Estimated Max RPM	Actual Max RPM	Max RPM Difference (%)
1-1	20	4.5	1.69	1.72	3.41	6315	6557.5	3.84
1-2	20	4.5	1.69	1.72	3.41	6315	6524.3	3.31
1-3	20	4.5	1.69	1.72	3.41	6315	6438.1	1.95
2-1	20	5.0	1.48	1.50	2.98	6688	6649.2	-0.58
2-2	20	5.0	1.48	1.50	2.98	6688	6659.6	-0.42
2-3	20	5.0	1.48	1.50	2.98	6688	6652.3	-0.53
3-1	30	5.0	1.55	1.59	3.14	6688	6707.1	0.29
3-2	30	5.0	1.55	1.59	3.14	6688	6720.5	0.49
3-3	30	5.0	1.55	1.59	3.14	6688	6778.8	1.36

From the test results we can see that the differences between the actual maximum RPM and the estimation are all **within 4%**. This further proves **the practical value of our trajectory optimization framework even in constrained cases such as narrow window passing**. By specifying the same maximum rotor thrust in the trajectory planning, we can accomplish the passing through narrow windows with different tilted angles using almost the same actual maximum rotor RPM. For the same scenario, **we can decide how aggressive (risky) we are willing to go by specifying corresponding maximum individual rotor thrust or RPM**.

The video of these narrow window passing flights is available at https://youtu.be/-P_Vcn1ulYs.

Chapter 8

Summary and Closing Remarks

In the first phase of this research, we successfully developed the complete optimization framework to find the most aggressive trajectory for a quadcopter based on the vehicle model, the waypoints, the required vehicle maneuver at waypoints, and the specified maximum rotor thrust. We achieved this by utilizing multi-segment polynomial trajectory optimization and differential flatness based inverse dynamics analysis. With this framework, we are able to find the optimal polynomial trajectory and the corresponding segment time allocation for a specified scenario and a desired aggressiveness level. Unlike other methods, we actually track the maximum individual rotor thrust required during trajectory optimization. This approach allows us to precisely manage the level of aggressiveness, ensuring it is not overly risky while still pursuing aggressiveness and, at the same time, avoiding excessive conservatism.

With this framework, we can also tell if the given scenario is beyond the capability of the given quadcopter by examining whether the solution exists. Another possible application is that we can use the algorithm to evaluate the minimum required rotor thrust to achieve the scenario. Instead of tracking the maximum force, we can also use the algorithm to track the total time. That is, given a desired flight time, we can find the optimal trajectory that has the minimum required rotor thrust. In the case when there is no vehicle heading requirement, we also provide a yaw trajectory optimization method in our framework to further improve the aggressive performance.

For clarity and simplicity, we adopted a commonly used simplified quadcopter dynamic model in the first phase. However, in aggressive flight some ignored effects could

become significant, and it could be unrealistic to find the optimal trajectory based on the simplified dynamic model. Therefore, in the second phase we incorporate additional effects into the quadcopter dynamic model.

During the second phase, we achieved the full compensation for aerodynamic, gyroscopic, and rotor rolling effects in both optimal constrained trajectory planning and geometric control trajectory tracking. Firstly, we integrated all those effects into the inverse dynamics analysis. We developed the full inverse dynamics derivation to incorporate the rotor drag and overcome the difficulty of unknown rotor speeds by proposing a numerical iterative method. Then, we integrate the aerodynamic drag into the trajectory derivative constraints for window passing to compensate the attitude change due to the drag force. This is also an iterative process and the modified inverse dynamics analysis is utilized to solve rotor speeds at each window waypoint iteratively. This framework shows how the differential flatness property can be preserved while introducing all these additional effects. This is important and can be applied to any differential flatness based trajectory planning or tracking control.

In all of our test cases, the iterative method practically solved the rotor speeds which are essential for the estimation of all the additional effects. The case study provided in Chapter 3 shows how the drag induced, gyroscopic and rotor rolling torques could be significant in a general aggressive trajectory flight and should not simply be neglected. In the window passing trajectory planning, we observed how the compensation for aerodynamic drag could significantly change the resultant aggressive trajectory. In the geometric control trajectory tracking simulations, we observed how the compensations improved the tracking performance step by step and that these compensations are all important for the accurate aggressive trajectory tracking.

Next we moved on to high fidelity simulations. With RotorS/Gazebo simulation, we successfully implemented the proposed trajectory tracking controller in the realistic virtual

environment and achieved precise tracking control. We also verified that our trajectory synthesis can be applied to generic multi-copters with minor modifications to the vehicle control allocation model by using the built-in AscTec Firefly hexacopter model. We also share our experience in establishing the comprehensive PX4 SITL (Software in the Loop) simulation environment. This a crucial step towards the successful flight tests because it allows for thorough testing of system integration, data communication, controller implementation, and even the detailed flight test procedure before conducting actual flights. Additionally, we present a comparison between the simulation results and the data obtained from actual flight tests to validate the degree of similarity.

Precise identification of system model parameters is crucial for both trajectory optimization and trajectory tracking control. We obtained accurate mass and inertia properties of the vehicle through detailed SolidWorks CAD modeling. Important rotor properties are obtained through tests on a rotor test bench. A series of flight tests is designed and conducted to identify all the parameters in our quadcopter dynamic model through maximum likelihood estimation. Additional yaw control effort has been discovered in these flight tests and finally inferred to be caused by slight rotor misalignment. We also identified this misalignment angle through maximum likelihood estimation and added this effect into our dynamic model. This is an important factor to the trajectory optimization because it affects individual rotor thrust and RPM.

Finally we conducted actual flight tests in a Vicon motion capture system equipped environment. The trajectory tracking control is based on the native PX4 flight controller and one companion trajectory controller running on the on-board computer to offer trajectory information and additional feedforward and compensation terms that were proposed in the second phase of this research to the flight controller. We tuned the rate and attitude controller gains for fast tracking. We overcome the issue of conservative attitude set-point mapping and attitude tracking delay caused by rotor time constant by pre-sending scaled

feedforward acceleration commands with a fixed lead time. Finally, the precise trajectory tracking is achieved, and the feasibility of the optimal aggressive constrained trajectory is confirmed through actual narrow window passing flight tests. The actual RPM data in the flight tests also verifies the specification of maximum rotor thrust in our framework and proves the practical value of our trajectory optimization framework and inverse dynamics analysis. The estimated RPM would not be so close to the actual flight data if the additional rotor misalignment identified is not considered in our dynamic model.

In fact, based on our framework, we can also modify the trajectory optimization to track maximum individual rotor RPM instead of maximum thrust. This could be more practical in real applications.

APPENDIX A

PX4 Parameters Used in This Research

In this research, we use PX4 firmware base version 1.13.0. Detailed parameter settings for EKF2, sensor and controller are tabulated below:

Table A.1: PX4 EKF2 parameter settings

Parameter	**Value**	**Parameter**	**Value**
EKF2_ABIAS_INIT	0.200000003	EKF2_MAG_E_NOISE	0.001
EKF2_ABL_ACCLIM	25	EKF2_MAG_GATE	3
EKF2_ABL_GYRLIM	3	EKF2_MAG_NOISE	0.050000001
EKF2_ABL_LIM	0.400000006	EKF2_MAG_TYPE	0
EKF2_ABL_TAU	0.5	EKF2_MAG_YAWLIM	0.25
EKF2_ACC_B_NOISE	0.003	EKF2_MCOEF	0.150000006
EKF2_ACC_NOISE	0.349999994	EKF2_MIN_RNG	0.100000001
EKF2_AID_MASK	24	EKF2_MULTI_IMU	2
EKF2_ANGERR_INIT	0.100000001	EKF2_MULTI_MAG	2
EKF2_ARSP_THR	0	EKF2_NOAID_NOISE	10
EKF2_ASPD_MAX	20	EKF2_NOAID_TOUT	5000000
EKF2_ASP_DELAY	100	EKF2_OF_DELAY	20
EKF2_AVEL_DELAY	5	EKF2_OF_GATE	3
EKF2_BARO_DELAY	0	EKF2_OF_N_MAX	0.5
EKF2_BARO_GATE	5	EKF2_OF_N_MIN	0.150000006
EKF2_BARO_NOISE	3.5	EKF2_OF_POS_X	0
EKF2_BCOEF_X	100	EKF2_OF_POS_Y	0
EKF2_BCOEF_Y	100	EKF2_OF_POS_Z	0
EKF2_BETA_GATE	5	EKF2_OF_QMIN	1

Continued on next page...

Parameter	Value	Parameter	Value
EKF2_BETA_NOISE	0.300000012	EKF2_PCOEF_XN	0
EKF2_DECL_TYPE	7	EKF2_PCOEF_XP	0
EKF2_DRAG_NOISE	2.5	EKF2_PCOEF_YN	0
EKF2_EAS_NOISE	1.399999976	EKF2_PCOEF_YP	0
EKF2_EVA_NOISE	0.050000001	EKF2_PCOEF_Z	0
EKF2_EVP_GATE	5	EKF2_PREDICT_US	10000
EKF2_EVP_NOISE	0.100000001	EKF2_REQ_EPH	3
EKF2_EVV_GATE	3	EKF2_REQ_EPV	5
EKF2_EVV_NOISE	0.100000001	EKF2_REQ_GPS_H	10
EKF2_EV_DELAY	5	EKF2_REQ_HDRIFT	0.100000001
EKF2_EV_NOISE_MD	0	EKF2_REQ_NSATS	6
EKF2_EV_POS_X	0	EKF2_REQ_PDOP	2.5
EKF2_EV_POS_Y	0	EKF2_REQ_SACC	0.5
EKF2_EV_POS_Z	0	EKF2_REQ_VDRIFT	0.200000003
EKF2_FUSE_BETA	0	EKF2_RNG_AID	1
EKF2_GBIAS_INIT	0.100000001	EKF2_RNG_A_HMAX	5
EKF2_GND_EFF_DZ	4	EKF2_RNG_A_IGATE	1
EKF2_GND_MAX_HGT	0.5	EKF2_RNG_A_VMAX	1
EKF2_GPS_CHECK	245	EKF2_RNG_DELAY	5
EKF2_GPS_DELAY	110	EKF2_RNG_GATE	5
EKF2_GPS_POS_X	0	EKF2_RNG_K_GATE	1
EKF2_GPS_POS_Y	0	EKF2_RNG_NOISE	0.100000001
EKF2_GPS_POS_Z	0	EKF2_RNG_PITCH	0

Continued on next page...

Parameter	Value	Parameter	Value
EKF2_GPS_P_GATE	5	EKF2_RNG_POS_X	0
EKF2_GPS_P_NOISE	0.5	EKF2_RNG_POS_Y	0
EKF2_GPS_V_GATE	5	EKF2_RNG_POS_Z	0
EKF2_GPS_V_NOISE	0.300000012	EKF2_RNG_QLTY_T	1
EKF2_GSF_TAS	15	EKF2_RNG_SFE	0.050000001
EKF2_GYR_B_NOISE	0.001	EKF2_SEL_ERR_RED	0.200000003
EKF2_GYR_NOISE	0.015	EKF2_SEL_IMU_ACC	1
EKF2_HDG_GATE	2.599999905	EKF2_SEL_IMU_ANG	15
EKF2_HEAD_NOISE	0.300000012	EKF2_SEL_IMU_RAT	7
EKF2_HGT_MODE	3	EKF2_SEL_IMU_VEL	2
EKF2_IMU_POS_X	0	EKF2_SYNT_MAG_Z	0
EKF2_IMU_POS_Y	0	EKF2_TAS_GATE	3
EKF2_IMU_POS_Z	0	EKF2_TAU_POS	0.25
EKF2_MAG_ACCLIM	0.5	EKF2_TAU_VEL	0.25
EKF2_MAG_B_NOISE	1.00E-04	EKF2_TERR_GRAD	0.5
EKF2_MAG_CHECK	1	EKF2_TERR_MASK	3
EKF2_MAG_DECL	0	EKF2_TERR_NOISE	5
EKF2_MAG_DELAY	0	EKF2_WIND_NOISE	0.100000001

Table A.2: PX4 sensor parameter settings

Parameter	**Value**	**Parameter**	**Value**
SENS_BARO_QNH	1013.25	SENS_EN_SF1XX	0
SENS_BARO_RATE	20	SENS_EN_SHT3X	0
SENS_BOARD_ROT	0	SENS_EN_SPL06	0
SENS_BOARD_X_OFF	1.35320282	SENS_EN_THERMAL	-1
SENS_BOARD_Y_OFF	-0.783595502	SENS_EN_TRANGER	0
SENS_BOARD_Z_OFF	0	SENS_EN_VL53L1X	0
SENS_CM8JL65_CFG	0	SENS_EXT_I2C_PRB	1
SENS_DPRES_OFF	0	SENS_FLOW_MAXHGT	3
SENS_EN_ADIS164X	0	SENS_FLOW_MAXR	2.5
SENS_EN_BATT	0	SENS_FLOW_MINHGT	0.699999988
SENS_EN_ETSASPD	0	SENS_FLOW_ROT	6
SENS_EN_IRLOCK	0	SENS_GPS_MASK	0
SENS_EN_LL40LS	0	SENS_GPS_PRIME	0
SENS_EN_MB12XX	0	SENS_GPS_TAU	10
SENS_EN_MPDT	0	SENS_IMU_AUTOCAL	1
SENS_EN_MS4515	0	SENS_IMU_MODE	0
SENS_EN_MS4525DO	0	SENS_LEDDAR1_CFG	0
SENS_EN_MS5525DS	0	SENS_MAG_AUTOCAL	0
SENS_EN_PAA3905	0	SENS_MAG_MODE	0
SENS_EN_PAW3902	0	SENS_MAG_RATE	15
SENS_EN_PCF8583	1	SENS_SF0X_CFG	0

Continued on next page...

Parameter	Value	Parameter	Value
SENS_EN_PGA460	0	SENS_TFLOW_CFG	0
SENS_EN_PMW3901	0	SENS_TFMINI_CFG	0
SENS_EN_PX4FLOW	0	SENS_ULAND_CFG	0
SENS_EN_SDP3X	0		
IMU_ACCEL_CUTOFF	30	IMU_GYRO_FFT_MAX	150
IMU_DGYRO_CUTOFF	30	IMU_GYRO_FFT_MIN	30
IMU_GYRO_CAL_EN	1	IMU_GYRO_FFT_SNR	10
IMU_GYRO_CUTOFF	40	IMU_GYRO_NF0_BW	20
IMU_GYRO_DNF_BW	15	IMU_GYRO_NF0_FRQ	0
IMU_GYRO_DNF_EN	0	IMU_GYRO_NF1_BW	20
IMU_GYRO_DNF_HMC	3	IMU_GYRO_NF1_FRQ	0
IMU_GYRO_FFT_EN	1	IMU_GYRO_RATEMAX	800
IMU_GYRO_FFT_LEN	512	IMU_INTEG_RATE	200

Table A.3: PX4 controller parameter settings

Parameter	**Value**	**Parameter**	**Value**
MC_ACRO_EXPO	0.689999998	MC_PITCHRATE_P	0.25
MC_ACRO_EXPO_Y	0.689999998	MC_PITCH_P	10
MC_ACRO_P_MAX	720	MC_PR_INT_LIM	0.300000012
MC_ACRO_R_MAX	720	MC_ROLLRATE_D	0.004
MC_ACRO_SUPEXPO	0.699999988	MC_ROLLRATE_FF	0
MC_ACRO_SUPEXPOY	0.699999988	MC_ROLLRATE_I	0.150000006
MC_ACRO_Y_MAX	540	MC_ROLLRATE_K	1
MC_AIRMODE	0	MC_ROLLRATE_MAX	220
MC_AT_APPLY	1	MC_ROLLRATE_P	0.25
MC_AT_EN	1	MC_ROLL_P	10
MC_AT_RISE_TIME	0.140000001	MC_RR_INT_LIM	0.300000012
MC_AT_START	0	MC_YAWRATE_D	0
MC_AT_SYSID_AMP	0.699999988	MC_YAWRATE_FF	0
MC_BAT_SCALE_EN	0	MC_YAWRATE_I	0.100000001
MC_MAN_TILT_TAU	0	MC_YAWRATE_K	1
MC_PITCHRATE_D	0.004	MC_YAWRATE_MAX	200
MC_PITCHRATE_FF	0	MC_YAWRATE_P	0.300000012
MC_PITCHRATE_I	0.150000006	MC_YAW_P	2.799999952
MC_PITCHRATE_K	1	MC_YAW_WEIGHT	0.400000006
MC_PITCHRATE_MAX	220	MC_YR_INT_LIM	0.300000012
MPC_ACC_DOWN_MAX	3	MPC_TKO_RAMP_T	3

Continued on next page...

Parameter	Value	Parameter	Value
MPC_ACC_HOR	3	MPC_TKO_SPEED	1.5
MPC_ACC_HOR_MAX	5	MPC_USE_HTE	1
MPC_ACC_UP_MAX	4	MPC_VELD_LP	5
MPC_ALT_MODE	0	MPC_VEL_MANUAL	10
MPC_HOLD_DZ	0.100000001	MPC_XY_CRUISE	5
MPC_HOLD_MAX_XY	0.800000012	MPC_XY_ERR_MAX	2
MPC_HOLD_MAX_Z	0.600000024	MPC_XY_MAN_EXPO	0.600000024
MPC_JERK_AUTO	4	MPC_XY_P	0.949999988
MPC_JERK_MAX	8	MPC_XY_TRAJ_P	0.5
MPC_LAND_ALT1	10	MPC_XY_VEL_ALL	-10
MPC_LAND_ALT2	5	MPC_XY_VEL_D_ACC	0.200000003
MPC_LAND_ALT3	1	MPC_XY_VEL_I_ACC	0.400000006
MPC_LAND_CRWL	0.300000012	MPC_XY_VEL_MAX	12
MPC_LAND_RC_HELP	0	MPC_XY_VEL_P_ACC	1.799999952
MPC_LAND_SPEED	0.699999988	MPC_YAWRAUTO_MAX	45
MPC_MANTHR_MIN	0.079999998	MPC_YAW_EXPO	0.600000024
MPC_MAN_TILT_MAX	35	MPC_YAW_MODE	0
MPC_MAN_Y_MAX	150	MPC_Z_MAN_EXPO	0.600000024
MPC_MAN_Y_TAU	0.079999998	MPC_Z_P	1
MPC_POS_MODE	4	MPC_Z_VEL_ALL	-3
MPC_SPOOLUP_TIME	1	MPC_Z_VEL_D_ACC	0
MPC_THR_CURVE	0	MPC_Z_VEL_I_ACC	2
MPC_THR_HOVER	0.5	MPC_Z_VEL_MAX_DN	1

Continued on next page...

Parameter	Value	Parameter	Value
MPC_THR_MAX	1	MPC_Z_VEL_MAX_UP	3
MPC_THR_MIN	0.119999997	MPC_Z_VEL_P_ACC	4
MPC_THR_XY_MARG	0.300000012	MPC_Z_V_AUTO_DN	1
MPC_TILTMAX_AIR	45	MPC_Z_V_AUTO_UP	3
MPC_TILTMAX_LND	12		

References

[1] E. Tal and S. Karaman, "Accurate tracking of aggressive quadrotor trajectories using incremental nonlinear dynamic inversion and differential flatness," IEEE Transactions on Control Systems Technology, 2020.

[2] F. Gao, W. Wu, J. Pan, B. Zhou, and S. Shen, "Optimal time allocation for quadrotor trajectory generation," in IEEE/RSJ International Conference on Intelligent Robots and Systems, Madrid, 2018.

[3] G. Yu, D. Cabecinhas, R. Cunha, and C. Silvestre, "Quadrotor trajectory generation and tracking for aggressive maneuvers with attitude constraints," IFAC-PapersOnLine, vol. 52-12, 2019.

[4] Z. Wang, X. Zhou, C. Xu, J. Chu, and F. Gao, "Alternating minimization based trajectory generation for quadrotor aggressive flight," IEEE Robotics and Automation Letters, vol. 5, no. 3, p. 4836–4843, 2020.

[5] G. Ryou, E. Tal, and S. Karaman, "Multi-fidelity black-box optimization for time-optimal quadrotor maneuvers," The International Journal of Robotics Research, July 2021.

[6] P. Foehn, A. Romero, and D. Scaramuzza, "Time-optimal planning for quadrotor waypoint flight," Science Robotics, vol. 6, no. 56, 2021.

[7] Z. Wang, X. Zhou, C. Xu, and F. Gao, "Geometrically constrained trajectory optimization for multicopters," 2021.

[8] D. Falanga, E. Mueggler, M. Faessler, and D. Scaramuzza, "Aggressive quadrotor flight through narrow gaps with onboard sensing and computing using active vi-

sion," 2017 IEEE International Conference on Robotics and Automation (ICRA), p. 5774–5781, 2017.

[9] A. Chakravarthy and D. Ghose, "Guidance for precision three-dimensional maneuvers through orifices using safe-passage cones," Journal of Guidance, Control, and Dynamics, vol. 39, no. 6, pp. 1325–1341, 2016.

[10] W. Zuo, K. Dhal, A. Keow, A. Chakravarthy, and Z. Chen, "Model-based control of a robotic fish to enable 3d maneuvering through a moving orifice," IEEE Robotics and Automation Letters, vol. 5, no. 3, pp. 4719–4726, 2020.

[11] G. Loianno, C. Brunner, G. McGrath, and V. Kumar, "Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and imu," IEEE Robotics and Automation Letters, vol. 2, no. 2, pp. 404–411, 2017.

[12] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for quadrotor flight," in International Conference on Robotics and Automation, 2013.

[13] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in International Conference on Robotics and Automation. IEEE, 2011.

[14] D. W. Mellinger, "Trajectory generation and control for quadrotors," Publicly Accessible Penn Dissertations, vol. 547, 2012.

[15] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in Robotics Research, M. Inaba and P. Corke, Eds. Cham: Springer International Publishing, 2016, pp. 649–666.

[16] T. Lee, M. Leok, and N. McClamroch, "Geometric tracking control of a quadrotor uav on se(3)," 49th IEEE Conference on Decision and Control, 2010.

[17] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in Robotics research. Springer, 2016, pp. 649–666.

[18] Z. Wang, X. Zhou, C. Xu, and F. Gao, "Geometrically constrained trajectory optimization for multicopters," arXiv preprint arXiv:2103.00190, 2021.

[19] E. Tal and S. Karaman, "Accurate tracking of aggressive quadrotor trajectories using incremental nonlinear dynamic inversion and differential flatness," IEEE Transactions on Control Systems Technology, vol. 29, no. 3, pp. 1203–1218, 2020.

[20] J. Jia, K. Guo, X. Yu, W. Zhao, and L. Guo, "Accurate high-maneuvering trajectory tracking for quadrotors: A drag utilization method," IEEE Robotics and Automation Letters, vol. 7, no. 3, pp. 6966–6973, 2022.

[21] Z. Liu and L. Cai, "Large-angle and high-speed trajectory tracking control of a quadrotor uav based on reachability," in 2022 International Conference on Robotics and Automation (ICRA). IEEE, 2022, pp. 1983–1988.

[22] S. Martini, S. Sönmez, A. Rizzo, M. Stefanovic, M. J. Rutherford, and K. P. Valavanis, "Euler-lagrange modeling and control of quadrotor uav with aerodynamic compensation," in 2022 International Conference on Unmanned Aircraft Systems (ICUAS). IEEE, 2022, pp. 369–377.

[23] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories," IEEE Robotics and Automation Letters, vol. 3, no. 2, pp. 620–626, 2017.

[24] M. Bangura, M. Melega, R. Naldi, and R. Mahony, "Aerodynamics of rotor blades for quadrotors," arXiv preprint arXiv:1601.00733, 2016.

[25] P. Martin and E. Salaün, "The true role of accelerometer feedback in quadrotor control," in 2010 IEEE international conference on robotics and automation. IEEE, 2010, pp. 1623–1629.

[26] J. Svacha, K. Mohta, and V. Kumar, "Improving quadrotor trajectory tracking by compensating for aerodynamic effects," in 2017 international conference on unmanned aircraft systems (ICUAS). IEEE, 2017, pp. 860–866.

[27] S. Bouabdallah, A. Noth, and R. Siegwart, "Pid vs lq control techniques applied to an indoor micro quadrotor," in 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566), vol. 3. IEEE, 2004, pp. 2451–2456.

[28] R. Niemiec and F. Gandhi, "Effects of inflow model on simulated aeromechanics of a quadrotor helicopter," in 72nd Annual Forum of the American Helicopter Society International, 2016.

[29] S. Bouabdallah and R. Siegwart, "Full control of a quadrotor," in 2007 IEEE/RSJ international conference on intelligent robots and systems. Ieee, 2007, pp. 153–158.

[30] W. Giernacki, J. Gośliński, J. Goślińska, T. Espinoza-Fraire, and J. Rao, "Mathematical modeling of the coaxial quadrotor dynamics for its attitude and altitude control," Energies, vol. 14, no. 5, p. 1232, 2021.

[31] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "Rotors—a modular gazebo mav simulator framework," in Robot operating system (ROS). Springer, 2016, pp. 595–625.

[32] J.-M. Kai, G. Allibert, M.-D. Hua, and T. Hamel, "Nonlinear feedback control of quadrotors exploiting first-order drag effects," IFAC-PapersOnLine, vol. 50, no. 1, pp. 8189–8195, 2017.

[33] A. Kolaei, D. Barcelos, and G. Bramesfeld, "Experimental analysis of a small-scale rotor at various inflow angles," International Journal of Aerospace Engineering, vol. 2018, 2018.

[34] G. Hoffmann, H. Huang, S. Waslander, and C. Tomlin, "Quadrotor helicopter flight dynamics and control: Theory and experiment," in AIAA guidance, navigation and control conference and exhibit, 2007, p. 6461.

[35] T.-L. Liu and K. Subbarao, "Optimal aggressive constrained trajectory synthesis and control for multi-copters," Aerospace, vol. 9, no. 6, 2022. [Online]. Available: https://www.mdpi.com/2226-4310/9/6/281

[36] D. P. Bertsekas, Nonlinear Programming. Belmont, MA: Athena Scientific, 1999.

[37] S. Boyd and L. Vandenberghe, Convex Optimization. Cambridge: Cambridge University Press, 2004.

[38] Q. Quan, Introduction to Multicopter Design and Control. Singapore: Springer, 2017.

[39] D. Mellinger, Trajectory generation and control for quadrotors. University of Pennsylvania, 2012.

[40] Q. Quan, Introduction to multicopter design and control. Springer, 2017.

[41] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se (3)," in 49th IEEE conference on decision and control (CDC). IEEE, 2010, pp. 5420–5425.

[42] "Flame wheel arf kit." [Online]. Available: https://www-v1.dji.com/flame-wheel-arf/feature.html

[43] "E305 tuned propulsion system." [Online]. Available: https://www-v1.dji.com/e305.html

[44] "X4rsb." [Online]. Available: https://www.frsky-rc.com/product/x4rsb/

[45] "Sik telemetry radio v3." [Online]. Available: https://holybro.com/products/sik-telemetry-radio-v3

[46] "Tfrpm01: Drone rpm tachometer sensor." [Online]. Available: https://www.tindie.com/products/thunderfly/tfrpm01-drone-rpm-tachometer-sensor/

[47] "Tfi2cadt01: Pixhawk i2c address translator." [Online]. Available: https://www.tindie.com/products/thunderfly/tfi2cadt01-pixhawk-i2c-address-translator/

[48] "Thunderfly tfrpm01 revolution counter." [Online]. Available: https://docs.px4.io/main/en/sensor/thunderfly_tachometer.html

[49] "Holybro pixhawk 4 mini." [Online]. Available: https://docs.px4.io/main/en/flight_controller/pixhawk4_mini.html

[50] "Get started with drone development." [Online]. Available: https://px4.io/

[51] A. R. Godbole, "Nonlinear control of unmanned aerial vehicles with cable suspended payloads," Ph.D. dissertation, The University of Texas at Arlington, 2019.

[52] "Raspberry pi 3 model b." [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-3-model-b/

[53] "mavros." [Online]. Available: http://wiki.ros.org/mavros

[54] "Qgroundcontrol." [Online]. Available: http://qgroundcontrol.com/

[55] "Taranis q x7." [Online]. Available: https://www.frsky-rc.com/product/taranis-q-x7-2/

[56] "What is motion capture." [Online]. Available: https://www.vicon.com/

[57] "vicon bridge." [Online]. Available: http://wiki.ros.org/vicon_bridge

[58] "Rotors simulator." [Online]. Available: https://github.com/ethz-asl/rotors_simulator

[59] "Simulation." [Online]. Available: https://docs.px4.io/main/en/simulation/

[60] A. Lorente, "Dji f450 quadcopter drone." [Online]. Available: https://grabcad.com/library/dji-f450-quadcopter-drone-1

[61] A. M. Martinez, "Onboard payload mass estimation and electric propulsion modeling for multicopters with application in unmanned aerial systems," Master's thesis, The University of Texas at Arlington, 2019.

[62] J. L. Crassidis and J. L. Junkins, Optimal estimation of dynamic systems. Chapman and Hall/CRC, 2004.

[63] K. W. Iliff, "Parameter estimation for flight vehicles," Journal of Guidance, Control, and Dynamics, vol. 12, no. 5, pp. 609–622, 1989.

Biographical Statement

Tsung-Liang Liu was born in Penghu, Taiwan, in 1979. He received his Bachelor of Engineering degree in Aerospace Engineering from Tamkang University, Taiwan, in 2002. He earned his Master of Science degree in Aerospace Engineering from National Cheng Kung University, Taiwan, in 2004. After graduation, Tsung-Liang has been working in Taiwan National Chung-Shan Institute of Science and Technology (NSCIST) as a simulator software research and development engineer. In fall 2019, Tsung-Liang started his Ph.D. program in Aerospace Engineering at the University of Texas at Arlington (UTA) which is sponsored by NCSIST. He joined the Aerospace Systems Laboratory in UTA in spring 2020 and has been working on multi-copter trajectory optimization, dynamic system modeling and simulation, and trajectory tracking flight control. After earning his Ph.D., Tsung-Liang will return to Taiwan and continue serving in NCSIST.