Intelligent Guidance, Navigation, and Control of Multi-Agent UASs with Validation and Verification

By

A Ram Kim

Submitted to the Department of Aerospace Engineering and the
Graduate Faculty of the University of Kansas
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Shawn Keshmiri, Chairperson

Weizheng Huang, Co-adviser

Committee members

Mark Ewing

Richard Hale

Saeed Farohki

Date defended: May 07, 2018

The dissertation Committee for A Ram Kim certifies
that this is the approved version of the following dissertation :

Intelligent Guidance, Navigation, and Control of Multi-Agent UASs with Validation and Verification

Shawn Keshmiri, Chairperson

Date approved: May 11, 2018

Abstract

Following the exponential growth in the usage of unmanned aerial systems (UASs) across the Aerospace Industry, more intelligent and robust guidance, navigation, and control algorithms are vital to cope with increasing levels of mission complexity. Additionally, many unmanned aerial operations require large payloads and long endurance such as extended reconnaissance, large-scale search and rescue and fine resolution terrain mapping. However, the stringent payload of a single agent or small UASs reduces their overall practicality and effectiveness. My research aims to address these inherent limitations of small UASs with a swarm by holding the required formation in order to distribute tasks and payload among multiple UASs. The goal of this research is to overcome the challenges of operating multi-agent systems by developing phasic navigation and guidance algorithms. Aircraft dynamics and their interactions with surrounding agents are highly nonlinear, which makes autonomous formation flight very sensitive to aircraft initial conditions. The phasic navigation algorithms are proposed and consist of hybrid mathematical approaches: Frenet-Serret curvature control, Hungarian algorithm and moving mesh methods. At the first phase, the curvature control alleviates the sensitivity to initial conditions of multi-agent UASs in unstructured environments by matching agents' heading angle to the united direction. A variation of Hungarian algorithm is implemented with a moving virtual terminal to assign each agent to the formation position. In the second phase of navigation, the moving mesh methods are applied for holding the formation by defining the outer agents' position for the boundary condition. The

significance of the moving mesh methods is a scalability and a inherent inter-collision avoidance. Due to the profound difference between the longitudinal and lateral-directional motion of a fixed-wing aircraft, a multi-scale moving point guidance algorithm has been designed to create the separate virtual reference points in the longitudinal and lateral-direction planes for the first time. This method has been shown to greatly reduce tracking oscillations and improve the overall tracking quality and coherency of the formation. Monte Carlo simulations are performed to ensure the stability and robustness of implementing proposed algorithms through an essentially exhaustive search. A broad range of random initial conditions have been used to validate the effectiveness of guidance, navigation, and control algorithms. Another unique contribution of this work is the validation and verification of proposed algorithms by the hardware-in-the-loop testbed and the numerous flight tests. The hardware-in-the-loop testbed is designed to test the avionics and communication before the flight test by simulating onboard 6-degrees of freedom nonlinear equations of motion. Over one hundred flight tests have been conducted using three distinct aircraft platforms between 2016 and 2018 to validate the fundamental building blocks of this architecture. In summary, this dissertation provides a conceptual and practical foundation for guidance, navigation, and control of multi-agent cooperative/collaborative UASs by unique approaches.

Acknowledgements

I want to thank by my heart to my own advisor, Dr.Shawn Keshmiri. I met Dr.Keshmiri at Flight Dynamics class in 2011 and he mold my soul and brain to grow as a scholar and adult. I cannot express my appreciate about his guidance and support enough. Biggest aspect that I have learned from Dr.Keshmiri is that the depth of knowledge and the quality of work I have done are only way that I can be successful for my future.

I also thank Dr.Saeed Farohki for his precious time and support for me. When I decided to continue my education for PhD program, Dr.Farokhi recommended for me to get into PhD program. I guess that he foresee my future and gave me courage to finish PhD program. Thank you so much, Dr. Farohki.

I want to appreciate Dr.Mark Ewing for supporting me from the professional level to the personal level. I admire Dr.Ewing since I met him at the aircraft structure course in 2011. The reason why I admire him is very personal since I wanted to be a fighter pilot and failed to get in Korea Airforce Academy and Dr.Ewing already did my dream job (He graduated US Airforce Academy). Then, Dr.Ewing has a lot of wisdoms for flight tests and thank you for sharing your wisdoms with our research team.

Dr.Hale, thank you so much for your thorough comments for my work from your class report to my dissertation. Anytime when I obtain your comments, I could grow and learn a lesson. I will keep the work ethics as you are keeping it.

Dr.Huang, I met you since 2013 and thank you so much for your support during

my PhD program as a co-advisor. I was honored to work with you for such a profound research area.

Lastly, I thank my family and my husband. Bo Sun Kim (my father) and Mi Suk Choi (my mother) had enormous support from South Korea. My courage and emotional support came from my parents. My brother, Hansol Kim, also supported me a lot to make sure that I can overcome my weakness (e.g., being afraid before I actually do something). My husband, Hyungkook Joo, got married with such a difficult person who get along with (me) and supported my last year of PhD program in Lawrence, KS after his military service is over.

Also, thank you so much for my nerdy colleagues (Aaron Blevins, Daksh Shukla, Hady Benyamen, Grant Godfrey, Thomas Le Pichon and Matt Tener).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Unmanned Aerial Systems (UASs) have been developed intensively in recent decades due to cost advantages, broad applications, saving human pilot's lives, efficiency, and flexibility to accomplish complex missions. Now that UASs have become an integral part of the aerospace industry, new areas of research have emerged. One of the new areas is multi-agent collaborative UASs that have become one of the most active research fields in Aerospace Engineering and Guidance, Navigation, and Control. The necessity of collaborative and cooperative UASs is fueled by increasingly elaborate missions. Examples of complex missions are (1) Search and rescue (2) Reconnaissance (3) Attack and (4) Earth science, see Ref. [24]. However, UASs have stringent payload capabilities when they are considered as individuals. By distributing the payload to multiple agents, multi-agent UASs have the required flexibility and robustness to reduce cost and operate complex mission as the single agent can perform. Instead of operating one Global hawk which has the unit cost of 131.4 million dollars (see Ref. [95]), multiple of cheaper systems can be operated. For example, the formation flight of multiple UASs reduces overall induced drag so that fuel is consumed efficiently. Ref. [69] shown that the maximum drag reduction of two and three agents forma-

tion flight is estimated 30±3% and 40±6%, respectively. In addition, multi-agent systems have a considerably larger coverage footprint when compared to the footprint an individual agent covers in the same amount of time.

Formation and swarm are neither a new nor a man-made phenomena. Many animals and insects perform swarm or aggregate themselves to achieve their goals to survive in wildlife environments (Ref. [10]). Army ants, known as the strongest ants in the world, swarm to defend themselves from predators like snakes, and protect other ants while they carry food (Ref. [65]). In the case of large birds such as swans, cranes, and geese, they fly in a V-shape formation flights. The reason is because a V shape helps save energy (e.g., increased L/D) by using the vortex generated from the front birds so that birds can minimize the collective and required energy to migrate long distances (Ref. [5]). In addition, birds take turns in the leader position and change their position with other adult birds to increase their range and endurance.

The multi agent UASs can take the same advantage of swarm or formation flight as birds do. Swarm is defined as the flight in which agents proceed to an average direction (e.g. the state of consensus) with unsteady proximity between agents. In other words, the state of consensus can change at any time. Swarm is also called aggregation (Ref. [14]). However, formation flights are a special case of swarm flights. While all agents travel toward a desired direction, they should also maintain a certain shape or relative distance between one another. Formation flights can reduce the induced drag and save fuel so that the endurance of UASs can be increased (Ref. [59, 66, 73, 101]). With the increased endurance, UASs can fly longer and finish given complex tasks. For example, more area can be mapped within a shorter time if five UASs collaborate for a reconnaissance mission rather than one UAS operating this task. However, autonomous multi-agent swarm or formation flight is not an analytically trivial nor a practical task to be achieved. According to the report by Ref. [89], the complications and intricacies of autonomous control increase exponentially as a function of mission requirements, see Figure. 1.1.

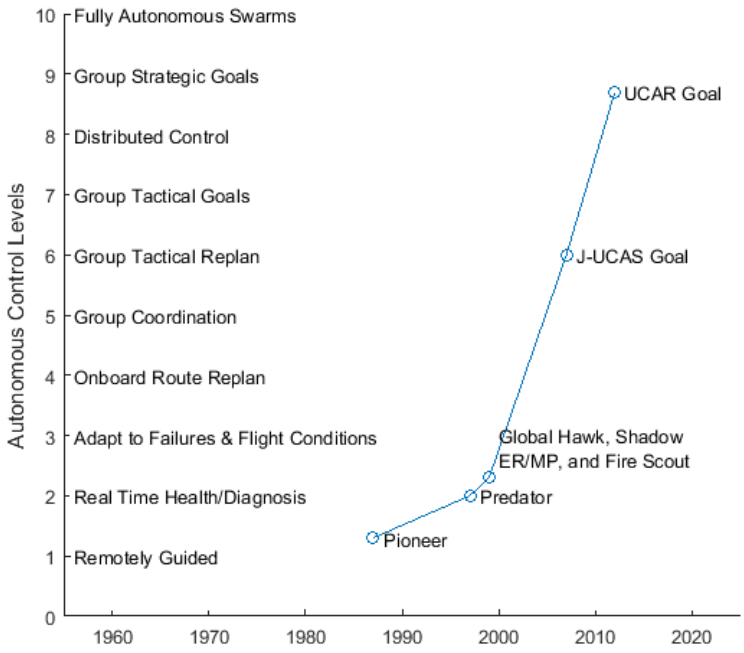


Figure 1.1: Autonomous Control Level based on mission requirements, [89]

1.2 Related Work

1.2.1 Guidance, Navigation, and Control for Multi-agent Systems

Theoretical challenges of self-organizing multi-agent UASs can be summarized as follows: (1) Guidance, navigation, and control (GNC) of swarm by multiple UASs in unstructured environments are rarely considered. These environments have the nonlinearity that makes swarm of UASs sensitive to an agent's initial conditions. (2) Many swarm algorithms have been designed for a small number of agents and lack scalability. (3) Although tools such as convexity and Lyapunov theory can be used for the stability analysis of multi-agent systems, the proof of the stability for large number of UASs is not a trivial task. In many cases, no analytical solution can be found. (4) Achieving close formation by large UASs with high speed and high inertia is very challenging due to the several requirements: following the desired trajectory, maintaining the relative distance between one another and collision avoidance at the same time. From a practical standpoint, difficulties of achieving a coherent

swarm of multi-agent UASs are exacerbated by the following items: communication delays or drop; flights in GPS-denied areas; input delays; GPS position errors; inter-collision; tracking errors; obstacle avoidance and morphing of swarm; and task allocations.

Before the multi-agent systems are considered, all individual autonomous flight systems require three main elements: Guidance, Navigation, and Control (GNC). Guidance provides the state commands such as speed, pitch angle, and roll angle to follow the designated path in 3D space. Navigation is used to plan the path towards the desired position by using the current location and the vehicle's attitude angles. Control executes guidance commands to follow the desired path.

When it comes to multi-agent flights, the navigation algorithm becomes exponentially more challenging. Beside following the desired path, all agents must maintain the formation shape and avoid collision among one another. Many recent swarm navigation algorithms are variations of the Boids model which has been developed (Ref. [82]). The Boids model proposed three simple rules to achieve the swarm: separation, alignment, and cohesion. Separation is the rule to keep agents a certain distance apart to avoid a collision. Alignment implies that all agents' velocity has to converge toward the average direction of all agents. The cohesion indicates that all agents should steer toward the average position of surrounding agents. Once these three conditions are satisfied, the agents aggregate to swarm. Many researchers have proposed the navigation for swarming by using Reynold's model (Ref. [36, 79, 96]).

Formation is the branch of swarm flight with higher level of complexities. The formation flight must not only follow Boids model, but also must hold the formation shape. Many approaches have been proposed for formation flight: (1) the artificial potential field using attractive and repulsive forces, (2) the optimization algorithm using the cost functions, (3) the virtual structure which considers the formation as a rigid body, (4) artificial intelligence, (5) graph theory approaches and (6) the leader-follower approaches.

The most popular method for the multi-agent UASs is the leader-follower approach

(Ref. [3, 7–9, 16–18, 20, 31, 32, 34, 47, 50, 51, 55, 61, 73, 81, 84, 103]). The leader-follower scheme designates one of the agents as the leader and all paths are planned around it. The advantage of this approach is that the leader provides the reference of the formation so the path planning algorithm can be designed in a way that is decoupled from the followers. In other words, the path planning for the leader is simple since the only task for the leader is to follow the desired path. Now, the goal for the follower is only to follow the leader. However, the leader-follower method lacks the robustness towards safeguarding against the failure of the leader or a point failure.

To remedy the drawback of the leader-follower scheme, the virtual leader-follower approach has been proposed. This scheme does not have a physical leader among agents (Ref. [6, 11, 14, 15, 21, 35, 37, 38, 44, 48, 57, 72, 76–78, 90, 104, 106]). Due to the absence of actual or physical leader, the navigation algorithm should be more intelligent compared to the leader-follower scheme. This is because each agent should assess the situation and decide on the proper course of action to fly itself and cooperate with other agents while also avoiding collision with one another.

The artificial potential field is one of the methods that has been most widely used because it provides the inter-collision avoidance among agents. Usually, the attractive and repulsive potential fields are combined together for multi-agent systems to hold agents by the attractive force and prevent a collision with others by the repulsive force. From the perspective of implementing algorithms, the potential field has the advantage of scalability by superposition properties. In addition, the nonlinear potential function guides agents efficiently to desired positions with nonlinearity of the gradient fields. Many researchers use the typical circular potential field which is not proper for high speeds and high inertia vehicles (Ref. [3, 12, 47, 96, 99, 108]). Due to the circular contour of potential fields, the path is not efficient to avoid obstacles and come back to the desired trajectory. Morphing potential field has been developed to improve the efficiency of the potential field and reflect the aircraft and obstacle dynamics (Ref. [88]).

Another popular method for the multi-agent system is the graph theory. A graph is the mathematical concept to visualize the pairing of agents within the group using vertices and edges. If the edge does not have any difference between the direction of edges (e.g., A to B is same as B to A), this type of the graph is called an undirected graph. This graph forces all agents to reach the desired formation at the same time (Ref. [12, 71, 90, 107]). On the other hand, the directed graph (which dictates the direction of the edge) has a lower convergence rate than the undirected graph since only one agent at a time changes its position (Ref. [7, 26, 93]). Many researchers use the graph theory to construct the formation and indicate the communication link between agents (Ref. [14, 21, 39, 103]). Similarly, the virtual structure approach considers the formation as a rigid body. This approach provides formation positions defined geometrically by the desired relative position (Ref. [20, 48, 105]). The disadvantage of these methods is the lack of adaptation towards external disturbances or a collision. To resolve this drawback, many researchers have proposed to combine the graph theory or the virtual structure method with other methods such as the potential field methods and the optimization approaches. Since the graph theory only provides the geometric parameters of the formation agents, it does not have the capability to adapt the path if the agents are too close to one another due to external disturbances. Similarly, the virtual structure approaches provide the desired agent positions for holding the formation but do not produce the attractive forces needed to converge the agents to these positions.

Another approach has been used for multi-agent system navigation based on the optimization using the cost function. Many researchers define a cost function by the distance between agents, trajectories and obstacles (Ref. [15, 54, 91, 100, 102]). The problem with the distance-based cost function is the inability to reflect the aircraft dynamics. The distance between the aircraft and the target does not consider the aircraft's movement. The distance that the aircraft travels would be longer than the Euclidean distance between the aircraft and the target due to the minimum turning radius constraint and the aircraft heading. Similarly, the artificial intelligence approaches are introduced by using the optimization techniques

employing the cost function (Ref. [19, 25]). The setbacks of the artificial intelligence-based approaches are that they need a vast amount of data to have better performance and the users lack the knowledge on how the system makes decisions which renders them unable to control the system.

The common denominator of many existing works is the point mass assumption for the vehicle dynamics which simplifies the guidance and control designs for aircraft (Ref. [6, 9, 14, 16, 47, 55, 71, 86]). The point mass assumption can be held for small mass and slow speeds; however, spatiotemporal requirements for the scientific missions often necessitate the high speeds and high inertia vehicles while holding the close proximity formation. This assumption may result in the dramatic discrepancies in performance between the simulations and actual flights.

Another challenge for the multi-agent system is the assignment sequence and strategy (e.g., where agents should form appropriate formations regardless of their initial positions and states). As the number of agents increases, the combinations of formation positions will be increased dramatically. The typical solutions for the assignment problem are known as the Hungarian and iterative closest point algorithms based on the distance error (Ref. [4, 62, 91, 94, 102]).

Guidance and control are essential elements for reaching the desired path from the current states. Most researchers have not investigated impact of guidance and control for multi-agent agent systems for the actual flight tests, nor have they used Commercially-Off-The-Shelf products since the complexity of designing embraced systems is very high (Ref. [8, 18, 55]). However, the development of entire GNC systems is very important not only for implementing intelligent algorithms reflecting the aircraft dynamics but also for completing multi-agent system designs to verify and validate them by actual flights.

To achieve a coherent multi-agent autonomous system, stability analysis must be performed. Analytically, researchers have shown the stability of swarm in mainly two ways: Lyapunov function (Ref. [47, 57, 81, 99]) and the bounded distance errors among agents and

desired formation positions (Ref. [37, 38, 92]). Exhaustive simulations and the Monte Carlo approach have also been used to show the convergence rate (Ref. [37]).

This research aims to address analytical and practical challenges of autonomous multi-agent UASs formation flights using a phasic navigation. Phasic navigation was introduced for multi-agent systems navigation by Dr. Shawn Keshmiri in 2017. This navigation has phases of guiding agents to assemble the formation from random initial conditions. The two phases are proposed to deal with randomness of initial conditions: assignment with aggregation and holding the formation. The curvature control and Hungarian algorithms are applied for aggregate agents into a united direction and assigning each agent to the proper formation position. The moving mesh methods have been utilized for holding the formation by using an adaptive numerical solution of partial differential equations. The sensitivity of the developed methods has been also investigated on three different UASs in the same UAS classification (less than 100 lb) with a wide range of inertial, aerodynamics, and propulsive characteristics. In the proposed work, the following items have been considered:

- The mass is not assumed as a point or a particle. Six degrees of freedom (6 DoF) aircraft dynamic models are obtained by using engineering level software (e.g. AAA and AVL).

- Proposed guidance, navigation, and control algorithms have been designed for three different UASs ranging from 9 to 73 lbs to investigate the impact on a broad range of UAS's geometry, trim speed, and stability and control derivatives.

- The overall stability of the developed algorithms for guidance, navigation, and control of swarming UASs has been verified by exhaustive search approaches and Monte Carlo simulations. With no prior information regarding initial conditions of the agents in the swarm, this process allows the improvements in the robustness towards random initial positions/states and also enhances the intelligence of the guidance and navigation algorithms to handle formation flights.

- A morphing potential algorithm is implemented to avoid external and inter-collisions on

top of inherent collision avoidance feature in the moving mesh methods. This potential field has a morphed contour field towards the relative velocity direction. Such redundancy has a profound impact on the swarm of high speed and high inertia vehicles where obstacles are avoided as a function of relative speed. The aircraft are able to converge into the desired path much quicker than the classical circular potential field.

- The virtual leader scheme is used to enhance the robustness towards a point failure. Intelligent layers are added for each agent to decide where to go and what to do so that each agent can be independent from the success or failure of other agents.

- A sequential and optimized formation assignment algorithm has been developed with an adaptive cost function to not only account for distances between agents but also their required heading and rotation angle changes. Such adaptive sequential assignments alleviate commanding large heading angles and abrupt control inputs.

1.2.2 Verification and Validation of the Multi-agent Autonomous System

The testbed is an essential tool for validating and refining all developed algorithms. Mainly, testbeds are divided in two parts: Hardware-in-The-Loop (HiTL) testing and actual UAS flight tests. Hardware-in-The-Loop testing has been designed to verify the integration of all algorithms into the embedded system and to simulate actual hardware and software performance using 6 DoF aircraft dynamic model and GNC modules. With given initial conditions and communication to ground station, the GNC algorithm embedded in the autopilot system continuously generates control outputs to accomplish a given mission. For a swarm of UASs, a mesh network must be in place so that agents can wirelessly communicate their dynamic states. This process is a practical way to assess the performance of algorithms and interaction among software, embedded operating systems, and hardware before the actual flight tests. As the integrated systems traverse through the design, development, and im-

plementation processes, a challenge appears in maintaining system robustness with respect to repeatability and reliability in adverse conditions. Therefore, simulating the system in its entirety with minimal assumptions and similarity to pragmatic flight conditions becomes exceedingly crucial and requires extensive tests and validation (Ref. [30, 60, 64]). There is substantial literature addressing the importance of Software-in-The-Loop development and testing (Ref. [22, 33]). HiTL simulation techniques provide one such tool to validate the developed GNC algorithms on the on-board computer systems in the following categories: testing sensor integration, the hardware reliability (e.g., data exchange and communication for multi-agent aircraft systems), see Ref. [70]. Such a tool can help in identifying any underlying deficiencies within the GNC algorithms before performing actual flight tests. Ref. [8] shows that a HiTL testbed for testing multiple fixed-wing UASs to explore higher level path planning and control by using a Commercial-Off-The-Shelf (COTS) autopilot for lower level control. Ref. [40] designed HiTL testbeds for eight UASs using centralized off-board path planning and a COTS autopilot, known as the Piccolo from Cloud Cap Technologies, with 900 MHz communication. Eventually flight tests must be performed to validate the performance of this system of systems. Flight test validation for fixed-wing UASs is relatively rare. Table 1.1 shows the related work for multi-agent flight using fixed wing vehicles.

As shown in Table 1.1, most UASs used in these flight tests have low weights except for the Brumby MK III done by the University of Sydney using only two agents. The payload capacity of most UASs in Table 1.1 is not sufficient to carry heavy equipment (e.g. KU CReSIS dual frequency radar system weighs 6-7 lb) (Ref. [1, 18, 39, 52, 58, 83, 87, 102]). Another common factor for many flight tests is the usage of a launcher for takeoff. Launching UASs is a convenient way to autonomously takeoff in the absence of pilots and it can create consistent UASs initial conditions. In this work, a high fidelity testbed has been designed for Software-in-The-Loop and Hardware-in-The-Loop tests. All algorithms have been also assessed through actual flight tests.

- The pilot manually launches each UASs for takeoff and landing as well as the autopilot

Table 1.1: Specification of UASs for related works

Research Institute	Weight [lbs]	Number of UAVs	Wing Span [ft]	Endurance [min]	Payload [lbs]	Cruise Speed [knots]	Known Initial Condition	Avg. Distance between agent [ft]	Platform
DoD	0.64	103	0.98	20	-	40 -60	O	Unknown	Perdix
ÉPFL, Swiss	0.92	10	2.6	20	0.19	<38	X	Unknown	Swinglet
Tsinghua Univ., China	1.69	119	4.9	25	2.9	21.6	O	Unknown	Skywalker
Bialystok Univ. of Tech, Poland	4	2	5	60	1.5	27	O	Unknown	Unicorn
University of Santa Barbara	4	2	5	60	1.5	27	O	98.4	Unicorn
NPS	5.5	50	4.67	60	1.5	35	O	141.1	Zephyr II
MIT	10.5	2	5.8	40	2	43 -52	X	164	Tower Trainer 60
ONR	13	30	4.8	60	1.98	60	O	Unknown	Coyote
University of Pennsylvania	17	4	12	240	5.5	39	O	Unknown	Stalker
University of Sydney	55	2	9.5	60	28.7	100	X	Unknown	Brumby Mk III

system. Since the pilot has only visual information for the vehicle, it is always very hard to engage the UASs at desired steady state conditions (no accelerations/trimmed condition) or at an exact desired geographic position. This is because the robustness of the initial conditions have been enhanced by intelligent guidance, navigation, and control.

- The Software-in-The-Loop test is performed by using numerous random initial conditions for each agent using Monte Carlo methods to verify stability and robustness of the developed algorithms for swarming UASs.

- In order to verify the integration of algorithms and avionics, the performance and reliability of GNC algorithms and avionics has been tested by Hardware-in-The-Loop test. The Robot Operating System (ROS) is used to implement the sensors and the developed algorithms in a way that they are modular, easy to manage and exchange data among them.

- Actual flight tests have also performed to validate the proposed GNC algorithms. After the flight test has also been performed, the dynamic model and algorithm parameters are

refined and adjusted to improve the effectiveness of the system of systems.

1.3 Outline

Figure 1.2 shows the outline of this dissertation. Chapter 2 presents the algorithm design for guidance, navigation, and control for the autonomous flight system. In Chapter 3, the design and procedure of multi-agent testbed is discussed. In Chapter 4, simulation and flight test results are presented for the validation of the proposed algorithms.

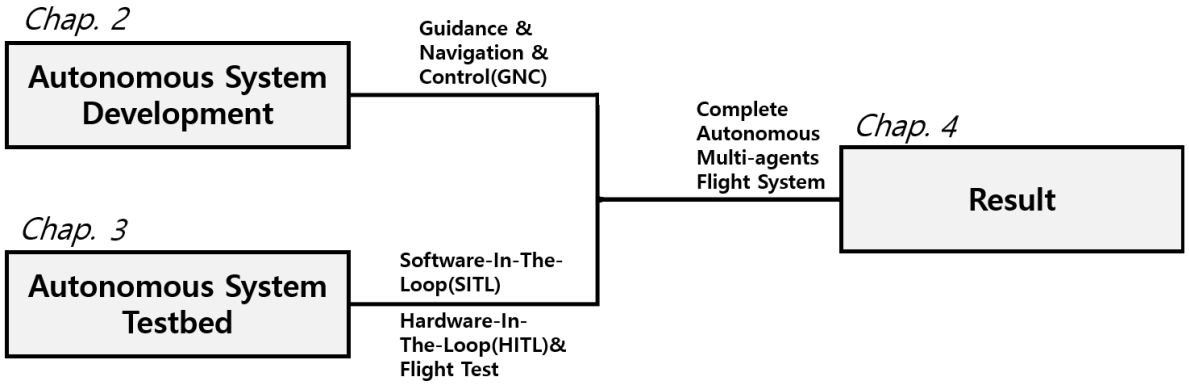


Figure 1.2: Thesis Outline

Chapter 2

Autonomous System Development

Intelligent Guidance, Navigation, and Control (GNC) are required to empower the autonomy in flights for Unmanned Aerial Systems (UASs). These components have roles identical to human pilots in flights. Guidance is to compute the required states (e.g., the pitch angle and speed for the longitudinal motion, the roll angle for the lateral-directional motion) of the aircraft to the desired states from the current states (e.g., position, Euler angles). Control provides how much the control surface deflections are required to achieve the desired states obtained from the guidance. Navigation identifies the path for the aircraft to follow the desired trajectory. In case of the multi-agent system, navigation algorithms have higher complexity than the single agent system since they should provide the path to avoid the other agents, construct the formation by considering other agents' location, and follow the desired trajectory at the same time. In this section, GNC development is discussed for the multi-agent autonomous systems. GNC algorithms have been designed to enhance the intelligence and robustness towards random initial conditions to overcome aforementioned limitations in Section 1.2 and 1.2.2:

- Payload capacity is not sufficient for the scientific missions. Most research has used small platforms with low speeds and small moments of inertia.
- The popular configuration for existing work is a flying wing. Moreover, the maximum

number of flight tested fixed-wings for multi-agent system is two agents.

- Pilots are not involved during takeoff nor engage the autonomous flight mode. However, the flights involving pilots have random initial conditions when (s)he engages the autopilot.

- The classical circular contour of the potential field has been applied for most collision avoidance path planning algorithms. This approach is not adequate for applications related to high moment of inertia and high speed flights.

Figure 2.1 shows the overview of this chapter.

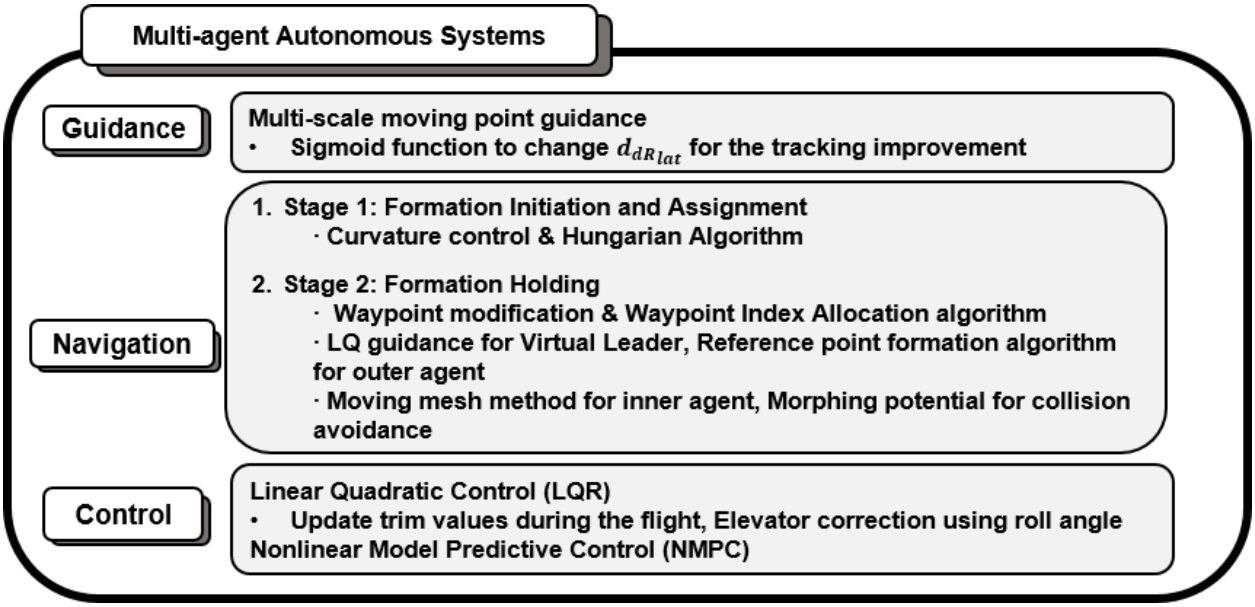


Figure 2.1: Road map for Chapter 2

Multi-scale guidance algorithm with the sigmoid function is newly developed to improve tracking. Individual algorithm for navigation has been developed such as the curvature control, Hungarian algorithm, and the moving mesh method. These algorithms are modified for a multi-agent system in this work. The curvature control algorithm has been combined with 6 DoF aircraft dynamics for the aircraft application. The cost function of Hungarian algorithm has been designed to reflect the aircraft dynamics by adding the difference of heading and rotation angle between the aircraft and the desired formation. In addition, the

moving mesh methods is applied by defining the boundary condition as the outer agents' position. In summary, the different engineering disciplines are modified to utilize as the path planning algorithms for multi-agent systems.

2.1 Guidance

2.1.1 Multi-Scale Moving Point Guidance

Multi-Scale Moving Point (MSMP) guidance has been developed to increase the quality of tracking since formation flights require very tight position control with respect to the desired trajectory for maintaining the shape of a formation. In this work, the state commands for the guidance algorithm are defined as follows: the speed ($V_{T_{cmd}}$), roll (ϕ_{cmd}), pitch (θ_{cmd}), and sideslip (β_{cmd}) angles. The number of agents does not impact the complexity of the guidance algorithm since it is applied to each agent individually. The fundamentals of MSMP guidance algorithm are rooted in a nonlinear guidance law, see Ref. [40]. It has been developed in two dimensional space and can be easily visualized by considering the 'carrot and stick' concept.

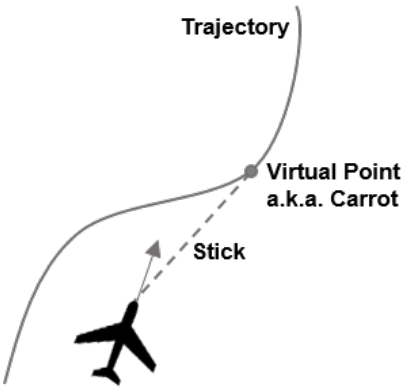


Figure 2.2: Carrot and stick approach

Garcia and Keshmiri expanded a nonlinear guidance law to three dimensional space by applying an identical algorithm on the longitudinal plane, see Ref. [27]. In addition, it became the moving point guidance by updating the desired trajectory segment at each time step, see Fig. 2.3.

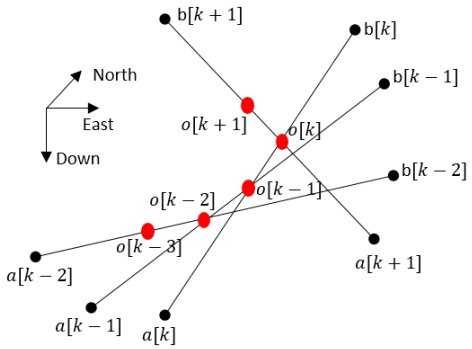


Figure 2.3: Moving points generation scheme, see Ref. [27]

This algorithm is significant for navigation since it allows tracking of an arbitrary shape. However, severe oscillations in the attitude angle commands (ϕ_{cmd} and θ_{cmd}) are caused due to the adaptive gains in calculating the desired roll and pitch angles and identical reference points (carrots) on both longitudinal and lateral-directional planes. In order to reduce the oscillations in commanded angles, MSMP guidance algorithm separates the reference points for longitudinal and lateral-directional planes. The nature of dynamics has been reflected in the guidance algorithm by separating the reference points in longitudinal and lateral-direction plane. For example, I_{xx_B} is usually smaller than I_{yy_B}. This means the time constant of a roll mode is much smaller than the short period. This leads to the fact that the rolling motion can react faster and handle large roll angle commands. The longitudinal motion has the structural constraints via the acceleration in the z body axis. In order to compute the attitude commands (θ_{cmd} and ϕ_{cmd}), the geometric parameters should be defined as shown in Figure. 2.4.

Consider points A and B creating a line segment by expanding the line made by two moving points ($o[k]$ and $o[k-1]$) for the desired trajectory where k is the current time step and $k-1$ is the previous time step. Moving points are computed from the navigation algorithms and defined mathematically in the inertial frame as follows:

$$o[k] : \{\overrightarrow{p}^{o}[k] = p_N^o[k], p_E^o[k], p_H^o[k]; \qquad \overrightarrow{v}^{o}[k] = v_N^o[k], v_E^o[k], v_H^o[k]\} \tag{2.1}$$

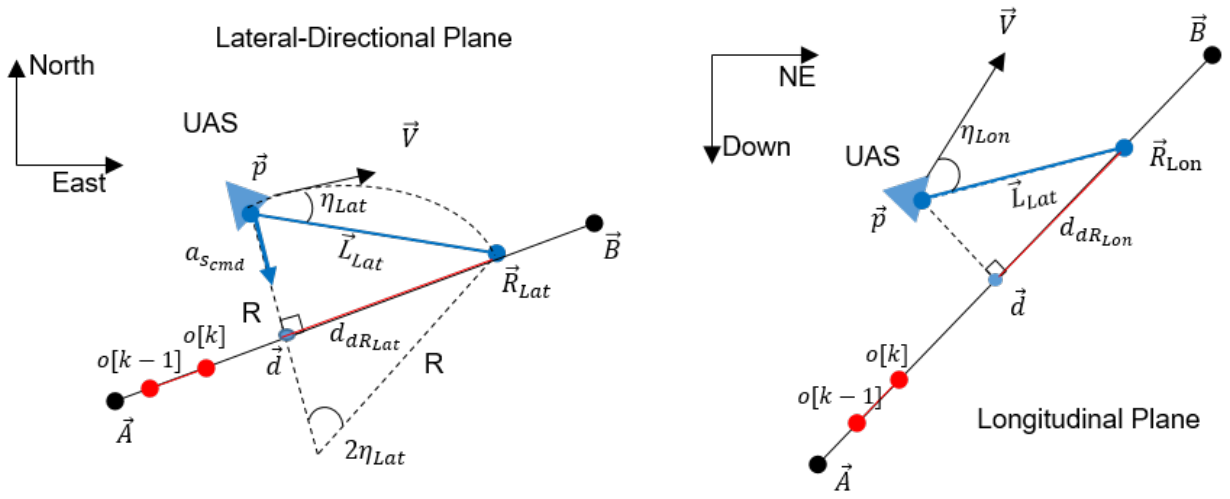


Figure 2.4: Multi-Scale Moving Point (MSMP) Guidance Law Geometric Concept

Eq. 2.1 presents a general moving point position ($\overrightarrow{p}[k]$) and velocity ($\overrightarrow{v}[k]$) in a discrete time (k) where superscript o indicates a moving point not the aircraft. $\overrightarrow{d}$ is perpendicularly projected point on the line segment as follows:

$$\overrightarrow{d} = \begin{bmatrix} d_{N_{Lat}}(t) \\ d_{E_{Lat}}(t) \\ d_{H_{Lat}}(t) \end{bmatrix}_I = \overrightarrow{A} + q_d(t)[\overrightarrow{B} - \overrightarrow{A}], \quad q_d(t) = \frac{\left[\overrightarrow{p} - \overrightarrow{A}\right] \cdot \left[\overrightarrow{B} - \overrightarrow{A}\right]}{d_{AB}^2} \tag{2.2}$$

where d_{AB} is the Euclidean distance between A and B point. $\overrightarrow{R}_{Lon}$ and $\overrightarrow{R}_{Lat}$ are virtual reference points that the aircraft wants to track on longitudinal and lateral-directional plane, respectively. This is the core of the MSMP guidance algorithm. The reference point of the lateral-directional and longitudinal plane are computed separately by using $d_{dR_{Lat}}$ and $d_{dR_{Lon}}$ as follows:

$$\overrightarrow{R}_{Lat}(t) = \begin{bmatrix} R_{N_{Lat}} \\ R_{E_{Lat}} \\ R_{H_{Lat}} \end{bmatrix} = \overrightarrow{d} + m_{r_{Lat}}(t)\left[\overrightarrow{B} - \overrightarrow{d}(t)\right], \quad m_{r_{Lat}} = \frac{d_{dR_{Lat}}}{d_{dB}(t)} \tag{2.3}$$

$$\overrightarrow{R}_{Lon}(t) = \begin{bmatrix} R_{N_{Lon}} \\ R_{E_{Lon}} \\ R_{H_{Lon}} \end{bmatrix} = \overrightarrow{d} + m_{r_{Lon}}(t)\left[\overrightarrow{B} - \overrightarrow{d}(t)\right], \quad m_{r_{Lon}} = \frac{d_{dR_{Lon}}}{d_{dB}(t)} \tag{2.4}$$

where $d_{dR_{Lat}}$ and $d_{dR_{Lon}}$ are predefined distances for the lateral-directional and longitudinal plane, respectively. $\overrightarrow{L}_{Lon}$ and $\overrightarrow{L}_{Lat}$ are the vectors from the position of the UAS to the reference points ($\overrightarrow{R}_{Lon}$ and $\overrightarrow{R}_{Lat}$). η_{Lon} and η_{Lat} are error angles between the velocity vector and $\overrightarrow{L}_{Lon}$ and $\overrightarrow{L}_{Lat}$, respectively. In order to compute η_{Lon} and η_{Lat}, the velocity vector ($\overrightarrow{V}$), L_{Lon}, and L_{Lat} should be transformed from the inertial frame to the tracking line frame (T), which is aligned with the extended line segment generated by two consecutive moving points. Figure 2.5 shows the attitude angles between the inertial frame, (I) and the tracking line frame, (T): α_{track}, β_{track}.

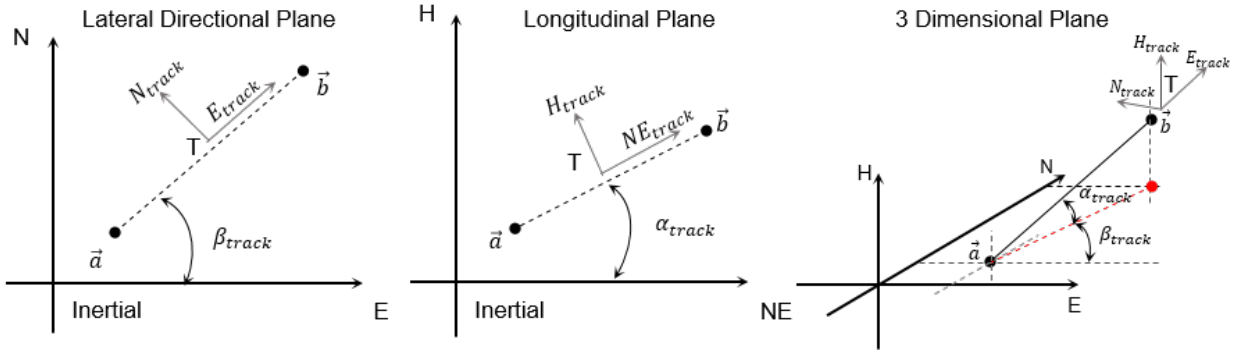


Figure 2.5: The tracking line frame and the attitude angle definition for the transformation

Since the velocity vector($\overrightarrow{V}$), L_{Lon}, and L_{Lat} are first order tensors, the rotation matrix (H_I^T) is used to transform the coordinate system (see Appendix A) and it was defined as follows:

$$H_I^T = \begin{bmatrix} \cos\alpha_{track}\cos\beta_{track} & \sin\beta_{track} & \sin\alpha_{track}\cos\beta_{track} \\ -\cos\alpha_{track}\sin\beta_{track} & \cos\beta_{track} & -\sin\alpha_{track}\sin\beta_{track} \\ -\sin\alpha_{track} & 0 & \cos\alpha_{track} \end{bmatrix} \tag{2.5}$$

Then, the coordinate system of $\overrightarrow{V}$, and $\overrightarrow{L}_{Lon}$ and $\overrightarrow{L}_{Lat}$ are transformed to the tracking line frame as follows:

$$\left[\overrightarrow{V}\right]_T = H_I^T \cdot \overrightarrow{V} = H_I^T \cdot \begin{bmatrix} \dot{p}_N(t) \\ \dot{p}_E(t) \\ \dot{p}_H(t) \end{bmatrix}_I = \begin{bmatrix} \dot{p}_{N_{track}}(t) \\ \dot{p}_{E_{track}}(t) \\ \dot{p}_{H_{track}}(t) \end{bmatrix}_T \tag{2.6}$$

$$\left[\overrightarrow{L}_{Lat}\right]_T(t) = H_I^T \cdot \overrightarrow{L}_{Lat} = H_I^T \cdot \left[\overrightarrow{R}_{Lat}(t) - \overrightarrow{p}(t)\right]_I = H_I^T \cdot \begin{bmatrix} R_{N_{Lat}} - p_N(t) \\ R_{E_{Lat}} - p_E(t) \\ R_{H_{Lat}} - p_H(t) \end{bmatrix} = \begin{bmatrix} R_{N_{Lat}} - p_N(t) \\ R_{E_{Lat}} - p_E(t) \\ R_{H_{Lat}} - p_H(t) \end{bmatrix}_T \tag{2.7}$$

$$\left[\overrightarrow{L}_{Lon}\right]_T(t) = H_I^T \cdot \overrightarrow{L}_{Lon} = H_I^T \cdot \left[\overrightarrow{R}_{Lon}(t) - \overrightarrow{p}(t)\right]_I = H_I^T \cdot \begin{bmatrix} R_{N_{Lon}} - p_N(t) \\ R_{E_{Lon}} - p_E(t) \\ R_{H_{Lon}} - p_H(t) \end{bmatrix} = \begin{bmatrix} R_{N_{Lon}} - p_N(t) \\ R_{E_{Lon}} - p_E(t) \\ R_{H_{Lon}} - p_H(t) \end{bmatrix}_T \tag{2.8}$$

The error angles (η_{Lat} and η_{Lon}) are calculated by the following equations:

$$\eta_{Lat} = \tan^{-1}\left(\frac{[R_{N_{Lat}} - p_N(t)]_T}{[R_{E_{Lat}} - p_E(t)]_T}\right) - \tan^{-1}\left(\frac{\dot{p}_{N_{track}}(t)}{\dot{p}_{E_{track}}(t)}\right) \tag{2.9}$$

$$\eta_{Lon} = \tan^{-1}\left(\frac{\dot{p}_{H_{track}}(t)}{\left[\overrightarrow{V}_{NE}\right]_T}\right) - \tan^{-1}\left(\frac{[R_{H_{Lon}} - p_H(t)]_T}{[L_{Lon_{NE}}]_T}\right) \tag{2.10}$$

$$\text{where } \left[\overrightarrow{V}_{NE}\right]_T = |[\dot{p}_{N_{track}}(t), \quad \dot{p}_{E_{track}}(t)]|\,,$$

$$[L_{Lon_{NE}}]_T = |[R_{N_{Lon}} - p_N(t), \quad R_{E_{Lon}} - p_E(t)]_T|$$

Since these error angles should be bounded between $-\pi$ to $+\pi$, 'atan2' function in MATLAB is used to wrap the angles into this range. Now, the attitude commands (ϕ_{cmd} and θ_{cmd}) can be found by error angles. In Ref. [27], the commands are computed based on PI controller approach. In this work, the lateral-directional guidance law was modified to translate the required lateral acceleration to the roll angle command (ϕ_{cmd}) in order to improve the quality of tracking.

From Figure 2.4 (left), R is the radius of the temporary circle trajectory to reach the virtual reference point ($\overrightarrow{R}_{Lat}$). $a_{s_{cmd}}$ is the desired lateral acceleration (Eq. 2.11) as follows:

$$a_{s_{cmd}} = \frac{V_T^2}{R} = \frac{2V_T^2 \sin\eta_{Lat}}{\overrightarrow{L}_{Lat}}, \quad \text{where } R = \frac{L_{Lat}}{(2\sin\eta_{Lat})} \tag{2.11}$$

The guidance algorithm commits two goals: the reduction of the position error between the aircraft and the desired trajectory and the alignment of the aircraft velocity to the direction of the desired trajectory. It can achieve these goals at once if the error angles (η_{Lat} and η_{Lon}) become zero. ϕ_{cmd} is computed by using the required lateral acceleration in a way that it reduces the lateral error angle (η_{Lat}), see Ref. [45].

$$\phi_{cmd} = \tan^{-1}\left(\frac{a_{s_{cmd}}}{g}\right) \tag{2.12}$$

In addition, an acceleration term ($V_T\omega_{z_I}$) is fed back to the lateral acceleration commands in order to improve the lateral-directional guidance performance, see Ref. [45]. A steady state level turn condition does not reflect the reality in actual flights so Eq. 2.11 is not always valid. The acceleration term ($V_T\omega_{z_I}$) is fed back to the lateral acceleration command (Eq 2.11). If we see the unit of this term, it is easy to observe that it is the acceleration term. The unit of V_T is ft/sec and the unit of ω_{z_I} is rad/sec. As the result, the unit of this term is ft/sec^2. The feedback loop scheme is shown in Fig. 2.6.

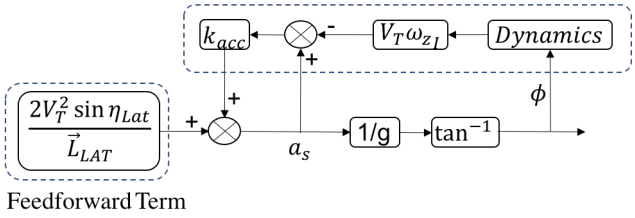


Figure 2.6: Acceleration Feedback Loop

For the longitudinal error angle (η_{Lon}), PI controller approach was kept to compute the pitch angle command (θ_{cmd}) from Ref. [27].

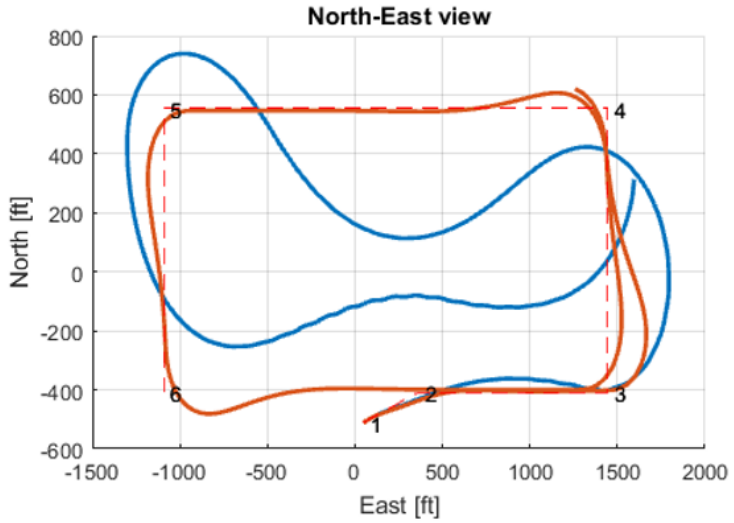
$$\theta_{cmd} = \tan^{-1}\left\{k_{a_{Lon}} \cdot \eta_{Lon} \cdot \left(k_{p_{Lon}} + \frac{k_{i_{Lon}}}{s}\right)\right\} \quad \text{where } K_{a_{Lon}} = \frac{2|\overrightarrow{V}|^2}{|\overrightarrow{L}_{Lon}|} \tag{2.13}$$

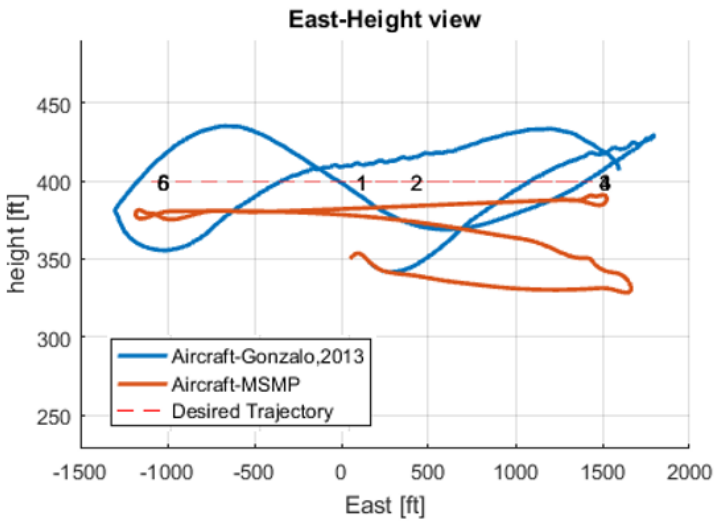
$k_{p_{Lon}}$ and $k_{i_{Lon}}$ are proportional and integral gains for a PI controller. $k_{a_{Lon}}$ is an adaptive gain and it is defined as a fixed value for this work to reduce the oscillation. The sideslip

angle command, β_{cmd} is designated as zero.

The speed command($V_{T_{cmd}}$) is fixed to the aircraft speed when the autopilot was engaged to reflect random initial conditions.

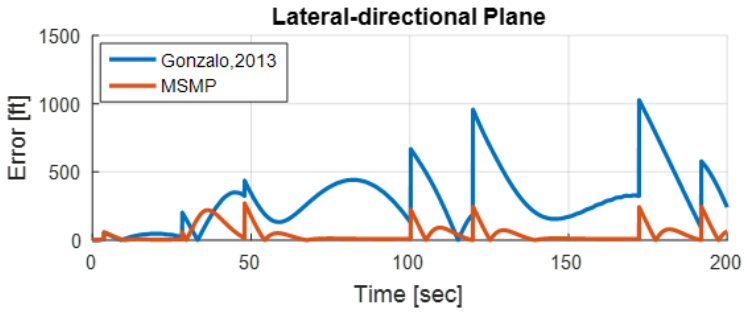
Figure 2.7 presents the comparison of position tracking. Indeed, the north and east position graph shows that the MSMP guidance algorithm tracked the desired race track pattern better than the moving point guidance. In order to quantify the tracking error, the root-mean-square (RMS) of distance error is computed. The tracking error is defined as the distance between the aircraft position and waypoint lines. MSMP guidance algorithm reduced 80% of the RMS for the tracking error in the lateral-directional plane for the specific initial condition shown in Fig. 2.7. Moreover, the oscillations in the tracking was decreased dramatically. On the other hand, RMS of the longitudinal tracking error is increased 44.5%. However, the aircraft could maneuver more smoothly comparing to the moving point guidance. In conclusion, the oscillation and quality of tracking are improved by using MSMP guidance algorithm comparing to the moving point guidance.

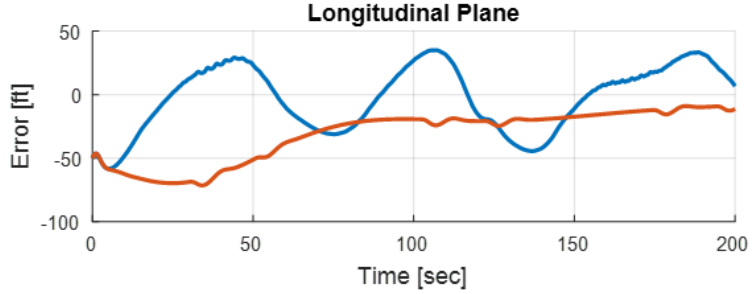




Initial Condition

P_0=-0.6558deg/sec
Q_0=-4.1465deg/sec
R_0=-3.9387deg/sec
ϕ_0=-2.7634deg
θ_0=7.934deg
ψ_0=74.4377deg
V_{T_0}=48ft/sec
N_0=-510.7684ft
E_0=49.0742ft
H_0=350ft
δ_{T_0}=60%
δ_{E_0}=1deg
δ_{A_0}=1.4652deg
δ_{R_0}=-0.378deg
α_0=0.24deg
β_0=-2.5346deg





RMS [ft]	Lateral	Longitudinal
MSMP	73.4 (80% ↓)	37.4 (44.6%↑)
Gonzalo, 2013	365.1	26.0

Guidance Parameter	MSMP	Gonzalo, 2013
$d_{dR_{Lat}}$	360	360
$d_{dR_{Lon}}$	5500	360
$K_{p_{Lat}}$	0.01	0.01
$K_{i_{Lat}}$	0.000625	0.000625
$K_{a_{Lat}}$	25	-
$K_{p_{Lon}}$	0.001	0.001
$K_{i_{Lon}}$	0.03	0.03
$K_{a_{Lon}}$	3	-

Figure 2.7: Comparison the tracking quality and oscillation using RMS

2.1.2 Impact of Guidance Parameters on Tracking and States

In this section, the results of the investigation on the guidance parameters is discussed. All guidance parameters are tuned for different vehicle dynamics affecting the aircraft behaviors.

2.1.2.1 $d_{dR_{Lat}}$ and $d_{dR_{Lon}}$ for ϕ_{cmd} and θ_{cmd}

A longer $d_{dR_{Lat}}$ and $d_{dR_{Lon}}$ cause smaller error angles (η_{Lat} and η_{Lon}), respectively, as shown in Fig. 2.8.

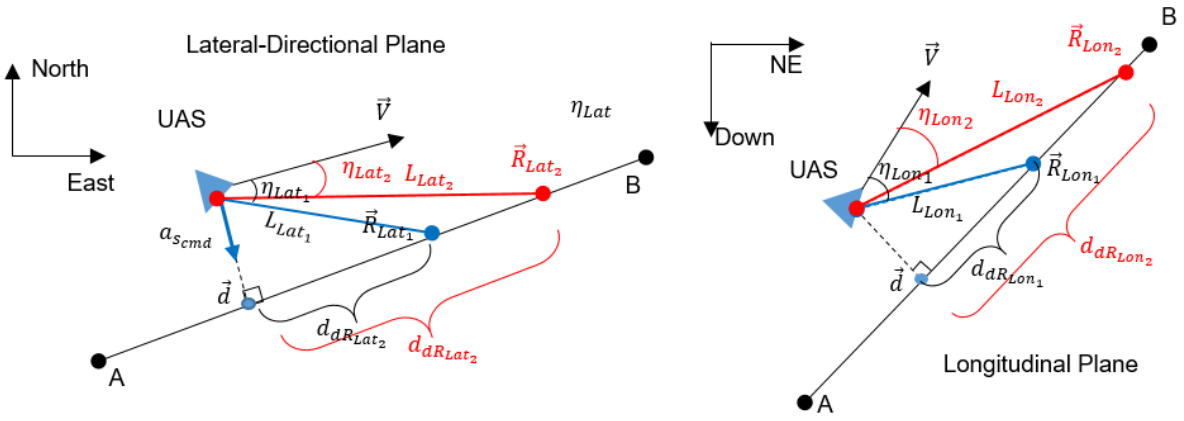


Figure 2.8: Impact of the length of $d_{dR_{Lat}}$ and $d_{dR_{Lon}}$

Sequentially, smaller error angles (η_{Lat_2} and η_{Lon_2} in Fig. 2.8) decrease the attitude commands (ϕ_{cmd} and θ_{cmd}). This phenomena is very profound in terms of the pitch angle command. Since the large magnitude and rapid changing pitch angle command, it can easily cause stall or instability of the aircraft. In other words, when the pitch angle command increases, the angle of attack also changes. Then, the aircraft velocity takes time to catch up the pitch angle command and the angle of attack becomes large which can cause stall. Therefore, the pitch angle command is always desired to be small. When it comes to the roll angle command, the roll mode is a very fast dynamic mode which means the damping ratio is larger than the phugoid and short period modes since roll mode is the first order system. This is because the roll angle command is allowed to be relatively larger than the pitch angle command. Figure. 2.9 shows the comparison between the short and long length of $d_{dR_{Lat}}$ and $d_{dR_{Lon}}$.

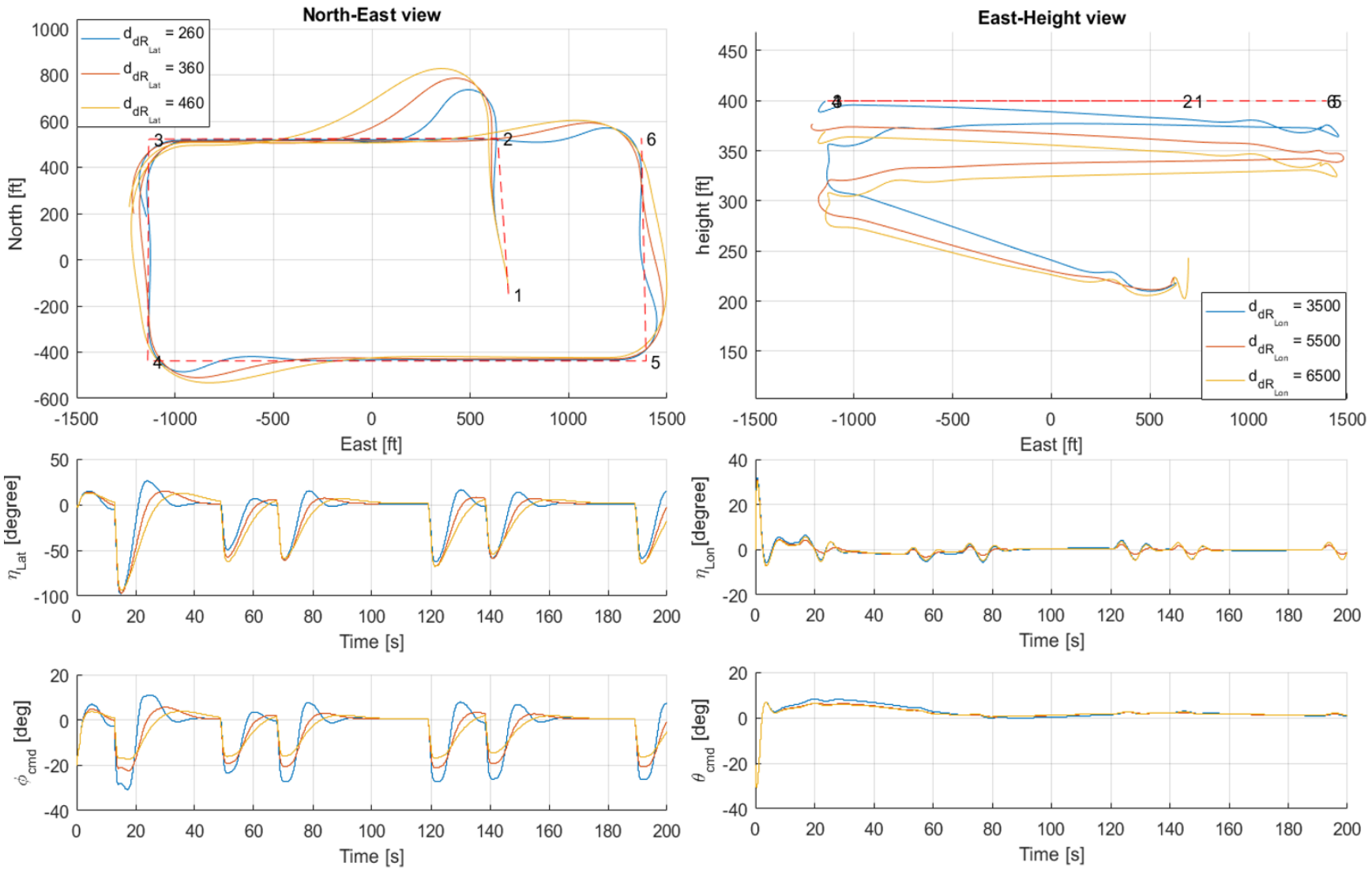


Figure 2.9: Impact of different $d_{dR_{Lat}}$ and $d_{dR_{Lon}}$ to η_{lat}, η_{lon}, and ϕ_{cmd} θ_{cmd}

As Fig. 2.9 shows clearly, less $d_{dR_{Lat}}$ and $d_{dR_{Lon}}$ cause more error angles (η_{lat} and η_{lon}). Due to larger error angles, the roll and pitch angle commands are increased. The lateral error angle (see Fig. 2.9 (left)) is increased from 12 to 25 degrees when $d_{dR_{Lat}}$ is decreased from 460 to 260 ft. Consequently, the roll angle command also has the same trend. It was increased from 5 to 10 degrees when shorter $d_{dR_{Lat}}$ is applied. For the longitudinal error angle, it has the identical phenomena but is less sensitive than the roll angle command. When $d_{dR_{Lat}}$ and $d_{dR_{Lon}}$ are longer, the tracking quality appears error-prone but the maneuver becomes very smooth.

2.1.2.2 The impact of the proportional, integral, and adaptive gains for the pitch angle command

In this section, the impact of longitudinal guidance parameters ($K_{p_{Lon}}$, $K_{i_{Lon}}$, and $K_{a_{Lon}}$) is discussed. Eq. 2.13 presents the pitch angle command (θ_{cmd}) for tracking the moving point on the longitudinal plane.

- $K_{p_{Lon}}$ is the proportional gain for calculating the pitch angle command using the longitudinal error angle (η_{Lon}). The pitch angle command increases by the larger error angle (η_{Lon}).

- $K_{i_{Lon}}$ is the integral gain for calculating the pitch angle command. As the longitudinal error angle (η_{Lon}) increases, the pitch angle command starts to build the integral of the error angle ($\int \eta_{Lon} dt$) to compensate the accumulated error.

- Regarding the adaptive gain ($K_{a_{Lon}}$), the induced oscillation is observed during the investigation of $K_{a_{Lon}}$ impact, see Eq. 2.13. The reason is because the pitch angle command changes rapidly as the aircraft speed varies. In other words, the adaptive gain ($K_{a_{Lon}}$) changes nonlinearly since it is the function of the square of the aircraft speed ($|V_T|^2$). When the aircraft speed is low, the oscillation is small so the performance is acceptable. However, if the aircraft speed is large, the pitch angle command increases dramatically

since the adaptive gain ($K_{a_{Lon}}$) increases nonlinearly with a large speed. This causes large elevator deflections, that are very dangerous for the aircraft dynamics. Large amount of elevator deflections induces large values of the pitch rate (Q). Eventually, large pitch rate leads the stall due to high angle of attack. Therefore, the adaptive gain ($K_{a_{Lon}}$) is set as a constant value to remove the induced oscillation and produce smooth pitch angle command in this work.

Figure. 2.10 shows the influence of the proportional, integral, and adaptive gains ($K_{p_{Lon}}$, $K_{i_{Lon}}$, and $K_{a_{Lon}}$) for the pitch angle command (θ_{cmd}). In order to investigate the impact independently, the integral and adaptive gains ($K_{i_{Lon}}$ and K_{a_Lat}) are set to constant as 0.03 and 3, respectively, while the proportional gain ($K_{p_{Lon}}$) changes. The proportional gain ($K_{p_{Lon}}$) does not change the pitch angle command dramatically while it changes from 0.0005 to 0.1. However, it shows that the higher pitch angle command when the proportional gain ($K_{p_{Lon}}$) is at (0.1). In order to test different integral gains, the proportional and adaptive gains ($K_{p_{Lon}}$ and $K_{a_{Lon}}$) are assumed as 0.03 and 4, respectively. The integral gain ($K_{i_{Lon}}$) has higher sensitivity then the impact of the proportional gain ($K_{p_{Lon}}$). When the integral gain ($K_{i_{Lon}}$) changes from 0.01 to 0.06, the pitch angle commands has higher value (from 3 to 10 degrees) with hight frequency comparing to the smaller integral gain ($K_{i_{Lon}}$). Regarding to the adaptive gain ($K_{a_{Lon}}$), larger ones cause higher pitch angle command from 3 to 10 degrees as it changes from 1 to 5. The proportional and integral gains ($K_{p_{Lon}}$ and $K_{i_{Lon}}$) are fixed at 0.1 and 0.03, respectively.

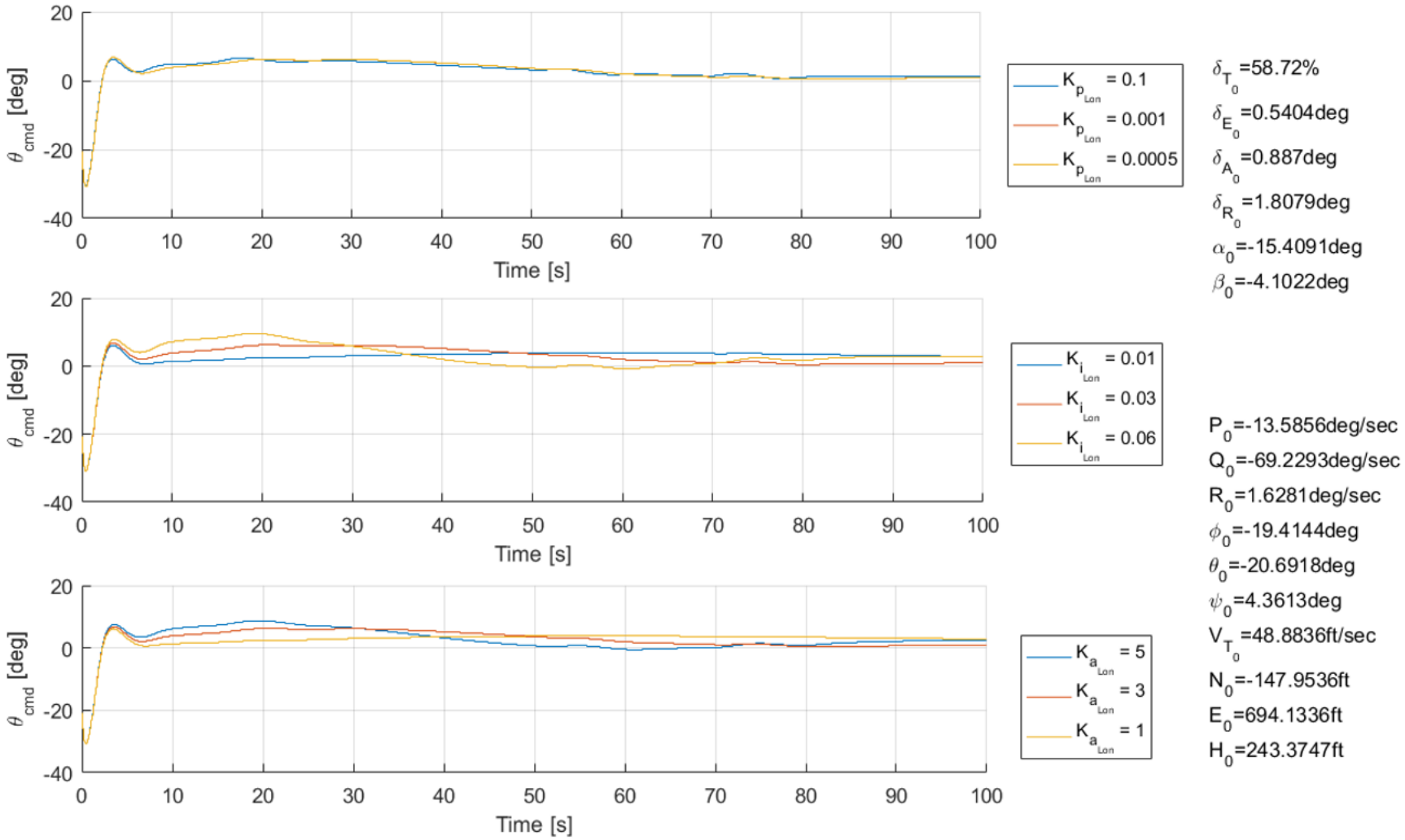


Figure 2.10: Impact of $K_{p_{Lon}}$, $K_{i_{Lon}}$, and $K_{a_{Lon}}$ for θ_{cmd}

2.1.3 Smoothing Function for Guidance and Control Commands

The smoothing function is a great tool for transitioning from the remote control (RC) to the autopilot flight. The guidance and control outputs are smoothed out to reduce the abrupt changes when the autopilot is engaged by using the following equation:

$$x_{smooth} = x_{RC} + \left(\frac{1}{2} - \frac{1}{2}\cos\left(\frac{\pi}{t_s}\Delta t\right)\right)(x_{Auto} - x_{RC}) \tag{2.14}$$

where x are the guidance and control outputs as follows:

$$x_{RC} = \{V_T, \quad \theta, \quad , \phi, \quad \beta, \quad \delta_T, \quad \delta_e, \quad \delta_a, \quad \delta_r\} \tag{2.15}$$

$$x_{Auto} = \{V_{T_{cmd}}, \quad \theta_{cmd}, \quad , \phi_{cmd}, \quad \beta_{cmd}, \quad \delta_{T_{cmd}}, \quad \delta_{e_{cmd}}, \quad \delta_{a_{cmd}}, \quad \delta_{r_{cmd}}\} \tag{2.16}$$

t_s is the desired amount of time for applying the smoothing effect; Δt is the sample time. Figure 2.11 shows the example of smoothing effect in arbitrary states and controls (x) for 10 seconds of t_s. x starts at zero and its desired value is 1 after 10 seconds.

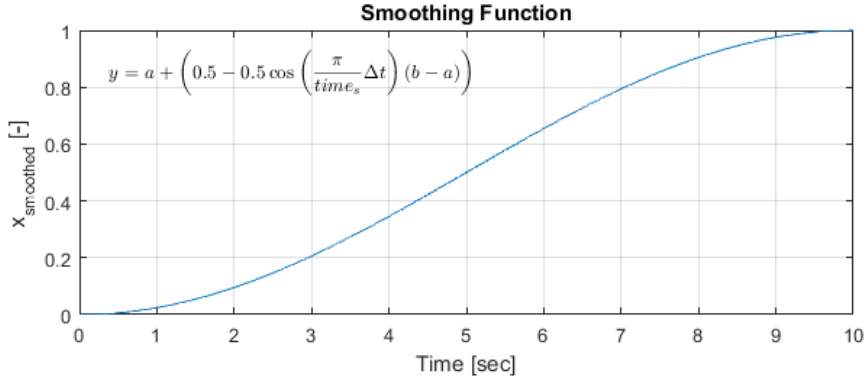


Figure 2.11: Smoothing function from 0 to 1. $x_{initial} = 0, \quad x_{final} = 1, \quad t_s = 10$

As Figure 2.11 shows well, x reaches to the desired value very smoothly after 10 seconds. Now, it is worth noting how this effect can help change guidance and control outputs smoothly. Figure 2.12 shows the comparison between the guidance outputs with and without applying the smoothing function. Firstly, the following figure shows the impact in guidance commands: the speed command ($V_{T_{cmd}}$), the pitch angle command (θ_{cmd}), the roll angle

command (ϕ_{cmd}), and sideslip angle command (β_{cmd}).

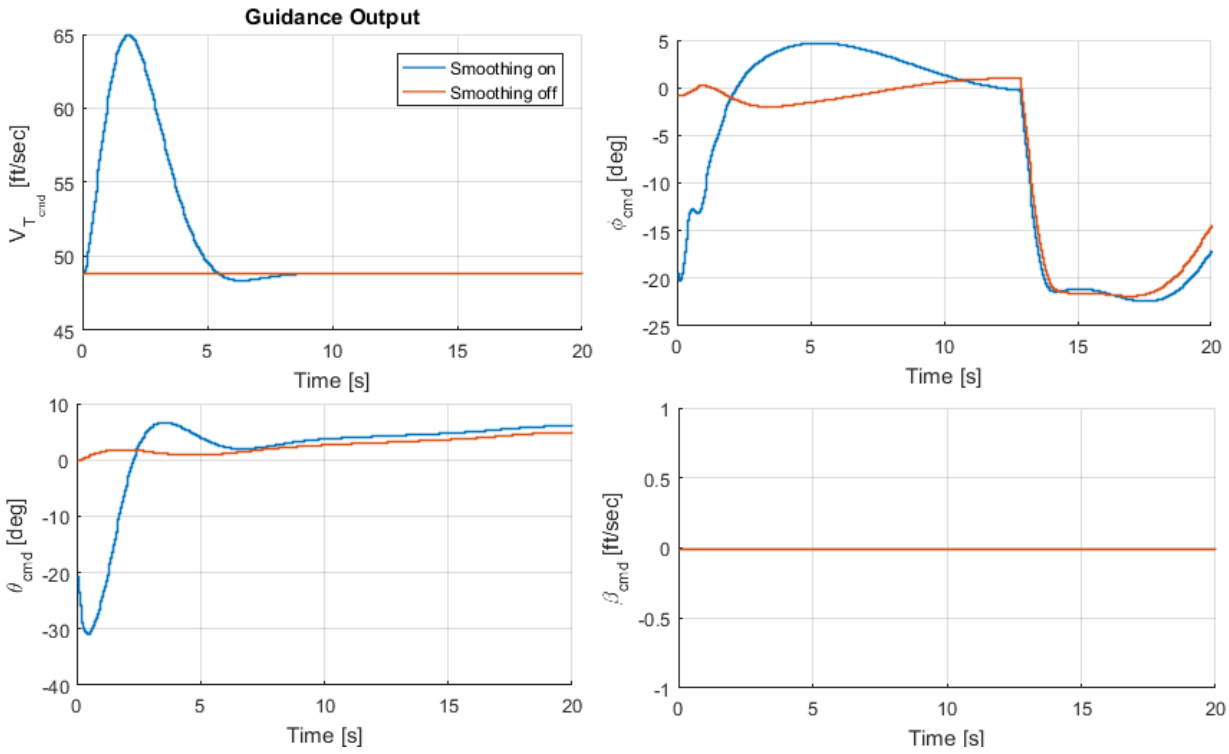


Figure 2.12: Guidance and control outputs comparison between smoothing off and smoothing on, $V_T = 49, \quad \phi = 20°, \quad \theta = -20°, \quad \beta = 0$

Initially, the speed, pitch angle, roll angle, and sideslip angle ($V_{T_{cmd}}$, θ_{cmd}, ϕ_{cmd} and β_{cmd}) starts at 49 ft/sec, -20 degrees, 20 degrees, and 0 degree, respectively. When the smoothing function is applied, the errors between states and guidance commands are zero to avoid abrupt changes in the control outputs. When the smoothing function is not applied, the pitch and roll angle have very large errors (20 degrees in magnitude of both angles) which can cause very high control deflections initially. Regarding the speed command, it varies from 49 ft/sec to 65 ft/sec when the smoothing function is applied. This is because it reflects the current aircraft speed while it is smoothing the command value. The speed command increases in this specific initial condition since the aircraft was descending. There is no benefit to change the speed command since it activates the phugoid mode which will affect to change the angle of attack. Therefore, the smoothing function is not applied to the speed command during the flight test. The sideslip angle is always commanded at zero so it

does not have any impact. This is because there is no sensor to measure the sideslip angle so it was always assumed to be zero. The following figure shows the comparison in control outputs.

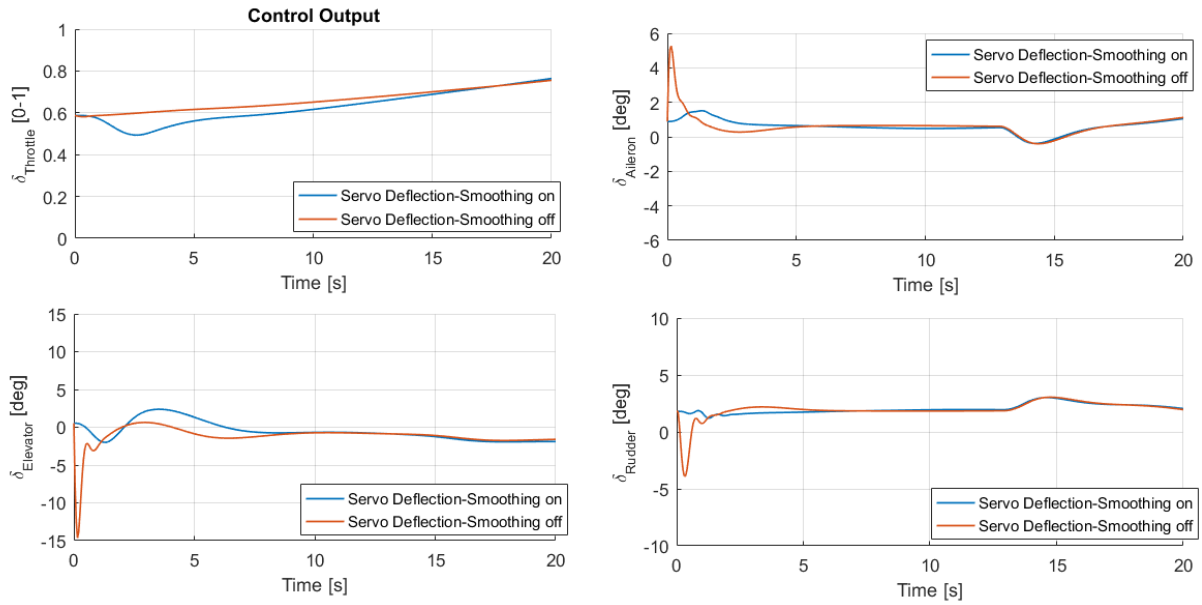


Figure 2.13: Control outputs comparison between the result when the smoothing function is on and off.

If the smoothing function is not applied to the control outputs, they are changed dramatically after the autopilot is engaged. The elevator is changed from 1 to -14 degrees within 2 seconds which is very aggressive for the aircraft due to high angle of attack and excessive structural loading. Moreover, the aileron is changed from 0 to 5 degrees within 1 second. The rudder is commanded from 2 to -4 degrees in less than 2 seconds. These maneuvers are not desired since they cause a quick rolling and yawing motion. When the smoothing effect is applied, the control surfaces change gradually. As an exception, the throttle changes more smoothly when the smoothing function is not applied to the speed command since it did not excite the phugoid mode.

2.1.4 Adaptive $d_{dR_{Lat}}$ Algorithm

Using a constant length of $d_{dR_{Lat}}$, the aircraft could track the moving point in lateral plane without oscillations. However, a large overshoot is observed due to the location of the reference point (R_{Lat}) when the waypoint line segments are changed.

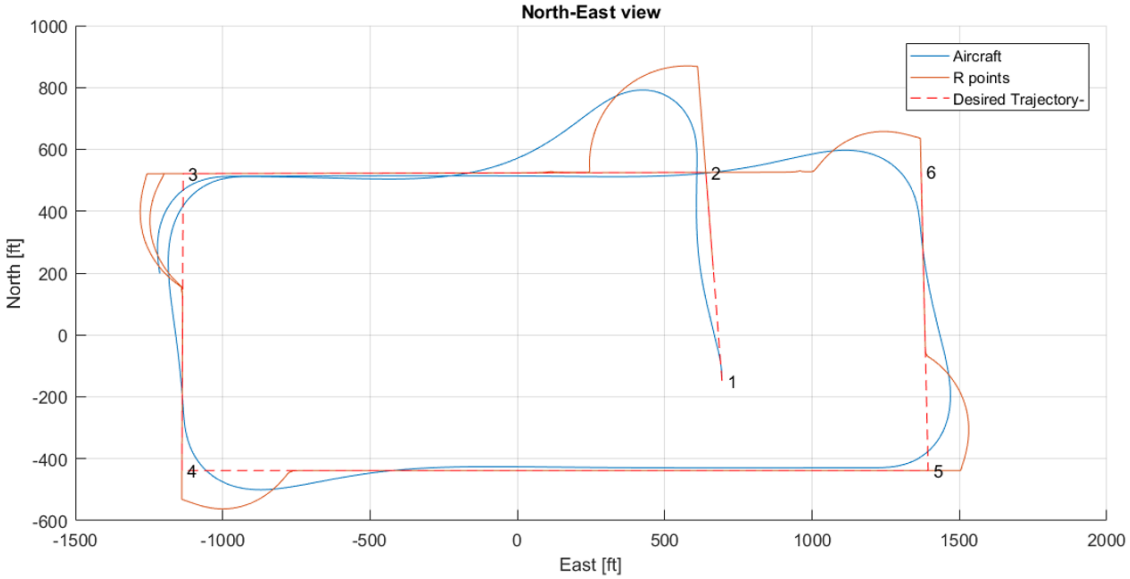


Figure 2.14: Overshoot due to position of the reference point, R_{lat}

As Figure 2.14 shows, the reference point is too far out from the corners and causes an overshoot in the aircraft trajectory. This is because the reference point is generated based on the projected aircraft position onto the waypoint line segment. Therefore, the reference point can be located outside of the waypoint box or lines, even though the moving point did not reach to the next line segment but the aircraft already reached.

The solution of reducing the overshoot is proposed by using the Sigmoid function to change $d_{dR_{Lat}}$ as a function of the yaw rate (R). The advantage of using the yaw rate is that $d_{dR_{Lat}}$ is automatically adaptive based on the aircraft dynamics. When the Sigmoid function is designed, it is important to have the abrupt slope of the Sigmoid function transition since the shallow one causes the oscillation similarly when the adaptive gain ($K_{a_{Lat}}$) is applied.

The other cause of the overshoot is due to the spatial constraint in the flight test area. The confined area forces the aircraft to turn more frequently. If there is no spatial constraint,

the overshoot can be reduced as Fig. 2.14 shows.

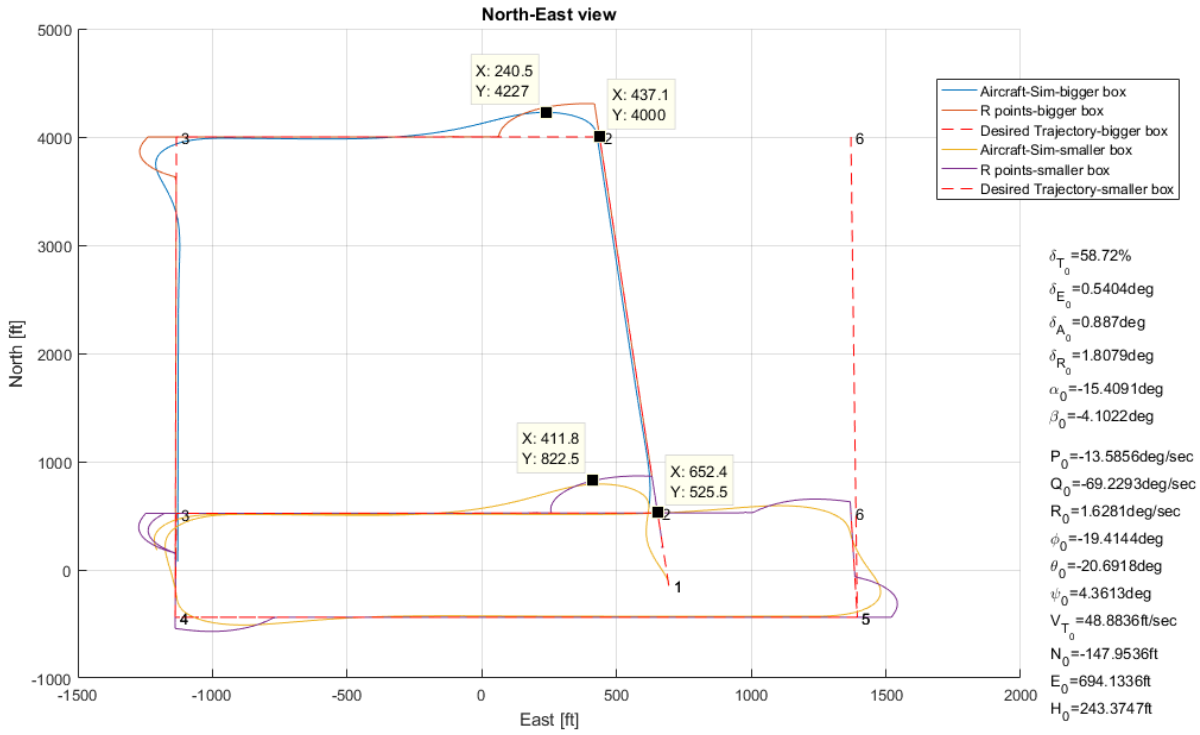


Figure 2.15: Comparison of the highest overshoot (at the first turning point, waypoint #2) between small and large box

The overshoot is reduced by 69.5 ft (from 296.5 to 227 ft) in this specific simulation when the waypoint box is larger.

2.1.4.1 Sigmoid Approach for Adaptive $d_{dR_{Lat}}$

The Sigmoid function produces the adaptive length of $d_{dR_{Lat}}$ using the yaw rate (R) as follows:

$$d_{dR_{Lat}\,Adaptive} = d_{dR_{Lat}} - \underbrace{\Delta d_{dR_{Lat}} \cdot \frac{1}{1+e^{-C_1(|R|-C_2)}}}_{\text{Correction term}} \tag{2.17}$$

$\Delta d_{dR_{Lat}}$ is the desired increment length for changing $d_{dR_{Lat}}$. C_1 is the positive constant to decide how shallow or steep the slope of Sigmoid function transition is. If C_1 is large, the Sigmoid function varies its values slowly from 0 to 1. For example, if C_1 is 6 deg/s, then the value changes within the large range of the yaw rate. If C_2 is 10 deg/s, then Sigmoid function

will change the values between 4 and 16 deg/s. C_2 is the positive constant which determines where the average value of transition meets in terms of the yaw rate. The average value of transition is 0.5 in this case since the value changes from 0 to 1. For example, if C_2 is 10 deg/s, then the middle value of transition in the Sigmoid function would happen at 10 deg/s of the yaw rate. In other words, it provides the threshold of changing $d_{dR_{Lat}}$. The correction term of Eq. 2.17 is subtracted from the original value of $d_{dR_{Lat}}$ instead of adding it due to the shorter $d_{dR_{Lat}}$ reduces the overshoot in position tracking by larger lateral acceleration commands.

As Figure 2.16 shows, $d_{dR_{Lat}}$ changes from 360 to 260 ft whenever the yaw rate (R) starts to change. Then, it comes back to the original value after the aircraft finishes turning maneuvers. The RMS of the distance error is reduced by 10.3 % due to the adaptive $d_{dR_{Lat}}$.

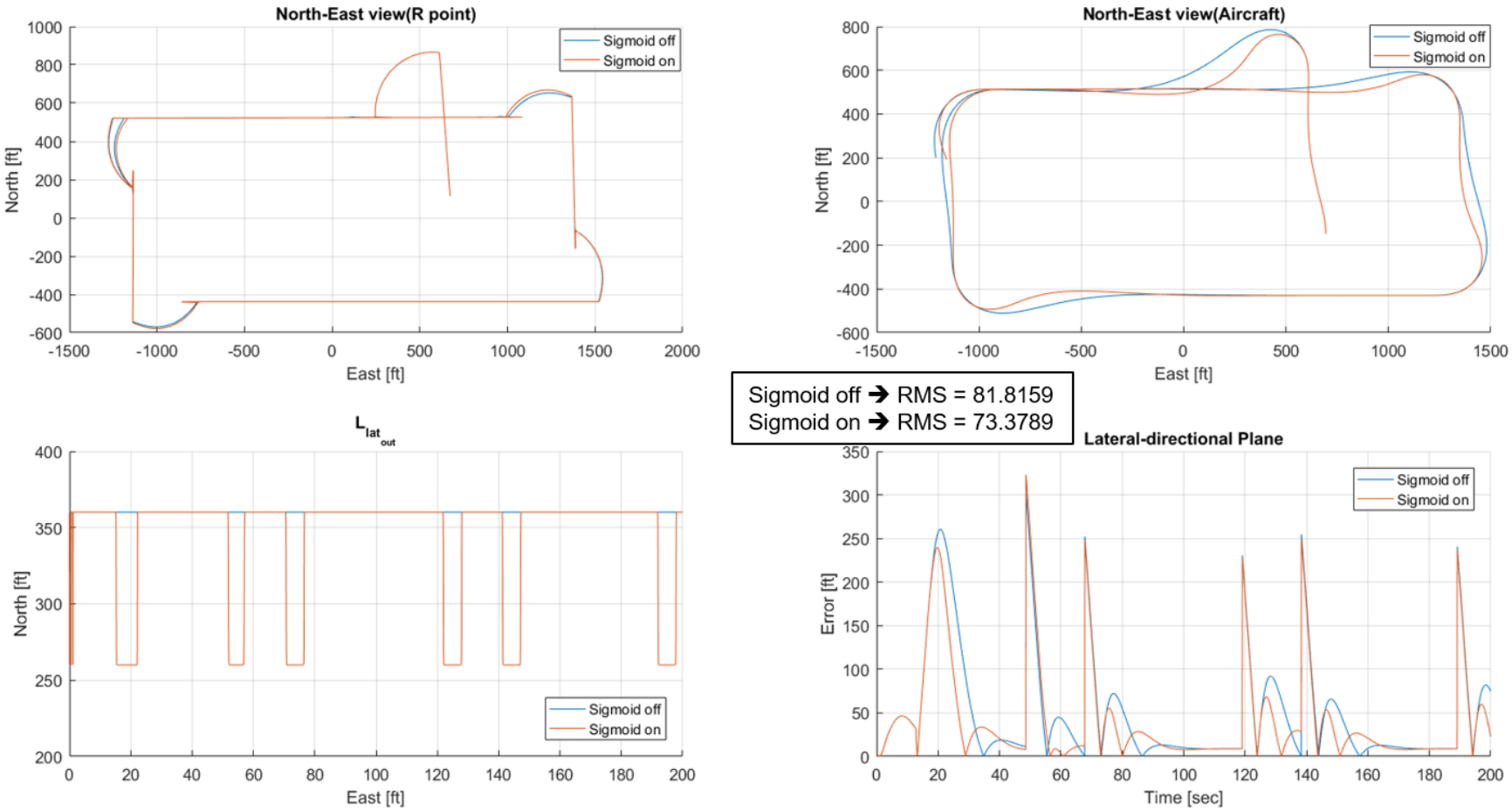


Figure 2.16: Adaptive $d_{dR_{Lat}}$ using $C_1 = 9$, $C_2 = 9.8$ and $\Delta d_{dR_{Lat}} = 100$

2.2 Navigation

In this section, 'phasic' navigation is introduced for path planning of the multi-agent systems. In the first phase, formation initiation and assignment algorithms are introduced: Curvature control and Hungarian Algorithm. In the second phase, six different algorithms have been developed to hold the formation: intelligent waypoint modification and index assignment, path planning by LQ guidance for the virtual leader, the virtual point formation for the outer agent, moving mesh methods for the inner agent and morphing potential for the collision avoidance.

2.2.1 Phase 1: Formation initiation and Formation Assignment

2.2.1.1 Curvature control path planning algorithm using Frenet-Serret Equations

This algorithm aligns the velocity of each moving point to the identical heading, which is the average or designated direction without the collision. This algorithm has been developed for robot cars with very slow speed in two dimensional space, see Ref. [68]. It has also been expanded to three dimensional space by Ref. [46]. In order to utilize this algorithm for the aircraft dynamics, two main features are improved: the application of 6 DoF dynamic constraints and the translation of the curvature and torsion into the heading and pitch angle change using the Frenet-Serret Equation, respectively. This algorithm has a critical drawback that it cannot hold the formation shape. However, this algorithm is very powerful to aggregate all random velocity vectors into a united direction. This is the reason why this algorithm is the first stage of phasic navigation to deal with the randomness of initial conditions. A Frenet-Serret frame and associated formulas are utilized to compute the desired curvature and torsion to align the velocity vector with other agents. This frame consists of the unit tangent, normal, and binormal vector (x_{fs}, y_{fs}, and z_{fs}, respectively) on the differentiable trajectory ($r(s)$) which is a function of arc length (s). The unit tangent vector

(x_{fs}) aligns with the moving point velocity. The unit normal vector (y_{fs}) always faces the center of the turning radius on the trajectory (r). The unit binormal vector (z_{fs}) is a cross product of the tangent and normal vectors. Figure 2.17 shows the geometric description of Frenet-Serret frame.

Using the Frenet-Serret frame, its formula can be presented as follows (subscript fs was omitted for simplicity):

$$\dot{x} = \kappa y \tag{2.18}$$

$$\dot{y} = -\kappa x + \tau z \tag{2.19}$$

$$\dot{z} = -\tau z \tag{2.20}$$

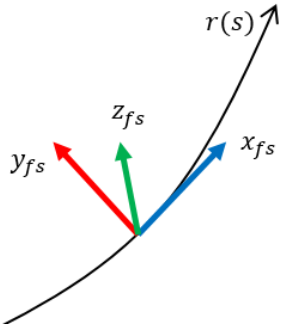


Figure 2.17: Frenet-Serret Frame

where κ is the curvature; τ is the torsion. These equations show three partial differential equations in respect to the arc length (s). Eq. 2.18 to Eq. 2.20 represent how the vectors (x_{fs}, y_{fs}, and z_{fs}) would change depending on the curvature and torsion. With Frenet-Serret formulas, the curvature and torsion should be computed to guide the agents to aggregate. These terms are calculated by following equations:

$$\kappa_k = \sum_{j \neq k} \kappa_{jk} \tag{2.21}$$

$$\text{where } \kappa_{jk} = \underbrace{-\eta_\kappa \left(\frac{r_{jk}}{|r_{jk}|} \cdot x_k\right)\left(\frac{r_{jk}}{|r_{jk}|} \cdot y_k\right)}_{term1} \underbrace{- f(|r_{jk}|)\left(\frac{r_{jk}}{|r_{jk}|} \cdot y_k\right)}_{term2} + \underbrace{\mu_\kappa x_k \cdot y_k}_{term3}$$

$$\tau_k = \sum_{j \neq k} \tau_{jk} \tag{2.22}$$

$$\text{where } \tau_{jk} = \underbrace{-\eta_\tau \left(\frac{r_{jk}}{|r_{jk}|} \cdot x_k\right)\left(\frac{r_{jk}}{|r_{jk}|} \cdot z_k\right)}_{term1} \underbrace{- f(|r_{jk}|)\left(\frac{r_{jk}}{|r_{jk}|} \cdot z_k\right)}_{term2} + \underbrace{\mu_\tau x_k \cdot z_k}_{term3}$$

where k indicates each agent; r_{jk} is the relative position vector from k^{th} agent to j^{th} agent ($r_j - r_k$). $|r_{jk}|$ is the distance between a pair of agents. α, η_κ,μ_κ, η_τ, and μ_τ are constants.

$f(|r_{jk}|)$ is the potential function to avoid collision between the agents as follows:

$$f(|r_{jk}|) = \alpha \left[1 - (\frac{r_0}{|r_{jk}|})^2 \right] \tag{2.23}$$

The first term of κ_{jk} is to align a pair of agents. Particularly, the velocity of agents would be perpendicular to the relative position vector (r_{jk}). The second term uses the potential field (Eq. 2.23) to generate the curvature for the collision avoidance between agents. The last term makes the tangent and normal vectors of agents into a common orientation. With these three terms, the curvature and torsion aggregates agents into one direction. κ_k and τ_k are the basic laws of swarming agents. Figure 2.18 shows the result of the simulation in two dimensional space.

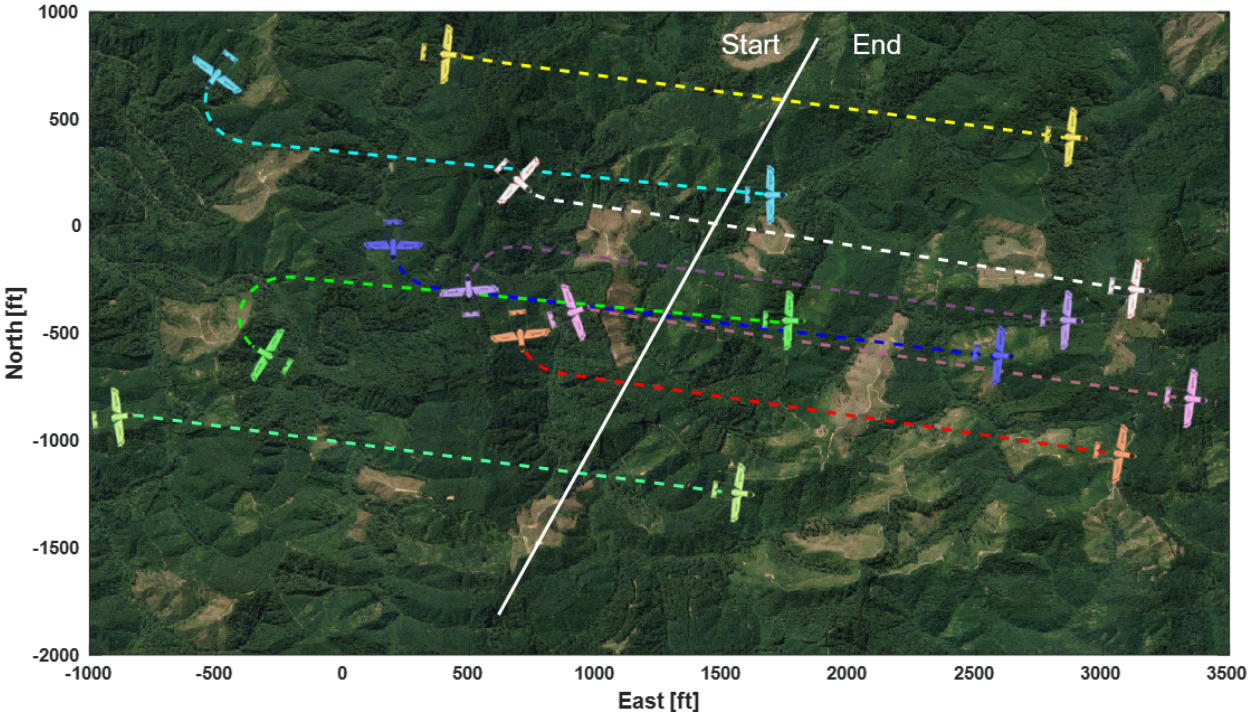


Figure 2.18: Aircraft simulation with the aircraft dynamics using the curvature, In this simulation, all agents travel at 50 ft/sec.

Agents have the random heading angles when the algorithm begins. After the curvature control algorithm is applied, all agents proceeds to the identical direction. During the simulation, the limitation in total curvature is applied not to violate the minimum turning

radius.

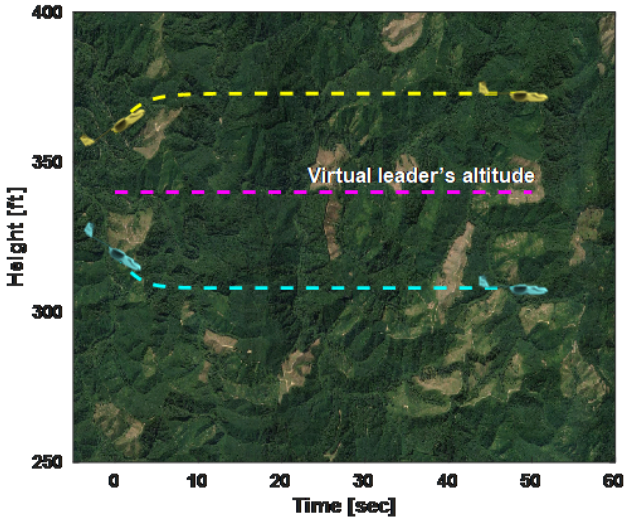


Figure 2.19: Aircraft simulation with the aircraft dynamics using torsion

Figure 2.19 shows the result of the torsion control. One agent ascends while the other one descends when the simulation is started. Torsion control can guide agents to converge to the altitude by matching the velocity direction to the virtual leader's velocity direction.

2.2.1.2 Formation Assignment Algorithm

When all agents proceed to one direction, each agent should determine which formation position is the desired one. This problem is called the assignment problem in mathematics. The optimal solution of the assignment problem is well known as Hungarian algorithm, see Ref. [53]. Let's consider that there are four people (person A,B,C, and D) and four jobs (job 1,2,3, and 4). Figure 2.20 shows the simple example of the cost matrix in the assignment problem. Figure 2.20 (right) shows the result of assignment job to each person. The assigned job has minimum cost with respect to each person.

		Job			
		A	B	C	D
Person	1	2	9	1	5
	2	9	2	4	3
	3	5	25	12	9
	4	8	6	14	5

		Job			
		A	B	C	D
Person	1	2	9	1	5
	2	9	2	4	3
	3	5	25	12	9
	4	8	6	14	5

Figure 2.20: Example of the assignment problem: given cost matrix(left), assigned result(right)

In order to utilize Hungarian algorithm, the cost function should be defined. Existing work in Section 1.2 only uses the distance error between each agent to the formation position to calculate the cost. However, this approach has the drawback of ignoring the aircraft dynamics. In this work, the heading and rotation angle errors are added to the cost function to include the aircraft motion. Fig 2.21 shows the definition of heading and rotation angles. Eventually, the cost function consists of three different categories: distance error between each agent to the formation position (term 1), the heading angle error (term 2), and the rotation angle error (term 3), see Eq. 2.24.

$$J_{ij} = \underbrace{c_1 \left| \overrightarrow{p}_i - \overrightarrow{p}_j \right|^2}_{term1} + \underbrace{c_2 * (\psi_i - \psi_j)^2}_{term2} + \underbrace{c_3 \left(\psi_{r_i} - \psi_{r_j}\right)^2}_{term3} \tag{2.24}$$

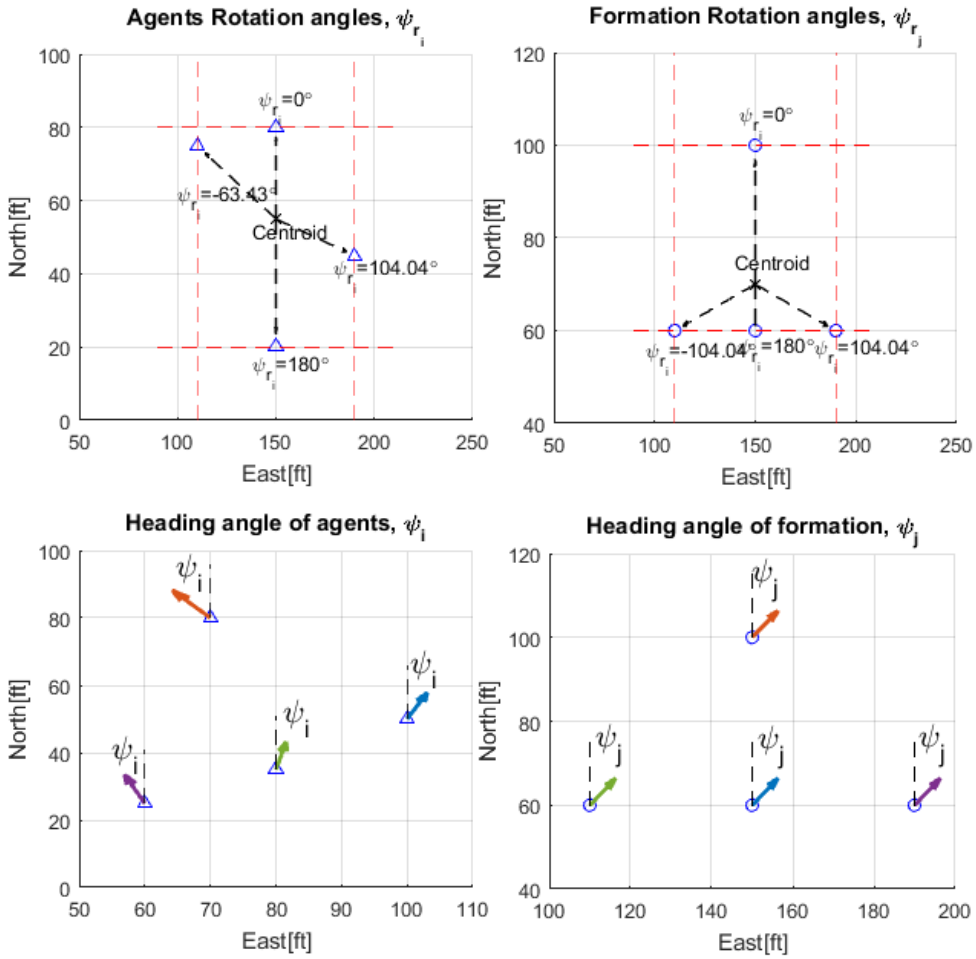


Figure 2.21: Definition of the rotation angle of the formation and agents(1st row) and the heading angle of the formation and agents (2nd rows)

where $i = 1, \cdots, n$ is the number of agents; $j = 1, \cdots, m$ is the number of the formation position; ψ_i is the aircraft heading angle; ψ_j is the desired heading angle; ψ_{r_i} is the rotation angle of the agent respect to the centroid of agents position; ψ_{r_i} is the rotation angle of the formation respect to the centroid of desired position; c_1, c_2, and c_3 are the constant coefficients. Once the cost matrix is obtained, the Hungarian algorithm can be applied as follows:

Step 1. For all rows, find the minimum cost for each row and subtract throughout all elements.

Step 2. For all columns, find minimum cost for each column and subtract throughout all

elements. If there is zero in the column, skip this step.

Step 3. Find the row or column that has only one zero among all elements. Cover all zeros until all rows and columns are checked. If the number of the assignment is needed as n, the same number of zeros should be assigned to each element. Otherwise, proceed to the next step.

Step 4. Find the rows or columns that are not covered from the previous stage. Find the minimum values (k) among uncovered elements and subtract it from them. However, if the element is already covered then add the minimum value (k) to them.

In order to provide the formation reference to agents, the virtual terminal is used. This terminal will wait for closest agents sequentially until all agents arrived at each formation position. The virtual terminal can wait but the moving point will travel continuously in order to design the suitable path planning for high speeds and high moment of inertia configuration. Each formation position is obtained by using 'Virtual Point Formation Algorithm' discussed in Section 2.2.2.3.

Details of path planning using Hungarian algorithm are presented as follows:

Step 1. The virtual terminal is defined by the center of formation and desired relative distance. The center of formation is location ahead of all agents. The design parameter in this step is how far the center will be from agents.

Step 2. Based on the virtual terminal, the Hungarian algorithm is applied and assigns all agents to the formation position.

Step 3. Once closest agents arrive at the desired formation position, fix this assignment and rerun the Hungarian algorithm except the agent already arrived.

Step 4. Then, the virtual terminal will be updated based on the next closest agent. In order to catch up to the formation position, the virtual terminal is assumed to travel 90% of the desired speed to prevent stall. Therefore, the relative speed will be 10% of

the desired trim speed. Since we know how fast the formation travels, the time can be calculated to reach next closest formation. The following equation is used to calculate the time to catch up the next formation position:

$$t_{arrival} = \frac{\overrightarrow{p}_i - \overrightarrow{p}_{ij}}{0.1|\overrightarrow{V}_{desired}|} \tag{2.25}$$

where $t_{arrival}$ is the time which takes to catch up from the agent to the formation position; $\overrightarrow{p}_i$ is the closest agent to the next virtual terminal; $\overrightarrow{p}_{ij}$ is the assigned formation position based on the next virtual terminal; $\overrightarrow{V}_{desired}$ is the desired trim speed of the aircraft.

Step 5. Once the closest agent arrives at the designated formation position by Hungarian algorithm, repeat from Step 3 until all agents are assigned to the formation position.

In order to generate a path and avoid collision, the morphing potential and path following algorithm is applied. The details of these algorithms are discussed in Sec. 2.2.2.5 and Sec. 2.2.2.6.

The result of Hungarian algorithm is presented in Figure 2.22.

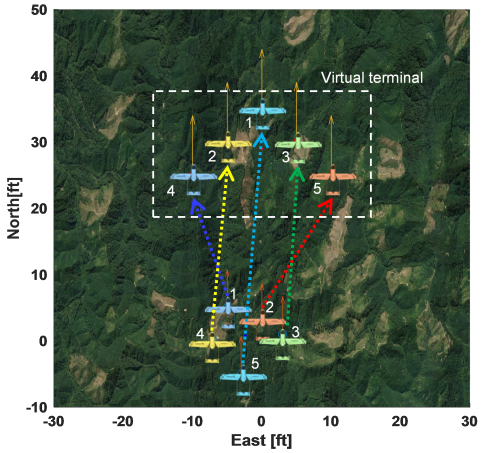


Figure 2.22: Simulation result of Hungarian method for the formation. Ref. [13] is used for the practical implementation.

2.2.2 Stage 2: Formation Holding

2.2.2.1 Intelligent waypoint modification and its index decision algorithm

- Intelligent waypoint modification algorithm

  This algorithm modifies the given waypoints based on the aircraft position and velocity in order to assist the smooth transition from the remote control to the autopilot. The first waypoint is set to the intersection point on the waypoint box by using the aircraft position and velocity. This provides the time to settle the aircraft attitude angles since the first task of the autopilot is to follow the straight line. In case of the multi agent flight, the average of position and velocity vector is used to find the intersection point. Based on the intersection point, the initial virtual leader position and velocity can be obtained as follows:

$$\overrightarrow{V}^{virtual} = \frac{1}{n}\sum_{i=1}^{n}\overrightarrow{V}_i^{UAV} \tag{2.26}$$

$$\overrightarrow{p}^{virtual} = \left(\frac{1}{n}\sum_{i=1}^{n}\overrightarrow{p}_i^{UAV}\right) + \overrightarrow{V}^{virtual}\cdot n_p\Delta t \tag{2.27}$$

  where n_p is the number of the time step; The equation shows that the initial position of the virtual leader is located ahead of all agents by the designated time step (n_p) to avoid abrupt maneuvers. However, the line aligned with the aircraft velocity can have the multiple intersection points among all line segments made by given waypoints. In order to select the appropriate intersection point, the following algorithm has been designed.

  Step. 1 Compute all intersection points between the aircraft velocity and all waypoints. In order to proceed, all lines (l_i) that can be made by a pair of consecutive waypoints are identified as follows:

$$l_i = \frac{\left(p_{i+1\,N}^{way} - p_i^{way}{}_N\right)}{\left(p_{i+1\,E}^{way} - p_i^{way}{}_E\right)}(x - p_i^{way}{}_E) + p_i^{way}{}_N \tag{2.28}$$

where $\overrightarrow{p}_i^{way}$ is the waypoints position defined as $\{{p_i^{way}}_N, {p_i^{way}}_E, {p_i^{way}}_H\}^T$. When the final waypoint is selected, the first waypoint is used to form a pair of points so that it can make the line.

Step. 2 Obtained lines are equated to the line aligned with the aircraft velocity (or average velocity vector for the multi-agent system), see Eq. 2.28 and Eq. 2.29.

$$l_{virtual} = \tan\left(\frac{V_N^{virtual}}{V_E^{virtual}}\right) x + b \tag{2.29}$$

$$l_{virtual} = l_i \tag{2.30}$$

where $b = {p_i^{way}}_N - \tan\left(\frac{V_N^{virtual}}{V_E^{virtual}}\right) {p_i^{way}}_E$.

Step 3. Once all lines are identified, the intersection points between each line can be determined by solving x and y.

Step 4. Select the proper intersection point based on the following criteria:

- The direction alignment criteria: The relative distance vector and the aircraft velocity should be in the identical direction. The relative distance vector is defined from the initial virtual leader position to the intersection point. Otherwise, the agent can proceed to the opposite direction from the first waypoint. This is not a desirable navigation strategy since the required heading angle change is 180 degrees which can cause abrupt changes in the roll angle command.
- The minimum turning radius criteria: The distance between the aircraft and the first waypoint should be at least larger than the minimum turning radius to plan the physically feasible flight path. If the first waypoint is closer than the minimum turning radius, the next possible intersection point is chosen.
- Staying the waypoint box area: The intersection point should be inside of the waypoint box area. If the intersection point is located outside of the waypoint box, it should not be chosen for the first waypoint.

For example, Figure. 2.23 shows that four intersection points can be found by equating Eq. 2.28 and Eq. 2.29. Based on these criteria, the intersection point 3 is chosen in this

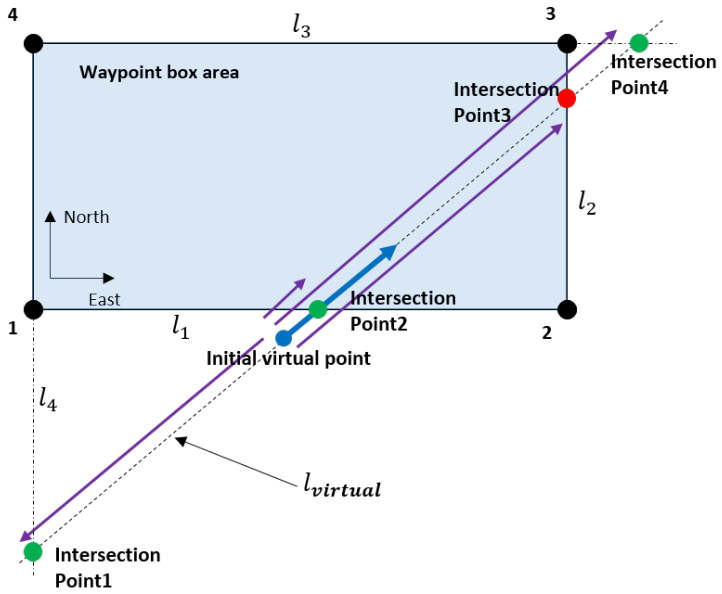


Figure 2.23: Example of intersection points using Eq. 2.28 and Eq. 2.29

example, see Fig 2.23. Intersection point 1 violates the direction alignment criteria. The aircraft should turn to a 180 degree heading angle to follow this point. At the same time, it is located outside of the waypoint box area. Intersection point 2 is closer than the minimum turning radius, so the next possible one will be considered. Intersection point 4 is located outside the waypoint box. Since the intersection point is properly chosen, the error angles in multi-scale moving point guidance will be very small. Consequently, the roll angle command should not be large or abrupt when the autonomous flight mode is engaged.

- Waypoint index decision algorithm

  This algorithm provides the guideline for choosing the sequence of waypoints in a way that requires less aggressive maneuvers. When the waypoint index or sequence is not updated, the tracking of waypoints has huge overshoot since the aircraft tries to follow the wrong sequence of waypoints. Figure 2.24 shows one of these cases.

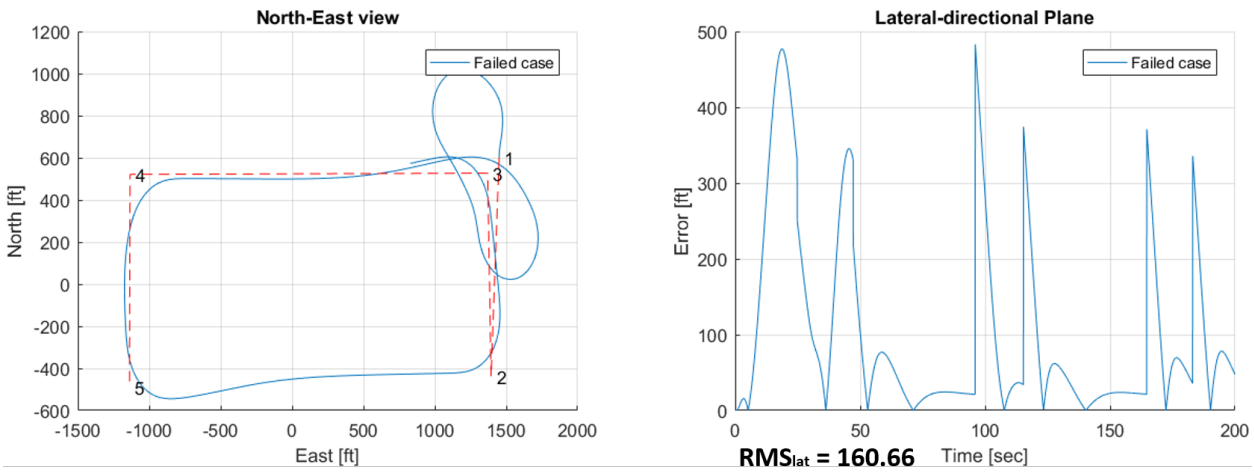


Figure 2.24: Simulation without waypoint index decision algorithm

As Figure 2.24 shows, the tracking error is very large. If the navigation is intelligent enough, it could guide the aircraft to follow the closest waypoint which is number 3 instead of number 2. The procedure of this algorithm is presented as follows:

– If an intersection point exists,

Step. 1 The first waypoint is the aircraft position and the intersection point is allocated as the second waypoint.

Step. 2 Since the line segment that crosses the intersection point is known, the required heading angle changes can be found by the current and neighbor waypoints. For example, let's assume the line segment where the intersection point is located is made by the waypoints of index 2. Then, the neighbor waypoints are index 1 and 3. Then, the required heading angle changes ($\Delta\psi_{ij}$) are calculated by the following equation:

$$\Delta\psi_{ij} = \tan^{-1}\left(\frac{\left(p_{E_j} - p_{E_{inter}}\right)}{\left(p_{N_j} - p_{N_{inter}}\right)}\right) - \tan^{-1}\left(\frac{v_{i_E}}{v_{i_N}}\right) \tag{2.31}$$

where $j = k$ or $k+1$; k is the waypoint index which contains the line where the intersection point is located; i is the agent; $\overrightarrow{p}_{inter}$ is the position of the chosen

intersection point. Repeat this step with the other neighbor waypoint.

Step. 3 Allocate the next waypoint which has smaller required heading angle changes ($\Delta\psi_{ij}$).

Step. 4 Repeat the previous step until all waypoints are assigned.

– Depending on the initial conditions, the aircraft velocity might not have the intersection points with the waypoint box. If the intersection point does not exist,

Step. 1 Calculate the distance between the initial aircraft position for a single agent or the initial virtual leader position for multi-agents as follows:

$$d_{ij} = \left| \overrightarrow{p}_i - \overrightarrow{j} \right| \tag{2.32}$$

where i is the agent; $j = 1, ..., n_{way}$ is the waypoint index of given n_{way} waypoints. Then, choose two points that are most closest from the aircraft ($\overrightarrow{p}_i$).

Step. 2 Compute the required heading angle with respect to two chosen points ($\overrightarrow{p}_c$ where $c = 1, 2$).

$$\Delta\psi_{ic} = \tan^{-1}\left(\frac{(p_{E_i} - p_i)}{(p_{N_i} - p_i)}\right) - \tan^{-1}\left(\frac{v_{i_E}}{v_{i_N}}\right) \tag{2.33}$$

Step. 3 Allocate the next waypoint by the one that has smaller $\Delta\psi_{ic}$.

Step. 4 Repeat the previous steps until all waypoints are allocated.

Figure 2.25 shows the result of the waypoint index decision algorithm. Figure 2.25 (left) shows the result when the waypoint index are not updated properly. The RMS of the failed case is 160.7 ft. Figure 2.25 (right) presents the result when the waypoint index decision algorithm is applied. Instead of planning the path towards to waypoint 2 (Figure 2.25 (left)), the waypoint index sequence is updated based on the required heading angle change. Now, the aircraft follows the different waypoint sequence. The tracking RMS is reduced 6.3 % for this specific initial conditions.

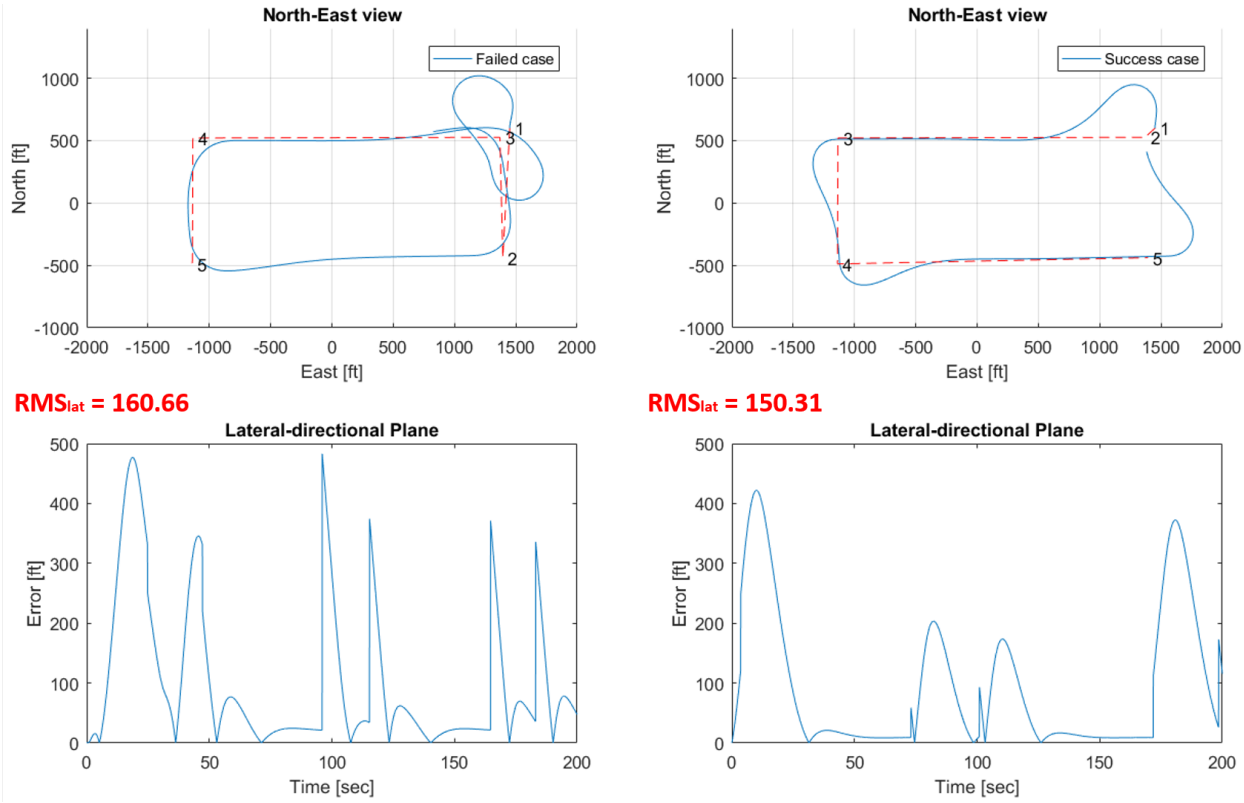


Figure 2.25: Result of the waypoint allocation based on the required amount of the heading angle change

2.2.2.2 Virtual Leader: LQ guidance Path Planning

In this work, this guidance has been utilized as the path planning algorithm. The advantage of this algorithm is that the commanded lateral acceleration is optimized based on Linear Quadratic Regulator (LQR) control theory. This approach is very useful since the principle of LQR control is to regulate all states to zero. In this case, the states are defined as the position and velocity error. LQ guidance has been developed in two-dimensional space with 3 DoF dynamics, see Ref. [80]. In this work, the LQ guidance algorithm is expanded to three dimensional space with 6 DoF aircraft dynamics. In order to obtain the state space used for solving the Ricatti equation, the error dynamics are found by the relationship between the current and desired position and velocity. Figure. 2.26 shows the geometry.

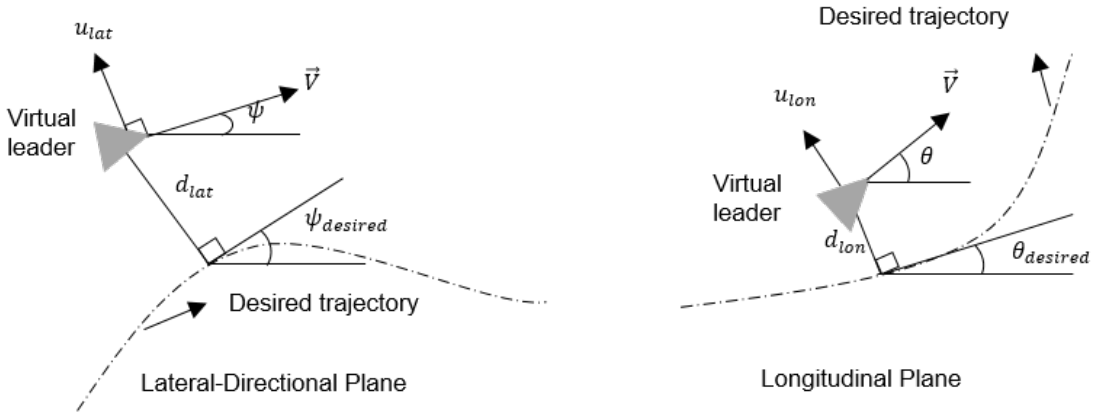


Figure 2.26: Formulation geometry for 3 dimensional space

The rate of position errors ($\dot{d}_{lat}$ and $\dot{d}_{lon}$) are expressed by the speed and differences of the heading (ψ) and pitch angles (θ) for lateral-directional and longitudinal plane, respectively.

$$\dot{d}_{lat} = \left|\overrightarrow{V}\right| \sin\left(\psi - \psi_{desired}\right) \tag{2.34}$$

$$\dot{d}_{lon} = \begin{cases} \left|\overrightarrow{V}\right| \sin\left(\theta_{desired} - \theta\right), & if \quad \theta_{desired} \geq 0 \\ \left|\overrightarrow{V}\right| \sin\left(\theta - \theta_{desired}\right), & otherwise \end{cases} \tag{2.35}$$

where $\psi_{desired}$ and $\theta_{desired}$ are the desired heading and pitch angle. The position errors (d_{lat} and d_{lon}) are considered on the bounded range with respect to the desired trajectory. These

error bounds are the constraints that define how aggressive the tracking is by using the accelerations.

$$|d_{lat}| \leq d_{b_{lat}} \tag{2.36}$$

$$|d_{lon}| \leq d_{b_{lon}} \tag{2.37}$$

From Fig. 2.26, the position and velocity errors rate can be found as follows:

$$v_{d_{lat}} = \dot{d}_{lat} = v \sin\left(\psi - \psi_{desired}\right) \tag{2.38}$$

$$v_{d_{lon}} = \dot{d}_{lon} = \begin{cases} v \sin\left(\theta_{desired} - \theta\right), & if \quad \theta_{desired} \geq 0 \\ v \sin\left(\theta - \theta_{desired}\right), & otherwise \end{cases} \tag{2.39}$$

In order to obtain the state space, the time derivative of velocity errors ($v_{d_{lat}}$ and $v_{d_{lon}}$) are derived as follows:

$$\dot{v}_{d_{lat}} = v \left(\dot{\psi} - \dot{\psi}_{desired}\right) \cos\left(\psi - \psi_{desired}\right) \tag{2.40}$$

$$\dot{v}_{d_{lon}} = \begin{cases} v \left(\dot{\theta}_{desired} - \dot{\theta}\right) \cos\left(\theta_{desired} - \theta\right), & if \quad \theta_{desired} \geq 0 \\ v \left(\dot{\theta} - \dot{\theta}_{desired}\right) \cos\left(\theta - \theta_{desired}\right), & otherwise \end{cases} \tag{2.41}$$

Since the trajectory information is arbitrary, the change of desired heading and pitch angles ($\dot{\psi}_{desired}$ and $\dot{\theta}_{desired}$) are assumed to be zero. Substitute Eq. 2.47 and Eq. 2.48 with the small angle assumption, so the errors between the current and desired angles are very small ($(\psi - \psi_{desired}) = (\theta_{desired} - \theta) = (\theta - \theta_{desired}) \rightarrow 0$). When all assumptions are applied to Eq 2.38 and Eq. 2.39, the following equations are obtained:

$$\dot{v}_{d_{lat}} = u_{lat} = v\dot{\psi} \tag{2.42}$$

$$\dot{v}_{d_{lon}} = u_{lon} = \begin{cases} v\dot{\theta}, & if \quad \theta_{desired} \geq 0 \\ -v\dot{\theta}, & otherwise \end{cases} \tag{2.43}$$

Therefore, the lateral-direction and longitudinal state space can be presented using a general linear model (Eq. 2.44) as follows:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{2.44}$$

- Lateral-directional state space

$$\begin{bmatrix} \dot{d}_{lat} \\ \dot{v}_{d_{lat}} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} d_{lat} \\ v_{d_{lat}} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_{lat} \tag{2.45}$$

- Longitudinal state space

$$\begin{bmatrix} \dot{d}_{lon} \\ \dot{v}_{d_{lon}} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} d_{lat} \\ v_{d_{lat}} \end{bmatrix} + \begin{bmatrix} 0 \\ p \end{bmatrix} u_{lon}, \quad p = \begin{cases} 1, & if \quad \theta_{desired} \geq 0 \\ -1, & otherwise \end{cases} \tag{2.46}$$

In addition, the control (u) should be identified to change the virtual leader's heading and pitch angle. The lateral and longitudinal acceleration are presented as follows:

$$u_{lat} = v\dot{\psi} \tag{2.47}$$

$$u_{lon} = \begin{cases} v\dot{\theta}, & if \quad \theta_{desired} \geq 0 \\ -v\dot{\theta}, & otherwise \end{cases} \tag{2.48}$$

where $v = \left|\overrightarrow{V}\right|$ is the speed of the virtual leader. Now, the cost function is considered to solve the desired accelerations by the Ricatti equation. The quadratic cost function is introduced as follows:

$$J = \frac{1}{2}\int_{t_0}^{\infty} \left[x^T \mathbf{Q} x + \mathbf{R} u^2(t)\right] dt \tag{2.49}$$

The cost function (J) contains the weighting matrices ($\mathbf{Q}$ and $\mathbf{R}$), the state vector (x) and control (u). In this work, R is assumed to be 1. By using the Riccati equation, the control can be found as follows:

$$\dot{x} = Ax + Bu \tag{2.50}$$

$$u^* = -R^{-1}B^T Px \tag{2.51}$$

P is the solution of Riccati equation which is presented as follows:

$$A^T P + PA - PBR^{-1}B^T P + Q = 0, \quad Q \geq 0, \quad R > 0 \tag{2.52}$$

The state weighting matrix ($\boldsymbol{Q}$) is considered to be adaptive by the term in the diagonal elements:

$$\boldsymbol{Q}_{lat} = \begin{bmatrix} q_{1_{lat}}^2 & 0 \\ 0 & q_{2_{lat}}^2 \end{bmatrix} \geq 0, \quad \boldsymbol{Q}_{lon} = \begin{bmatrix} q_{1_{lon}}^2 & 0 \\ 0 & q_{2_{lon}}^2 \end{bmatrix} \geq 0 \tag{2.53}$$

The way to calculate the adaptive weighting is presented at the end of this section. R_{lat} and R_{lon} are assumed to be 1.

$$R_{lat} = R_{lon} = 1 \tag{2.54}$$

Now, the Riccatti equation is solved by substituting the aforementioned terms as follows:

- Lateral-directional

  By using Eq. 2.51, the lateral control is presented as follows:

$$u^* = -R_{lat}^{-1}\boldsymbol{B}_{lat}^T \boldsymbol{P} x_{lat}, \quad \text{where} \quad \boldsymbol{P} = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} \tag{2.55}$$

  Substitute all matrices to the Riccati equation.

$$\begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} + \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$
$$- \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} 1^{-1} \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} + \begin{bmatrix} q_{1_{lat}}^2 & 0 \\ 0 & q_{2_{lat}}^2 \end{bmatrix} = 0 \tag{2.56}$$
$$\begin{bmatrix} 0 & 0 \\ p_{11} & p_{12} \end{bmatrix} + \begin{bmatrix} 0 & p_{11} \\ 0 & p_{21} \end{bmatrix} - \begin{bmatrix} p_{12}p_{21} & p_{12}p_{22} \\ p_{22}p_{21} & p_{22}^2 \end{bmatrix} + \begin{bmatrix} q_{1_{lat}}^2 & 0 \\ 0 & q_{2_{lat}}^2 \end{bmatrix} = 0 \tag{2.57}$$
$$\begin{bmatrix} -p_{12}p_{21} + q_{1_{lat}}^2 & p_{11} - p_{12}p_{22} \\ p_{11} - p_{22}p_{21} & p_{12} + p_{21} - p_{22}^2 + q_{2_{lat}}^2 \end{bmatrix} = 0 \tag{2.58}$$

Solve for p_{11}, p_{12}, p_{21}, and p_{22}.

$$p_{12} = p_{21} = q_1 \tag{2.59}$$
$$p_{22} = \sqrt{2q_1 + q_{2_{lat}}^2} \tag{2.60}$$
$$p_{11} = q_1\sqrt{2q_1 + q_{2_{lat}}^2} \tag{2.61}$$

Therefore, the optimal control (u_{lat}^*) is the following equation:

$$u_{lat}^* = - \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} \begin{bmatrix} d_{lat} \\ v_{d_{lat}} \end{bmatrix} = - \left[q_{1_{lat}} d_{lat} + \sqrt{2q_{1_{lat}} + q_{2_{lat}}^2} v_{d_{lat}} \right] \tag{2.62}$$

The adaptive gain ($q_{1_{lat}}$) is used with the maximum allowable position error ($d_{b_{lat}}$).

$$q_{1_{lat}} = \left| \frac{d_{b_{lat}}}{d_{b_{lat}} - d_{lat}} \right| \tag{2.63}$$

Eq. 2.62 can be expressed by using Eq. 2.63 as follows:

$$u^*_{lat} = -\left[\left|\frac{d_{b_{lat}}}{d_{b_{lat}} - d_{lat}}\right| d_{lat} + \sqrt{2\left|\frac{d_{b_{lat}}}{d_{b_{lat}} - d_{lat}}\right| + q^2_{2_{lat}}} v_{d_{lat}}\right] \tag{2.64}$$

- Longitudinal

  Since the derivation of the control (u^*_{lon}) for the positive desired pitch angle ($\theta_{desired} \geq 0$) is similar to the lateral-directional control, the derivation of a negative desired pitch angle is presented. Substitute all matrices to the Riccati equation.

$$\begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} + \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$
$$- \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} \begin{bmatrix} 0 \\ -1 \end{bmatrix} 1^{-1} \begin{bmatrix} 0 & -1 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} + \begin{bmatrix} q^2_{1_{lon}} & 0 \\ 0 & q^2_{2_{lon}} \end{bmatrix} = 0 \tag{2.65}$$

$$\begin{bmatrix} 0 & 0 \\ p_{11} & p_{12} \end{bmatrix} + \begin{bmatrix} 0 & p_{11} \\ 0 & p_{21} \end{bmatrix} - \begin{bmatrix} p_{12}p_{21} & p_{12}p_{22} \\ p_{22}p_{21} & p^2_{22} \end{bmatrix} + \begin{bmatrix} q^2_{1_{lat}} & 0 \\ 0 & q^2_{2_{lat}} \end{bmatrix} = 0 \tag{2.66}$$

$$\begin{bmatrix} -p_{12}p_{21} + q^2_{1_{lon}} & p_{11} - p_{12}p_{22} \\ p_{11} - p_{22}p_{21} & p_{12} + p_{21} - p^2_{22} + q^2_{2_{lon}} \end{bmatrix} = 0 \tag{2.67}$$

Solve for p_{11}, p_{12}, p_{21}, and p_{22}.

$$p_{12} = p_{21} = q_1 \tag{2.68}$$

$$p_{22} = \sqrt{2q_1 + q^2_{2_{lon}}} \tag{2.69}$$

$$p_{11} = q_1\sqrt{2q_1 + q^2_{2_{lon}}} \tag{2.70}$$

Therefore, the optimal control (u^*_{lon}) is the following equation:

$$u^*_{lon} = -\begin{bmatrix} 0 & -1 \end{bmatrix} \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} \begin{bmatrix} d_{lat} \\ v_{d_{lat}} \end{bmatrix} = -\left[-q_{1_{lon}} d_{lon} - \sqrt{2q_{1_{lon}} + q^2_{2_{lon}}} v_{d_{lon}} \right] \tag{2.71}$$

The adaptive gain ($q_{1_{lon}}$) is used with the maximum allowable position error ($d_{b_{lon}}$).

$$q_{1_{lon}} = \left| \frac{d_{b_{lon}}}{d_{b_{lon}} - d_{lon}} \right| \tag{2.72}$$

Eq. 2.71 can be rewritten by using Eq. 2.72.

$$u^*_{lon} = \left[\left| \frac{d_{b_{lon}}}{d_{b_{lon}} - d_{lon}} \right| d_{lon} + \sqrt{2 \left| \frac{d_{b_{lon}}}{d_{b_{lon}} - d_{lon}} \right| + q^2_{2_{lat}}} v_{d_{lon}} \right] \tag{2.73}$$

The following conditions are presented to check the stability of the system:

- Controllability: $|\{\boldsymbol{B}, \boldsymbol{AB}\}|$ should not be zero. By using Eq. 2.45 and Eq. 2.46, the determinant of the controllability matrices are -1.

$$|\{\boldsymbol{B}_{lat}, \boldsymbol{A}_{lat}\boldsymbol{B}_{lat}\}| = \left| \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \right| = -1 \tag{2.74}$$

$$\begin{cases} |\{\boldsymbol{B}_{lon}, \boldsymbol{A}_{lon}\boldsymbol{B}_{lon}\}| = \left| \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \right| = -1, \quad if \quad p = 1 \\ |\{\boldsymbol{B}_{lon}, \boldsymbol{A}_{lon}\boldsymbol{B}_{lon}\}| = \left| \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} \right| = -1, \quad otherwise \end{cases} \tag{2.75}$$

- $\boldsymbol{B}_{lat}$ and $\boldsymbol{B}_{lon}$ are not zero. ($\boldsymbol{B} \neq 0$)

- $\boldsymbol{Q} \geq 0$ and $R > 0$: Eq. 2.53 and Eq. 2.54 satisfy these conditions.

- $f(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} \in C^1$ and $f(0) = 0$ should be satisfied.

$$\boldsymbol{A}_{lat}\boldsymbol{x}_{lat} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} d_{lat} \\ v_{d_{lat}} \end{bmatrix} = \begin{bmatrix} d_{lat} \\ 0 \end{bmatrix}, \quad if \quad d_{lat} = 0, \quad f(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2.76}$$

$$\boldsymbol{A}_{lon}\boldsymbol{x}_{lon} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} d_{lon} \\ v_{d_{lon}} \end{bmatrix} = \begin{bmatrix} d_{lon} \\ 0 \end{bmatrix}, \quad if \quad d_{lon} = 0, \quad f(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2.77}$$

In order to implement the result of the derivation for the flight simulation, the linear trajectory is considered as follows:

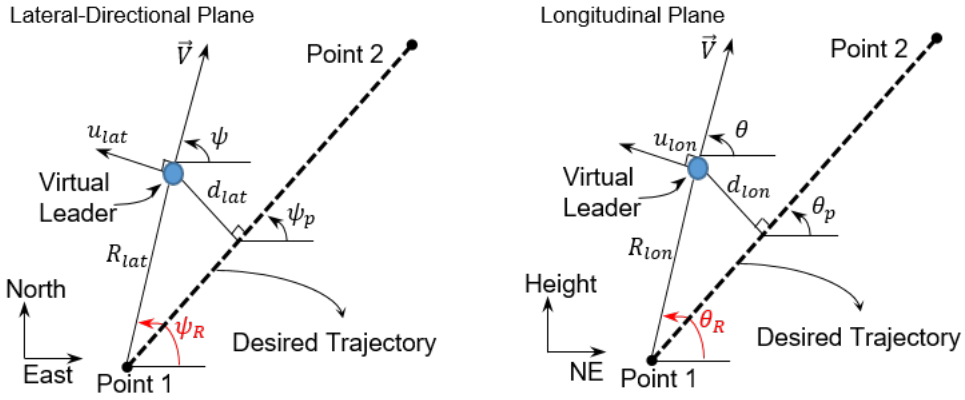


Figure 2.27: LQ guidance geometries for the linear trajectory: lateral-directional (left) and longitudinal (right)

R_{lat} and R_{lon} are the length from waypoint 1 to the virtual leader; ψ_R and θ_R are angles measured from the east axis to R_{lat} and R_{lon}; ψ_p and θ_p are desired heading and pitch angles; $\overrightarrow{V}$ is the virtual leader velocity vector. The position and velocity errors are defined for three dimensional space as follows:

$$d_{lat} = R_{lat} \sin(\psi_R - \psi_p) \tag{2.78}$$

$$\dot{d}_{lat} = v \sin(\psi - \psi_p) \tag{2.79}$$

$$d_{lon} = R_{lon} \sin(\theta_R - \theta_p) \tag{2.80}$$

$$\dot{d}_{lon} = v \sin(\theta - \theta_p) \tag{2.81}$$

Therefore, the desired accelerations (u_{lat} and u_{lon}) can be obtained by Eq. 2.64 and Eq. 2.73. They are translated to the desired heading and pitch angles (Eq. 2.47 and Eq. 2.48) to update the virtual leader position and velocity.

Lastly, the desired trajectory line segment should be updated to follow the race track pattern. The threshold called 'switching distance' is introduced to provide the criteria to update the next desired line segment. The switching distance is usually designed to be larger than the minimum turning radius to achieve a physically feasible trajectory. If the distance between the virtual leader and the next waypoint is smaller than the switching distance (called 'Point based waypoint index update algorithm'), then the waypoint index is updated so that the new line segment can be selected. When the virtual point reaches the last waypoints, it returns to the first index of the given waypoints. From Section 2.2.2.1, the intersection waypoint is added from the predefined waypoints. Therefore, newly added waypoints should be ignored after the first lap of the race track is completed. Figure 2.28 shows the geometries of the switching distance for point-based and line-based waypoint index update algorithm. Fig 2.29

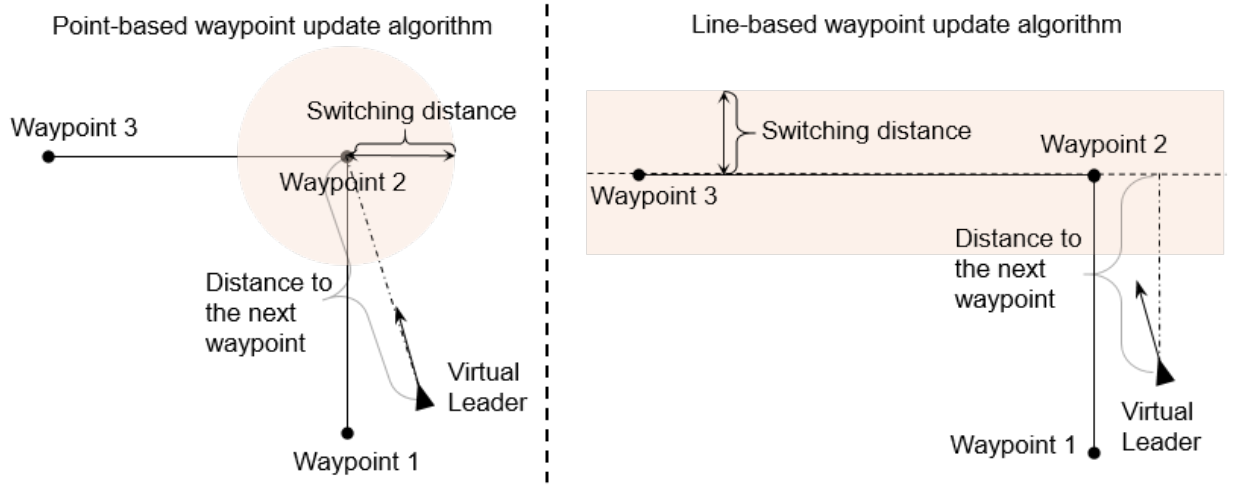


Figure 2.28: Comparison between the point-based and line-based switching distance.

(left) shows the failure of updating the waypoint index. When the line segment length is not long enough to arrive in time to the designated waypoint, the waypoint index cannot be updated because it cannot reach the switching distance zone. In contrast, Fig 2.29 (right)

shows that the line based waypoint index update algorithm resolves these failures. Instead of calculating the distance between the virtual leader and next waypoint, the remaining distance is calculated between the next waypoint line (not a coordinate of waypoint) and the virtual leader position. As a result, the waypoint index can be updated even though the virtual leader does not arrive at the switching distance zone in Figure 2.28 (left).

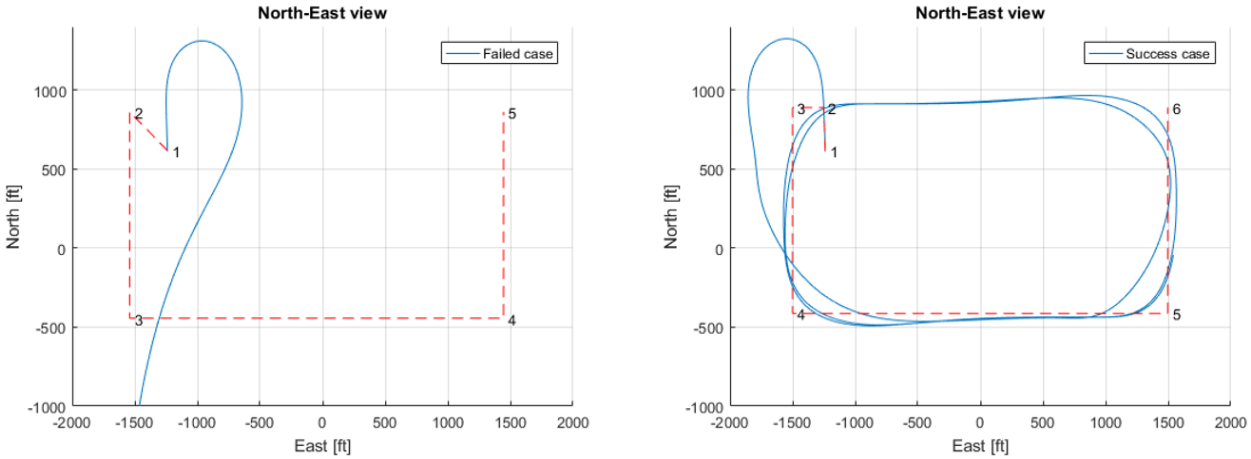


Figure 2.29: Line based waypoint index update algorithm could resolve the issue. (right)

The following figure shows the example of the moving point position in 3D plane. The moving point starts to travel at 300 ft AGL and gradually planned to ascend the altitude to 500 ft AGL.

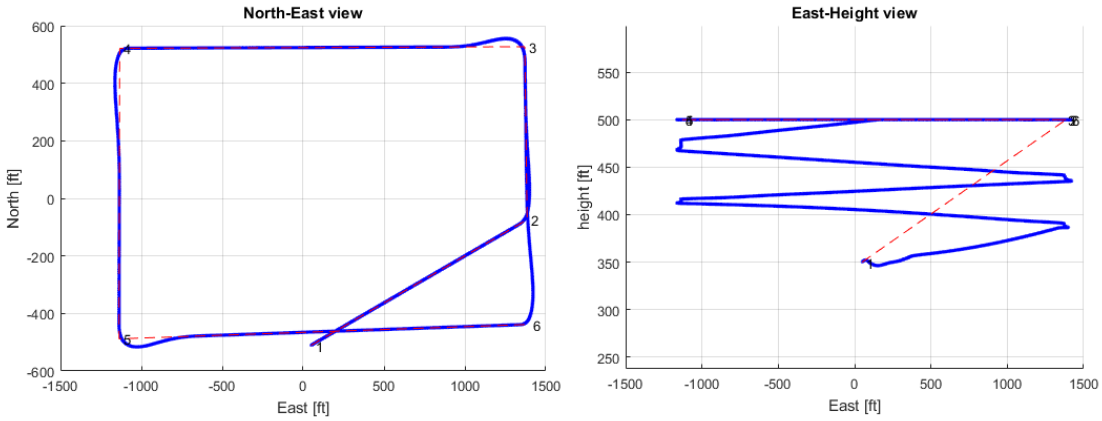


Figure 2.30: The result of the moving point position by using LQ guidance path planning in 3D space

2.2.2.3 Outer Agents: Virtual Point Formation Algorithm

In order to maintain the consistency of the formation shape, the virtual point formation algorithm is used. This algorithm assigns the path to each agent with respect to the virtual leader position by using the desired relative distance. Each agent has the identical heading direction as the virtual leader to keep the coherent formation. Figure 2.31 shows the geometric description of this algorithm. ψ_{ref} is the heading angle of the virtual leader. The following equations calculates each agent formation position by using the desired relative distances and the position and heading angle of the virtual leader:

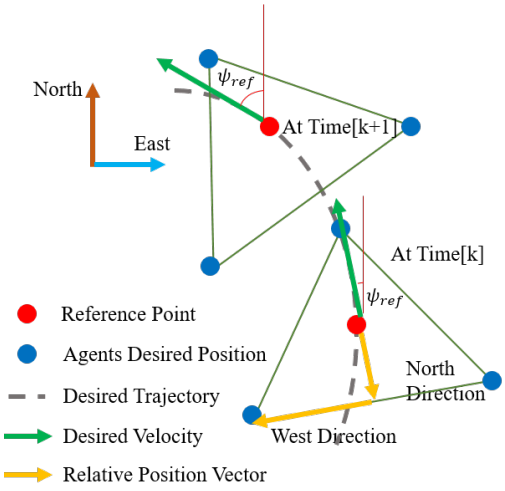


Figure 2.31: Reference Point Formation Algorithm

$$p_{N_i} = p_{N_{RP}} - \|E_i\| \sin\psi_{RP} + \|N_i\| \cos\psi_{RP} \tag{2.82}$$

$$p_{E_i} = p_{E_{RP}} + \|E_i\| \cos\psi_{RP} + \|N_i\| \sin\psi_{RP} \tag{2.83}$$

$$p_{H_i} = p_{H_{RP}} \tag{2.84}$$

where p_{N_i}, p_{E_i}, p_{H_i} are the north, east and height of each agent ($i = 1, \cdots, N_{agent}$) position; $\|E_i\|$, and $\|N_i\|$ are desired relative distances in the east and north direction from the virtual leader; $p_{N_{RP}}$, $p_{E_{RP}}$, $p_{H_{RP}}$ the north, east and height of the virtual leader position; ψ_{RP} is a heading angle of the virtual leader. In this research, the formation height is set by the aircraft altitude when the autopilot is engaged.

2.2.2.4 Inner Agents: Moving Mesh Methods

Moving Mesh Methods (MMM) have been developed to enhance the resolution (or quality) of the solution for partial differential equations (PDE). Huang. et al developed these methods, see Ref. [43]. MMM have been utilized in various fields such as applied mathematics, engi-

neering and biology. The multi-phase flow for a diesel injector nozzle has been investigated by using MMM (Ref. [63]). Incompressible Navier-Stokes equations have been solved by using MMM very efficiently, see Ref. [23]. The glaciers model developed by Oerlemans in 1984 has been solved by the MMM (Ref. [75]). Ref. [56] proposed the modeling of tumor growth using the MMM. The common ground of using the MMM is to help the numerical solution of partial differential equations for complex geometries. In this work, the MMM play a role of creating the path for the multi-agent systems since they generate the node position in a way that they do not collide each other and find the optimal position to make the mesh distribution as uniform as possible. Mesh creation by MMM does not make the nodes overlap or tangle with each other. Nodes organize themselves using energy distribution between mesh elements or nodes. These features are very profound for the multi-agent system to perform formation flight in the perspective of coordinating with other agents safely and efficiently. Unlike the existing work, the agents' position does not have a fixed boundary. For this reason, the path planning of the outer agent should be developed to provide the boundary condition for the moving mesh methods. Hereafter, inner agents should be defined as agents located within the formation shape. For example, inner agents are defined as ones who stay inside of the triangle made by agent 1, 2, and 3, see Fig. 2.32. In this work, moving mesh methods provide the desired path for inner agents by solving moving mesh partial differential equations (MMPDE). In order to apply MMM to the path planning algorithm, the moving mesh nodes are considered as the aircraft position. For generating the physically feasible flight path, 6 DoF dynamic constraints (e.g., the minimum turning radius) are applied to the nodes' movement. Fundamentals of the moving mesh methods begins with understanding the mesh or triangulation with geometric concepts. Mesh or triangulation is the division of the geo-

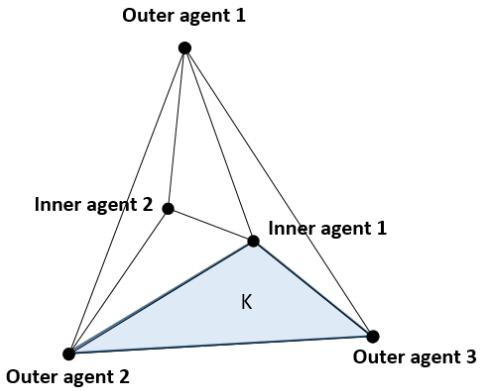


Figure 2.32: Mesh example for the triangle formation

metric area with triangles in two dimension or tetrahedrons in three dimensional space. The following lists are the properties of the meshes:

- There is no empty element. Mesh cannot be degenerate.
- Union of sub-domain in an element (K) is equal to a designated domain ($\bar{\Omega}$)
- The mesh elements cannot be overlapped.
- Boundary of the mesh element (K) is Lipschitz-continuous.
- Any vertices of the mesh cannot be located inside of the mesh or the edge of an element.

Simplexes are introduced to generalize triangles in any dimension to present the meshes in the mathematical description. The mesh element (K) is described as d-simplex which is the convex hull of points (x_i) as follows:

$$K = \left\{ x = \sum_{i=0}^{d} \lambda_i x_i : 0 \leq i \leq 1, i = 0, ..., d, \sum_{i=0}^{d} \lambda_i = 1 \right\} \tag{2.85}$$

Edge matrix is defined by the collection of the relative distance vector of all points (x) in d dimension, see Eq. 2.86. If the meshes are defined in d dimension, there are $d+1$ points to deal with. For example, 2 dimensional meshes has 3 points to describe its simplex which is a triangle. The edge matrix are presented as follows:

$$E_K = [x_1 - x_0, \cdots, x_d - x_0] = [\mathbf{p}_1^K - \mathbf{p}_0^K, \cdots, \mathbf{p}_d^K - \mathbf{p}_0^K] \tag{2.86}$$

where $\mathbf{p}_i^K$ is the position of the aircraft ($\mathbf{p}_i = (p_N^i, p_E^i, p_H^i)^T$); $i = 1, \cdots, n$ is the number of agents. The volume of the mesh element (K) is presented by the edge matrix and the dimension (d) as follows:

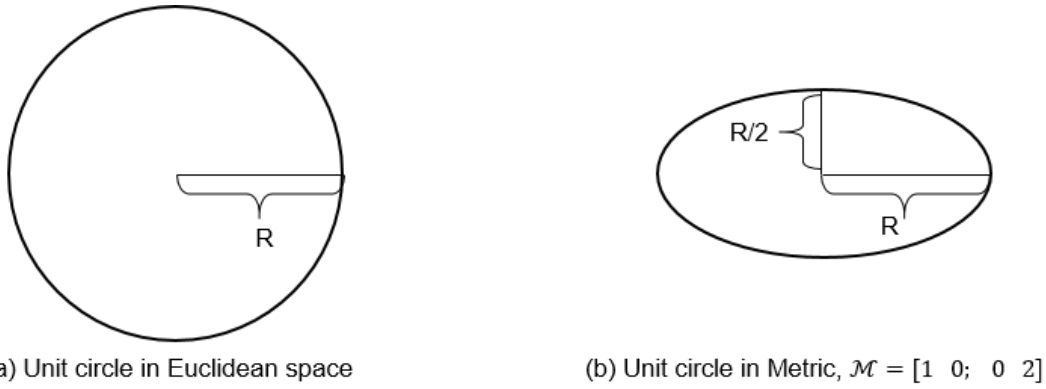
$$|K| = \frac{1}{d!\,|det(E_K)|} \tag{2.87}$$

Equidistribution and alignment of the mesh is derived from simplicial mesh properties. Simplicial mesh consist of d-simplexes and are mapped into the reference mesh element ($\hat{K}$)

by using the affine mapping. The affine mapping is the technique for rotating, scaling, and transforming the geometry in one domain to other domains. In two dimensional space, the reference mesh element ($\hat{K}$) is usually a right triangle or an equilateral triangle. It is also defined by the other shape of the triangle depending on the metric definition. For example, the uniform mesh can be defined in the metric M as follows:

$$M = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \tag{2.88}$$

In this case, the right square mesh is not the uniform in metric M anymore, see Fig. 2.33.



(a) Unit circle in Euclidean space

(b) Unit circle in Metric, $\mathcal{M}$ = [1 0; 0 2]

Figure 2.33: Example of the uniform circle in different metric: Euclidean and Metric, M

In order to implement MMM, the Delaunay triangulation method is used to find the connectivity, boundary or outer nodes and mesh elements. The connectivity shows the collection of each mesh element by the node index. In addition, this method can identify the nodes (e.g., boundary nodes or outer nodes). This is how the outer agents are found in the algorithm so the inner agents can plan their path from MMM. Figure 2.34 shows the example of the Delaunay triangulation.

K_1 through K_5 is the triangular mesh by six agent. The agent position is presented at the right side of Fig. 2.34. The connectivity of Delaunay triangulation is given by the node index as follows:

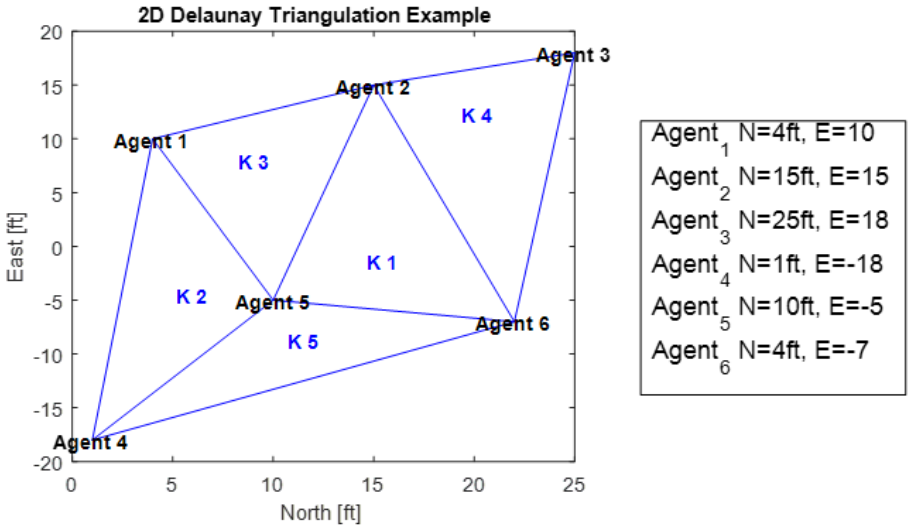


Figure 2.34: 2D Delaunay Triangulation Example Mesh with the position of Agents

$$\text{Connectivity} = \begin{bmatrix} 4 & 6 & 5 \\ 6 & 1 & 5 \\ 5 & 1 & 2 \\ 6 & 4 & 1 \\ 5 & 2 & 3 \end{bmatrix} \tag{2.89}$$

The subscription of agents in Fig. 2.34 is the node index. Now, the two mathematical conditions are introduced to make the uniform mesh based on the mesh information obtained by using the Delaunay triangulation method as follows:

1. All mesh elements should have the identical size. (Condition 1)

$$|K| = \frac{|\Omega|}{N}, \quad \forall K \in T_h \tag{2.90}$$

where $|K|$ is the volume of the mesh element (K); T_h is the defined mesh; Ω is the mesh domain; N is the number of mesh elements.

2. All mesh elements are similar or identical to the reference element ($\hat{K}$). (Condition 2)
   In order to express this condition mathematically, the mesh edge length should have the same ratio comparing to the reference mesh element, see the following equation:

$$\| x_i^K - x_0^K \| = \theta_K^{1/2} \| \xi_i - \xi_0 \|, \quad i = 1, \cdots, d \tag{2.91}$$

   where x_i^K is the coordinate of the mesh nodes, and ξ is the reference element coordinates ($\hat{K}$). θ_K is the ratio which is a positive constant. The affine mapping matrix (F_K) provides the following equation:

$$F_K(\xi_i) = x_i^K, \quad i = 1, \cdots, d \tag{2.92}$$

   Jacobian matrix ($\mathbb{J}_K = F'(\xi_i)$) is utilized as follows:

$$\mathbb{J}_K(\xi_i - \xi_0) = x_i^K - x_0^K, \quad i = 1, \cdots, d \tag{2.93}$$

   Substitute Eq. 2.93 to Eq. 2.91.

$$(\xi_i - \xi_0)^T (\mathbb{J}_K)^T (\mathbb{J}_K)(\xi_i - \xi_0) = \theta_K (\xi_i - \xi_0)^T (\xi_i - \xi_0), \quad i = 1, \cdots, d \tag{2.94}$$

   Using Lemma 4.1.1 from Ref. [43], the equation is simplified as follows:

$$(\mathbb{J}_K)^T (\mathbb{J}_K) = \theta_K I \tag{2.95}$$

   where I is the identity matrix; The sufficient condition is found by using arithmetic-mean and geometric-mean inequality (Ref. [43]):

$$\frac{1}{d}\text{trace}\left(\mathbb{J}_K \mathbb{J}_K^T\right) = |K|^{-\frac{d}{2}}, \quad \text{for any element } K \tag{2.96}$$

   where $\mathbb{J}_K$ is the Jacobian matrix which is defined as $\hat{K}E_K^{-1}$ to apply the affine mapping

from the mesh element (K) to the reference mesh element ($\hat{K}$).

As it is mentioned above, the moving mesh methods distributes the energy equally for all mesh elements. A free energy is defined as follows:

$$I_h = \sum_{\text{all elements } K} |K| G(\mathbb{J}_K, \det(\mathbb{J}_K)), \tag{2.97}$$

$\det(\mathbb{J}_K)$ is the determinant of matrix $\mathbb{J}_K$ and

$$G(\mathbb{J}_K, \det(\mathbb{J}_K)) = \frac{1}{3}\left(\text{trace}(\mathbb{J}_K \mathbb{J}_K^T)\right)^d + \frac{1}{3} d^d \left(\det(\mathbb{J}_K)\right)^2. \tag{2.98}$$

I_h is a Riemann sum of a free energy, see Ref. [41]. By minimizing I_h with respect to the agents' position, all mesh elements should be as uniform as possible and meet two aforementioned conditions. The gradient field of I_h is defined by the mesh velocity to find the new inner agents' position as follows:

$$\frac{dp_N^i}{dt} = -\frac{1}{\tau}\frac{\partial I_h}{\partial p_N^i}, \quad \frac{dp_E^i}{dt} = -\frac{1}{\tau}\frac{\partial I_h}{\partial p_E^i}, \quad \frac{dp_H^i}{dt} = -\frac{1}{\tau}\frac{\partial I_h}{\partial p_H^i}, \tag{2.99}$$

or in a more compact form,

$$\frac{d\vec{p}^i}{dt} = -\frac{1}{\tau}\left[\frac{\partial I_h}{\partial \vec{p}^i}\right]^T, \tag{2.100}$$

where τ is a positive and constant parameter used to adjust a time scale of mesh movement. Eq. 2.98, and the scalar-by-matrix differentiation are used (see Ref. [42]) to find the analytical solution of Eq. 2.97 and Eq. 2.100. As the result, the following equation can be obtained:

$$\frac{d\vec{p}^i}{dt} = \frac{1}{\tau}\sum_{K \in \omega_i} |K| \vec{v}_{i_K}^K, \tag{2.101}$$

where ω_i is the collection of mesh elements having the position of agent i as one of its vertices, i_K denotes the local index of agent i in K, and $\vec{v}_{i_K}^K$ denotes the part of the velocity contributed by an element K to agent i. Eq. 2.101 is valid for all interior mesh nodes. This equation

is very important since it is a weighted sum of the velocity contributed from surrounding elements to i^{th} agent by the volume, $|K|$. Also, this can be expressed mathematically as follows:

$$\begin{bmatrix} \left(\vec{v}_1^K\right)^T \\ \vdots \\ \left(\vec{v}_d^K\right)^T \end{bmatrix} = -GE_K^{-1} + E_K^{-1}\frac{\partial G}{\partial \mathbb{J}}\widehat{E}E_K^{-1} + \frac{\partial G}{\partial r}\frac{\det(\widehat{E})}{\det(E_K)}E_K^{-1}, \tag{2.102}$$

$$\vec{v}_0^K = -(\vec{v}_1^K + \cdots + \vec{v}_d^K), \tag{2.103}$$

where G is defined in Eq. 2.98 and $\partial G/\partial \mathbb{J}$ and $\partial G/\partial r$ are derivatives of G with respect to its first and second arguments; r is defined as $1/3d^d(det(\mathbb{J}))$. Derivatives can be found as

$$\frac{\partial G}{\partial \mathbb{J}} = \frac{2d}{3}\left(\text{trace}(\mathbb{J}_K\mathbb{J}_K^T)\right)^{d-1}\mathbb{J}_K^T,$$
$$\frac{\partial G}{\partial r} = \frac{2}{3}d^d \det\left(\mathbb{J}_K\right).$$

The new position of inner agents can be computed by integrating Eq. 2.101 in respect to time. The following lists are the procedure to operate the moving mesh method algorithm practically:

(i) At the current mesh, compute edge matrices for all elements.

(ii) Compute velocities contributed by each mesh element (K) to its vertices according to Eq. 2.102 and 2.103.

(iii) The mesh velocity for each interior vertex is computed according to Eq. 2.101 by assembling velocities contributed by its neighboring elements to the vertex.

(iv) Each interior vertex is updated by using computed node velocities and a time integration scheme such as Euler's method.

In summary, the moving mesh method has two significant features: nonsingularity and

inter-collision avoidance, see Ref. [98]. If the moving mesh is nonsingular initially, the minimal height and volume exists by the total number of mesh elements, see Eq. 2.101. This feature makes the mesh elements bound by the positive number so that they can be separated at all times. This is also similar to the virtual forces (e.g. attractive and repulsive) since the forces would be balanced when the agents are located as uniformly as possible.

2.2.2.5 Morphing Potential Field Collision Avoidance Path Planning

The morphing potential algorithm provides the direction vector to avoid the static and dynamic obstacles, see Ref. [88]. The general form of the potential field is usually in a circular shape as follows:

$$pf = A \exp\left\{-\frac{d}{\sigma}\right\}^2 \tag{2.104}$$

where pf indicates a potential function; A is an amplitude of a potential function; d is the distance between obstacles and the agents; σ is the radius of the obstacle size. The biggest advantage of the morphing potential field is the ability to avoid collision in advance based on the relative speed between the aircraft and obstacle. The morphing potential function is defined as follows:

$$mpf = \exp\left\{-\Gamma \frac{\left|\overrightarrow{p}^{obs} - \overrightarrow{p}^{UAS} + \overrightarrow{S}\right|}{\sigma}\right\}^2 \tag{2.105}$$

where mpf is the morphing potential function; Γ is the morphing factor; $\overrightarrow{p}^{obs}$ is the position of the obstacle; $\overrightarrow{p}^{UAS}$ is the aircraft position; $\overrightarrow{S}$ is the shifting vector. The shifting term is added to make the potential field efficient. By shifting the center of the obstacle towards the aircraft, the actual path can be planned to avoid the unnecessary cost after the agent passes the obstacle. The shifting term ($\overrightarrow{S}$) is defined by the radius (R, see Eq. 2.109).

$$|\overrightarrow{S}| = d_s = \frac{1}{3}R \tag{2.106}$$

The following equation shows the morphing factor (Γ) for the potential function:

$$\Gamma = (1 - \Gamma_0)\sin^2\eta_v + \Gamma_0; \quad \Gamma_0 \in (0, 1], \quad \eta_v \in [-\pi, \pi] \tag{2.107}$$

where Γ_0 is the extension term of the morphing factor, see Eq. 2.108. The squared sine function is used to obtain the smooth bell shape potential contour when the potential field is morphed. η_v is the error angle between the relative distance and velocity vector. The relative distance and velocity vectors are defined between the agent and the obstacle. The magnitude of the extension term (Γ_0) is defined as follows:

$$\Gamma_0 = \begin{cases} \left(\frac{\sigma}{\sigma+R-d_s}\right)^2, & |\eta_v| < \pi/2 \\ \left(\frac{\sigma}{\sigma+d_s}\right)^2, & \text{else} \end{cases} \tag{2.108}$$

When the error angle is 90°, the morphing factor (Γ) is same as Γ_0. Also, it becomes as same as the radius of the obstacle (σ) if the shifting distance is zero. The radius (R) is defined based on the relative speed as follows:

$$R = \frac{|\overrightarrow{v}_r|^2}{g\tan\phi_{max}} \tag{2.109}$$

where $\overrightarrow{v}_r$ is the relative velocity; g is the gravity acceleration; ϕ_{max} is the maximum allowable roll angle.

Figure 2.35 describes the geometric definition for the morphing potential field. The obstacle velocity is assumed as zero to consider it as the static obstacle. The negative gradient of the morphing potential function is calculated to obtain the direction vector for avoiding the collision. The following equation presents the negative gradient of the morphing potential:

$$\nabla mpf = R_{mpf} = -mpf\frac{2}{\sigma^2}\left\{\overrightarrow{d}_c - \frac{(1-\Gamma_0)\cos\eta_v\left|\overrightarrow{d}_c\right|}{|\overrightarrow{v}_r|}\overrightarrow{v}_r\right\} \tag{2.110}$$

Figure 2.35 (right) shows the negative gradient field of the morphing potential field presented in Figure 2.35 (left).

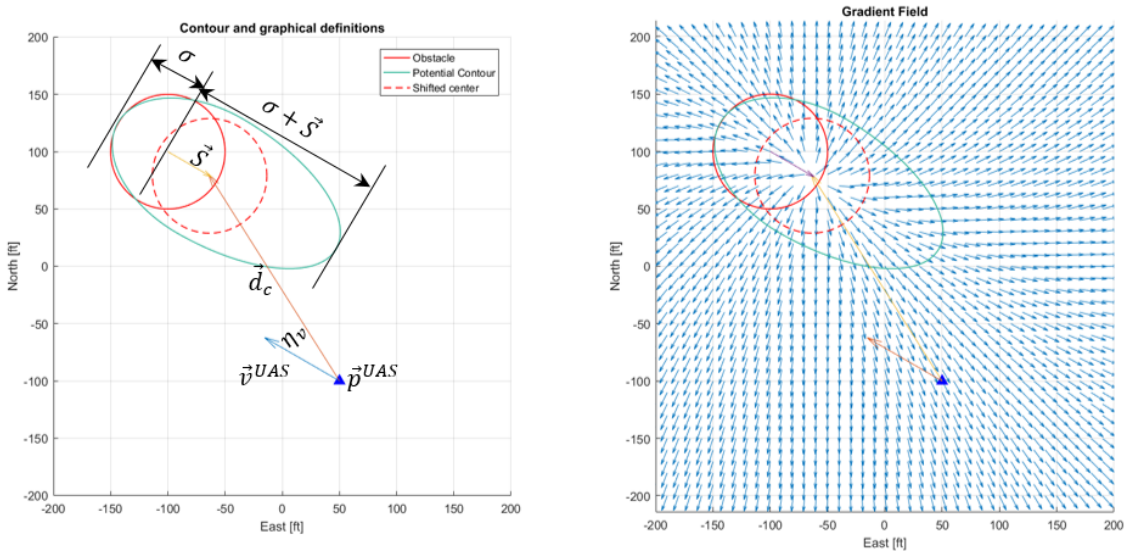


Figure 2.35: Left: Graphical Description of Morphing Potential Contour and Definitions, $\overrightarrow{d}_c$ is the relative distance vector. Right: Negative Gradient field of Morphing Potential. The aircraft velocity is $[42, -72.8, 0]^T$ ft/sec for the north, east, and height in the inertial coordinate. The maximum allowable roll angle is designated as 60°. The radius of the obstacle (σ) is 50 ft.

2.2.2.6 Path Following Algorithm using the Vector Field

In addition to the morphing potential algorithm, the path following logic is required to combine the negative gradient field of the morphing potential. This algorithm has been developed in order to take into account the presence of the wind for UASs for a guidance algorithm, see Ref. [67]. In this work, it is utilized as the path planning algorithm. The following equations are used to determine the vector field.

$$\psi_{app} = \psi^{\infty} \frac{2}{\pi} \tan^{-1}(ky) \tag{2.111}$$

$$R_{traj} = [\cos(\psi_{app} + \psi_{traj}), \quad \sin(\psi_{app} + \psi_{traj})]^T \tag{2.112}$$

where ψ_{app} is an approach heading angle; ψ_{traj} is the desired heading angle; R_{traj} is a unit vector for reaching the desired trajectory; k is the constant for the transition speed from ψ^{∞} to the desired angle; ψ^{∞} is the maximum allowable heading angle for the moving point; As k is larger, ψ^{∞} converges faster to the desired angle. The lateral distance from the trajectory to the aircraft (y) is calculated by the following equation:

$$y = \left(p_N^o - p_N^{traj}\right)\sin\psi_{traj} - \left(p_E^o - p_E^{traj}\right)\cos\psi_{traj} \tag{2.113}$$

where p_N^{traj} and p_E^{traj} are the north and east coordinates of the desired trajectory in the inertial frame. For example, Figure. 2.36 (left) shows the result of the vector field and Figure. 2.36 (right) shows the combination of the vector field and the negative gradient field of the morphing potential.

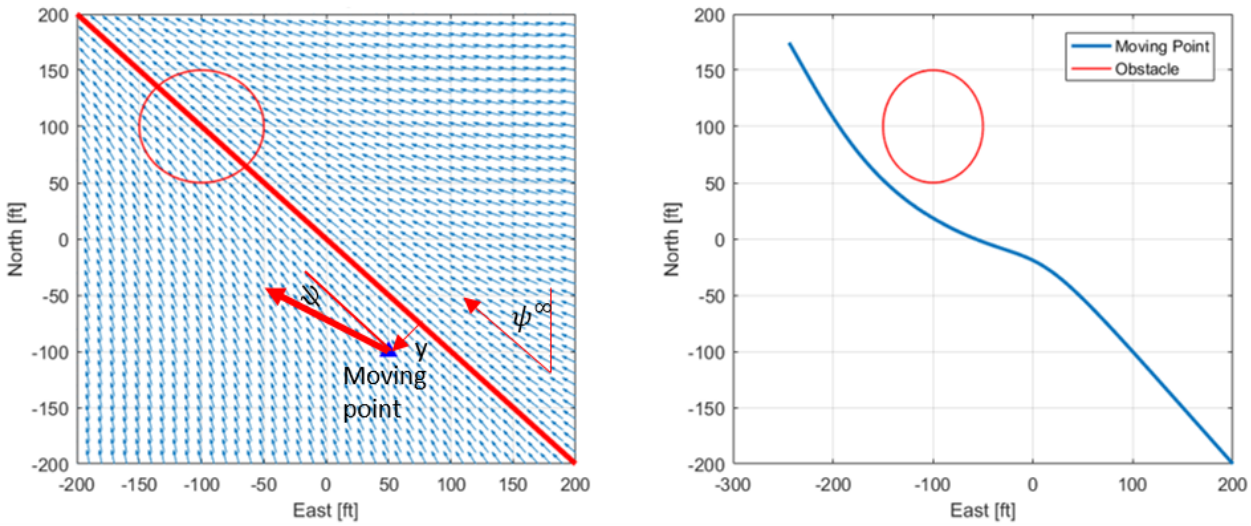


Figure 2.36: The example of the vector field path following

where ψ is the heading angle of the moving point; The maximum allowable heading angle (ψ^{∞}) is assumed to be 45 deg; The desired heading angle (ψ_{traj}) is -45 deg; k is 0.01; The moving point velocity $\overrightarrow{V} = [100, -100]$ ft/sec (north, east component, respectively). The advantage of this algorithm is the robustness towards the random initial condition. No matter where the moving points are initially located, this vector field can direct to the desired

trajectory. Now, the negative gradient field of the morphing potential can be combined with the path following vector field in a weighted way to have the final desired path. If the morphing potential is higher, the gradient of morphing potential has more impact on the final direction vector (see Eq. 2.114) than the path following vector field has. The following equation shows the weighted sum of two fields:

$$R_{Final} = mpf \cdot R_{mpf} + (1 - mpf)R_{traj} \tag{2.114}$$

After the moving point position and velocity is planned, the heading angle change ($\Delta\psi$) is checked and compared with the maximum heading angle.

$$\Delta\psi_{max} = \frac{g \tan \phi_{max}}{|\overrightarrow{V}|}\Delta t \tag{2.115}$$

If the heading angle change is larger than the maximum value ($\Delta\psi_{max}$), then the moving point position and velocity is redefined using the maximum heading angle ($\Delta\psi = \Delta\psi_{max}$).

2.3 Control

This section discusses the linear quadratic regulator (LQR) controller and the nonlinear model predictive controller (NMPC). Based on the flight test result, the elevator correction and updating the trim control are introduced to improve the tracking and reduce the abrupt changes when the autopilot is engaged.

2.3.1 Linear Quadratic Regulator Controller

A Linear Quadratic Regulator (LQR) controller is designed in this work to execute the guidance state commands. LQR controller is established from the linear optimal control theory. The linear time invariant (LTI) state space is applied to the linear dynamics equation

as follows:

$$\dot{x} = Ax + Bu, \quad x(t_0), u(t_0) \quad given \tag{2.116}$$

where x is the state vector in the size of $n \times 1$; u is the control vector in the size of $m \times 1$; $x(t_0)$ and $u(t_0)$ are the initial states and controls. Since the LQR control theory is based on the optimization approach, the cost function or performance index ($\mathbb{J}$) should be defined. In LQR control theory, the quadratic cost function is used, see Eq. 2.117

$$\mathbb{J} = \int_{t_0}^{T} (x^T Q x + u^T R u) dt \tag{2.117}$$

where t_0 is the initial time; T is the final time; Q and R are the weighting matrices. In this work, only Lagrangian type is considered.

As the problem formulation, the regulator problem is defined as follows:

> When the plant has nonzero initial states, the regulator provides inputs ($u^*(t)$) to bring the plant back to states as zeros. Therefore, the regulator problem is to minimize the defined performance index, $\mathbb{J}$ by finding optimal control $u^*(t), \quad t \in [t_0, T]$.

In order to find the solution for minimizing the performance index, the state (x) and control (u) should be zero. This is why this theory is called the regulator. Next, the Lagrange multiplier (λ) is used to derived the three conditions to minimize the cost function.

$$\mathbb{J} = \frac{1}{2} \int_{t_0}^{T} \left[(x^T Q x + u^T R u) + 2\lambda^T (Ax + Bu - \dot{x}) \right] \tag{2.118}$$

$$d\mathbb{J} = \left(\frac{\partial \mathbb{J}}{\partial x} \right) dx + \left(\frac{\partial \mathbb{J}}{\partial u} \right) du + \left(\frac{\partial \mathbb{J}}{\partial \lambda} \right) d\lambda = 0 \tag{2.119}$$

For the first condition, the partial derivative in respect to u is considered.

$$\frac{\partial \mathbb{J}}{\partial u} = \int_{t_0}^{T} \left\{ \frac{\partial}{\partial u} \left(\frac{1}{2} u^T R u + \lambda^T B u \right) dt \right\} = \int_{t_0}^{T} (Ru + B^T \lambda) dt = 0 \tag{2.120}$$

$$Ru + B^T\lambda = 0 \tag{2.121}$$

$$\boxed{u = -R^{-1}B^T\lambda} \tag{2.122}$$

Next, the partial derivative in respect to λ is considered.

$$\frac{\partial \mathbb{J}}{\partial \lambda} = 2\int_{t_0}^{T} \frac{\partial}{\partial \lambda}\left(\lambda^T(Ax + Bu - \dot{x})\right) dt \tag{2.123}$$

$$= Ax + Bu - \dot{x} = 0 \tag{2.124}$$

$$\boxed{\dot{x} = Ax + Bu} \tag{2.125}$$

Lastly, if we take the partial derivative of the performance index in respect to the state (x), the result is the following equation.

$$\boxed{\dot{\lambda} = -A^T\lambda - Qx} \tag{2.126}$$

Now, the symmetric matrix P is used to linearly transform the Lagrange multiplier to the state ($\lambda = Px$). As the result of combining condition 3 and 2, the Ricatti equation is found as follows:

$$-\dot{P} = 0 = PA + A^TP - PBR^{-1}B^TP + Q \tag{2.127}$$

Since the regulator makes all states zero (or make the system in the steady state), this leads to $\dot{P}$ becoming zero. Therefore, the optimal control (u^*) can be expressed as follows:

$$u^*(t) = -R^{-1}B'Px(t) \tag{2.128}$$

Using the generated guidance outputs ($V_{T_{cmd}}$, ϕ_{cmd}, θ_{cmd}, β_{cmd}), the role of the controller is to compute the desired control surface deflections to follow the guidance commands. In this work, the decoupled LQR controllers has been designed by separating the lateral and

longitudinal controllers. The state space obtained from ***Appendix A*** is used to formulate the LQR gain. The regulator terms are added to solve the non-zero set point problem. The following equation shows the augmented linear state space and states for each LQR controller.

$$\dot{\mathbf{x}}_{aug_{lat}} = A_{aug_{lat}}\mathbf{x}_{aug_{lat}} + B_{aug_{lat}}\mathbf{u}_{aug_{lat}} \tag{2.129}$$

$$\dot{\mathbf{x}}_{aug_{lon}} = A_{aug_{lon}}\mathbf{x}_{aug_{lon}} + B_{aug_{lon}}\mathbf{u}_{aug_{lon}} \tag{2.130}$$

All states denoted here are *perturbed states*. For calculating the regulator terms, they are in total states so that the difference between guidance commands and states are considered as perturbed values.

$$\mathbf{x}_{lat} = \begin{bmatrix} \beta \\ \phi \\ p \\ r \\ \int \beta_{cmd} - \beta \\ \int \phi_{cmd} - \phi \end{bmatrix} \quad \mathbf{u}_{lat} = \begin{bmatrix} \Delta\delta_a \\ \Delta\delta_r \end{bmatrix}, \quad \mathbf{x}_{lon} = \begin{bmatrix} V_T \\ \alpha \\ \theta \\ q \\ \int V_{T_{cmd}} - V_T \\ \int \theta_{cmd} - \theta \end{bmatrix} \quad \mathbf{u}_{lon} = \begin{bmatrix} \Delta\delta_T \\ \Delta\delta_e \end{bmatrix} \tag{2.131}$$

The following equations are presenting the numerical result of the augmented linear state space.

$$A_{lat} = \begin{bmatrix} -0.44127 & 0.5449 & -0.0074 & -0.9847 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ -10.7386 & 0 & -20.9688 & 2.9327 & 0 \\ 10.0800 & 0 & -1.2881 & -0.6602 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \end{bmatrix}, \quad B_{lat} = \begin{bmatrix} 0 & 0.1616 \\ 0 & 0 \\ 140.5772 & 0.6242 \\ -6.7747 & -6.4997 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{2.132}$$

$$A_{lon} = \begin{bmatrix} -0.2030 & -7.7004 & -32.1874 & 0 & 0 \\ -0.0177 & -9.6416 & -0.0151 & 0.9632 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0.2003 & -121.3565 & 0.0093 & -5.4812 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}, \quad B_{lon} = \begin{bmatrix} 3.0391 & -0.8475 \\ 2.327 \cdot 10^{-5} & -0.3067 \\ 0 & 0 \\ -1.7311 & -40.8156 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{2.133}$$

With this state space model, the Q and R weighting matrices have been designed for each controller. By considering the behavior of the DG808 from the simulation result and parametric studies, the following Q and R matrices were designed.

$$\mathbf{Q}_{lat} = diag\,(131.31,\, 0.912,\, 2.05,\, 2.05,\, 11.46,\, 11.46) \tag{2.134}$$

$$\mathbf{R}_{lat} = diag\,(50,\, 300) \tag{2.135}$$

$$\mathbf{Q}_{lon} = diag\left(0.00012,\, 3 \cdot 10^{-6},\, 0.099,\, 0.003,\, 0.003,\, 15\right) \tag{2.136}$$

$$\mathbf{R}_{lon} = diag\,(300,\, 300) \tag{2.137}$$

where $diag(a,\, b,\, \cdots,\, d)$ indicates the diagonal matrix in $\mathbb{R}^{n \times n}$ whose diagonal have numbers such as $a,\, b,\, \cdots,\, d$. With all the information, the LQR gain was computed as shown in Table 2.1.

Table 2.1: Lateral and Longitudinal LQR gain

Lateral mode $\begin{bmatrix} 0.0065 & 0.0057 & -0.011 & -0.001 & -0.0015 & -0.198 \\ 4.3 \cdot 10^{-5} & 0.22 & -0.31 & -0.02 & -0.003 & 0.103 \end{bmatrix}$
Longitudinal mode $\begin{bmatrix} 1.016 & 0.82 & 0.128 & -0.229 & -0.152 & -0.45 \\ 0.084 & -0.083 & -0.00124 & -0.068 & -0.18 & 0.062 \end{bmatrix}$

2.3.2 Elevator correction for improving altitude tracking

Due to the nature of the aircraft dynamics, the roll motion causes a reduction of lift when the aircraft starts to turn from the steady state level wing condition. Due to the decreasing lift, the altitude is decreased during the roll maneuvers, see Fig. 2.37. In order to improve

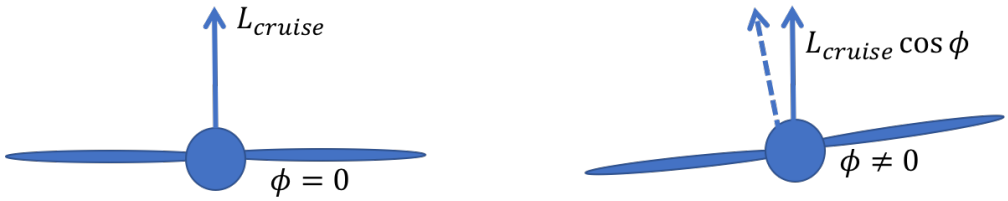


Figure 2.37: Reduced lift force due to the rolling motion.

the altitude tracking, elevator correction is implemented. The elevator correction is derived from the lift equations.

$$L_{cruise} = C_L \bar{q} S, \quad L_{roll} = C_L \bar{q} S \cos\phi \tag{2.138}$$

Now, the lift at cruise is equated to the lift at the roll motion. Then, the elevator correction can be found by compensating the lift difference.

$$L_{cruise} - \frac{L_{roll}}{\cos\phi} = L_{\Delta\delta_e} \tag{2.139}$$

$$C_L \bar{q} S \left(1 - \frac{1}{\cos\phi}\right) = C_{L_{\delta_e}} \bar{q} S \Delta\delta_e \tag{2.140}$$

$$\Delta\delta_e = \frac{C_L}{C_{L_{\delta_e}}} \left(1 - \frac{1}{\cos\phi}\right) \tag{2.141}$$

We can consider $\frac{C_L}{C_{L_{\delta_e}}}$ as the constant K. Therefore, the following equation represents the elevator correction using the roll angle.

$$\Delta\delta_e = K \left(1 - \frac{1}{\cos\phi}\right) \tag{2.142}$$

Figure 2.38 shows the impact of this algorithm. The elevator commands increase whenever the aircraft turns the corner. The result when the correction is not applied shows the altitude drop and also pitch angle decrease. After the correction is applied, the altitude and the pitch angle changes are smoothed out.

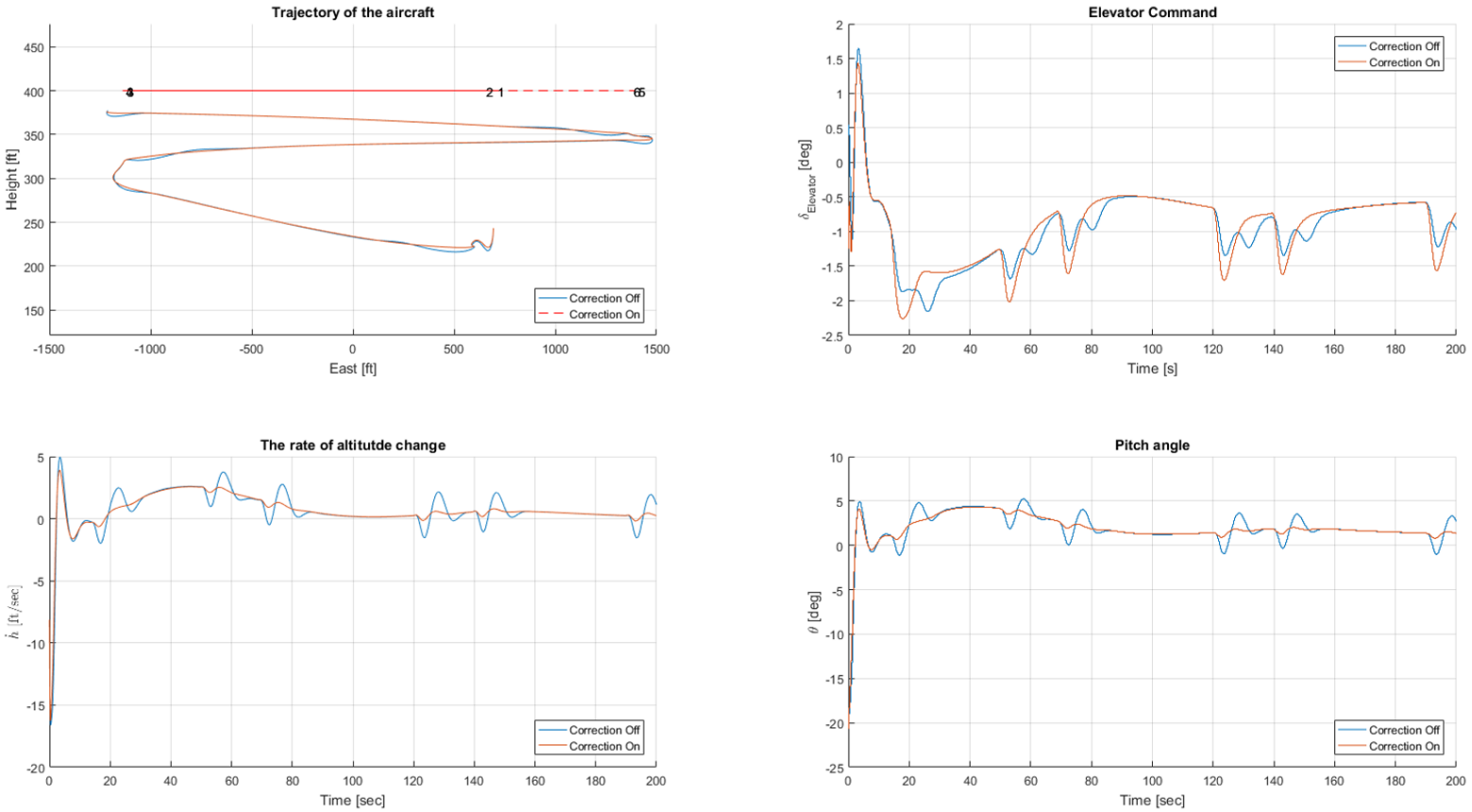


Figure 2.38: Comparison between the results of when the elevator correction algorithm is on and off, $K = -0.5$

2.3.3 Trim States and Control Surface Updates During the Flight

When the controller is implemented into on-board avionics, the trim states and trim conditions are set to fixed constant values. When the autopilot engages, the controller does not have any information about the trim values and states that the pilot uses. The drawback of missing trim control and states causes the dramatic changes in the control outputs: throttle, elevator, aileron, and rudder. Figure 2.39 shows the probability density function for controls usage of three difference pilots during the flight test. As Fig. 2.23 shows, each pilot uses

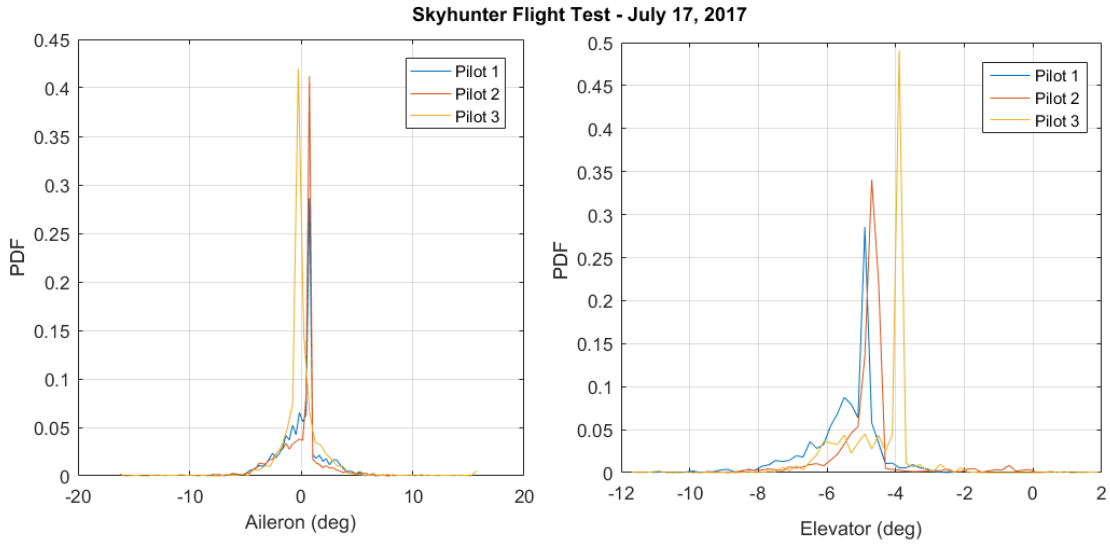


Figure 2.39: The probability density function for the aileron and elevator usage of three different pilots.

the different amount of control deflections for the Skyhunter. When the pilot engages the autopilot, it is very difficult to engage at the exact trim states and controls based on the visual cues. Since the pilot uses difference combinations of throttle and elevator to trim the aircraft, the abrupt control commands are induced when the pilot engages the autopilot. The solution of this drawback is proposed to implement the adaptive trim states and controls. Figure 2.40 shows the behavior of the controller between the fixed and adaptive trim values. With the fixed trim values, the control surfaces moves aggressively and generates large amounts of control deflections to correct the error. However, the adaptive trim value algorithm could alleviate the abrupt changes in the control outputs.

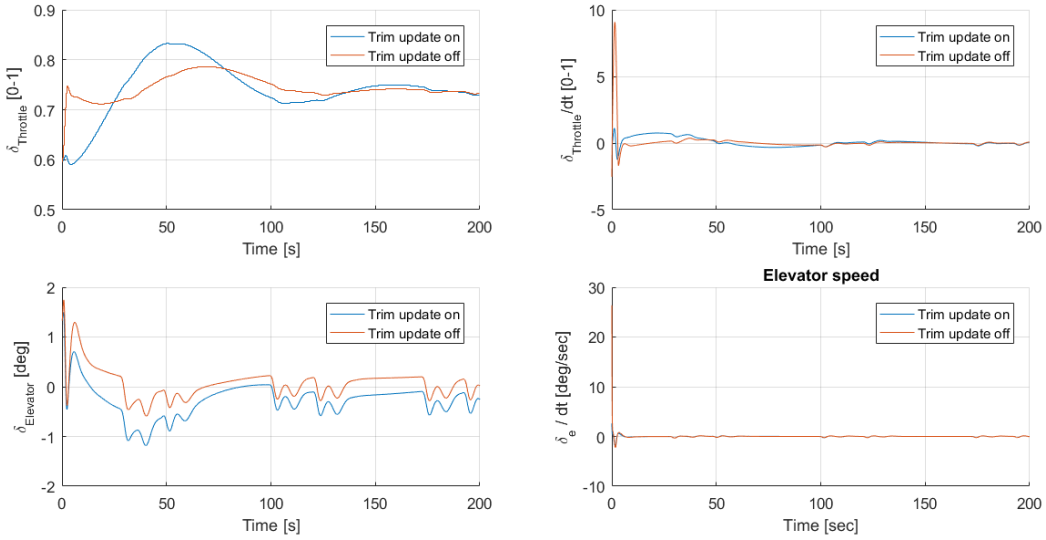


Figure 2.40: Comparison between the fixed trim value and adaptive trim values for throttle and elevator.

2.4 Nonlinear Model Predictive Controller

In this section, Nonlinear Model Predictive Controller (NMPC) is presented. This control is the variation of Model Predictive Control which uses the linear model. Since NMPC uses the nonlinear model, the prediction of the plant or system can be improved and described more delicately. NMPC requires the sequence of the optimal control outputs based on the physics based model. The sequence of control outputs are called the horizon which relates a certain amount of time towards the future. In this work, the length of the horizon is 20 samples which predicts within 1 second of the future with 0.05 seconds of the sampling time. Besides of the optimization of the control outputs, NMPC can also consider the control surface constraints. Figure 2.41 shows the control diagram of NMPC.

Since the multi-agent system has been designed by using the virtual leader scheme, the decentralized GNC algorithms are implemented in this work. The operation of centralized NMPC depends on the computation capability of the processors on board. In addition, this approach is prone to single-point failure. Practically, the centralized NMPC is very infeasible to operate as its workload increases exponentially as the number of agents increases. Once the

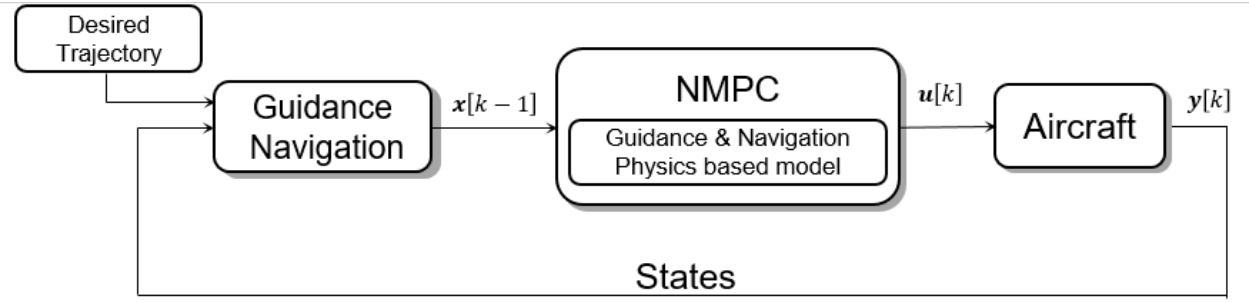


Figure 2.41: A guidance, navigation, and augmented NMPC

horizon of moving points are given to NMPC, it generates the sequence of control deflections to follow the given moving points by considering the control surface constraints, see Table. 2.2.

Table 2.2: Control surfaces constraints

Control surfaces	Range	Control surface	Range
$\delta_{T_{min}}$	10%	$\delta_{a_{min}}$	-25°
$\delta_{T_{max}}$	100%	$\delta_{a_{max}}$	25°
$\delta_{e_{min}}$	-15°	$\delta_{r_{min}}$	-15°
$\delta_{e_{max}}$	15°	$\delta_{r_{max}}$	15°

The decentralized implementation of GNC is illustrated in Fig. 2.42.

An integrated guidance and NMPC scheme (Ref. [28]) has demonstrated good performance for agents with high speeds and high inertia during tight turns. The implemented guidance allows all agents to follow the time-varying moving points (see Ref. [27, 29, 74] for the detailed development). In order to optimize the control deflections, the cost function is defined, see Eq. 2.143

$$\mathbf{J}\left(\boldsymbol{e},\boldsymbol{u}\right)=\sum_{k=1}^{N}\boldsymbol{e}_{k+1}^{T}\boldsymbol{Q}_{k+1}\boldsymbol{e}_{k+1}+\boldsymbol{u}_{k}^{T}\boldsymbol{R}_{k}\boldsymbol{u}_{k} \tag{2.143}$$

where the aircraft discrete dynamics is defined as $\boldsymbol{x}_{k+1}=f_d\left(\boldsymbol{x}_k,\boldsymbol{u}_k\right)$. $\boldsymbol{Q}_k$ and $\boldsymbol{R}_k$ are the weighting matrices which are positive definite. N is the length of the horizon. The additional

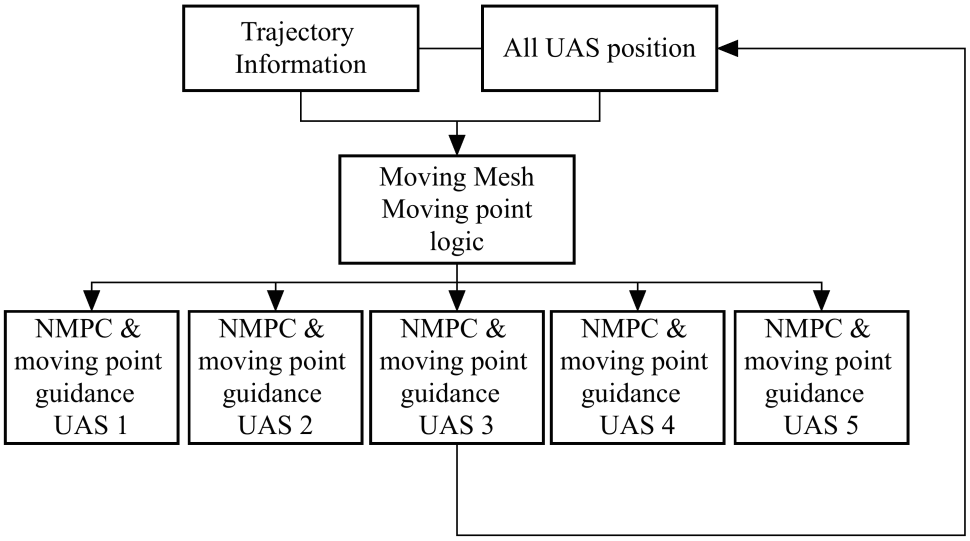


Figure 2.42: The process of operating decentralized NMPC and centralized moving points guidance

cost function ($Y(\boldsymbol{u}, \lambda)$) is defined to included the control deflection constraints:

$$Y(\boldsymbol{u}, \lambda) = \mathbf{J}(\boldsymbol{u}) - \lambda^T \bar{c}(\boldsymbol{u}) \tag{2.144}$$

where $\bar{c}(\boldsymbol{u})$ is the updated constraint vector based on the currently violated constraints. For the optimization methods, the Sequential Quadratic Programming (SQP) is used from Ref. [27]. SQP acts only when the constraints are violated. Otherwise, the optimization problem can be reduced to a Newtonian type search. Once the constraints are violated, the correction terms are computed for the length of the horizon, $i = 1, 2, \cdots, N$. Therefore, the optimized control deflection are defined by the sum of current control outputs and the correction terms.

$$\begin{bmatrix} \boldsymbol{u}^{i+1} \\ \lambda^{i+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{u}^{i} \\ \lambda^{i} \end{bmatrix} + \begin{bmatrix} \Delta\boldsymbol{u}^{i} \\ \Delta\lambda^{i} \end{bmatrix} \tag{2.145}$$

The correction terms are calculated by using the following equation:

$$\begin{bmatrix} \nabla^2 \mathbf{J}\left(\boldsymbol{u}^i\right) & -\boldsymbol{A}^T\left(\boldsymbol{u}^i\right) \\ \boldsymbol{A}\left(\boldsymbol{u}^i\right) & 0 \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{u}^i \\ \Delta \lambda^i \end{bmatrix} = \begin{bmatrix} \nabla \mathbf{J}\left(\boldsymbol{u}^i\right) + \boldsymbol{A}^T\left(\boldsymbol{u}^i\right) \lambda^T \\ \bar{c}\left(\boldsymbol{u}\right) \end{bmatrix} \tag{2.146}$$

where $\boldsymbol{A}^T$ is the Jacobian of the constraints vector ($\bar{c}$); $\nabla \mathbf{J}\left(\boldsymbol{u}^i\right)$ and $\nabla^2 \mathbf{J}\left(\boldsymbol{u}^i\right)$ are the gradient and Hessian of the cost function ($\mathbf{J}$).

Chapter 3

Autonomous System Testbed Development

In this section, Hardware-in-The-Loop (HiTL) testbed is introduced. The testbed development has been done with the collaboration with the Department of Electrical Engineering in KU, see Ref. [97]. Before the actual flight tests are conducted, the systems including the hardware and software should be verified for the purpose of evaluating safety, risk, and cost related to the flight test operation. HiTL test involves the flight simulation with nonlinear 6 DoF equations of motion using the actual onboard hardware and software. This has the profound advantages of identifying possible failures in the hardware and software that could not be seen or discovered in the SIMULINK environment. The important point for HiTL testing is that it is operating the systems as close as possible to actual flight test conditions. A 900 MHz telemetry radio is used to test communications between the aircraft avionics and the ground station system. The following sections are discussing the details of the HiTL process.

3.1 Hardware-in-The-Loop Testbed

HiTL testing requires all hardware and software that would be used in the flight test. The purpose of this test is to verify whether all system elements work correctly and flawlessly

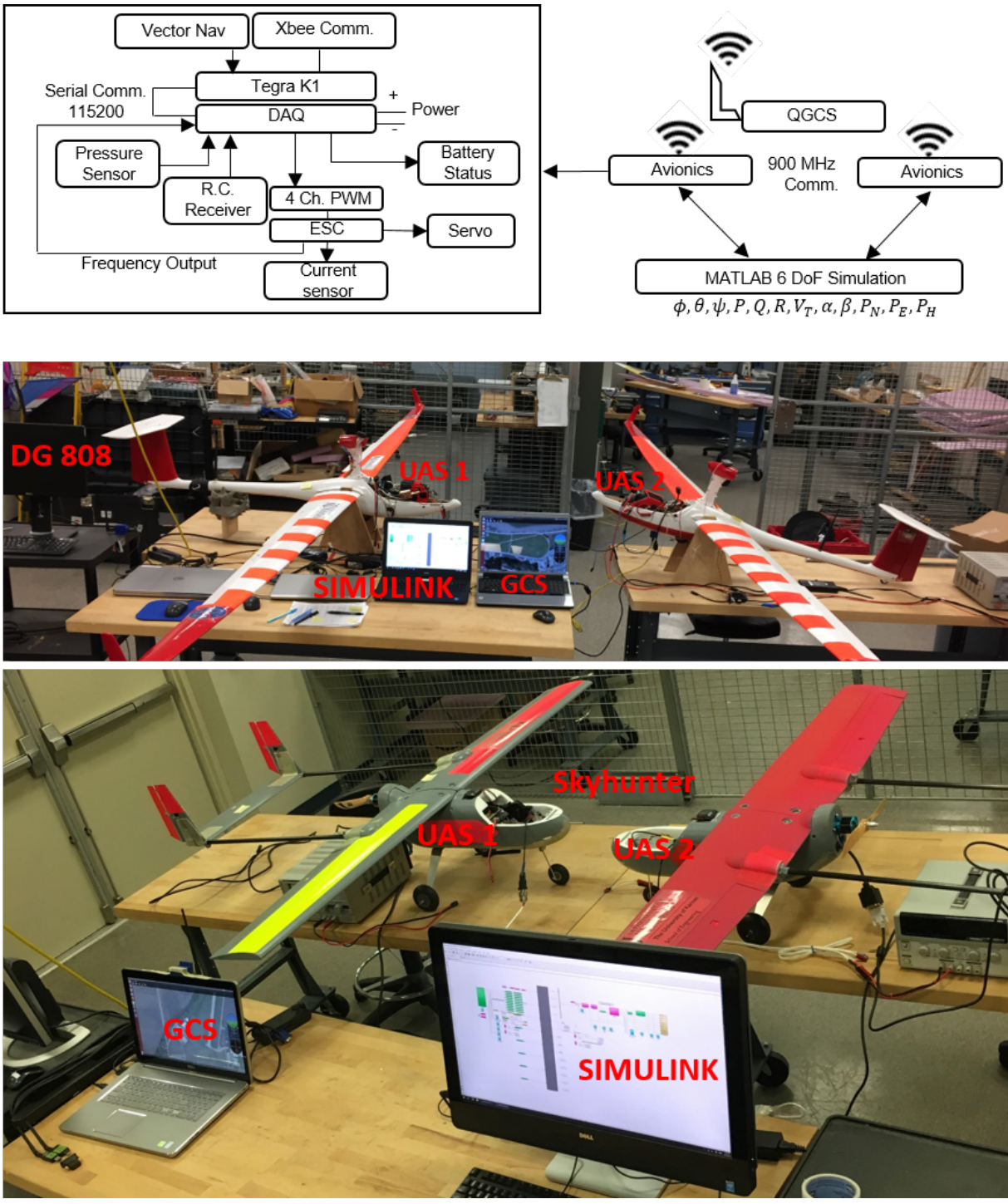


Figure 3.1: Hardware-in-The-Loop Schematic

before the flight test is conducted. Figure. 3.1 shows the HiTL setup. The two platforms (top: DG808 and bottom: Skyhunter) are sitting on the bench and are connected to the laptop which runs the SIMULINK model. Nonlinear 6 DoF equations of motion are simulated by SIMULINK model to generate the aircraft states required for the guidance and control systems in the onboard avionics. The Ground Control System (GCS) communicates via 900 MHz radio to each aircraft. The GCS plays the role of observing outputs of the controller, uploading waypoints to the aircraft, and displaying the tracking of the aircraft.

The procedure of the HiTL test is designed as close as possible to the actual flight test to verify the system properly and thoroughly. For example, the waypoints are known in the simulation environment. However, the waypoints should be sent to the onboard avionics by GCS when the flight test is conducted. HiTL provides a great opportunity to test this procedure. In addition, the actual communication radios are used to help ensure full system integration. For example, during HiTL testing, an issue in the software algorithm was found when the radio was physically disconnected. In addition, the setting of HiTL testing is universal so any different type of platforms can be tested.

Another example of discovering potential malfunction is the communication algorithm failure. When the communication cable is intentionally disconnected, the communication program is terminated immediately and it never reconnects to the communication despite the physical connection being recovered. In actual flights, the physical connection of a communication module can be lost due to the vibration of the airframe. If the communication link does not reestablish itself, the aircraft is very prone to accidents and stall if the vehicle is in the autonomous flight mode. During HiTL testing, this potential risk can be identified and the algorithm can be modified to restart the communication link whenever physical connection is available again. The communication loss problem motivated the development of the algorithm for the recovery procedure during the multi-agent formation flight. Loss of communication during formation flight can cause collision or risky situations to occur. To prevent this, a wait time was added to the algorithm in order to provide time to recover the

communication and continue the formation flight. If the communication never comes back, the aircraft transfers to the solo flight mode to provide the reaction time for the pilot and prevent stalling the aircraft. In the following section, the details of hardware and software implementation are discussed.

3.2 Hardware and Software Implementation

In this section, the avionics and software implementation are discussed. For autonomous flights, the sensors, microprocessor, and data acquisition board are essential. For the modular implementation of these devices in software, Robot Operating System (ROS) is used.

3.2.1 Hardware

Two main processor boards are used for the hardware: a Tegra K1 and a custom built Data Acquisition Board (DAQ), see Ref. [97]. For the communication, an Xbee 900 MHz device is used with typical UASs state sensors, see Fig. 3.1.

- **Sensors**: State sensors include GPS (Global Positioning System), IMU (Inertial Measurement Unit) and a dynamic pressure sensor. For the GPS and IMU measurements, a VectorNAV VN-200 module is used. An AMS 5812 pressure sensor is used for measuring the dynamic pressure (airspeed) via a pitot tube.

- **Tegra K1 and Odroid**: This is a powerful processor containing four ARM Cortex-A15 cores running at 2.3 GHz, and has 192 Kepler based GPU cores. The VectorNAV sensor and the DAQ board are connected to this board. The GNC algorithm runs as a ROS node on the Tegra K1. Additionally, Odroid is used for Skyhunter specifically to reduce the weight. The software is available for various platforms.

- **DAQ**: This board acts as an interface for data exchange between the sensors and the Tegra. The board is connected to all sensors except the GPS and IMU. It collects

data from these sensors and sends it to Tegra through serial communication. The DAQ board also receives the controller outputs from Tegra and sends them to the servos. This board houses an Arduino Mega processor that is its primary computational platform.

- **Communication device**: A telemetry radio device Xbee Pro 900 XBP9B-DMST-002 is used as the main communication transceiver for exchanging data between the ground control station and both aircraft. Communication reliability and robustness are the most important factors for maintaining formation in cooperative autonomous flight. The Xbee supports mesh networking that enables efficient and fast data packet transfers in real time. Experiments show that the maximum round trip time taken by two aircraft in exchanging data is 2ms.

- **Remote Control Receiver**: This device accepts the signal from the remote control and sends the PWM signals to the Electric Speed Controller (ESC) to actuate the control surface servos and motor. In addition, it has an auxiliary channel that is used to change the flight mode from manual flight by the pilot to the autonomous flight mode.

3.2.2 Software

ROS plays a very significant role in the software implementation. Due to various features of ROS such as ad-hoc network, modular scheme using nodes, and so on, a large amount of code can be managed and operated with using simple procedures. A "node" in ROS is an element executing one program. Thus, there exists separate nodes for each functionality such as IMU, GNC, and communication as examples. If designers wanted to add any other function, (s)he can add more nodes to perform the desired functionality. ROS has the feature to transfer data between nodes through the topic using messages. For the detail of topics, see Fig. 3.2.

- **Operating System** : Tegra runs on Linux operating system (Ubuntu 14.04 LTS). This is also because ROS has its most stable version in Ubuntu 14.04 LTS. Linux systems have many advantages when operating embedded systems such as the avionics in this work. It is an open source operating system and is very efficient and easy to update the kernal to support hardware chips such as I2C and SPI devices.

- **GNC source code** : For GNC algorithms, the Code generation toolbox in MATLAB is used to generate the source code in C language to implement via ROS. When the code is generated, the configuration setting should be carefully selected to ensure that the correct code is produced, such as the fixed time solver setting rather than using the variable time setting. Due to ROS running at certain frequency, it is very important to set the configuration and the entire SIMULINK simulation development under the discrete time scheme with the fixed time step solver.

- **Ground station software** : Mavlink protocol is used to perform data exchange between ground station and both aircraft. Q Ground Control (Q-GC) is used for the ground station software. It is the open source software and customized for the autonomous flight by the Flight System Team at the University of Kansas.

- **Data acquisition and transfer software**: The Arduino Mega runs the program that enables communication between the Tegra, the sensors, and the DAQ board. It uses I^2C, interrupts, and servo library from Arduino IDE to read sensor data and generate servo commands.

- **ROS** : A powerful meta-operating system Robot Operating System (ROS) framework is used as the main communication server that manages all the individual sensors and GNC sub-routines. Figure 3.2 shows the software architecture.

    1. Controller node: In this node, the GNC algorithm was implemented using auto-generated code in C language. It publishes the autopilot output (throttle, elevator,

aileron, and rudder) to "`servo_op`" as the message and sends to Arduino node. In this node, all information is published in the "`data`" topic to send to ground station and log all data.

2. Arduino node: This node communicates with the DAQ board. Once the controller publishes the message, this node subscribes the autopilot output, converts the signal to PWM and sends it to the DAQ board. Also, it publishes the message to "`pwm`" topic with the autopilot switch, pressure sensor data, voltage of battery, and remote control command values.

3. VectorNav node: This node is responsible for data acquisition from the IMU sensor. All IMU data publishes to "`ins`" topic. The message includes GPS position, GPS velocity, attitude, angular rate, and accelerometer measurements.

4. Ground station node: This node deals with the Xbee 900 MHz communication. It receives data from the aircraft and publishes it to the ground station and topic "`data`". It also obtains waypoint information from the ground station and publishes it to the "`way`" topic and sends it to the controller node.

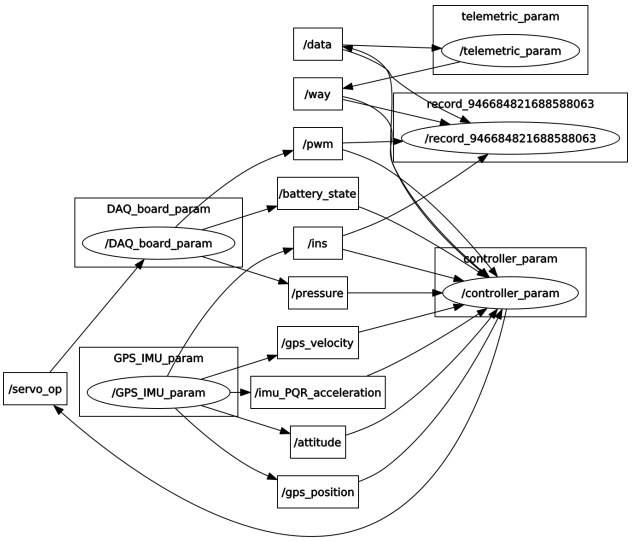


Figure 3.2: ROS based software architecture

Chapter 4

Result

This section discusses the results of simulation and flight tests by using aforementioned algorithms. The following figure shows the three fixed-wing vehicles used in this work.

Figure 4.1: Platforms used for the flight tests by KU Flight System Research Lab. Left: Skyhunter, Middle: DG808, Right: Yak-54 40%.

Simulation results for phasic navigation are presented. Phasic navigation contains the curvature control, Hungarian algorithm, and the moving mesh methods. The comprehensive simulation result is shown to describe each stage of phasic navigation algorithm. In addition, Hardware-in-The-Loop (HiTL) test results are presented for various scenarios: swarming, swarming with external disturbances, communication loss, and collision avoidance. Initial conditions are selected from actual flight test data to enhance simulation reliability. Lastly, the flight test validation results are presented. Simulation results are compared with flight test results with identical initial conditions. All guidance parameters, control weighting matrices and gains are presented in ***Appendix E***. Figure 4.2 shows the roadmap for this chapter.

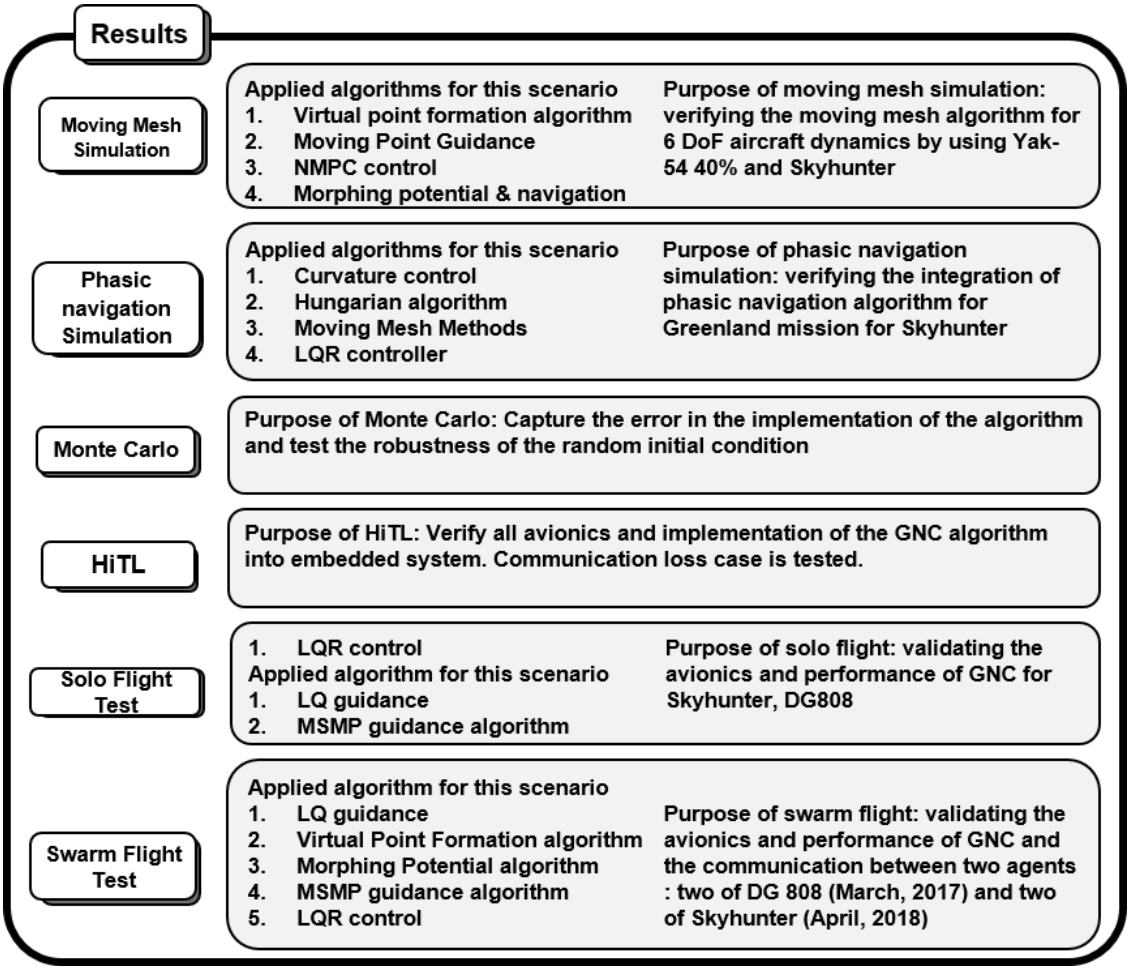


Figure 4.2: Roadmap of Chapter 4

4.1 Formation holding using Moving Mesh Methods

In this section, simulation results are presented for five Yak-54 40% (Fig. 4.1 (right)) in two different flight formation scenarios: diamond and triangle. Four trajectory scenarios have been simulated: a straight line with the diamond formation, a circular curve, an arbitrary curve and a sinusoidal curve with triangle formation. The diamond formation is made by four outer agents and one inner agent. In order to investigate the impact of the moving mesh methods, all agents are allocated for the outer and inner agent in advance. In the next section of this chapter, the formation assignment using Hungarian algorithm is shown. The path for the outer agents are generated by using the virtual point formation algorithm. The inner agent position is corrected adaptively by using the moving mesh methods. The

triangle formation consists of three outer agents and two inner agents. The simulation has been operated in three dimensional environment with 6 DoF nonlinear aircraft dynamic model.

4.1.1 Diamond Formation Shape Simulation

Fig. 4.3 shows the simulation results of holding a diamond shape formation. In Fig. 4.3(a), the dots represent the moving point position and the thin line represents the aircraft's trajectory. In order to shape the diamond formation, the horizontal and vertical relative distances are defined at 30.5 m. Therefore, one inner agent should have the distance to other agents as 15.2 m since it is located at the center of a diamond formation. Fig. 4.3 demonstrates 5 seconds of the flight simulation where agents were flying at 60 knots speed with 20 Hz update rate. The agents are commanded to the heading angle of +45 degree. The virtual leader are computed using Eq. 2.82, 2.83, and 2.84. Fig. 4.3(b) provides a close view of the formation to show the effect of the moving mesh methods clearly. The triangle markers shows the moving point position and they are rotated by the heading angle. Fig. 4.3(c) represents the wingtip-to-wingtip distance between the inner and outer agents. The tracking of formation is satisfactory since the distance from the inner agent to another agents converges to 15.2 m which is the desired distance from the center of formation. In addition, the collision is avoided since the distance between agents are above 3 m which is the length of a wingspan. Fig. 4.3 (d) shows that the heading angle of all agents are converged at 45 degrees. Fig. 4.4(a) shows the commanded values for the control surfaces deflections and Fig. 4.4(b) shows the angle of attack, sideslip angle, bank angle and pitch angle. Hereafter, the aircraft performance assessed based on the criteria, see Table 4.1. Since UASs do not have the handling qualities, the criteria acquired from numerous flight tests. Regarding the controller performance, the threshold of the overshoot is set to 5%. In the general control theory, the damping ratio is desired to be above 0.7 which is corresponding to 5% overshoot.

Table 4.1: Performance assessment criteria

Items	Acceptable Range	Items	Acceptable range
P	$\pm$25 deg/sec	ϕ_{cmd}	5% Overshoot or $\pm$3 degree
Q	$\pm$15 deg/sec	θ_{cmd}	5% Overshoot or $\pm$3 degree
R	$\pm$15 deg/sec	$V_{T_{cmd}}$	5% Overshoot
ϕ	$\pm$60 deg	Lateral TE	5% Overshoot
θ	$\pm$10 deg	Longitudinal TE	5% Overshoot
$\dot{\delta}_T$	$\pm$8 %/sec	$\dot{\delta}_A$	$\pm$20 deg/sec
$\dot{\delta}_E$	$\pm$10 deg/sec	$\dot{\delta}_R$	$\pm$10 deg/sec
Collision avoidance	$\sigma \leq distance \leq 2\sigma$		

where TE is the tracking error. In addition, the sideslip angle is desired to be bounded between $\pm$5 degrees. Based on Table 4.1, the pitch angle tracking has 33% overshoot and it damped out after 10 seconds. This is caused due to the turning maneuver at the beginning of this simulation. When the turning maneuver is over, the pitch angle tracking is acceptable since the overshoot is 3.3%. The pitch angle performance is also acceptable since its magnitude is stayed within 10 degrees. The roll angle tracking is also similar to the pitch angle tracking. At the beginning of the simulation, the roll angle error is 40 degrees than the controller could follow the command after 12 seconds. The overshoot of the roll angle tracking is around 20% which is larger than the acceptable overshoot. The sideslip angle is acceptable since the magnitude of the sideslip angle is within 5 degrees. In addition, NMPC tries to correct error very fast since the speed of elevator at the beginning it changes between -5 and 5 degrees within 0.1 second (100 deg/sec). Overall, the aircraft heading angle has 45 degree error to follow the moving point and converged to the desired states: the sideslip angle converges to zero, the roll angle went to zero since all agents heading angle reaches 45 degrees, and the pitch angle converges to 3.2 degrees. The initial position of each agent is presented in Table 4.2.

Table 4.2: The initial position for diamond shape formation

	Outer				Inner
	Agent 1	Agent 2	Agent 3	Agent 4	Agent 5
North (m)	30.5	15	30.5	46	23
East(m)	30.5	15	0	13	13
Height(m)	122	122	122	122	122

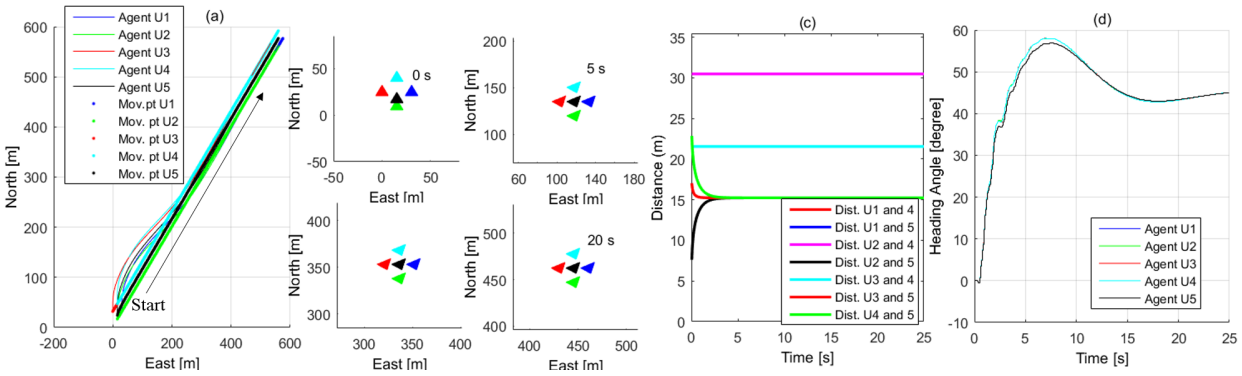


Figure 4.3: Simulation result of Diamond formation. (a) Moving points and actual agents' position. (b) Formation shape at distinct sample times to show effect of the moving mesh method. (c) Distance between outer and inner agents. (c.g.-to-c.g.) (d) Heading angle of actual agents

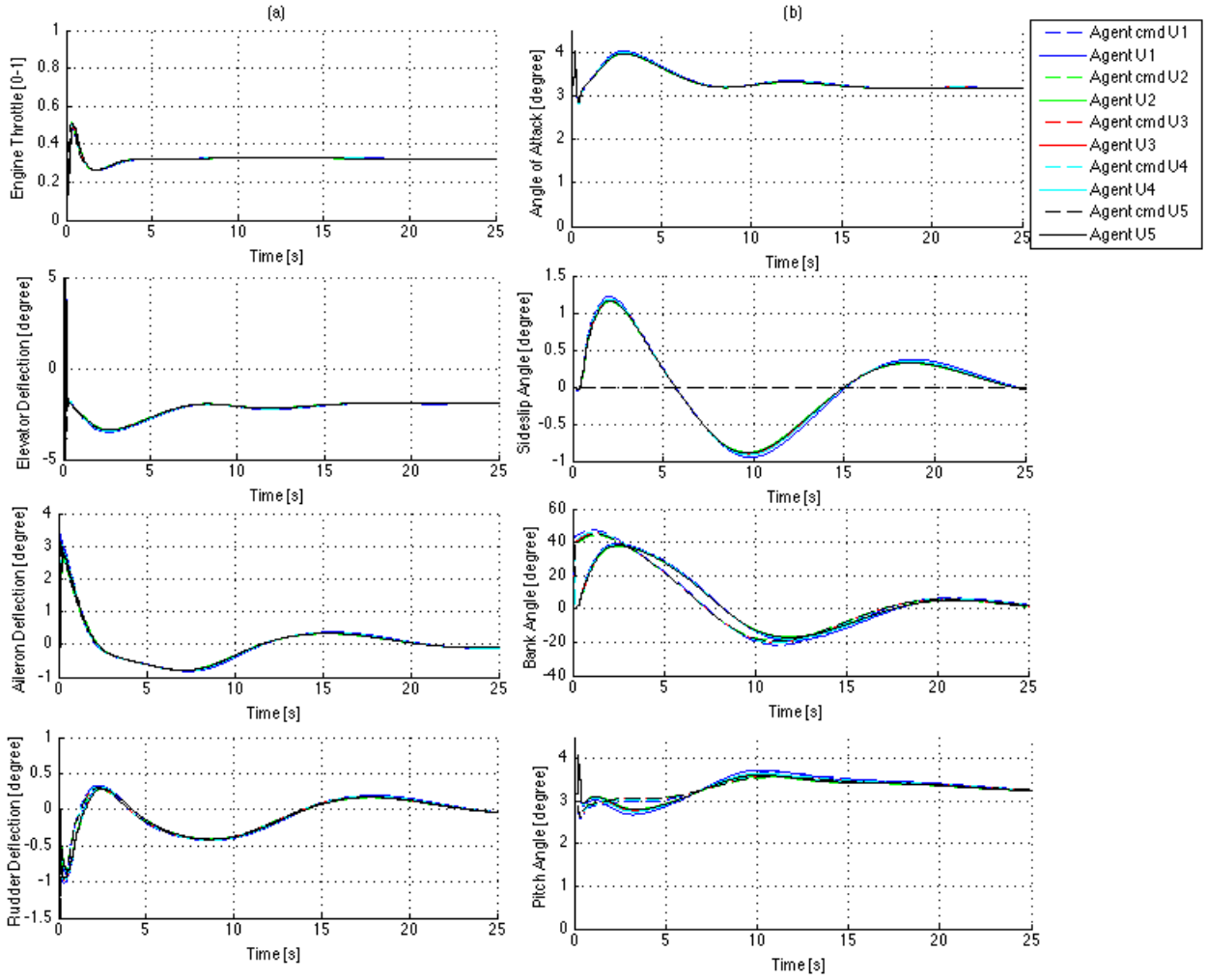


Figure 4.4: Commanded values and states in 3D simulation environment (a) Commanded control values for the control surfaces and throttle. (b) States (Angle of attack, Sideslip angle, Bank angle, and Pitch angle)

4.1.2 Triangle Formation Shape Simulation

In this simulation, the circular trajectory is defined by Eq. 2.82, 2.83, 2.84 and the following equation:

$$\begin{cases} p_{RP_N}[1,\cdots,N+2] = p_{RP_N}[0] + t_{fs}\cdot\Delta t\cdot\|V_d\| + r\cdot\sin\left(\frac{\|V_d\|}{r}\cdot\Delta t\cdot\left(t_{fs}-t_{fs_d}+(0:N+1)\right)\right), \\ p_{RP_E}[1,\cdots,N+2] = p_{RP_E}[0] + r\left(\cos\left(\frac{\|V_d\|}{r}\Delta t\cdot((t_{fs}-t_{fs_d})+(0:N+1))\right)-1\right), \\ p_{RP_H}[1,\cdots,N+2] = p_{RP_H}[0], \end{cases} \tag{4.1}$$

where t_{fs} is the flight sample time, t_{fs_d} is the desired flight sample time to change to the circular trajectory, and r is the radius of the circular trajectory at 305 m. This trajectory has been intentionally designed to be challenging (e.g., 180 degree heading angle changes) for high speeds and high inertia aircraft in order to assess the performance of the moving mesh methods in the desired triangle formations defined in Table 4.3. The triangle for the formation is defined as the height and base at 18 m. Three outer agents generate the triangle's circumference and two inner agents are placed inside the triangle. The simulation has been operated by 1200 iterations at 20 Hz, which corresponds to 1 minute of flight time. The trim speed of the aircraft is 60 knots. Table 4.3 presents the initial position and the desired relative distance from the virtual leader in this scenario.

Fig. 4.6 (a) shows the trajectory of the moving points and the actual agents. All agents start at [0,0] m in north and east coordinate and follows the demanded circular trajectory. As Fig. 4.6(b) shows, the inner agents (light blue and black color markers) stay inside of the formation. At the flight time 0 seconds, the inner agents are located closer to the outer agents that are at the base of the triangle comparing to the first outer agent. The distances between inner agents to the first outer agents are 13 and 19 meters while the distances between inner agents to the second and third agent are 7.2 and 5.7 meters. The moving

Table 4.3: The initial position for triangle shape formation (left) and relative distance for virtual point formation (right)

(a) Initial Position Coordinates in Inertial Frame

	Outer			Inner	
Agent #	1	2	3	4	5
North (m)	18	0	0	6	4
East (m)	0	-9	9	-5	13
Height (m)	122	122	122	122	122

(b) Desired relative distance from the virtual leader

	Outer		
Agent #	1	2	3
North (m)	13	-5	-5
East (m)	0	-9	9

mesh methods corrects the inner agent location towards the center to make the mesh as uniform as possible. In order to quantify this feature, the distances from each inner agents to outer agents are averaged and compared with the mean of distances from the outer agents to the centroid of the triangle formation(11.2 m). At the initial condition, the averaged distances from each inner agents to outer agents are 11.8 and 15.56 m for inner agent 4 and 5, respectively. After the moving mesh methods are applied, they became 11.65 and 11.53 m for inner agent 4 and 5, respectively. We can observe these values are more closer to the averaged distances from outer agents to the triangle centroid (11.2 m). Fig. 4.6(c) presents the distance between the inner and outer agents. All agents are able to keep a specified distance from one another (see Figure 4.5) so that they do not collide each other (Figure 4.6 shows the distances are always kept above zero.). In addition, the inner distance between agents is very close (their distance is 2.5 m which is shorter than the length of its wingspan as 3 meter.) without collision considering the size and speed of Yak-54 40% platforms. Fig. 4.6(d) shows that the heading angles of the agents are coherent during the simulation. All agents changes their heading angle from zero to -180 degrees which matches with the desired trajectory. Fig. 4.7(a) shows all control surfaces deflections and Fig. 4.7(b) presents the following states: the angle of attack, sideslip angle, bank angle, and pitch angle. The sideslip angle is met the criteria since the magnitude is kept within 5 degrees defined in Table 4.1.

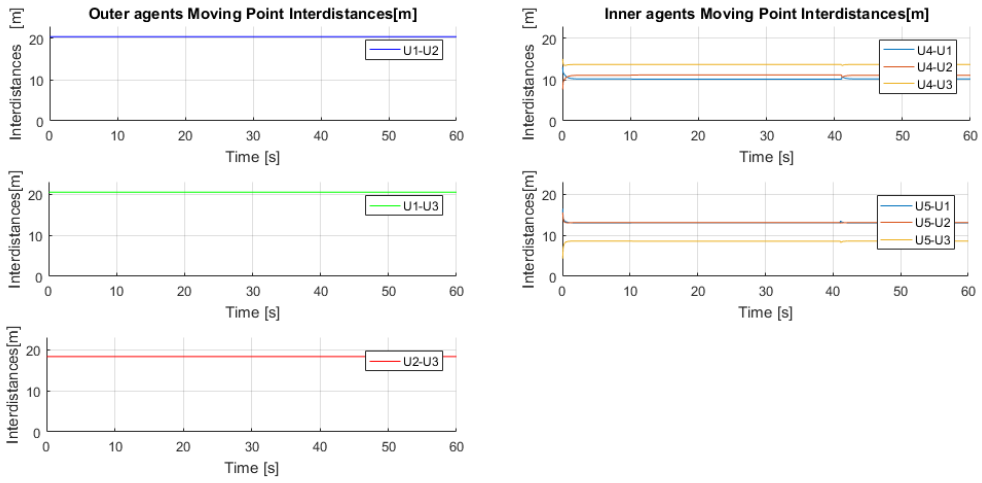


Figure 4.5: The distances between the moving points position for outer and inner agents

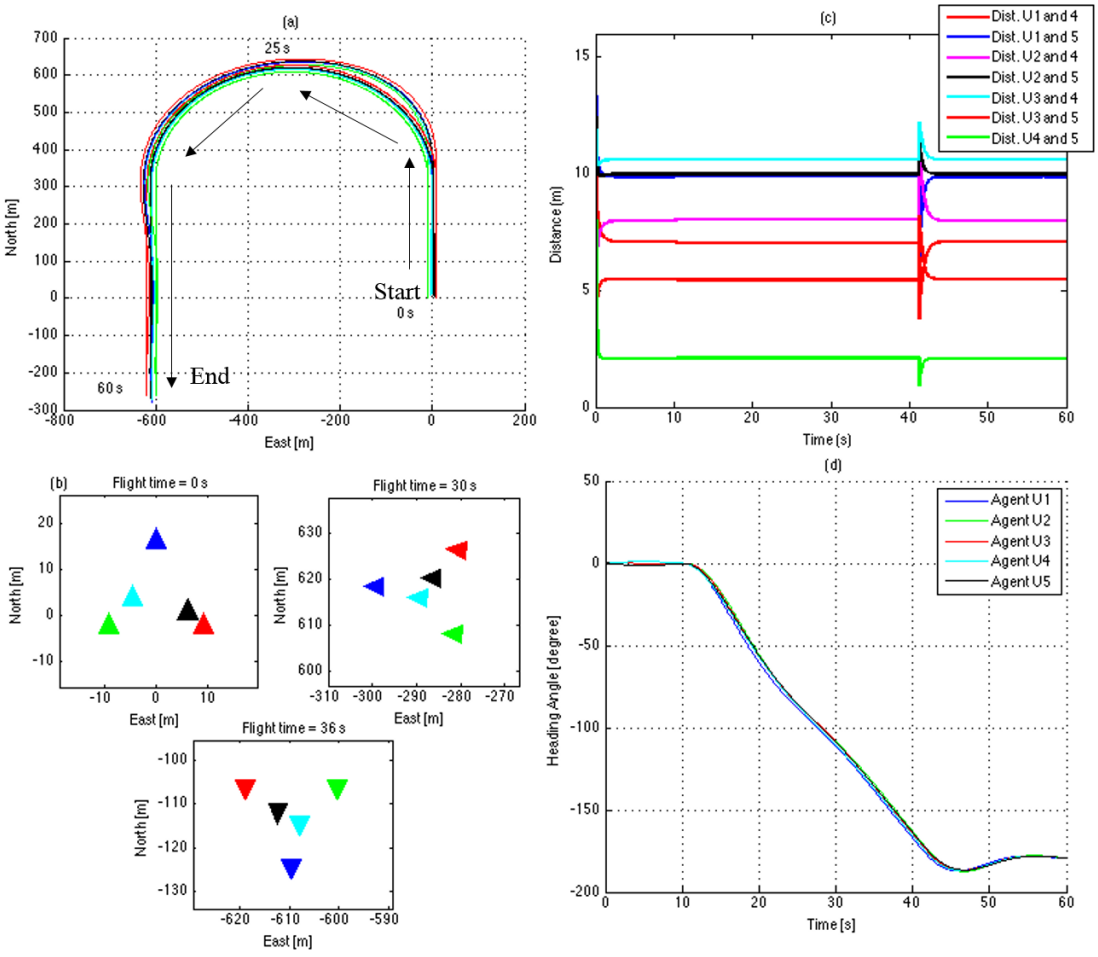


Figure 4.6: Simulation result of Triangle formation. (a) Moving points and actual agents' position. (b) Formation shape at distinct sample times to show the effect of the moving mesh method. (c) Distance between outer and inner agents. (c.g.-to-c.g.) (d) Heading angle of actual agents. All figure legends are identical to Fig. 4.3 and 4.4.

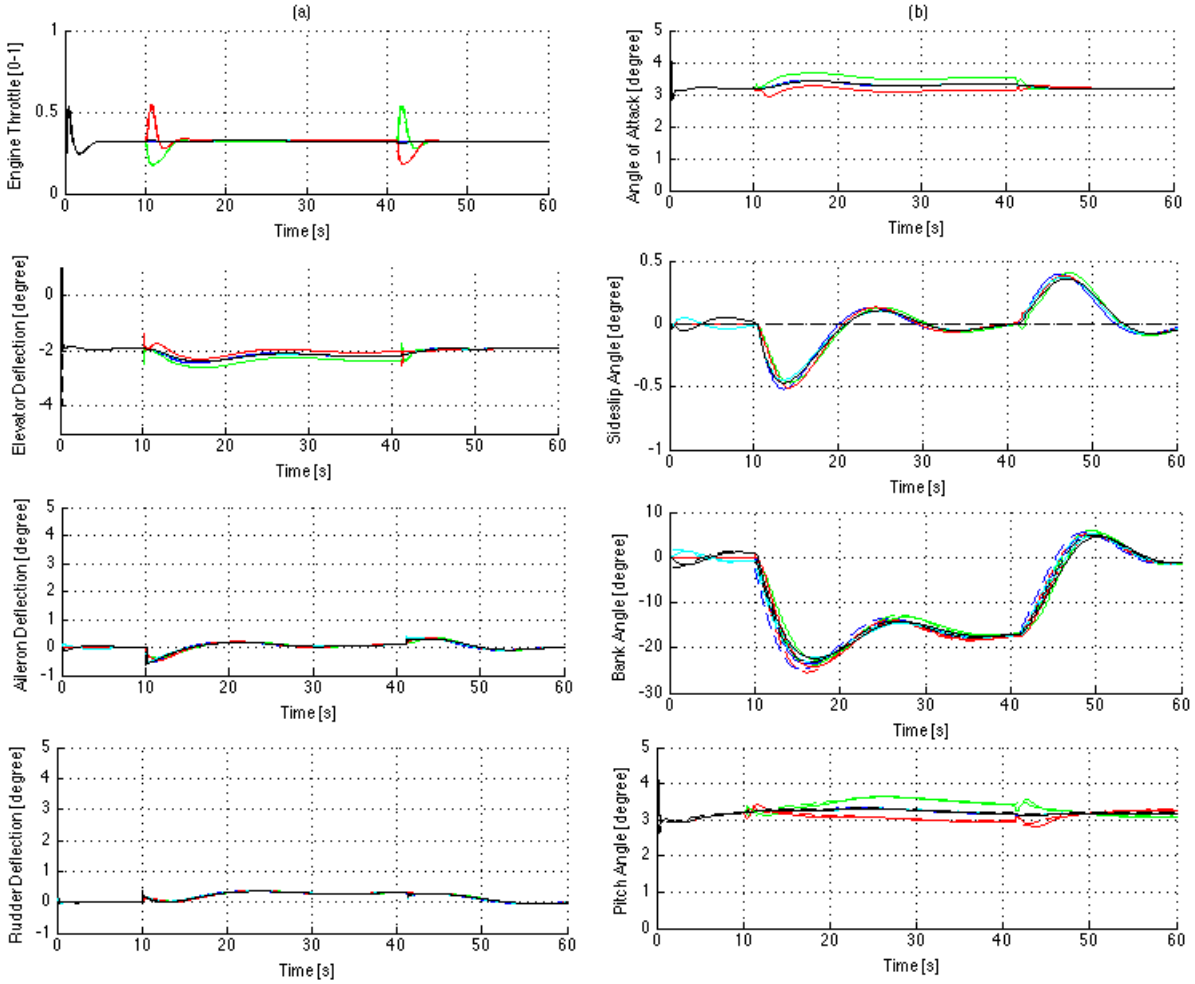


Figure 4.7: Commanded values and states in 3D simulation environment (a) Commanded control values for control surfaces and throttle. (b) States (Angle of attack, Sideslip angle, Bank angle, and Pitch angle). All figure legends are identical to Fig. 4.3 and 4.4.

4.1.3 Arbitrary Curve Simulation

In this scenario, the arbitrary curve trajectory is chosen to assess the ability of holding the formation by using the moving mesh methods. The virtual leader was commanded to change the heading angle, shown in Fig. 4.9(d). Table 4.4 shows the initial agents' position and the desired relative distance which makes the triangle formation defined by the height and base length at 19.5 m and 24 m, respectively.

Table 4.4: The initial position for a triangle formation (left) and relative distance for virtual point formation (right)

(a) Initial Position Coordinates in Inertial Frame

	Outer			Inner	
Agent #	1	2	3	4	5
North (m)	15	-3	-3	6	4.5
East (m)	0	-12	12	-4	5
Height (m)	122	122	122	122	122

(b) Desired relative distance from the virtual leader

	Outer		
Agent #	1	2	3
North (m)	14	-4.5	-4.5
East (m)	0	-12	12

The simulation is conducted for 190 seconds at 20 Hz with the trim speed of 60 knots. Fig. 4.9(a) displays the trajectory of the moving points and the agents.

All agents followed the virtual leader position with high accuracy and smoothness.

Fig. 4.9(b) presents the closer look for the moving points at the certain flight time. The triangle shape was rotated as the virtual leader heading is changed by the virtual point formation logic. Then, the moving mesh methods is applied and the inner agents are located at the balanced positions in respect to the free energy (Eq. 2.97).

Fig. 4.9(c) and 4.8 shows the distance between the outer and inner agents' moving points. As the previous results showed, all agents avoid the collision since the distances are always larger than zero between agents. In addition, the outer formations is kept by maintaining the desired distances as 22 (between U1&U2 and U1&U3) and 24 (between U2&U3) meters. In Fig. 4.10(a) and (b), control surface deflections and states are presented. Within 5 seconds of the simulation, the abrupt changes are observed in the control deflections and the states also fluctuate consequently. For example, the throttle of all agents are changed from 31.5%

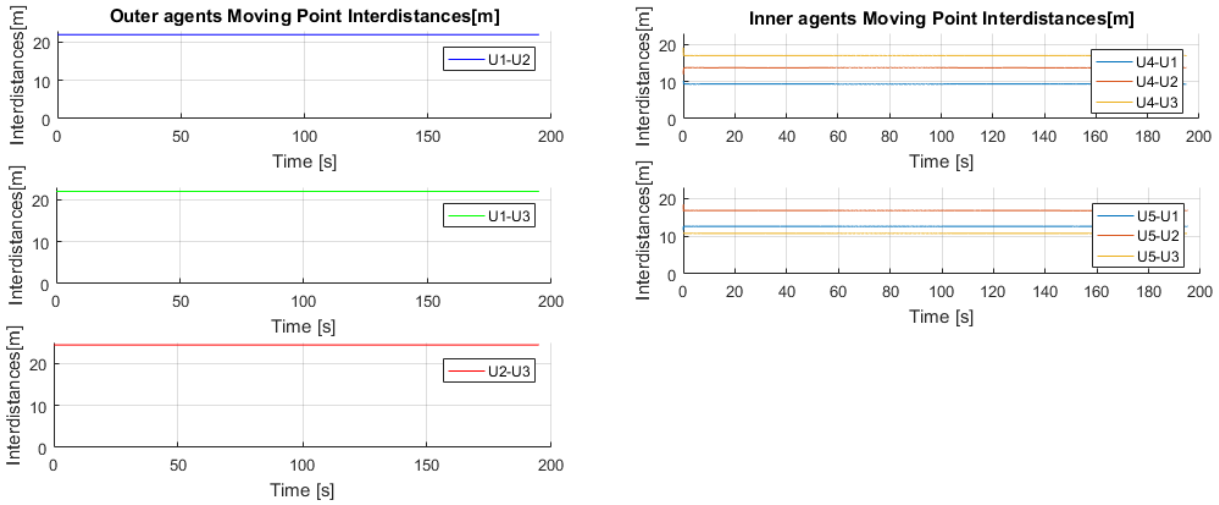


Figure 4.8: The distance between outer and inner agents' moving point

to 91.44% within 0.15 seconds which is corresponding to 399.6%/sec exceeding the defined criteria in Table 4.1. The sideslip angle is acceptable result but it has oscillation between -1 and 1 degree by 1 Hz frequency whichi is not desirable. The pitch angle tracking is acceptable since the maximum error is 1.7 degree ($\leq$ 3 degree). The roll angle tracking is not satisfied since the maximum error is 1 degree larger than 3 degrees criteria. Since NMPC has aggressive reactions (states are not met the defined criteria in Table 4.1) to very small errors, an LQR controller is used for the flight tests.

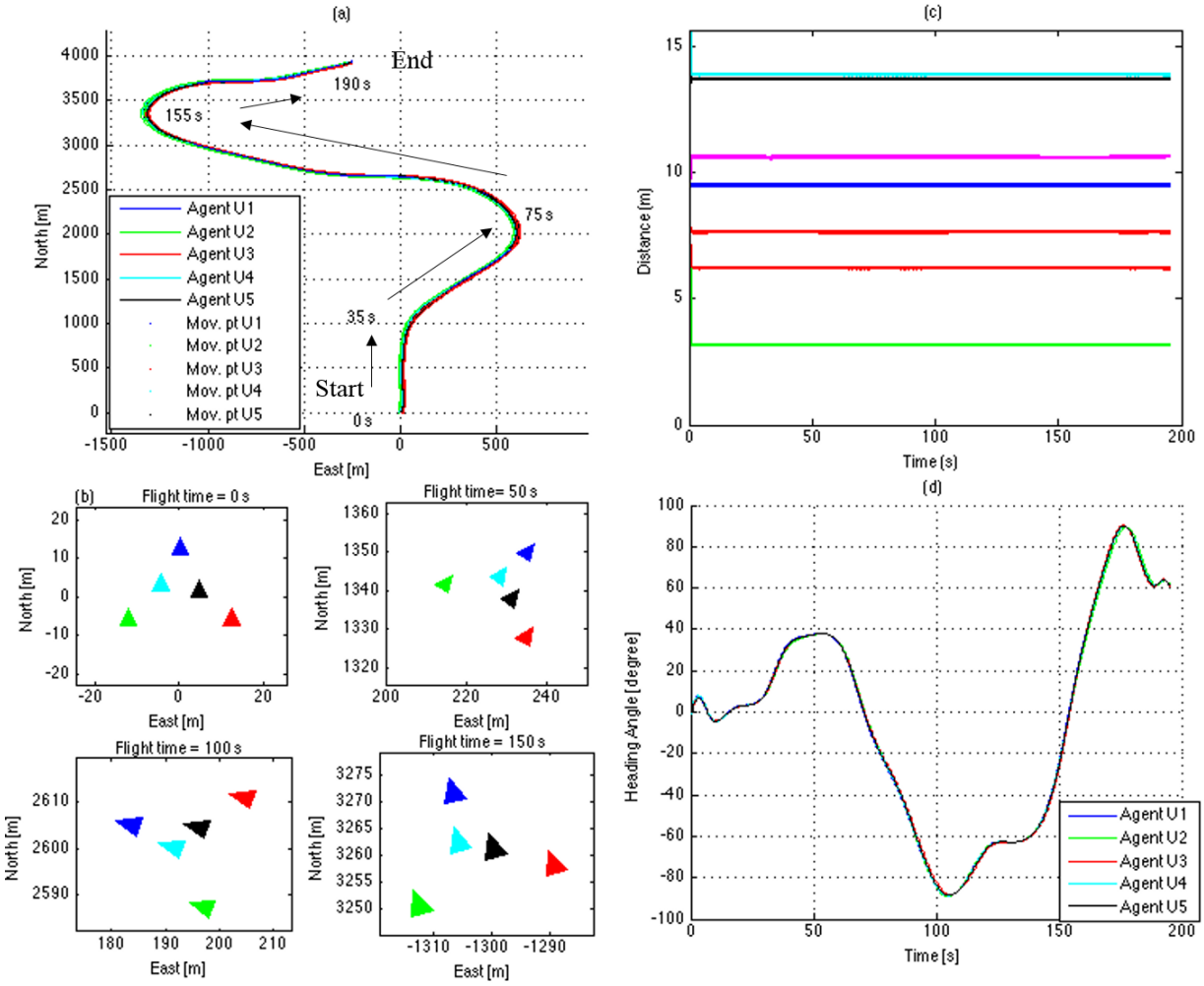


Figure 4.9: Simulation results for the arbitrary trajectory: (a) Moving points and actual agents' position. (b) Moving point position at certain flight time to see the effect of the moving mesh method. (c) Distance between outer and inner agents. (c.g.-to-c.g) (d) Heading angle of actual agents. The legends are equivalent to Fig. 4.3 and 4.4.

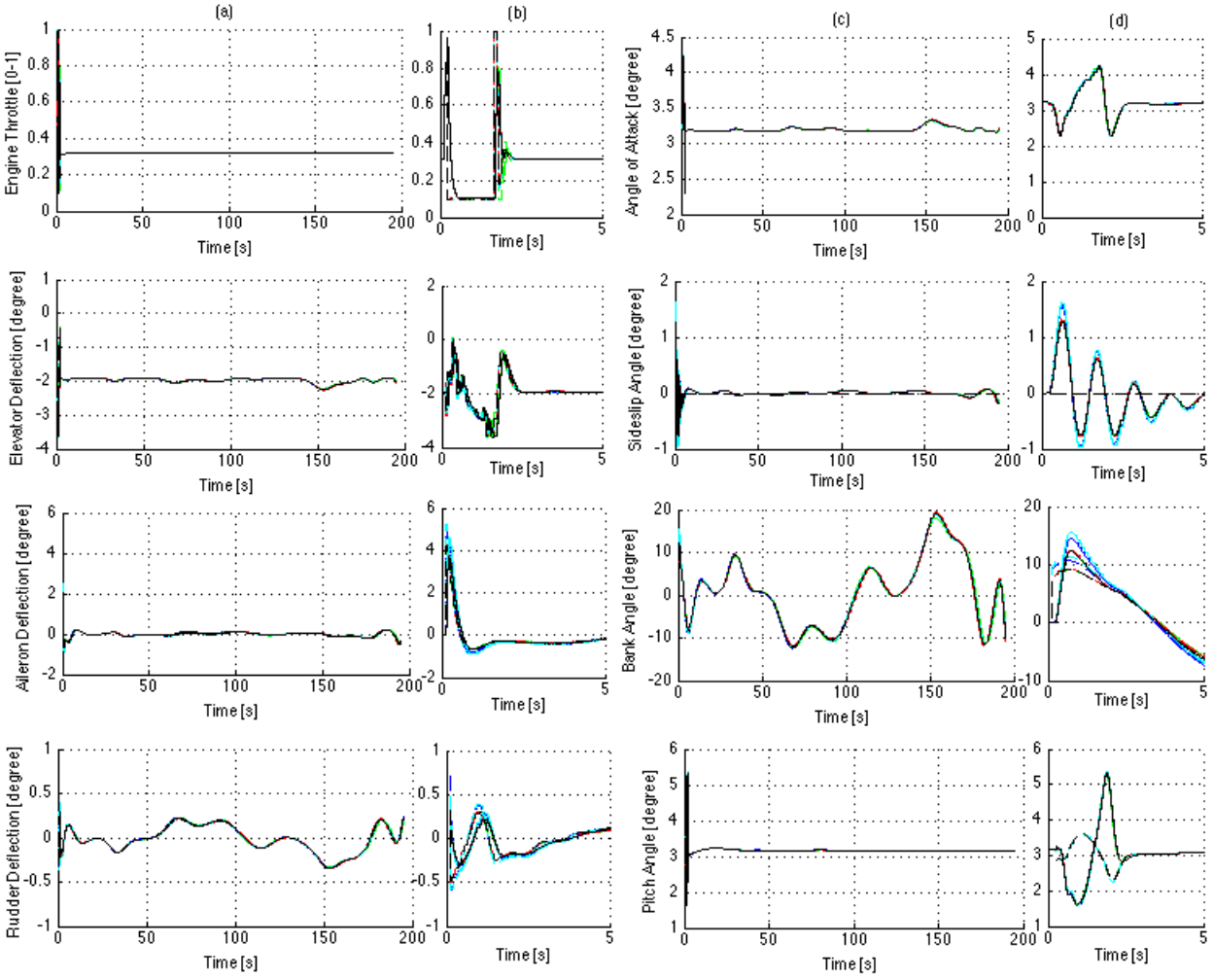


Figure 4.10: Commanded values and states in 3D simulation environment (a) The control deflections. (b) Zoomed-in view for (a) plot for the first 5 seconds. (c) States (Angle of attack, Sideslip angle, Bank angle, and Pitch angle). (d) Zoomed-in view for (c) plot for the first 5 seconds. The legends are equivalent to Fig. 4.3 and 4.4.

4.1.4 Sinusoidal Curve Simulation

The sinusoidal trajectory is simulated to evaluate the comprehensive performance of the moving mesh methods and virtual point formation algorithm. The triangle formation is used and defined by the height and base length of 18 m. Three agents compose the boundary of the triangle formation and the remaining two agents are placed inside of the triangle. This simulation is performed for 235 seconds at 60 knots speed. The initial agents' position and the desired relative distances from the virtual leader are presented in Table. 4.5. Fig. 4.11

Table 4.5: The initial position for triangle shape formation (left) and relative distance for virtual point formation (right)

(a) Initial Position Coordinates in Inertial Frame

	Outer			Inner	
Agent #	1	2	3	4	5
North (m)	18	0	0	6	4
East (m)	0	-9	9	-5	13
Height (m)	122	122	122	122	122

(b) Desired relative distance from the virtual leader

	Outer		
Agent #	1	2	3
North (m)	13	-5	-5
East (m)	0	-9	9

(a) displays the moving points and the agents position. Agents make 180 degree turns four times which is a demanding maneuver for such a tight formation. Fig. 4.11 (b) shows a closer look of the moving point position at the distinct flight times. The virtual point formation algorithm provides the outer agents position based on the desired relative distance in given Table. 4.5. Then, the moving mesh methods generate the inner agents' position based on the outer agents' position. Fig. 4.11 (d) presents the heading angle of the actual agents. Based on these figures, it is observed that the triangle formation is rotated as the virtual leader makes the turn to keep the coherency of the formation flight as the heading angle is changed (the overshoot from the desired heading angle is 3.6% ($\leq$ 10% defined in Table 4.1)). Fig. 4.11 (c) and Fig. 4.12 shows the distance between all agents' moving points. The outer agents kept the desired formation distances that are 20.1 (between agent 1&2 and 1&3) and 18 m (between agent 2&3). Fig. 4.13 shows the control surface deflections and the states. The pitch angle tracking is acceptable since the maximum error is 1.2 degree ($\leq$ 3 degree from the criteria) at the beginning of the simulation (0.2 seconds). The roll angle tracking

is also acceptable since the maximum error is 0.6 degree which is less than 3 degree criteria.

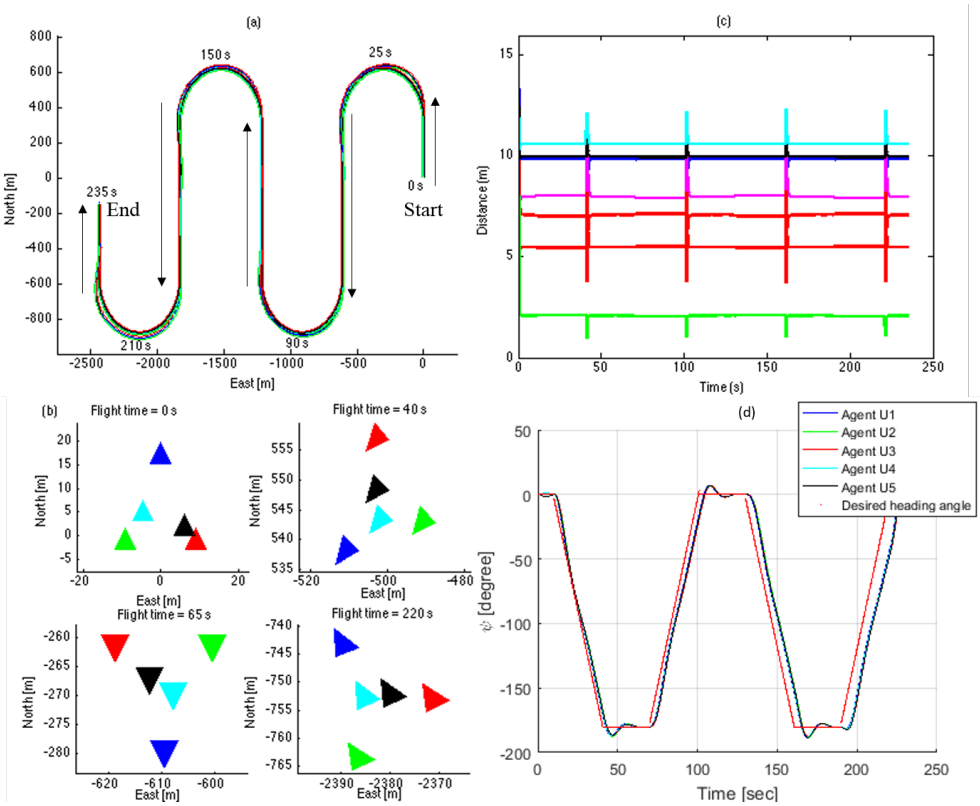


Figure 4.11: Simulation result of Triangle formation. (a) Moving points and actual agents' position. (b) Formation shape at distinct times to see effect of the moving mesh method. (c) Distance between outer and inner agents. (c.g.-to-c.g.) (d) Heading angle of actual agents Legends for all figures were referred from Fig. 4.3 and 4.4.

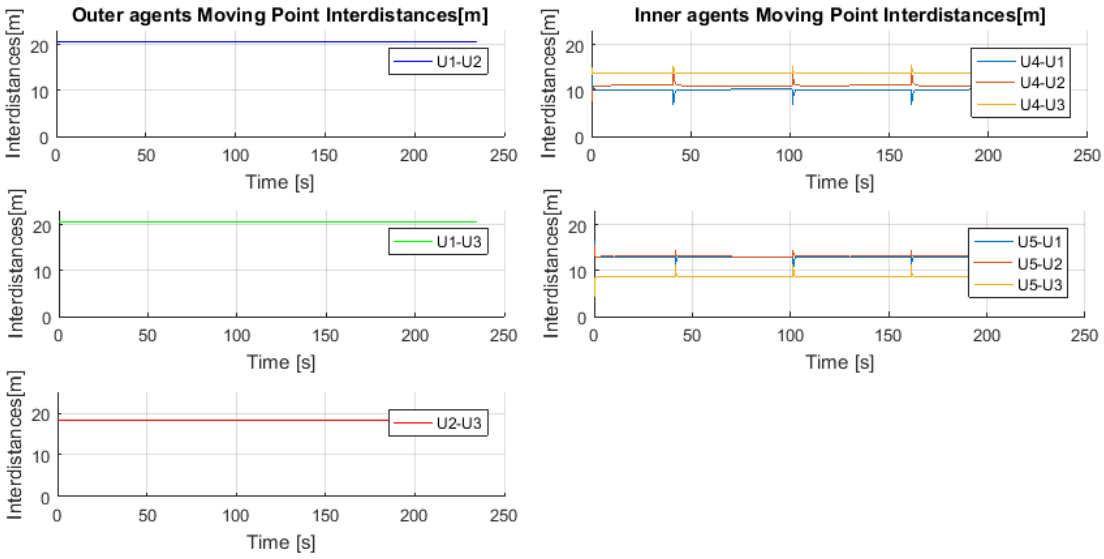


Figure 4.12: The distance between outer and inner agents' moving points

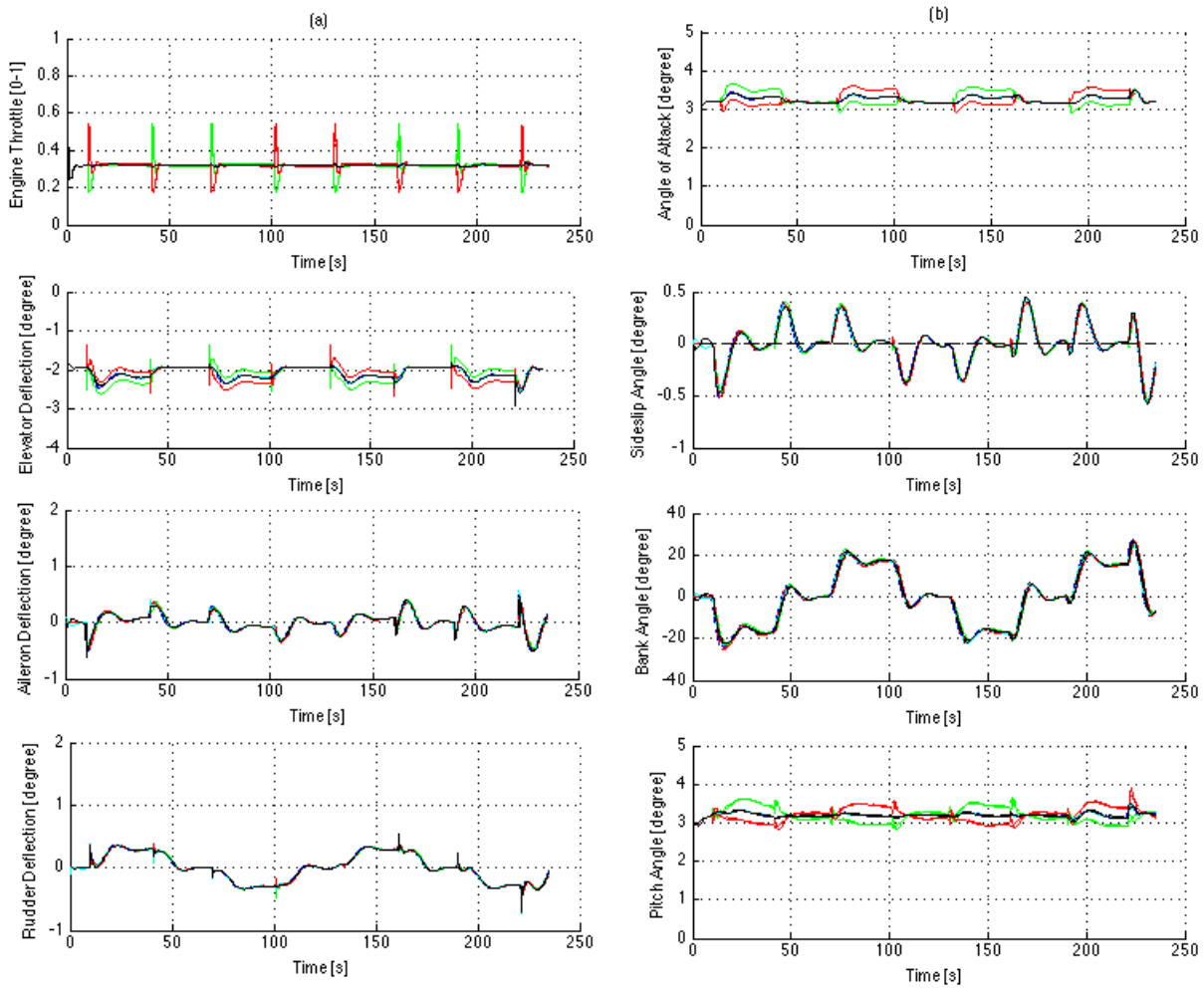


Figure 4.13: Commanded values and states in 3D simulation environment (a) Commanded control values for the control surfaces and throttle. (b) States (Angle of attack, Sideslip angle, Bank angle, and Pitch angle) Legends for all figures were referred from Fig. 4.3 and 4.4.

4.1.5 Wind Gust Evaluation

To investigate the robustness of the moving mesh methods algorithms, the simulation is performed with external disturbances (e.g., a wind field). The agents are exposed to a cross wind which is a 5 knots (or 2.6 m/s) East wind field shown in Fig. 4.15. To simulate the unsteadiness of the wind field, the magnitude of the wind is increased after 15 seconds by 20% (6 knots or 3.1 m/s). As the wind is applied to agents individually, the actual aircraft position is shifted to the west direction due to the wind. Depending on the heading angle of agents, they face the different direction wind (e.g., head wind, tail wind, and cross wind). For example, a tail wind is applied when agents follow a circular portion of the trajectory. While agents are exposed to the external disturbances, they hold the desired formation (the outer agents are kept at 20.1 (between agent 1&2 and 1&3) and 18 m (between agent 2&3)) and do not collide with each other (distance between agents is kept above zero), see Figure 4.14. Although they are disturbed and drifted 8 m (at the beginning of the simulation) and 22 m (after the virtual leader finished turning) by the cross wind, they come back to follow the virtual leader.

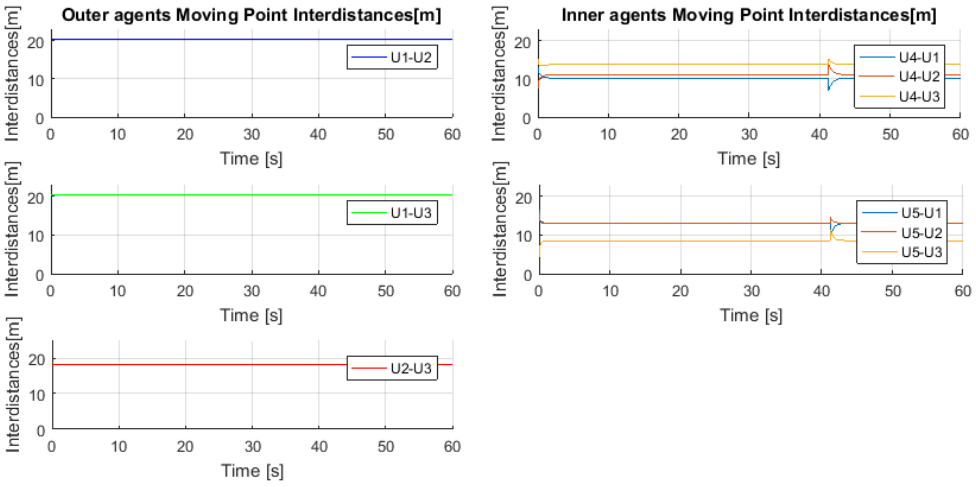


Figure 4.14: Distance between outer and inner agents' moving point

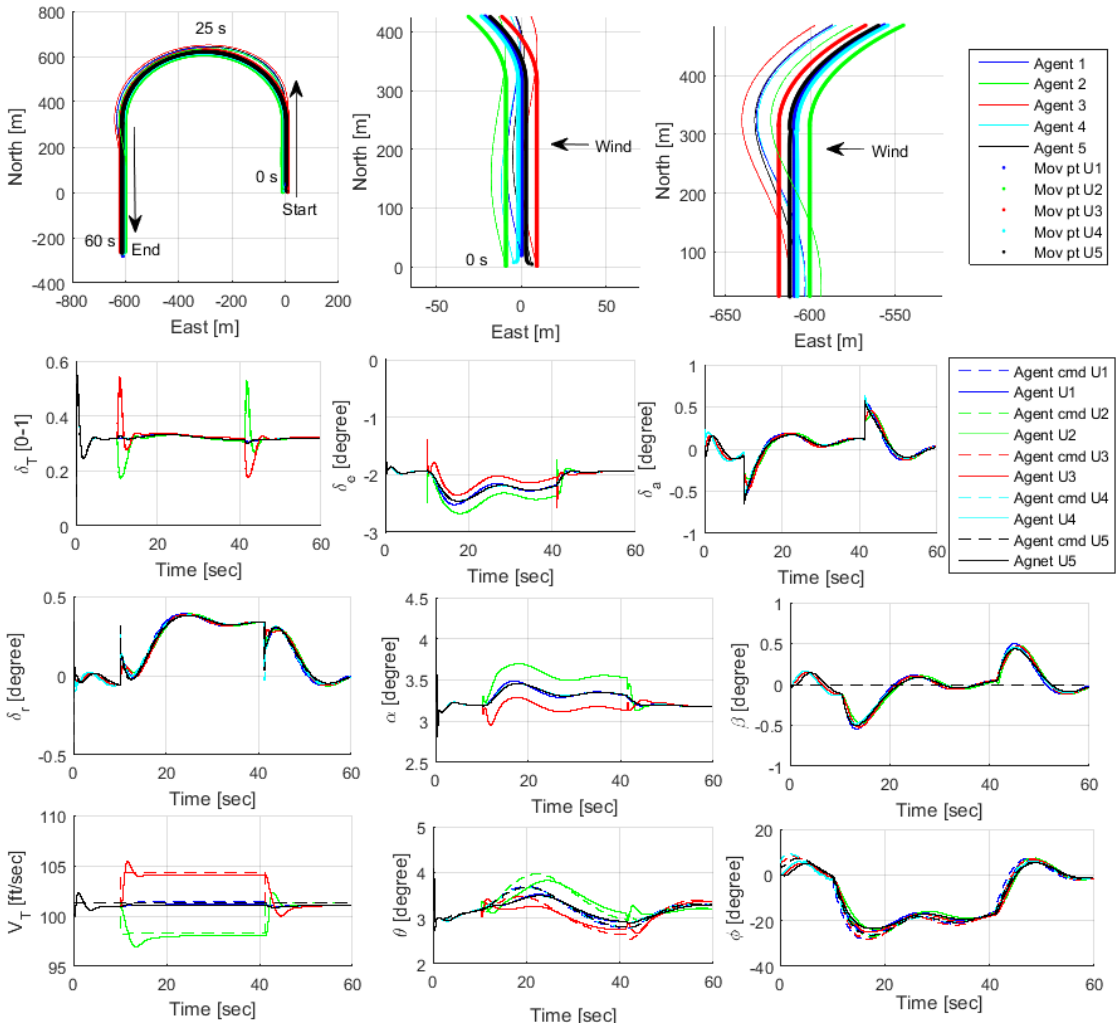


Figure 4.15: Wind gust simulation for the circular trajectory. Wind blows from east (until 15 seconds flight, 5 knots east wind (2.6 m/s) is applied. After 15 seconds of flight, 6 knots (3.1 m/s) east wind was applied)

4.1.6 Tracking Error Anaylsis

The initial condition and flight conditions are identical with Sec. 4.1.1 and 4.1.2. To observe how the tracking error settles, the simulation time is extended for each scenario. The diamond shape formation with a linear trajectory is simulated for 75 seconds. The triangle shape formation with the circular trajectory is simulated for 100 seconds. The tracking error is defined as the lateral distance from the line made by two consecutive moving points. As shown in Fig. 4.16, the tracking error is reduced to zero after approximately 40 seconds for the linear trajectory scenario. For the circular trajectory, the error settles to zero after ap-

proximately 70 seconds. The circular trajectory takes longer time because the moving point heading is changed continuously comparing to the linear trajectory (The linear trajectory heading is not changed at all). The linear trajectory scenario has higher tracking error since the required heading angle change is 45 degrees while the circular trajectory requires zero heading angle change. If the initial heading angles of all agents are zero for both scenarios then moving points are planned as the virtual point proceeds.

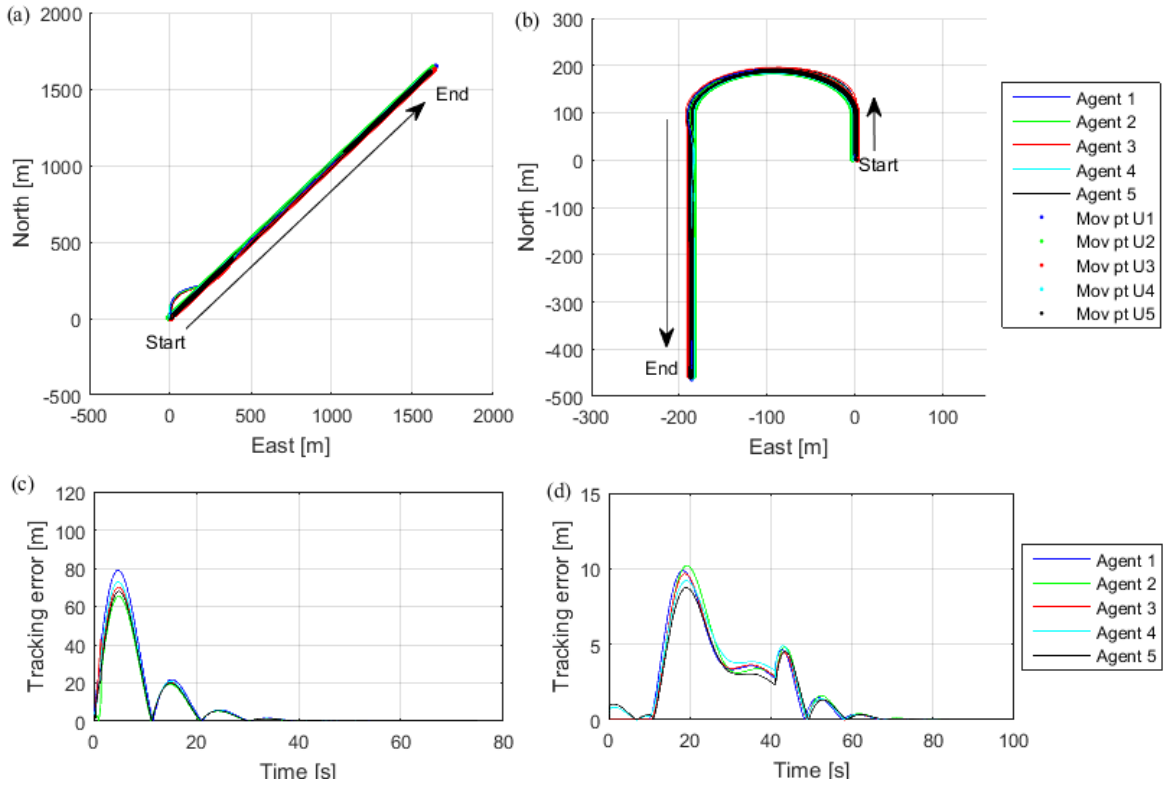


Figure 4.16: Tracking error analysis. (a) Linear trajectory with the diamond shape formation, (b) Circular trajectory with the triangular shape formation, (c) Tracking error for the linear trajectory simulation (d) Tracking error for the circular trajectory simulation

4.1.7 Moving mesh methods with the navigation algorithms

This section addressed the result of implementation with the moving mesh method and the navigation algorithms: the intelligent waypoint modification and index decision algorithm, LQ guidance path planning, and the virtual point formation algorithm. The waypoint is

given an example of the remote sensing mission in Greenland. Six Skyhunters are used with the desired formation shape in this simulation as shown in the following table.

Table 4.6: The desired relative distance from the virtual leader

Agent	North [ft]	East [ft]
Outer 1	50	0
Outer 2	0	-50
Outer 3	0	50
Outer 4	-50	0

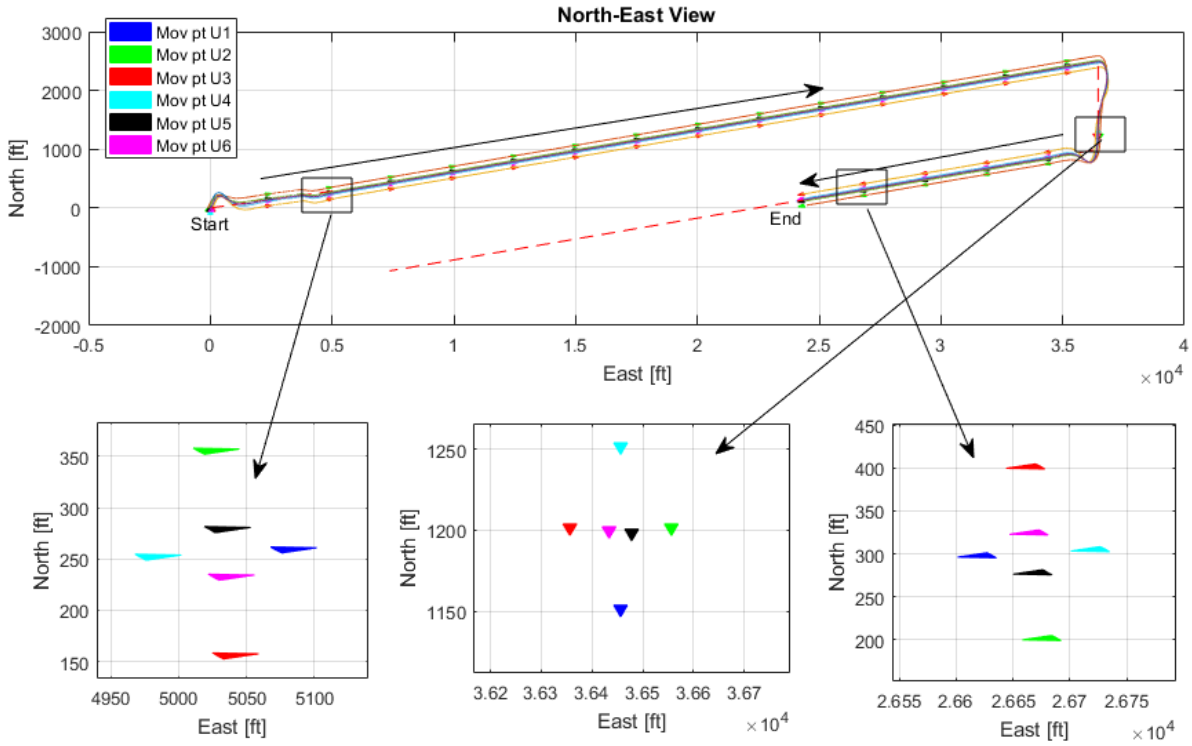


Figure 4.17: Moving mesh methods with the intelligent navigation algorithm

As Figure 4.17 shows, the moving points can be planned correctly. Formation is kept throughout the flight while the virtual leader follows the given waypoints. The intelligent waypoint modification and index decision algorithm connects the initial virtual leader position to the waypoint box.

4.1.8 Moving mesh methods with the morphing potential field

In this simulation, the morphing potential algorithm is implemented to the moving mesh methods. The goal of this simulation is that the virtual leader avoids the two obstacles and all agents keep their formation shape. The assessment criteria is specified in Table 4.1 that the farthest distance from the obstacle should be within the range of $[\sigma, 2\sigma]$ where σ is the radius of the obstacle size. Five of Yak-54 40% are used with the desired relative distance shown in Fig. 4.18. The two obstacles are located at $[213.4, -15.2]$ and $[335.3, 15.2]$ m with

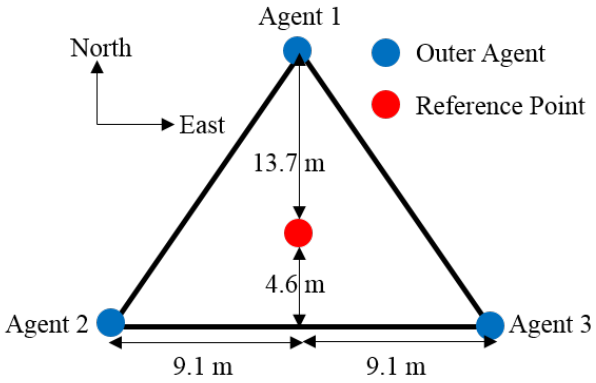


Figure 4.18: The desired relative distance from the virtual leader

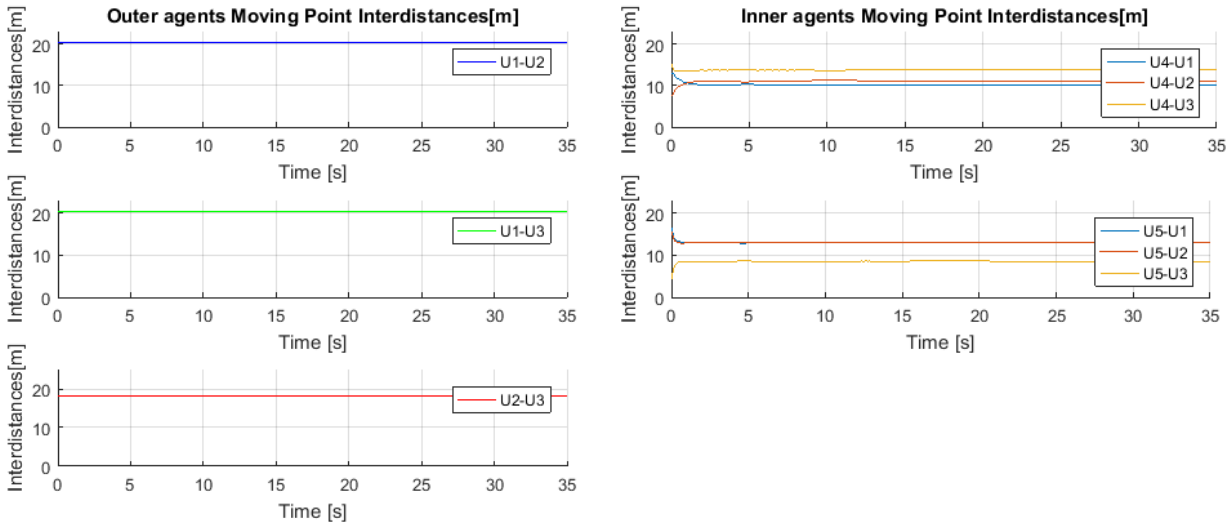


Figure 4.19: Distance between the outer and inner agents' moving points

the radius of 15.2 and 24.4 m, respectively. Fig. 4.20 presents the positions of the moving points and all agents. In order to show this accurately, Figure 4.21 shows a closer view of

the moving point position. All moving points kept the formation shape by the virtual point formation algorithm while the virtual leader avoided two obstacles. Figure 4.19 shows that the outer agents kept the desired formation distance as 20.45 (between agent 1&2 and 1&3) and 18.2 m (between agent 2&3). After the obstacle is cleared, all moving points go back to the desired trajectory of a zero heading angle in this simulation. The coherency of the formation shape and the obstacle avoidance are achieved successfully. In order to assess the avoidance assessment, only agent 2 is considered since it is closest toward the obstacles. The avoidance assessment is not met the criteria and the result is conservative comparing to the criteria. The farthest distance for smaller obstacle is 65.5 and 52.54 for the moving point and the agent 2 position, respectively (this is larger than $2\sigma = 30.4$ m). The farthest distance for larger obstacle is 44.54 and 69.34 m for the moving point and the agent 2, respectively (this is larger than $2\sigma = 48.8$ m). In order to meet the avoidance criteria, more tight tracking is required. Fig. 4.22 shows various states of the agents throughout the simulation.

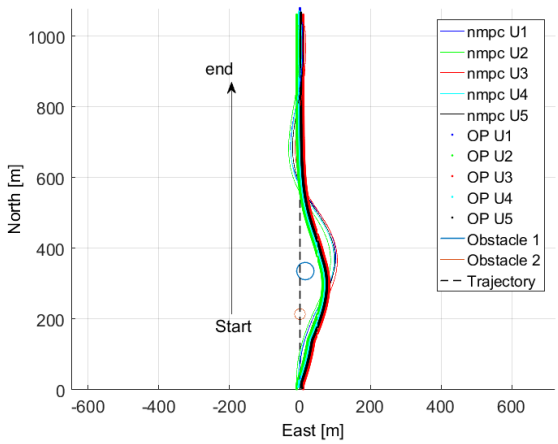


Figure 4.20: Morphing potential implementation with the moving mesh methods

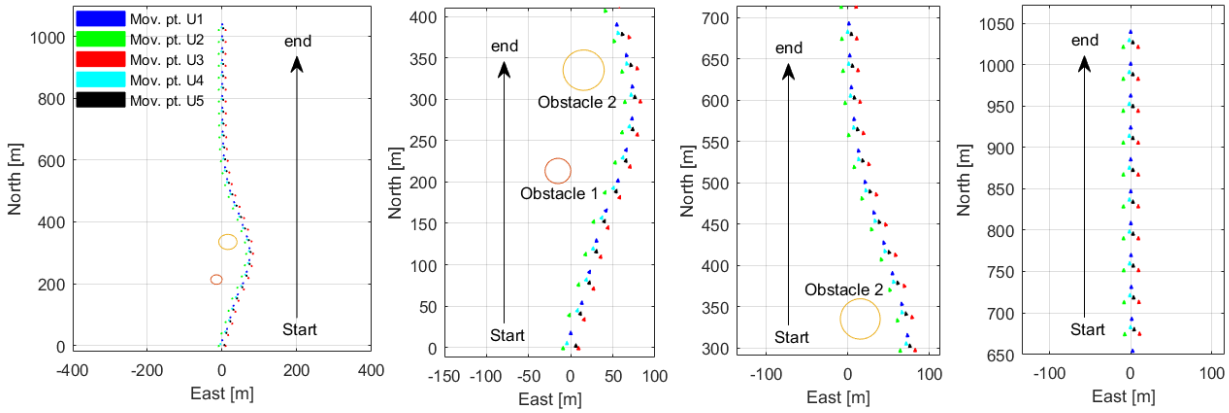


Figure 4.21: Closer view for the moving points position

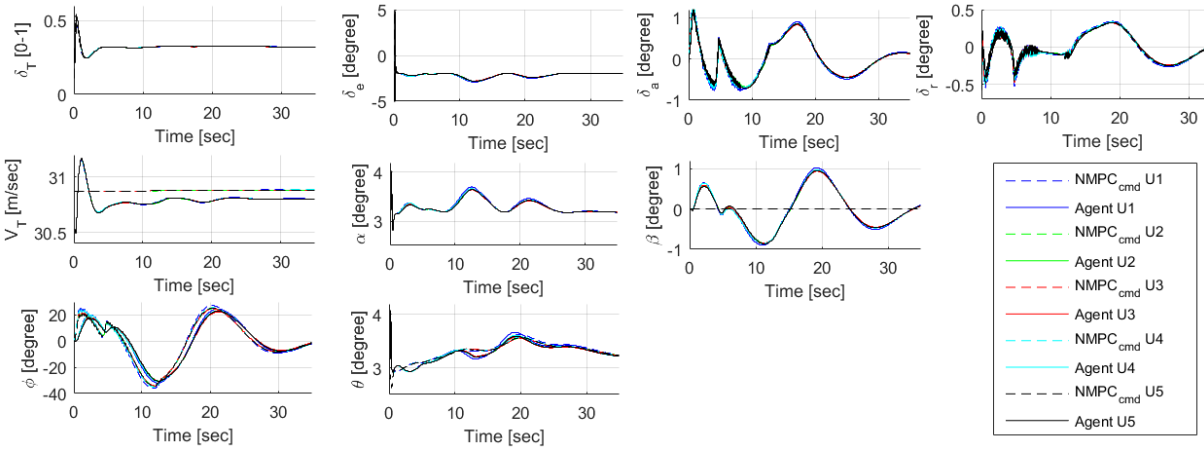


Figure 4.22: Morphing potential implementation with the moving mesh methods

4.1.9 Summary of Moving Mesh Methods Swarm Simulation

The virtual point formation logic and moving mesh methods are utilized to plan the moving points to maintain the multi-agent formation coherently. Dynamic constraints (e.g., the minimum turning radius) are added to produce dynamically feasible flight paths. Decentralized NMPC and moving point guidance are applied to track the planned moving points. Four different flight trajectory scenarios are simulated using five fixed-wing aircraft with high speed and high inertia. As the simulation results show, the proposed decentralized guidance and control and centralized navigation are effective to keep the formation coherency (the outer agents distances are always kept at the desired distance as it was presented in the previous sections), avoiding inner collisions (the farthest distances from the obstacle are met the criteria or conser), and performing challenging maneuvers (such as 180 turns and sinusoidal curves). Wind gust is simulated to validate robustness, coherency, and internal collision avoidance. Regardless of the formation shape, the inner agents are kept inside of the formation in a way that makes the mesh as uniform as possible. The moving mesh methods as the navigation algorithm has advantages such as scalability and adaptivity regardless of the number of agents and the formation shape. As long as the outer agents are defined by the formation shape, the inner agents can be located inside of the formation by the moving mesh methods. Two additional simulations have been done to show the application of the moving mesh methods with the intelligent navigation and the morphing potential field. The collaboration with the intelligent navigation algorithm guides the virtual leader to follow the given waypoint box. Consequently, the moving mesh methods are utilized to operate the science mission like the remote sensing of ice sheets. Lastly, the morphing potential field is also added to the moving mesh methods to avoid the obstacles.

4.2 Phasic Navigation Simulation Result

This section presents the results of the phasic navigation algorithms. This simulation is to verify the phasic navigation algorithm using practical examples. As it is mentioned in the previous section, the large waypoint box is given for the science mission. This algorithm goes through the phase of each navigation to form the formation. The first phase is the aggregation and assignment. The key part of this stage is to match all agents' velocity direction to the desired direction. The initial heading and position for six Skyhunters is presented in Fig. 4.24 and the following table. Fig. 4.23 (a) shows the position of the moving

Table 4.7: Initial heading angle for all agents

	U_1	U_2	U_3	U_4	U_5	U_6
ψ_0 [deg]	-164	107	-122	72	-93	18

point and all agents throughout the simulation. The different stages (Fig 4.23 (b), (c), (d)) are presented individually to show each stage better. In the first phase of the phasic navigation, all moving points aggregated to the desired direction (90 degree of the heading angle), which is east in this simulation. After all agents' moving points aggregates, the Hungarian algorithm starts to assign each moving point to the desired formation position sequentially. The virtual terminals are updated based on the arrival of the moving points to the formation position. Figure 4.25 shows the sequentially assigned formation with the updated virtual terminals. The hollow red triangle shows the desired formation position and different colors of triangles presents each agents. As it shows that the virtual terminals are updated when each agent arrives the assigned formation position. Finally, all agents' moving points arrive to the desired formation and the moving mesh algorithm correct the inner agents' position by using MMPDE as it was discussed in the previous section. Figure 4.25 shows that colored triangle (agents' moving point) arrived at the hollow triangle (the virtual terminal). In addition, 3D curvature control shows the altitude also converged to the virtual leader's direction, see Fig. 4.24. As a result, the random heading angles (Table 4.7) are converged to the desired direction (90 degrees of the heading angle) by the curvature control

algorithm. Then, the formation shape is formed by assigning agents to the formation position using Hungarian algorithm, see Figure 4.25. Finally, the moving mesh methods holds the formation, see Figure 4.23(d).

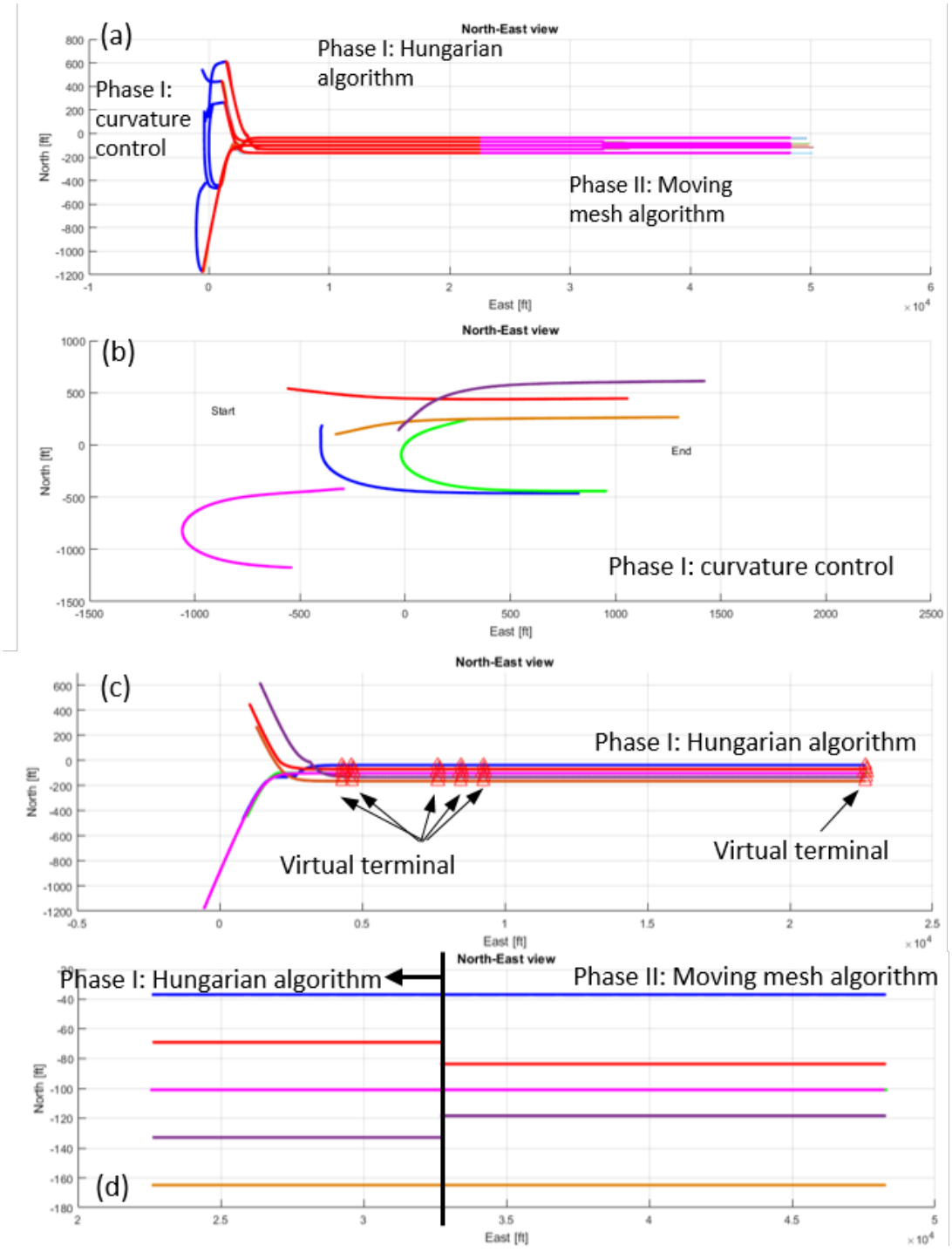
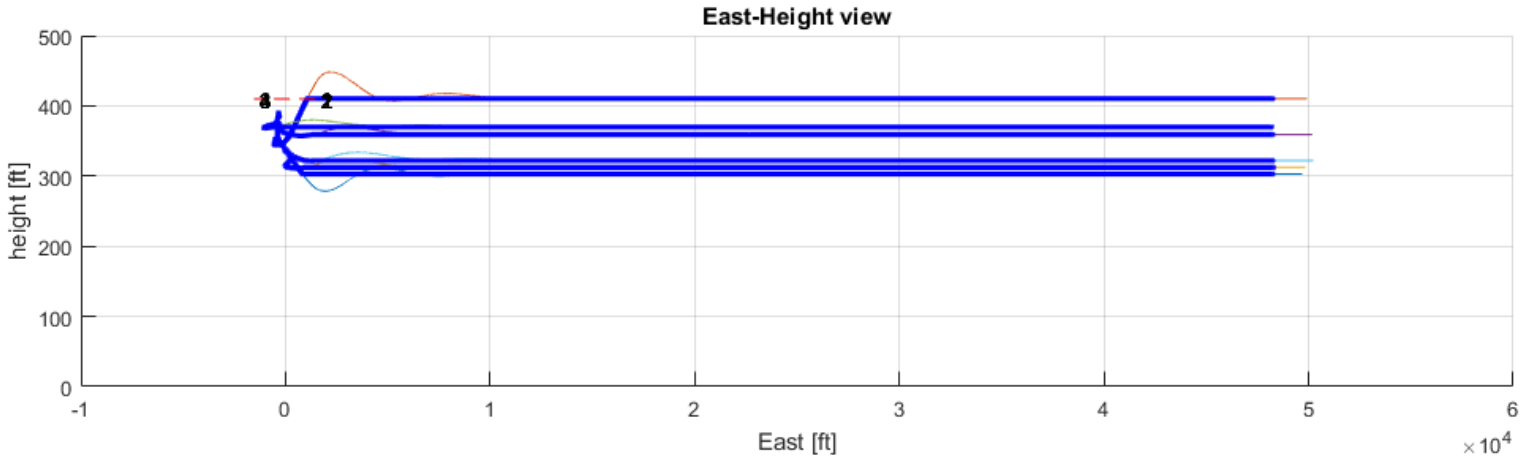


Figure 4.23: Phasic navigation algorithm simulation result



U1	U2	U3	U4	U5	U6
P_0=-25deg/sec	P_0=-19deg/sec	P_0=-23deg/sec	P_0=24deg/sec	P_0=26deg/sec	P_0=25deg/sec
Q_0=13deg/sec	Q_0=0deg/sec	Q_0=0deg/sec	Q_0=19deg/sec	Q_0=-6deg/sec	Q_0=-9deg/sec
R_0=8deg/sec	R_0=-2deg/sec	R_0=19deg/sec	R_0=2deg/sec	R_0=-12deg/sec	R_0=11deg/sec
ϕ_0=-29deg	ϕ_0=-7deg	ϕ_0=11deg	ϕ_0=-15deg	ϕ_0=-15deg	ϕ_0=20deg
θ_0=-9deg	θ_0=11deg	θ_0=6deg	θ_0=0deg	θ_0=13deg	θ_0=3deg
ψ_0=-164deg	ψ_0=107deg	ψ_0=-122deg	ψ_0=72deg	ψ_0=-93deg	ψ_0=18deg
V_{T_0}=48.0737ft/sec	V_{T_0}=52.6827ft/sec	V_{T_0}=48.3997ft/sec	V_{T_0}=45.575ft/sec	V_{T_0}=47.1495ft/sec	V_{T_0}=54.1865ft/sec
N_0=186.5735ft	N_0=540.2665ft	N_0=251.2378ft	N_0=102.3213ft	N_0=-420.8472ft	N_0=139.2536ft
E_0=-394.576ft	E_0=-558.6647ft	E_0=305.624ft	E_0=-331.4257ft	E_0=-290.9901ft	E_0=-32.0534ft
H_0=371ft	H_0=344ft	H_0=327ft	H_0=375ft	H_0=384ft	H_0=335ft
δ_{T_0}=63.2%	δ_{T_0}=63.2%	δ_{T_0}=63.2%	δ_{T_0}=63.2%	δ_{T_0}=63.2%	δ_{T_0}=63.2%
δ_{E_0}=1.5deg	δ_{E_0}=1.5deg	δ_{E_0}=1.5deg	δ_{E_0}=1.5deg	δ_{E_0}=1.5deg	δ_{E_0}=1.5deg
δ_{A_0}=0deg	δ_{A_0}=0deg	δ_{A_0}=0deg	δ_{A_0}=0deg	δ_{A_0}=0deg	δ_{A_0}=0deg
δ_{R_0}=0deg	δ_{R_0}=0deg	δ_{R_0}=0deg	δ_{R_0}=0deg	δ_{R_0}=0deg	δ_{R_0}=0deg
α_0=0.73deg	α_0=0.73deg	α_0=0.73deg	α_0=0.73deg	α_0=0.73deg	α_0=0.73deg
β_0=0deg	β_0=0deg	β_0=0deg	β_0=0deg	β_0=0deg	β_0=0deg

Figure 4.24: East and height view - Hungarian algorithm in 3D space and initial conditions of all agents

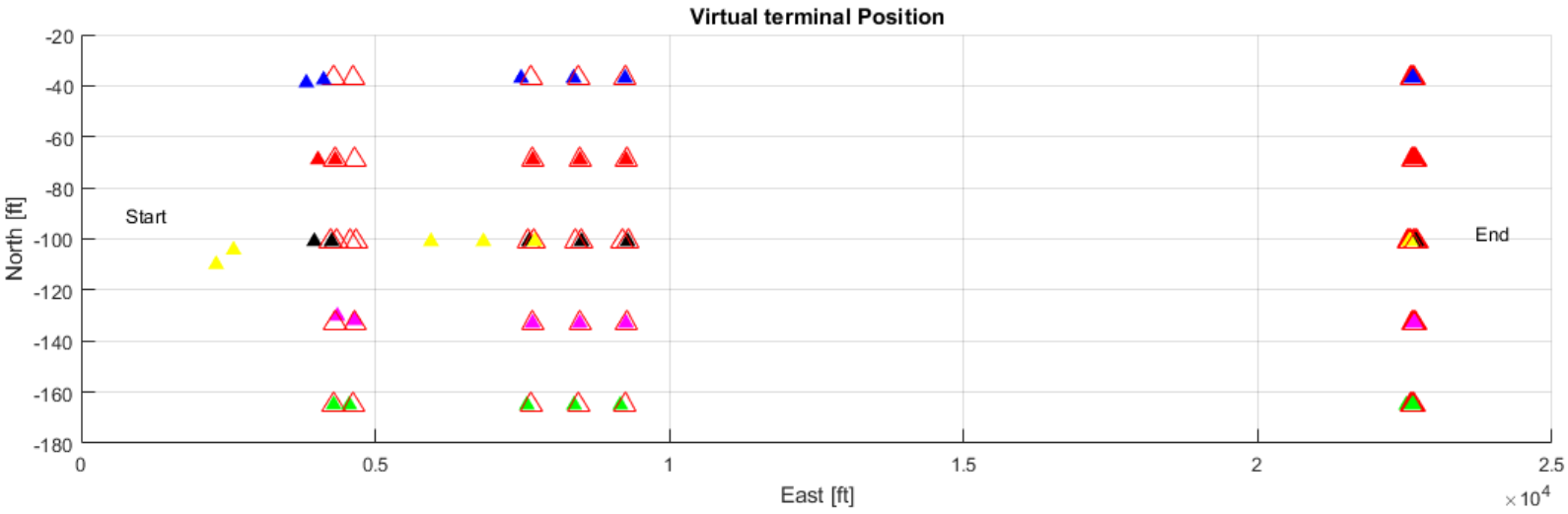


Figure 4.25: The virtual terminal moves sequentially by Hungarian algorithm

4.3 Monte Carlo exhaustive search

In this section, the result and practical advantage of Monte Carlo exhaustive search is presented. Exhaustive Monte Carlo simulation is conducted for various flight test scenarios and captures the error in the implementation very effectively. In this section, the swarm flight Monte Carlo simulation is introduced. In order to begin the Monte Carlo simulation, the following range of states are selected randomly.

Table 4.8: The range of states for Monte Carlo

Item	Minimum	Maximum	Interval
North [ft]	-588	659	100
East [ft]	-1700	1544	100
Height [ft]	250	450	10
ψ [deg]	-180	180	15
ϕ [deg]	-30	30	5
θ [deg]	-10	10	5
P [deg/s]	-50	50	5
Q [deg/s]	-20	20	5
R [deg/s]	-20	20	5

For the swarm simulation, 1405 cases are simulated. The filtering criteria check if the distance between two agents is less than double wing span ($2b$). As a result, 478 cases (34% of the simulation) are closer than the double wing span. The error in the assignment of the formation is found due to the implementation failure. Two agents are able to be closer since they are assigned to the same formation position. In addition, the simulation produced null values (NaN) of the moving point position velocity due to the initial heading angle. The initial virtual leader position is generated by the average velocity vector of two agents. However, two agents moving in the opposite direction to one another meaning the difference of two agents' heading angle is 180 degree, see Fig. 4.26. This error is resolved by adding the little bias of the heading angle (2 degrees, this bias can be the design parameter) to mitigate the simulation failure due to the special situation when the average velocity is a zero. In summary, the Monte Carlo simulation is able to be applied for any algorithm development to check the implementation flaws against random initial condition before the flight tests. The

common error during the implementation is the division by zero. Monte Carlo simulation has the ability to capture of the error cases by drawing random initial conditions.

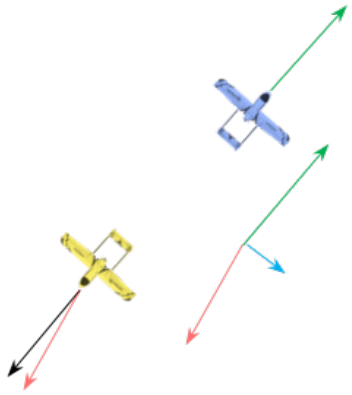


Figure 4.26: Failed cases during the Monte Carlo simulation

4.4 Multi-agent Autonomous System Hardware-in-The-Loop (HiTL) Testbed

In this section, HiTL simulations are performed for four different scenarios: swarming, swarming with external disturbances, communication loss, and collision avoidance. Each case is discussed with results for the moving point and the aircraft trajectory together with the user defined waypoints' trajectory. A simple setup of the close proximity formation flight is tested as the first scenario, shown in Fig. 4.27 (left). Two agents start to fly left of the race track with different heading angle (ψ): the first agent has 45 degrees and the second agent has 135 degrees. Fig. 4.27 (right) shows the distance between agents during the simulation. During the first 20 seconds, the distance between agents increases from 60 to 110 m since the agents change their own heading (45 and 135 degrees) to the average heading angle (90 degrees of the heading angle). Then, the agents are gradually converging to the desired distance (70 m). This result is not satisfactory since the distance between aircraft is ended up at 50 meter (28.6% error from the desired distance) after 200 seconds of the swarm flight

due to the fixed length of $d_{dR_{Lat}}$.

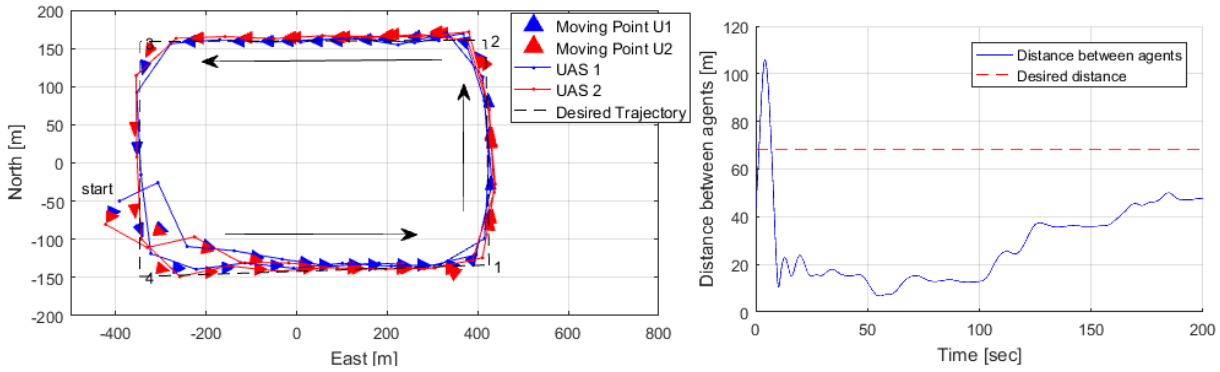


Figure 4.27: Case 1 LEFT - Blue Solid: Aircraft 1, Red Solid: Aircraft 2, Blue Triangles: Moving Points for aircraft 1, Red Triangles: Moving Points for aircraft 2 and Black Dashed: Desired waypoint Trajectory. Right: Distance between agents

Fig. 4.28 shows the snapshot of the ground station view during HiTL test.

Figure 4.28: Q-GCS snapshots

The second scenario is the communication loss by intentionally disconnecting the Xbee communication radio from one of the aircraft, shown in Figure 4.29. Fig. 4.29 (b) shows the moving point position throughout the simulation. As Fig. 4.29 (c) shows, the communication

is lost two times at 25 and 75 seconds. When the communication is lost, each aircraft waits for 2 seconds to check if the communication is recovered. Then, each agent changes the flight mode from swarm to solo and follow the desired race track. After the communication is recovered, two agents reinitialize the swarm flight mode and the path planning changes from solo to swarm mode. This shows the robustness of the communication loss for the navigation algorithm. In addition, the ROS integration of communication link is also robust since the communication algorithm starts automatically after the radio port is detected by the Tegra. Fig. 4.29 (a) shows the position of the moving point and aircraft. Fig. 4.29 (d) shows the distance between agents in respect to north and east component. Due to reinitialization of swarm navigation algorithm, the tracking of formation shape fluctuates. The reason why it fluctuates is that the moving points are planned to follow the race track starting from the current aircraft position, when the navigation algorithm is reinitialized. Then, the agents starts to follow the race track and tries to form the formation shape again when the communication link is recovered.

The third scenario is a HiTL simulation with the wind of NE wind at 13 knots. Figure 4.30 (left) shows the position of agents and moving points. Due to the wind, the agents have drifted (122.5 meters maximum at the switching distance area) to the east direction. The distance between agents is also impacted by the wind. Figure 4.30 (right) shows that the distance between agents has average 32 m error when the desired distance is 70m. However, the vehicle did not collide with one another since the minimum distance between agents is 20 m, which is five times larger than the wingspan of DG-808 (4m).

The final simulation is the collision avoidance verification by using the morphing potential field in HiTL testbed. One of the agents is assumed to be stationary on the ground while it communicates and sends its GPS position and velocity to the other flying aircraft. Figure 4.31 shows the position of agents and moving point. The red triangle indicates the static agent on the ground (The east coordinate is 0m and the north coordinate is -133.5m). The circle around this agent presents the size of the obstacle which is 24.4m. As Fig. 4.31 shows,

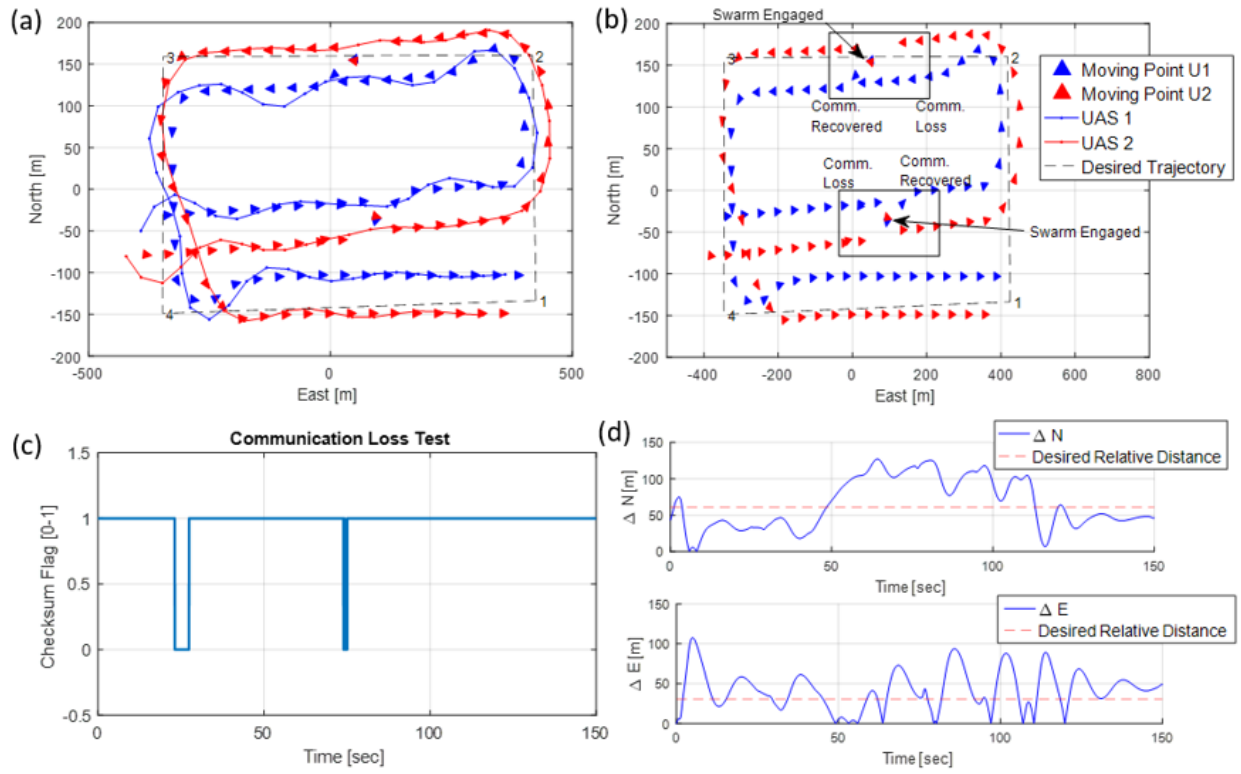


Figure 4.29: Case 2 Blue Solid: Aircraft 1, Red Solid: Aircraft 2, Blue Triangles: Moving Points for aircraft 1, Red Triangles: Moving Points for aircraft 2 and Black Dashed: Desired waypoint Trajectory

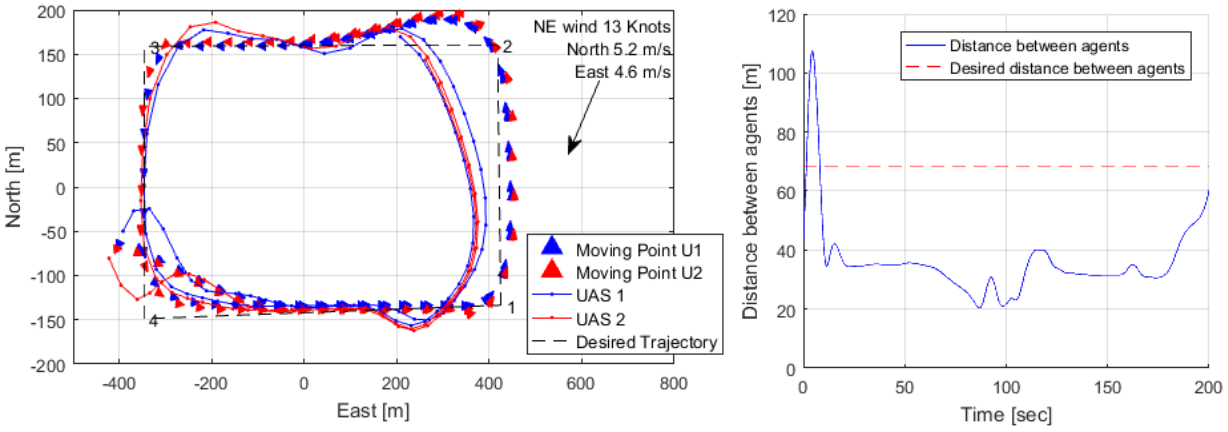


Figure 4.30: Case 3 - LEFT - Blue Solid: Aircraft 1, Red Solid: the aircraft 2, Blue Triangles: Moving Points for the aircraft 1, Red Triangles: Moving Points for aircraft 2 and Black Dashed: Desired waypoint Trajectory. Right: Distance between agents

the moving points are planned to avoid the obstacle. Consequently, the other agent could avoid the obstacle. For avoiding the obstacle, the result is considered to be acceptable if the

farthest distance from the obstacle is between σ and 2σ where σ is the radius of the obstacle. In this simulation, the result is satisfactory since the farthest distance from the aircraft to the obstacle is 37.8m which is larger than 24.4m(σ) and smaller than 48.78m (2σ).

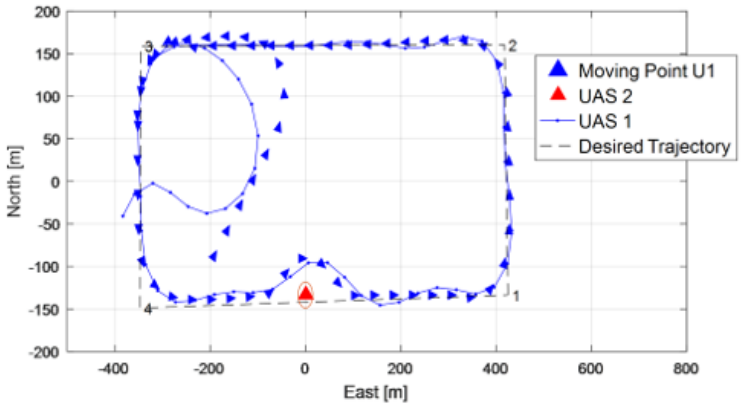


Figure 4.31: Case 4 - Blue Solid: Aircraft 1, Red Solid: Aircraft 2, Blue Triangles: Moving Points for aircraft 1, Red Triangles: Moving Points for aircraft 2 and Black Dashed: Desired waypoint Trajectory

In conclusion, the implementation of two agents in avionics has been validated since the system could generate states and control correctly (the aircraft followed the moving point) and visualized in the ground station. The performance of keeping the formation is not satisfactory since average of 29% distance error from the desired formation distance (70m). The communication delay for 2 and 4 seconds is applied and the mechanism of changing the flight mode between solo and swarm flight is validated by observing the moving point position, see Fig. 4.29(b). In addition, the wind field (NE wind 13 knots) has been applied to conduct more practical hardware-in-the-loop test and maximum 30% tracking error is occurred with incapability of holding the formation.

4.5 Flight Test Validation

In this section, the simulation and the flight test results are compared to validate the proposed GNC algorithms discussed in Chapter 2. The flight plan is outlined to incrementally validate the system from the solo flight to the swarm flight.

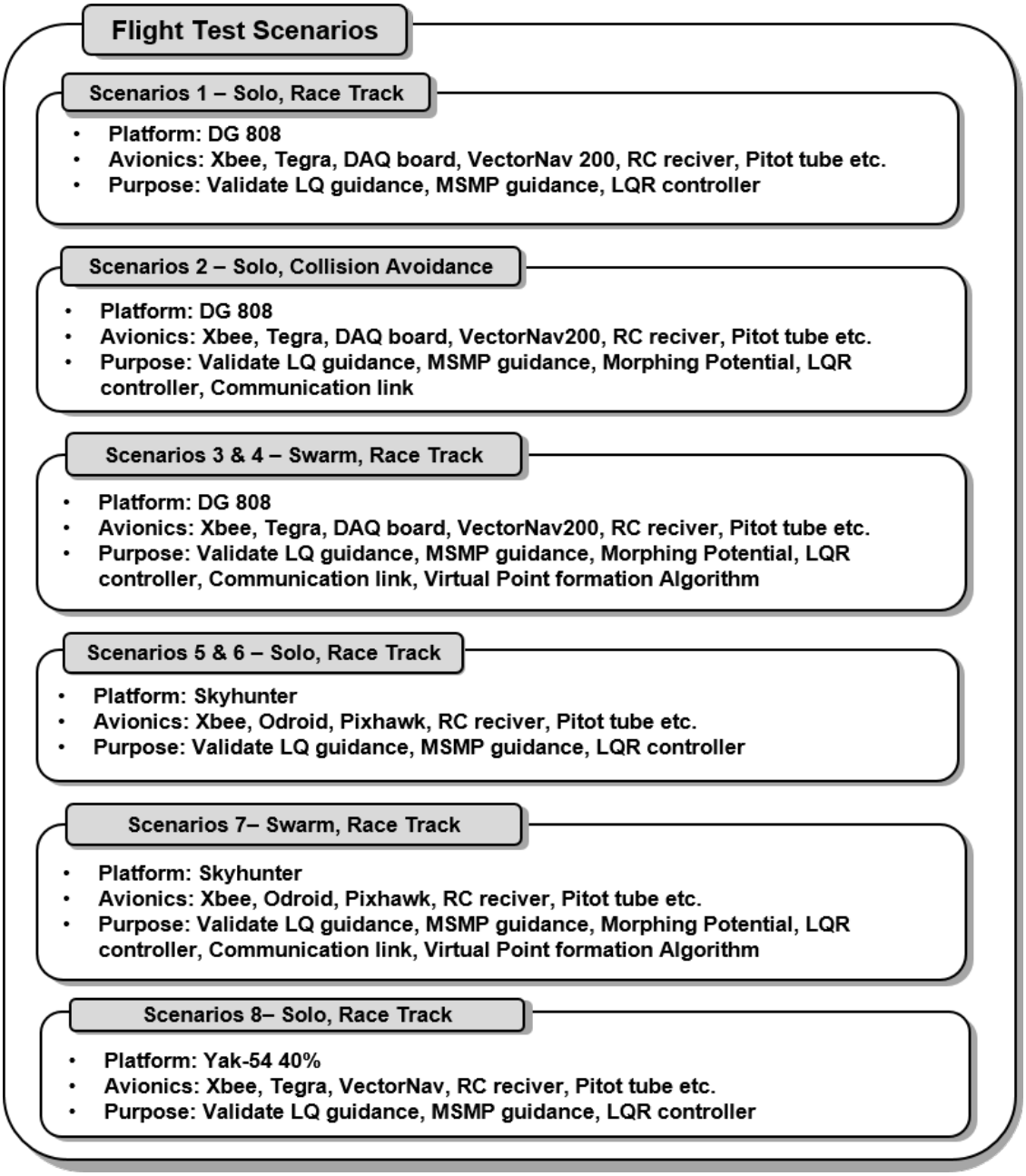


Figure 4.32: Flight Test Scenarios

Each scenario has been designed sequentially in order to conduct the multi-agent flight test. DG-808 flight test starts with the solo mode to validate the guidance and control for each agent. In addition, the collision avoidance algorithm is validated before the multi-agent flight tests are performed. Then, two of DG-808 are used for the formation flight test. In order to validate GNC for different platform, Skyhunter is used for the solo flight test and

two of Skyhunter are used for the swarm flight test. In order to validate developed GNC for various platform, the flight test has been conducted for Yak-54 40%.

The following table provides the criteria for assessing the validation. All performance is assessed except the turning maneuver.

Table 4.9: Performance assessment criteria

Items	Acceptable Range	Items	Acceptable range
P	$\pm$25 deg/sec	ϕ_{cmd}	10% Overshoot or $\pm$3 degree
Q	$\pm$15 deg/sec	θ_{cmd}	10% Overshoot or $\pm$3 degree
R	$\pm$15 deg/sec	$V_{T_{cmd}}$	10% Overshoot
ϕ	$\pm$60 deg	Lateral TE	10% Overshoot
θ	$\pm$10 deg	Longitudinal TE	10% Overshoot
$\dot{\delta}_T$	$\pm$8 %/sec	$\dot{\delta}_A$	$\pm$20 deg/sec
$\dot{\delta}_E$	$\pm$10 deg/sec	$\dot{\delta}_R$	$\pm$10 deg/sec
Collision avoidance	$\sigma \leq distance \leq 2\sigma$		

where TE is the tracking error. The 5% more overshoot is considered as the satisfied performance comparing to Table 4.1 by considering the flight tests conducted in unstructured environment.

4.5.1 Solo flight: DG-808, March 9th 2017

- DG-808 Solo Flight, March 9th 2017

  In this flight test, one of DG-808 flew in the autonomous mode and the results are compared with the simulation. Figure 4.33 presents the aircraft trajectory and the rate of altitude. Figure 4.34 presents the guidance output and moving point position. Figure 4.35 and 4.36 shows the longitudinal and lateral-directional states and control surfaces deflection, respectively.

  – Navigation performance

    As Fig. 4.33 shows, the aircraft follows the race track pattern well based on Table 4.9. The lateral tracking quality is assessed at only relatively small roll angle ($\pm$5 deg). The average error is 29.6 ft (6.4% overshoot $<$ 10% criteria) in the lateral tracking after

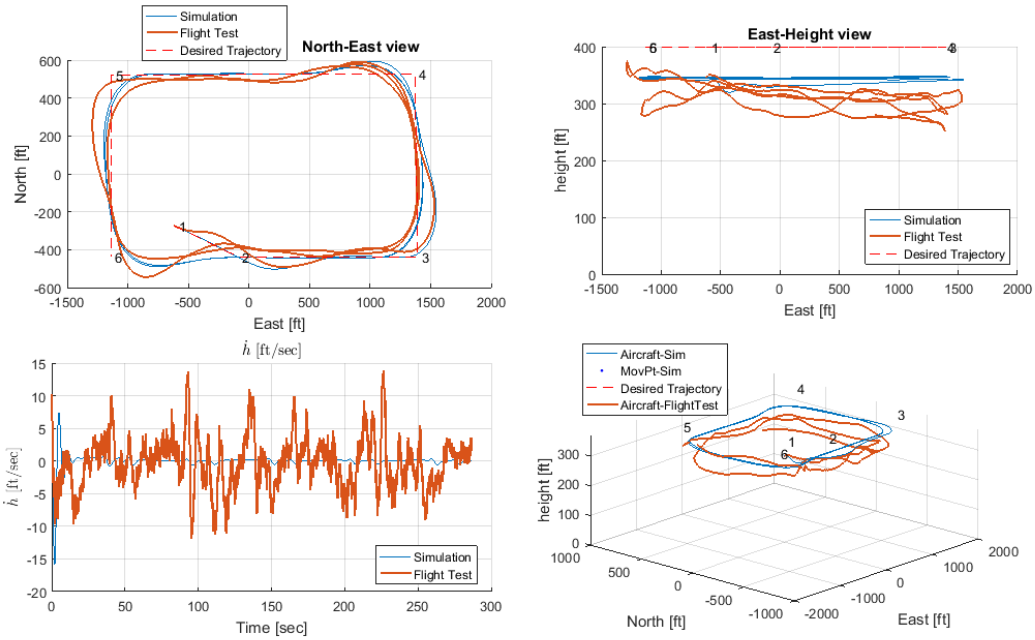


Figure 4.33: DG808 Solo Flight test and simulation comparison, Trajectory and rate of altitude changes

aircraft converges to the longer legs of the race track (line made of waypoint 4 and 5 and waypoint 6 and 3). In addition, Ref. [2] shows the average lateral tracking error 185.9 ft which is larger than this flight lateral tracking error (29.6 ft). The waypoint modification worked efficiently when the autopilot is engaged at waypoint 1 in Fig. 4.33 (top left). The intersection point is found at waypoint 2 and provide the smooth transition to the race track pattern meaning the pitch and roll angle commands are relatively smaller (e.g., maximum roll angle command on the intersection line is 15 degree) comparing to the turning maneuver (up to -34 degrees). In addition, the altitude tracking fluctuates from 253 to 373.7 ft (average 17.39% overshoot) from the desired altitude (345 ft AGL) in the flight test while the simulation shows better altitude tracking (maximum 32 ft error which is 9.3% overshoot) because it does not have any wind information. The altitude tracking is worse than Ref. [2] (13.12 ft overshoot in the altitude tracking). This is also shown well in Fig. 4.33 (bottom left) that the altitude rate changes between -10 and 10 ft/sec in the flight test while the simulation has smaller altitude rate ($\pm 0.4 ft/sec$

after the altitude is settled at the desired altitude). Since the overshoot of longitudinal tracking error is larger 10% criteria, the longitudinal control needs more improvement to reduce the overshoot. Figure 4.34 (right) presents the moving point position comparison and the simulation and flight test results are matched very well (overshoot at the right north corner is 9.6% error which is less than 10%). The intelligent waypoint modification and index decision algorithm is validated since the intersection point and index are generated as same as the simulation does. The LQ guidance path planning are validated since the moving point followed the race track as simulation does. The lateral tracking quality met the criteria (6.4% overshoot $<$ 10% criteria) but the longitudinal tracking is not met the criteria.

– Guidance and control performance

  Figure 4.34 (left) presents the guidance outputs. The velocity command is identical between the simulation and the flight test. Both has 59.07 ft/sec as the speed command. The roll angle commands matches during the turns but the magnitude of the roll angle command from the flight test is larger than simulation (maximum 16 degree) due to the wind field. The pitch angle command has the relatively large discrepancy (maximum 7 degrees) comparing to the simulation command values that are between 0 and 3 degrees. The possible reasons of the discrepancy are the uncertainty of the dynamic model and the external disturbances. Since the magnitude of the pitch angle command difference (maximum 7 degrees) is fairly significant comparing to the simulation (average 1 degree), the discrepancy can be explained by the lack of gust modeling in the simulation environment. Similarly, the longitudinal controller has a more loose tracking (maximum 7 degree error in the flight test) than the simulation (average 2 degree after the controller settled the pitch angle), shown in Figure 4.35. In case of the throttle, we can observe that 15% less throttle deflection is applied in the flight test. This discrepancy comes from the uncertainty in the longitudinal dynamic model (e.g., engine model and drag estimation). The drag model might be over-estimated or

the engine model might be under-estimated to generate the thrust force. On the other hand, the roll angle is tracked undesirably since the maximum overshoot is 18% which is larger than 10% criteria from Table 4.9, although the roll rate data was noisy (The signal oscillates 5Hz in the roll rate while the simulation does not have any oscillation).

In general, DG-808 could follow the race track autonomously. The lateral-directional tracking quality is acceptable and the longitudinal tracking quality is not satisfied. All avionics worked correctly so all information could be retrieved from the aircraft and communicate to the ground station during the flight. The controller performance is not met the criteria defined in Table 4.9 due to the impact of external disturbances. In the next section, the collision avoidance algorithm is validated before the swarm flight is performed.

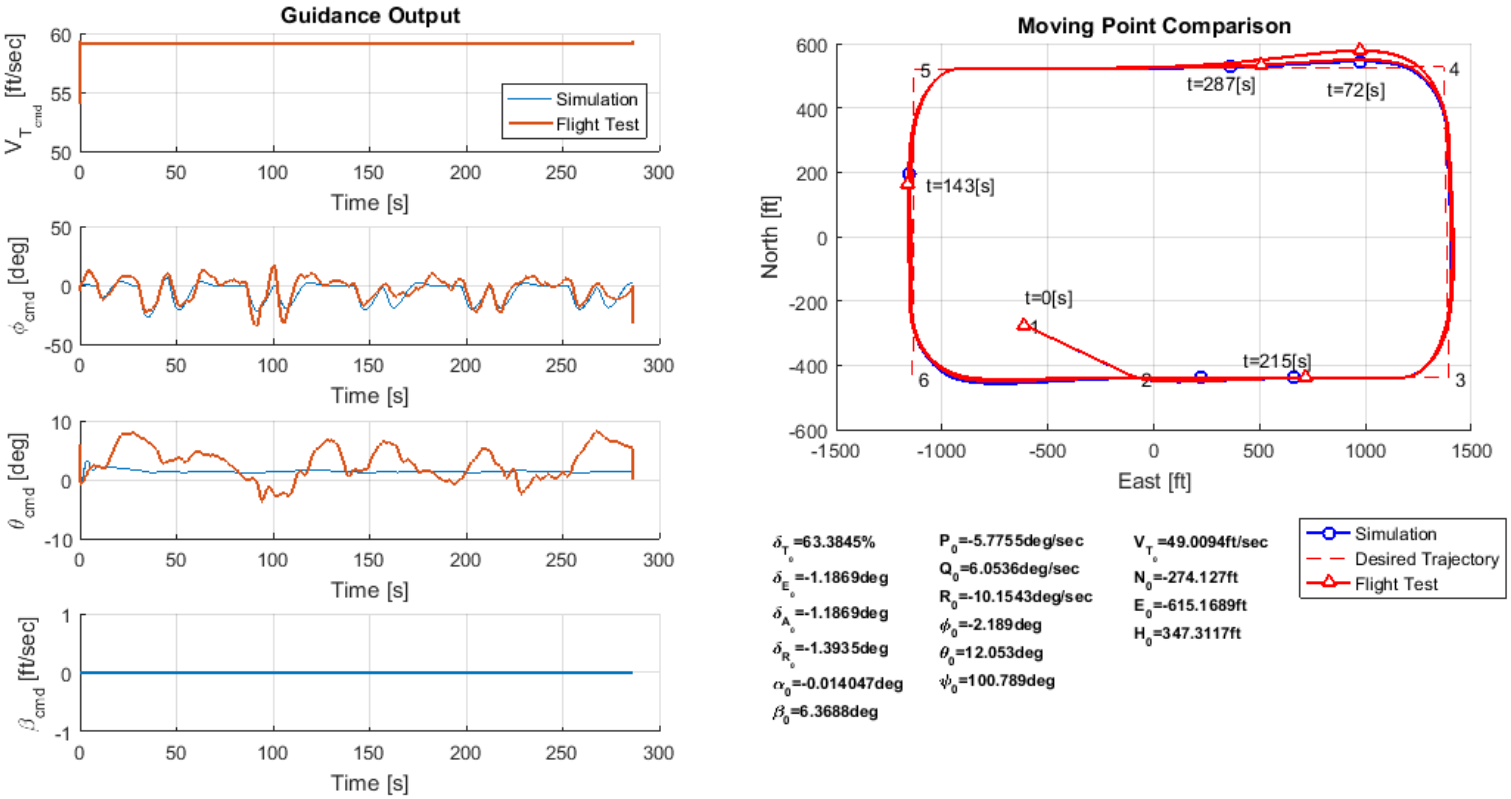


Figure 4.34: DG808 Solo Flight test and simulation comparison, Guidance outputs and Moving point position

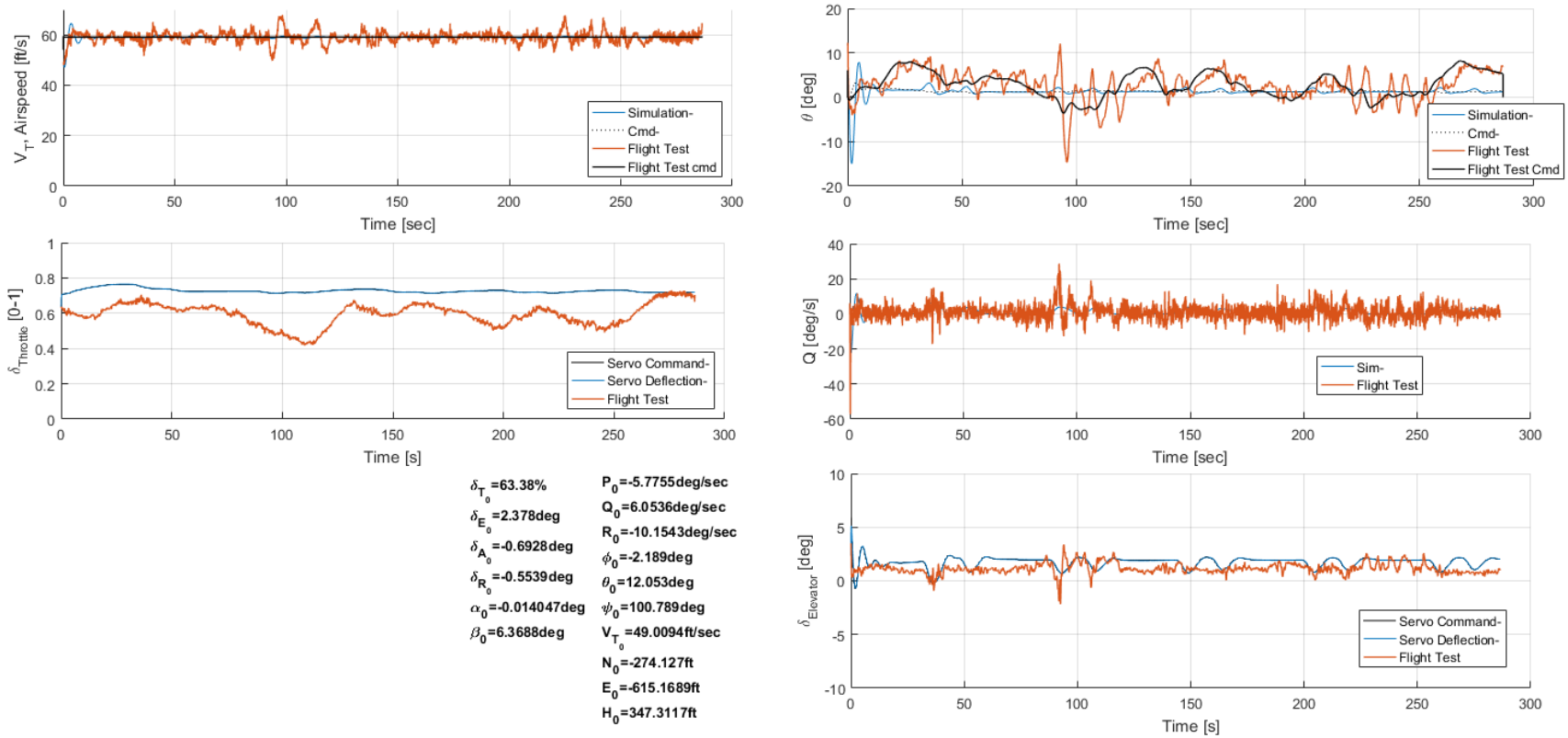


Figure 4.35: DG808 Solo Flight test and simulation comparison, Longitudinal States and Control Surfaces

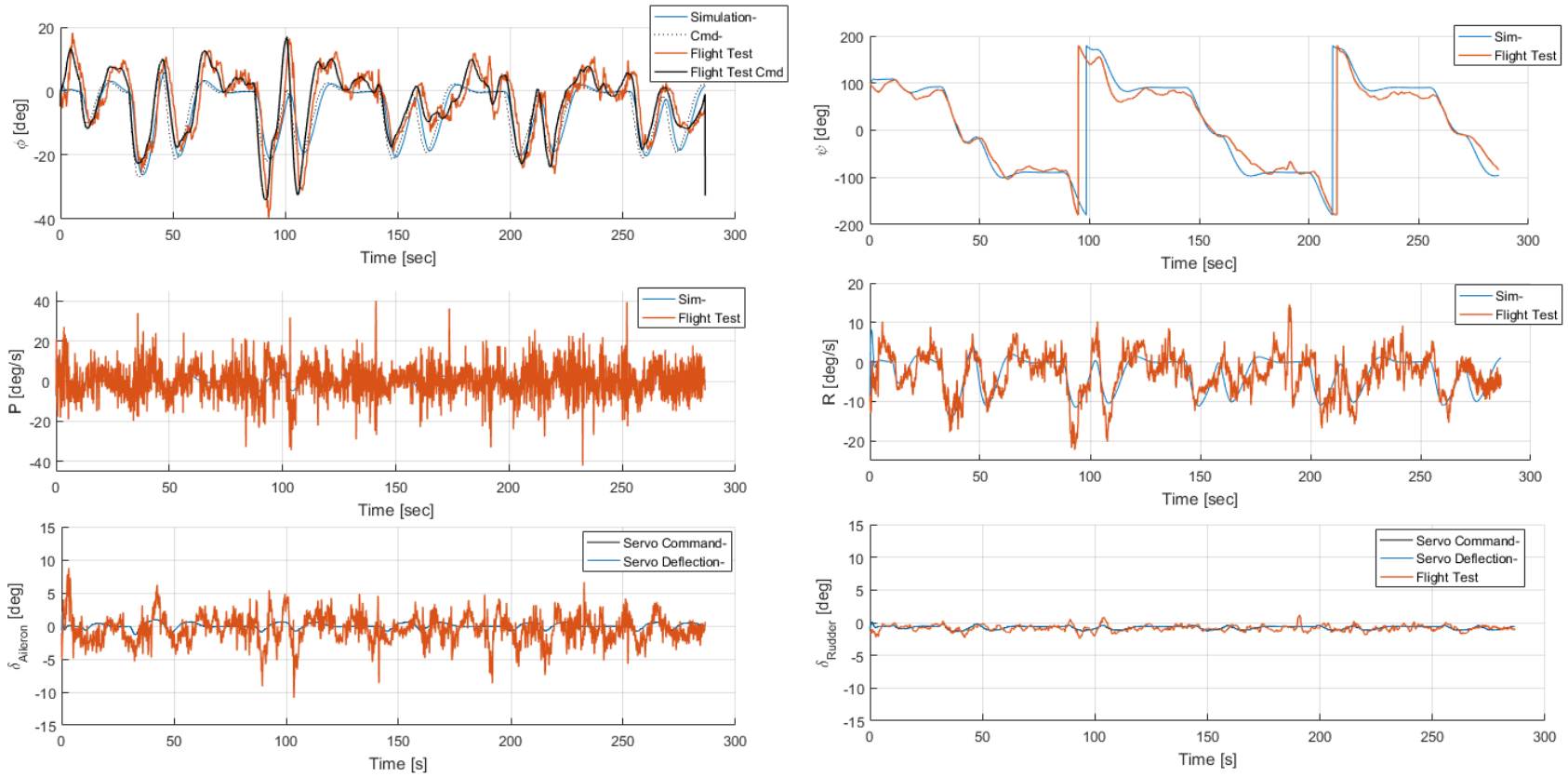


Figure 4.36: DG808 Solo Flight test and simulation comparison, Lateral-directional States and Control Surfaces

- DG-808 Flight Collision Avoidance, March 9th 2017

  This flight test is conducted to validate the collision avoidance path planning using the morphing potential field. In this flight test, one of the agents was placed on the ground along the southern side of the racetrack pattern while the other aircraft flies around the race track. Figure 4.37 shows the trajectory and the altitude rate. Figure 4.39 presents the guidance output and the moving point position comparison. Figure 4.40 and 4.41 present the longitudinal and lateral-directional states and control surface deflections.

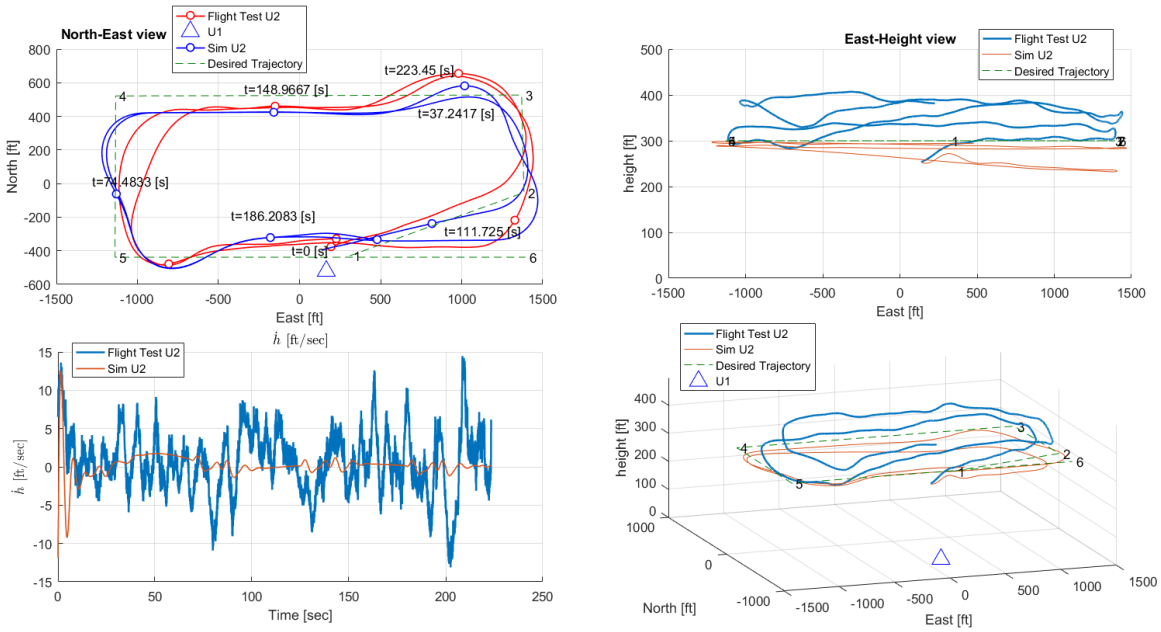


Figure 4.37: DG808 Collision Avoidance Flight test and simulation comparison, Trajectory and rate of altitude changes. The morphing potential radius is set to 200 ft.

  – Navigation performance

    Figure 4.38 shows the closer look of the trajectory and the agent on the ground.

    The moving points are planned to avoid the obstacle which has the radius of 200 ft. In the flight test, the moving point position is acceptable since the farthest distance from the obstacle (the agent on the ground) is 243.13 ft ($200 \leq 243.13 \leq 400$ ft) which meets the criteria defined in Table 4.9. Figure 4.38 shows the aircraft trajectory closer around the agent. The aircraft passes the obstacle two times and the farthest distances from the obstacle are 168.4 and 190.5 ft which does not meet the criteria. This phenomena

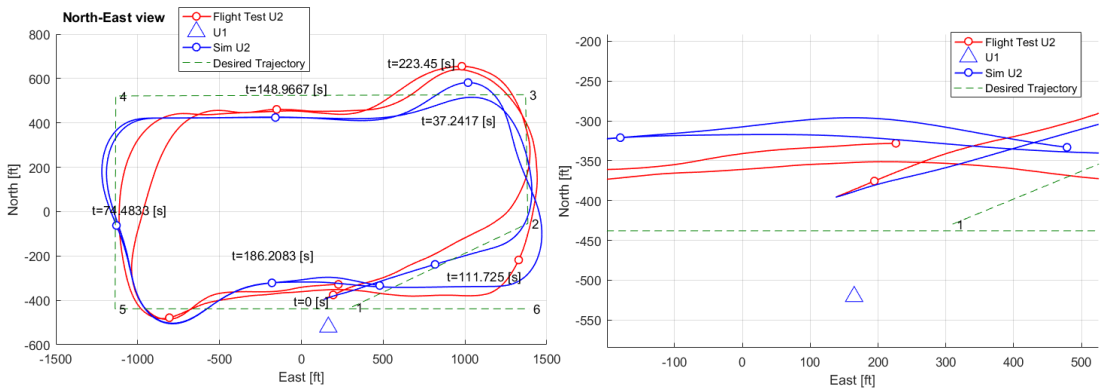


Figure 4.38: DG808 Collision Avoidance Flight test and simulation comparison, Close look around the agent on the ground

is caused due to the feature of the moving point guidance. The reference point is always ahead of the agent by the length of $d_{dR_{Lat}}$ which is 360 ft in this flight test. Therefore, the moving point avoids the obstacle sooner than the agent. More precise lateral tracking quality is required to meet the criteria. The waypoint modification and index decision algorithm is able to find the intersection point between the waypoint 3 and 6. In addition, the presence of the wind is clearly observed since the aircraft is drifted 154 ft to the east direction when it flies the northeast corner comparing to the simulation result. Regarding the altitude tracking, the same behavior is observed from the previous flight test result. The altitude fluctuates between 252.6 and 386 ft which is corresponding to 28.3% overshoot. The longitudinal tracking does not meet the defined criteria in Table 4.9.

– Guidance and Control performance

As the result discussed in the previous flight test, the guidance and control performance is similar. The pitch angle tracking is loose (the maximum error is 5 degrees $\geq$ 3 deg) for the longitudinal control performance (Figure 4.40). Besides the longitudinal tracking quality, the pitch angle commands has the large discrepancy (maximum 10 degress) due to the external disturbances. The On top of the aforementioned reasons from the previous flight test, the remedy of this phenomena would be two folds: the adjustment

of longitudinal guidance parameters and the longitudinal LQR weighting matrices (Q_{Lon} and R_{Lon}). As the impact of guidance and control parameters are discussed in Chapter 2, shorter L_{Lat} (e.g., from 360 to 300 ft) and larger $K_{p_{Lon}}$ (e.g., from 0.001 to 0.05) and $K_{I_{Lon}}$ (e.g., from 0.03 to 0.05) can help improve the longitudinal tracking. In this flight test, average 15% less throttle is used as well. In addition, the roll angle tracking has the maximum error 5 degrees which is 2 degrees larger than the acceptable range.

The purpose of this flight test is satisfied to validate the collision avoidance path planning. Specifically, the moving points are planned correctly to avoid the obstacle comparing to the simulation (the defined criteria for avoiding the obstacle is met). Also, the avoidance criteria is met for the generated moving point position. But the controller and guidance performance should be improved to meet the defined criteria.

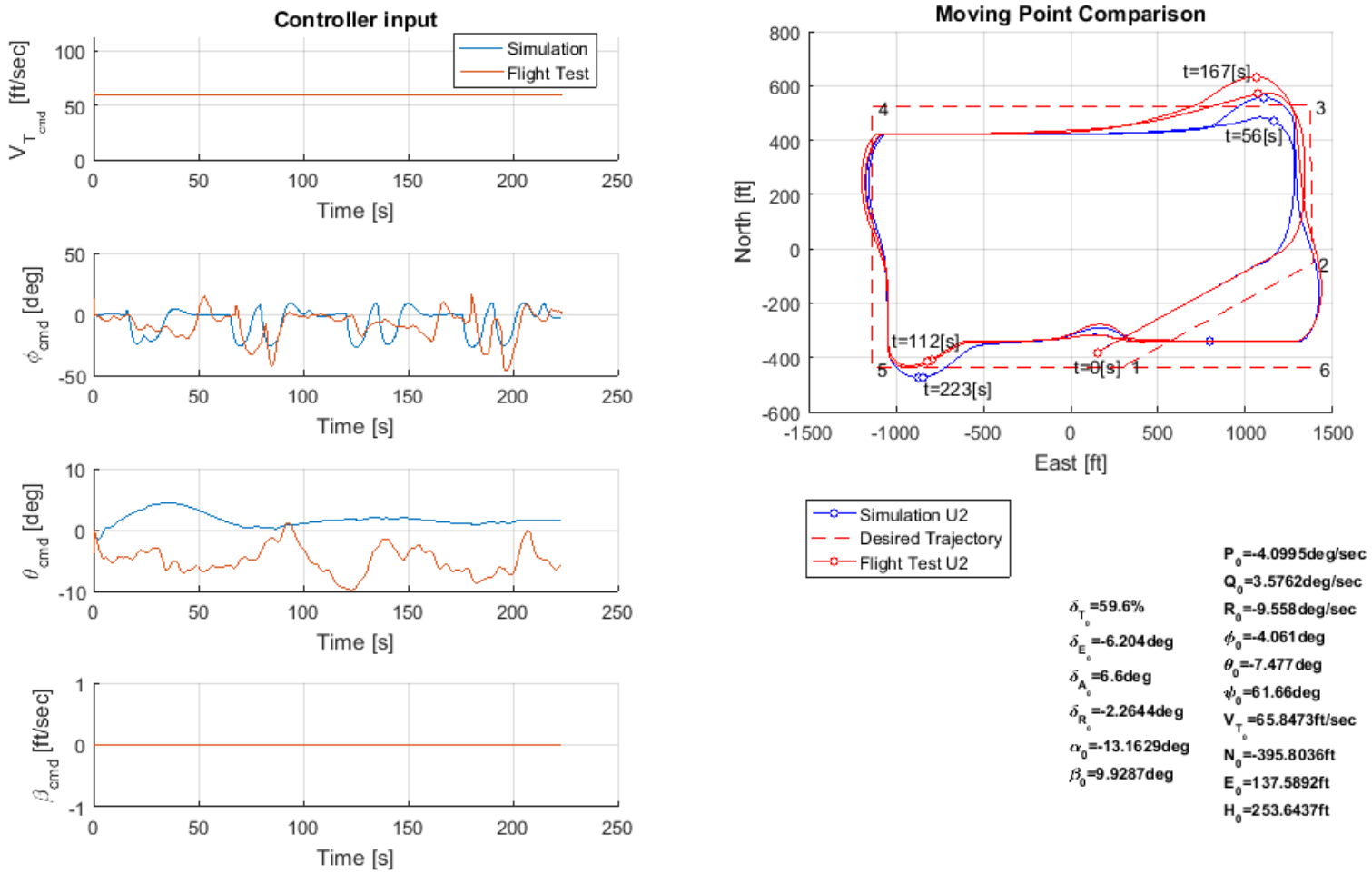


Figure 4.39: DG-808 collision avoidance flight test and the corresponding simulation comparison: the moving point position and guidance output

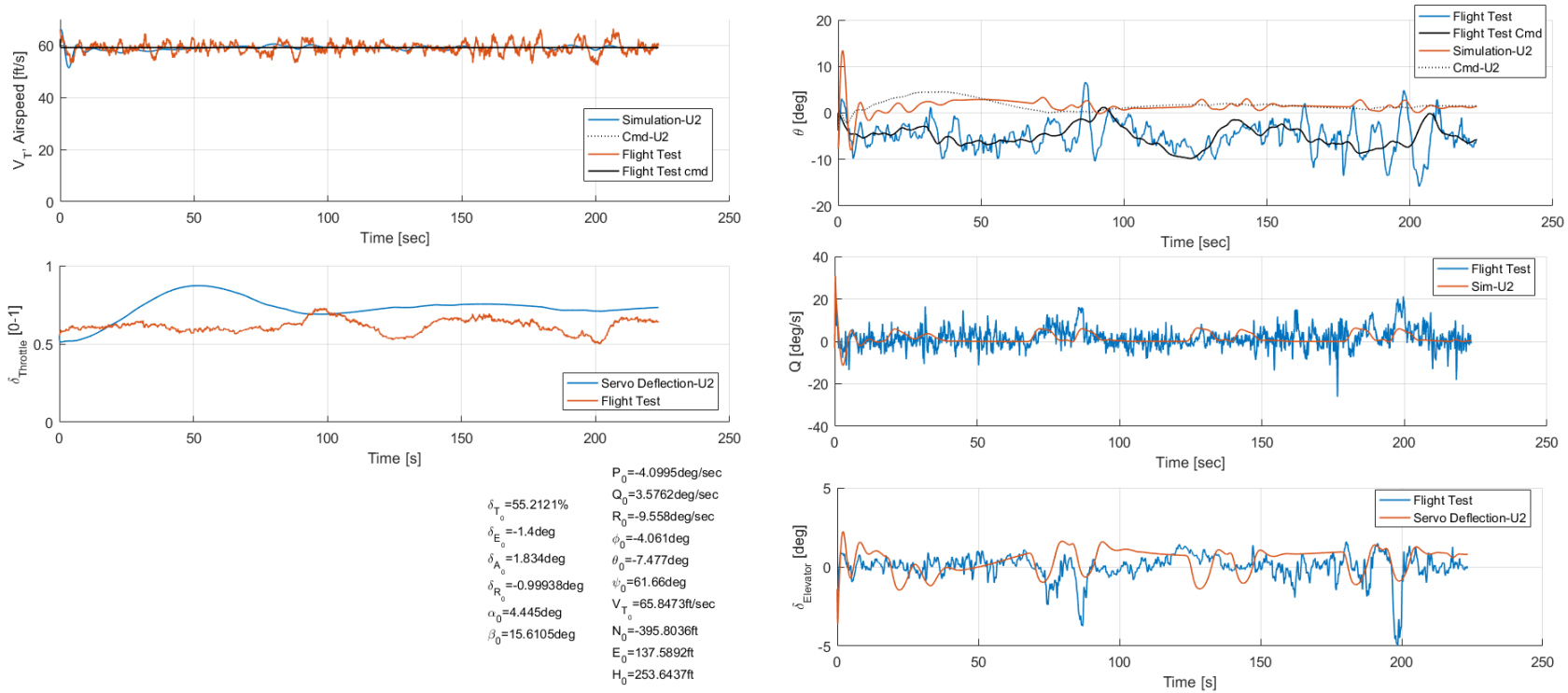


Figure 4.40: DG808 Collision Avoidance Flight test and simulation comparison, Longitudinal States and Control Surfaces

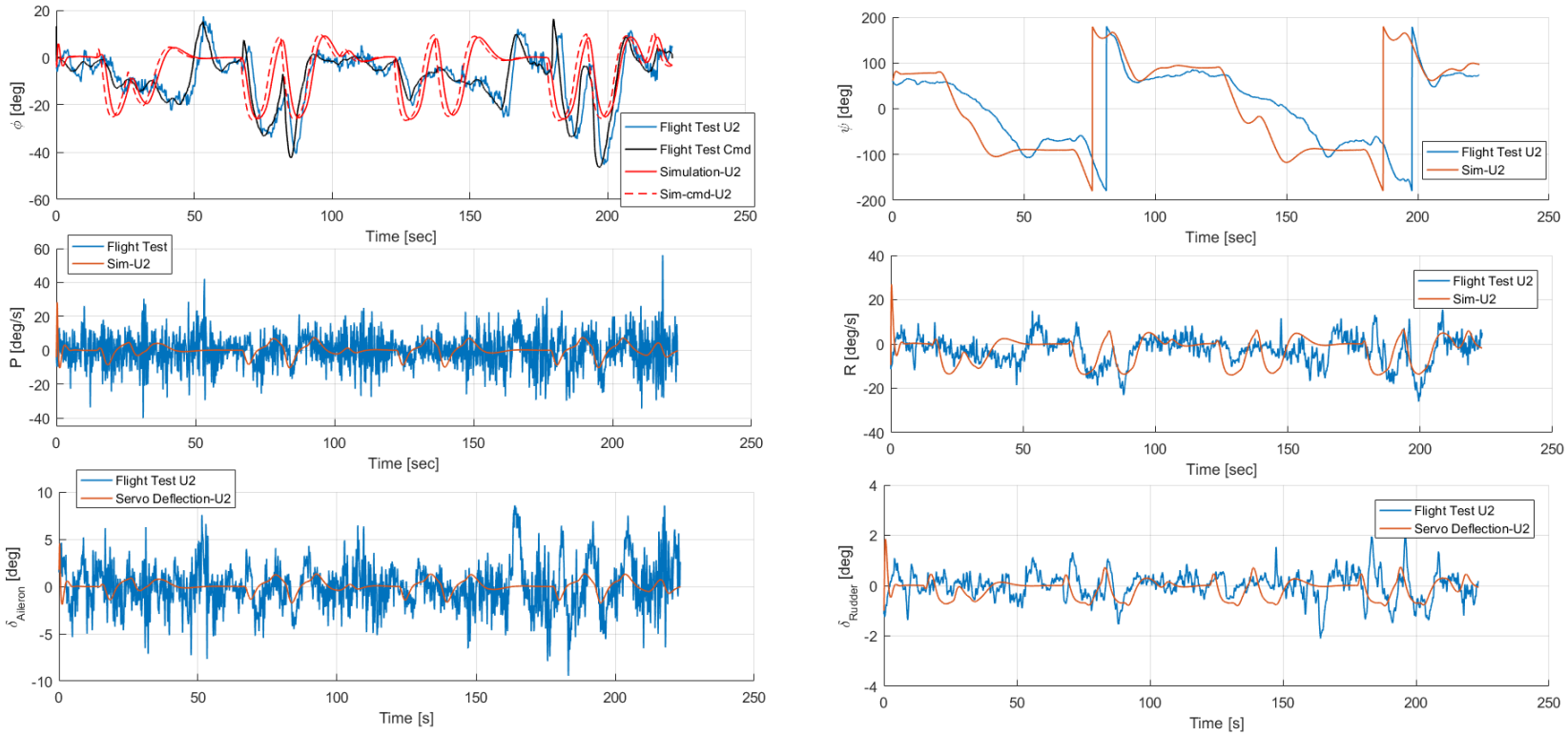


Figure 4.41: DG808 Collision Avoidance Flight test and simulation comparison, Lateral-directional States and Control Surfaces

4.5.2 Swarm flight: DG808 March 26th 2017

The purpose of this flight test is to validate the virtual formation algorithm with LQ guidance path planning for two DG-808. The virtual leader is planned by LQ guidance path planning algorithm and the virtual point formation algorithm provides the formation position for each agent. The guidance and control scheme is identical from the previous flight tests. The average wind speed was 4.6 mph from the west.

- 1st Attempt

  In this flight test, the swarm flight is tested to validate the aforementioned GNC in Chapter 2. The details of information are presented: the agents' trajectory (Fig. 4.44 and 4.48), the comparison of the position of the moving points (Fig. 4.45), the longitudinal states and controls for each aircraft (Fig. 4.46 and 4.49) and the lateral states and controls for each aircraft (Fig. 4.47 and 4.50).

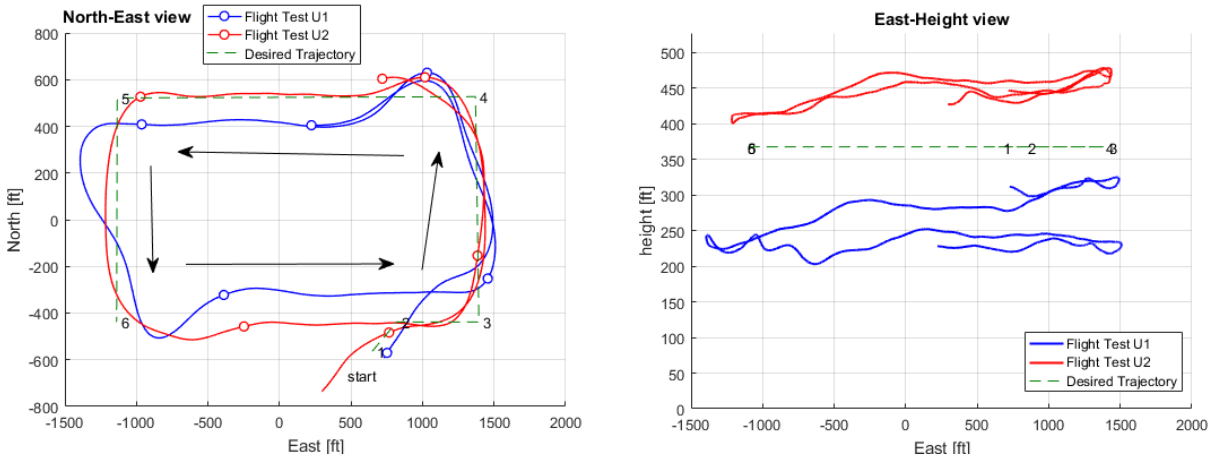


Figure 4.42: Swarm flight using two of DG-808, 1st Attempt

  – Navigation performance

    Figure. 4.42 shows the aircraft position in the north-east (left) and the east-height view (right). Two agents are engaged the autonomous mode the south east corner of the race track. The desired relative distances are presented in Table 4.10. Based on Fig. 4.45, aircraft 1 is assigned to the left side of the formation and aircraft 2 is assigned to the

Table 4.10: The desired relative distance from the virtual leader

Agent	North [ft]	East [ft]
1st agent	100	-50
2nd agent	-100	50

right side of the formation. In unstructured environment, it is observed that the aircraft is affected by the wind (the overshoot is occurred around northwest corner as 212 ft). As Figure 4.44 shows, the aircraft 1 which should follow a smaller race track pattern (the desired relative distance is -50 ft meaning that aircraft 1 should always stay inside of the defined race track) and the simulation result did not have any oscillation comparing to the flight test. In addition, Fig. 4.43 shows that the distance between two aircraft is average of 306 ft larger than the desired distance (223.6 ft). This results show how challenging the formation flight at the unstructured environment. In Figure 4.45, the moving point positions are compared and the overshoot between the simulation and flight test is 3.4% which meets the defined criteria in Table 4.9. The reason why the discrepancy exist in the moving point generation between the simulation and flight test is that the agents' inertial position (the morphing potential algorithm uses the agents' position and velocity) does not match between them due to the uncertainty of the dynamic model and the lack of external disturbances in the simulation environment.

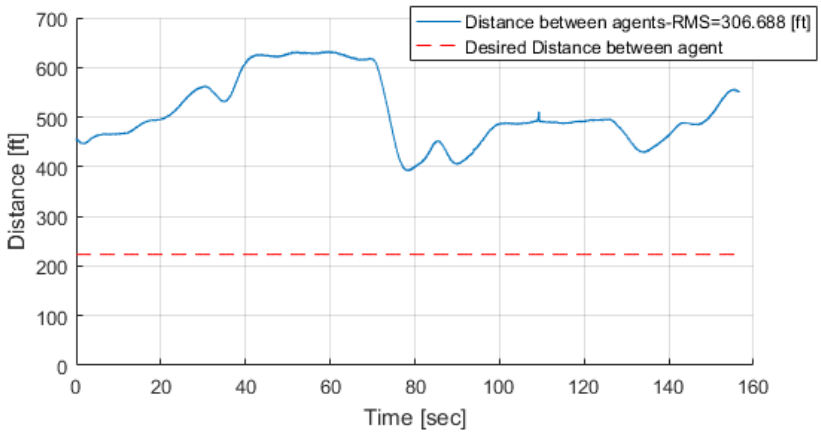


Figure 4.43: Distance between agents in the swarm flight using two of DG-808, 1st Attempt

– Guidance and Control performance

Two aircraft start the autonomous flight at the different altitude. The aircraft 1 has significant oscillations in the pitch angle comparing to the aircraft 2. The possible reasons are the gust at the different altitude and the tighter and more frequent turning maneuvers required for aircraft 1. Turning maneuvers cause the descending altitude since the lift is reduced due to the reduced surface area to generate the lift. Specifically, the aircraft is tilted and the wing area becomes the projection of itself to the horizon by the roll angle. This phenomena keep the aircraft in descent and the controller tries to correct slowly. However, the aircraft should turn again before the controller sends the aircraft to the desired altitude, shown in Fig. 4.42 (right). Comparing to aircraft 1 (maximum 22 degree magnitude), aircraft 2 (maximum 10 degree) has less oscillations in the pitch angle tracking (Fig. 4.46 and 4.49). The roll angle tracking for both aircraft is very close to the defined range (maximum 3 and 4 degree error (3 degree is the threshold), respectively) though the little oscillations are induced by the wind (Fig. 4.47 and 4.50).

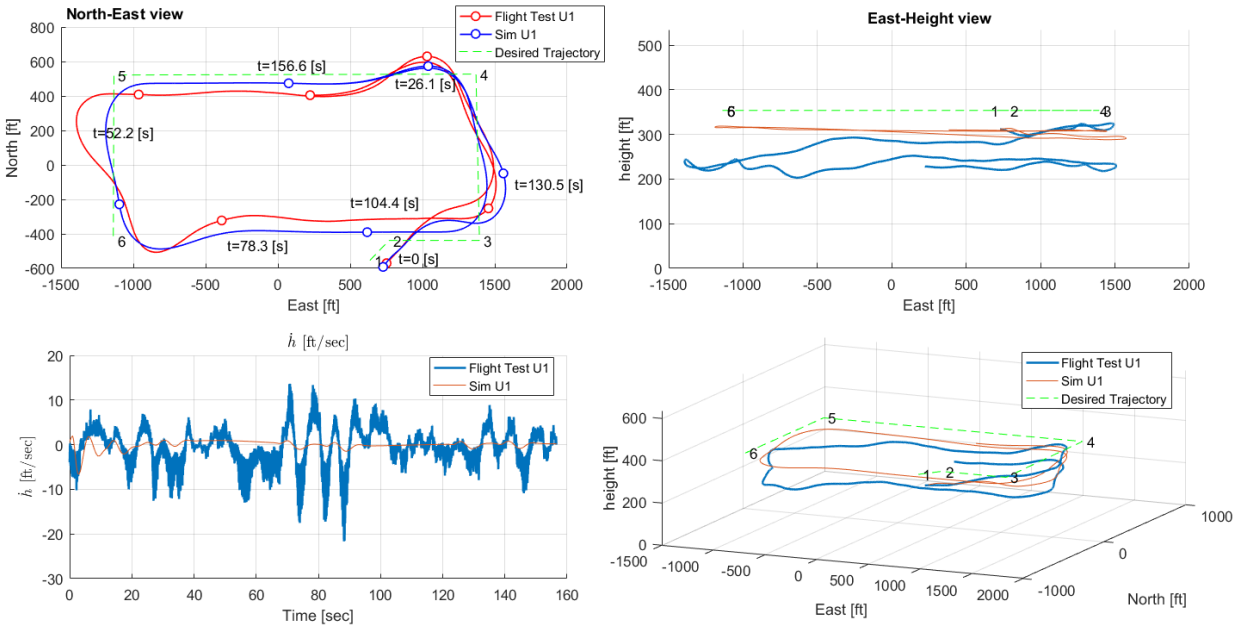


Figure 4.44: DG808 Swarm Flight test and simulation comparison, Trajectory and rate of altitude changes, 1st attempt - The 1st agent

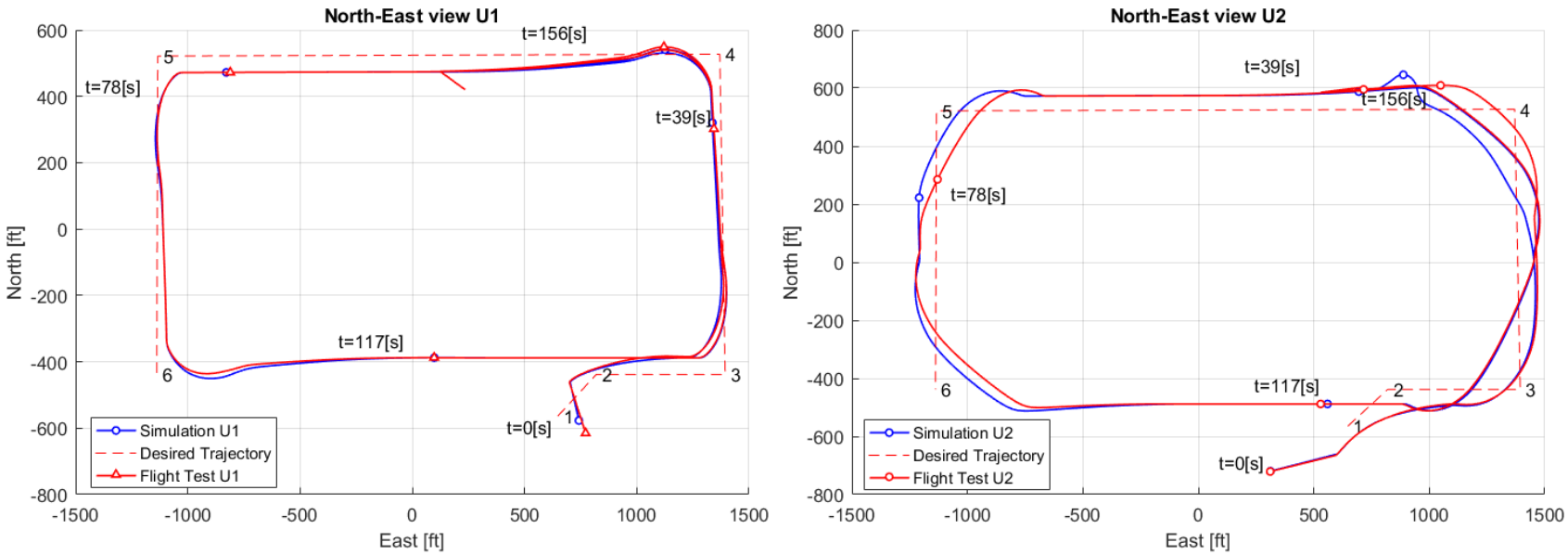


Figure 4.45: DG808 Swarm Flight test and simulation comparison, Moving point comparison, 1st attempt

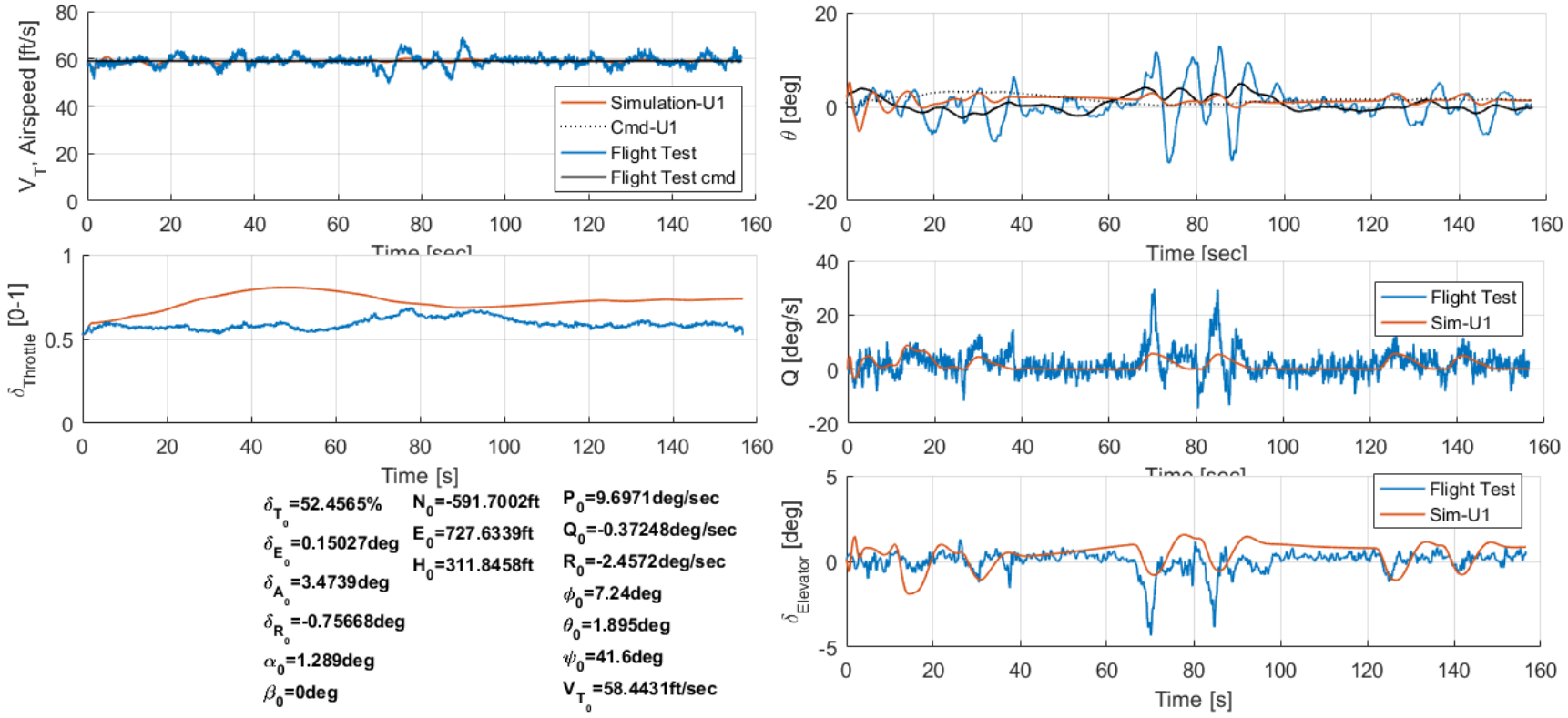


Figure 4.46: DG808 Swarm Flight test and simulation comparison, Longitudinal States and Control Surfaces, 1st attempt - The 1st agent

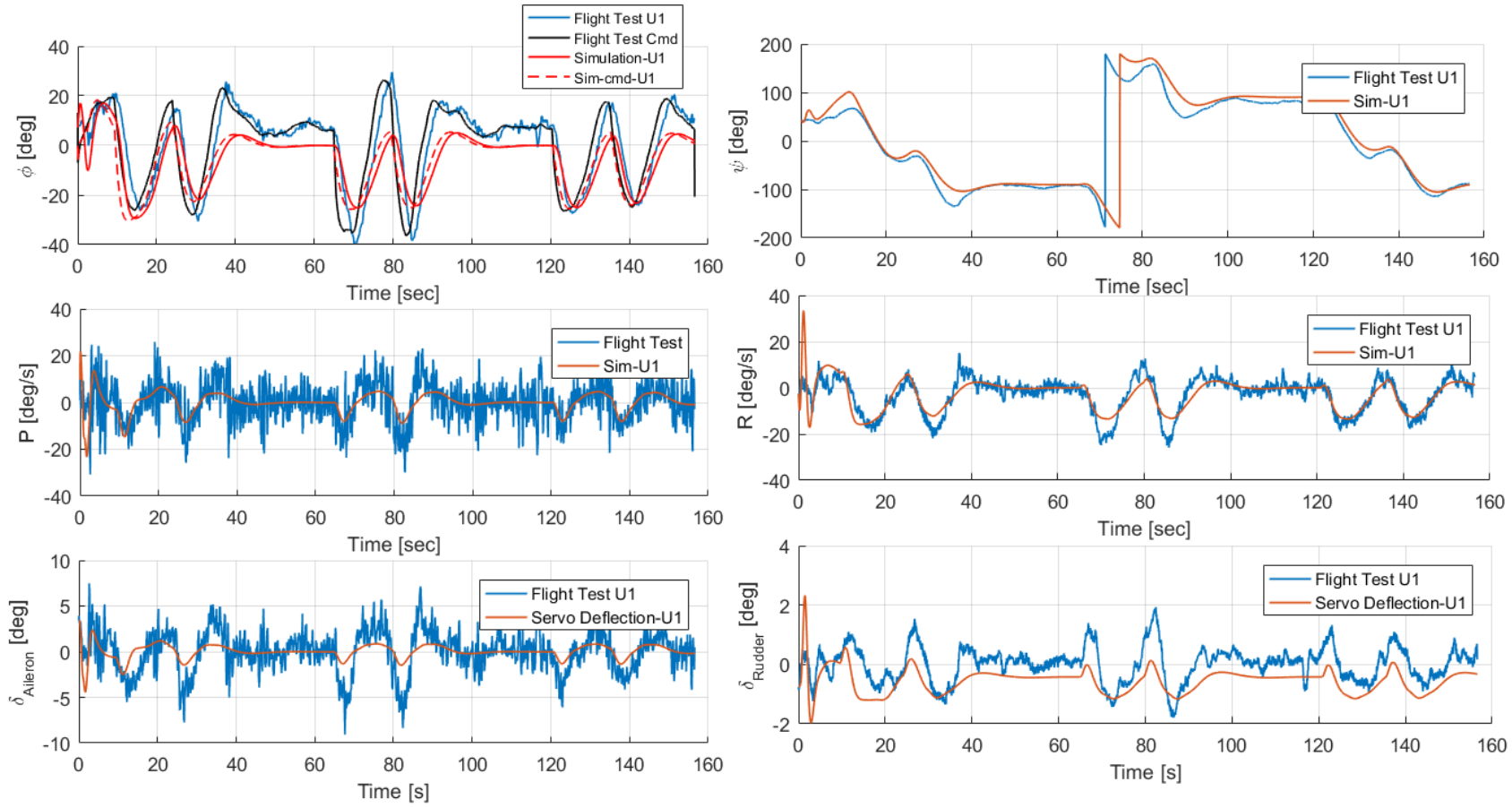


Figure 4.47: DG808 Swarm Flight test and simulation comparison, Lateral-directional States and Control Surfaces, 1st attempt - The 1st agent

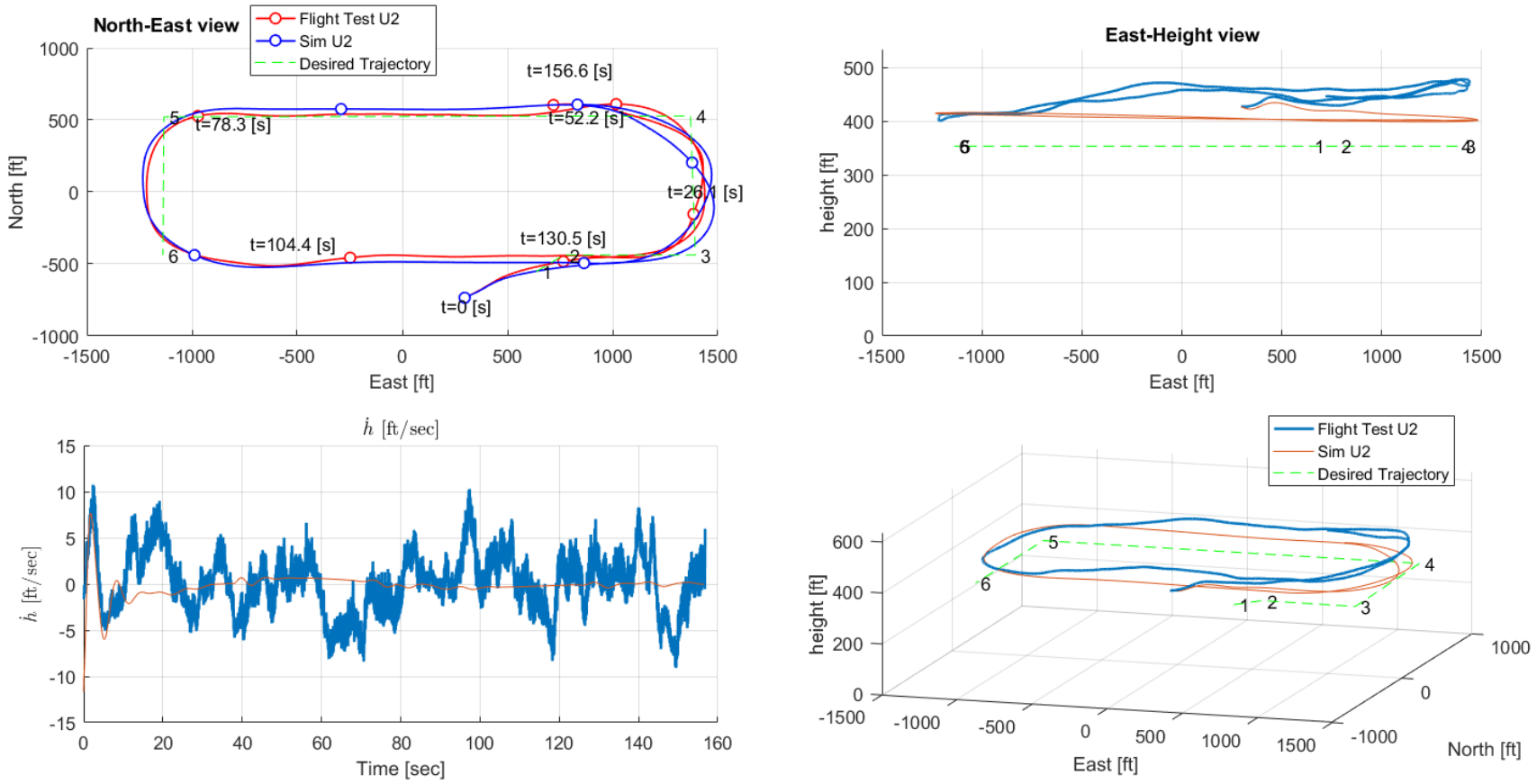


Figure 4.48: DG808 Collision Avoidance Flight test and simulation comparison, Trajectory and rate of altitude changes, 1st attempt - The 2nd agent

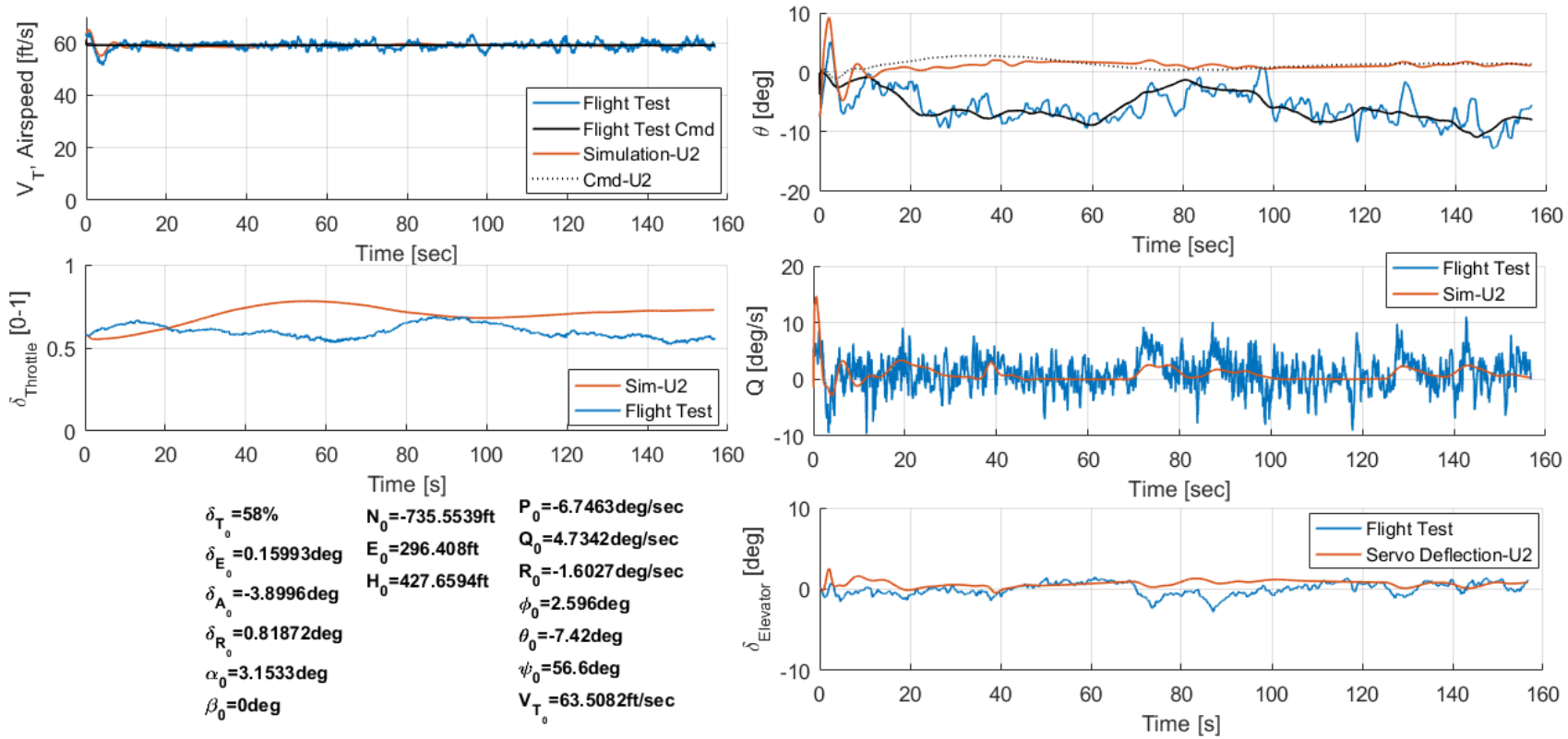


Figure 4.49: DG808 Swarm Flight test and simulation comparison, Longitudinal States and Control Surfaces, 1st attempt - The 2nd agent

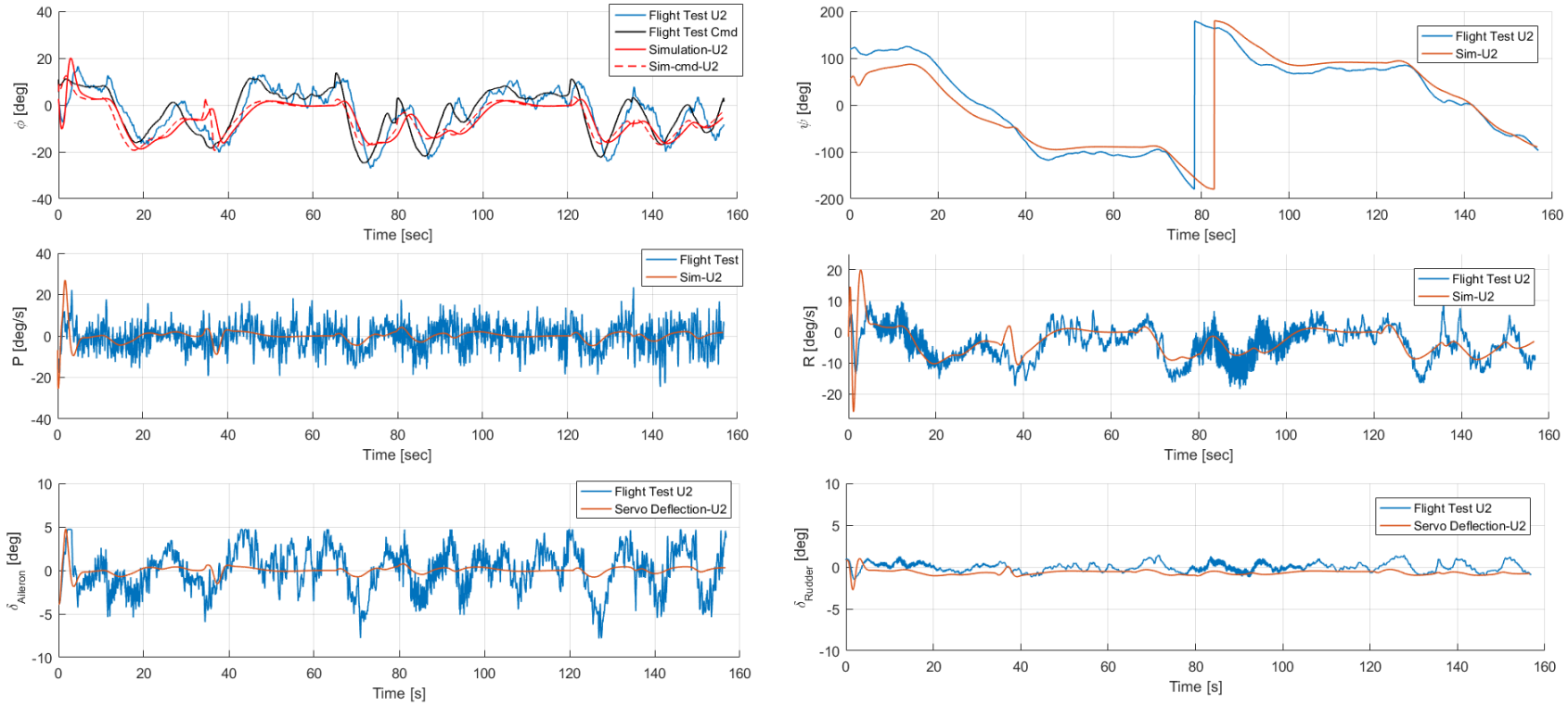


Figure 4.50: DG808 Swarm Flight test and simulation comparison, Lateral-directional States and Control Surfaces, 1st attempt - The 2nd agent

- 2nd Attempt

  In this swarm flight test, more challenging initial conditions are tested to validate the robustness of the random initial conditions. The 1st attempt of the swarm flight has relatively similar heading angles (41.6 and 56.6 degree for aircraft 1 and 2, respectively) when the autopilot is engaged. In this flight test, the heading angles of two aircraft (ψ_{U1} and ψ_{U2}) are oriented differently: -80.2 and 54.8 degrees, respectively. The details of flight test graphs are presented: the aircraft position (Figure 4.53 and 4.57), the moving point position comparison (Fig. 4.54), the longitudinal states and control for both aircraft (Fig. 4.55 and 4.58) and the lateral-directional states and control (Fig. 4.56 and 4.59).

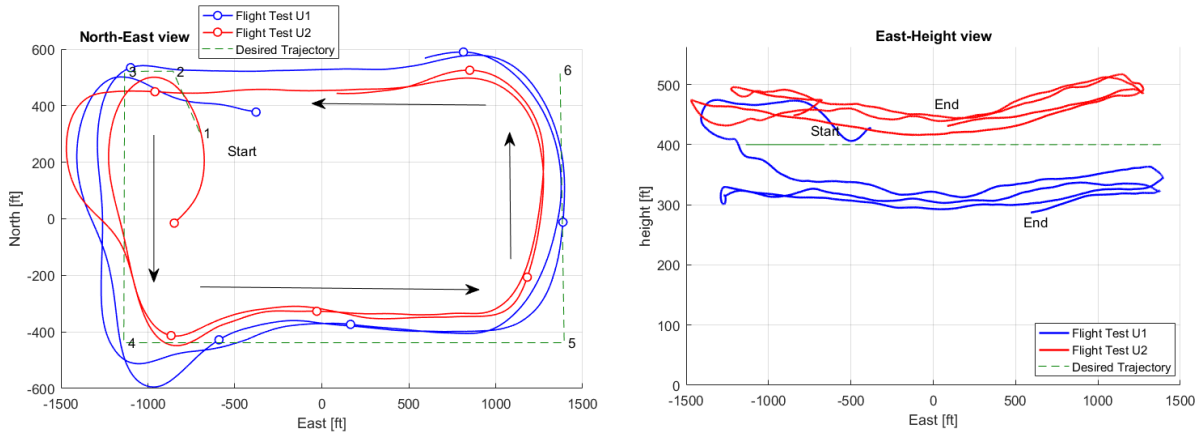


Figure 4.51: Swarm flight using two of DG-808, 2nd Attempt

– Navigation performance

  The moving point is planned successfully by proposed navigation algorithms to keep the formation coherency. The following table shows the desired relative distance from the virtual leader.

Table 4.11: The desired relative distances from the virtual leader

Agent	North [ft]	East [ft]
1st agent	100	50
2nd agent	-100	-50

  The waypoint modification and index decision algorithm finds the intersection by using the averaged position and velocity vectors from two agents. Then, the virtual leader

follows the race track by LQ guidance path planning. As Fig. 4.51 shows, aircraft 1 stays right of aircraft 2. Again, the presence of the wind in the unstructured environment caused the shifting in the moving point tracking towards the west by 117 and 130 ft for aircraft 1 and 2, respectively, shown in Fig. 4.53 and 4.57. Figure 4.52 presents the distance between two agents and the RMS is 129 ft, which is closer than the 1st attempt (306 ft). The different initial conditions, external disturbances and the confined area cause difficulty in keeping a formation throughout the flight.

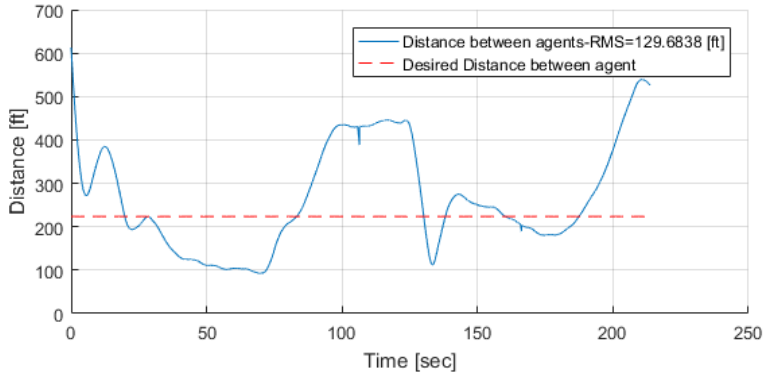


Figure 4.52: Distance between agents in the swarm flight using two of DG-808, 2nd Attempt

– Guidance and Control performance

Guidance and control performance is similar to the previous flight test. The pitch angle tracking is not satisfactory since the maximum error is 4 and 6 degrees for aircraft 1 and 2, respectively (the error assessment is considered after the aircraft follows first segment to exclude the impact of initial condition). In addition, the roll angle is mostly well tracked meaning within 3 degrees error by the defined criteria in Table 4.9 except one or two spots (exceed 3 degrees error, maximum 13 and 6 degree error for aircraft 1 and 2, respectively) due to the external disturbances. The speed tracking does not meet the criteria since the maximum overshoot is 15% and 13.3% for aircraft 1 and 2, respectively, which is larger than 10%. The different initial condition and the wind field causes the tracking error (the average 123.5 ft drifting is occurred) as LQR is very prone to the un-modeled dynamics.

Two attempts of the swarm flight tests show the difficulties of keeping the formation shape in the unstructured environment. The discrepancies of the agents' position and moving positions are observed between the simulation and flight test due to the uncertainty and error of the dynamic modeling and the presence of the external disturbances. Next, the flight test is conducted by using Skyhunter in order to validate the proposed GNC for the different platform.

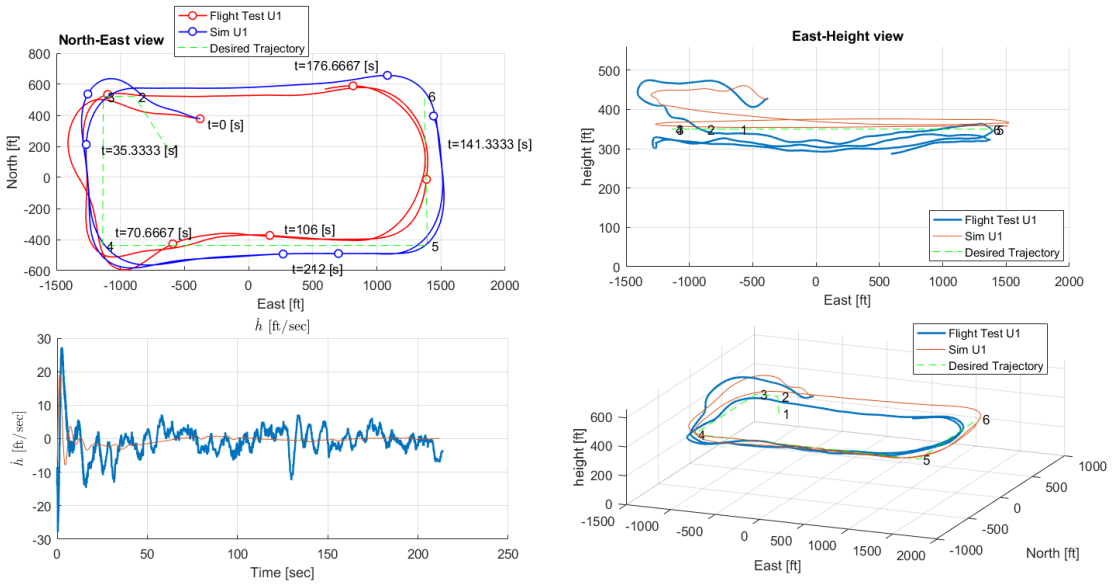


Figure 4.53: DG808 Swarm Flight test and simulation comparison, Trajectory and rate of altitude changes, 2nd attempt - The 1st agent

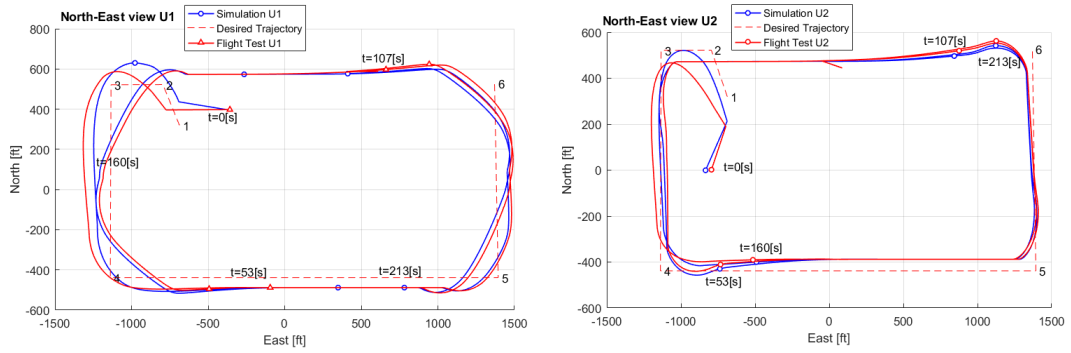


Figure 4.54: DG808 Swarm Flight test and simulation comparison, Moving Point Position Comparison, 2nd attempt

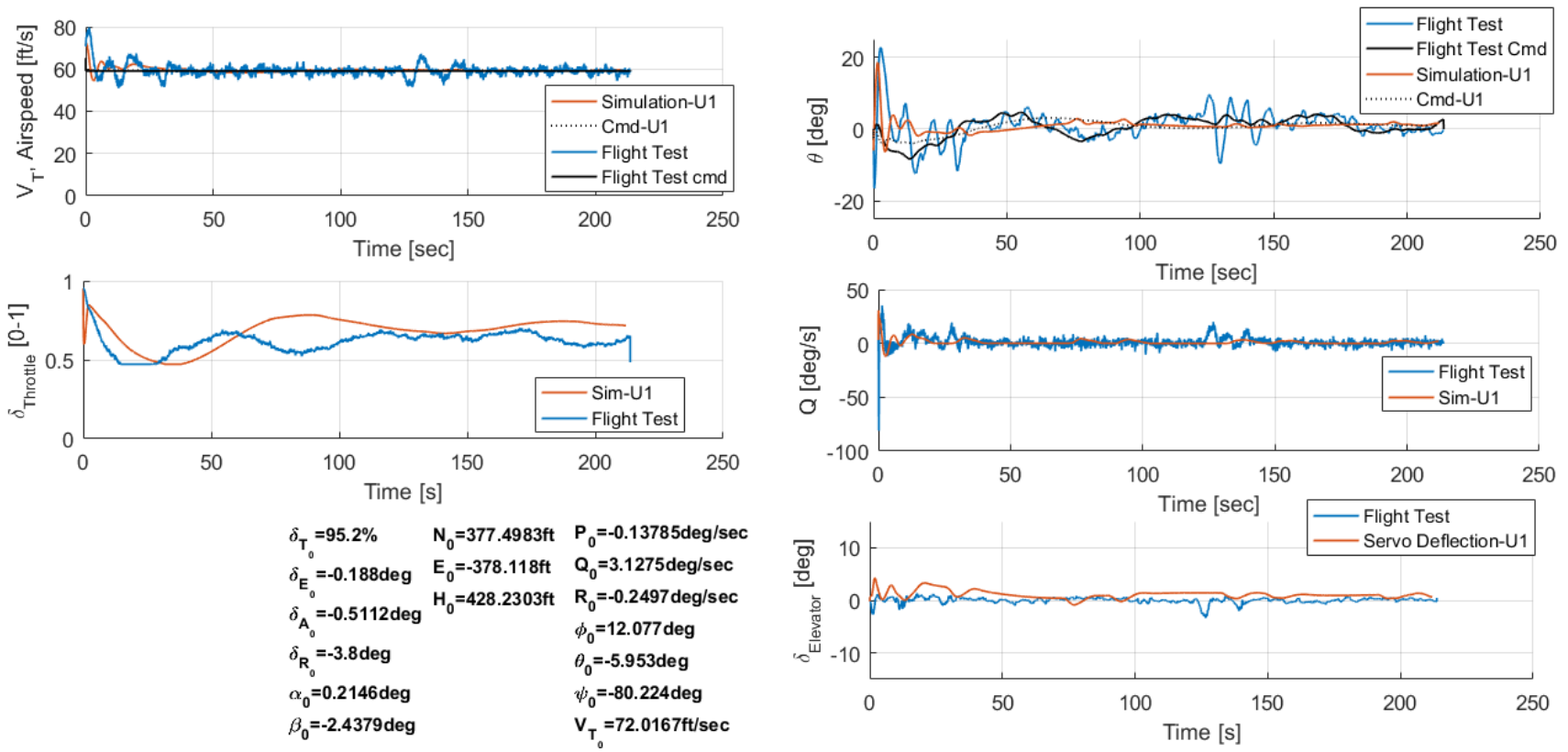


Figure 4.55: DG808 Swarm Flight test and simulation comparison, Longitudinal States and Control Surfaces, 2nd attempt - The 1st agent

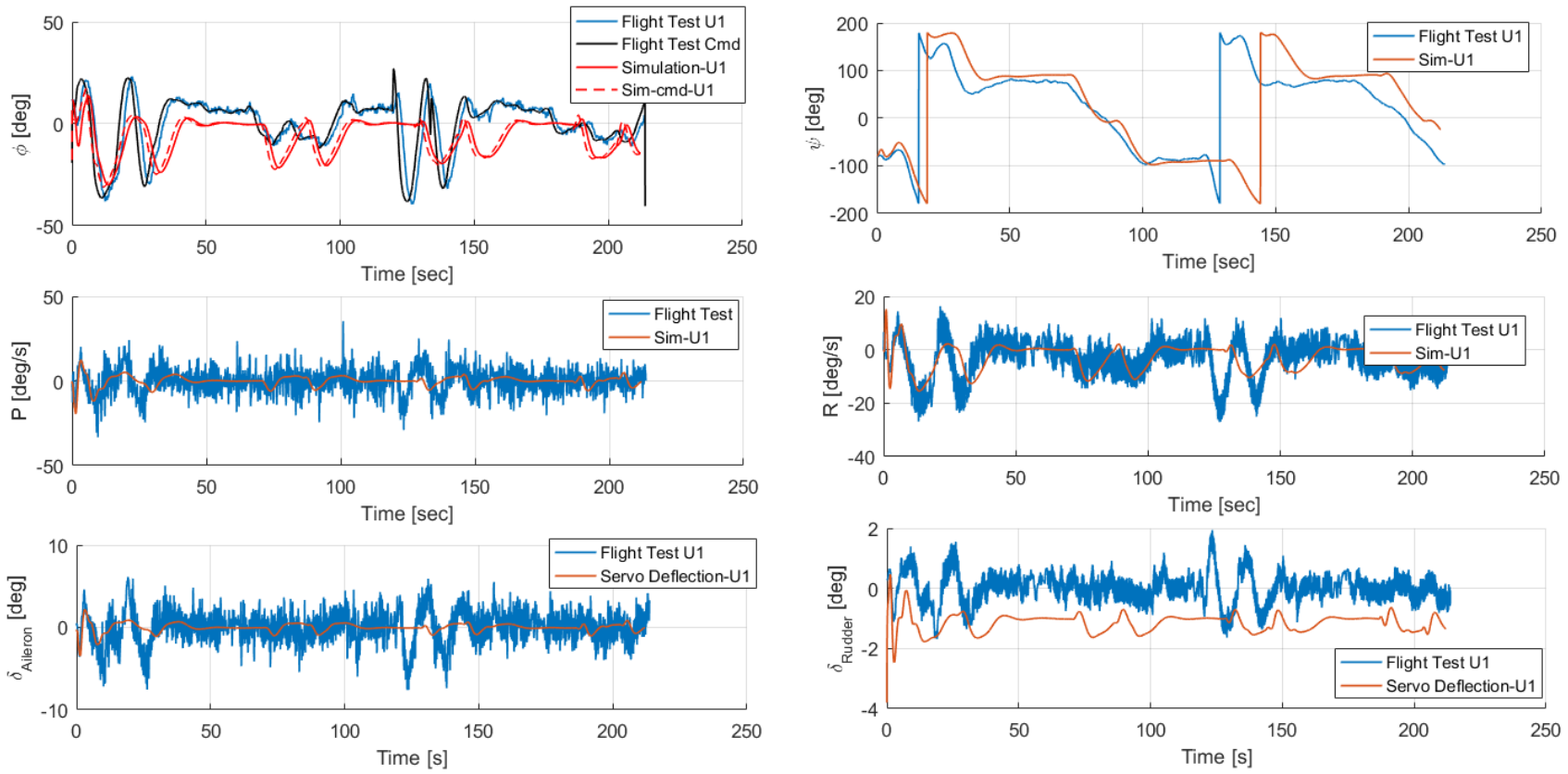


Figure 4.56: DG808 Swarm Flight test and simulation comparison, Lateral-directional States and Control Surfaces, 2nd attempt - The 1st agent

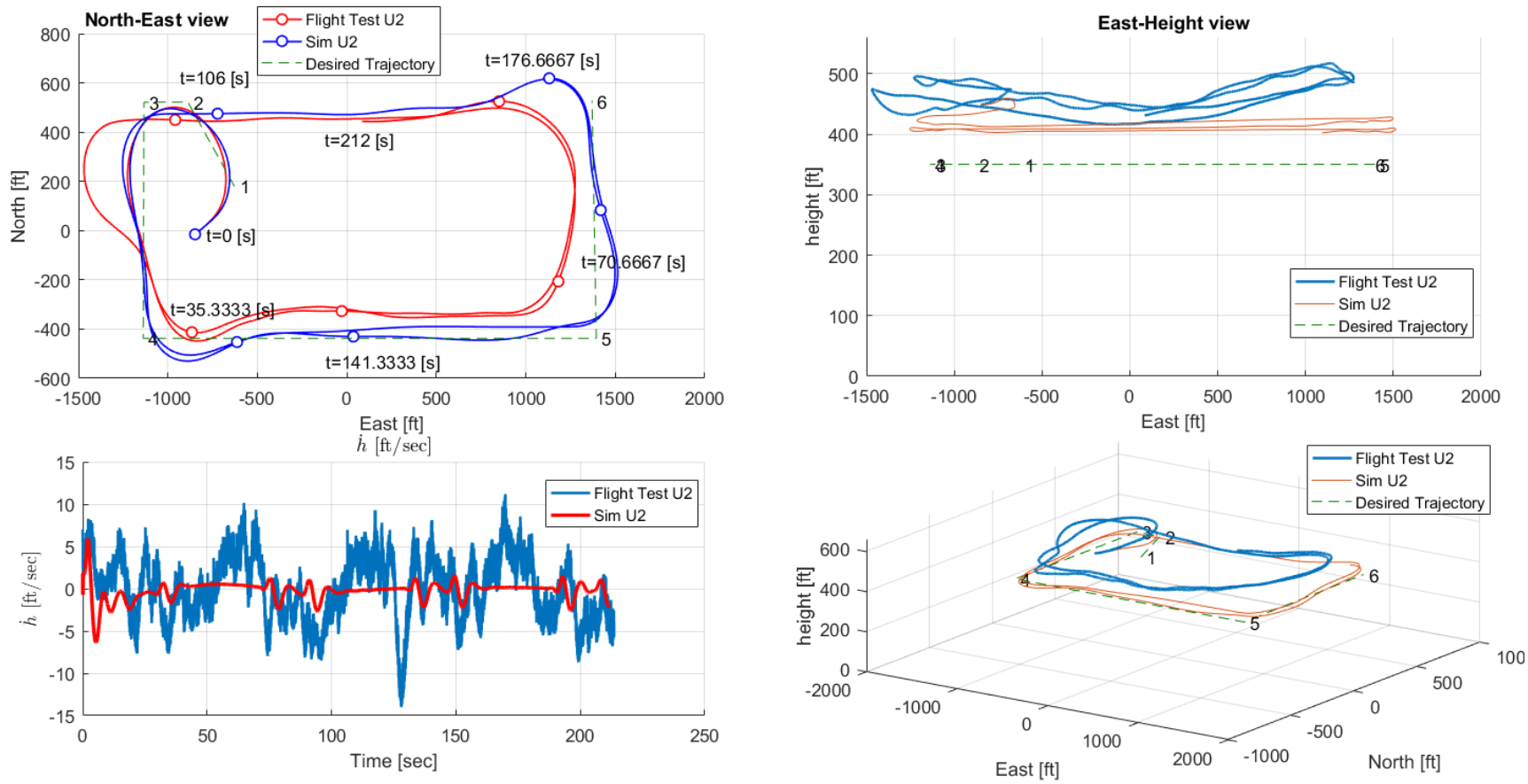


Figure 4.57: DG808 Swarm Flight test and simulation comparison, Trajectory and rate of altitude changes, 2nd attempt - The 2nd agent

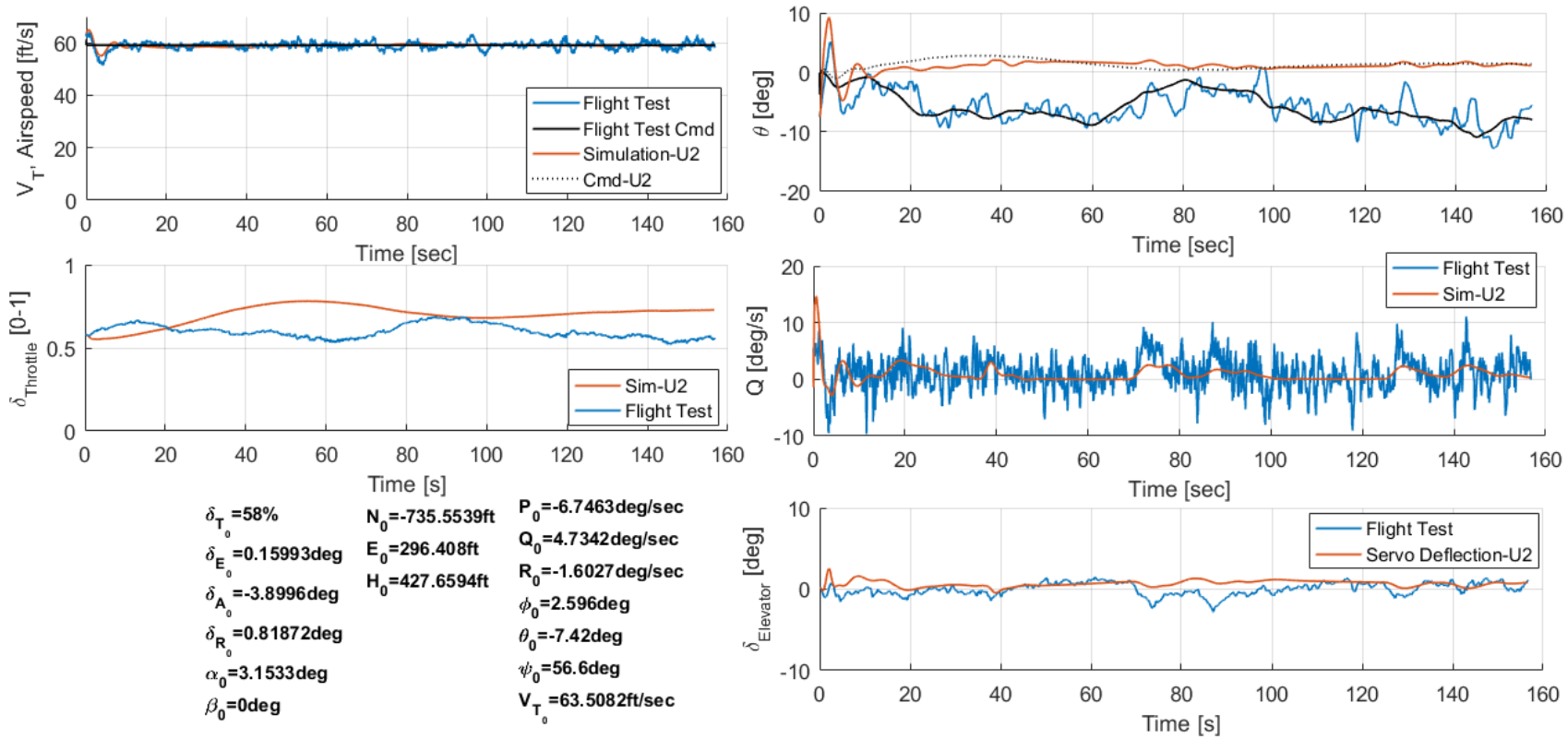


Figure 4.58: DG808 Swarm Flight test and simulation comparison, Longitudinal States and Control Surfaces, 2nd attempt - The 2nd agent

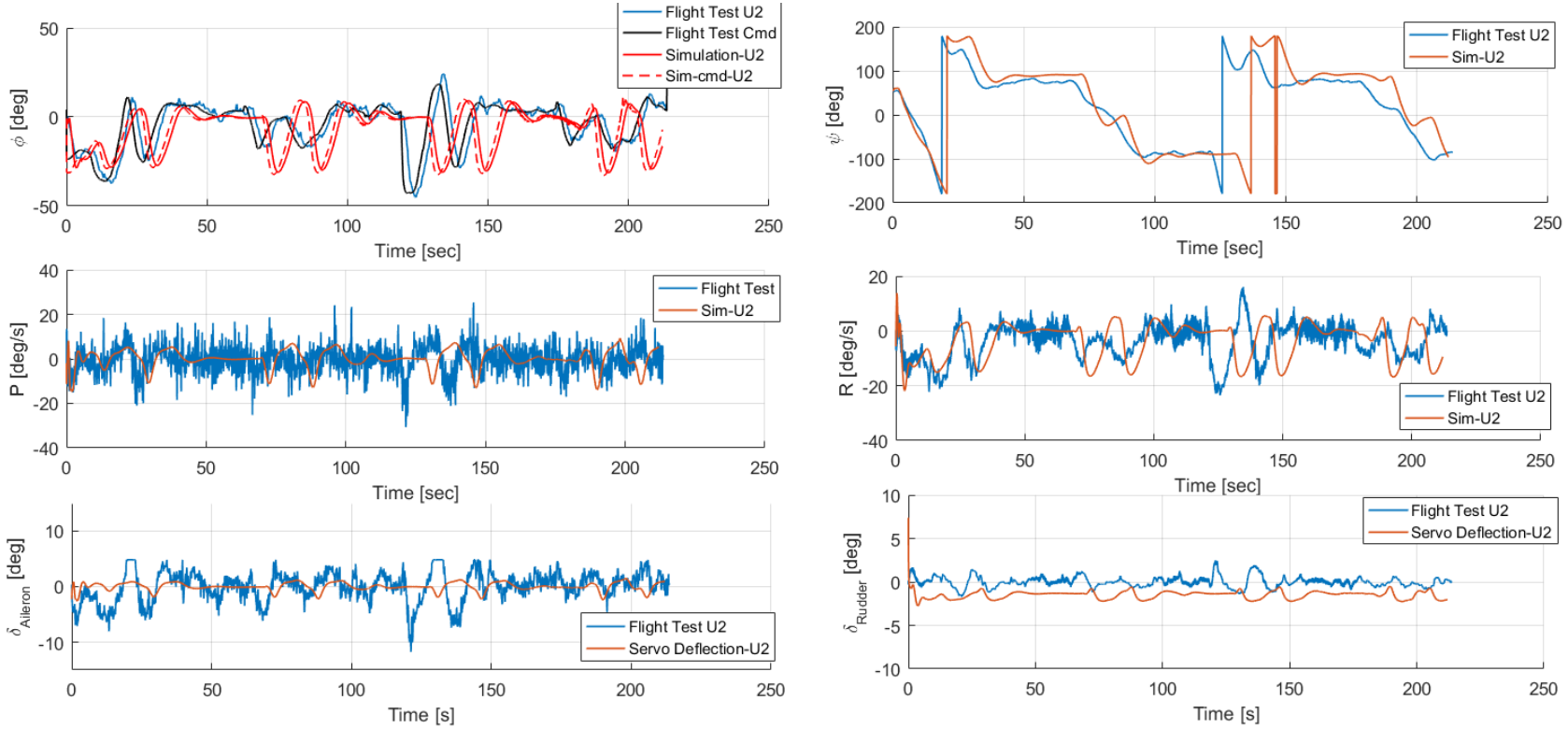


Figure 4.59: DG808 Swarm Flight test and simulation comparison, Lateral-directional States and Control Surfaces, 2nd attempt - The 2nd agent

4.5.3 Solo flight: Skyhunter, August 14th 2017

This flight test is to validate the solo flight by using Skyhunter with the aforementioned GNC algorithms. Since the dynamic models of DG-808 and Skyhunter are different, the guidance parameters and controller gains are tuned for Skyhunter. The following figures describe the result of this flight test: the position of the aircraft (Figure 4.60), the guidance outputs with lateral-directional and longitudinal error angles (Figure 4.61) and the longitudinal, and lateral-directional states and controls (Figure 4.62 and 4.63, respectively).

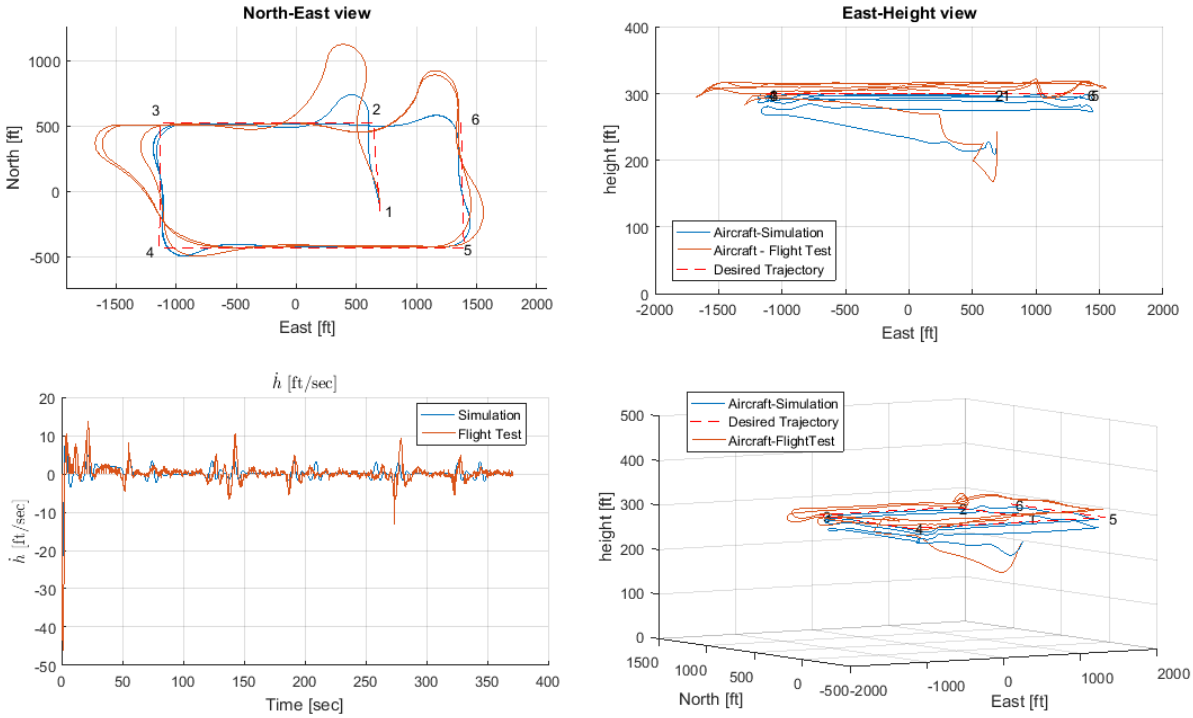


Figure 4.60: Skyhunter Flight test and simulation comparison, Trajectory and rate of altitude changes

- Navigation performance

  This flight test is a good example to show the impact on the tracking due to the external disturbance. Fig. 4.60 (top left) shows that the lateral-directional tracking has very large overshoot (594.5 ft error, 113% overshoot $\gg$ 10% criteria). This behavior is caused due to the speed difference between the moving point and the aircraft. This can be explained

by two performance parameters: the aircraft speed and the moving point traveling speed. When the aircraft proceeds from the waypoint 1 to 2, the aircraft flies faster than the moving point due to the wind field. Once the aircraft arrives at waypoint 2, the moving point is still lying on the line segment made by the waypoint 1 and 2. This is the reason why the aircraft cross the next line segment made by the waypoint 2 and 3. Fig. 4.61 (top right) shows that the aircraft speed command is set to the initial condition which is 48 ft/sec. When the autopilot is engaged, the speed command and the moving point speed is defined by the initial aircraft speed. Supposedly, they should travel along each other but the aircraft can fly faster or slower depending on the wind direction. This is also shown in Fig. 4.60 (top left). The two corners of the race track at the waypoint 4 and 5 do not have large overshoot unlike the waypoint 2, 3, and 6. On the other hand, the altitude tracking is not acceptable for both flight test and simulation since the overshoot is 43.7 and 29.33%, respectively (Figure 4.60 (top right)).

- Guidance and Control performance

  The roll angle command has maximum 19 degree larger magnitude than the simulation due to the external disturbances as it is discussed above. On the other hand, the roll angle tracking is acceptable since the maximum roll angle error is 3.4 degree (≈ 3). The speed and pitch angle commands are tracked well (the maximum overshoot is 8.9% and 2 degrees, respectively, see Table 4.9) when the aircraft tracks the straight legs (longer lines of the race track made by waypoint 6 & 3 and 4 & 5). Regarding the discrepancy of the throttle behavior (16% less usage during the flight test), the modeling error exists in the thrust model or the drag model as it is mentioned in the previous flight test (DG-808 solo flight test). Even though the pitch rate initial condition was around -70 deg/sec, the LQR controller could overcome with the fast pitch rate and returned zero after 3.8 seconds.

Throughout these flight tests, the proposed GNC algorithms are applied for two different platforms: DG-808 and Skyhunter. The flight test data shows the impact of the external disturbances comparing to the simulation (the aforementioned drifting in the tracking). De-

pending on the initial conditions, the multi-agent formation distance is affected. The 1st attempt swarm flight has 306 ft RMS of the distance between agents but the 2nd attempt swarm flight has 129 ft RMS in the distance between agents which is 58% lower than the 1st attempt swarm flight.

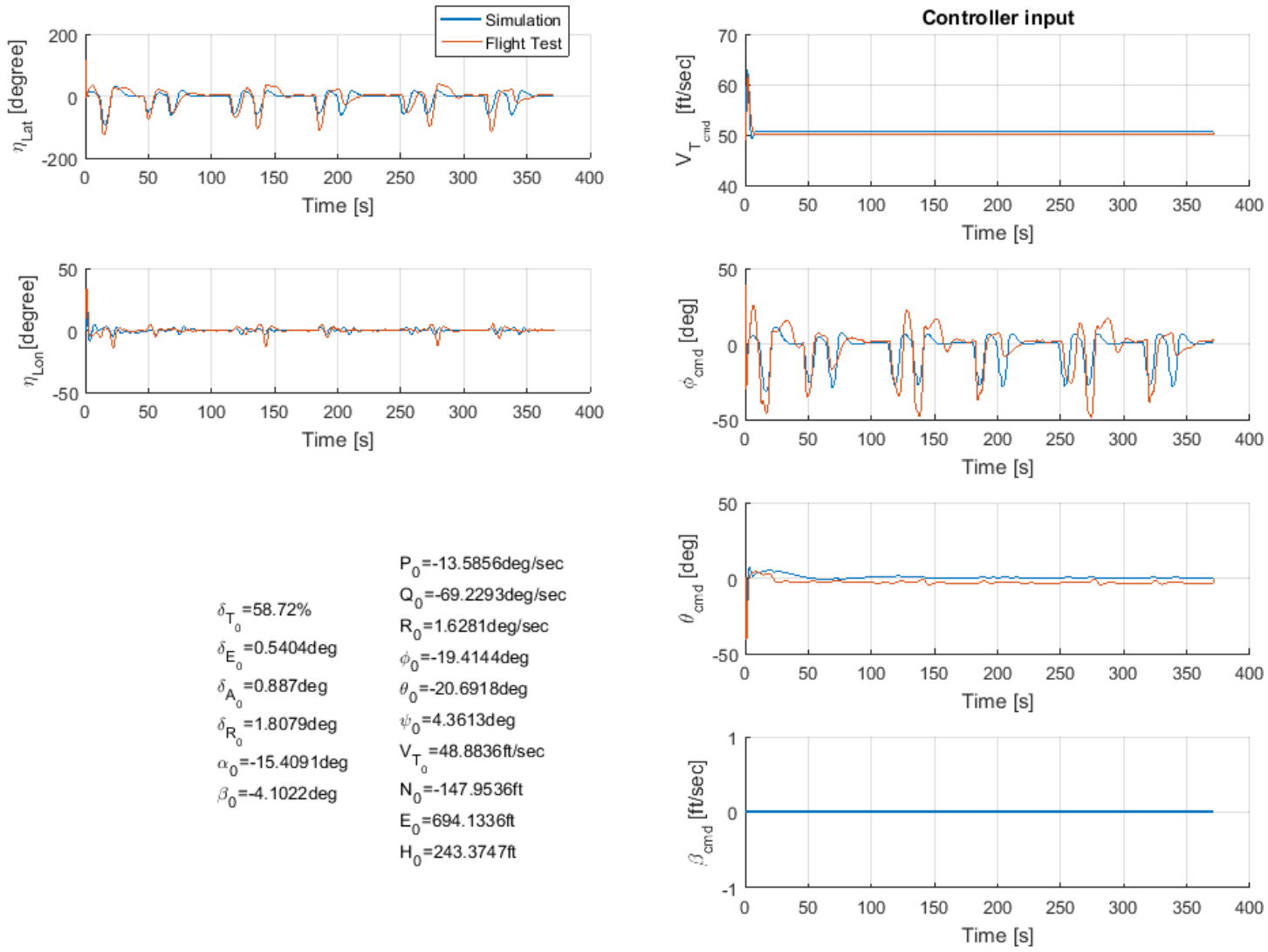


Figure 4.61: Skyhunter Flight test and simulation comparison, Longitudinal States and Control Surfaces

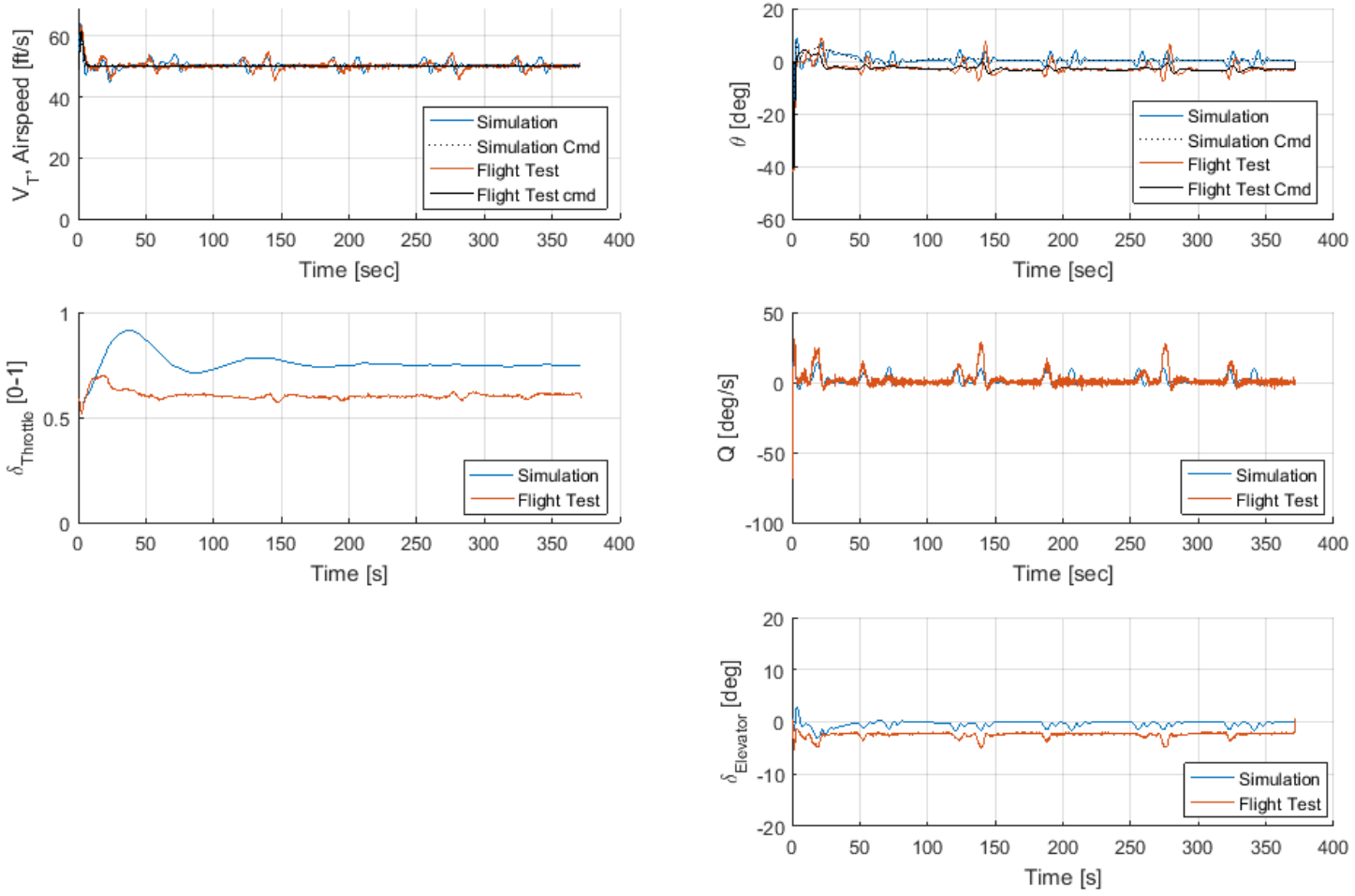


Figure 4.62: Skyhunter Flight test and simulation comparison, Longitudinal States and Control Surfaces

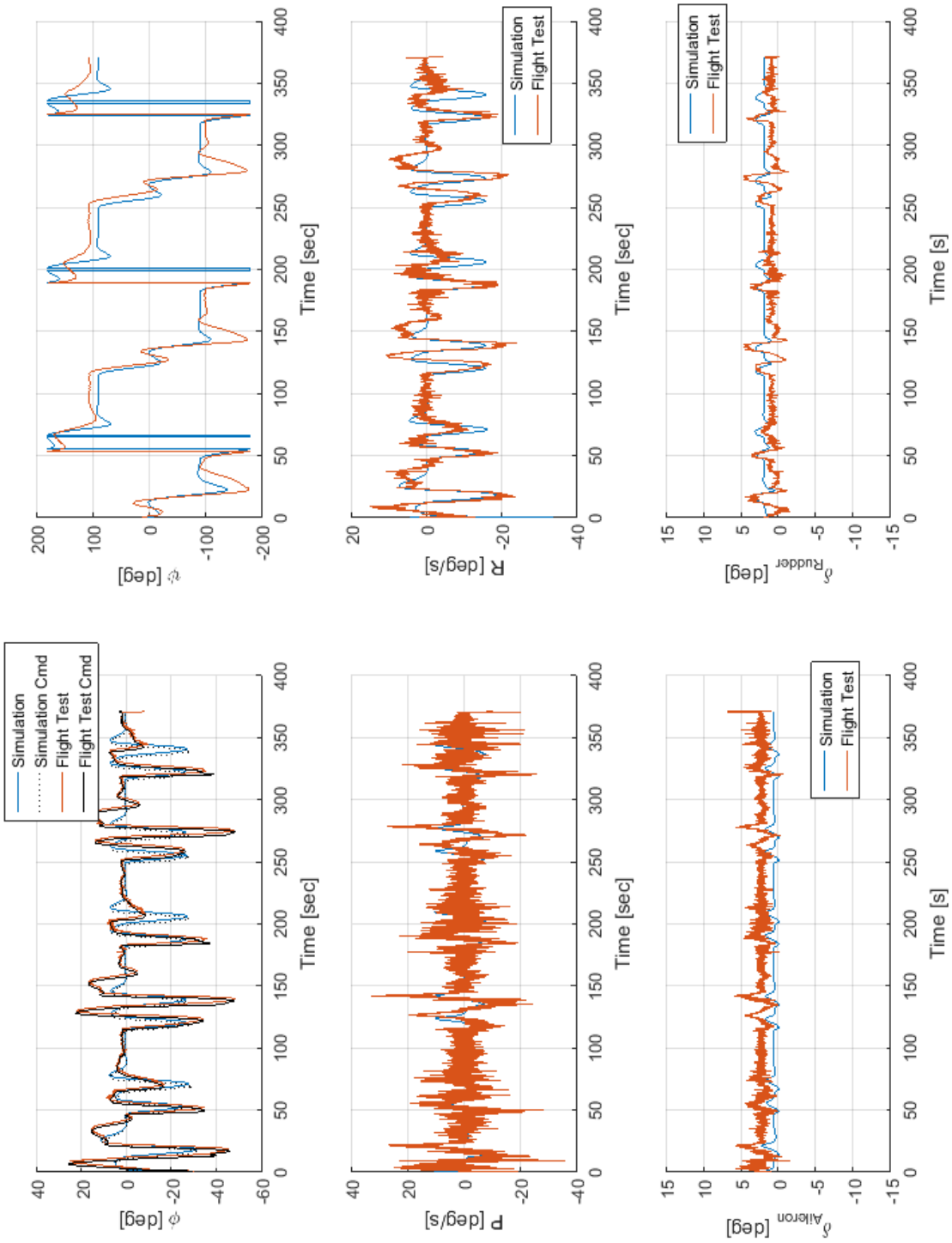


Figure 4.63: Skyhunter Flight test and simulation comparison, Lateral-directional States and Control Surfaces

4.5.4 Solo flight: Skyhunter, April 26th 2018

This flight test is to validate the improvement of the tracking by the adaptive $d_{dR_{Lat}}$ mentioned in Chapter 2. In addition, the longitudinal guidance parameters are adjusted to improve the altitude tracking as well to meet the defined criteria in Table 4.9 and the impact of varying these parameters has been discussed in Chapter 2.1.2. Mainly, $d_{dR_{Lat}}$ is reduced from 2500 to 700 ft. Figure 4.64 shows the comparison of the position of the aircraft between the flight test and simulation. The length of $d_{dR_{Lat}}$ varies from 320 to 420 ft. These parameters are adjusted in the real time during the flight test with observing the aircraft behavior. Figure 4.66 and 4.67 shows the longitudinal and lateral-direction states and control,

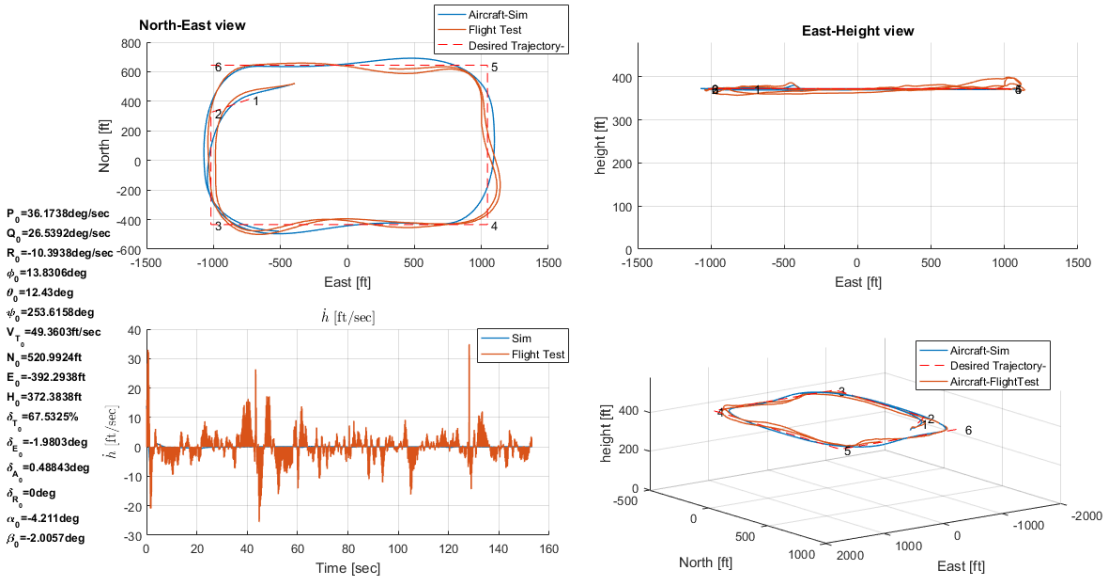


Figure 4.64: Skyhunter solo flight test and simulation comparison, Position

respectively. In order to quantify the tracking performance, RMS of the tracking error is calculated when the roll angle is small for both flight tests in August, 2017 and April, 2018, see Figure 4.65. The red portions are the selected data for calculating RMS of tracking error. The lateral-directional and longitudinal tracking is improved 99.7% and 99.4%, respectively. This is a significant improvement and essential for the swarm flight to keep the formation shape. Regarding the controller performance, the pitch angle tracking is loose (maximum

8.6 degrees between 70 and 85 seconds (the time when small roll angle)) while the speed command is tracked very well (the maximum overshoot 6.25% $\leq$ 10% criteria in Table 4.9). Despite the initial condition of the states (the roll rate is 36 deg/s and the pitch rate is 26.5 deg/s, there are out of bound from the defined assessment criteria), the controller is able to stabilize the aircraft and operate the flight for 150 seconds. In addition, this flight test result surpasses the existing work (the lateral tracking error is 185.9 ft and the altitude overshoot is 13.3 ft from Ref. [2]).

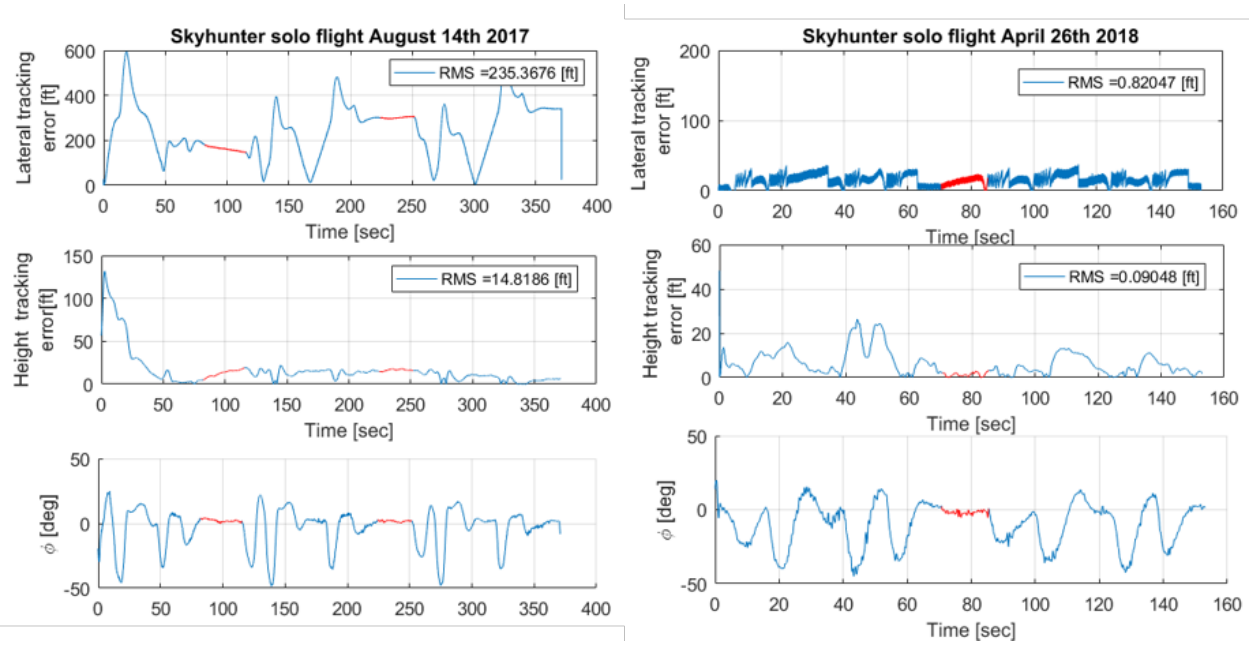


Figure 4.65: Tracking RMS comparison between August, 2017 and April, 2018

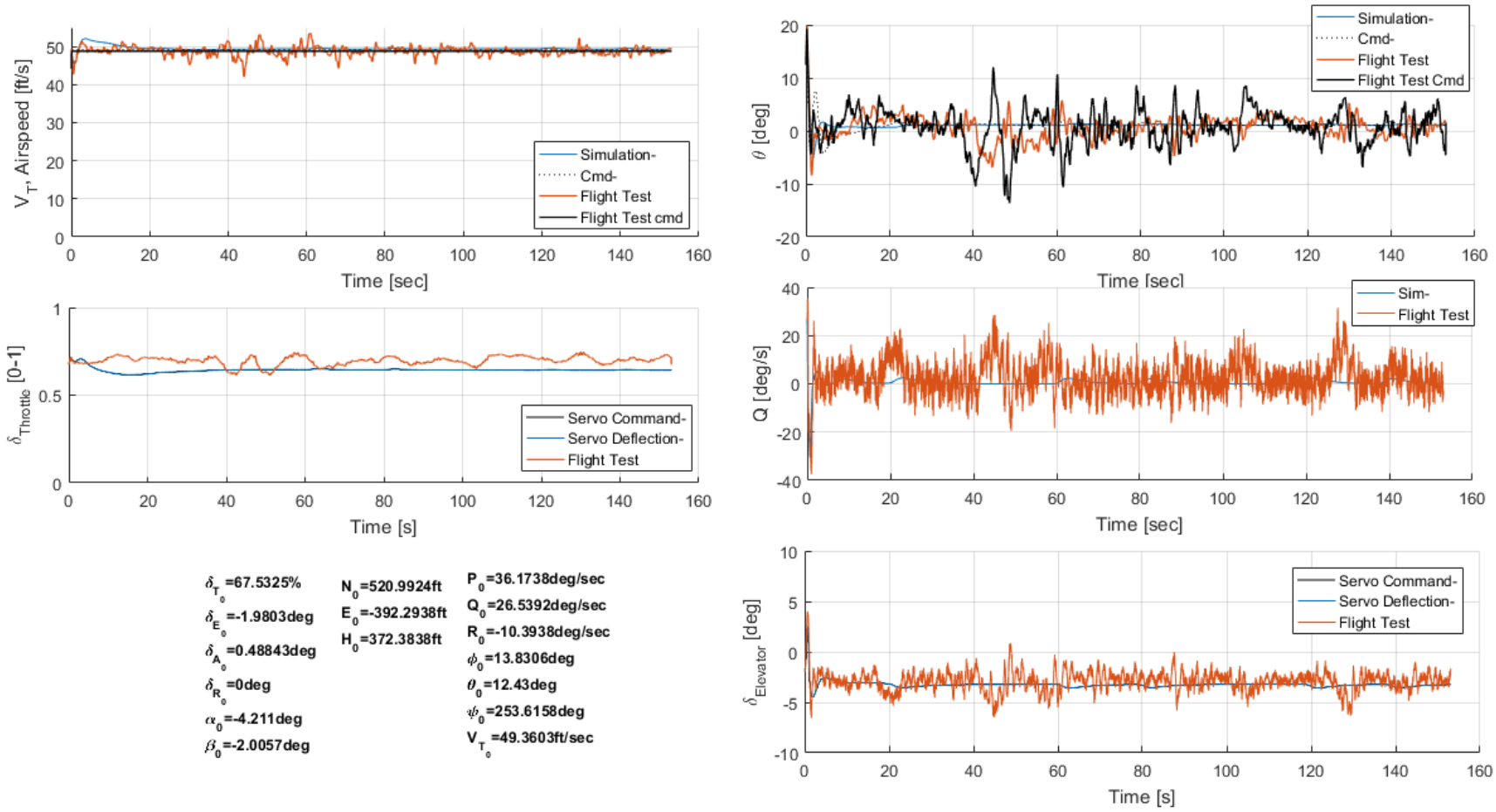


Figure 4.66: Skyhunter solo flight test and simulation comparison, Longituidnal states and control

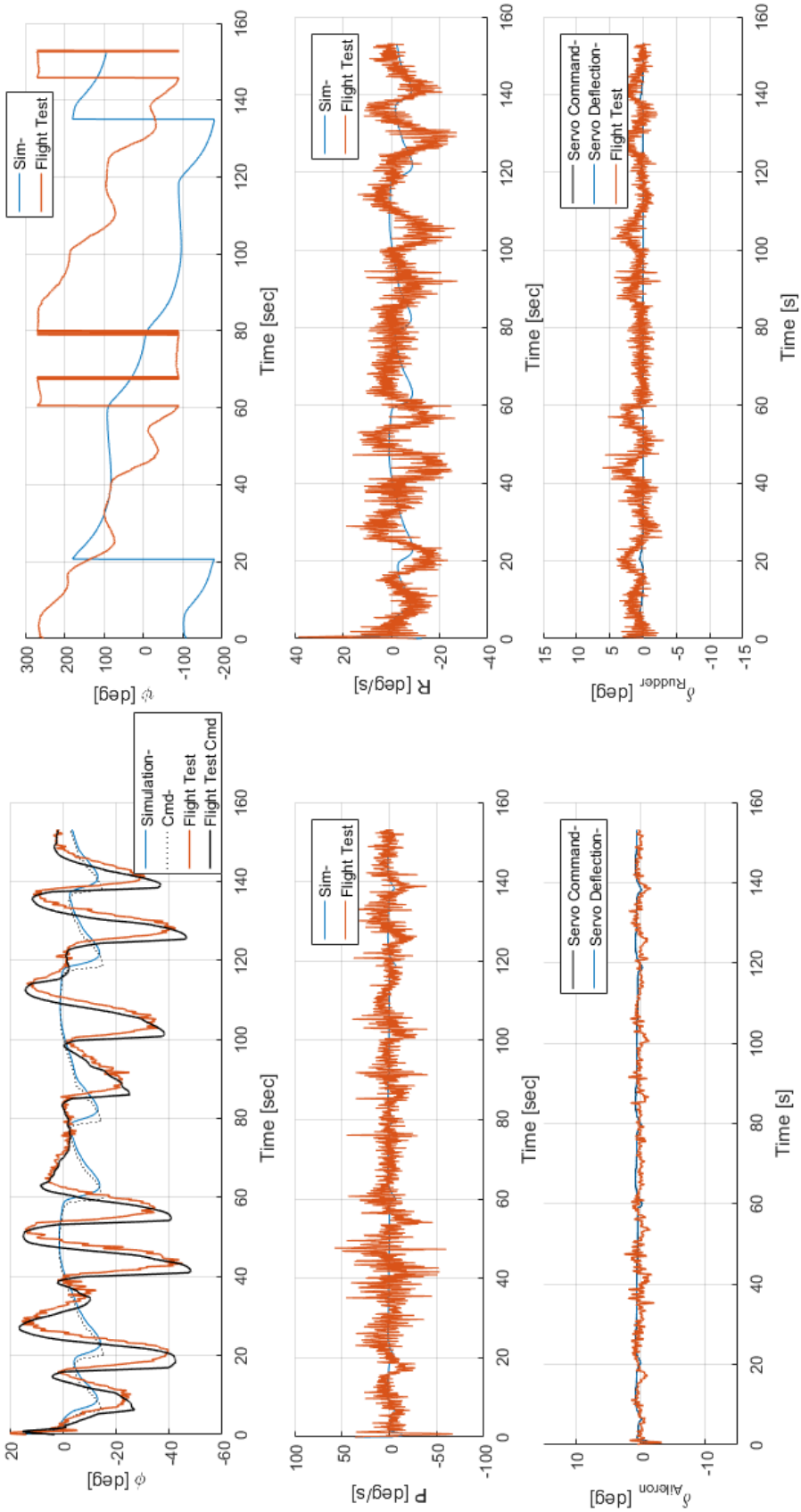


Figure 4.67: Skyhunter solo flight test and simulation comparison, Lateral states and control

4.6 Swarm flight: Skyhunter, April, 28, 2018

In this flight test, the swarm of two Skyhunters is tested. This flight uses the identical scenario as the aforementioned DG-808 swarm flight. The virtual leader is initiated by the average position and velocity of two agents and planned to follow the race track pattern. The desired relative distance from the virtual leader is presented in the following table.

Table 4.12: Desired relative distance from virtual leader

Agent	North [ft]	East [ft]
1st agent	0	0
2nd agent	-50	50

The altitude of the higher agent is set to stay at 475 ft AGL while the lower is set to stay at 377 ft AGL and the assessment criteria is defined in Table 4.9. Improved tracking shown in the previous flight test helped improve swarm flight performance as well. Two agents are engaged at different position and heading angles as presented in Figure 4.73 and Figure 4.76. The first and second agents heading angles are 154 and 75 degree, respectively.

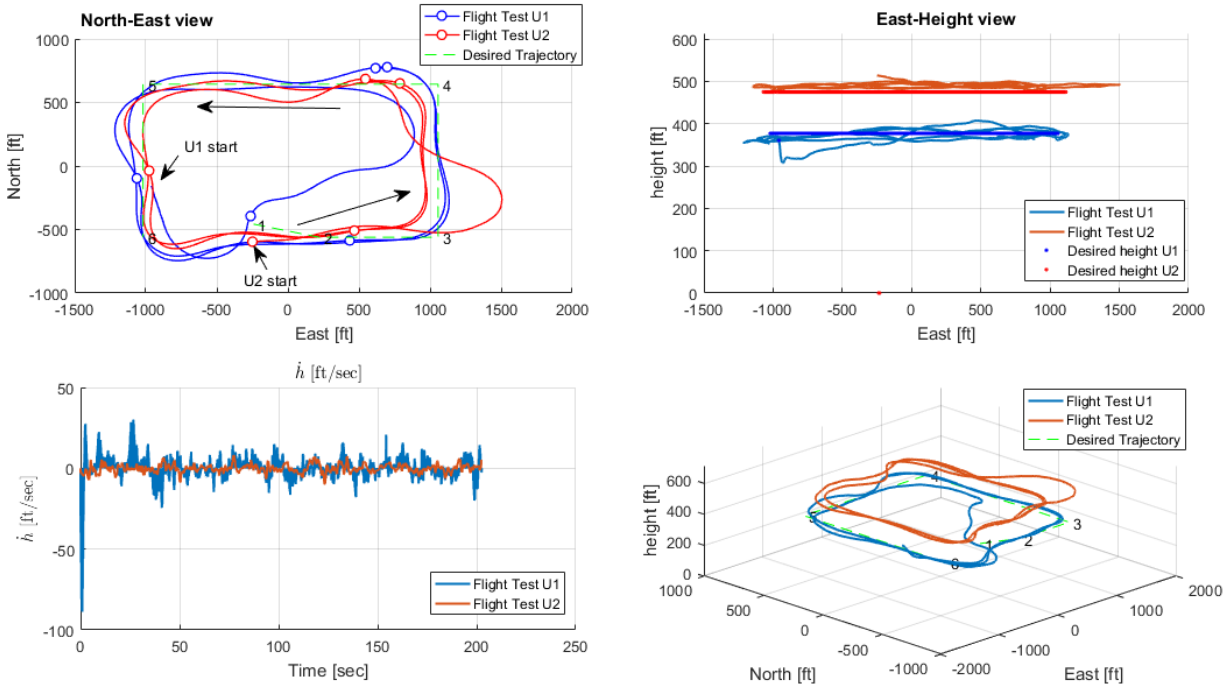


Figure 4.68: Skyhunter formation flight results, Position and the altitude rate

Then, two agents are aggregated to the formation. Figure 4.69 shows the RMS of 2D and the altitude tracking. Due to the confined flight area, the actual formation holding is considered only on the straight line segment in the race track pattern. This is because the aircraft needs time to change its heading. While the aircraft turns, the altitude and the formation holding is not accurate due to the reduced lift by the roll angle. The RMS of this flight test is significantly lower 70.2% and 30% than the DG-808 1st and 2nd attempt in the swarm flight test, even though the race track is smaller than DG-808 flight test (West-east lag is 469 ft shorter than the DG-808 race track), see Figure 4.71. The average of the tracking quality is improved by 50%. Besides of the smaller race track, the wind speed is also higher than the day when DG-808 flight tests was conducted (the average wind speed was 4.6 mph). In this flight test, the average wind speed was 10.4 mph and gusting up to 12.7 mph. The wind direction was east and northeast. However, the formation distance (average RMS is 91 ft) could not keep at the desired formation distance (70.71 ft). The controller can hold the aircraft speed fairly well (the overshoot is 7.73% ($\leq$ 10%) for aircraft 2, see Figure 4.70) and the altitude hold quality (the overshoot is 2.4 (aircraft 1) and 3.16% (aircraft 2) $\leq$ 10% by the defined criteria, see Table 4.9)is also excellent consequently. For aircraft 1, the overshoot of the speed command tracking is 12.3% so it is not acceptable result.

The detail of the comparison between the flight test and simulation for each agents is presented as follows. Figure 4.72 and Figure 4.75 shows the position comparison of the aircraft between the flight test and the simulation. Figure 4.73 and Figure 4.76 show the longitudinal states and controls for each agent. Figure 4.74 and Figure 4.77 show the lateral-directional states and controls for each agent. As these figures show, there is discrepancy between the simulation and flight test due to the external disturbances (e.g., wind). The lateral tracking is drifted 248.8 ft (northwest corner for aircraft 1) and 200 ft (southeast corner for aircraft 2) comparing to the simulation results. In addition, the speed command is chosen by the average of two agents' GPS velocity. For aircraft 2, the speed command is set to 56.9 ft/sec. However, the speed command for aircraft 1 changed several times due

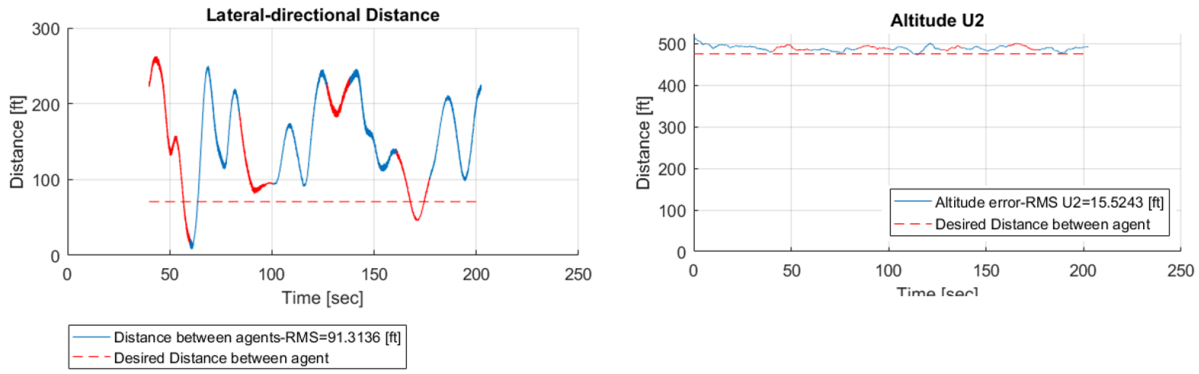


Figure 4.69: Formation distance comparison between flight test and simulation with RMS

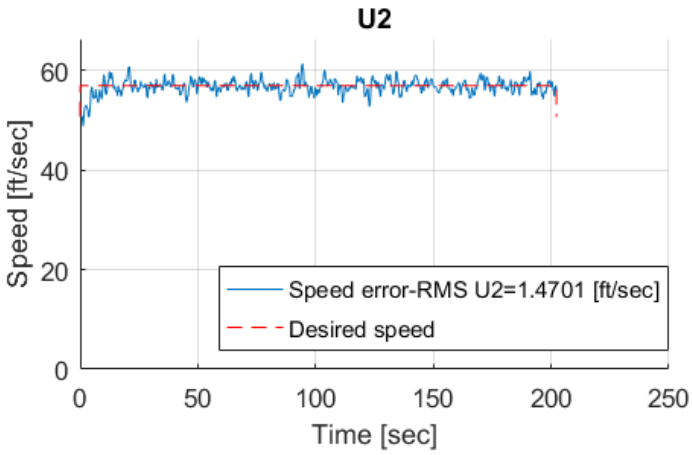


Figure 4.70: Skyhunter swarm flight speed tracking RMS

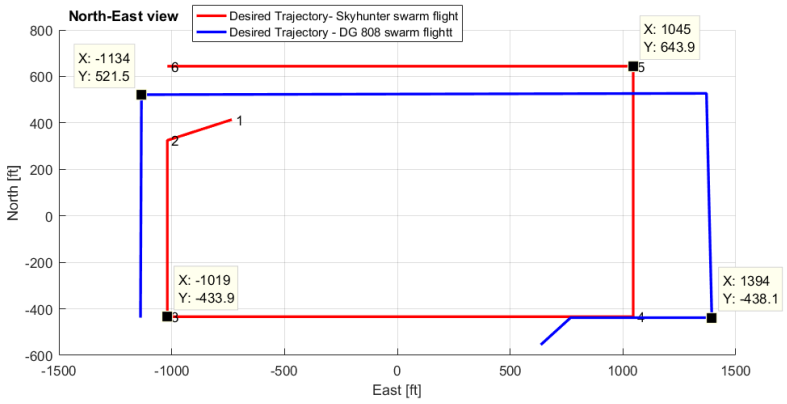


Figure 4.71: Given waypoints for Skyhunter swarm flight and DG-808 swarm flight

to the intermittent communication link. Aircraft 1 initialized the swarm flight mode several times and recalculate the speed command. In the simulation, the lack of wind results in

lower speed commands. Regarding aircraft 2, the pitch angle tracking is not acceptable (the maximum error is 7 degrees$\geq$ 3 degrees). The roll angle tracking is required further improvement for the confined area. The maximum roll angle error is 6 degree for aircraft 2 even only small roll angle section is considered. Aircraft 1 has unacceptable pitch angle tracking since it has the steady state error about 7 degrees. For the roll angle tracking of aircraft 1, maximum roll angle error is 8 degrees which also does not meet the criteria.

- Aircraft 1

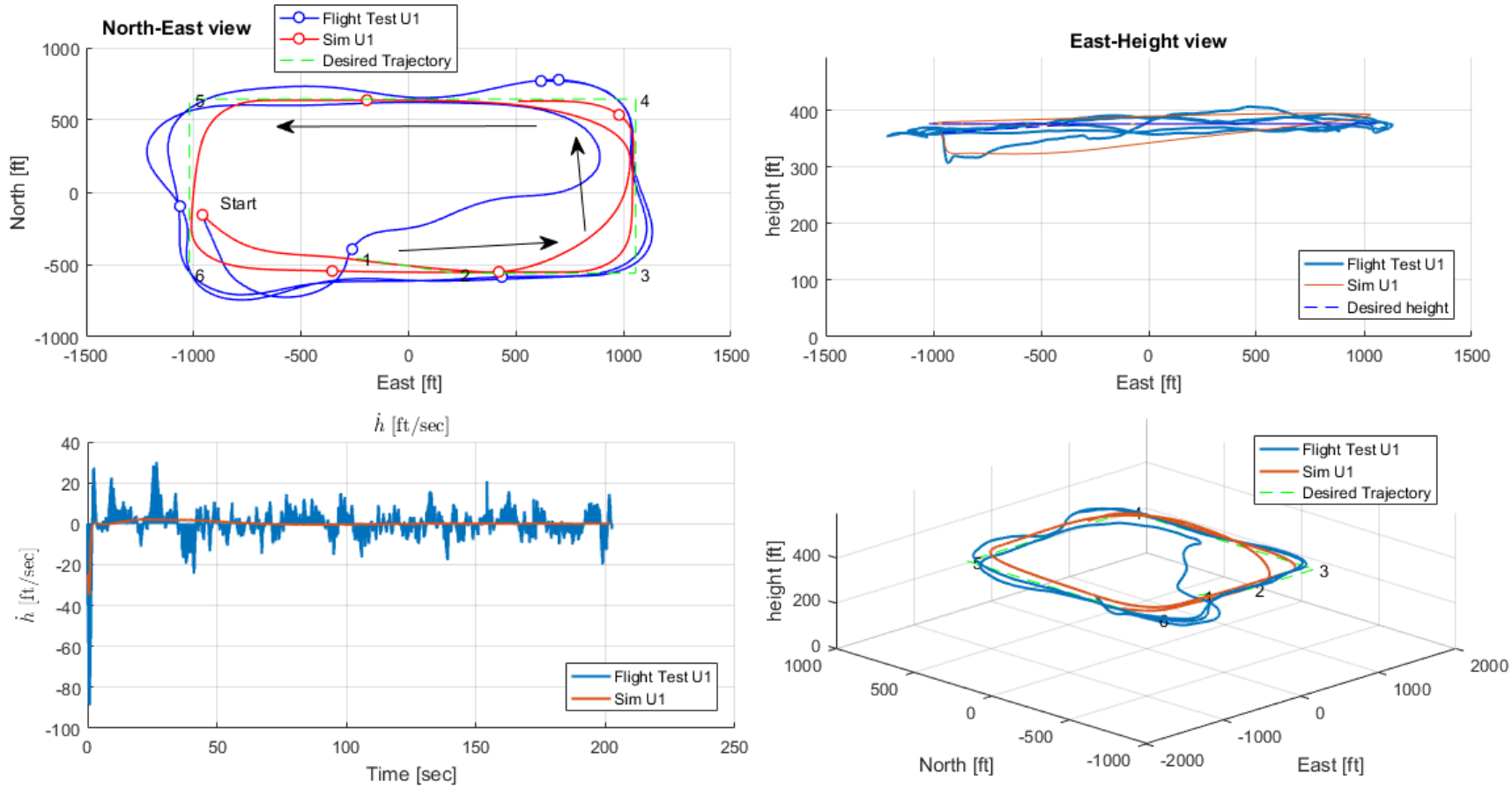


Figure 4.72: Skyhunter formation flight test and simulation comparison, Position

- Aircraft 2

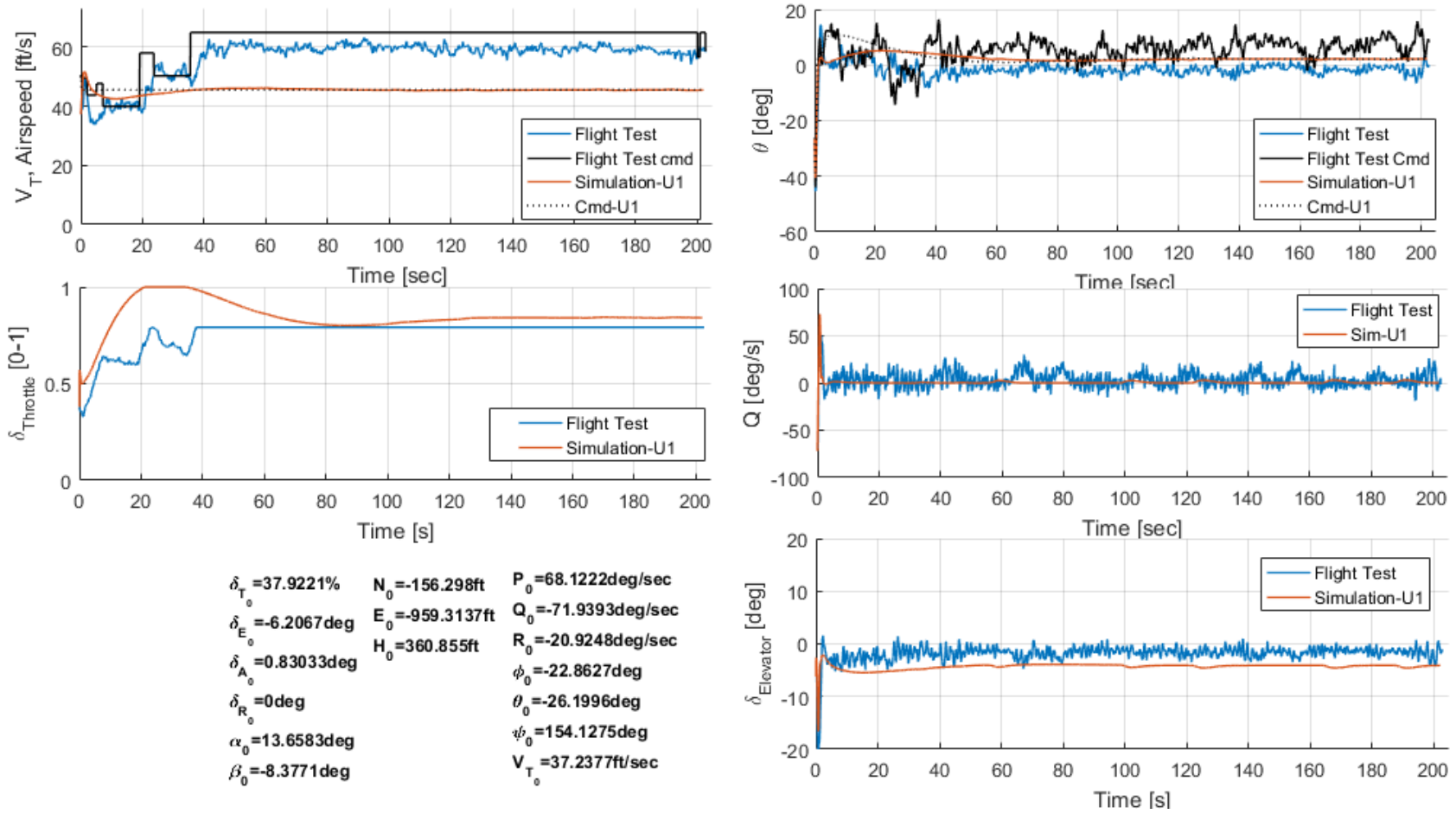


Figure 4.73: Skyhunter formation flight test and simulation comparison, Longitudinal states and controls

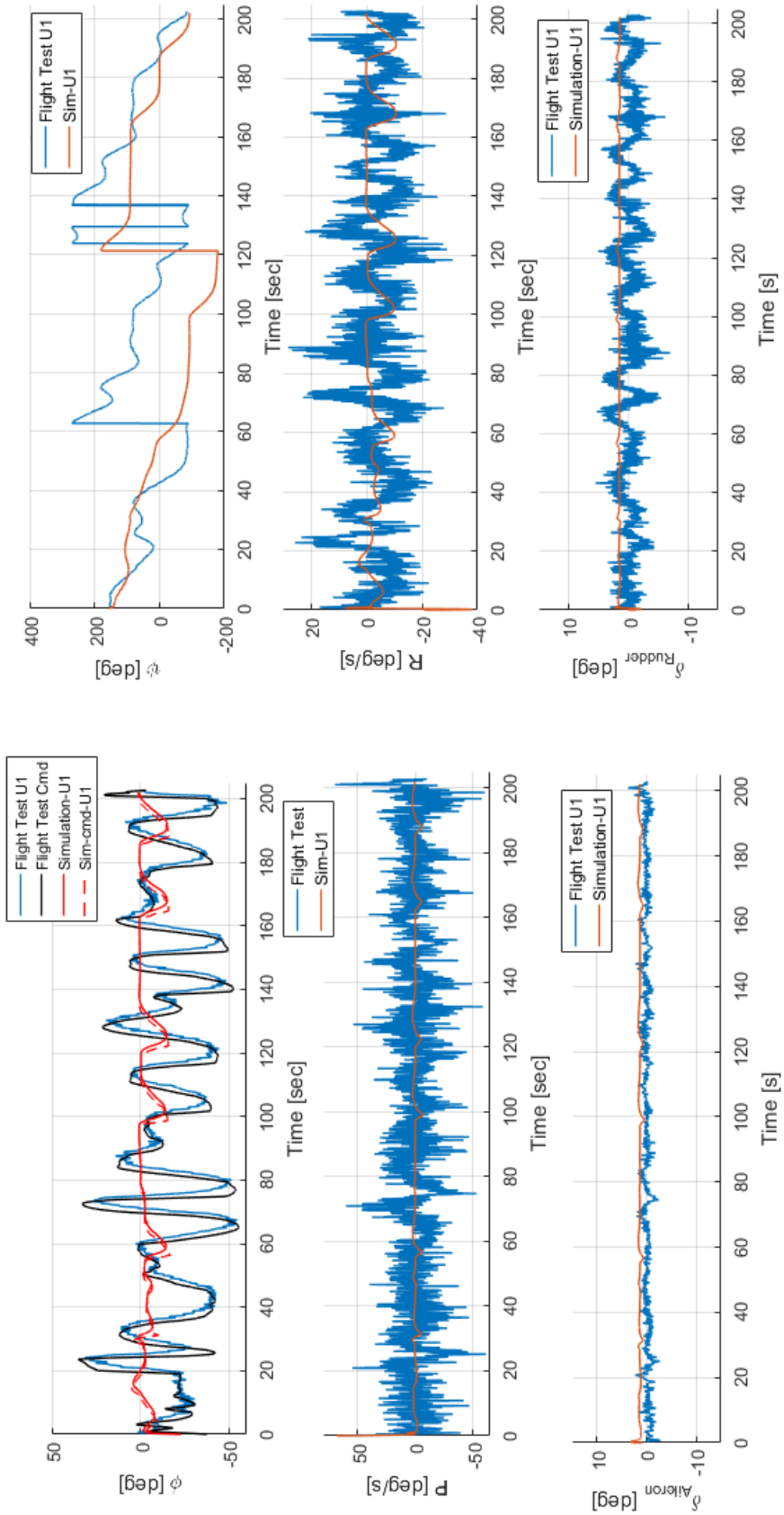


Figure 4.74: Skyhunter formation flight test and simulation comparison, Lateral states and controls

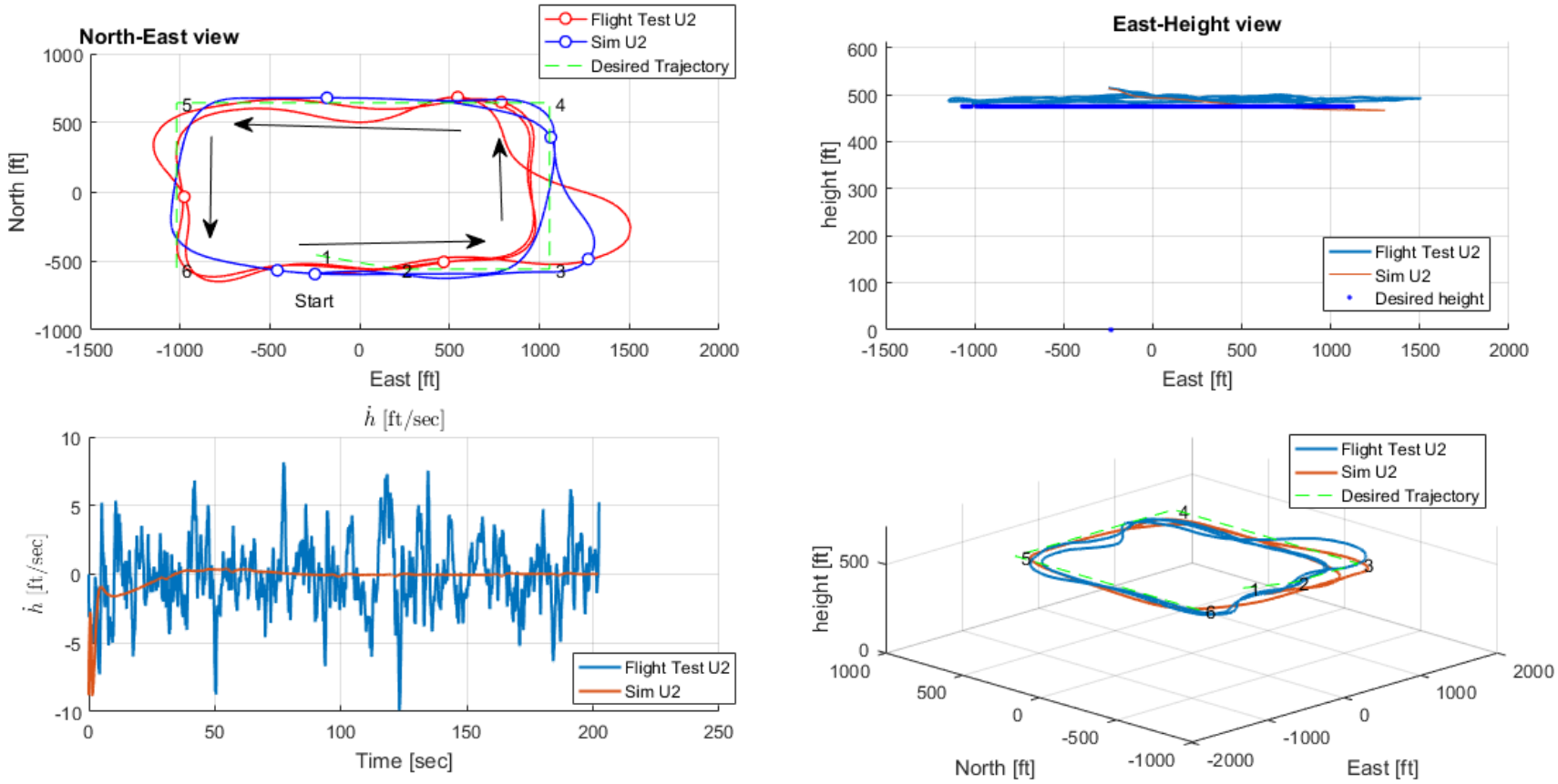


Figure 4.75: Skyhunter formation flight test and simulation comparison, Position

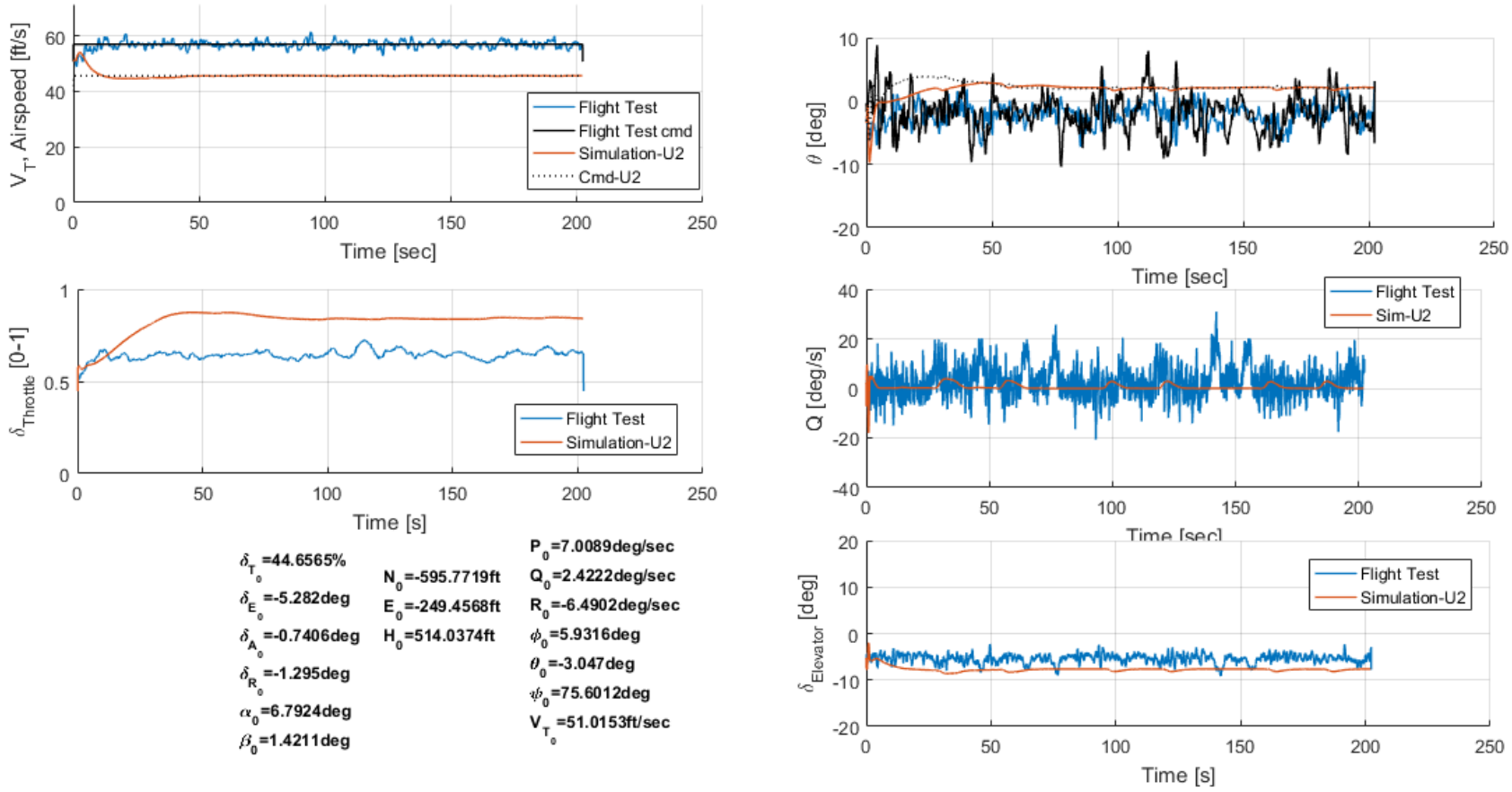


Figure 4.76: Skyhunter formation flight test and simulation comparison, Longitudinal states and controls

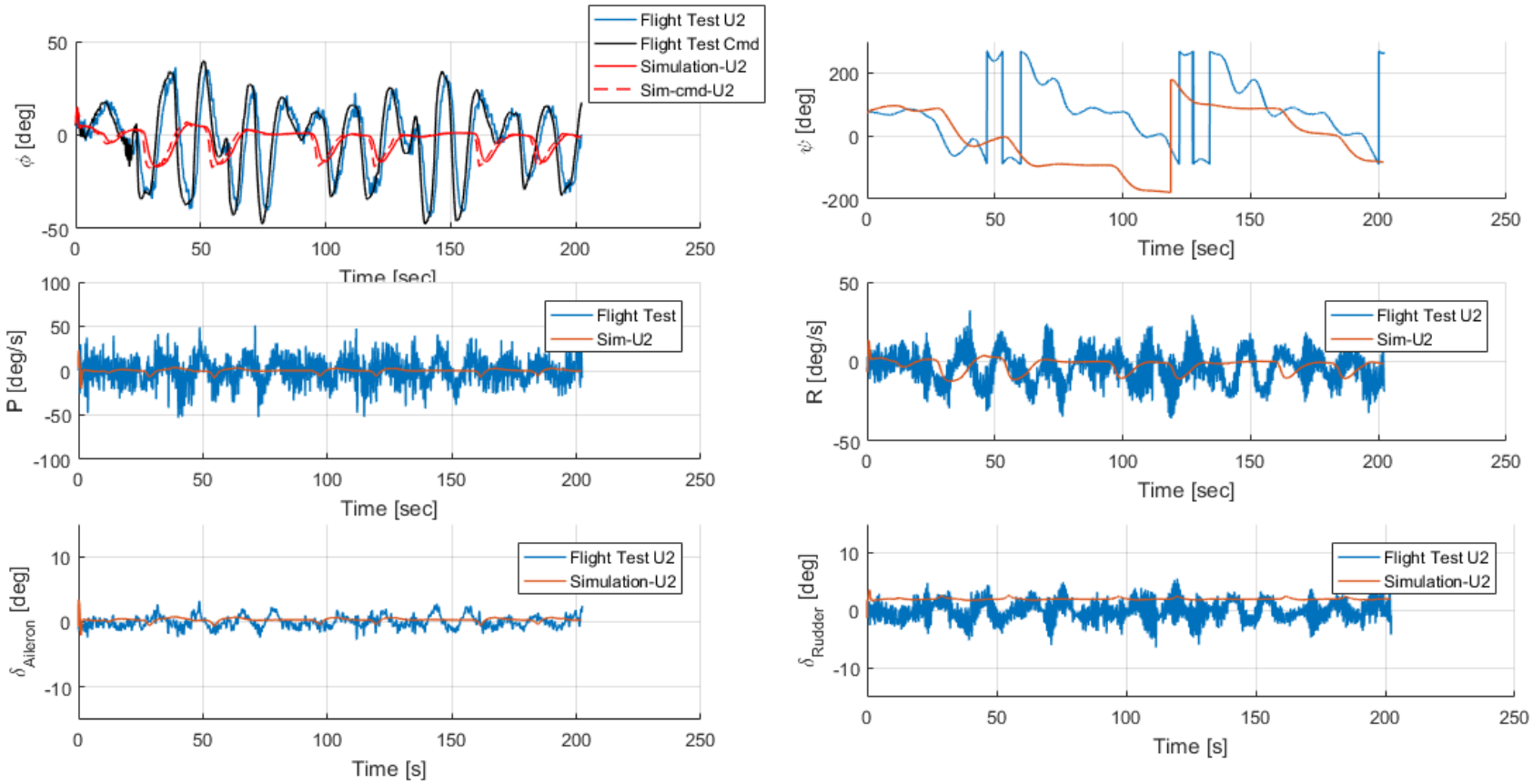


Figure 4.77: Skyhunter formation flight test and simulation comparison, Lateral states and controls

4.7 Yak-54 40% solo flight test, December 20th 2017

This flight test is to validate the solo flight by using Yak-54 40% with the aforementioned GNC algorithms. The guidance parameters and controller gains are tuned for Yak-54 40% due to the changed dynamic model from DG-808 and Skyhunter. The following figures describe the result of this flight test: the position of the aircraft (Figure 4.78), the lateral-directional, and longitudinal states and controls (Figure 4.79 and 4.80, respectively).

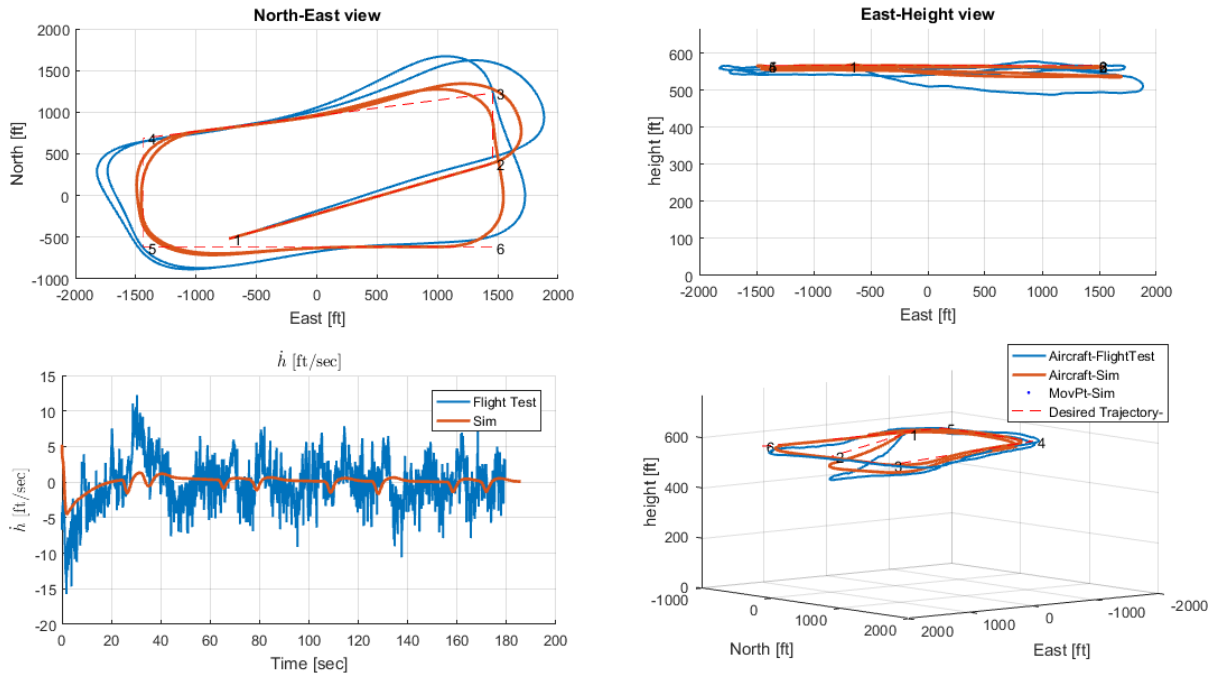


Figure 4.78: G1X Position and the altitude rate

Figure 4.79 and 4.80 shows the lateral-directional and longitudinal states and control, respectively. In order to quantify the tracking performance, RMS of the tracking error is calculated when the roll angle is small. The overshoot in the lateral-tracking when the aircraft flies the straight line is 1.8% ($\leq$ 10%) which meets the criteria defined in Table 4.9. Regarding the controller performance, the roll angle tracking is not met the criteria (maximum 7 degrees ($\geq$ 3 degrees) between 99 and 101 seconds). The speed command is tracked loosely (the maximum overshoot 17.55% $\geq$ 10% criteria in Table 4.9). In addition, the pitch angle

tracking is not met the requirement since the maximum pitch angle error is 4 degrees (≥ 3 degrees). Pitch rate is not also acceptable since it exceed 15 degrees in magnitude when the aircraft turns. Overall, the controller performance is required to be improved to meet the defined criteria.

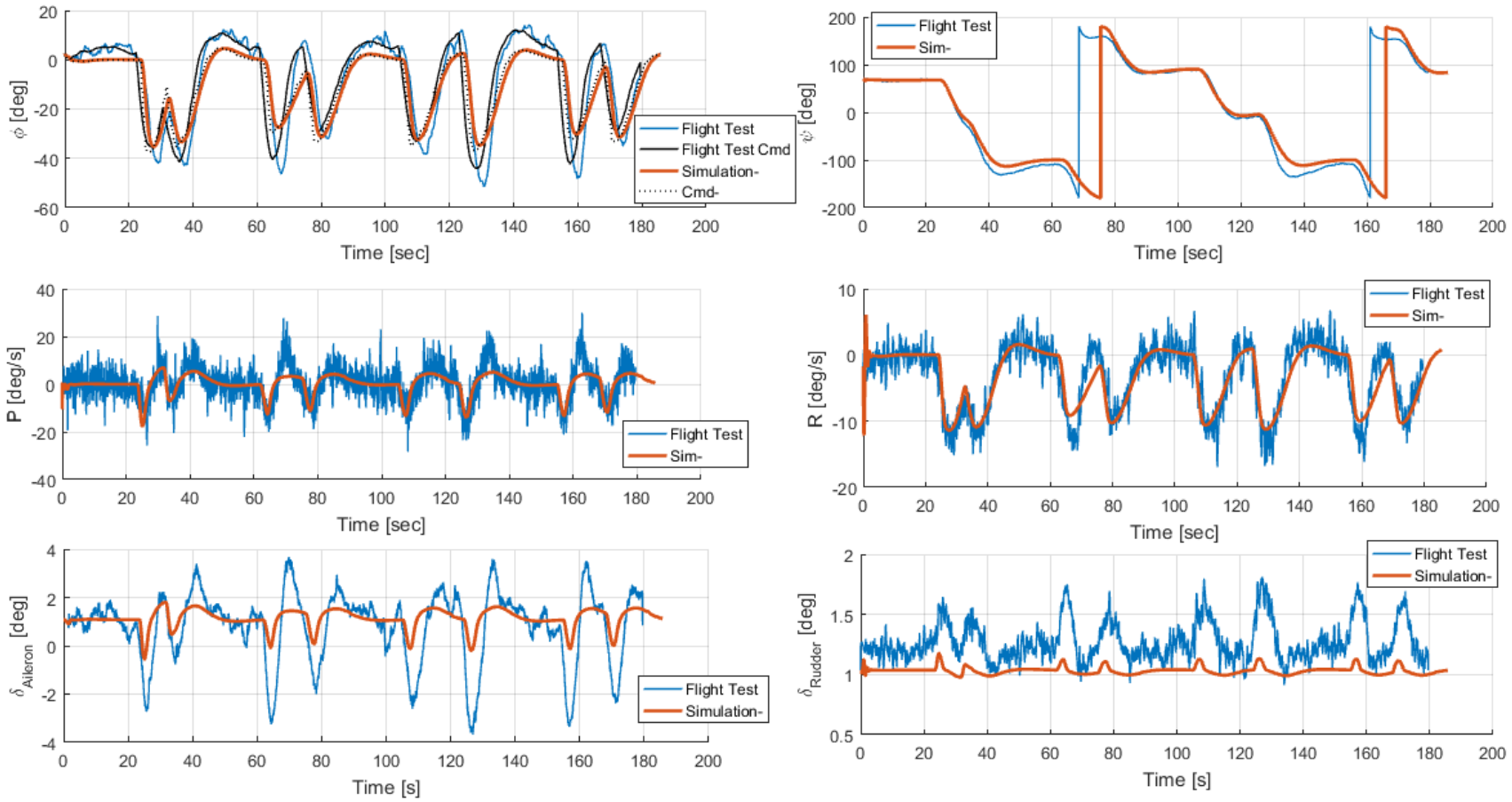


Figure 4.79: G1X Lateral-directional states and controls

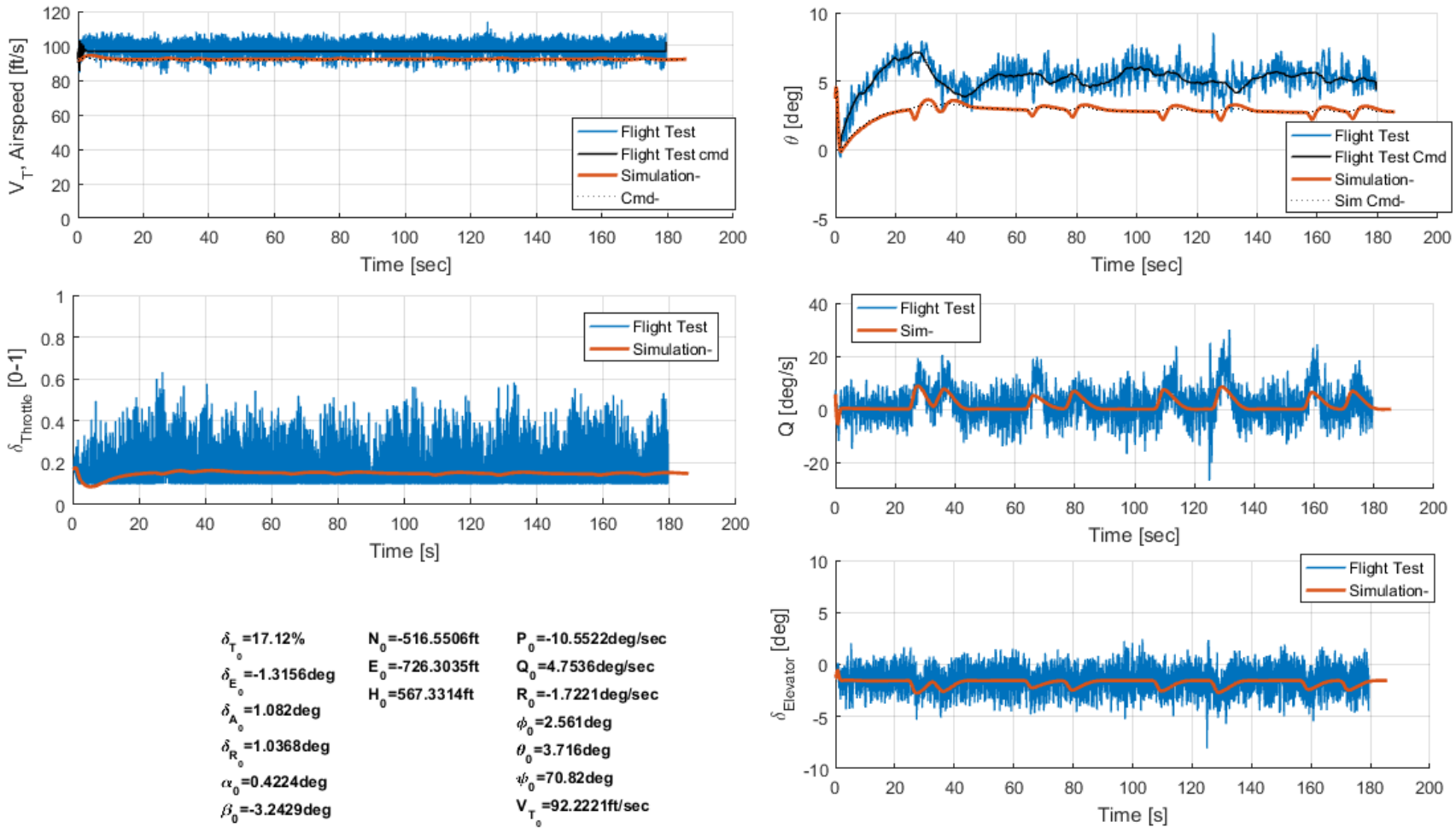


Figure 4.80: G1X Longitudinal states and controls

Chapter 5

Discussion and Conclusion

5.1 Discussion

In this research, intelligent guidance, navigation, and control algorithms are developed in order to utilize the multi-agent UASs for complex missions. The phasic navigation has been proposed to overcome challenges in the multi-agent autonomous flight (e.g., random initial conditions). The navigation algorithm for the multi-agent systems must contain the significant elements such as scalability, collision avoidance, and dynamic constraints, see Eq. 2.115. In order to achieve all these required features, curvature control, the Hungarian algorithm, and the moving mesh methods are proposed as phasic navigation. The curvature control algorithm is able to deal with random heading angles to converge to the desired direction. This implies that it possesses scalability, as convergence is not affected by the number of agents. By limiting curvature with the minimum turning radius, the planned path does not violate any dynamic constraints. After all heading angles have converged, the Hungarian algorithm provides the optimal solution based on the proposed cost function to assign each aircraft to the formation position. A new feature of the Hungarian algorithm is the moving virtual terminal to assign the position sequentially until all agents are assigned. The new cost function is suggested to consider the agents' dynamics by adding the rotation angle and

heading angle errors. The moving mesh algorithms showed that the inner agents' position always stayed inside of the formation even though the desired trajectory is challenging to complete (e.g., 180 degree turn). In order to utilize the moving mesh methods, the outer agents are distinguished by Delaunay Triangulation to define the mesh boundary. The phasic navigation algorithm has been implemented with the 6 DoF aircraft dynamics simulator. With the given random initial conditions of multiple agents, the navigation algorithm is able to guide the agents to the desired formation shape, see Sec. 4.2. The flight testbed is designed by using the DG-808 and Skyhunter to validate the fundamental blocks of developed GNC architecture. Mainly, the waypoint modification and index decision algorithm, LQ guidance path planning, virtual point formation, multi-scale moving point guidance and LQR controllers are assessed by the criteria defined in Table 4.1 and 4.9. The multi-scale moving point has been developed since the identical reference point in the lateral-directional and longitudinal plane causes oscillation in the trajectory tracking. In order to reflect the aircraft dynamics and improve tracking, the reference points are calculated separately in the lateral-directional and longitudinal plane. The tracking quality is improved by 80% on the lateral-directional plane comparing to the moving point guidance. Though the RMS of the longitudinal tracking error is increased by 45%, the maneuvers performed by the aircraft become significantly smoother (the original algorithm fluctuates from 340 ft to 440 ft but the multi-scale guidance slowly increase the altitude from 350 ft to 400 ft in 200 seconds.) comparing to the moving point guidance. However, the difference between the aircraft speed command and the moving point traveling speed causes 113% overshoot in the tracking, see Sec. 4.5.3. The sigmoid function is implemented to improve the lateral tracking by adjusting the length to the reference point ($d_{dR_{Lat}}$). As a result, the RMS of tracking error is reduced by 99% in both lateral-directional and longitudinal plane comparing to the Skyhunter flight test conducted in August, 2017. In addition, the swarm flight tracking in the flight test is improved by 50% in the tracking quality comparing to the DG-808 flight test conducted in March, 2017. Monte Carlo simulations are conducted to increase the credibility of the algo-

rithm implementation. The random initial conditions are drawn during the simulation and the algorithms are tested exhaustively and the implementation error (e.g., generating NaN values) can be identified. In addition to the multi-scale moving point guidance algorithm, the LQR controller is designed with the observation of the control surface rate. The smoothing algorithms and the trim control surface update algorithm are implemented to prevent the abrupt changes when the autopilot is engaged. For example, this solution can reduce the controller surface rate by 7 %/s for the throttle and 26 deg/s for the elevator. As a result, the DG-808 and the Skyhunter do not have abrupt behavior during the flight test. Overall, the pitch angle tracking is not satisfied for the defined criteria in the flight tests. The controller design should be performed to enhance the pitch angle tracking. Also, the 3 degrees criteria for the roll and pitch angle tracking might be too narrow since the unstructured environment introduces the external disturbances (e.g., the wind field and gust).

5.2 Conclusion

This research solves existing problems of multi-agent systems, specifically robustness towards random initial conditions of each agent in 3D space. The combination of seemingly heterogeneous algorithms from different engineering disciplines are modified to evolve a new generation of guidance, navigation and control algorithms for scalable numbers of fixed wing multi-agent UASs. Uniquely, fundamental blocks of guidance, navigation, and control were validated through numerous simulations, hardware-in-the-loop tests, and actual UAS flight tests. In order to deal with the complexity of multi-agent systems, intelligent and robust guidance, navigation, and control systems were designed and validated by flight tests. Phasic navigation was developed for multi-agent systems to aggregate, assemble a formation, and maintain the formation shape during flight. Furthermore, the multi-scale moving point guidance helps reduce oscillation and improves the tracking of the formation. An LQR controller was designed practically by taking into consideration control surface rates in order

to limit abrupt effects on the aircraft. Testbeds for the hardware-in-the-loop were designed to verify communication links and the implementation of algorithms of the on-board avionics. Lastly, flight tests were conducted to thoroughly validate the developed platforms and algorithms. In conclusion, this research shows that the complexity of multi-agent systems can be overcome by phasic navigation and multi-scale guidance. Flight tests expand the boundary of this research by validating the fundamental building blocks of the multi-agent system architecture. Although the RMS of distance between agents was reduced from 306 ft to 91 ft but for the purpose of science missions, the distance between agents must be several folds smaller than validation and verification flight test (e.g., 15 ft for 35 MHz radar). In addition, the scalability of UASs has the practical bottleneck which is a mesh network.

5.3 Recommendations

- The focus of this research was on advancing guidance and navigation of multi-agent UASs and improvements were mainly achieved by pushing the boundaries on the performance and functionality of guidance and navigation algorithms. Future work should include replacing existing control algorithms with more advanced controllers featuring robustness and adaptivity capabilities.

- In this work, the connectivity of mesh network telemetry has been investigated in the laboratory setting only using three systems. The connectivity tests should be expanded to incorporate more agents and should include actual flight testing. The robustness of guidance and navigation algorithms should be tested and investigated with respect to loss of communication between agents. The robustness of guidance and navigation algorithms must be also tested as a function of latency and periodic communication loss.

- The guidance, navigation and control of multi agent UASs in urban areas are fundamentally different from Earth science missions. Morphing, splitting and random scalability should be investigated in dynamically changing environments.

- This work investigated the stability of proposed guidance, navigation and control of multi-agent systems using limited number of agents and exhaustive search (Monte Carlo analysis). It is recommended to use mathematical methods to prove stability of nonlinear guidance, navigation and control for a random number of UASs.

- Validation and verification section of this work focused on important blocks of guidance, navigation, and control of multi-agent systems. However, more advanced hardware-in-the-loop simulations involving several agents must be conducted to identify potential flaws in proposed guidance, navigation and control algorithms.

- Validation and verification flight test must be conducted for larger number of agents (minimum four agents due to the moving mesh constraints) to evaluate performance to phasic guidance, navigation methods in actual scenarios.

- Quantification of performance goals for further enhancement by flight tests with an advanced controller to meet the criteria defined in Table 4.9.

- It is recommended the implementation of proper extended Kalman filter to estimate the wind and the airflow angles (α and β) Currently, the airflow angles does not included for calculating the control commands. If the airflow angles can be estimated properly, it will help enhancing the controller performance.

References

[1] (2017). Department of defense announces successful micro-drone demonstration. Department of Defense, IMMEDIATE RELEASE, Release No: NR-008-17.

[2] Almeida, P., Bencatel, R., Gonçalves, G. M., Sousa, J. B., and Ruetz, C. (2007). Experimental results on command and control of unmanned air vehicle systems. *IFAC Proceedings Volumes*, 40(15):239 – 244. 6th IFAC Symposium on Intelligent Autonomous Vehicles.

[3] Almurib, H. A. F., Nathan, P. T., and Kumar, T. N. (2011). Control and path planning of quadrotor aerial vehicles for search and rescue. In *SICE Annual Conference 2011*, pages 700–705.

[4] Alonso-Mora, J., Breitenmoser, A., Rufli, M., Siegwart, R., and Beardsley, P. (2011). Multi-robot system for artistic pattern formation. In *2011 IEEE International Conference on Robotics and Automation*, pages 4512–4517.

[5] Andersson, M. and Wallander, J. (2004). Kin selection and reciprocity in flight formation? *Behavioral Ecology*, 15(1):158–162.

[6] Barnes, L., Fields, M. A., and Valavanis, K. (2007). Unmanned ground vehicle swarm formation control using potential fields. *2007 Mediterranean Conference on Control and Automation*, pages 1–8.

[7] Bayezit, I. and Fidan, B. (2013). Distributed cohesive motion control of flight vehicle formations. *IEEE Transactions on Industrial Electronics*, 60(12):5763–5772.

[8] Bayraktar, S., Fainekos, G. E., and Pappas, G. J. (2004). Experimental cooperative control of fixed-wing unmanned aerial vehicles. In *2004 43rd IEEE Conference on Decision and Control (CDC) (IEEE Cat. No.04CH37601)*, volume 4, pages 4292–4298 Vol.4.

[9] Boskovic, J. D., Li, S.-M., and Mehra, R. K. (2002). Formation flight control design in the presence of unknown leader commands. In *Proceedings of the 2002 American Control Conference (IEEE Cat. No.CH37301)*, volume 4, pages 2854–2859 vol.4.

[10] Brierley, A. S. and Cox, M. J. (2010). Shapes of krill swarms and fish schools emerge as aggregation members avoid predators and access oxygen. *Current Biology*.

[11] Campa, G., Gu, Y., Seanor, B., Napolitano, M., Pollini, L., and Fravolini, M. (2007). Design and flight-testing of non-linear formation control laws. *Control Engineering Pratice*.

[12] Cao, H., Bai, Y., and Liu, H. (2012). Distributed rigid formation control algorithm for multi-agent systems. *Kybernetes*, 41(10):1650–1661.

[13] Cao, Y. (2008). Menkres assignment algorithm.

[14] Chang, D. E., Shadden, S., Marsden, J. E., and Olfati-Saber, R. (2003). Collision avoidance for multiple agent systems. In *Proceedings of the 42nd IEEE Conference on Decision and Control*.

[15] Chao, Z., Zhou, S.-L., Ming, L., and Zhang, W.-G. (2012). Uav formation flight based on nonlinear model predictive control. *Mathematical Problems in Engineering*.

[16] Chen, G., Ziyang, Z., Huajun, G., and Yili, S. (2014). Constraints for unmanned aerial vehicles formation flight path. In *Proceedings of 2014 IEEE Chinese Guidance, Navigation and Control Conference*, pages 1106–1109.

[17] Chen, Q. and Li, Y. (2016). Uavs formation flight control based on following of the guidance points. In *2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC)*, pages 730–735.

[18] Chung, T. H., Clement, M. R., Day, M. A., Jones, K. D., Davis, D., and Jones, M. (2016). Live-fly, large-scale field experimentation for large numbers of fixed-wing uavs. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1255–1262.

[19] Conde, R., Llata, J. R., and Torre-Ferrero, C. (2017). Time-varying formation controllers for unmanned aerial vehicles using deep reinforcement learning. *CoRR*, abs/1706.01384.

[20] Cordeiro, T. F. K., Ferreira, H. C., and Ishihara, J. Y. (2017). Non linear controller and path planner algorithm for an autonomous variable shape formation flight. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1493–1502.

[21] D'Amato, E., Notaro, I., Silvestre, F., and Mattei, M. (2017). Bi-level flight path optimization for uav formations. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 690–697.

[22] Day, M. A., Clement, M. R., Russo, J. D., Davis, D., and Chung, T. H. (2015). Multi-uav software systems and simulation architecture. In *International Conference on Unmanned Aircraft Systems (ICUAS)*.

[23] Di, Y., Li, R., Tang, T., and Zhang, P. (2005). Moving mesh finite element methods for the incompressible navier–stokes equations. *SIAM Journal on Scientific Computing*, 26(3):1036–1056.

[24] Elias, B. (2012). Pilotless drones: Background and considerations for congress regarding unmanned aircraft operations in the national airspace system. *CRS Report for Congress*.

[25] Fan, D. D., Theodorou, E., and Reeder, J. (2017). Evolving cost functions for model predictive control of multi-agent uav combat swarms. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '17, pages 55–56, New York, NY, USA. ACM.

[26] Fidan, B., Yu, C., and Anderson, B. D. O. (2007). Acquiring and maintaining persistence of autonomous multi-vehicle formations. *IET Control Theory Applications*, 1(2):452–460.

[27] Garcia, G. (2013). *Decentralized Robust Nonlinear Model Predictive Controller For Unmanned Aerial Systems*. PhD thesis, The University of Kansas.

[28] Garcia, G. and Keshmiri, S. (2011). Nonlinear model predictive controller for navigation, guidance and control of a fixed-wing uav. In *AIAA Guidance, Navigation, and Control Conference*.

[29] Garcia, G., Keshmiri, S., and Colgren, R. (2010). Advanced h-infinity trainer autopilot. In *AIAA Modeling and Simulation Technologies Conference, American Institute of Aeronautics and Astronautics*.

[30] Garcia, R. and Barnes, L. (2010). Multi-uav simulator utilizing x-plane. *Journal of Intelligence Robot System*, 57:393–406.

[31] Giulietti, F., Innocenti, M., Napolitano, M., and Pollini, L. (2005). Dynamic and control issues of formation flight. *Aerospace Science and Technology*, 9(1):65 – 71.

[32] Giulietti, F., Pollini, L., and Innocenti, M. (2000). Autonomous formation flight. *IEEE Control Systems*, 20(6):34–44.

[33] Goktogan, A. H. and Sukkarieh, S. (2009). Distributed simulation and middleware for networked uas. *Journal of Intelligence Robot System*, 54:331–357.

[34] Guelman, M., Schilling, K., and Barnett, D. L. (2012). Formation flight line of sight guidance. *Acta Astronautica*, 71(Supplement C):163 – 169.

[35] Hattenberger, G., Alami, R., and Lacroix, S. (2006). Planning and control for unmanned air vehicle formation flight. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5931–5936.

[36] Hauert, S., Leven, S., Varga, M., Ruini, F., Cangelosi, A., Zufferey, J. C., and Floreano, D. (2011). Reynolds flocking in reality with fixed-wing robots: Communication range vs. maximum turning rate. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5015–5020.

[37] Hino, T. and Takeshi, T. (2013). *Decentralised Formation Control of Unmanned Aerial Vehicles Using Virtual Leaders*, pages 133–144. Springer Japan.

[38] Hino, T. and Tsuchiya, T. (2011). Formation control of small unmanned aerial vehicles using virtual leader and point-to-multipoint communication. *TRANSACTIONS OF THE JAPAN SOCIETY FOR AERONAUTICAL AND SPACE SCIENCES*, 54(184):83–89.

[39] How, J., King, E., and Kuwata, Y. (2004a). Flight demonstrations of cooperative control for uav teams. In *AIAA 3rd "Unmanned Unlimited" Technical Conference, Workshop and Exhibit*.

[40] How, J., King, E., and Kuwata, Y. (2004b). Flight demonstrations of cooperative control for uav teams. In *AIAA 3rd Unmanned Unlimited Technical Conference Workship and Exhibit*.

[41] Huang, W. (2001). Variational mesh adaptation: isotropy and equidistribution. *J. Comput. Phys.*

[42] Huang, W. and Kamenski, L. (2015). A geometric discretization and a simple implementation for variational mesh generation and adaptation. *J. Comput. Phys.*

[43] Huang, W. and Russell, R. D. (2011). *Adaptive Moving Mesh Methods*. Springer, New York.

[44] Huang, W., Wang, Y., Yang, H., Yi, X., and Yang, X. (2016). *Distributed Control for Formation Switch of Fixed Wing MAVs*. Springer Singapore.

[45] Hui, M. Y., Yang, C. Q., Xi, H. Z., and Zheng, G. (2015). An improved nonlinear guidance law for unmanned aerial vehicles path following. In *2015 34th Chinese Control Conference (CCC)*, pages 5271–5276.

[46] Justh, E. W. and Krishnaprasad, P. S. (2005). Natural frames and interacting particles in three dimensions. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 2841–2846.

[47] Kai, C., Yuanqing, X., and Kaoli, H. (2016). UAV formation control design with obstacle avoidance in dynamic three-dimensional environment. *SpringerPlus*, 5(1):1124.

[48] Kim, A. R., Keshmiri, S., Huang, W., and Garcia, G. (2016). Guidance of multi-agent fixed-wing aircraft using a moving mesh method. *Unmanned Systems*, 04(03):227–244.

[49] Kim, A. R., Vivekanandan, P., McNamee, P., Sheppard, I., Blevins, A., and Sizemore, A. (2017). Dynamic modeling and simulation of a quadcopter with motor dynamics. In *AIAA Modeling and Simulation Technologies Conference*.

[50] Kim, B. S., Calise, A. J., and Sattigeri, R. J. (2007). Adaptive, integrated guidance and control design for line-of-sight-based formation flight. *Journal of Guidance, Control, and Dynamics*.

[51] Kim, S. J. and Whang, I. H. (2012). Acceleration constraints for maneuvering formation flight trajectories. *IEEE Transactions on Aerospace and Electronic Systems*, 48(2):1052–1060.

[52] Kownacki, C. and Ołdziej, D. (2016). Fixed-wing uavs flock control through cohesion and repulsion behaviours combined with a leadership. *International Journal of Advanced Robotic Systems*, 13(1):36.

[53] Kuhn, H. W. (2010). *The Hungarian Method for the Assignment Problem*, pages 29–47. Springer Berlin Heidelberg, Berlin, Heidelberg.

[54] Lamont, G. B., Slear, J. N., and Melendez, K. (2007). Uav swarm mission planning and routing using multi-objective evolutionary algorithms. In *2007 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making*, pages 10–20.

[55] Lee, J. W. and Kim, H. J. (2007). Trajectory generation for rendezvous of unmanned aerial vehicles with kinematic constraints. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 1056–1061.

[56] Lee, T., Baines, M., Langdon, S., and Tindall, M. (2013). A moving mesh approach for modelling avascular tumour growth. *Applied Numerical Mathematics*, 72(Supplement C):99 – 114.

[57] Leonard, N. E. and Fiorelli, E. (2001). Virtual leaders, artificial potentials and coordinated control of groups. In *Proceedings of the 40th IEEE Conference on Decision and Control (Cat. No.01CH37228)*, volume 3, pages 2968–2973 vol.3.

[58] Leven, S. (2011). *Enabling Large-Scale Collective Systems in Outdoor Aerial Robotics*. PhD thesis, ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE.

[59] Lissaman, P. and Shollenberger, C. (1970). Formation flight of birds. *American Association of rhte Advancement of Science*.

[60] Lugo-Cardenas, I., Salazar, S., and Lozano, R. (2016). The mav3dsim hardware in the loop simulation platform for research and validation of uav controller. In *International Conference on Unmanned Aircraft Systems (ICUAS)*.

[61] Luo, D., Zhou, T., and Wu, S. (2013). Obstacle avoidance and formation regrouping strategy and control for uav formation flight. In *2013 10th IEEE International Conference on Control and Automation (ICCA)*, pages 1921–1926.

[62] Macdonald, E. A. (2011). Multi-robot assignment and formation control. Master's thesis, Georgia Institute of Technology.

[63] Margot, X., Hoyas, S., Fajardo, P., and Patouna, S. (2010). A moving mesh generation strategy for solving an injector internal flow problem. *Mathematical and Computer Modelling*, 52(7):1143 – 1150. Mathematical Models in Medicine, Business and Engineering 2009.

[64] Meng, W., Hu, Y., Lin, J., Lin, F., and Teo, R. (2015). Ros+unity: An efficient high-fidelity 3d multi-uav navigation and control simulator in gps-denied environments. In *IECON 2015*.

[65] Miller, P. (2007). The genius of swarms. *National Geographic*.

[66] Muijres, F. T. and Dickinson, M. H. (2014). Bird flight: Fly with a little flap from your friends. *Nature*.

[67] Nelson, D. R., Barber, D. B., McLain, T. W., and Beard, R. W. (2007). Vector field path following for miniature air vehicles. *IEEE Transactions on Robotics*, 23(3):519–529.

[68] Nguyen, B. Q., Chuang, Y.-L., Tung, D., Hsieh, C., Jin, Z., Shi, L., Marthaler, D., Bertozzi, A., and Murray, R. M. (2005). Virtual attractive-repulsive potentials for cooperative control of second order dynamic vehicles on the caltech mvwt. In *Proceedings of the 2005, American Control Conference, 2005.*, pages 1084–1089 vol. 2.

[69] Ning, S. A. (2011). *Aircraft Drag Reduction Through Extended Formation Flight*. PhD thesis, Stanford University.

[70] Odelga, M., Stegagno, P., Bulthoff, H. H., and Ahmad, A. (2015). A setup for multi-uav hardware-in-the-loop simulations. In *2015 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*.

[71] Olfati-Saber, R., Dunbar, W. B., and Murray, R. M. (2003). Cooperative control of multi-vehicle systems using cost graphs and optimization. In *Proceedings of the 2003 American Control Conference, 2003.*, volume 3, pages 2217–2222 vol.3.

[72] Olfati-Saber, R. and Murray, R. M. (2002). Graph rigidity and distributed formation stabilization of multi-vehicle systems. In *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, volume 3, pages 2965–2971.

[73] Pachter, M., D'Azzo, J., and Proud, A. (2001). Tight formation flight control. *Journal of Guidance, Control, and Dynamics*.

[74] Park, S., Deyst, J., and How, J. (2004). A new nonlinear guidance logic for trajectory tracking. In *AIAA Guidance, Navigation, and Control Conference and Exhibit, American Institute of Aeronautics and Astronautics*, Rhode Island, USA.

[75] Partridge, D. and Baines, M. (2011). A moving mesh approach to an ice sheet model. *Computers and Fluids*, 46(1):381 – 386. 10th ICFD Conference Series on Numerical Methods for Fluid Dynamics (ICFD 2010).

[76] Paul, T., Krogstad, T., and Gravdahl, J. T. (2008a). Uav formation flight using 3d potential field. In *2008 16th Mediterranean Conference on Control and Automation*, pages 1240–1245.

[77] Paul, T., Krogstad, T. R., and Gravdahl, J. T. (2008b). Modelling of uav formation flight using 3d potential field. *Simulation Modelling Practice and Theory*, 16(9):1453 – 1462.

[78] Peterson, C. K. and Barton, J. (2015). Virtual structure formations of cooperating uavs using wind-compensation command generation and generalized velocity obstacles. In *2015 IEEE Aerospace Conference*, pages 1–7.

[79] Quintero, S. A. P., Collins, G. E., and Hespanha, J. P. (2013). Flocking with fixed-wing uavs for distributed sensing: A stochastic optimal control approach. In *2013 American Control Conference*, pages 2025–2031.

[80] Ratnoo, A., Sujit, P., and Kothari, M. (2011). Adaptive optimal path following for high wind flights. In *18th IFAC World Congress*, volume 44.

[81] Regina, N. and Zanzi, M. (2013). Surface target-tracking guidance by self-organizing formation flight of fixed-wing uav. In *2013 IEEE Aerospace Conference*, pages 1–10.

[82] Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *Computer Graphics*.

[83] Riehl, J. R., Collins, G. E., and Hespanha, J. P. (2011). Cooperative search by uav teams: A model predictive approach using dynamic graphs. *IEEE Transactions on Aerospace and Electronic Systems*, 47(4):2637–2656.

[84] Rinaldi, F., Chiesa, S., and Quagliotti, F. (2013). Linear quadratic control for quadrotors uavs dynamics and formation flight. *Journal of Intelligent & Robotic Systems*, 70(1):203–220.

[85] Roskam, J., editor (2011). *Airplane Flight Dynamics and Automatic Flight Controls*. DARcorporation.

[86] Seo, J., Kim, Y., Kim, S., and Tsourdos, A. (2017). Collision avoidance strategies for unmanned aerial vehicles in formation flight. *IEEE Transactions on Aerospace and Electronic Systems*, PP(99):1–1.

[87] Smalley, D. (2015). Locust: Autonomous, swarming uavs fly into the future. Office of Naval Research, Corporate Strategic Communications, Media Releases.

[88] Stastny, T., Garcia, G., and Keshmiri, S. (2014). Collision and obstacle avoidance in

unmanned aerial systems using morphing potential field navigation and nonlinear model predictive control. *ASME Journal of Dyn Sys Meas Control*, 137.

[89] Stephen, C., Kenneth, K., Peter, P., and Linton, W. (2005). *Unmanned Aircraft Systems Roadmap 2005-2030*. Office of the Undersecretary of Defense for Acquistion, Technology and Logistics.

[90] Su, H., Wang, X., and Lin, Z. (2007). Flocking of multi-agents with a virtual leader part ii: with a virtual leader of varying velocity. In *2007 46th IEEE Conference on Decision and Control*, pages 1429–1434.

[91] Sui, Z., Pu, Z., and Yi, J. (2017). Optimal uavs formation transformation strategy based on task assignment and particle swarm optimization. In *2017 IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 1804–1809.

[92] Tanner, H. G., Pappas, G. J., and Kumar, V. (2004). Leader-to-formation stability. *IEEE Transactions on Robotics and Automation*, 20(3):443–455.

[93] Tonetti, S., Hehn, M., Lupashin, S., and D'Andrea, R. (2011). Distributed control of antenna array with formation of uavs. *IFAC Proceedings Volumes*, 44(1):7848 – 7853. 18th IFAC World Congress.

[94] Turpin, M., Michael, N., and Kumar, V. (2014). Capt: Concurrent assignment and planning of trajectories for multiple robots. *Int. J. Rob. Res.*, 33(1):98–112.

[95] United States Government Accountability Office (2013). Defense acquisitions: Assessments of selected weapon programs. *Report to Congressional Committees, GAO-13-294SP*.

[96] Vásárhelyi, G., Virágh, C., Somorjai, G., Tarcai, N., Szörényi, T., Nepusz, T., and Vicsek, T. (2014). Outdoor flocking and formation flight with autonomous aerial robots. *CoRR*, abs/1402.3588.

[97] Vivekanandan, P., Garcia, G., Yun, H., and Keshmiri, S. (2016). A simplex architecture for intelligent and safe unmanned aerial vehicles. In *2016 IEEE 22nd International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 69–75.

[98] W. Huang and L. Kamenski (2016). On the mesh nonsingularity of the moving mesh pde method. *WIAS Preprint*, 2218.

[99] Wang, J., Wu, X., and Xu, Z. (2008). Potential-based obstacle avoidance in formation control. *Journal of Control Theory and Applications*, 6(3):311–316.

[100] Wei, Y., Madey, G. R., and Blake, M. B. (2013). Agent-based simulation for uav swarm mission planning and execution. In *Proceedings of the Agent-Directed Simulation Symposium*, ADSS 13, pages 2:1–2:8, San Diego, CA, USA. Society for Computer Simulation International.

[101] Weimerskirch, H., Martin, J., Clerquin, Y., Alexandre, P., and Jiraskova, S. (2001). Energy saving in flight formation. *Nature*.

[102] Whitzer, M., Keller, J., Bhattacharya, S., Kumar, V., Sands, T., Ritholtz, L., Pope, A., and Dickmann, D. (2016). In-flight formation control for a team of fixed-wing aerial vehicles. In *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 372–380.

[103] Wu, X., Yang, Z., Huo, J., and Zhang, J. (2015). Uav formation control based on consistency. In *2015 7th International Conference on Modelling, Identification and Control (ICMIC)*, pages 1–5.

[104] Xi, X. and Abed, E. H. (2005). Formation control with virtual leaders and reduced communications. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 1854–1860.

[105] Zeyu, J., Hui, C., Zefeng, Z., Xiaowu, Z., Xinyao, N., Weibin, L., Yuxiang, Z., and Shing, W. W. (2016). Autonomous formation flight of uavs: Control algorithms and field experiments. In *2016 35th Chinese Control Conference (CCC)*, pages 7585–7591.

[106] Zheping, Y., Yibo, L., Jiajia, Z., and Gengshi, Z. (2015). Moving target following control of multi-auvs formation based on rigid virtual leader-follower under ocean current. In *2015 34th Chinese Control Conference (CCC)*, pages 5901–5906.

[107] Zou, Y. and Pagilla, P. R. (2009). Distributed formation flight control using constraint forces. *Journal of Guidance, Control, and Dynamics*.

[108] Zunli, N., Xuejun, Z., and Xiangmin, G. (2017). Uav formation flight based on artificial potential force in 3d environment. In *2017 29th Chinese Control And Decision Conference (CCDC)*, pages 5465–5470.

Appendix A

Aircraft Flight Dynamics

The fundamental theory to design Unmanned Aerial Systems (UASs) starts from Flight dynamics because it provides mathematical tools for analyzing dynamic modes, simulating an aircraft in software environment, and designing controllers. The procedure of quantifying aircraft characteristics (e.g., stability and control derivatives) is called a dynamic modeling. This chapter will provide the mathematical derivation of the dynamic modeling: six Degrees of Freedom (DoF) nonlinear equations of motion and Linear Time Invariant (LTI) state space. Mathematical models begin with Newton's second law and Euler's law for translational and rotational motion, respectively. These laws lead to nonlinear 6 DoF equations of motion. A perturbation method is applied to describe various motions at off-trim conditions during the steady state flight. Using these linearized perturbed equations of motion, LTI state space model can be developed to design controllers and perform modal analysis.

A.1 Nonlinear 6 Degrees of Freedom Equations of motion

A.1.1 Frames and Coordinate System

Frame has a very important role in Flight Dynamics since it provides the reference of the motion. In order to construct the frame, the following properties should be met.

- At least 3 non-colinear points are needed and they have to be orthonormal with respect to the origin.
- Frame is a continuous and unbounded set of points that are time invariant.

In this work, Earth frame is chosen as the inertial frame: North, East, and Down axes. It is the reference frame that does not accelerate with time. Inertial frame is important since it provides the reference of the motion. In addition, the available information or measured states for the controllers are in the inertial coordinate system from sensors such as Global Positioning System (GPS) and Inertial Measurement Unit (IMU). There are various coordinate systems used in this work: body coordinate, and stability coordinate. Body coordinate system originates at the center of gravity and moves with the body. x_B axis is along with the nose of the body, and y_B axis aligned with the right wing. z_B axis is the down direction and the result of cross product of x_B and y_B. The body coordinate is significant since forces and moments are acting on the body of a vehicle not on the inertial frame. The stability coordinate system is a modified version of the body frame. The x axis of the stability axis (x_S) is aligned with the projection of the velocity vector onto the xz plane. y_S is aligned with y_B. z_S is perpendicular to x_S. All aforementioned coordinate systems in this section are orthonormal. Figure A.1 shows the details of the coordinate axes and the inertial frame.

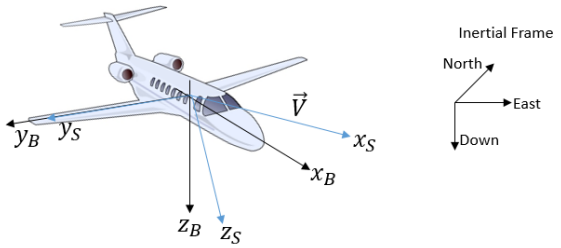


Figure A.1: Coordinate systems with inertial frame

A.1.2 Tensors and Transformation of Coordinate Systems

Coordinate systems are very useful for describing the motion of the aircraft. However, it cannot be utilized without the transformation between the coordinate systems because the forces and moments act on the body coordinate but measured information (e.g., position, velocity etc.) is in the inertial coordinate system. Tensor is a useful tool for transforming the coordinate system. It is a generalized set of vectors and describes the coordinate elements of physical properties such as position, velocity, and acceleration. The order of tensors (n) is determined by the dimensions (m) of the vector. In flight dynamics, zero, first, and second order tensors are interested since the motion of the aircraft acts in three dimensional space. The number of element in the tensor can be determined as m^n. The attitude angle information is required to transform the coordinate system by using tensors. (e.g., Euler Angles, $\Theta_I = [\phi, \theta, \psi]^T$). The transformation rule and the number of elements in tensors are introduced as follows:

- Zero order tensor: $3^0 = 1$, there is only 1 element in zero order tensor. Scalar or magnitudes are zero order tensor (e.g. mass, speed). There is no transformation rule for the zero order tensor. Mass or speed is identical in any coordinate system under the inertial frame.

- First order tensor: $3^1 = 3$, there are three elements in the first order tensor. Position, velocity, and acceleration are typical first order tensors. In order to transform the coordinate system of the first order tensor ($[\mathbf{x}]_A$) from A to C, the following equation is used: $[\mathbf{x}]_C = H_A^C [\mathbf{x}]_A$. For the rate of the first order tensor, the following equation can be used for the transformation.

$$\left[\frac{d\mathbf{x}}{dt}\right]_A = \left[\left[\frac{d\mathbf{x}}{dt}\right]_C\right]_A + \omega_A \times [\mathbf{x}]_A \tag{A.1}$$

- Second order tensor: $3^2 = 9$, there are nine elements in the second order tensor (e.g.

moment of inertia, stress, strain). For second order tensors, the following equation is used for transforming the coordinate system from A to C:

$$[\mathbf{x}]_C = H_A^C [\mathbf{x}]_A H_C^A \tag{A.2}$$

H is the rotation or cosine matrix. The subscript of H means the current coordinate and superscript indicates the desired coordinate. ω_A is the angular velocity in the A coordinate system. Then, the rotation matrix should be derived. Rotation direction has the convention, which rotates from the Z axis, Y axis and X axis.

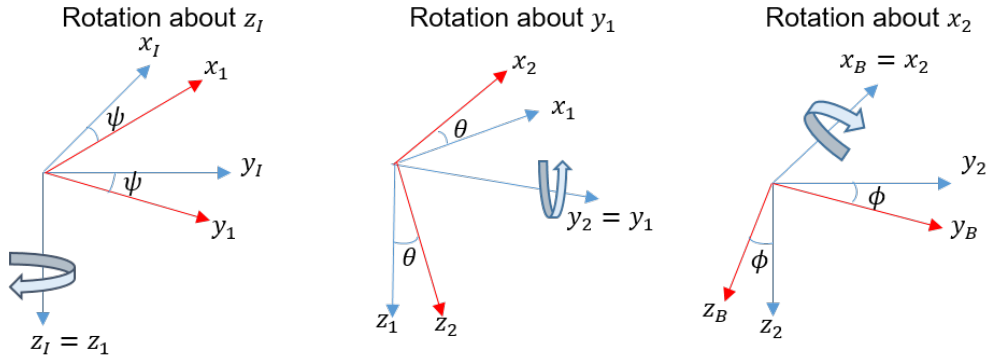


Figure A.2: Coordinate Rotation Convention and Each rotation with Euler angles

Therefore, the transformation matrix can be found as the following equation.

$$H_I^B = H_2^B(\phi) H_1^2(\theta) H_I^1(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.3}$$

Since the inertial frame and body coordinate system are orthonormal, the inverse of the rotation matrix (H_I^B) is identical as the transpose of H_I^B.

$$H_B^I = \left(H_I^B\right)^{-1} = \left(H_I^B\right)^T \tag{A.4}$$

A.1.3 Derivation of Nonlinear 6 Dof Nonlinear Equations of Motion and Kinematic Equations

The derivation of aircraft translational and rotational motion begins with Newton's second law (Eq. A.5) and Euler's law (Eq. A.6).

$$\left[\sum F_{A+P+G}\right]_I = \left[\frac{dP}{dt}\right]_I \tag{A.5}$$

$$\left[\sum M_{A+P}\right]_I = \left[\frac{dh}{dt}\right]_I \tag{A.6}$$

Subscript I indicates the inertial coordinate system. The sum of aerodynamic forces, propulsion force, and the gravity force are equal to the derivative of momentum in translation and rotation directions. The velocity in the body frame can be integrated and transferred to the inertial frame to obtain a trajectory of the vehicle using the tensors. Now, Eq. A.5 and Eq. A.6 can be expanded using the transformation rule in the rate of the first order tensor, Eq. A.1.

$$\left[\frac{dP}{dt}\right]_I = \left[\left[\frac{dP}{dt}\right]_B\right]_I + \omega_I \times P_I = H_B^I \left[\frac{dP}{dt}\right]_B + \omega_I \times P_I \tag{A.7}$$

$$\left[\frac{dh}{dt}\right]_I = \left[\left[\frac{dh}{dt}\right]_B\right]_I + \omega_I \times h_I = H_B^I \left[\frac{dh}{dt}\right]_B + \omega_I \times h_I \tag{A.8}$$

With a cross product equivalent matrix which is a skew symmetric, the computation can be convenient. The following equation is the cross product equivalent.

$$\widetilde{\omega}_I \triangleq \omega_I \times = \begin{bmatrix} 0 & -\omega_z & +\omega_y \\ +\omega_z & 0 & -\omega_x \\ -\omega_y & +\omega_x & 0 \end{bmatrix} \tag{A.9}$$

Eq. A.9 is applied to Eq. A.7 and Eq. A.8.

$$H_B^I \left[\frac{dP}{dt}\right]_B = [F_{A+P+G}]_I - \widetilde{\omega}_I P_I \tag{A.10}$$

$$H_B^I \left[\frac{dh}{dt}\right]_B = [M_{A+P}]_I - \widetilde{\omega}_I h_I \tag{A.11}$$

Multiply H_I^B on both hand sides.

$$H_I^B \left\{ H_B^I \left[\frac{dP}{dt}\right]_B = [F_{A+P+G}]_I - \widetilde{\omega}_I P_I \right\} \tag{A.12}$$

$$H_I^B \left\{ H_B^I \left[\frac{dh}{dt}\right]_B = [M_{A+P}]_I - \widetilde{\omega}_I h_I \right\} \tag{A.13}$$

Expand the equations.

$$\left[\frac{dP}{dt}\right]_B = H_I^B [F_{A+P+G}]_I - H_I^B \widetilde{\omega}_I P_I = [F_{A+P+G}]_B - H_I^B \widetilde{\omega}_I H_B^I P_B \tag{A.14}$$

$$\left[\frac{dh}{dt}\right]_B = H_I^B [M_{A+P}]_I - H_I^B \widetilde{\omega}_I h_I = [M_{A+P}]_B - H_I^B \widetilde{\omega}_I H_B^I h_I \tag{A.15}$$

Using second order tensor transformation (Eq. A.1), the cross product equivalent can be transformed to the body coordinate system.

$$\left[\frac{dP}{dt}\right]_B = [F_{A+P+G}]_B - \widetilde{\omega}_B P_B \tag{A.16}$$

$$\left[\frac{dh}{dt}\right]_B = [M_{A+P}]_B - \widetilde{\omega}_B h_I \tag{A.17}$$

Then, the chain rule is applied to expand the left hand side.

$$\left[\frac{dm}{dt}\right]_B \overrightarrow{V}_B + m \left[\frac{d\overrightarrow{V}}{dt}\right]_B = [F_{A+P+G}]_B - \widetilde{\omega}_B P_B \tag{A.18}$$

$$\left[\frac{dI}{dt}\right]_B \omega_B + I \left[\frac{d\omega}{dt}\right]_B = [M_{A+P}]_B - \widetilde{\omega}_B h_I \tag{A.19}$$

where $P = mV$, and $h = I\omega$. The rate of mass change is assumed to be less than 3% to

5% in 60 seconds, so the change of mass ($\dot{m}$) and moment of inertia ($\dot{I}$) are negligible.

$$m\left[\frac{d\overrightarrow{V}}{dt}\right]_B = \left[\sum F_{A+P}\right]_B + \left[\sum F_G\right]_B - m\widetilde{\omega}_B\overrightarrow{V}_B \tag{A.20}$$

$$I\left[\frac{d\omega}{dt}\right]_B = \left[\sum M_{A+P}\right]_B - \widetilde{\omega}_B I_B \omega_B \tag{A.21}$$

The gravity vector is also transformed from the inertial coordinate to the body coordinate system.

$$\left[\sum F_G\right]_B = H_I^B \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix}_I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix}_I \tag{A.22}$$

Expand the equation and it is simplified as the following:

$$\left[\sum F_G\right]_B = mg \begin{bmatrix} -\sin\theta \\ \cos\theta\sin\phi \\ \cos\theta\cos\phi \end{bmatrix} \tag{A.23}$$

Therefore, the 6 DoF nonlinear Equations of motion can be found by substituting Eq. A.9 and Eq. A.23.

$$\begin{bmatrix} \dot{U} \\ \dot{V} \\ \dot{W} \end{bmatrix}_B = \frac{1}{m}\begin{bmatrix} F_{x_{A+P}} \\ F_{y_{A+P}} \\ F_{z_{A+P}} \end{bmatrix}_B + \begin{bmatrix} -g\sin\theta \\ g\cos\theta\sin\phi \\ g\cos\theta\cos\phi \end{bmatrix}_B - \begin{bmatrix} 0 & -R & Q \\ R & 0 & -P \\ -Q & P & 0 \end{bmatrix}_B \begin{bmatrix} U \\ V \\ W \end{bmatrix}_B \tag{A.24}$$

$$\begin{bmatrix} I_{xx} & 0 & -I_{zz} \\ 0 & I_{yy} & 0 \\ -I_{zx} & 0 & I_{zz} \end{bmatrix}_B \begin{bmatrix} \dot{P} \\ \dot{Q} \\ \dot{R} \end{bmatrix}_B = \begin{bmatrix} L_{A+P} \\ M_{A+P} \\ N_{A+P} \end{bmatrix}_B - \begin{bmatrix} 0 & -R & Q \\ R & 0 & -P \\ -Q & P & 0 \end{bmatrix}_B \begin{bmatrix} I_{xx} & 0 & -I_{zz} \\ 0 & I_{yy} & 0 \\ -I_{zx} & 0 & I_{zz} \end{bmatrix}_B \begin{bmatrix} P \\ Q \\ R \end{bmatrix}_B \tag{A.25}$$

In this work, the aircraft was assumed to be fixed-wing vehicle, and to have symmetry configurations on xz plane, so I_{yz} and I_{xy} are zero. The moment of inertia can be estimated

using Advanced Aircraft Analysis (AAA) or the Bifilar pendulum experiment, see Ref. [49]. If this matrix is expanded, nonlinear six DoF equations of motion can be obtained as the form of differential equations.

$$\frac{dU}{dt} = \frac{F_{x_{A+P}}}{m} - g\sin\theta + VR - WQ \tag{A.26}$$

$$\frac{dV}{dt} = \frac{F_{y_{A+P}}}{m} - g\cos\theta\sin\phi - UR - WP \tag{A.27}$$

$$\frac{dW}{dt} = \frac{F_{z_{A+P}}}{m} - g\cos\theta\cos\phi + UQ - VP \tag{A.28}$$

$$I_{xx}\frac{dP}{dt} - I_{xz}\frac{dR}{dt} = L_{A+P} + I_{xz}PQ + \left(I_{yy} - I_{zz}\right)RQ \tag{A.29}$$

$$I_{yy}\frac{dQ}{dt} = M_{A+P} + \left(R^2 - P^2\right)I_{xz} + \left(I_{zz} - I_{xx}\right)PR \tag{A.30}$$

$$I_{xx}\frac{dR}{dt} - I_{xz}\frac{dP}{dt} = N_{A+P} + \left(I_{xx} - I_{yy}\right)PQ - I_{xz}QR \tag{A.31}$$

The important aspect of equations of motion is non-linearity from the trigonometry function and coupled terms (e.g. VR, WQ, UR). For example, the nonlinear terms in Eq. A.29 show the coupled motion. If the pitch rate changes in longitudinal motion, it also affects the lateral motion which is roll and yaw. Moreover, the left hand side of Eq. A.29 and Eq. A.31 describes the coupled motion between roll and yaw. They have two differential terms, which are the rate of roll and yaw. Since six DoF equations indicates the dynamic constraints in different direction of the aircraft motion, they allow to describe the dynamics of the aircraft more accurately than the point mass model discussed in Section 1.2.

Using translational three DoF equations of motion from Eq. A.26 to Eq. A.28, the trajectory of the aircraft can be found by transforming the velocity vector in body coordinates to the inertial frame and integrating it. Obviously, the initial condition of position should be given in order to integrate them.

$$\begin{aligned}\begin{bmatrix} dp_N/dt \\ dp_E/dt \\ dp_D/dt \end{bmatrix}_I &= H_B^I \begin{bmatrix} U \\ V \\ W \end{bmatrix}_B \\ &= \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}^T \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}^T \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix}^T \begin{bmatrix} U \\ V \\ W \end{bmatrix}_B \end{aligned} \tag{A.32}$$

In order to transform from the body coordinates frame to the inertial frame, Euler angles are required. However, Euler angles changes with time, which means there is a time derivative. The rate of the Euler angles in the inertial frame can be presented by the angular rates in the body coordinate system. The following equation describes the relationship between the rate of Euler angles and the angular rates. These equations are called kinematic equations.

$$\begin{bmatrix} P \\ Q \\ R \end{bmatrix}_B = \mathcal{L}_I^B \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}_I = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \cos\theta\sin\phi \\ 0 & -\sin\phi & \cos\theta\cos\phi \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}_I \tag{A.33}$$

where $\mathcal{L}_I^B$ is the transformation matrix for the Euler angles rate to the angular rates in the body coordinate system. Unlike the transformation matrices (H_I^B or H_B^I), $\mathcal{L}_I^B$ is not the orthonormal.

A.2 Forces and Moments

To complete equations of motion in the previous section, the computation of the forces and moments should be discussed. The gravity force is already discussed in Section A.1.3. The remaining elements are the aerodynamic and propulsive forces and moments.

A.2.1 Linear Aerodynamic Forces and Moments

The forces and moments acting on the body coordinate system can be presented as follows:

$$\sum F_A = \begin{bmatrix} F_{x_A} \\ F_{y_A} \\ F_{z_A} \end{bmatrix}_B = \begin{bmatrix} \bar{q}SC_X \\ \bar{q}SC_Y \\ \bar{q}SC_Z \end{bmatrix}, \quad \sum M_A = \begin{bmatrix} L_A \\ M_A \\ N_A \end{bmatrix}_B = \begin{bmatrix} \bar{q}SbC_l \\ \bar{q}S\bar{c}C_m \\ \bar{q}SbC_n \end{bmatrix}_B \tag{A.34}$$

where C_X, C_Y, C_Z, C_l, C_m, C_n are non-dimensional coefficients. These coefficients are described in detail below.

S is the reference or wing area; $\bar{c}$ is the mean geometric chord of the wing; b is the wing span; $\bar{q}$ is the dynamic pressure that is $0.5\rho|\overrightarrow{V}|^2$ where ρ is the air density and $\overrightarrow{V}$ is the velocity vector. Since all aerodynamic forces and moments are in the stability coordinate system (see Figure A.3), non-dimensional coefficients mentioned above can be presented as follows:

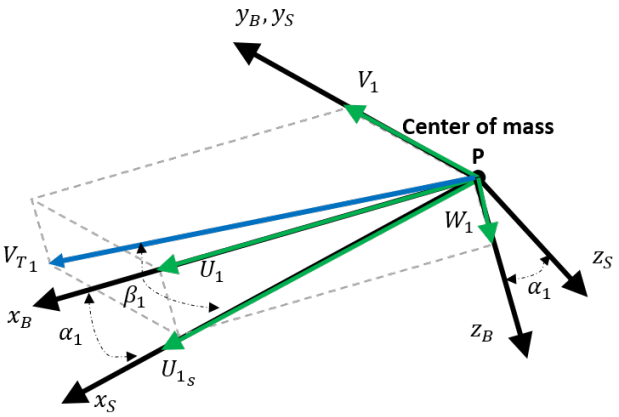


Figure A.3: Body and Stability Coordinate Systems

$$\begin{bmatrix} C_X \\ C_Y \\ C_Z \end{bmatrix}_B = \begin{bmatrix} -[C_D]_S \cos\alpha_1 + [C_L]_S \sin\alpha_1 \\ [C_Y]_S \\ -[C_D]_S \sin\alpha_1 - [C_L]_S \cos\alpha_1 \end{bmatrix}_B, \quad \begin{bmatrix} C_l \\ C_m \\ C_n \end{bmatrix}_B = \begin{bmatrix} [C_l]_S \cos\alpha_1 - [C_n]_S \sin\alpha_1 \\ [C_m]_S \\ [C_l]_S \sin\alpha_1 + [C_n]_S \cos\alpha_1 \end{bmatrix}_B \tag{A.35}$$

where subscript 1 means the steady state, subscript S means the stability coordinate system, C_L is the lift coefficient, C_D is the drag coefficient, C_Y is the side force coefficient, C_l is the rolling moment coefficient, and C_m is the pitching moment coefficient, and C_n is the yawing moment coefficient. The coordinate of Eq. A.35 is transformed in accordance with α_1 from the stability to the body coordinate system. The side force and the pitching

moment coefficients ($C_Y = [C_Y]_S$, $C_m = [C_m]_S$) are identical in the stability and the body coordinate system since the Y axes are aligned. The non-dimensional coefficients are modeled by the component build-up method using perturbed states, see Ref. [85]. Perturbed states are the small deviation from the steady state, $x = X - X_1$ where X is the state (e.g., U,V,W,P,Q,R). Total forces and moment coefficients in the stability coordinate systems are presented as follows:

$$C_L = C_{L_1} + C_{L_\alpha}(\alpha - \alpha_1) + C_{L_q}\frac{\bar{c}(Q - Q_1)}{2U_1} + C_{L_{\dot{\alpha}}}\frac{\dot{\alpha}\bar{c}}{2U_1} + C_{L_u}\frac{(U - U_1)}{U_1} + C_{L_{\delta_E}}(\delta_E - \delta_{E_1}) \tag{A.36}$$

$$C_Y = C_{Y_\beta}(\beta - \beta_1) + C_{Y_p}\frac{(P - P_1)b}{2U_1} + C_{Y_r}\frac{(R - R_1)b}{2U_1} + C_{Y_{\delta_A}}(\delta_A - \delta_{A_1}) + C_{Y_{\delta_R}}(\delta_R - \delta_{R_1}) \tag{A.37}$$

$$C_D = \overline{C_{D0}} + \frac{C_L^2}{\pi ARe} \tag{A.38}$$

$$C_l = C_{l_\beta}(\beta - \beta_1) + C_{l_p}\frac{(P - P_1)b}{2U_1} + C_{l_r}\frac{(R - R_1)b}{2U_1} + C_{l_{\delta_A}}(\delta_A - \delta_{A_1}) + C_{l_{\delta_R}}(\delta_R - \delta_{R_1}) \tag{A.39}$$

$$\begin{aligned} C_m = {} & C_{m_1} + C_{m_\alpha}(\alpha - \alpha_1) + C_{m_q}\frac{(Q - Q_1)\bar{c}}{2U_1} + C_{m_{\dot{\alpha}}}\frac{\dot{\alpha}\bar{c}}{2U_1} + (C_{m_u} + 2C_{m_1})\frac{(U - U_1)}{U_1} \\ & + C_{m_{\delta_E}}(\delta_E - \delta_{E_1}) + \left(C_{m_{T_u}} + 2C_{m_{T_1}}\right)\frac{(U - U_1)}{U_1} + C_{m_{T_\alpha}}(\alpha - \alpha_1) \end{aligned} \tag{A.40}$$

$$C_n = C_{m_\beta}(\beta - \beta_1) + C_{n_p}\frac{(P - P_1)b}{2U_1} + C_{n_r}\frac{(R - R_1)b}{2U_1} + C_{n_{\delta_A}}(\delta_A - \delta_{A_1}) + C_{n_{\delta_R}}(\delta_R - \delta_{R_1}) \tag{A.41}$$

where AR is the aspect ratio of the wing, and e is the Oswald efficiency factor, which is assumed to be 0.88 in this work. In the steady state level wing flight, the following can be considered: $P_1 = Q_1 = R_1 = \beta_1 = \dot{\alpha}_1 = 0$. The stability and control derivatives used from Eq. A.36 to Eq. A.41 can be estimated using two kinds of software: Advanced Aircraft Analysis (AAA) and Athena Vortex Lattice (AVL). AAA has advantages by using vast databases of existing aircraft and airfoils. AVL is a software developed by MIT, and AVL uses an extended vortex lattice method. AVL cannot analyze the geometry with the fuselage unlike AAA. AAA and AVL are used in order to compare stability and control derivatives, so this comparison provides an interpretation of uncertainty. Figure. A.4 shows both geometric models.

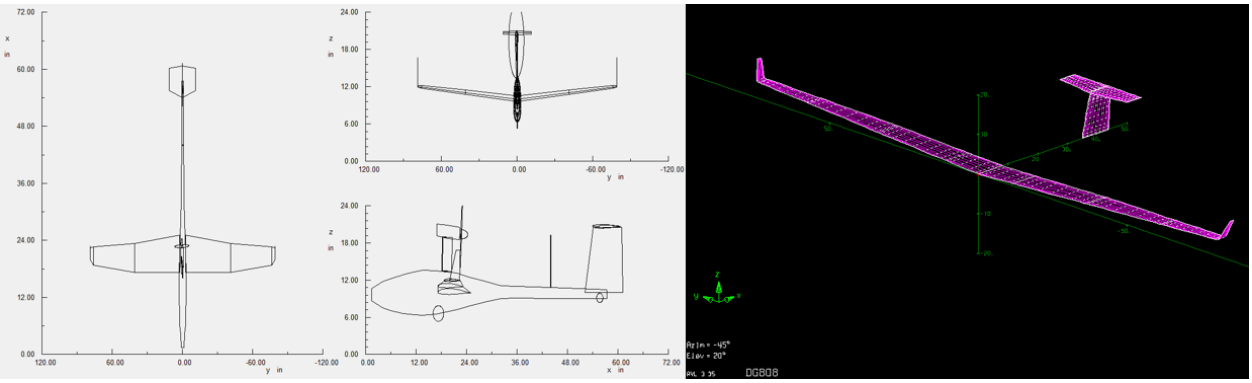


Figure A.4: AAA(left) and AVL(right) software

A.2.2 Propulsive Forces and Moments

Thrust force is considered only in the X axis of the body coordinates. Regarding to the moments, the thrust can affect the pitch moment depending on the location of the engine. Thrust forces in the body coordinate system are presented as follows:

$$\sum F_P = \begin{bmatrix} X_T \\ Y_T \\ Z_T \end{bmatrix} = \begin{bmatrix} X_{T_0} + X_{T_1}\delta_T + X_{T_2}\delta_T^2 \\ 0 \\ 0 \end{bmatrix}, \quad \sum M_P = \begin{bmatrix} L_T \\ M_T \\ N_T \end{bmatrix} = \begin{bmatrix} 0 \\ T \cdot dT \\ 0 \end{bmatrix} \tag{A.42}$$

where X_{T_0}, X_{T_1} and X_{T_2} are the nonlinear thrust model polynomial coefficients; T is the amount of thrust generated by the engine; dT is the distance from the engine to the center of gravity of the aircraft in z_B. The sign of M_T can vary depending on the location of the engine.

A.3 Linear Time Invariant State Space using Perturbed Equations of Motion

Linear Time Invariant (LTI) state space model is the mathematical way to describe the aircraft dynamics by the linearization of nonlinear equations of motion with state and control

vectors. Since LTI state space model represents the aircraft dynamics, it can be used for modal analysis by eigenvalues and eigenvectors. Handling qualities are also identified by the dynamic characteristics (e.g., natural frequency, damping ratio, time to double amplitude). In order to obtain the LTI model of the aircraft, the perturbed equations of motion are derived. Then, the dimensional stability and control derivatives are introduced to present the perturbed equations of motion as LTI state space model.

A.3.1 Perturbed Equations of Motion

In this section, perturbed equations of motion are discussed. Perturbation methods are applied to describe the motion of the aircraft by considering the small changes from the steady state. The sum of perturbed and steady states are called total states. Total states can be presented as follows:

$$
\begin{array}{llll}
U = U_1 + u & V = V_1 + v & W = W_1 + w & P = P_1 + p \\
Q = Q_1 + q & R = R_1 + r & \Psi = \Psi_1 + \psi & \Theta = \Theta_1 + \theta \\
\Phi = \Phi_1 + \phi & F_{A_x} = F_{A_{x_1}} + f_{A_x} & F_{A_y} = F_{A_{y_1}} + f_{A_y} & F_{A_z} = F_{A_{z_1}} + f_{A_z} \\
F_{P_x} = F_{P_{x_1}} + f_{P_x} & F_{P_y} = F_{P_{y_1}} + f_{P_y} & F_{P_z} = F_{P_{z_1}} + f_{P_z} & L_A = L_{A_{x_1}} + l_{A_x} \\
M_A = M_{A_{x_1}} + m_{A_x} & N_A = N_{A_{x_1}} + n_{A_x} & L_P = L_{P_{x_1}} + l_{P_x} & M_P = M_{P_{x_1}} + m_{P_x} \\
N_P = N_{P_{x_1}} + n_{P_x} & & &
\end{array}
$$

The steady states indicate zero acceleration, so all left hand sides from Eq. A.26 to Eq. A.31 become zero. Three typical steady states are recti-linear, symmetrical pull-up, and level turn cases. Perturbed Euler angles are assumed to be very small to begin perturbing equations. The following relationships are used for perturbations:

$$\cos\phi \cong 1, \quad \sin\phi \cong \phi \tag{A.43}$$

$$\cos\theta \cong 1, \quad \sin\theta \cong \theta \tag{A.44}$$

$$\cos\psi \cong 1, \quad \sin\psi \cong \psi \tag{A.45}$$

Due to the small angle assumption and the definition of perturbation, products of perturbed values are negligible. In addition, the rate of change of the Euler angles ($\dot{\phi}$, $\dot{\theta}$, and $\dot{\psi}$) is

assumed to be zero as well. Once all states are perturbed and assumptions are applied, perturbed equations of motion can be obtained as follows: (Details of the derivation are presented in Appendix B.)

$$\frac{du}{dt} = \frac{f_{x_{A+P}}}{m} - g\theta\cos\theta_1 + (V_1 r + vR_1) - (W_1 q + wQ_1) \tag{A.46}$$

$$\frac{dv}{dt} = \frac{f_{y_{A+P}}}{m} + g\left(-\theta\sin\phi_1\sin\theta_1 + \phi\cos\theta_1\cos\phi_1\right) - (uR_1 + U_1 r) + (wP_1 + W_1 p) \tag{A.47}$$

$$\frac{dw}{dt} = \frac{f_{z_{A+P}}}{m} + g\left(-\theta\cos\phi_1\sin\theta_1 - \phi\cos\theta_1\sin\phi_1\right) + (uQ_1 + U_1 q) - (vP_1 + V_1 p) \tag{A.48}$$

$$I_{xx}\frac{dp}{dt} - I_{xz}\frac{dr}{dt} = l_{A+P} + (pQ_1 + P_1 q)\, I_{xz} + (I_{yy} - I_{zz})\,(R_1 q + rQ_1) \tag{A.49}$$

$$I_{yy}\frac{dq}{dt} = m_{A+P} + (2R_1 r - 2P_1 p)\, I_{xz} + (P_1 r + pR_1)\,(I_{xz} - I_{xx}) \tag{A.50}$$

$$I_{xx}\frac{dr}{dt} - I_{xz}\frac{dp}{dt} = n_{A+P} + (P_1 q + pQ_1)\,(I_{xx} - I_{yy}) - (Q_1 r + qR_1)\, I_{xz} \tag{A.51}$$

A.3.2 Perturbed Forces and Moments

To compute the perturbed forces and moments, dimensionless stability and control derivatives are required to utilize the component build-up method, see Ref. [85]. Stability and control derivatives are partial derivatives of forces and moments coefficients with respect to states and controls. Because forces and moments are aligned with the stability coordinate system, they are developed in stability axes. Stability and control derivatives are dimensionless values, and this feature enables fair comparisons between various vehicles regardless of the size of aircraft. Stability and control derivatives can obtained from AAA or AVL as mentioned in Section A.2.1. However, one more hurdle still exists: units of partial derivatives are different depending on the states. For example, when an aerodynamic force in x_S direction (F_{A_x}) is taken the partial derivative with respect to u, its unit is ft/sec. If we take the partial derivative of F_{A_x} respect to the time derivative of angle of attack ($\dot{\alpha}$), the

units of $\dot{\alpha}$ is rad/sec. In order to make the partial derivative respect to u, trim speed, U_1 was multiplied and divided. Examples follows:

$$\frac{\partial F_{A_x}}{\partial u}u = \frac{\partial F_{A_x}}{\partial u}u\frac{U_1}{U_1} = \frac{\partial F_{A_x}}{\partial \frac{u}{U_1}}\frac{u}{U_1} \tag{A.52}$$

Similarly, the partial derivatives respect to the rate of angle of attack can be treated as follows:

$$\frac{\partial F_{A_x}}{\partial \dot{\alpha}}\dot{\alpha} = \frac{\partial F_{A_x}}{\partial \dot{\alpha}}\dot{\alpha}\frac{\frac{\bar{c}}{2U_1}}{\frac{\bar{c}}{2U_1}} = \frac{\partial F_{A_x}}{\partial \frac{\dot{\alpha}\bar{c}}{2U_1}}\frac{\dot{\alpha}\bar{c}}{2U_1} \tag{A.53}$$

Regarding to the partial derivative respect to the pitch rate (q), the same approach can be applied in Eq. A.53. For the lateral states ($\dot{\beta}$, p, and r), $b/(2U_1)$ is multiplied and divided to make the partial derivatives non-dimensional. The reason that longitudinal and lateral-directional equations use different reference length ($\bar{c}$ and b) is because a moment arm is different due to the axis of the rotating motion. Taylor series of the partial derivative provides the model of perturbed forces and moments:

$$f_{A_x} = \frac{\partial F_{A_x}}{\partial \frac{u}{U_1}}\frac{u}{U_1} + \frac{\partial F_{A_x}}{\partial \alpha}\alpha + \frac{\partial F_{A_x}}{\partial \frac{\dot{\alpha}\bar{c}}{2U_1}}\frac{\dot{\alpha}\bar{c}}{2U_1} + \frac{\partial F_{A_x}}{\partial \frac{q\bar{c}}{2U_1}}\frac{q\bar{c}}{2U_1} + \frac{\partial F_{A_x}}{\partial \delta_E}\delta_E \tag{A.54}$$

$$f_{A_y} = \frac{\partial F_{A_y}}{\partial \beta}\beta + \frac{\partial F_{A_y}}{\partial \frac{\dot{\beta}b}{2U_1}} + \frac{\partial F_{A_y}}{\partial \frac{pb}{2U_1}}\frac{pb}{2U_1} + \frac{\partial F_{A_y}}{\partial \frac{rb}{2U_1}}\frac{rb}{2U_1} + \frac{\partial F_{A_y}}{\partial \delta_A}\delta_A + \frac{\partial F_{A_y}}{\partial \delta_R}\delta_R \tag{A.55}$$

$$f_{A_z} = \frac{\partial F_{A_z}}{\partial \frac{u}{U_1}}\frac{u}{U_1} + \frac{\partial F_{A_z}}{\partial \alpha}\alpha + \frac{\partial F_{A_z}}{\partial \frac{\dot{\alpha}\bar{c}}{2U_1}}\frac{\dot{\alpha}}{2U_1} + \frac{\partial F_{A_z}}{\partial \frac{q\bar{c}}{2U_1}}\frac{q\bar{c}}{2U_1} + \frac{\partial F_{A_z}}{\partial \delta_E}\delta_E \tag{A.56}$$

$$l_A = \frac{\partial L_A}{\partial \beta}\beta + \frac{\partial L_A}{\partial \frac{\dot{\beta}b}{2U_1}}\frac{\dot{\beta}b}{2U_1} + \frac{\partial L_A}{\partial \frac{pb}{2U_1}}\frac{pb}{2U_1} + \frac{\partial L_A}{\partial \frac{rb}{2U_1}}\frac{rb}{2U_1} + \frac{\partial L_A}{\partial \delta_A}\delta_A + \frac{\partial L_A}{\partial \delta_R}\delta_R \tag{A.57}$$

$$m_A = \frac{\partial M_A}{\partial \frac{u}{U_1}}\frac{u}{U_1} + \frac{\partial M_A}{\partial \alpha}\alpha + \frac{\partial M_A}{\partial \frac{\dot{\alpha}\bar{c}}{2U_1}}\frac{\dot{\alpha}\bar{c}}{2U_1} + \frac{\partial M_A}{\partial \frac{q\bar{c}}{2U_1}}\frac{q\bar{c}}{2U_1} + \frac{\partial M_A}{\partial \delta_E}\delta_E \tag{A.58}$$

$$n_A = \frac{\partial N_A}{\partial \beta}\beta + \frac{\partial N_A}{\partial \frac{\dot{\beta}b}{2U_1}}\frac{\dot{\beta}b}{2U_1} + \frac{\partial N_A}{\partial \frac{pb}{2U_1}}\frac{pb}{2U_1} + \frac{\partial N_A}{\partial \frac{rb}{2U_1}}\frac{rb}{2U_1} + \frac{\partial N_A}{\partial \delta_A}\delta_A + \frac{\partial N_A}{\partial \delta_R}\delta_R \tag{A.59}$$

$$f_{P_x} = \frac{\partial F_{P_x}}{\partial \frac{u}{U_1}} + \frac{\partial F_{P_x}}{\partial \alpha}\alpha \tag{A.60}$$

$$f_{P_y} = \frac{\partial F_{P_y}}{\partial \frac{u}{U_1}} + \frac{\partial F_{P_y}}{\partial \alpha}\alpha \tag{A.61}$$

$$f_{P_z} = \frac{\partial F_{P_z}}{\partial \frac{u}{U_1}}\frac{u}{U_1} + \frac{\partial F_{P_z}}{\partial \alpha}\alpha \tag{A.62}$$

$$l_P = \frac{\partial L_P}{\partial \beta}\beta \tag{A.63}$$

$$m_P = \frac{\partial M_P}{\partial \frac{u}{U_1}}\frac{u}{U_1} + \frac{\partial M_P}{\partial \alpha}\alpha \tag{A.64}$$

$$n_P = \frac{\partial N_P}{\partial \beta}\beta \tag{A.65}$$

Note that all states are in **perturbations** from the steady state. Using Eq. A.35 and the chain rule of derivatives, the following equations show perturbed forces and moments using stability and control derivatives developed above:

$$\begin{aligned}\frac{f_{A+P_x}}{\bar{q}_1 S} = &-\left(C_{D_u} + 2C_{D_1} - C_{T_{x_u}} - 2C_{T_{x_1}}\right)\frac{u}{U_1} + \left(-C_{D_\alpha} + C_{L_1}\right)\alpha - C_{D_{\dot{\alpha}}}\frac{\dot{\alpha}\bar{c}}{2U_1} - C_{D_q}\frac{q\bar{c}}{2U_1} \\ &- C_{D_{\delta_e}}\delta_e\end{aligned} \tag{A.66}$$

$$\frac{f_{A+P_y}}{\bar{q}_1 S} = C_{y_\beta}\beta + C_{y_{\dot{\beta}}}\frac{\dot{\beta}b}{2U_1} + C_{y_p}\frac{pb}{2U_1} + C_{y_r}\frac{rb}{2U_1} + C_{y_{\delta_a}}\delta_a + C_{y_{\delta_r}}\delta_r \tag{A.67}$$

$$\frac{f_{A+P_z}}{\bar{q}_1 S} = -\left(C_{L_u} + 2C_{L_1}\right)\frac{u}{U_1} + \left(-C_{L_\alpha} - C_{D_1}\right)\alpha - C_{L_{\dot{\alpha}}}\frac{\dot{\alpha}\bar{c}}{2U_1} - C_{L_q}\frac{q\bar{c}}{2U_1} - C_{L_{\delta_e}}\delta_e \tag{A.68}$$

$$\frac{l_{A+P}}{\bar{q}_1 S b} = C_{l_\beta}\beta + C_{l_{\dot{\beta}}}\frac{\dot{\beta}b}{2U_1} + C_{l_p}\frac{pb}{2U_1} + C_{l_r}\frac{rb}{2U_1} + C_{l_{\delta_a}}\delta_a + C_{l_{\delta_r}}\delta_r \tag{A.69}$$

$$\begin{aligned}\frac{m_{A+P}}{\bar{q}_1 S\bar{c}} = &\left(C_{m_u} + 2C_{m_1} + C_{m_{T_u}} + 2C_{m_{T_1}}\right)\frac{u}{U_1} + \left(C_{m_\alpha} + C_{m_{T_\alpha}}\right)\alpha + C_{m_{\dot{\alpha}}}\frac{\dot{\alpha}\bar{c}}{2U_1} + C_{m_q}\frac{q\bar{c}}{2U_1} \\ &+ C_{m_{\delta_e}}\delta_e\end{aligned} \tag{A.70}$$

$$\frac{n_{A+P}}{\bar{q}_1 S b} = \left(C_{n_\beta} + C_{n_{T_\beta}}\right)\beta + C_{n_{\dot{\beta}}}\frac{\dot{\beta}b}{2U_1} + C_{n_p}\frac{pb}{2U_1} + C_{n_r}\frac{rb}{2U_1} + C_{n_{\delta_a}}\delta_a + C_{n_{\delta_r}}\delta_r \tag{A.71}$$

C_{D_q} and $C_{D_{\dot{\alpha}}}$ are assumed to be negligible.

A.3.3 Linear Time Invariant Model

The Linear Time Invariant (LTI) model is essential to analyze dynamics of the aircraft and design controllers. To obtain linearized equations of motion, perturbed equations should be recalled again. In addition, assumptions for the linearization are presented as follows for the

longitudinal and lateral-directional equations of motion:

$$p = \dot{\phi}, \quad q = \dot{\theta}, \quad r = \dot{\psi} \tag{A.72}$$

$$v = U_1\beta, \quad w = U_1\alpha \tag{A.73}$$

Because perturbed equations of motion are developed in stability axes, the moment of inertia should be transformed into stability axes from the body frame using the following equation.

$$\begin{bmatrix} I_{xx_S} \\ I_{zz_S} \\ I_{xz_S} \end{bmatrix} = \begin{bmatrix} \cos^2\alpha_1 & \sin^2\alpha_1 & -\sin 2\alpha_1 \\ \sin^2\alpha_1 & \cos^2\alpha_1 & \sin 2\alpha_1 \\ 0.5\sin 2\alpha_1 & -0.5\sin 2\alpha_1 & \cos 2\alpha_1 \end{bmatrix} \begin{bmatrix} I_{xx_B} \\ I_{zz_B} \\ I_{xz_B} \end{bmatrix} \tag{A.74}$$

I_{yy_B} and I_{yy_S} are identical since the rotation axis is about the y axis. For a detailed explanation, the x direction translational equation is considered for an example. Using Eq. A.46 and Eq. A.66, the following equation is presented:

$$\begin{aligned} \frac{du}{dt} = \frac{\bar{q}_1 S}{m} \Bigg(& -\left(C_{D_u} + 2C_{D_1} - C_{T_{x_u}} - 2C_{T_{x_1}}\right)\frac{u}{U_1} \\ & + \left(-C_{D_\alpha} + C_{L_1}\right)\alpha - C_{D_{\delta_e}}\delta_e \Bigg) - g\theta\cos\theta_1 + \left(V_1 r + vR_1\right) - \left(W_1 q + wQ_1\right) \end{aligned} \tag{A.75}$$

V_1, W_1, Q_1, and R_1 are zero at a steady state because the recti-linear steady state is assumed. Therefore, Eq. A.75 can be simplified as follows:

$$\frac{du}{dt} = \frac{\bar{q}_1 S}{m}\left(-\left(C_{D_u} + 2C_{D_1} - C_{T_{x_u}} - 2C_{T_{x_1}}\right)\frac{u}{U_1} + \left(-C_{D_\alpha} + C_{L_1}\right)\alpha - C_{D_{\delta_e}}\delta_e\right) - g\theta\cos\theta_1 \tag{A.76}$$

Terms in right hand side are then expanded to define newly dimensional stability and control derivatives:

$$\frac{du}{dt} = \left(\frac{-\bar{q}_1 S\left(C_{D_u} + 2C_{D_1}\right)}{mU_1}u + \frac{\bar{q}_1 S\left(C_{T_{x_u}} + 2C_{T_{x_1}}\right)}{mU_1}u + \frac{-\bar{q}_1 S\left(C_{D_\alpha} - C_{L_1}\right)}{m}\alpha + \frac{-\bar{q}_1 S C_{D_{\delta_e}}}{m}\delta_e\right) - g\theta\cos\theta_1 \tag{A.77}$$

Eq. A.77 can be rewritten again using dimensional stability and control derivatives. For example:

$$X_u = \frac{-\bar{q}_1 S\left(C_{D_u} + 2C_{D_1}\right)}{mU_1} \tag{A.78}$$

Finally, the following equation can be obtained:

$$\dot{u} = (X_u + X_{T_u})u + X_\alpha \alpha + X_{\delta_e}\delta_e - g\cos\theta_1\theta \tag{A.79}$$

Similarly, other equations can be rewritten by using dimensional stability and control derivatives. The resulting perturbed equations of motion using dimensional stability and control derivatives are shown below:

$$\dot{\theta} = q \tag{A.80}$$

$$\dot{u} = -g\cos\theta_1\theta + (X_u + X_{T_u})u + X_\alpha\alpha + X_{\delta_e}\delta_e + X_{\delta_t}\delta_t \tag{A.81}$$

$$(U_1 - Z_{\dot{\alpha}})\dot{\alpha} = -g\sin\theta_1\theta + Z_u u + Z_\alpha\alpha + (U_1 + Z_q)q + Z_{\delta_e}\delta_e + Z_{\delta_t}\delta_t \tag{A.82}$$

$$\dot{q} - M_{\dot{\alpha}}\dot{\alpha} = (M_u + M_{T_u})u + (M_\alpha + M_{T_\alpha})\alpha + M_q q + M_{\delta_e}\delta_e + M_{\delta_t}\delta_t \tag{A.83}$$

$$\dot{\phi} = p \tag{A.84}$$

$$\dot{\psi} = r \tag{A.85}$$

$$U_1\dot{\beta} = g\cos\theta_1 + Y_\beta\beta + Y_p p + (Y_r - U_1)r + Y_{\delta_a}\delta_a + Y_{\delta_r}\delta_r \tag{A.86}$$

$$\dot{p} - \bar{A}_1\dot{r} = L_\beta\beta + L_p p + L_r r + L_{\delta_a}\delta_a + L_{\delta_r}\delta_r \tag{A.87}$$

$$\dot{r} - \bar{B}_1\dot{p} = (N_\beta + N_{T_\beta})\beta + N_p p + N_r r + N_{\delta_a}\delta_a + N_{\delta_r}\delta_r \tag{A.88}$$

where $\bar{A}_1 = \frac{I_{xz}}{I_{xx}}$ and $\bar{B}_1 = \frac{I_{xz}}{I_{zz}}$. In addition, X_{δ_t}, Z_{δ_t}, and M_{δ_t} are added for the throttle control. $C_{D_{\delta_t}}$, $C_{L_{\delta_t}}$, and $C_{m_{\delta_t}}$ are computed from AAA software. Then, the following dimensional control derivatives are applied:

$$X_{\delta_t} = -\frac{\bar{q}_1 S C_{D_{\delta_t}}}{m}, \quad Z_{\delta_t} = -\frac{\bar{q}_1 S C_{L_{\delta_t}}}{m}, \quad M_{\delta_t} = -\frac{\bar{q}_1 S\bar{c} C_{m_{\delta_t}}}{m} \tag{A.89}$$

Eq. A.80 through Eq. A.83 are in longitudinal axes, while Eq. A.84 through Eq. A.88 are

in the lateral-directional axis. With these equations, the linear state space can be formed, see Eq. A.90:

$$\dot{\overrightarrow{x}} = A\overrightarrow{x} + Bu \tag{A.90}$$

Before the state space is formed, states and controls for each axis should be defined using the following equation:

$$\overrightarrow{x}_{long} = \begin{bmatrix} \theta \\ u \\ \alpha \\ q \end{bmatrix}, \quad \overrightarrow{u}_{long} = \begin{bmatrix} \delta_e \\ \delta_t \end{bmatrix}, \quad \overrightarrow{x}_{lat} = \begin{bmatrix} \phi \\ \psi \\ \beta \\ p \\ r \end{bmatrix}, \quad \overrightarrow{u}_{lat} = \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix} \tag{A.91}$$

For the longitudinal axis, the following equations are expressed in a matrix form:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & U_1 - Z_{\dot{\alpha}} & 0 \\ 0 & 0 & -M_{\dot{\alpha}} & 1 \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ \dot{u} \\ \dot{\alpha} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ -g\cos\theta_1 & X_u + X_{T_u} & X_\alpha & 0 \\ -g\sin\theta_1 & Z_u & Z_\alpha & U_1 + Z_q \\ 0 & M_u + M_{T_u} & M_\alpha + M_{T_\alpha} & M_q \end{bmatrix} \begin{bmatrix} \theta \\ u \\ \alpha \\ q \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ X_{\delta_e} & X_{\delta_t} \\ Z_{\delta_e} & Z_{\delta_t} \\ M_{\delta_e} & M_{\delta_t} \end{bmatrix} \begin{bmatrix} \delta_e \\ \delta_t \end{bmatrix} \tag{A.92}$$

The following equation is always valid under the assumption that the determinant of K exists, so that the inverse is possible:

$$K\dot{\overrightarrow{x}} = A'\overrightarrow{x} + B'u \tag{A.93}$$

Finally, A_{long} and B_{long} are obtained:

$$A_{long} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & U_1 - Z_{\dot{\alpha}} & 0 \\ 0 & 0 & -M_{\dot{\alpha}} & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 & 0 & 0 & 1 \\ -g\cos\theta_1 & X_u + X_{T_u} & X_\alpha & 0 \\ -g\sin\theta_1 & Z_u & Z_\alpha & U_1 + Z_q \\ 0 & M_u + M_{T_u} & M_\alpha + M_{T_\alpha} & M_q \end{bmatrix} \tag{A.94}$$

$$B_{long} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & U_1 - Z_{\dot{\alpha}} & 0 \\ 0 & 0 & -M_{\dot{\alpha}} & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 & 0 \\ X_{\delta_e} & X_{\delta_t} \\ Z_{\delta_e} & Z_{\delta_t} \\ M_{\delta_e} & M_{\delta_t} \end{bmatrix} \tag{A.95}$$

For the lateral-directional axis, more algebra was required since derivatives of roll and yaw moments are coupled. Eq. A.87 and Eq. A.88 can be combined as follows:

$$\begin{aligned} (1 - \bar{A}_1\bar{B}_1)\dot{p} &= \left(\bar{A}_1(N_\beta + N_{T_\beta}) + L_\beta\right)\beta \\ &\quad + (\bar{A}_1 N_p + L_p)p + (\bar{A}_1 N_r + L_r)r + (\bar{A}_1 N_{\delta_a} + L_{\delta_a})\delta_a + (\bar{A}_1 N_{\delta_r} + L_{\delta_r})\delta_r \end{aligned} \tag{A.96}$$

$$\begin{aligned} (1 - \bar{A}_1\bar{B}_1)\dot{r} &= (\bar{B}_1 L_\beta + N_\beta + N_{T_\beta})\beta \\ &\quad + (\bar{B}_1 L_p + N_p)p + (\bar{B}_1 L_r + N_r)r + (\bar{B}_1 L_{\delta_a} + N_{\delta_a})\delta_a + (\bar{B}_1 L_{\delta_r} + N_{\delta_r})\delta_r \end{aligned} \tag{A.97}$$

Now, the equations can be expressed in matrix form as follows:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & U_1 & 0 & 0 \\ 0 & 0 & 0 & 1 - \bar{A}_1\bar{B}_1 & 0 \\ 0 & 0 & 0 & 0 & 1 - \bar{A}_1\bar{B}_1 \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\psi} \\ \dot{\beta} \\ \dot{p} \\ \dot{r} \end{bmatrix} =$$

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ g\cos\theta_1 & 0 & Y_\beta & Y_p & Y_r - U_1 \\ 0 & 0 & \bar{A}_1(N_\beta + N_{T_\beta}) + L_\beta & \bar{A}_1 N_p + L_p & \bar{A}_1 N_r + L_r \\ 0 & 0 & \bar{B}_1 L_\beta + N_\beta + N_{T_\beta} & \bar{B}_1 L_p + N_p & \bar{B}_1 L_r + N_r \end{bmatrix} \begin{bmatrix} \phi \\ \psi \\ \beta \\ p \\ r \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ Y_{\delta_a} & Y_{\delta_r} \\ \bar{A}_1 N_{\delta_a} + L_{\delta_a} & \bar{A}_1 N_{\delta_r} + L_{\delta_r} \\ \bar{B}_1 L_{\delta_a} + N_{\delta_a} & \bar{B}_1 L_{\delta_r} + N_{\delta_r} \end{bmatrix} \begin{bmatrix} \delta_a \\ \delta_r \end{bmatrix} \tag{A.98}$$

Using identical assumptions in the longitudinal axis (Eq. A.93), A_{lat} and B_{lat} can be computed as the following equation:

$$A_{lat} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & U_1 & 0 & 0 \\ 0 & 0 & 0 & 1 - \bar{A}_1\bar{B}_1 & 0 \\ 0 & 0 & 0 & 0 & 1 - \bar{A}_1\bar{B}_1 \end{bmatrix}^{-1} \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ g\cos\theta_1 & 0 & Y_\beta & Y_p & Y_r - U_1 \\ 0 & 0 & \bar{A}_1(N_\beta + N_{T_\beta}) + L_\beta & \bar{A}_1 N_p + L_p & \bar{A}_1 N_r + L_r \\ 0 & 0 & \bar{B}_1 L_\beta + N_\beta + N_{T_\beta} & \bar{B}_1 L_p + N_p & \bar{B}_1 L_r + N_r \end{bmatrix} \tag{A.99}$$

$$B_{lat} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & U_1 & 0 & 0 \\ 0 & 0 & 0 & 1 - \bar{A}_1\bar{B}_1 & 0 \\ 0 & 0 & 0 & 0 & 1 - \bar{A}_1\bar{B}_1 \end{bmatrix}^{-1} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ Y_{\delta_a} & Y_{\delta_r} \\ \bar{A}_1 N_{\delta_a} + L_{\delta_a} & \bar{A}_1 N_{\delta_r} + L_{\delta_r} \\ \bar{B}_1 L_{\delta_a} + N_{\delta_a} & \bar{B}_1 L_{\delta_r} + N_{\delta_r} \end{bmatrix} \tag{A.100}$$

Finally, the linear dynamics of the system for longitudinal and lateral-directional motion are completed. With these models, the modal analysis can be performed and the handling qualities can be observed. The eigenvalues of these models provides the dynamic modes of the system: short period, phugoid, spiral, roll, and dutch roll modes. Moreover, the eigenvectors indicate the impact against each state. Lastly, these models are the foundation for the control design. All numerical results are presented in Appendix D.

Appendix B

Derivation of the Perturbed equations of motion

Assumptions

- Perturbed angles are small.
- The results of multiplication of perturbed values are small enough to be negligible.

** Nonlinear terms were highlighted in yellow.

Translational Equations of Motion

Acceleration in x_B direction.

$$\frac{dU}{dt} = \frac{F_{A+P_x}}{m} - g\sin\theta + VR - WQ \tag{B.1}$$

@Steady state,

$$\frac{dU_1}{dt} = \frac{F_{A+P_{x1}}}{m} - g\sin\theta_1 + V_1R_1 - W_1Q_1 = 0 \tag{B.2}$$

Substitute the total values as the addition of steady state value and perturbed value.

$$\frac{d\left(U_1+u\right)}{dt}=\frac{F_{A+P_{x1}}}{m}+\frac{f_{A+P_x}}{m}-g\sin\left(\theta_1+\theta\right)+\left\{\left(V_1+v\right)\left(R_1+R\right)-\left(W_1+w\right)\left(Q_1+q\right)\right\} \tag{B.3}$$

Expand the equation.

$$\begin{aligned}\frac{dU_1}{dt}+\frac{du}{dt}=\frac{F_{A+P_{x1}}}{m}+\frac{f_{A+P_x}}{m}&-g\left(\sin\theta_1\cos\theta+\cos\theta_1\sin\theta\right)\\&+\left\{V_1R_1+V_1r+vR_1+vr-\left(W_1Q_1+W_1q+wQ_1+wq\right)\right\}\end{aligned} \tag{B.4}$$

Apply the small angle assumption. ($\cos\theta \fallingdotseq 1$, $\sin\theta \fallingdotseq \theta$)

$$\begin{aligned}\frac{dU_1}{dt}+\frac{du}{dt}=\frac{F_{A+P_{x1}}}{m}+\frac{f_{A+P_x}}{m}&-g\left(\sin\theta_1\cdot 1+\cos\theta_1\cdot\theta\right)\\&+\left\{V_1R_1+V_1r+vR_1+vr-\left(W_1Q_1+W_1q+wQ_1+wq\right)\right\}\end{aligned} \tag{B.5}$$

$$\begin{aligned}\frac{dU_1}{dt}+\frac{du}{dt}=\frac{F_{A+P_{x1}}}{m}+\frac{f_{A+P_x}}{m}&-g\sin\theta_1\ -g\theta\cos\theta_1\\&+\left\{V_1R_1+V_1r+vR_1+vr-\left(W_1Q_1+W_1q+wQ_1+wq\right)\right\}\end{aligned} \tag{B.6}$$

Substitute the steady state.

$$\begin{aligned}\frac{F_{A+P_{x1}}}{m}-g\sin\theta_1+V_1R_1-W_1Q_1+\frac{du}{dt}=\frac{F_{A+P_{x1}}}{m}+\frac{f_{A+P_x}}{m}&-g\sin\theta_1\ -g\theta\cos\theta_1\\&+\left\{V_1R_1+V_1r+vR_1+vr-\left(W_1Q_1+W_1q+wQ_1+wq\right)\right\}\end{aligned} \tag{B.7}$$

Cancel out the identical terms on the left hand side and right hand side.

$$\frac{du}{dt} = \frac{f_{A+P_x}}{m} - g\theta\cos\theta_1 + \left\{V_1 r + vR_1 + vr - \left(W_1 q + wQ_1 + wq\right)\right\} \tag{B.8}$$

Apply the assumption regarding to multiplication of perturbed values.

$$\boxed{\frac{du}{dt} = \frac{f_{A+P_x}}{m} - g\theta\cos\theta_1 + (V_1 r + vR_1 - W_1 q - wQ_1)} \tag{B.9}$$

Acceleration in y_B direction.

$$\frac{dV}{dt} = \frac{F_{A+P_y}}{m} + g\cos\theta\sin\phi - UR + WP \tag{B.10}$$

@Steady state

$$\frac{dV_1}{dt} = \frac{F_{A+P_{y1}}}{m} + g\cos\theta_1\sin\phi_1 - U_1R_1 + W_1P_1 \tag{B.11}$$

Substitute the total values as the addition of steady state value and perturbed value.

$$\frac{d\left(V_1+v\right)}{dt} = \frac{F_{A+P_{y1}}}{m} + \frac{f_{A+P}}{m} + g\cos\left(\theta_1+\theta\right)\sin\left(\phi_1+\phi\right) - \left(U_1+u\right)\left(R_1+r\right) + \left(W_1+w\right)\left(P_1+p\right) \tag{B.12}$$

Expand the equation.

$$\begin{aligned}\frac{dV_1}{dt} + \frac{dv}{dt} = {} & \frac{F_{A+P_{y1}}}{m} + \frac{f_{A+P}}{m} + g\left(\cos\theta_1\cos\theta - \sin\theta_1\sin\theta\right)\left(\sin\phi_1\cos\phi + \cos\phi_1\sin\phi\right) \\ & - \left(U_1R_1 + U_1r + uR_1 + ur\right) + \left(W_1P_1 + W_1p + P_1w + wp\right)\end{aligned} \tag{B.13}$$

Apply the small angle assumption. ($\cos\theta \fallingdotseq 1$, $\sin\theta \fallingdotseq \theta$)

$$\frac{dV_1}{dt} + \frac{dv}{dt} = \frac{F_{A+P_{y_1}}}{m} + \frac{f_{A+P}}{m} + g\left(\cos\theta_1 \cdot 1 - \sin\theta_1 \cdot \theta\right)\left(\sin\phi_1 \cdot 1 + \cos\phi_1 \cdot \phi\right) - \left(U_1R_1 + U_1r + uR_1 + ur\right) + \left(W_1P_1 + W_1p + P_1w + wp\right) \tag{B.14}$$

$$\frac{dV_1}{dt} + \frac{dv}{dt} = \frac{F_{A+P_{y_1}}}{m} + \frac{f_{A+P}}{m} + g\left(\cos\theta_1 - \theta\sin\theta_1\right)\left(\sin\phi_1 + \phi\cos\phi_1\right) - \left(U_1R_1 + U_1r + uR_1 + ur\right) + \left(W_1P_1 + W_1p + P_1w + wp\right) \tag{B.15}$$

$$\frac{dV_1}{dt} + \frac{dv}{dt} = \frac{F_{A+P_{y_1}}}{m} + \frac{f_{A+P}}{m} + g\left(\cos\theta_1\sin\phi_1 + \phi\cos\theta_1\cos\phi_1 - \theta\sin\theta_1\sin\phi_1 - \theta\phi\sin\theta_1\cos\phi_1\right) - \left(U_1R_1 + U_1r + uR_1 + ur\right) + \left(W_1P_1 + W_1p + P_1w + wp\right) \tag{B.16}$$

Substitute the steady state.

$$\frac{F_{A+P_{y_1}}}{m} + g\cos\theta_1\sin\phi_1 - U_1R_1 + W_1P_1 + \frac{dv}{dt} = \frac{F_{A+P_{y_1}}}{m} + \frac{f_{A+P}}{m} + g\left(\cos\theta_1\sin\phi_1 + \phi\cos\theta_1\cos\phi_1 - \theta\sin\theta_1\sin\phi_1 - \theta\phi\sin\theta_1\cos\phi_1\right) - \left(U_1R_1 + U_1r + uR_1 + ur\right) + \left(W_1P_1 + W_1p + P_1w + wp\right) \tag{B.17}$$

Cancel out the identical terms on the left hand side and right hand side.

$$\frac{dv}{dt} = \frac{f_{A+P}}{m} + g\left(\phi\cos\theta_1\cos\phi_1 - \theta\sin\theta_1\sin\phi_1 - \theta\phi\sin\theta_1\cos\phi_1\right) - \left(U_1r + uR_1 + ur\right) + \left(W_1p + P_1w + wp\right) \tag{B.18}$$

Apply the assumption regarding to multiplication of perturbed values.

$$\boxed{\frac{dv}{dt} = \frac{f_{A+P}}{m} + g\left(\phi\cos\theta_1\cos\phi_1 - \theta\sin\theta_1\sin\phi_1\right) - U_1r - uR_1 + W_1p + P_1w} \tag{B.19}$$

Acceleration in z_B direction.

$$\frac{dW}{dt} = \frac{F_{A+P_z}}{m} + g\cos\theta\cos\phi + UQ - VP \tag{B.20}$$

@Steady state

$$\frac{dW_1}{dt} = \frac{F_{A+P_{z1}}}{m} + g\cos\theta_1\cos\phi_1 + U_1Q_1 - V_1P_1 \tag{B.21}$$

Substitute the total values as the addition of steady state value and perturbed value.

$$\frac{d\left(W_1+w\right)}{dt} = \frac{F_{A+P_{z1}}}{m} + \frac{f_{A+P_z}}{m} + g\cos\left(\theta_1+\theta\right)\cos\left(\phi_1+\phi\right) + \left(U_1+u\right)\left(Q_1+q\right) - \left(V_1+v\right)\left(P_1+p\right) \tag{B.22}$$

Expand the equation.

$$\begin{aligned}\frac{dW_1}{dt} + \frac{dw}{dt} = \frac{F_{A+P_{z1}}}{m} + \frac{f_{A+P_z}}{m} &+ g\cos\left(\cos\theta_1\cos\theta - \sin\theta_1\sin\theta\right)\left(\cos\phi_1\cos\phi - \sin\phi_1\sin\phi\right) \\ &+ \left(U_1Q_1 + U_1q + uQ_1 + uq\right) - \left(V_1P_1 + V_1p + vP_1 + vp\right)\end{aligned} \tag{B.23}$$

Apply the small angle assumption. ($\cos\theta \fallingdotseq 1$, $\sin\theta \fallingdotseq \theta$)

$$\begin{aligned}\frac{dW_1}{dt} + \frac{dw}{dt} = \frac{F_{A+P_{z1}}}{m} + \frac{f_{A+P_z}}{m} &+ g\cos\left(\cos\theta_1\cdot 1 - \sin\theta_1\cdot\theta\right)\left(\cos\phi_1\cdot 1 - \sin\phi_1\cdot\phi\right) \\ &+ \left(U_1Q_1 + U_1q + uQ_1 + uq\right) - \left(V_1P_1 + V_1p + vP_1 + vp\right)\end{aligned} \tag{B.24}$$

$$\frac{dW_1}{dt} + \frac{dw}{dt} = \frac{F_{A+P_{z1}}}{m} + \frac{f_{A+P_z}}{m} + g\cos\left(\cos\theta_1 - \theta\sin\theta_1\right)\left(\cos\phi_1 - \phi\sin\phi_1\right)$$

$$+ (U_1 Q_1 + U_1 q + u Q_1 + u q) - (V_1 P_1 + V_1 p + v P_1 + v p) \tag{B.25}$$

$$\frac{dW_1}{dt} + \frac{dw}{dt} = \frac{F_{A+P_{z1}}}{m} + \frac{f_{A+P_z}}{m} + g\cos\left(\cos\theta_1\cos\phi_1 - \phi\cos\theta_1\sin\phi_1 - \theta\sin\theta_1\cos\phi_1 + \theta\phi\sin\theta_1\sin\phi_1\right)$$
$$+ (U_1 Q_1 + U_1 q + u Q_1 + u q) - (V_1 P_1 + V_1 p + v P_1 + v p) \tag{B.26}$$

Cancel out the steady state terms.

$$\frac{dw}{dt} = \frac{f_{A+P_z}}{m} + g\cos\left(-\phi\cos\theta_1\sin\phi_1 - \theta\sin\theta_1\cos\phi_1 + \theta\phi\sin\theta_1\sin\phi_1\right)$$
$$+ \left(U_1 q + u Q_1 + uq\right) - \left(V_1 p + v P_1 + vp\right) \tag{B.27}$$

Apply the assumption regarding to multiplication of perturbed values.

$$\boxed{\frac{dw}{dt} = \frac{f_{A+P_z}}{m} + g\cos\left(-\phi\cos\theta_1\sin\phi_1 - \theta\sin\theta_1\cos\phi_1\right) + \left(U_1 q + u Q_1\right) - \left(V_1 p + v P_1\right)} \tag{B.28}$$

Rotational Equations of Motion

Moment in x_B direction.

$$I_{xx}\frac{dP}{dt} - I_{xz}\frac{dR}{dt} = L_{A+P} + PQI_{xz} + \left(I_{yy} - I_{zz}\right)RQ \tag{B.29}$$

@Steady state,

$$I_{xx}\frac{dP_1}{dt} - I_{xz}\frac{dR_1}{dt} = L_{A+P_1} + P_1 Q_1 I_{xz} + \left(I_{yy} - I_{zz}\right)R_1 Q_1 \tag{B.30}$$

Substitute the total values as the addition of steady state value and perturbed value.

$$I_{xx}\frac{d\left(P_1+p\right)}{dt}-I_{xz}\frac{d\left(R_1+r\right)}{dt}=L_{A+P_1}+l_{A+P}+\left(P_1+p\right)\left(Q_1+q\right)I_{xz}+\left(I_{yy}-I_{zz}\right)\left(R_1+r\right)\left(Q_1+q\right) \tag{B.31}$$

Expand the equations.

$$\begin{aligned}I_{xx}\frac{dP_1}{dt}+I_{xx}\frac{dp}{dt}-I_{xz}\frac{dR_1}{dt}-I_{xz}\frac{dr}{dt}&=L_{A+P_1}+l_{A+P}+\left(P_1Q_1+P_1q+pQ_1+pq\right)I_{xz}\\&\quad+\left(I_{yy}-I_{zz}\right)\left(R_1Q_1+R_1q+Q_1r+rq\right)\end{aligned} \tag{B.32}$$

Cancel out the steady state terms.

$$I_{xx}\frac{dp}{dt}-I_{xz}\frac{dr}{dt}=l_{A+P}+\left(P_1q+pQ_1+pq\right)I_{xz}+\left(I_{yy}-I_{zz}\right)\left(R_1q+Q_1r+rq\right) \tag{B.33}$$

Apply the assumption regarding to multiplication of perturbed values.

$$\boxed{I_{xx}\frac{dp}{dt}-I_{xz}\frac{dr}{dt}=l_{A+P}+\left(P_1q+pQ_1\right)I_{xz}+\left(I_{yy}-I_{zz}\right)\left(R_1q+Q_1r\right)} \tag{B.34}$$

Moment in y_B direction.

$$I_{yy}\frac{dQ}{dt}=M_{A+P}+(R^2-P^2)I_{xz}+PR\left(I_{zz}-I_{xx}\right) \tag{B.35}$$

@Steady state

$$I_{yy}\frac{dQ_1}{dt}=M_{A+P_1}+(R_1^2-P_1^2)I_{xz}+P_1R_1\left(I_{zz}-I_{xx}\right) \tag{B.36}$$

Substitute the total values as the addition of steady state value and perturbed value.

$$I_{yy}\frac{d\left(Q_1+q\right)}{dt} = M_{A+P_1} + m_{A+P} + ((R_1+r)^2 - (P_1+p)^2)I_{xz} + (P_1+p)(R_1+r)\left(I_{zz}-I_{xx}\right) \tag{B.37}$$

Expand the equation.

$$\begin{aligned} I_{yy}\frac{dQ_1}{dt} + I_{yy}\frac{dq}{dt} = M_{A+P_1} + m_{A+P} + ((R_1^2+2R_1r+r^2) - (P_1^2+2P_1p+p^2))I_{xz} \\ + \left(P_1R_1 + P_1r + pR_1 + pr\right)\left(I_{zz}-I_{xx}\right) \end{aligned} \tag{B.38}$$

Cancel out the steady state terms.

$$I_{yy}\frac{dq}{dt} = m_{A+P} + ((2R_1r+r^2) - (2P_1p+p^2))I_{xz} + \left(P_1r + pR_1 + pr\right)\left(I_{zz}-I_{xx}\right) \tag{B.39}$$

Apply the assumption regarding to multiplication of perturbed values.

$$\boxed{I_{yy}\frac{dq}{dt} = m_{A+P} + ((2R_1r) - (2P_1p))I_{xz} + \left(P_1r + pR_1\right)\left(I_{zz}-I_{xx}\right)} \tag{B.40}$$

Moment in z_B direction.

$$I_{xx}\frac{dR}{dt} - I_{xz}\frac{dP}{dt} = N_{A+P} + PQ(I_{xx}-I_{yy}) - QRI_{xz} \tag{B.41}$$

@Steady state

$$I_{xx}\frac{dR_1}{dt} - I_{xz}\frac{dP_1}{dt} = N_{A+P_1} + P_1Q_1(I_{xx}-I_{yy}) - Q_1R_1I_{xz} \tag{B.42}$$

Substitute the total values as the addition of steady state value and perturbed value.

$$I_{xx}\frac{d(R_1+r)}{dt} - I_{xz}\frac{d(P_1+p)}{dt} = N_{A+P_1} + n_{A+P} + (P_1+p)(Q_1+q)(I_{xx}-I_{yy}) - (Q_1+q)(R_1+r)I_{xz} \tag{B.43}$$

Expand the equation.

$$\begin{aligned} I_{xx}\frac{dR_1}{dt} + I_{xx}\frac{dr}{dt} - I_{xz}\frac{dP_1}{dt} - I_{xz}\frac{dp}{dt} = N_{A+P_1} + n_{A+P} + (P_1Q_1 + P_1q + pQ_1 + pq)(I_{xx}-I_{yy}) \\ - (Q_1R_1 + Q_1r + qR_1 + qr)I_{xz} \end{aligned} \tag{B.44}$$

Cancel out the steady state terms.

$$I_{xx}\frac{dr}{dt} - I_{xz}\frac{dp}{dt} = n_{A+P} + (P_1q + pQ_1 + pq)(I_{xx}-I_{yy}) - (Q_1r + qR_1 + qr)I_{xz} \tag{B.45}$$

Apply the assumption regarding to multiplication of perturbed values.

$$I_{xx}\frac{dr}{dt} - I_{xz}\frac{dp}{dt} = n_{A+P} + (P_1q + pQ_1)(I_{xx}-I_{yy}) - (Q_1r + qR_1)I_{xz} \tag{B.46}$$

Appendix C

Flight Test Results

C.0.1 Impact of Servo Dynamics, Skyhunter November 26th 2017 flight test

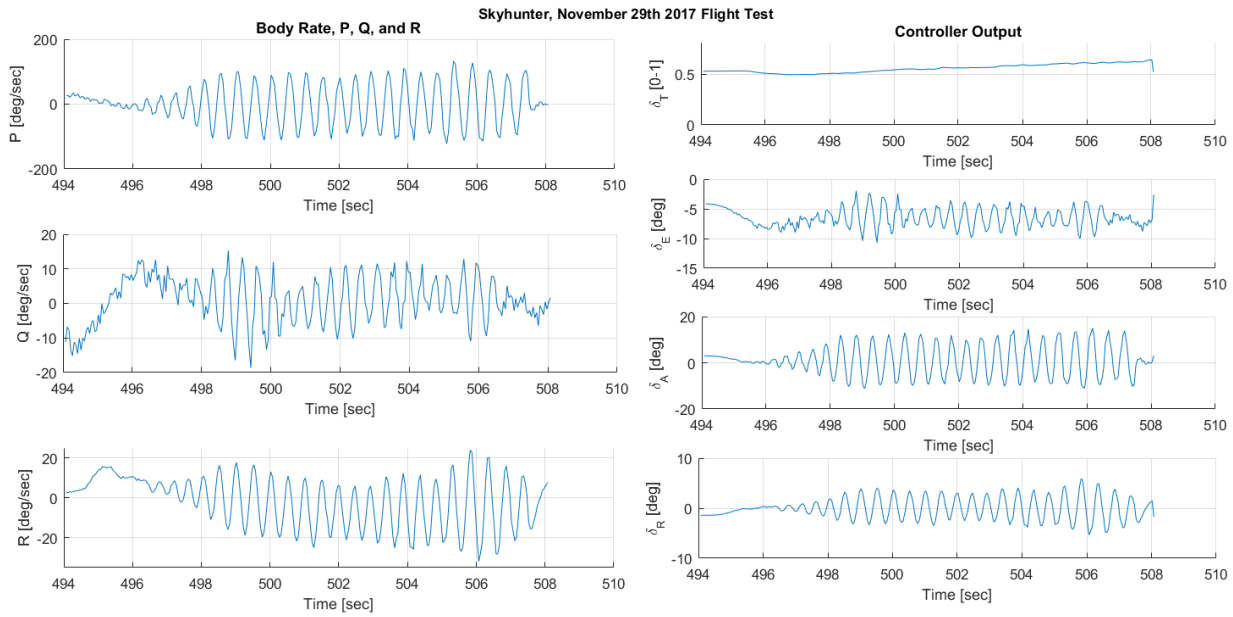


Figure C.1: Impact of Servo Dynamics due to the lack of robustness in un-modeled dynamics of LQR controller

In this flight test, all servo dynamics were assumed to be $10/(s+10)$. The servo dynamics

for recent flight tests are adjusted as follows:

$$\frac{\delta_T(s)}{\delta_{T_{cmd}}} = \frac{40}{s+40}, \quad \frac{\delta_E(s)}{\delta_{E_{cmd}}(s)} = \frac{35}{s+35}, \quad \frac{\delta_A(s)}{\delta_{A_{cmd}}(s)} = \frac{30}{s+30}, \quad \frac{\delta_R(s)}{\delta_{R_{cmd}}(s)} = \frac{50}{s+50} \qquad \text{(C.1)}$$

C.0.1.1 Greenland G1X flight test tracking RMS

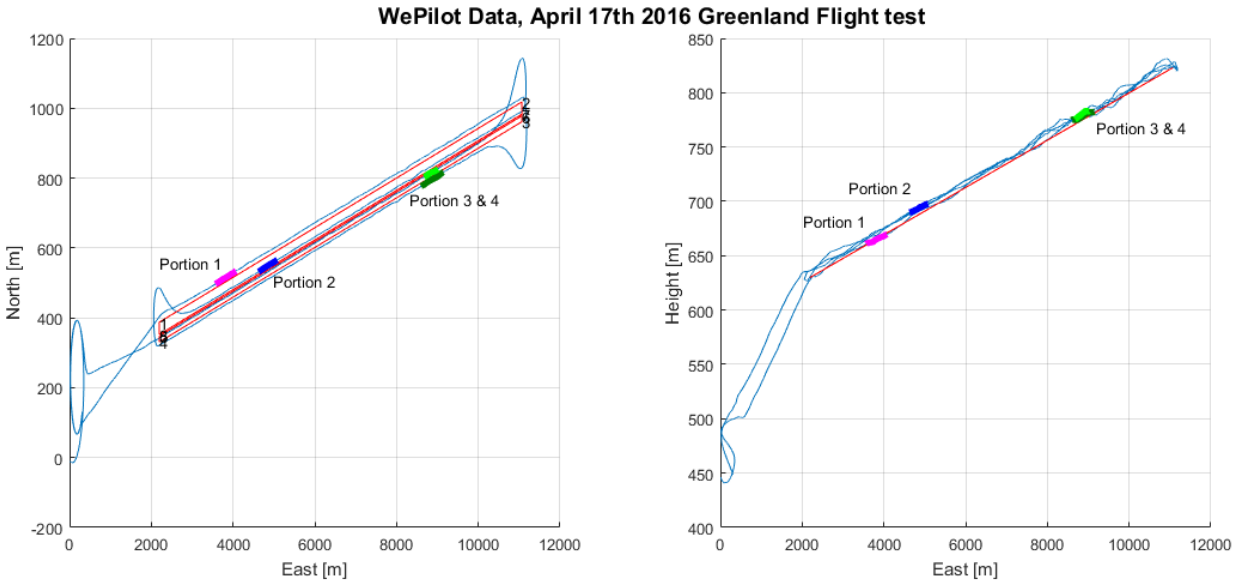


Figure C.2: Greenland flight test result: position and selected portions for calculating RMS

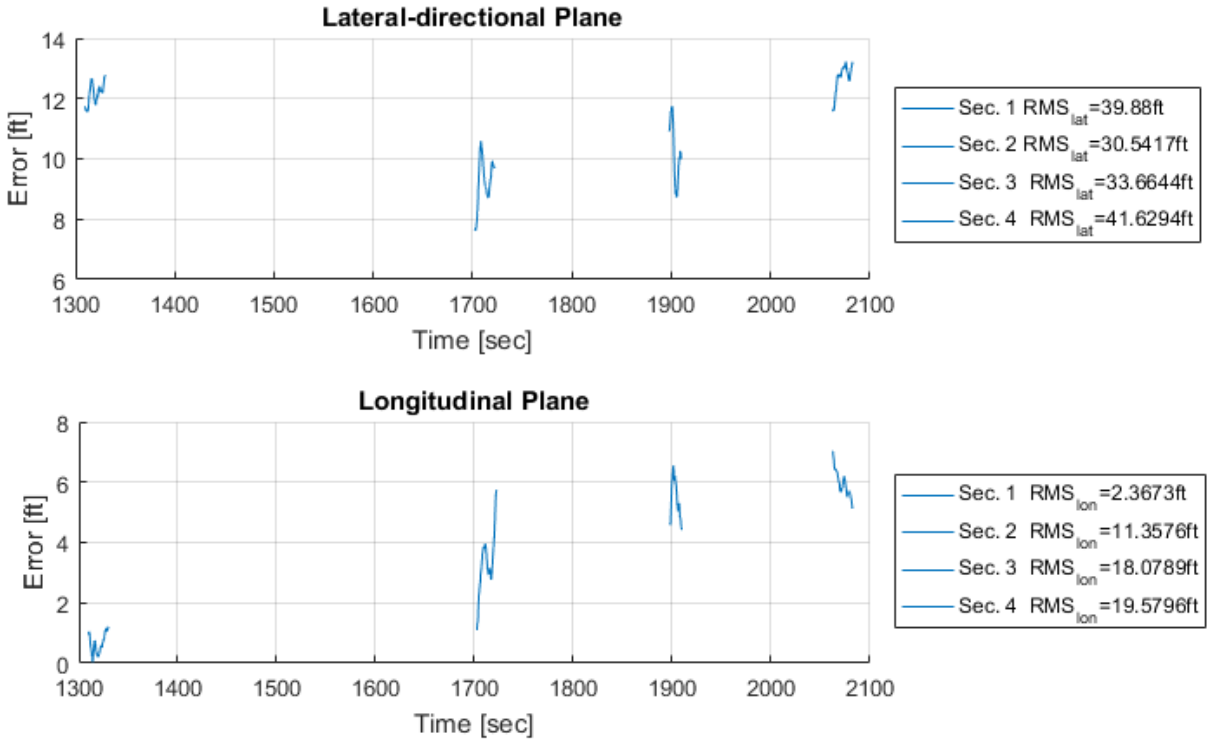


Figure C.3: Greenland flight test result: RMS for selected portion

Appendix D

Dynamic model of three UASs

In this section, dynamics information of three fixed wing vehicles is presented: DG 808, Skyhunter, and Yak-54 40%.

D.0.1 Geometric information

The following table presents major geometric information for three vehicles.

Table D.1: Geometric information for Skyhunter, DG 808, and Yak-54 40%

	Skyhunter		DG 808		Yak-54 40%	
S	4.82	ft^2	6.96	ft^2	20.71	ft^2
$\bar{c}$	0.731	ft	0.6	ft	1.9995	ft
b	6.875	ft	13.08	ft	11.03	ft
AR	9.81	-	25.98	-	5.87	-
Weight	8.27	lb	9.53	lb	64.84	lb
I_{xx_B}	0.44	slug-ft^2	1.1	slug-ft^2	4.2	slug-ft^2
I_{yy_B}	0.54	slug-ft^2	0.3	slug-ft^2	9.9	slug-ft^2
I_{zz_B}	0.35	slug-ft^2	1.43	slug-ft^2	11.5	slug-ft^2
α_1	0.7	deg	1.5	deg	3.4	deg
δ_{E_1}	8.27	deg	-1.7	deg	-1.27	deg
V_{T_1}	50.6	ft/sec	59.3	ft/sec	98	ft/sec

D.0.2 DG 808

Using the LTI model derivation presented in the previous section, following numerical results are achieved at a 35 knots trim speed, 1.6 degree of angle of attack, and 9.6 lbs weight at the trim.

Table D.2: Stability and Control derivatives for DG 808

C_{L_1}	0.3879	C_{D_1}	0.02312	C_{m_1}	0	C_{y_β}	-0.3266	C_{l_β}	-0.0376	C_{n_β}	0.0454
C_{L_0}	0.2087	$\overline{C_{D0}}$	0.00229	$C_{m_{T_1}}$	-0.035	C_{y_p}	-0.0491	C_{l_p}	-0.6637	C_{n_p}	-0.0582
C_{L_α}	7.1481	C_{D_u}	0	C_{m_0}	0.0267	C_{y_r}	0.1017	C_{l_r}	0.0926	C_{n_r}	-0.0263
C_{L_q}	5.1439	C_{D_α}	0.4844	C_{m_α}	-0.25	$C_{y_{\delta_a}}$	0	$C_{l_{\delta_a}}$	0.496	$C_{n_{\delta_a}}$	-0.0268
$C_{L_{\dot{\alpha}}}$	0.7327	C_{D_q}	0	C_{m_q}	-25	$C_{y_{\delta_r}}$	0.1197	$C_{l_{\delta_r}}$	0.00201	$C_{n_{\delta_r}}$	-0.0297
C_{L_u}	0.001	$C_{D_{\dot{\alpha}}}$	0	$C_{m_{\dot{\alpha}}}$	-3.9521						
$C_{L_{\delta_e}}$	0.2281	$C_{D_{\delta_e}}$	0.0106	C_{m_u}	0.00016						
				$C_{m_{\delta_e}}$	-1.2306						
				$C_{m_{T_u}}$	0.4057						
				$C_{m_{T_\alpha}}$	-3.5711						

$$A_{long} = \begin{bmatrix} 0 & 0 & 0 & 1.0000 \\ -32.1874 & -0.1970 & -7.7004 & 0 \\ -0.0151 & -0.0177 & -9.6386 & 0.9632 \\ 0.0093 & 0.0858 & -121.3584 & -5.4821 \end{bmatrix}, \qquad B_{long} = \begin{bmatrix} 0 \\ -0.8475 \\ -0.3067 \\ -40.8156 \end{bmatrix} \tag{D.1}$$

$$A_{lat} = \begin{bmatrix} 0 & 0 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 0 & 1.0000 \\ 0.5449 & 0 & -0.4412 & -0.0074 & -0.9847 \\ 0 & 0 & -10.7386 & -20.9688 & 2.9327 \\ 0 & 0 & 10.0800 & -1.2881 & -0.6602 \end{bmatrix}, \quad B_{long} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0.1616 \\ 140.5772 & 0.6242 \\ -6.7747 & -6.4997 \end{bmatrix} \tag{D.2}$$

Matrices of both axes represent the dynamics of the vehicle. These are very useful to perform modal analysis. The following table shows the modal analysis of DG 808.

τ is the time constant, ω_n and ω_d are the natural and damping frequency, respectively. ζ

Table D.3: Modal Analysis of DG 808

Mode	Poles	Dynamic Mode	τ [s]	ω_n [rad/s]	ω_d [rad/s]	ζ [-]
Longitudinal	$-0.0076 \pm 0.75i$	Phugoid Mode	13.2	0.75	0.7462	0.1
	$-7.6 \pm 10.6i$	Short Period	0.13	13.03	10.5715	0.582
Lateral-Diretional	0	-	∞	0	0	-1
	0.0052	Spiral	19.4	-	-	-1
	-20.8	Roll	0.005	-	-	1
	$-0.64 \pm 3.31i$	Dutch Roll	1.56	3.38	3.3184	0.19

is the damping ratio. As the result, longitudinal modes are stable since all poles are at left hand side of S-plane. For the lateral-directional mode, the spiral mode is unstable, which is typical for most aircraft.

D.1 Skyhunter

As identical principles are applied in the previous section, the following table shows the stability and control derivatives for Skyhunter.

Table D.4: Stability and Control derivatives for Skyhunter

C_{L_1}	0.5867	C_{D_1}	0.0499	C_{m_1}	0.0069	C_{y_β}	-0.5557	C_{l_β}	-0.1616	C_{n_β}	0.1235
C_{L_0}	0.5099	$\overline{C_{D0}}$	0.0384	$C_{m_{T_1}}$	-0.0069	C_{y_p}	-0.2067	C_{l_p}	-0.5470	C_{n_p}	-0.0686
C_{L_α}	5.8307	C_{D_u}	0	C_{m0}	0.0497	C_{y_r}	0.2749	C_{l_r}	0.1650	C_{n_r}	-0.1144
C_{L_q}	6.4797	C_{D_α}	0.2404	C_{m_α}	-0.9208	$C_{y_{\delta a}}$	0	$C_{l_{\delta a}}$	0.3349	$C_{n_{\delta a}}$	0
$C_{L_{\dot{\alpha}}}$	1.3634	C_{D_q}	0	C_{m_q}	-15.6737	$C_{y_{\delta r}}$	0.2282	$C_{l_{\delta r}}$	0.0176	$C_{n_{\delta r}}$	-0.0948
C_{L_u}	0.0012	$C_{D_{\dot{\alpha}}}$	0	$C_{m_{\dot{\alpha}}}$	-4.7723						
$C_{L_{\delta_e}}$	0.2643	$C_{D_{\delta_e}}$	0.0142	C_{m_u}	1.898						
				$C_{m_{\delta_e}}$	-1.0283						
				$C_{m_{T_u}}$	0.0222						
				$C_{m_{T_\alpha}}$	-0.1134						

The following matrices are obtained at 29 knots trim speed, 0.67 degree of angle of attack and 8.27 lbs weight at the trim.

$$A_{long} = \begin{bmatrix} -0.1240 & 19.0662 & -32.1974 & 0 \\ -0.0250 & -6.2646 & -0.0080 & 0.9405 \\ 0 & 0 & 0 & 1.0000 \\ 0.0224 & -14.6920 & 0.0051 & -2.7085 \end{bmatrix}, \qquad B_{long} = \begin{bmatrix} 5.9203 & -0.7755 \\ -0.0544 & -0.3158 \\ 0 & 0 \\ 0.2737 & -19.4782 \end{bmatrix} \quad \text{(D.3)}$$

$$A_{lat} = \begin{bmatrix} -0.6048 & 0.6359 & -0.0153 & -0.9797 \\ 0 & 0 & 1 & 0 \\ -35.6645 & 0 & -8.2161 & 2.4729 \\ 34.3765 & 0 & -1.3188 & -2.1512 \end{bmatrix}, \quad B_{long} = \begin{bmatrix} 0 & 0.2484 \\ 0 & 0 \\ 74.0763 & 3.8384 \\ 0.2236 & -26.3674 \end{bmatrix} \quad \text{(D.4)}$$

Modal analysis is shown in the following table.

Table D.5: Modal Analysis of Skyhunter

Mode	Poles	Dynamic Mode	τ [s]	ω_n [rad/s]	ω_d [rad/s]	ζ [-]
Longitudinal	$-0.095 \pm 0.72i$	Phugoid Mode	106	0.718	0.6691	0.013
	$-4.54 \pm 3.33i$	Short Period	0.22	5.63	3.33	0.81
Lateral-Diretional	0	-	∞	0	0	-1
	0.015	Spiral	68.1	-	-	-1
	-8.76	Roll	0.0114	-	-	1
	$-1.11 \pm 6.31i$	Dutch Roll	0.89	6.4	6.3024	0.174

As the result, longitudinal modes are stable since all poles are at left hand side of S-plane. For the lateral-directional mode, the spiral mode is unstable typically as most of the airplane are as well.

D.2 Yak-54 40%

Table D.6: Stability and Control derivatives for Yak-54 40%

C_{L_1}	0.2806	C_{D_1}	0.0356	C_{m_1}	-0.0035	C_{y_β}	-0.4141	C_{l_β}	-0.0567	C_{n_β}	0.1147
C_{L_0}	-0.0062	$\overline{C_{D0}}$	0.0311	$C_{m_{T_1}}$	0.0036	C_{y_p}	-0.0766	C_{l_p}	-0.4089	C_{n_p}	-0.0358
C_{L_α}	5.0248	C_{D_u}	0	C_{m0}	0.0101	C_{y_r}	0.3192	C_{l_r}	0.0637	C_{n_r}	-0.1634
C_{L_q}	6.3457	C_{D_α}	0.1555	C_{m_α}	-0.7370	$C_{y_{\delta_a}}$	0	$C_{l_{\delta_a}}$	0.2878	$C_{n_{\delta_a}}$	-0.0126
$C_{L_{\dot{\alpha}}}$	1.8707	C_{D_q}	0	C_{m_q}	-10.7547	$C_{y_{\delta_r}}$	0.2285	$C_{l_{\delta_r}}$	0.0034	$C_{n_{\delta_r}}$	-0.1213
C_{L_u}	0.0022	$C_{D_{\dot{\alpha}}}$	0	$C_{m_{\dot{\alpha}}}$	-4.9971						
$C_{L_{\delta_e}}$	0.4935	$C_{D_{\delta_e}}$	-0.0004	C_{m_u}	0.0003						
				$C_{m_{\delta_e}}$	-1.3182						
				$C_{m_{T_u}}$	-0.0064						
				$C_{m_{T_\alpha}}$	0.07962						

LTI model results are obtained at a 65 knots trim speed, 2.67 degree of angle of attack

and 65 lbs weight at the trim.

$$A_{long} = \begin{bmatrix} -0.1238 & 14.1920 & -32.1434 & 0 \\ -0.0065 & -5.7329 & -0.0191 & 0.9050 \\ 0 & 0 & 01 & \\ 0.0124 & -16.8495 & 0.0448 & -7.1927 \end{bmatrix}, \qquad B_{long} = \begin{bmatrix} 11.1087 & 0.0472 \\ -0.0282 & -0.5591 \\ 0 & 0 \\ -0.1367 & -59.5192 \end{bmatrix} \quad \text{(D.5)}$$

$$A_{lat} = \begin{bmatrix} -0.4795 & 0.3280 & -0.0050 & -0.9792; 0 \quad 0 \quad 1.0000 \quad 0 \\ -36.4634 & 0 & -13.7393 & 2.3576 \\ 25.8252 & 0 & 0.0732 & -2.1212 \end{bmatrix}, \quad B_{long} = \begin{bmatrix} 0 & 0.2646 \\ 0 & 0 \\ 172.6516 & 4.7791 \\ -9.2658 & -26.8321 \end{bmatrix} \quad \text{(D.6)}$$

The following table shows modal analysis of Yak-54 40%.

Table D.7: Modal Analysis of Yak-54 40%

Mode	Poles	Dynamic Mode	τ [s]	ω_n [rad/s]	ω_d [rad/s]	ζ [-]
Longitudinal	$-0.053 \pm 0.314i$	Phugoid Mode	18.9	0.35	0.32	0.017
	$-6.47 \pm 3.84i$	Short Period	0.156	7.53	3.84	0.86
Lateral-Diretional	0	-	∞	0	0	-1
	-0.0146	Spiral	68.5	-	-	1
	-13.8	Roll	0.0724	-	-	1
	$-1.26 \pm 5.02i$	Dutch Roll	0.797	5.17	5.019	0.243

As the result, longitudinal modes are stable since all poles are at left hand side of S-plane. For the lateral-directional mode, all modes are stable since all poles are located at left hand side of S-plane.

Appendix E

Simulation Parameters

E.1 Solo Flight & Swarm Collision Avoidance, DG-808: March 9th 2017

Table E.1: Guidance Parameters for DG-808 Solo flight

Parameter	Value	Unit
Time for smoothing of guidance commands	0.1	sec
Time for smoothing of control outputs	0.5	sec
$K_{p_{Lon}}$	0.001	-
$K_{i_{Lon}}$	0.03	-
$K_{a_{Lon}}$	4	-
$d_{dR_{Lon}}$	5500	-
$d_{dR_{Lat}}$	260	-
Elevator trim correction	on	-
K for elevator correction	-0.5	-
q_{22} for LQ guidance	0.3	-
d_b for LQ guidance	0.1	ft
σ for morphing potential	200	ft
shifting factor for morphing potential	0.3	-
gyro effect for morphing potential	0.1	-
$n_{formation}$, desried relative distance (north) for Agent 1	100	ft
$e_{formation}$, desried relative distance (east) for Agent 1	-100	ft
$n_{formation}$, desried relative distance (north) for Agent 2	-100	ft
$e_{formation}$, desried relative distance (east) for Agent 2	100	ft

- Longitudinal Control weighting matrix

$$Q = diag\left(\begin{bmatrix} 0.00012 & 3e-06 & 0.098484 & 0.003 & 0.003 & 15 \end{bmatrix}\right) \tag{E.1}$$

$$R = diag\left(\begin{bmatrix} 300 & 300 \end{bmatrix}\right) \tag{E.2}$$

- Longitudinal Controller gain

$$K_{LQR} = \begin{bmatrix} 0.0065 & 0.0057 & -0.011 & -0.00094 & -0.0015 & -0.198 \\ 4.2889e-05 & 0.22 & -0.31 & -0.018 & -0.0028 & 0.103 \end{bmatrix}$$

- Lateral Control weighting matrix

$$Q = diag\left(\begin{bmatrix} 131.31 & 0.91189 & 2.0518 & 2.0518 & 11.459 & 11.459 \end{bmatrix}\right) \tag{E.3}$$

$$R = diag\left(\begin{bmatrix} 50 & 300 \end{bmatrix}\right) \tag{E.4}$$

- Lateral Controller gain

$$K_{LQR} = \begin{bmatrix} 1.0164 & 0.82324 & 0.12751 & -0.22901 & -0.15218 & -0.4539 \\ 0.084184 & -0.08266 & -0.0012406 & -0.06836 & -0.1853 & 0.062127 \end{bmatrix}$$

E.2 Swarm flight, DG-808: March 26th 2017

Table E.2: Guidance Parameters for DG-808 Swarm flight

Parameter	Value	Unit
Time for smoothing of guidance commands	0.1	sec
Time for smoothing of control outputs	2	sec
$K_{p_{Lon}}$	0.001	-
$K_{i_{Lon}}$	0.03	-
$K_{a_{Lon}}$	4	-
$d_{dR_{Lon}}$	5500	-
$d_{dR_{Lat}}$	350	-
Elevator trim correction	on	-
K for elevator correction	-0.5	-
q_{22} for LQ guidance	0.3	-
d_b for LQ guidance	0.1	ft
σ for morphing potential	50	ft
shifting factor for morphing potential	0.3	-
gyro effect for morphing potential	0.1	-
$n_{formation}$, desried relative distance (north) for Agent 1	100	ft
$e_{formation}$, desried relative distance (east) for Agent 1	-50	ft
$n_{formation}$, desried relative distance (north) for Agent 2	-100	ft
$e_{formation}$, desried relative distance (east) for Agent 2	50	ft

- Longitudinal Control weighting matrix

$$Q = diag\left(\begin{bmatrix} 0.00012 & 3e-06 & 0.098484 & 0.003 & 0.003 & 15 \end{bmatrix}\right) \tag{E.5}$$

$$R = diag\left(\begin{bmatrix} 300 & 300 \end{bmatrix}\right) \tag{E.6}$$

- Longitudinal Controller gain

$$K_{LQR} = \begin{bmatrix} 0.0065 & 0.0057 & -0.011 & -0.00094 & -0.0015 & -0.198 \\ 4.2889e-05 & 0.22 & -0.31 & -0.018 & -0.0028 & 0.103 \end{bmatrix}$$

- Lateral Control weighting matrix

$$Q = diag\left(\begin{bmatrix} 131.31 & 0.91189 & 2.0518 & 2.0518 & 11.459 & 11.459 \end{bmatrix}\right) \tag{E.7}$$

$$R = diag\left(\begin{bmatrix} 50 & 300 \end{bmatrix}\right) \tag{E.8}$$

- Lateral Controller gain

$$K_{LQR} = \begin{bmatrix} 1.0164 & 0.82324 & 0.12751 & -0.22901 & -0.15218 & -0.4539 \\ 0.084184 & -0.08266 & -0.0012406 & -0.06836 & -0.1853 & 0.062127 \end{bmatrix}$$

E.3 Solo Flight, Skyhunter: August 14th 2017

Table E.3: Guidance Parameters for Skyhunter

Parameter	Value	Unit
Time for smoothing of guidance commands	10	sec
Time for smoothing of control outputs	3	sec
$K_{p_{Lon}}$	0.001	-
$K_{i_{Lon}}$	0.03	-
$K_{a_{Lon}}$	4	-
$d_{dR_{Lon}}$	5500	-
$d_{dR_{Lat}}$	360	-
Elevator trim correction	off	-
K for elevator correction	-0.25	-
q_{22} for LQ guidance	0.3	-
d_b for LQ guidance	2	ft

- Longitudinal Control weighting matrix

$$Q = diag\left(\begin{bmatrix} 0.00012 & 3e-06 & 0.098484 & 0.075 & 0.003 & 15 \end{bmatrix}\right) \tag{E.9}$$

$$R = diag\left(\begin{bmatrix} 300 & 400 \end{bmatrix}\right) \tag{E.10}$$

- Longitudinal Controller gain

$$K_{LQR} = \begin{bmatrix} 0.0062822 & 0.017652 & -0.018508 & -6.6562e-05 & -0.0016927 & -0.18888 \\ 0.00089597 & 0.18907 & -0.34441 & -0.049962 & -0.0023133 & 0.10365 \end{bmatrix}$$

- Lateral Control weighting matrix

$$Q = diag\left(\begin{bmatrix} 131.31 & 0.91189 & 8.1225 & 2.0518 & 11.459 & 11.459 \end{bmatrix}\right) \tag{E.11}$$

$$R = diag\left(\begin{bmatrix} 300 & 80 \end{bmatrix}\right) \tag{E.12}$$

- Lateral Controller gain

$$K_{LQR} = \begin{bmatrix} -0.15819 & 0.2989 & 0.10577 & 0.01562 & -0.050486 & -0.18881 \\ 0.62265 & -0.015633 & -0.0024902 & -0.19247 & -0.36562 & 0.097766 \end{bmatrix}$$

E.4 Solo & Swarm Flight, Skyhunter: April 26th and 28th 2018

Table E.4: Guidance Parameters for Skyhunter

Parameter	Value	Unit
Time for smoothing of guidance commands	4	sec
Time for smoothing of control outputs	1	sec
$K_{p_{Lon}}$	0.5	-
$K_{i_{Lon}}$	0.05	-
$K_{a_{Lon}}$	3	-
$d_{dR_{Lon}}$	700	-
$d_{dR_{Lat}}$	360	-
Elevator trim correction	off	-
K for elevator correction	-0.25	-
q_{22} for LQ guidance	2.5	-
d_b for LQ guidance	1	ft

- Longitudinal Control weighting matrix

$$Q = diag\left(\begin{bmatrix} 0.0001 & 3e-06 & 0.0985 & 50 & 0.003 & 15 \end{bmatrix}\right) \tag{E.13}$$

$$R = diag\left(\begin{bmatrix} 300 & 400 \end{bmatrix}\right) \tag{E.14}$$

- Longitudinal Controller gain

$$K_{LQR} = \begin{bmatrix} 0.0066 & 0.0144 & -0.0262 & 0.0042 & -0.0015 & -0.1950 \\ 0.0037 & 0.3075 & -0.5044 & -0.2680 & -0.0024 & 0.0948 \end{bmatrix}$$

- Lateral Control weighting matrix

$$Q = diag\left(\begin{bmatrix} 131.31 & 0.91189 & 8.1225 & 2.0518 & 11.459 & 11.459 \end{bmatrix}\right) \tag{E.15}$$

$$R = diag\left(\begin{bmatrix} 300 & 80 \end{bmatrix}\right) \tag{E.16}$$

- Lateral Controller gain

$$K_{LQR} = \begin{bmatrix} -0.15819 & 0.2989 & 0.10577 & 0.01562 & -0.050486 & -0.18881 \\ 0.62265 & -0.015633 & -0.0024902 & -0.19247 & -0.36562 & 0.097766 \end{bmatrix}$$

E.5 Solo Flight, G1X: December 20th 2017

Table E.5: Guidance Parameters for G1X Solo flight

Parameter	Value	Unit
Time for smoothing of guidance commands	6	sec
Time for smoothing of control outputs	2	sec
$K_{p_{Lon}}$	0.001	-
$K_{i_{Lon}}$	0.03	-
$K_{a_{Lon}}$	4	-
$d_{dR_{Lon}}$	4500	-
$d_{dR_{Lat}}$	650	-
Elevator trim correction	off	-
q_{22} for LQ guidance	5	-
d_b for LQ guidance	5	ft

- Longitudinal Control weighting matrix

$$Q = diag\left(\begin{bmatrix} 0.2 & 0.0001 & 3.2828 & 500 & 0.1 & 8500 \end{bmatrix}\right) \tag{E.17}$$

$$R = diag\left(\begin{bmatrix} 1600 & 5500 \end{bmatrix}\right) \tag{E.18}$$

- Longitudinal Controller gain

$$K_{LQR} = \begin{bmatrix} 0.0296 & 0.0726 & -0.1211 & -0.0030 & -0.0074 & -0.7928 \\ 0.0034 & 0.2462 & -0.9837 & -0.2288 & -0.0015 & 1.1673 \end{bmatrix}$$

- Lateral Control weighting matrix

$$Q = diag\left(\begin{bmatrix} 0.01 & 0.1 & 30 & 300 & 0.01 & 700 \end{bmatrix}\right) \tag{E.19}$$

$$R = diag\left(\begin{bmatrix} 5000 & 6500 \end{bmatrix}\right) \tag{E.20}$$

- Lateral Controller gain

$$K_{LQR} = \begin{bmatrix} 0.1152 & 0.3286 & 0.0337 & -0.0391 & -0.0002 & -0.3687 \\ -0.0412 & -0.0044 & -0.0011 & -0.1520 & -0.0012 & 0.0558 \end{bmatrix}$$