4-2017

Leader-Follower Trajectory Generation and Tracking for Quadrotor Swarms

Michael James Campobasso

LEADER-FOLLOWER TRAJECTORY GENERATION AND TRACKING FOR QUADROTOR SWARMS

BY
MICHAEL JAMES CAMPOBASSO

A Thesis
Submitted to the Department of Physical Sciences
and the Committee on Graduate Studies
In partial fulfillment of the requirements
for the degree of
Master in Science in Engineering Physics

04/2017
Embry-Riddle Aeronautical University
Daytona Beach, Florida

LEADER-FOLLOWER TRAJECTORY GENERATION AND TRACKING FOR QUADROTOR SWARMS

by

Michael James Campobasso

This thesis was prepared under the direction of the candidate's Thesis Committee Chair, Dr. Mahmut Reyhanoglu, Professor, Daytona Beach Campus, and Thesis Committee Members Dr. William Mackunis, Associate Professor, Daytona Beach Campus, and Dr. John Hughes, Associate Professor, Daytona Beach Campus, and has been approved by the Thesis Committee. It was submitted to the Department of Physical Sciences in partial fulfillment of the requirements of the degree of
Master of Science in Engineering Physics

THESIS COMMITTEE:

M. Reyhanoglu

Dr. Mahmut Reyhanoglu,
Committee Chair

Dr. William Mackunis,
Committee Member

Dr. John Hughes,
Committee Member

Terry D. Oswalt

Dr. Terry Oswalt,
Department Chair, Physical Sciences

Karen F. Gaines

Dr. Karen Gaines,
Dean, College of Arts and Sciences

Dr. Christopher Grant,
Vice Chancellor

Acknowledgments

First, I would like to thank my advisor, Dr. Reyhanoglu for his encouragement and support. It has been a pleasure doing research with him.

I would also like to thank, Dr. MacKunis and Dr. Hughes, for being a part of this experience as the other two members of the Thesis Committee.

Muhammad Rehan has been an enormous help in my completing this thesis. I greatly appreciate him and all that he has done for me throughout this experience.

Finally, I would like to thank my family for always standing behind me in my studies and all aspects of my life.

Thank you,

Michael James Campobasso

Abstract

Swarm control is an essential step in the progress of robotic technology. The use of multiple agents to perform tasks more effectively and efficiently than a single agent allows for the expansion of robot use in all aspects of life. One of the foundations of this area of research is the concept of Leader-Follower swarm control. A crucial aspect of this idea is the generation of trajectories with respect to the leader's path and some desired formation. With these trajectories generated, one can use a tracking controller specific to the swarm vehicle of choice to accomplish the desired swarm formation. In this paper, a Leader-Follower trajectory generator is developed for a planar triangular formation with offset vertical positions. A tracking controller is used to achieve formation flight for the quadrotor application. A well-accepted model for quadrotor vehicles is used, with simulation parameters comparable to those of a small commercial quadrotor. The swarm control objective is achieved in simulation and is proved to be effective theoretically through the Lyapunov analysis.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Unmanned Aerial Vehicles (UAVs) have been rapidly growing in popularity over the past decade. With new military [Orfanus et al. (2016)] [McConnell (2007)], medical [Nedjati et al. (2016)] [Qi et al. (2016)], commercial [Hongxia and Qi (2016)] [Torrés-Sanchez et al. (2014)], and academic applications [Grøtli and Johansen (2012)] [Bürkle et al. (2011)], UAV research has met an increased demand for sophistication. A branch of UAVs that is gaining attention is the quadrotor vehicle. This is a helicopter-like vehicle with four propellers providing lift. The main advantages to this type of UAV are that it is low-cost, agile, able to hover, and mechanically simpler than a helicopter. In the natural progression of this technology, it has been concluded that there is substantial need for the use of multiple quadrotors, working together, to accomplish more difficult, large-scale missions [Schwager et al. (2011)] [Franchi et al. (2016)]. More than one quadrotor autonomously working in a collaborative group is referred to as a swarm. A quadrotor swarm can be more efficient and more capable than a single quadrotor [Goodarzi and Lee (2016)] [Lee (2017)].

1.2 Swarm Control Approaches

There are three approaches to swarm control; leader-follower, behavioral, and virtual structure. They each have benefits and downfalls that should be taken into consideration when choosing an approach for a specific application.

1.2.1 Leader-Follower Approach

The leader-follower approach is arguably the most popular swarm control approach. In the swarm, one quadrotor is the leader and the rest are followers. The leader quadrotor has the necessary technology to achieve the desired state of the swarm, while the follower quadrotors have enough technology to track a function of the leader's state. In other words, the leader can determine and execute the actions necessary to lead the swarm toward achieving the goal, while the followers must simply follow the leader [Roldão et al. (2014)].

1.2.2 Behavioral Approach

The behavioral approach is commonly used for distributed robotic systems. It uses a set of simple behaviors that is predefined to characterize the state of the swarm and control the movement of the swarm [Xu et al. (2014)]. To give context to this idea, example predefined behaviors include take-off, hover, and land. These and other behaviors can be used by the quadrotor swarm to achieve the goal.

1.2.3 Virtual Structure Approach

The virtual structure approach has a range of applications. The swarm consists entirely of follower quadrotors that follow the state of a leader quadrotor, which is not physically present. This virtual leader determines the desired states of the swarm, in the form of a structure, while the followers have enough technology to track the state of this structure. This can be thought of as a swarm having a particular shape that moves in time according to the control law given to the virtual leader [Mehrjerdi et al. (2011)].

1.3 Advantages, Disadvantages, and Applications of Each Approach

Each swarm control approach has advantages and disadvantages. Consequently, in application, there is typically one approach that works better than the other two.

1.3.1 Leader-Follower Approach Application

The leader-follower approach has the advantage of being minimalist and simple [Pereira et al. (2017)]. The leader quadrotor must be equipped with the range of instrumentation to determine the desired state history of the swarm. However, all of the followers need only have enough technology to communicate with the leader, or another follower if the network is connected, and achieve some function of the leader's state [Mahmood and Kim (2015)] [Rabah and Qinghe (2015)] [Vargas-Jacob et al. (2016)]. This eliminates the need to make many highly sophisticated quadrotors. Instead, one, usually expensive, quadrotor is used as the leader, with many inexpensive followers. This allows the use of many followers with respect to a fixed budget.

The main disadvantage of the leader-follower approach is that it has a single point of failure. If the leader fails, the entire swarm fails.

In application, the leader-follower approach is used due to cost restrictions and hardware limitations [Hu and Feng (2010)] [Cui et al. (2010)]. Additionally, the leader-follower approach has the application driven advantage of providing the ability to integrate a manned UAV. A manned UAV can be used as the leader, with many followers following for auxiliary aid and functionality. This is highly preferred since a manned vehicle is less prone to failure and does not require the highly sophisticated control law and instrumentation that a typical leader would. It does, however, require additional size and safety accommodations.

1.3.2 Behavioral Approach Application

The behavioral approach has the advantage of accommodating distributed systems of highly autonomous robots, allowing a more diverse and robust swarm [Lawton et al. (2003)] [Balch and Arkin (1998)] [Schneider-Fontan and Mataric (1998)]. Since there are less stringent conditions on the state of the swarm and individuals, and more focus on the desired action, or behavior, of the swarm, it is easier to incorporate obstacle and threat avoidance. Additionally, there is no single point of failure, so if one member fails, the swarm can still continue on the mission [Parker (1998)].

However, the behavioral approach can be considered the most complicated and expensive approach since each member of the swarm must be highly sophisticated with respect to on board technology and autonomy [Veloso et al. (1999)].

1.3.3 Virtual Structure Approach Application

The virtual structure approach has advantages as well. This approach does not have a single point of failure in the swarm itself, making it a viable option for UAVs in general [Mortazavi et al. (2015)]. Additionally, the structures can be either rigid or flexible, depending on the application [Sun and Xia (2016)] [Nadjim and Karim (2014)] [Lewis and Tan (1997)]. A decentralized approach can also be used with the virtual structure concept [Ren and Beard (2004)]. However, this approach requires each of the followers to have the computing power to track a function of the virtual leader's desired state history while also maintaining the structure, with respect to the other followers. This lends itself to more sophisticated control and instrumentation for each agent, which increases cost.

1.4 Contribution of Thesis

This thesis provides a Leader-Follower formation control for quadrotors. A Lyapunov-based integrator-backstepping trajectory generator is used for all followers with respect to a predefined leader's path. Additionally, a backstepping, sliding-mode tracking controller is used for the quadrotor vehicle application to track the generated trajectories. Simulation results are presented for one leader and two followers.

1.5 Organization of Thesis

This thesis is organized as such. Chapter 2 explains the necessary mathematical background for the developed theory. Chapter 3 includes the theoretical derivations and simulation results for the trajectory generation. Chapter 4 shows theoretical derivations and simulation results for the quadrotor tracking controller. Chapter 5 concludes this thesis with future research proposals and further applications and experiments.

Chapter 2

Mathematical Model

2.1 Objectives

The purpose of this chapter is to develop the mathematical model that describes the motion of a quadrotor vehicle. This is important because these nonlinear quadrotor dynamics will be used to simulate the tracking controller presented in Chapter 4. Additionally, the Lyapunov analysis methodology is explored. This is used to show asymptotic stability of the formation flight trajectory generator. Finally, LaSalle's invariance principle is explained. Since the Lyapunov analysis has a negative semi-definite result, LaSalle's invariance principle must be used to show global asymptotic stability of the formation flight trajectory generator.

2.2 Quadrotor Dynamics

For the quadrotor tracking simulation presented in Chapter 4, a plant describing the quadrotor dynamics is necessary to show that the control law is effective. The translational and rotational dynamics are derived in [Wie (2008)]. The quadrotor is assumed to translate and rotate as a rigid body. This implies that rigid body dynamics can be used for this model. These dynamics will be developed for a right-handed coordinate system in which the body-fixed positive z-direction is oriented in the direction of thrust due to the propellers of the quadrotor as shown in Figure 2.1. For translational motion, the development starts with Newton's Law:

$$\mathbf{F}_t = m\ddot{\mathbf{p}} \tag{2.1}$$

where $\mathbf{F}_t$ is the force on the quadrotor, m is the mass of the quadrotor, and $\mathbf{p}$ is the inertial position of the quadrotor represented by:

$$\mathbf{p} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{2.2}$$

For this formulation, the drag force on the quadrotor will be considered negligible. Thus, Equation (2.1) can be expanded as such:

$$m\ddot{\mathbf{p}} = (mu_1\mathscr{R}^T\hat{e}_3 - mg\hat{e}_3) \tag{2.3}$$

where u_1 is the acceleration magnitude due to the thrust of the quadrotor propellers, g is the acceleration magnitude due to gravity, $\hat{e}_3$ is the vertical unit vector, and $\mathscr{R}$ is the rotation matrix represented by:

$$\mathscr{R} = \begin{bmatrix} c(\phi)c(\theta) & s(\psi)c(\theta) & -s(\theta) \\ c(\psi)s(\phi)s(\theta) - c(\phi)s(\psi) & s(\psi)s(\phi)s(\theta) + c(\phi)c(\psi) & c(\theta)s(\phi) \\ c(\psi)c(\phi)s(\theta) + s(\phi)s(\psi) & s(\psi)c(\phi)s(\theta) - s(\phi)c(\psi) & c(\theta)c(\phi) \end{bmatrix} \tag{2.4}$$

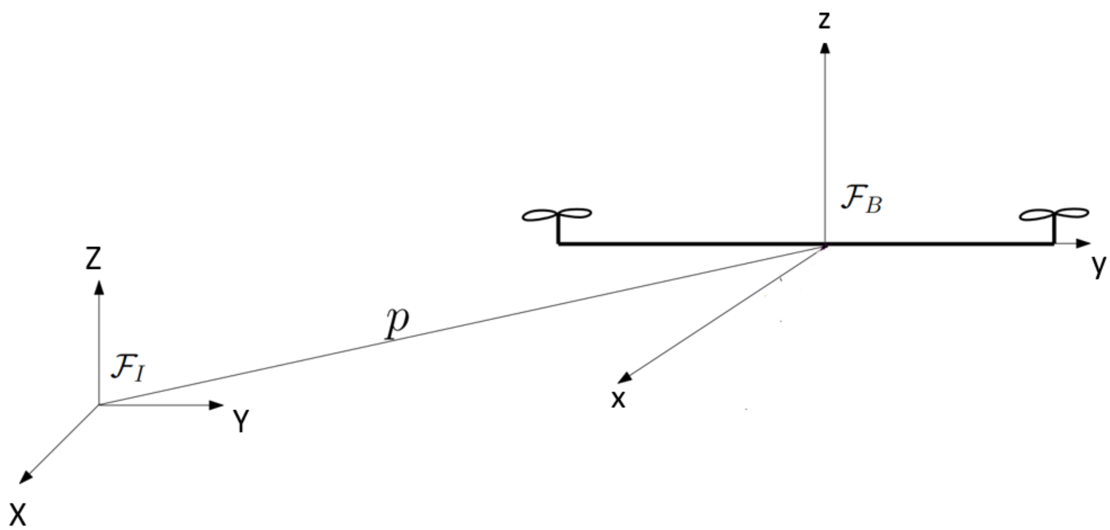

Figure 2.1: Inertial and Body Reference Frames.

where ϕ is the roll angle, θ is the pitch angle, and ψ is the yaw angle. Here the abbreviations $c(\cdot) = \cos(\cdot)$ and $s(\cdot) = \sin(\cdot)$ have been used to shorten the notation.

After simplifying Equation (2.3) the following translational quadrotor dynamics are obtained:

$$\ddot{X} = (cos(\phi)sin(\theta)cos(\psi) + sin(\phi)sin(\psi))u_1 \tag{2.5}$$

$$\ddot{Y} = (cos(\phi)sin(\theta)sin(\psi) - sin(\phi)cos(\psi))u_1 \tag{2.6}$$

$$\ddot{Z} = cos(\phi)cos(\theta)u_1 - g \tag{2.7}$$

In addition to the translational dynamics, the rotational dynamics for a rigid body are used to determine the angular acceleration terms. This development begins with Euler's rotational equations of motion:

$$J\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times J\boldsymbol{\omega} = \boldsymbol{\tau} \tag{2.8}$$

where J is the inertia matrix with respect to the body frame, $\boldsymbol{\omega}$ is the vector angular velocity in the body frame, and τ includes the moments about each axis that are caused by thrust components and gyroscopic terms. To simplify the inertia matrix, the quadrotor is assumed to be axisymmetric. This results in a diagonal inertia matrix shown in Equation (2.9).

$$J = \begin{bmatrix} J_{xx} & 0 & 0 \\ 0 & J_{yy} & 0 \\ 0 & 0 & J_{zz} \end{bmatrix} \tag{2.9}$$

where J_{xx}, J_{yy}, and J_{zz} are the moments of inertia about the x, y, and z axes, respectively.

After simplifying Equation (2.8), the following equations are obtained:

$$J_{xx}\dot{\omega}_x - (J_{yy} - J_{zz})\omega_y\omega_z = u_2 + J_r u_g \omega_y \tag{2.10}$$

$$J_{yy}\dot{\omega}_y - (J_{zz} - J_{xx})\omega_x\omega_z = u_3 - J_r u_g \omega_x \tag{2.11}$$

$$J_{zz}\dot{\omega}_z = J_{zz}u_4 \tag{2.12}$$

where u_2 is the angular acceleration about the roll axis caused by the thrust component, u_3 is the angular acceleration about the pitch axis caused by the thrust component, u_4 is the angular acceleration about the yaw axis caused by the thrust component, J_r is the moment of inertia of the rotors, and u_g is the gyroscopic input term.

Equations (2.10), (2.11), and (2.12) can be rearranged to solve for the angular accelerations of the quadrotor as such:

$$\dot{\omega}_x = J_{yzx}\omega_y\omega_z + \frac{J_r}{J_{xx}}u_g\omega_y + u_2 \tag{2.13}$$

$$\dot{\omega}_y = J_{zxy}\omega_x\omega_z - \frac{J_r}{J_{yy}}u_g\omega_x + u_3 \tag{2.14}$$

$$\dot{\omega}_z = u_4 \tag{2.15}$$

where $J_{yzx} = \frac{J_{yy}-J_{zz}}{J_{xx}}$ and $J_{zxy} = \frac{J_{zz}-J_{xx}}{J_{yy}}$.

In Chapter 4, u_1, u_2, u_3, and u_4 are developed. Finally, as shown in [Wie (2008)], the angular rates of the body relative to the inertial frame can be expressed as:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \frac{1}{cos(\theta)} \begin{bmatrix} cos(\theta) & sin(\phi)sin(\theta) & cos(\phi)sin(\theta) \\ 0 & cos(\phi)cos(\theta) & -sin(\phi)cos(\theta) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \tag{2.16}$$

2.2.1 Simplified Quadrotor Dynamics

These nonlinear quadrotor dynamics can now be simplified for utilization in the tracking control law development. This simplification process requires the assumption that ϕ, θ, and ψ remain small. This is a well studied method, for which the result is only used for the control law development. The nonlinear dynamics are still used as the simulation plant for the results presented in Chapter 4. Making the assumption that $\cos(\cdot) = 1$, $\sin(\cdot) = (\cdot)$, and:

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{2.17}$$

the dynamics in equations (2.5), (2.6), (2.7), and (2.16) can be simplified. The simplified equations are more manageable to work with in the control law development and are as follows:

$$\ddot{X} = \theta u_1 \tag{2.18}$$

$$\ddot{Y} = -\phi u_1 \tag{2.19}$$

$$\ddot{Z} = u_1 - g \tag{2.20}$$

$$\ddot{\phi} = J_{yzx}\dot{\theta}\dot{\psi} + J_r u_g \dot{\theta} + u_2 \tag{2.21}$$

$$\ddot{\theta} = J_{zxy}\dot{\phi}\dot{\psi} - J_r u_g \dot{\phi} + u_3 \tag{2.22}$$

$$\ddot{\psi} = u_4. \tag{2.23}$$

2.3 Lyapunov Analysis

One of Aleksandr Lyapunov's main contributions to control theory involves his method of determining stability of nonlinear systems. Lyapunov's stability criteria and theorems play a role in both the translational and rotational control schemes developed in this thesis. In developing these control schemes, Lyapunov's direct (or second) stability theorem is used to prove that the formation trajectory generation control law is effective. This chapter briefly describes Lyapunov's stability criteria and summarizes the results on Lyapunov's second stability method.

Let $\mathbf{x} = (x_1, \ldots, x_n)^T$ denote an n dimensional state vector and consider an autonomous nonlinear dynamical system written in the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) \tag{2.24}$$

where the $\mathbf{f}(\mathbf{x})$ function is considered to be continuously differentiable. Let $\mathbf{x}_e$ denote an equilibrium state, i.e. let

$$\mathbf{f}(\mathbf{x}_e) = 0 \tag{2.25}$$

- The equilibrium state $\mathbf{x}_e$ is said to be *Lyapunov stable* if for any $\varepsilon > 0$ there exists a real positive number $\delta(\varepsilon, t_0)$ such that

$$\|\mathbf{x}(t_0) - \mathbf{x}_e\| \leq \delta(\varepsilon, t_0) \Rightarrow \|\mathbf{x}(t_0) - \mathbf{x}_e\| \leq \varepsilon \tag{2.26}$$

for all $t \geq t_0$ where $\|\mathbf{x}\| \equiv \sqrt{\mathbf{x}^T \mathbf{x}}$.

- The equilibrium state $\mathbf{x}_e$ is said to be *locally asymptotically stable* if it is *Lyapunov stable* as explained above and if

$$\|\mathbf{x}(t_0) - \mathbf{x}_e\| \leq \delta \Rightarrow \mathbf{x}(t) \rightarrow \mathbf{x}_e \tag{2.27}$$

as $t \rightarrow \infty$.

Finally, the equilibrium point $\mathbf{x}_e$ is said to be *globally asymptotically stable* if both of the above conditions are met for *any* initial conditions $\mathbf{x}(t_0)$. Essentially, if it can be shown that the control laws presented here provide global asymptotic stability, then starting from *any* initial condition the system will reach the desired equilibrium state.

Proving stability of nonlinear systems with the basic stability definitions and without resorting to local approximations can be quite tedious and difficult. Lyapunov's direct method provides a tool to make rigorous, analytical stability claims of nonlinear systems by studying the behavior of a scalar, energy-like Lyapunov function.

Let $V(\mathbf{x})$ be a continuously differentiable function defined on a domain $D \subset C^n$, which contains the equilibrium state $\mathbf{x}_e$. Then we have the following definitions:

- $V(\mathbf{x})$ is said to be positive definite if $V(\mathbf{x}_e) = 0$ and

$$V(\mathbf{x}_e) > 0 \forall \mathbf{x} \in D - \mathbf{x}_e \tag{2.28}$$

- $V(\mathbf{x})$ is positive semidefinite in the same domain if

$$V(\mathbf{x}) \geq 0 \forall \mathbf{x} \in D \tag{2.29}$$

Negative definite and negative semidefinite are defined as: if $-V$ is positive definite or if $-V$ is positive semidefinite, respectively

2.4 Lyapunov's Second Stability Theorem

Consider a dynamical system and assume that $\mathbf{x}_e$ is an isolated equilibrium state. If a positive-definite scalar function $V(\mathbf{x})$ exists in a region D around the equilibrium state $\mathbf{x}_e$, with continuous first partial derivatives with respect to $\mathbf{x}$, where the following conditions are met:

1. $V(\mathbf{x}) > 0$ for all $\mathbf{x} \neq \mathbf{x}_e$ in D, $V(\mathbf{x}_e) = 0$.

2. $\dot{V}(\mathbf{x}) \leq 0$ for all $\mathbf{x} \neq \mathbf{x}_e$ in D.

then the equilibrium point is *stable*. Figure 2.2 shows examples of both stable and unstable systems, while Figure 2.3 shows an example of the Lyapunov function for a stable system. If, in addition to 1 and 2,

3 $\dot{V}(\mathbf{x})$ is not identically zero along any solution of the dynamical system *other* than $\mathbf{x}_e$, then the equilibrium point is *locally asymptotically stable*.

If, in addition to 3,

4 there exists in the entire state space a positive-definite function $V(\mathbf{x})$ which is radially unbounded; i.e., $V(\mathbf{x}) \to \infty$ as $||\mathbf{x}|| \to \infty$, then the equilibrium point is *globally asymptotically stable*, i.e. $\mathbf{x}(t) \to \mathbf{x}_e$ as $t \to \infty$ for any initial condition $\mathbf{x}(t_0)$.

Note that conditions 3 and 4 follow directly from LaSalle's invariance principle.

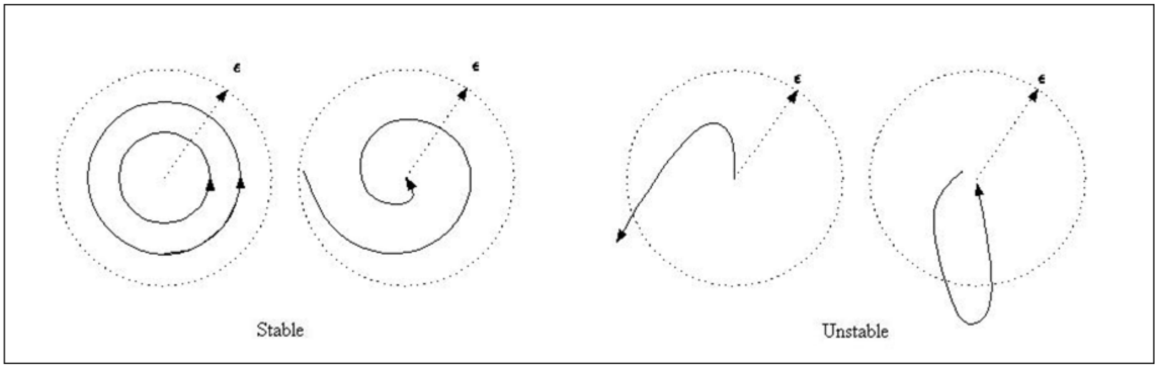

Figure 2.2: Stable and unstable systems.

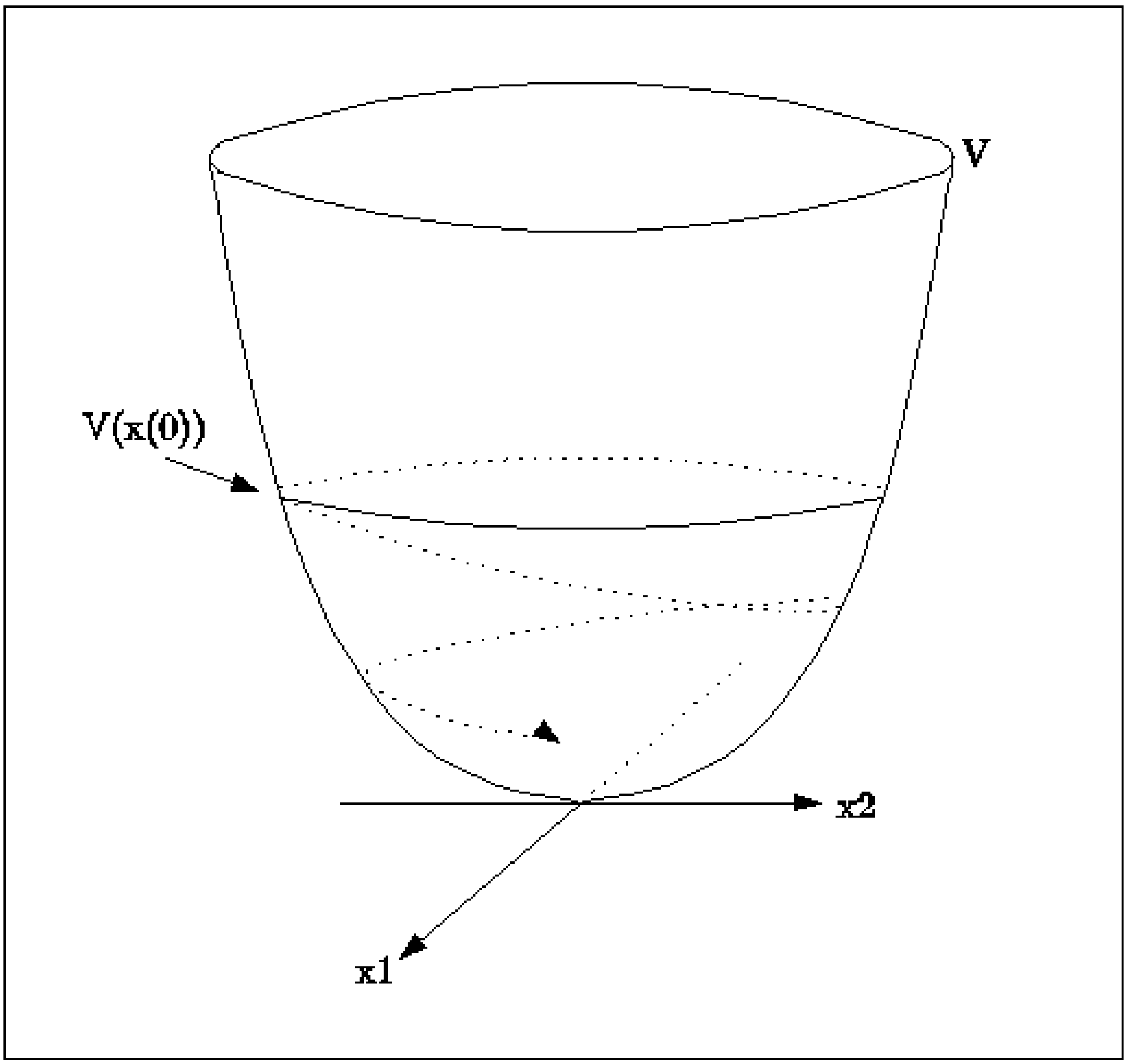

Figure 2.3: Example of a Lyapunov function.

Chapter 3

Leader-Follower Trajectory Generation

In this chapter, a three-dimensional formation trajectory generator is developed. A Lyapunov analysis is used to show stability of the system. Finally, a simulation is shown where a triangular formation "figure-eight" trajectory with staggered vertical positions is generated for arbitrary initial conditions of a simplified model. Successful trajectory generation is achieved. This formulation was adapted from [Roldão et al. (2014)]. The current design is shown to achieve improved results from the original design.

3.1 Trajectory Generation Design

In this section, a trajectory generator is designed so that a virtual follower can follow the predefined trajectory of the virtual leader of the swarm. This trajectory generation allows the virtual follower to follow at any predefined distance from the virtual leader's trajectory. This formulation can be applied to any number of virtual followers to achieve the desired formation.

3.1.1 Planar Formation

A two-dimensional formation trajectory generator is developed first. A simplified model is used with capability of two actuations, planar thrust and torque. The planar thrust generates velocity in the x-direction and y-direction, while the planar torque generates rotation in the plane. Rotations in the plane about the angle ψ can be achieved using the rotation matrix in equation (3.1).

$$\mathscr{R} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix} \tag{3.1}$$

Using this idea, the position of the leader with respect to a follower can be expressed as:

$$^{F}\mathbf{p}_L = \mathscr{R}^T(\mathbf{p}_L - \mathbf{p}_F) \tag{3.2}$$

where $^{F}\mathbf{p}_L$ is the distance between the leader and the follower, $\mathbf{p}_L$ is the position of the leader, and $\mathbf{p}_F$ is the position of the follower.

The goal for the trajectory generator is to drive this distance to some desired distance vector, which is represented as such:

$$\mathbf{d} = \begin{bmatrix} d_x \\ d_y \end{bmatrix} \tag{3.3}$$

where d_x is the x-direction distance between the leader and the follower and d_y is the y-direction distance between the leader and the follower.

As in [Roldão et al. (2014)] the kinematics of a simplified system, such as that described above, can be expressed as:

$$\dot{\mathscr{R}} = \mathscr{R}S(r) \tag{3.4}$$

$$\dot{\mathbf{p}}_F = \mathscr{R}\begin{bmatrix} u \\ 0 \end{bmatrix} \tag{3.5}$$

where u is the linear speed of the follower, r is the angular speed of the follower, and $S(r)$ is a skew-symmetric matrix given by $S(r) = \begin{bmatrix} 0 & -r \\ r & 0 \end{bmatrix}$.

Additionally, the dynamics of such a simplified system can be expressed as:

$$\dot{u} = T \tag{3.6}$$

$$\dot{r} = \tau \tag{3.7}$$

where $\dot{u}$ is the linear acceleration of the follower, $\dot{r}$ is the angular acceleration of the follower, T is the thrust of the follower, and τ is the torque of the follower.

The control scheme is developed using the error dynamics of the system. The first error coordinate is shown in equation (3.8), designed with the goal of driving the actual distance between the leader and follower to the predefined desired distance between the leader and follower

$$\mathbf{e}_1 = {}^F\mathbf{p}_L - \mathbf{d} \tag{3.8}$$

where $\mathbf{e}_1$ is the first error coordinate.

In the spirit of determining the error dynamics, the time derivative of the first error coordinate must be determined:

$$\dot{\mathbf{e}}_1 = -S(r)(\mathbf{e}_1 + \mathbf{d}) + \mathscr{R}^T \dot{\mathbf{p}}_L - \begin{bmatrix} u \\ 0 \end{bmatrix} \tag{3.9}$$

where the goal is to drive $\dot{\mathbf{e}}_1$ to zero.

A positive definite Lyapunov function is developed, using the first error coordinate, with the intention of driving this error coordinate to the origin, such that the actual distance between the leader and follower converges to the predefined desired distance between the leader and follower:

$$V_1 = \frac{1}{2k_1} \mathbf{e}_1^T \mathbf{e}_1 \tag{3.10}$$

where V_1 is the first Lyapunov function and k_1 is a constant control gain.

In accordance with the Lyapunov analysis, the time derivative of the first Lyapunov function is determined:

$$\dot{V}_1 = \frac{1}{k_1}\mathbf{e}_1^T\left\{ -S(r)(\mathbf{e}_1 + \mathbf{d}) + \mathscr{R}^T\dot{\mathbf{p}}_L - \begin{bmatrix} u \\ 0 \end{bmatrix} \right\}. \tag{3.11}$$

A saturation term is introduced into the first Lyapunov derivative with the first error coordinate as the argument to smooth any spikes in the error:

$$\dot{V}_1 = -\mathbf{e}_1^T\boldsymbol{\sigma}_K(\mathbf{e}_1) + \mathbf{e}_1^T\left[\boldsymbol{\sigma}_K(\mathbf{e}_1) + \frac{1}{k_1}\left\{ -S(r)(\mathbf{e}_1 + \mathbf{d}) + \mathscr{R}^T\dot{\mathbf{p}}_L - \begin{bmatrix} u \\ 0 \end{bmatrix} \right\}\right] \tag{3.12}$$

where $\boldsymbol{\sigma}_K$ is a saturation function such that:

$$\boldsymbol{\sigma}_K(x) = \begin{bmatrix} \sigma_K(x_1) \\ \sigma_K(x_2) \end{bmatrix} \tag{3.13}$$

$$\sigma_K(0) = 0 \tag{3.14}$$

$$x\sigma_K(x) > 0 \text{ for all } x \neq 0 \tag{3.15}$$

$$\lim_{x\to\pm\infty} \sigma_K(x) = \pm K \text{ for some } K > 0 \tag{3.16}$$

The following approximation was chosen as the saturation function in accordance with the saturation function properties given by equations (3.13), (3.14), and (3.16):

$$\sigma_K(x) \approx K\frac{x}{|x| + \varepsilon} \tag{3.17}$$

$$\dot{\sigma}_K(x) \approx K\frac{\varepsilon}{(|x| + \varepsilon)^2}\dot{x} \tag{3.18}$$

where ε is a constant governing the steepness of the saturation function.

In the spirit of a backstepping approach, a second error coordinate is developed:

$$\mathbf{e}_2 = \boldsymbol{\sigma}_K(\mathbf{e}_1) + \frac{1}{k_1}\left\{ -S(r)\mathbf{d} + \mathscr{R}^T\dot{\mathbf{p}}_L - \begin{bmatrix} u \\ 0 \end{bmatrix}) \right\} \tag{3.19}$$

where $\mathbf{e}_2$ is the second error coordinate.

As in equation (3.9), the derivative of the second error coordinate is found because it is needed in the Lyapunov analysis. The second error coordinate is given by:

$$\dot{\mathbf{e}}_2 = \dot{\boldsymbol{\sigma}}_K(\mathbf{e}_1) + \frac{1}{k_1}(\boldsymbol{\delta} - \boldsymbol{\Gamma}\boldsymbol{\mu}) + \mathbf{b} \tag{3.20}$$

where $\mathbf{b}$ is a two-dimensional unknown constant disturbance and

$$\boldsymbol{\Gamma} = \begin{bmatrix} 1 & -d_y \\ 0 & d_x \end{bmatrix}, \qquad \boldsymbol{\mu} = \begin{bmatrix} T \\ \tau \end{bmatrix}, \qquad \boldsymbol{\delta} = -S(r)\mathscr{R}^T\dot{\mathbf{p}}_L + \mathscr{R}^T\ddot{\mathbf{p}}_L, \tag{3.21}$$

A second Lyapunov function is constructed, using the second error coordinate and, as the backstepping procedure dictates, the first Lyapunov function. The second Lyapunov function is designed as such:

$$V_2 = V_1 + \frac{1}{2k_2}\mathbf{e}_2^T\mathbf{e}_2 \tag{3.22}$$

where V_2 is the second Lyapunov function and k_2 is a constant control gain.

Once again, the time derivative of the Lyapnuov function is taken, and shown as:

$$\dot{V}_2 = -\mathbf{e}_1^T\sigma_K(\mathbf{e}_1) + \mathbf{e}_1^T\mathbf{e}_2 + \frac{\mathbf{e}_2^T}{k_1k_2}(k_1\dot{\sigma}_K(\mathbf{e}_1) + \boldsymbol{\delta} - \boldsymbol{\Gamma}\boldsymbol{\mu} + k_1\mathbf{b}). \tag{3.23}$$

Instead of the typical backstepping procedure, the integral backstepping approach is taken to allow for disturbance rejection of the constant disturbance $\mathbf{b}$. Thus, an integral term is added to the error dynamics as such:

$$\dot{\boldsymbol{\xi}} = \mathbf{e}_2 \tag{3.24}$$

where $\boldsymbol{\xi}$ is the integral term.

This allows for the construction of a third Lyapunov function, using the second Lyapunov function, which by design includes the first Lyapunov function, and the integral term as shown:

$$V_3 = V_2 + \frac{k_3}{2k_2}(\boldsymbol{\xi} - \frac{1}{k_3}\mathbf{b})^T(\boldsymbol{\xi} - \frac{1}{k_3}\mathbf{b}) \tag{3.25}$$

where V_3 is the third Lyapunov function and k_3 is a constant control gain.

As done previously, the time derivative of the Lyapunov function is taken:

$$\dot{V}_3 = \dot{V}_2 + \frac{k_3}{k_2}(\boldsymbol{\xi} - \frac{1}{k_3}\mathbf{b})^T\mathbf{e}_2 \tag{3.26}$$

and expanded as

$$\dot{V}_3 = -\mathbf{e}_1^T\boldsymbol{\sigma}_K(\mathbf{e}_1) + \mathbf{e}_1^T\mathbf{e}_2 + \frac{\mathbf{e}_2^T}{k_1k_2}(k_1\dot{\boldsymbol{\sigma}}_K(\mathbf{e}_1) + \boldsymbol{\delta} - \boldsymbol{\Gamma}\boldsymbol{\mu} + k_1k_3\boldsymbol{\xi}). \tag{3.27}$$

Now, $\boldsymbol{\mu}$ is designed to satisfy the Lyapunov stability criteria. This control law is different from that in [Roldão et al. (2014)] and allows for a proof of asymptotic stability using both the negative semi-definite Lyapunov result and LaSalle's Invariance Principle instead of bounding arguments. The modified control law is shown as:

$$\boldsymbol{\mu} = \boldsymbol{\Gamma}^{-1}(\boldsymbol{\delta} + k_1\dot{\boldsymbol{\sigma}}_K(\mathbf{e}_1) + k_1k_2\mathbf{e}_2 + k_1k_3\boldsymbol{\xi} + k_1k_2\mathbf{e}_1) \tag{3.28}$$

Substituting equation (3.28) into (3.27) the following negative semi-definite Lyapunov result is achieved:

$$\dot{V}_3 = -\mathbf{e}_2^T\mathbf{e}_2 - \mathbf{e}_1^T\boldsymbol{\sigma}_K(\mathbf{e}_1) \tag{3.29}$$

This negative semi-definite result indicates stability of the error dynamics with the chosen control law, $\boldsymbol{\mu}$. For the implementation of this scheme, the error dynamics are simplified with the substitution of the control law term. The error dynamics are shown to be:

$$\dot{\mathbf{e}}_1 = -S(r)\mathbf{e}_1 + k_1\mathbf{e}_2 - k_1\boldsymbol{\sigma}_K(\mathbf{e}_1) \tag{3.30}$$

$$\dot{\mathbf{e}}_2 = -k_2\mathbf{e}_2 - k_3\boldsymbol{\xi}' - k_2\mathbf{e}_1 \tag{3.31}$$

$$\dot{\boldsymbol{\xi}}' = \mathbf{e}_2 \tag{3.32}$$

where $\boldsymbol{\xi}' = \boldsymbol{\xi} - \frac{1}{k_3}\mathbf{b}$ to simplify the notation while including the constant disturbance in the integral term of the error dynamics.

Since the Lyapunov analysis has a negative semi-definite result, LaSalle's Invariance Principle is used to prove asymptotic stability of the error dynamics. From equation (3.29) the only trajectory that results in a solution of $\dot{V}(x) = 0$ is that which includes $\mathbf{e}_2 = 0$ and $\mathbf{e}_1 = 0$. For LaSalle's Invariance Principle to hold, this must imply that $\boldsymbol{\xi} = 0$ is also the result of this solution. Since $\mathbf{e}_1 = 0$ and $\mathbf{e}_2 = 0$, it must also be true that $\dot{\mathbf{e}}_1 = 0$ $\dot{\mathbf{e}}_2 = 0$. Thus, if $\mathbf{e}_1 = 0$, $\mathbf{e}_2 = 0$, $\dot{\mathbf{e}}_1 = 0$, and $\dot{\mathbf{e}}_2 = 0$, the only way that equation (3.31) can be true is if $\boldsymbol{\xi}' = 0$. This concludes that the only solution for which $\dot{V}(x) = 0$ is the trivial solution. LaSalle's Invariance Principle holds and the system is asymptotically stable.

3.1.2 Internal Dynamics

Based on [Roldão et al. (2014)] and trial by simulation, it is not sufficient to use the simple relationship, $\dot{\psi} = r$ since there are an infinite number of solutions to this equation for a given desired distance vector. Thus, a relationship between the linear and angular velocities of the leader and the linear and angular velocities of the follower is necessary.

After convergence of the error dynamics, the following equation holds true:

$$\begin{bmatrix} u \\ \dot{\psi} \end{bmatrix} = \boldsymbol{\Gamma}^{-1}\mathscr{R}^T\dot{\mathbf{p}}_L \tag{3.33}$$

Based on the geometry, let the velocity of the leader be represented as such:

$$\dot{\mathbf{p}}_L = V_L \begin{bmatrix} \cos(\psi_L) \\ \sin(\psi_L) \end{bmatrix} \tag{3.34}$$

where V_L is the linear velocity of the leader and ψ_L indicates the direction of the leader's velocity. Substituting the relationships from equation (3.34) into (3.33), the following internal dynamics are determined.

$$\begin{bmatrix} u \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} V_L \cos(\psi_L - \psi) + \frac{V_L d_y}{d_x} \sin(\psi_L - \psi) \\ \frac{V_L}{d_x} \sin(\psi_L - \psi) \end{bmatrix} \tag{3.35}$$

To prevent the followers from orbiting the leader at the predetermined desired distance, the relationships in equation (3.35) must be used, since they encompass the geometry of the leader's heading with respect to the follower's heading.

3.1.3 Vertical Formation Trajectory Generation Design

To achieve three-dimensional formation trajectory generation, vertical formation trajectory generation must be added to the planar formation trajectory generation. A desired vertical distance between the leader and a follower must be predetermined for any number of followers. The vertical error coordinate is defined for the follower as such:

$$e_z = p_{Lz} - p_{Fz} + d_z \tag{3.36}$$

where e_z is the vertical error coordinate, p_{Lz} is the z-direction position of the leader, p_{Fz} is the z-direction position of the follower, and d_z is the desired vertical distance between the follower and the leader. This is different from the vertical error coordinate presented in [Roldão et al. (2014)] in that a positive d_z will result in the follower trajectory being that distance above the leader and a negative d_z will result in the follower being that distance below the leader. The vertical control law is given in the form of an acceleration in the z-direction as:

$$\ddot{p}_{Fz} = \sigma_K(\ddot{p}_{Lz}) + \sigma_K(k_{z1} e_z + k_{z2} \dot{e}_z) \tag{3.37}$$

where $\ddot{p}_{Fz}$ is the z-direction acceleration of the follower, $\ddot{p}_{Lz}$ is the z-direction acceleration of the leader, $\dot{e}_z$ is the z-direction velocity error between the leader and follower, and k_{z1} and k_{z2} are constant control gains. Using this controller, the error dynamics for the vertical direction are shown to be:

$$\dot{e}_{z1} = e_{z2} \tag{3.38}$$

$$\dot{e}_{z2} = \ddot{p}_{Lz} - \sigma_K(\ddot{p}_{Lz}) - \sigma_K(k_{z1}e_{z1} + k_{z2}e_{z2}) \tag{3.39}$$

where $e_{z1} = e_z$ and $e_{z2} = \dot{e}_z$. By inspection one can conclude that this scheme asymptotically stabilizes the actual vertical distance between the follower and leader to the desired vertical distance between the follower and leader given that $|\ddot{p}_{Lz}| < K$.

3.2 Simulation Results

The formation trajectory generation scheme discussed in Section 3.1 has been simulated in MATLAB. One can see that the control law proposed in Section 3.1 shows improved performance when compared with that of [Roldão et al. (2014)]. The formation trajectory generator has been simulated with the same parameters as those in [Roldão et al. (2014)] with the exception of the control gains, a nonzero constant disturbance, and the addition of the vertical trajectory generation. It is reasonable to assume that the inclusion of a nonzero disturbance would have decreased the performance of the controller. However, with the modified control law, the performance is highly improved, even with the nonzero disturbance. The simulation was run with two followers and the following parameters:

$$\mathbf{p}_L(t) = \begin{bmatrix} 2\cos(0.25t) \\ \sin(0.5t) \\ 3 \end{bmatrix}$$

$$b = 0.5 \qquad \varepsilon = 10000$$

$$\mathbf{d}_{F1} = \begin{bmatrix} 0.35 \\ 0.35 \\ 0.35 \end{bmatrix} \qquad \mathbf{p}_{F1}(0) = \begin{bmatrix} 3 \\ 3 \\ 0 \end{bmatrix}$$

$$u_{F1}(0) = 0.5 \qquad r_{F1}(0) = -0.5 \qquad \psi_{F1}(0) = \frac{3\pi}{2}$$

$$\mathbf{d}_{F2} = \begin{bmatrix} 0.35 \\ -0.35 \\ -0.35 \end{bmatrix} \qquad \mathbf{p}_{F2}(0) = \begin{bmatrix} 3 \\ 1 \\ 0 \end{bmatrix}$$

$$u_{F2}(0) = 0 \qquad r_{F2}(0) = 0.5 \qquad \psi_{F2}(0) = 0$$

$$k_1 = 0.4 \qquad k_2 = 2 \qquad k_3 = 0.005$$

$$k_{z1} = 7500 \qquad k_{z2} = 7500 \qquad K = 5$$

As shown in Figure 3.1, the two followers move in a "figure-eight" trajectory and qualitatively maintain a constant distance from the leader. The virtual followers must compensate for the undesirable nonzero initial conditions to converge to the desired path. A more quantitative visual is presented in Figure 3.2. The position errors converge to zero within 11 s, which is approximately 40% shorter for Virtual Follower 1 and approximately 10% shorter for Virtual Follower 2 than the results in [Roldão et al. (2014)]. Figure 3.3 shows the velocity errors converging to zero within approximately 11 s, which is comparable to the results when the original control law is applied.

The angular positions and angular velocities shown in Figure 3.4 converge in less time than the original result, as well. The angular positions converge between followers after 6.5 s which is approximately 60% shorter than the original result. The angular rates converge after 13 s, which is approximately 30% shorter than the original result.

Figure 3.5 shows the distances between the followers and between each follower and

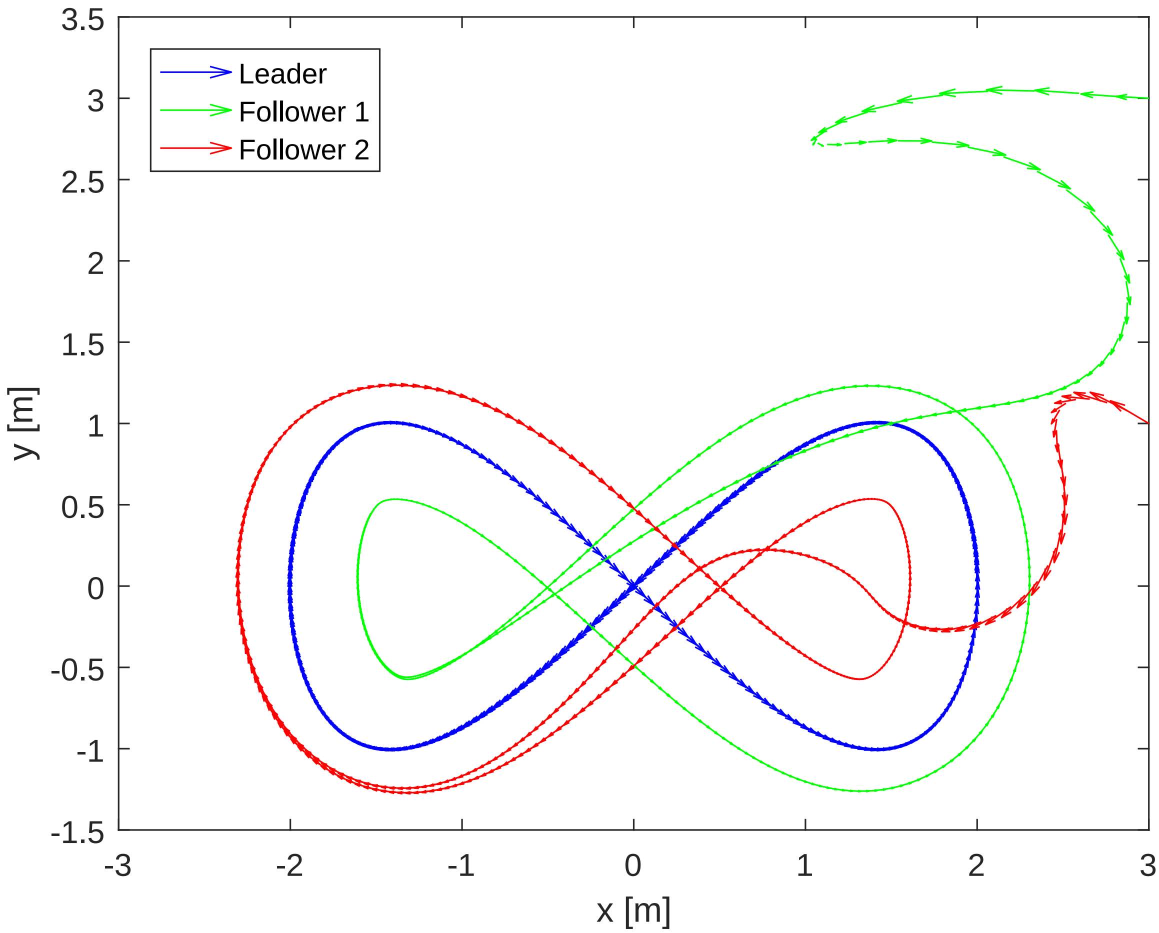

Figure 3.1: Two Virtual Followers Following the Predefined Figure-Eight Virtual Leader Trajectory in a Planar Triangular Formation.

the leader. Each virtual follower converges to the desired distances from the leader, 0.495 m, and a constant distance from each other, 0.7 m, with maximum peak-to-peak variations of 0.018 m, 0.002 m, and 0.013 m for the distances between Virtual Follower 1 and the Virtual Leader, Virtual Follower 2 and the Virtual Leader, and Virtual Follower 2 and Virtual Follower 1, respectively. These variations are expected due to the constant disturbance; however, these values are acceptable being that they are only 3.7%, 0.4%, and 1.9% of their total distance values, respectively. Additionally, the convergence occurs within 11 s, as expected from Figure 3.2. Figure 3.6 shows convergence of the vertical distances between

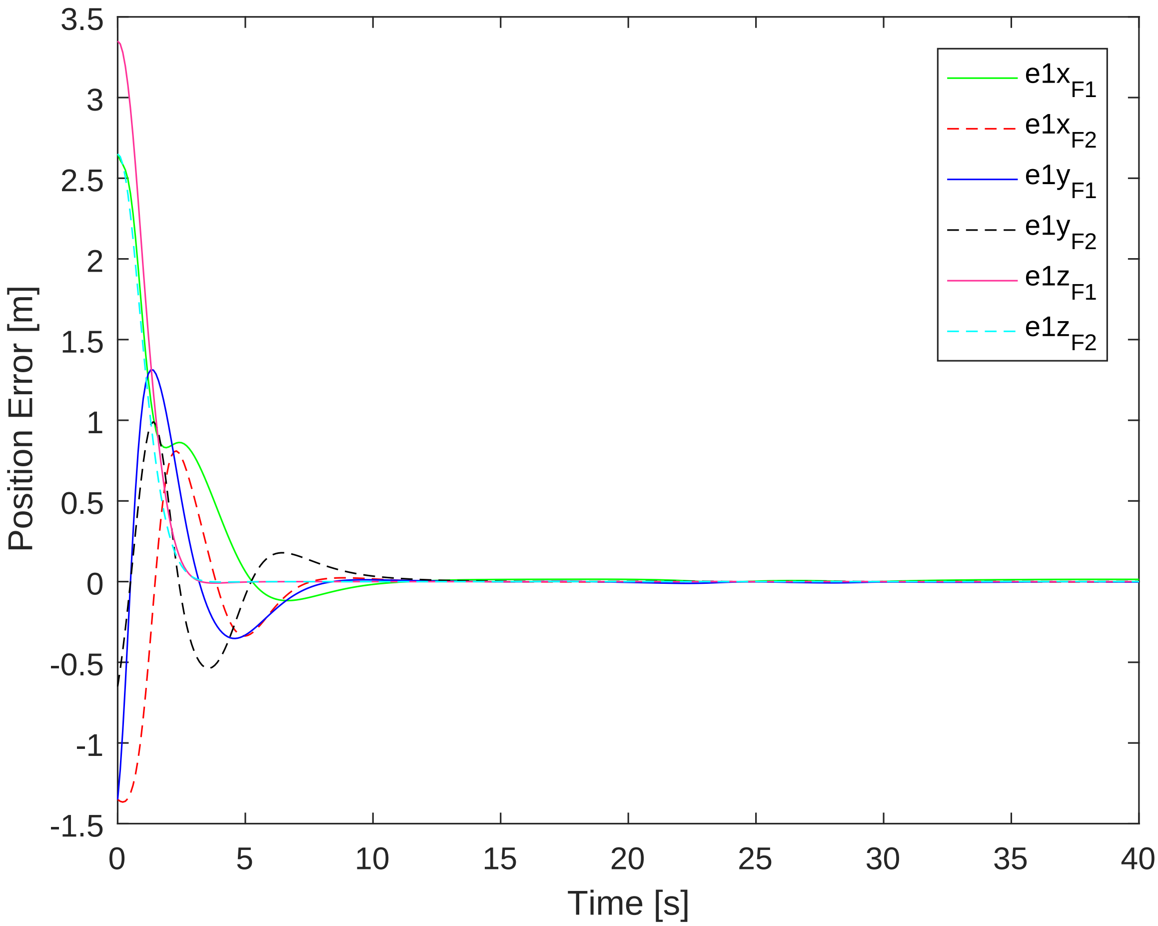

Figure 3.2: Stabilization of Position Error in the x-direction for Virtual Follower 1 (*green*) and Virtual Follower 2 (*red*), y-direction for Virtual Follower 1 (*blue*) and Virtual Follower 2 (*black*), and z-direction for Virtual Follower 1 (*pink*) and Virtual Follower 2 (*cyan*).

Virtual Follower 1 and the Virtual Leader, Virtual Follower 2 and the Virtual Leader, and Virtual Follower 2 and Virtual Follower 1 to 0.35 m, 0.35 m, and 0.7 m, respectively. The convergence occurs in 4 s and has no peak-to-peak variation. This result was not included in the original result. Thus, no comparison can be made. Finally, a three-dimensional view of the trajectory generation is displayed in Figure 3.7. With this, the goal of three-dimensional trajectory generation is achieved and shown to be improved from the original result. It is worth noting that the faster convergences could cause increase thrust and torque requirements from the quadrotors. This concern is explored in the next chapter.

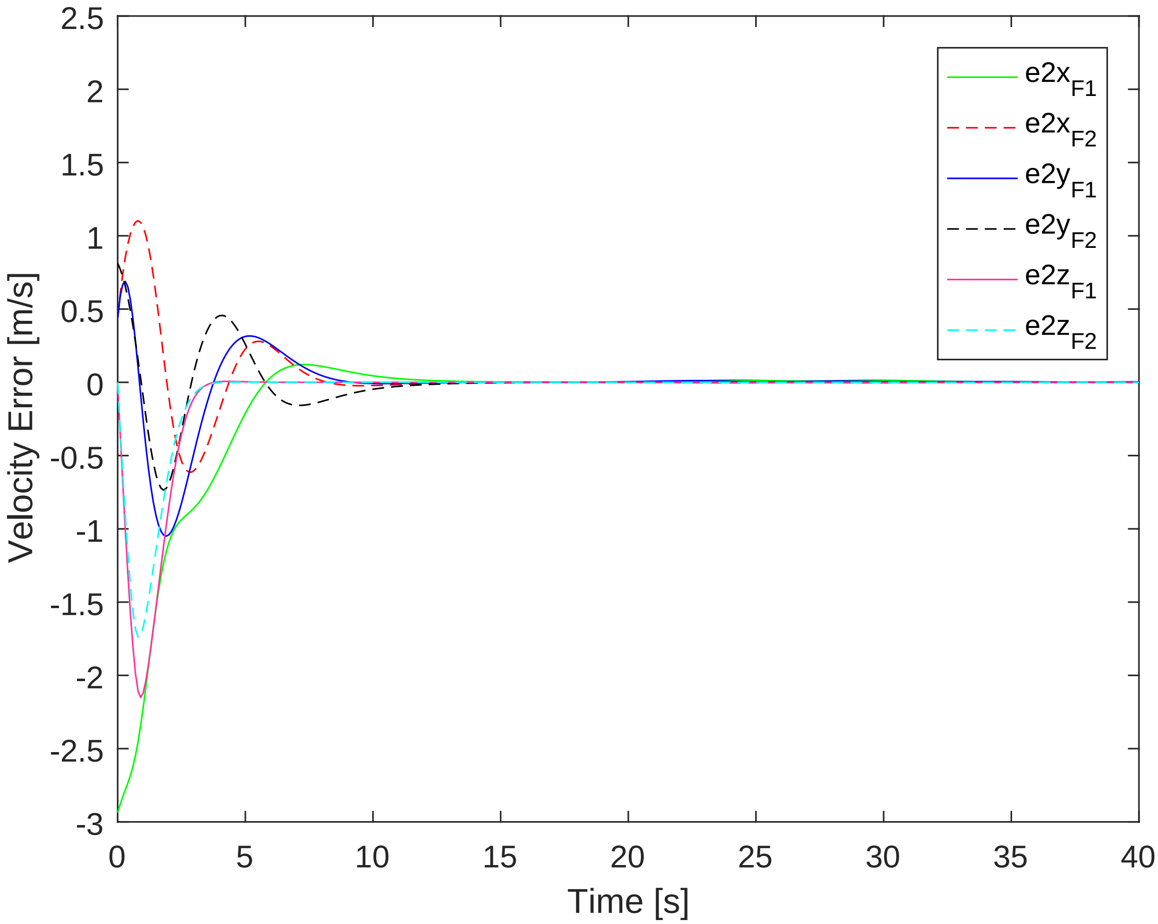

Figure 3.3: Stabilization of Velocity Error in the x-direction for Virtual Follower 1 (*green*) and Virtual Follower 2 (*red*), y-direction for Virtual Follower 1 (*blue*) and Virtual Follower 2 (*black*), and z-direction for Virtual Follower 1 (*pink*) and Virtual Follower 2 (*cyan*).

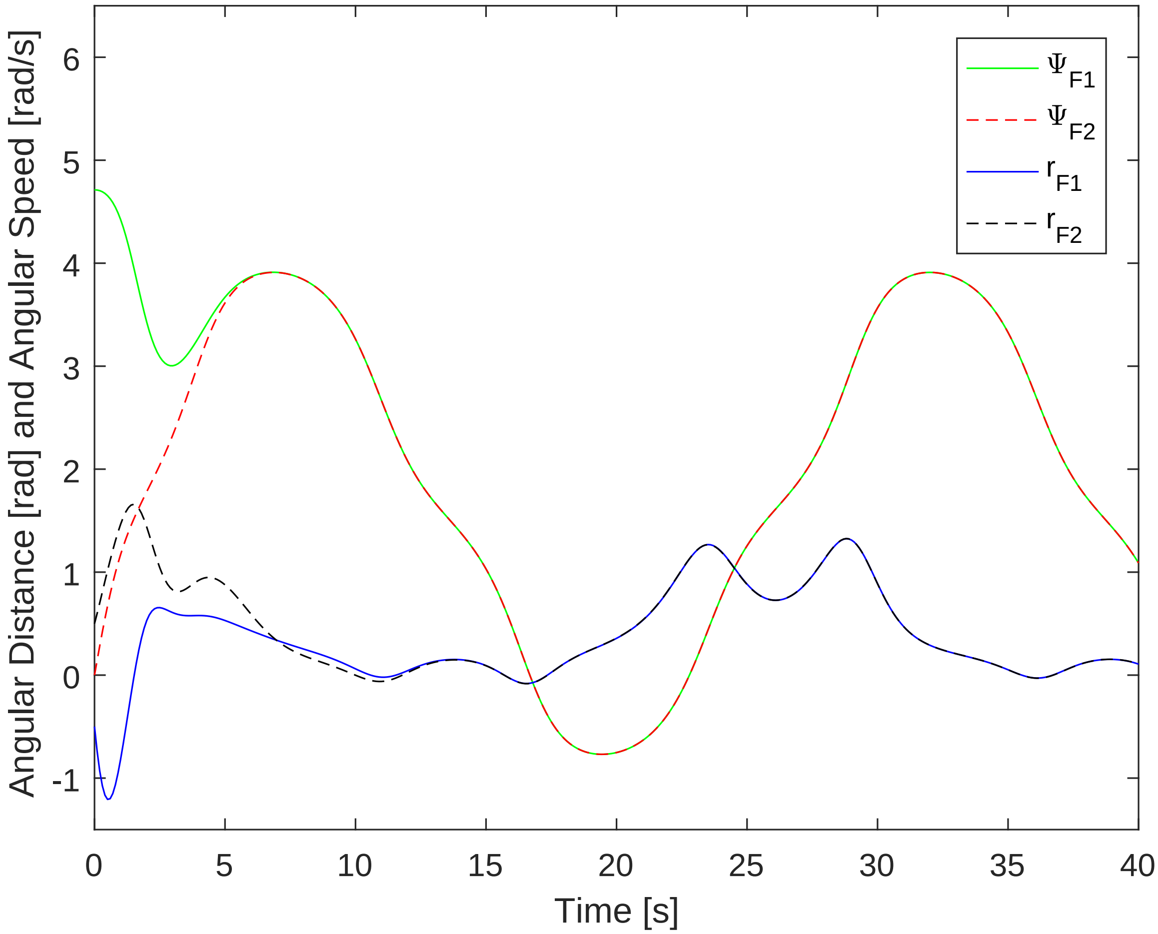

Figure 3.4: Tracking Convergence of Angular Distances (*red*) and (*green*) and Angular Velocities (*blue*) and (*black*) As Both Virtual Followers Follow the Virtual Leader's Figure-Eight.

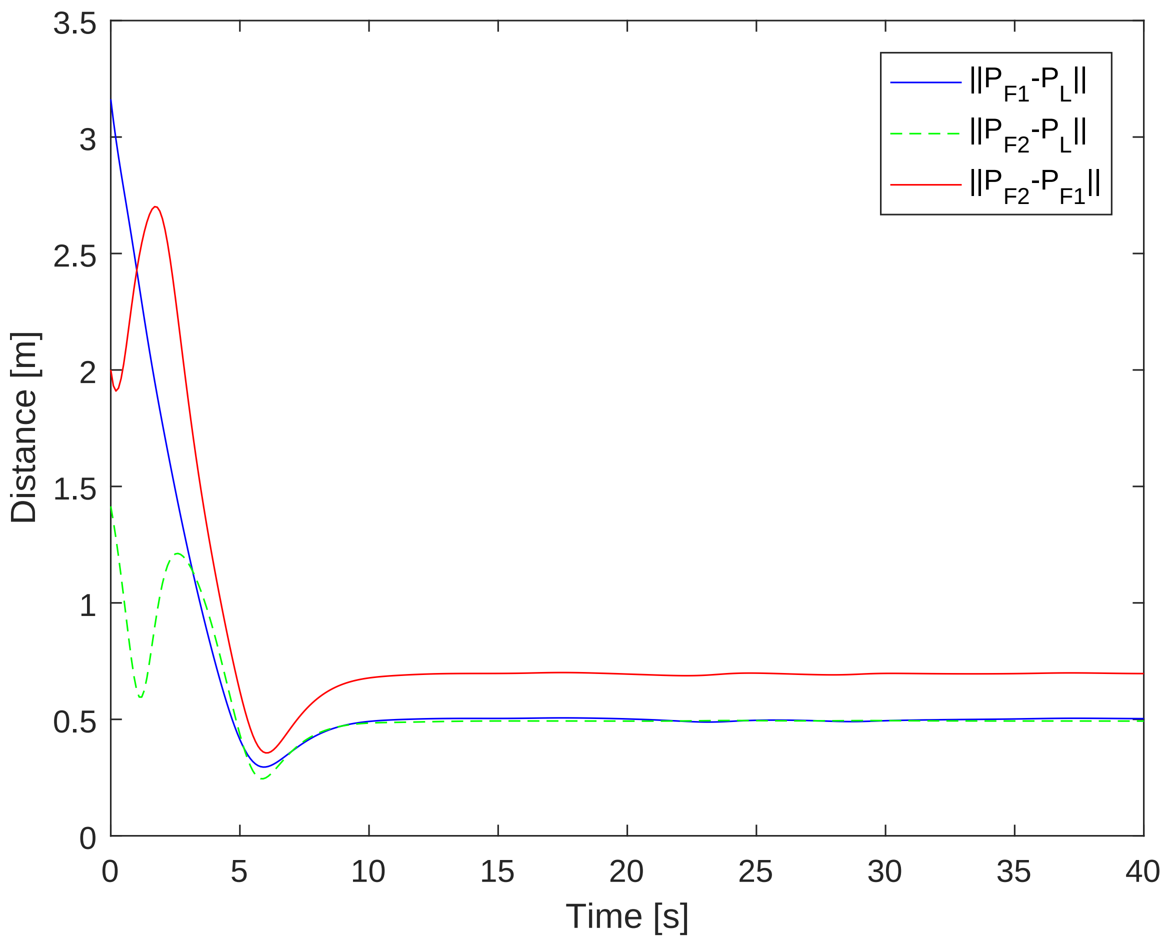

Figure 3.5: Convergence of Triangular Formation Planar Distances Between Virtual Follower 1 and the Virtual Leader (*blue*), Virtual Follower 2 and the Virtual Leader (*green*), and Virtual Follower 2 and Virtual Follower 1 (*red*).

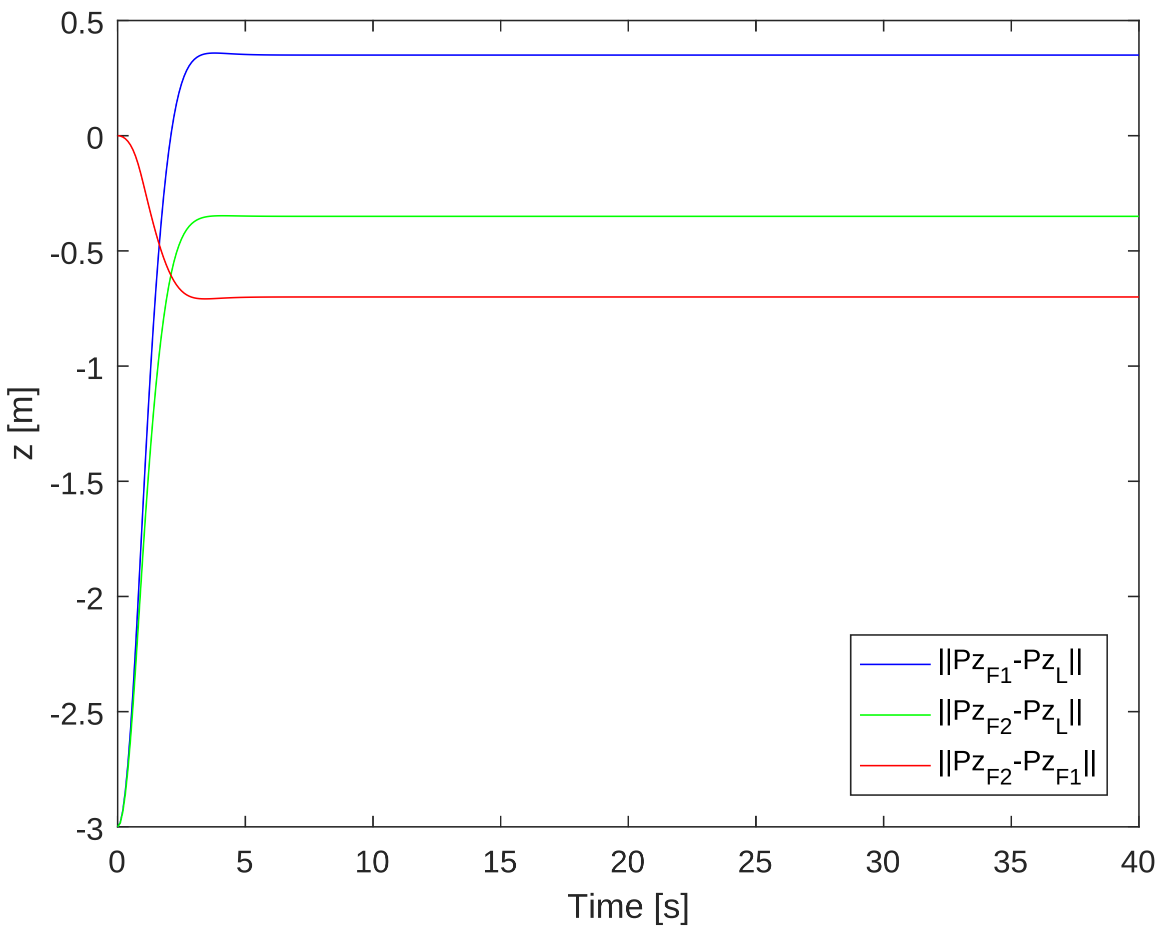

Figure 3.6: Convergence of Distances in the z-direction Between Virtual Follower 1 and the Virtual Leader (*blue*), Virtual Follower 2 and the Virtual Leader (*green*), and Virtual Follower 2 and Virtual Follower 1 (*red*).

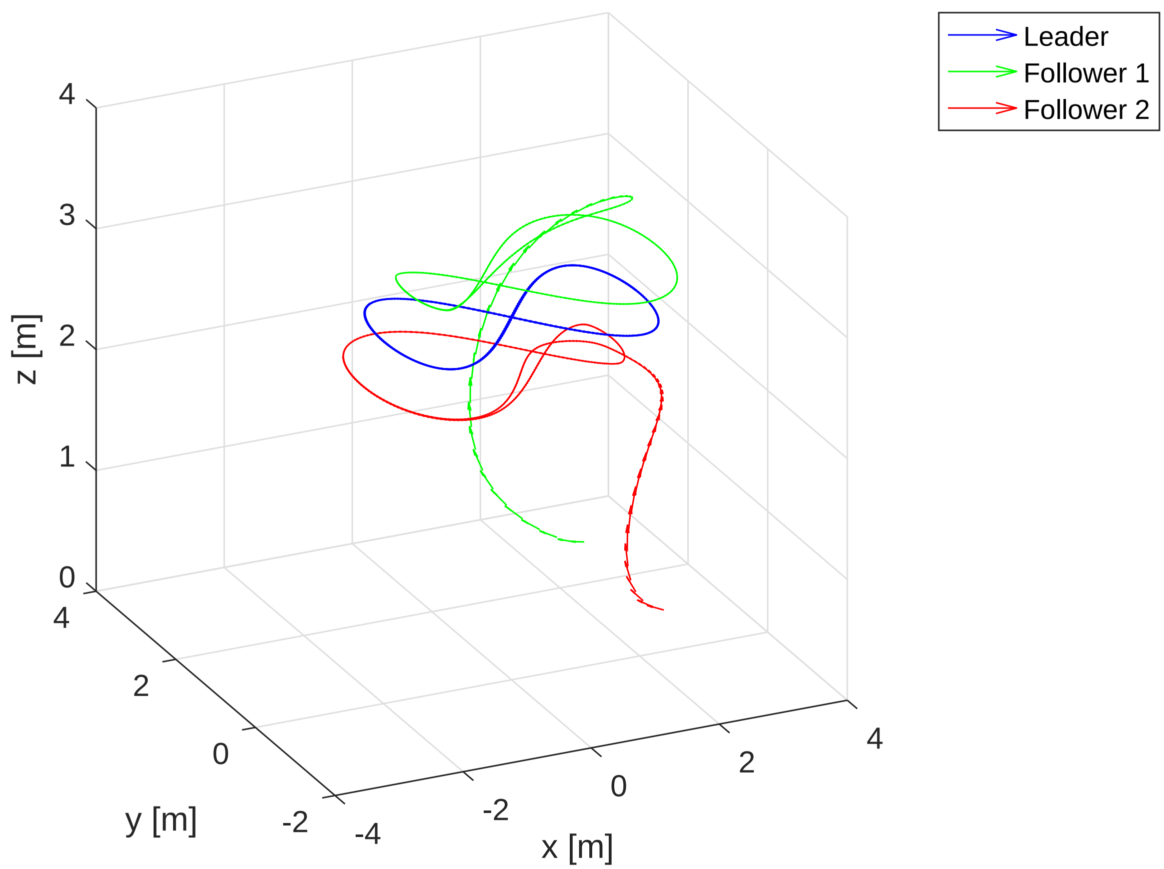

Figure 3.7: Three-Dimensional Representation of Two Virtual Followers Following the Predefined Figure-Eight Virtual Leader Trajectory in a Planar Triangular Formation Offset by Equal Heights.

Chapter 4

Quadrotors Tracking Generated Trajectories

The purpose of this chapter is to develop a nonlinear tracking controller for the quadrotor vehicle. This allows any quadrotor to track a known path. A tracking technique is used because the desired trajectory of the leader is known and the desired trajectories of the followers are generated by the leader-follower trajectory generator. A sliding mode tracking controller is developed in theory, and successful tracking of the leader and two followers is demonstrated in simulation. This formulation is adapted from Dr. Reyhanoglu's point-to-point stabilization sliding mode controller.

4.1 Quadrotor Tracking Controller Design

In this section, the quadrotor tracking controller is designed. The linearized quadrotor dynamics in equations (2.18), (2.19), (2.20), (2.21), (2.22), and (2.23) are used to develop the control law. This control law is then used on the nonlinear quadrotor dynamics in equations (2.5), (2.6), (2.7), and (2.16) for the simulation. The following equations are developed such that the roll and pitch angles are dependent on the position and velocity errors of the coordinate in the direction which that angle generates translational motion, i.e. roll causes motion in the Y-direction and pitch causes motion in the X-direction:

$$\theta = -k_{T1}(X - X_d) - k_{T2}(\dot{X} - \dot{X}_d) \tag{4.1}$$

$$\phi = k_{T3}(Y - Y_d) + k_{T4}(\dot{Y} - \dot{Y}_d) \tag{4.2}$$

where X_d is the desired X-direction position on the trajectory at a particular time instance, $\dot{X}_d$ is the desired X-direction velocity, Y_d is the desired Y-direction position on the trajectory at a particular time instance, $\dot{Y}_d$ is the desired Y-direction velocity, and k_{T1}, k_{T2}, k_{T3}, and k_{T4} are constant, positive control gains.

In the spirit of sliding mode control, two functions are determined such that they encompass the information from equations (4.1) and (4.2), but are also equal to zero. The equations and their derivatives are as follows:

$$y_1 = \theta + k_{T1}(X - X_d) + k_{T2}(\dot{X} - \dot{X}_d) \tag{4.3}$$

$$y_2 = \phi - k_{T3}(Y - Y_d) - k_{T4}(\dot{Y} - \dot{Y}_d) \tag{4.4}$$

$$\dot{y}_1 = \dot{\theta} + k_{T1}(\dot{X} - \dot{X}_d) + k_{T2}(\ddot{X} - \ddot{X}_d) \tag{4.5}$$

$$\dot{y}_2 = \dot{\phi} - k_{T3}(\dot{Y} - \dot{Y}_d) - k_{T4}(\ddot{Y} - \ddot{Y}_d) \tag{4.6}$$

where y_1 and y_2 are the variables that are equal to zero and encompass the information in equations (4.1) and (4.2). After substituting equations (2.18) and (2.19) into (4.5) and (4.6), the following relationships are obtained for use in the sliding surface:

$$\dot{y}_1 = \dot{\theta} + k_{T1}(\dot{X} - \dot{X}_d) + k_{T2}(g\theta - \ddot{X}_d) \tag{4.7}$$

$$\dot{y}_2 = \dot{\phi} - k_{T3}(\dot{Y} - \dot{Y}_d) + k_{T4}(g\phi + \ddot{Y}_d) \tag{4.8}$$

The following sliding surfaces are defined for the system:

$$s_1 = \dot{y}_1 + \alpha_{T1} y_1 \tag{4.9}$$

$$s_2 = \dot{y}_2 + \alpha_{T2} y_2 \tag{4.10}$$

where s_1 and s_2 are the sliding surfaces, and α_{T1} and α_{T2} are constant, positive control gains.

Substituting equations (4.5), (4.7), (4.6) and (4.8) into equations (4.9) and (4.10), the following relationships are obtained:

$$s_1 = \dot{\theta} + k_{T1}(\dot{X} - \dot{X}_d) + k_{T2}(g\theta - \ddot{X}_d) + \alpha_{T1}(\theta + k_{T1}(X - X_d) + k_{T2}(\dot{X} - \dot{X}_d)) \tag{4.11}$$

$$s_2 = \dot{\phi} - k_{T3}(\dot{Y} - \dot{Y}_d) + k_{T4}(g\phi + \ddot{Y}_d) + \alpha_{T2}(\phi - k_{T3}(Y - Y_d) - k_{T4}(\dot{Y} - \dot{Y}_d)). \tag{4.12}$$

In the spirit of sliding mode control, the time derivatives are taken of the sliding surfaces as such:

$$\dot{s}_1 = \ddot{\theta} + k_{T1}(\ddot{X} - \ddot{X}_d) + k_{T2}(g\dot{\theta} - \dddot{X}_d) + \alpha_{T1}(\dot{\theta} + k_{T1}(\dot{X} - \dot{X}_d) + k_{T2}(\ddot{X} - \ddot{X}_d)) \tag{4.13}$$

$$\dot{s}_2 = \ddot{\phi} - k_{T3}(\ddot{Y} - \ddot{Y}_d) + k_{T4}(g\dot{\phi} + \dddot{Y}_d) + \alpha_{T3}(\dot{\phi} + k_{T3}(\dot{Y} - \dot{Y}_d) + k_{T4}(\ddot{Y} - \ddot{Y}_d)) \tag{4.14}$$

Substituting equations (2.21) and (2.22) into (4.13) and (4.13), while neglecting the gyroscopic terms, the following equations are found for the slide surface time derivatives:

$$\begin{aligned} \dot{s}_1 = J_{zxy}\dot{\phi}\dot{\psi} + u_3 + k_{T1}(\ddot{X} - \ddot{X}_d) + k_{T2}(g\dot{\theta} - \dddot{X}_d) \\ + \alpha_{T1}(\dot{\theta} + k_{T1}(\dot{X} - \dot{X}_d) + k_{T2}(\ddot{X} - \ddot{X}_d)) \end{aligned} \tag{4.15}$$

$$\begin{aligned} \dot{s}_2 = J_{yzx}\dot{\theta}\dot{\phi} + u_2 - k_{T3}(\ddot{Y} - \ddot{Y}_d) + k_{T4}(g\dot{\phi} + \dddot{Y}_d) \\ + \alpha_{T2}(\dot{\phi} + k_{T3}(\dot{Y} - \dot{Y}_d) + k_{T4}(\ddot{Y} - \ddot{Y}_d)) \end{aligned} \tag{4.16}$$

An asymptotically stable relationship between the sliding surface and its time-derivative is used with a saturation function for smoothing of the sliding surface. In in this case, the hyperbolic tangent function is used as such:

$$\dot{s}_1 = -\lambda_{T1}\tanh(q_T s_1) \tag{4.17}$$

$$\dot{s}_2 = -\lambda_{T2}\tanh(q_T s_2) \tag{4.18}$$

where λ_{T1} and λ_{T2} are constant, positive control gains and m is a constant that dictates the steepness of the hyperbolic tangent function.

Rearranging equations (4.15) and (4.16) and substituting equations (4.17) and (4.18), the following controls are developed:

$$\begin{aligned} u_3 = -\lambda_{T1}\tanh(q_T s_1) - J_{zxy}\dot{\phi}\dot{\psi} - k_{T1}(\ddot{X} - \ddot{X}_d) - k_{T2}(g\dot{\theta} - \dddot{X}_d) \\ - \alpha_{T1}(\dot{\theta} + k_{T1}(\dot{X} - \dot{X}_d) + k_{T2}(\ddot{X} - \ddot{X}_d)) \end{aligned} \tag{4.19}$$

$$\begin{aligned} u_2 = -\lambda_{T2}\tanh(q_T s_2) - J_{yzx}\dot{\theta}\dot{\phi} + k_{T3}(\ddot{Y} - \ddot{Y}_d) - k_{T4}(g\dot{\phi} + \dddot{Y}_d) \\ - \alpha_{T3}(\dot{\phi} + k_{T3}(\dot{Y} - \dot{Y}_d) + k_{T4}(\ddot{Y} - \ddot{Y}_d)) \end{aligned} \tag{4.20}$$

Additionally, the other two controls are used to asymptotically stabilize the vertical position and velocity and the yaw angle and rate with those of the desired trajectory:

$$u_1 = g - k_{zT1}(Z - Z_d) - k_{zT2}(\dot{Z} - \dot{Z}_d) \tag{4.21}$$

$$u_4 = -l_1(\psi - \psi_d) - l_2(\dot{\psi} - \dot{\psi}_d) \tag{4.22}$$

where k_{zT1}, k_{zT2}, l_1, and l_2 are constant, positive control gains.

Thus, with the use of the controls u_1, u_2, u_3, and u_4, and their asymptotically stable nature, successful quadrotor tracking control has been achieved.

4.2 Quadrotor Formation Tracking Simulation Results

With the tracking controller developed, a leader and two follower quadrotors can be shown to effectively track the predefined virtual leader's trajectory and the two virtual follower generated trajectories presented in Section 3.2. This is done through simulation in MATLAB. The nonlinear quadrotor dynamics presented in Section 2.2 are used as the plant, while the tracking controller presented in Section4.1 is used to drive the quadrotors to the desired paths. To set up the desired paths, the simulation in Section 3.2 is run prior to initiating the tracking controller. The following simulation parameters are used in the simulation for the tracking control portion:

$$k_{T1} = 6.5 \qquad k_{T2} = 0.5 \qquad k_{T3} = 6.5 \qquad k_{T4} = 0.5$$

$$k_{zT1} = 2.5 \qquad k_{zT2} = 4.5$$

$$l_1 = 2 \qquad l_2 = 2$$

$$\lambda_{T1} = 4 \qquad \lambda_{T2} = 4$$

$$\alpha_1 = 3 \qquad \alpha_2 = 3$$

$$J_{xx} = 0.001 kg \cdot m^2 \qquad J_{yy} = 0.001 kg \cdot m^2 \qquad J_{zz} = 0.002 kg \cdot m^2$$

$$q_T = 100 \qquad g = 9.81 \frac{m}{s^2}$$

where the control gains have been tuned for optimum performance and the quadrotor parameters are equal to those in [Roldão et al. (2014)]. Additionally, the quadrotor initial conditions have been chosen to match those of the virtual agent initial conditions presented in Section 3.2. As another note, the third derivative terms for the desired states are ignored due to their small magnitude.

In figures 4.1, 4.2, and 4.3 the paths of the virtual agents and their respective quadrotor counterparts are shown. One can see that all three quadrotors track the desired "figure-eight" trajectories. Figure 4.4 shows all three quadrotors following their respective three-dimensional trajectories. This is the desired result of this thesis. Now, the analysis of the performance of the tracking controller is of importance.

In Figure 4.5, one can see the magnitude of distance between each virtual agent and its respective quadrotor. These distances converge to a desired value of zero, in other words, the quadrotors track the desired trajectories. This convergence takes approximately 11 s. With a maximum peak-to-peak variation of only 0.03 m, 0.03 m, and 0.04 m, for the leader, follower 1, and follower 2, respectively, this result is considered successful. This variation is due to the chattering nature of a sliding mode controller.

Figures 4.6, 4.7, and 4.8 show the x-direction, y-direction, and z-direction positions of all quadrotors, alongside their virtual agent counterparts. One can see convergence of the quadrotors to their desired tracking positions, in approximately 11 s. This result is comparable to that achieved in [Roldão et al. (2014)], where it takes approximately 11 s for the positions to converge. The tracking is smooth with regard to the positions. Additionally, the quadrotor velocities are presented alongside their virtual agent counterpart velocities,

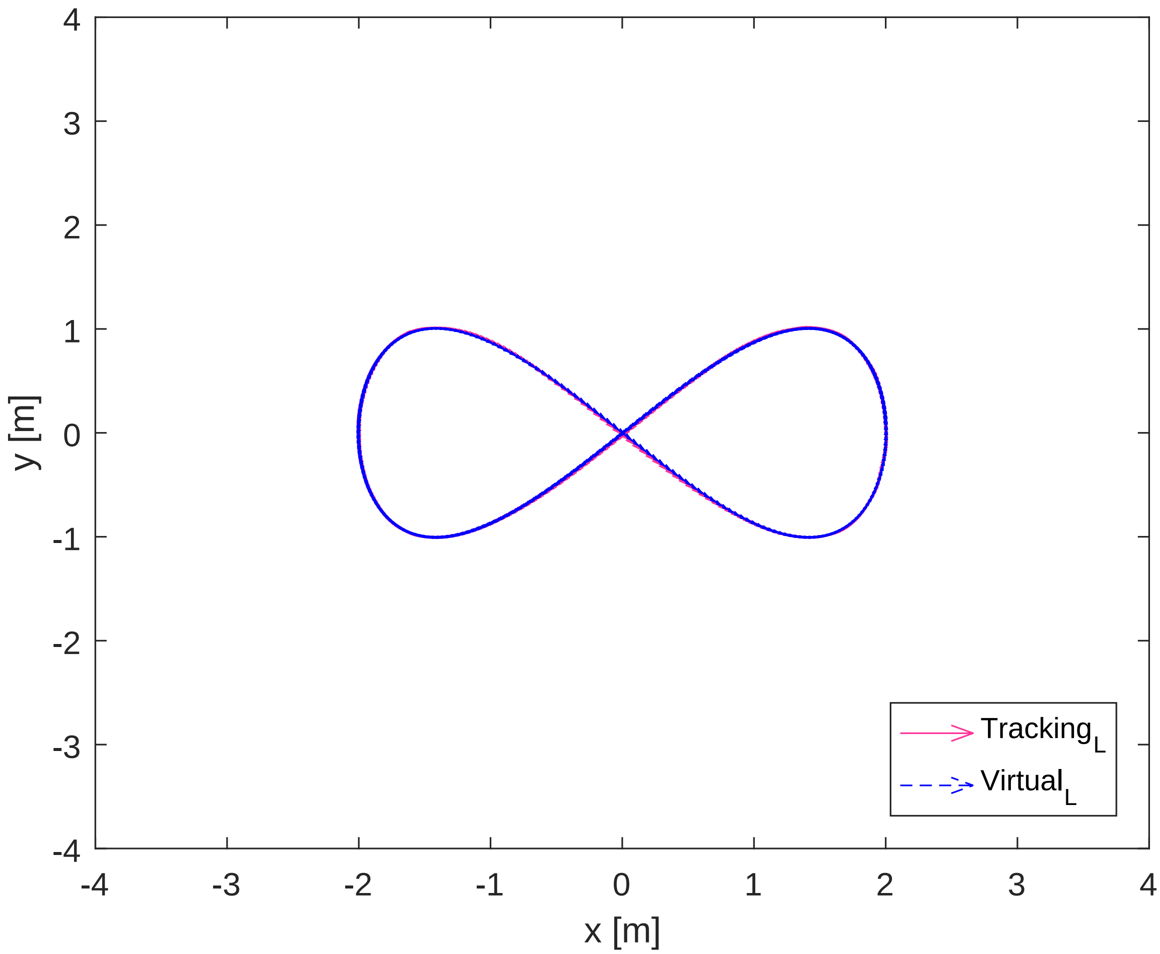

Figure 4.1: Two-Dimensional Representation of Leader Quadrotor Tracking Predefined Virtual Leader Trajectory

in figures 4.9, 4.10, and 4.11. Again, tracking ensues after approximately 11 s. One notices the chattering in the velocity tracking. This is expected due to the nature of sliding mode control. With maximum deviations from the desired velocities of only 0.03 $\frac{m}{s}$, the velocity tracking can be considered a success, especially when considering that there is no noticeable chattering in the position tracking.

Perhaps the most important result is the planar desired distance vector convergence between the leader and followers. This was the goal of this thesis, leader-follower formation flight of three quadrotors at desired distance vectors. The desired vertical distances were

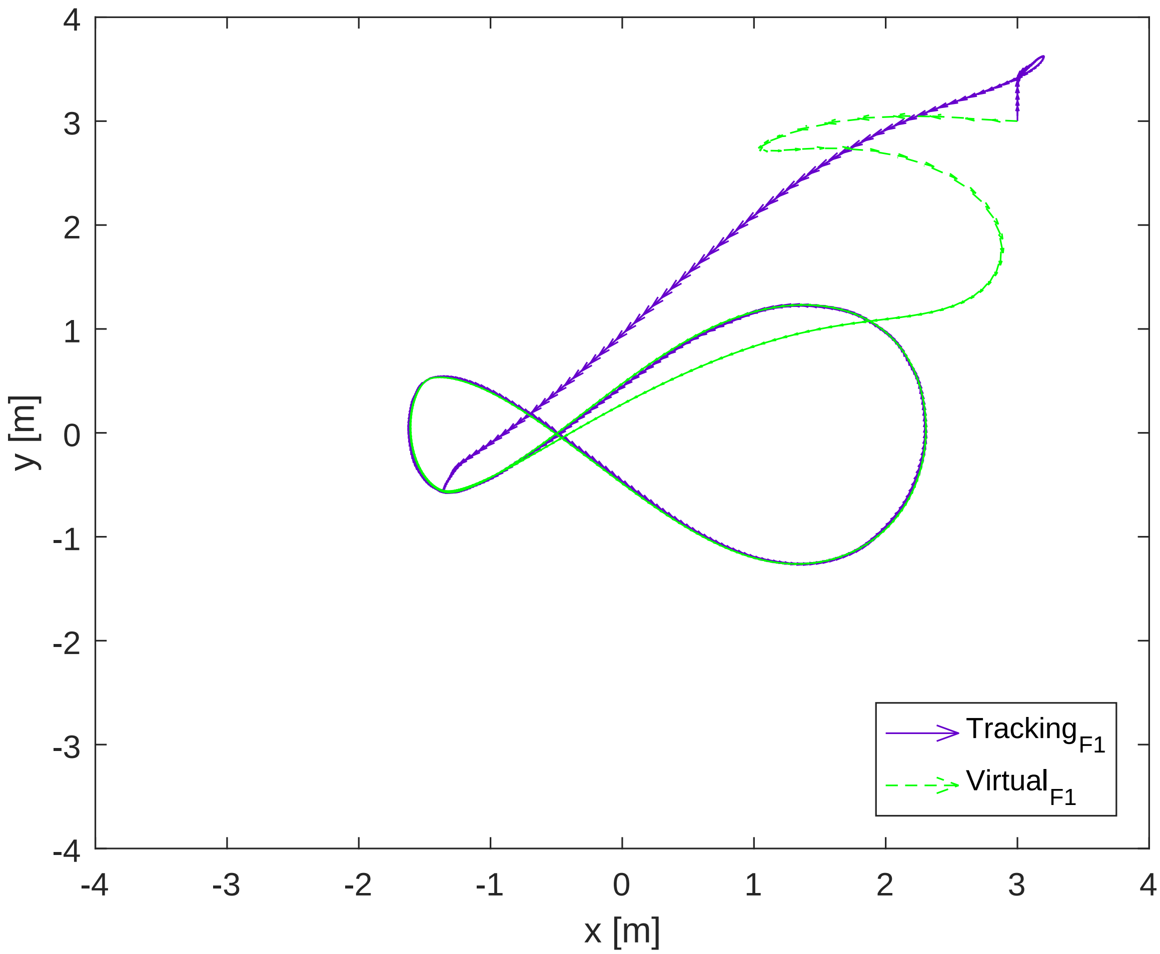

Figure 4.2: Two-Dimensional Representation of Follower 1 Quadrotor Tracking Generated Virtual Follower 1 Formation Trajectory

achieved within 0.001 m, which is practically unnoticeable, as presented in Figure 4.8. The planar distance vectors between the quadrotor leader and quadrotor follower 1, the quadrotor leader and quadrotor follower two, and quadrotor follower 2 and quadrotor follower 1 are presented in Figure 4.12. This shows a highly improved result when compared with that of [Roldão et al. (2014)], where tracking was achieved with peak-to-peak variations within 20% of the total distance vectors. In the result presented in Figure 4.12, maximum peak-to-peak variations of 0.06 m, 0.04 m, and 0.04 m for the planar distance vectors between the quadrotor leader and quadrotor follower 1, the quadrotor leader and quadrotor

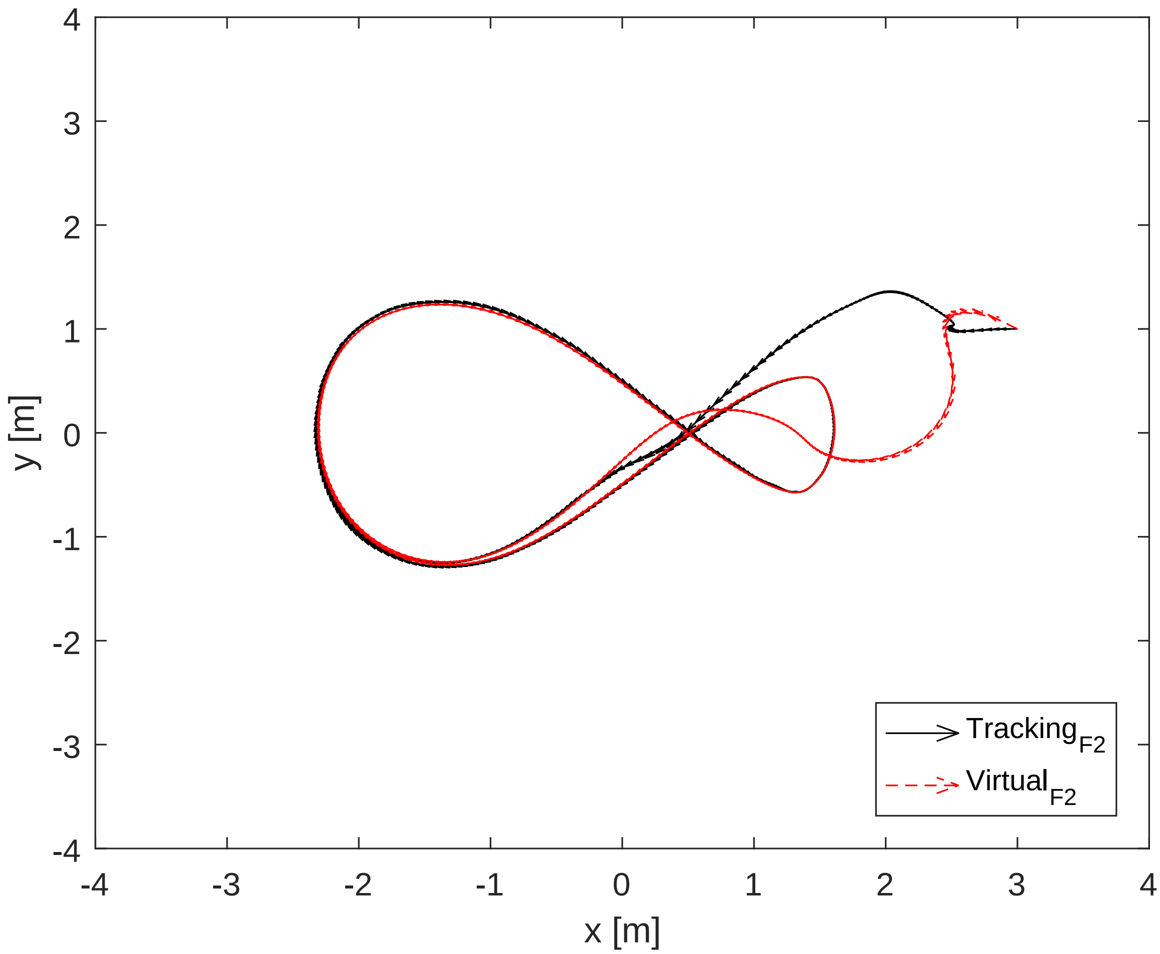

Figure 4.3: Two-Dimensional Representation of Follower 2 Quadrotor Tracking Generated Virtual Follower 2 Formation Trajectory

follower two, and quadrotor follower 2 and quadrotor follower 1, respectively. This corresponds to 12%, 8%, and 6% of the desired distances of 0.495 m, 0.495 m, and 0.7 m for the planar distance vectors between the quadrotor leader and quadrotor follower 1, the quadrotor leader and quadrotor follower two, and quadrotor follower 2 and quadrotor follower 1, respectively. This result is considered to be highly successful, especially when compared with the result achieved in [Roldão et al. (2014)].

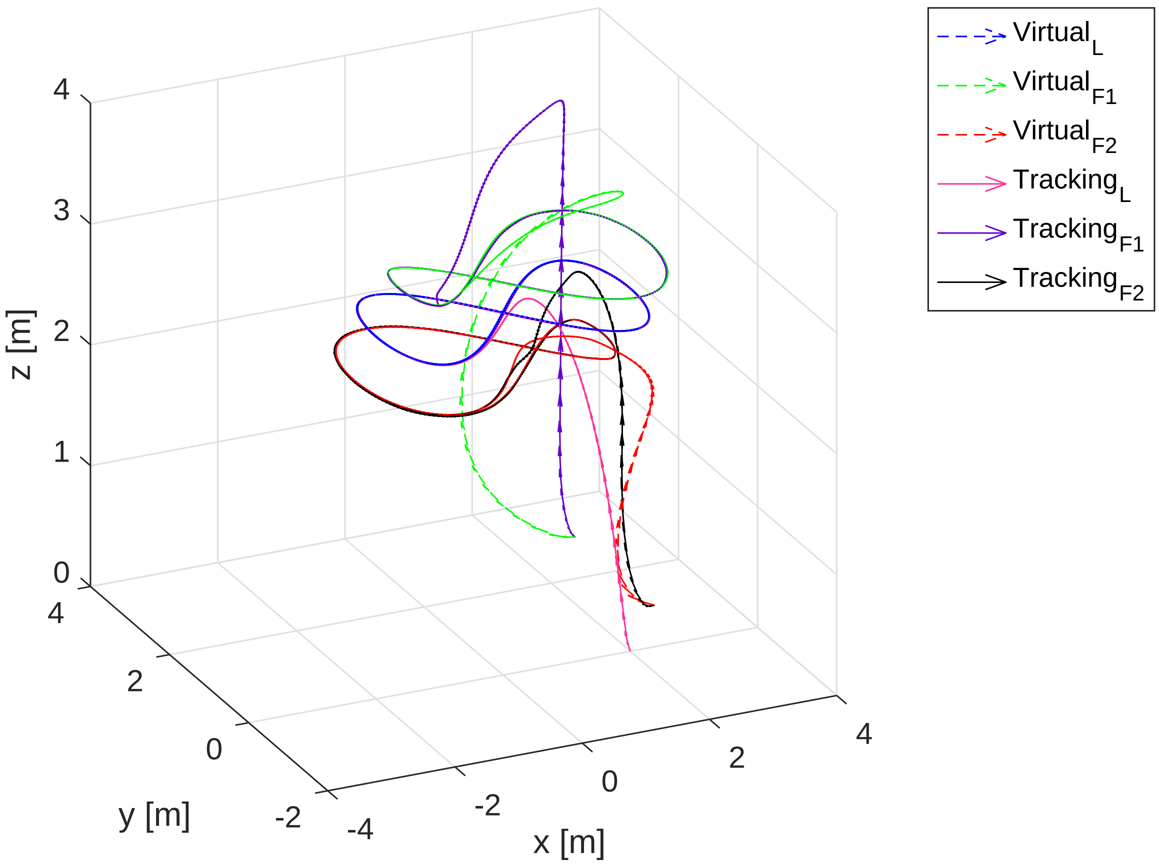

Figure 4.4: One Leader and Two Follower Quadrotors Achieving Three-Dimensional Tracking Control of Generated Formation Flight Trajectories

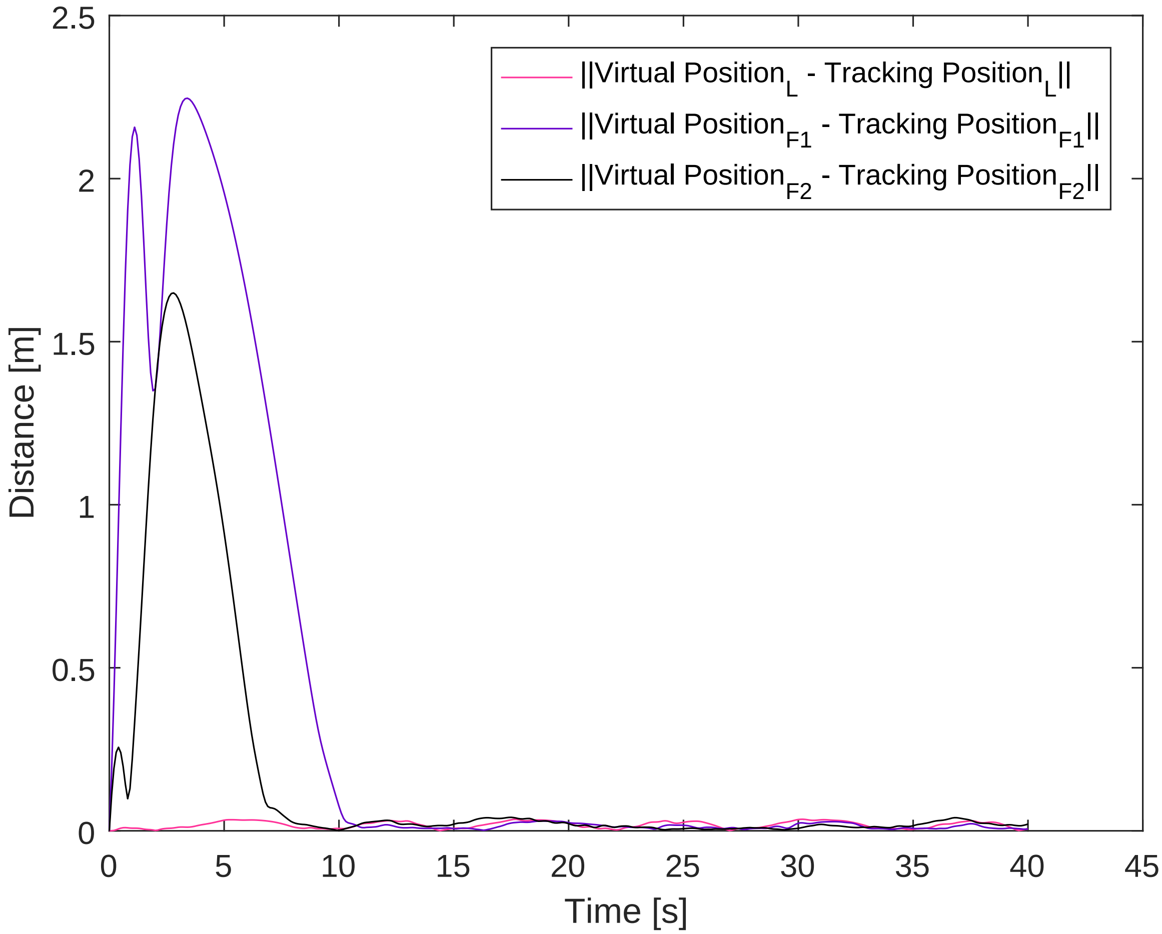

Figure 4.5: Convergence of Planar Distances Between the Virtual Leader and the Quadrotor Leader $(pink)$, Virtual Follower 1 and Quadrotor Follower 1 $(purple)$, and Virtual Follower 2 and Quadrotor Follower 2 $(black)$.

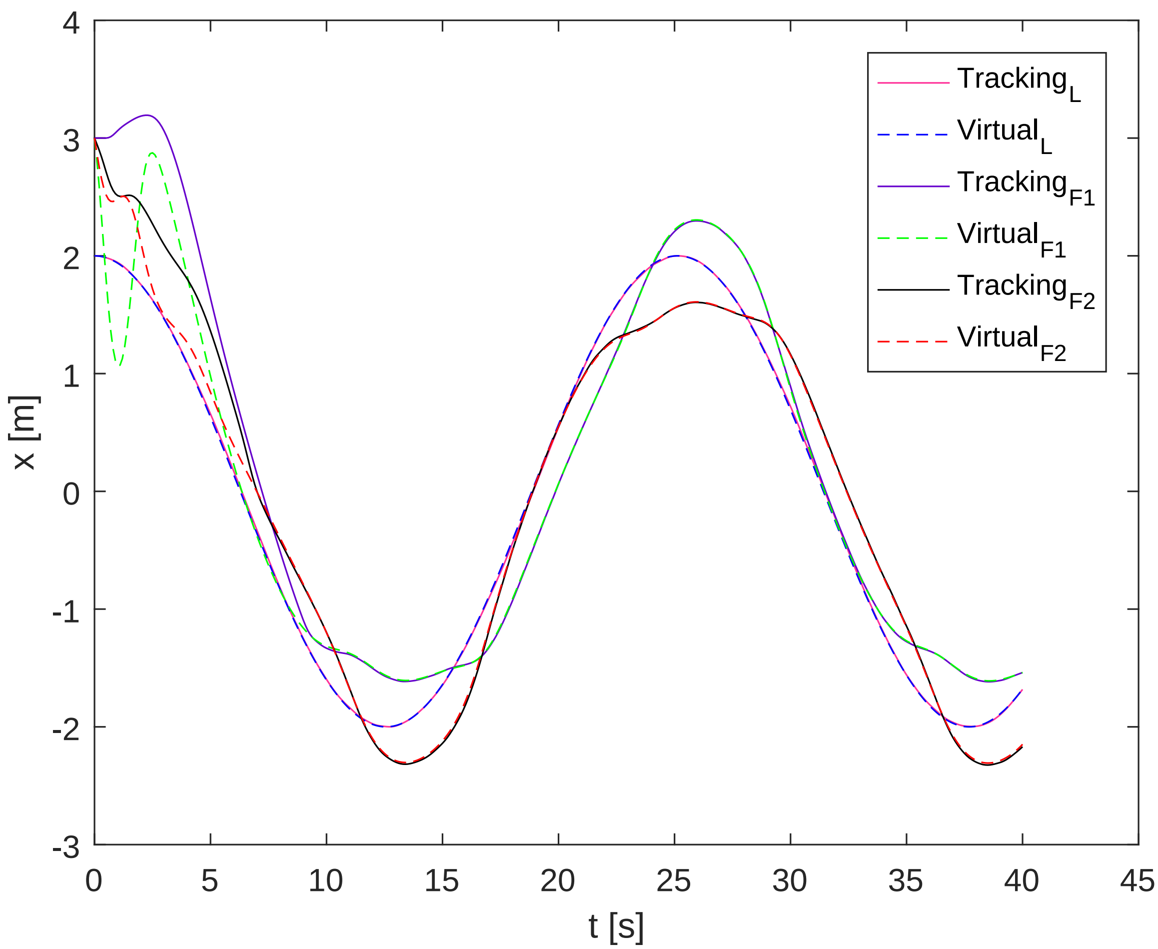

Figure 4.6: Convergence of All Three Quadrotors' X-Direction Positions To Their Respective Virtual Agents' X-Direction Positions

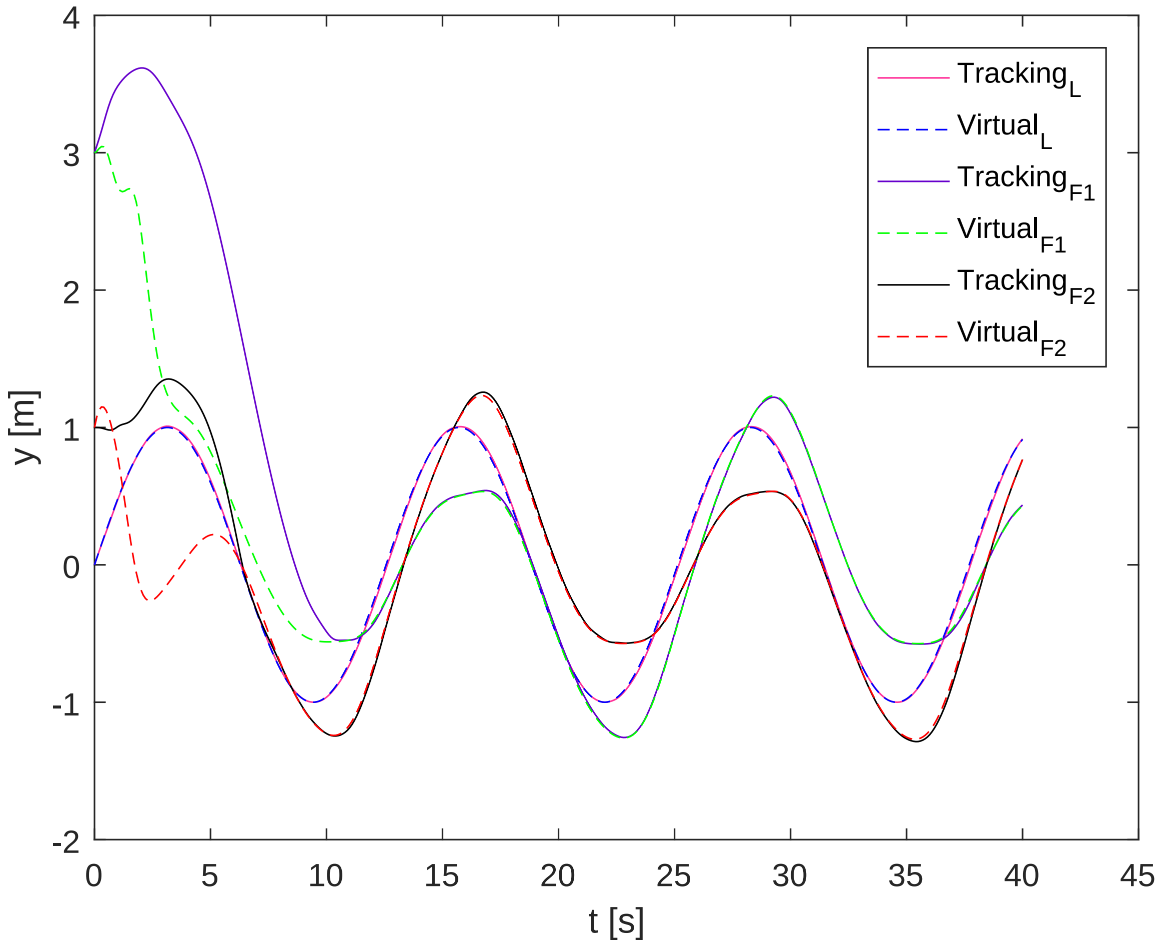

Figure 4.7: Convergence of All Three Quadrotors' Y-Direction Positions To Their Respective Virtual Agents' Y-Direction Positions

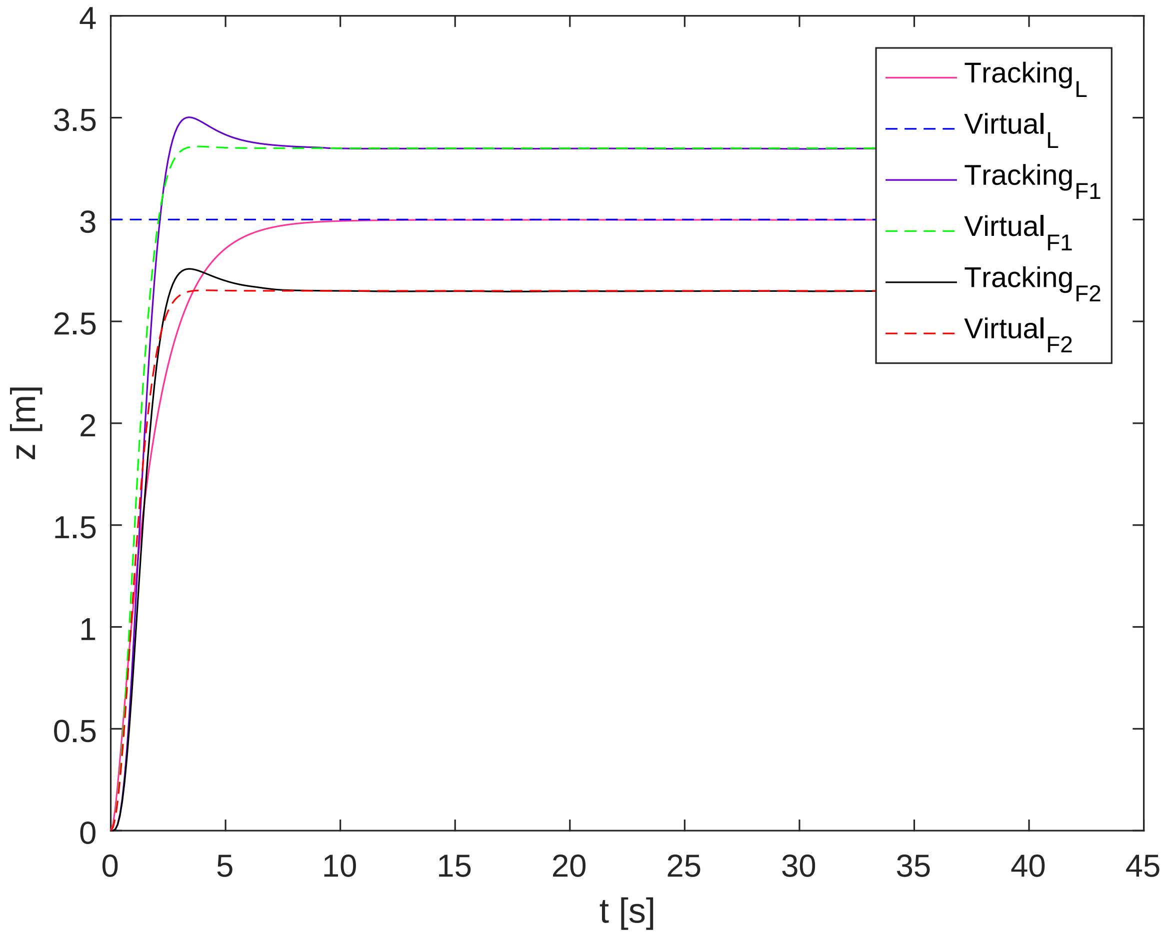

Figure 4.8: Convergence of All Three Quadrotors' Z-Direction Positions To Their Respective Virtual Agents' Z-Direction Positions

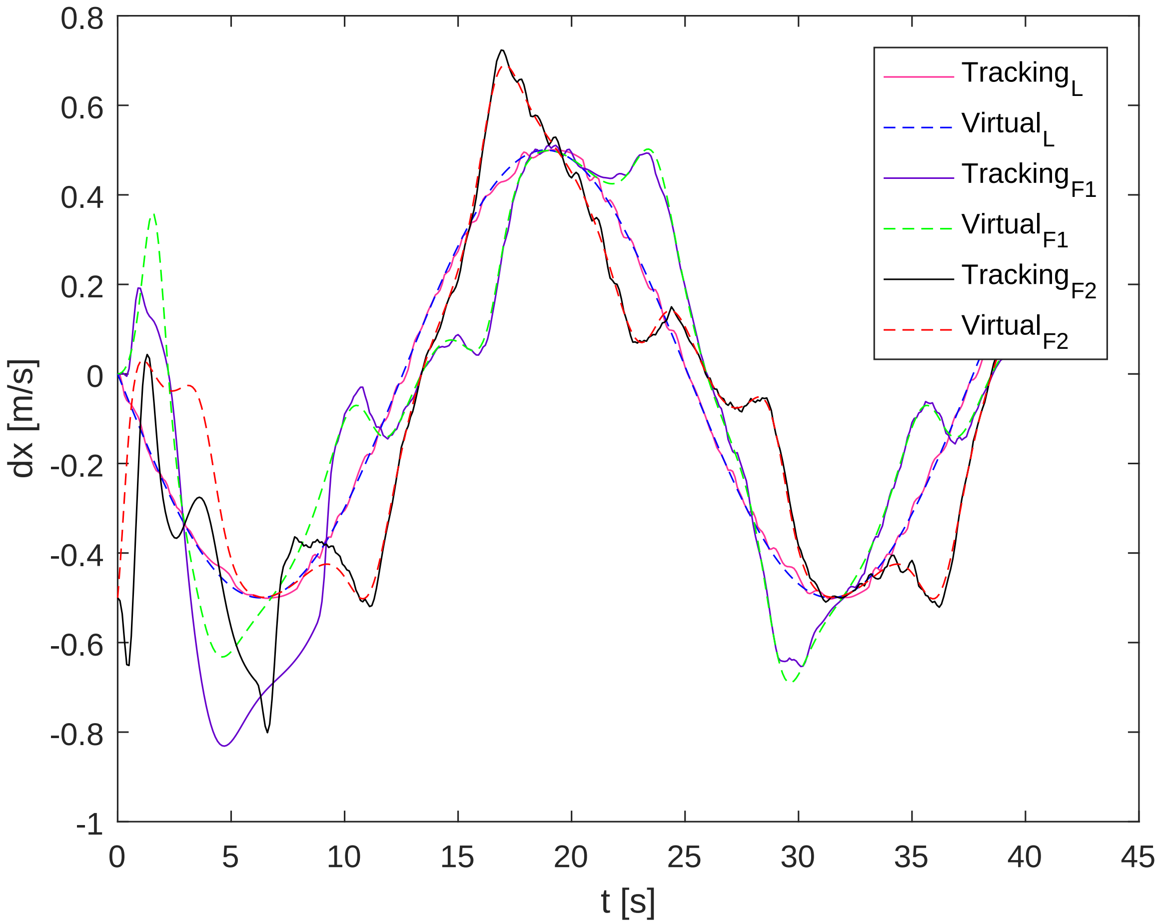

Figure 4.9: Convergence of All Three Quadrotors' X-Direction Velocities To Their Respective Virtual Agents' X-Direction Velocities

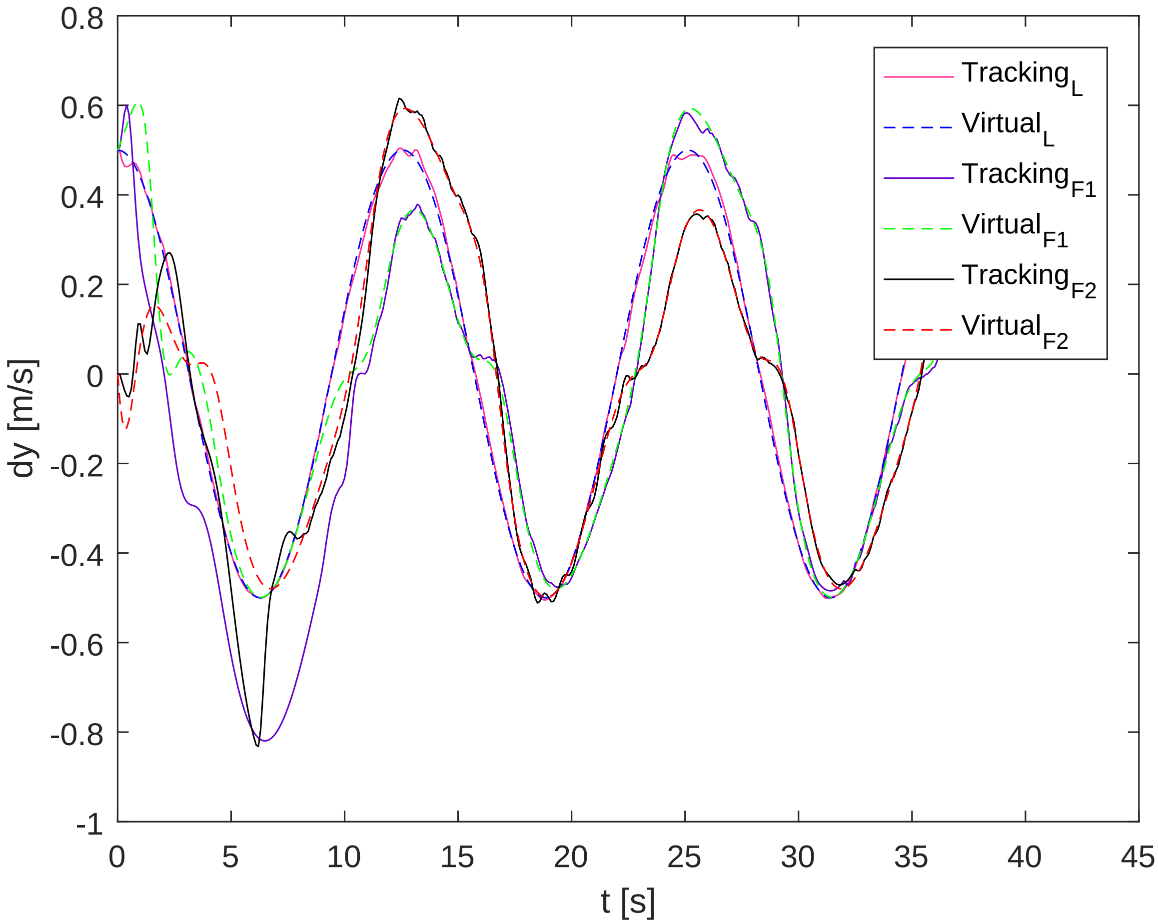

Figure 4.10: Convergence of All Three Quadrotors' Y-Direction Velocities To Their Respective Virtual Agents' Y-Direction Velocities

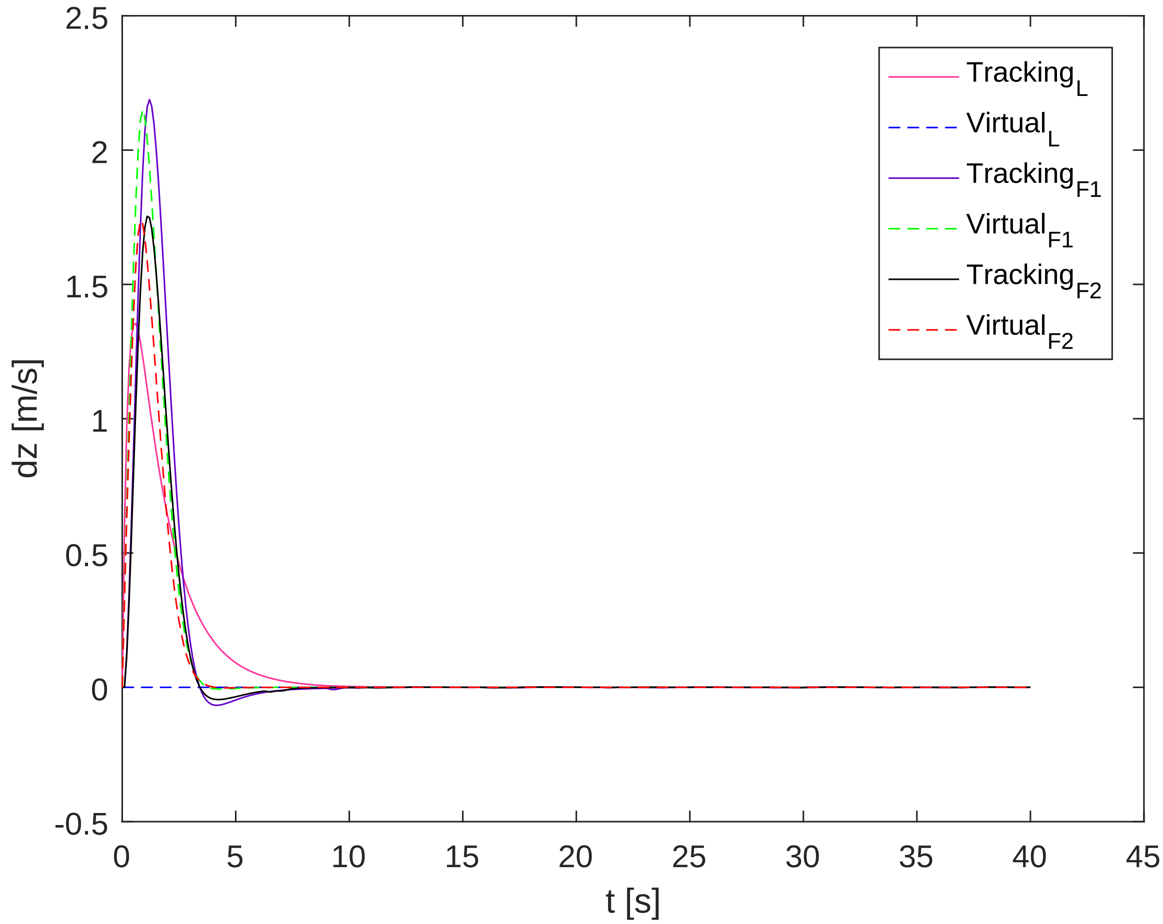

Figure 4.11: Convergence of All Three Quadrotors' Z-Direction Velocities To Their Respective Virtual Agents' Z-Direction Velocities

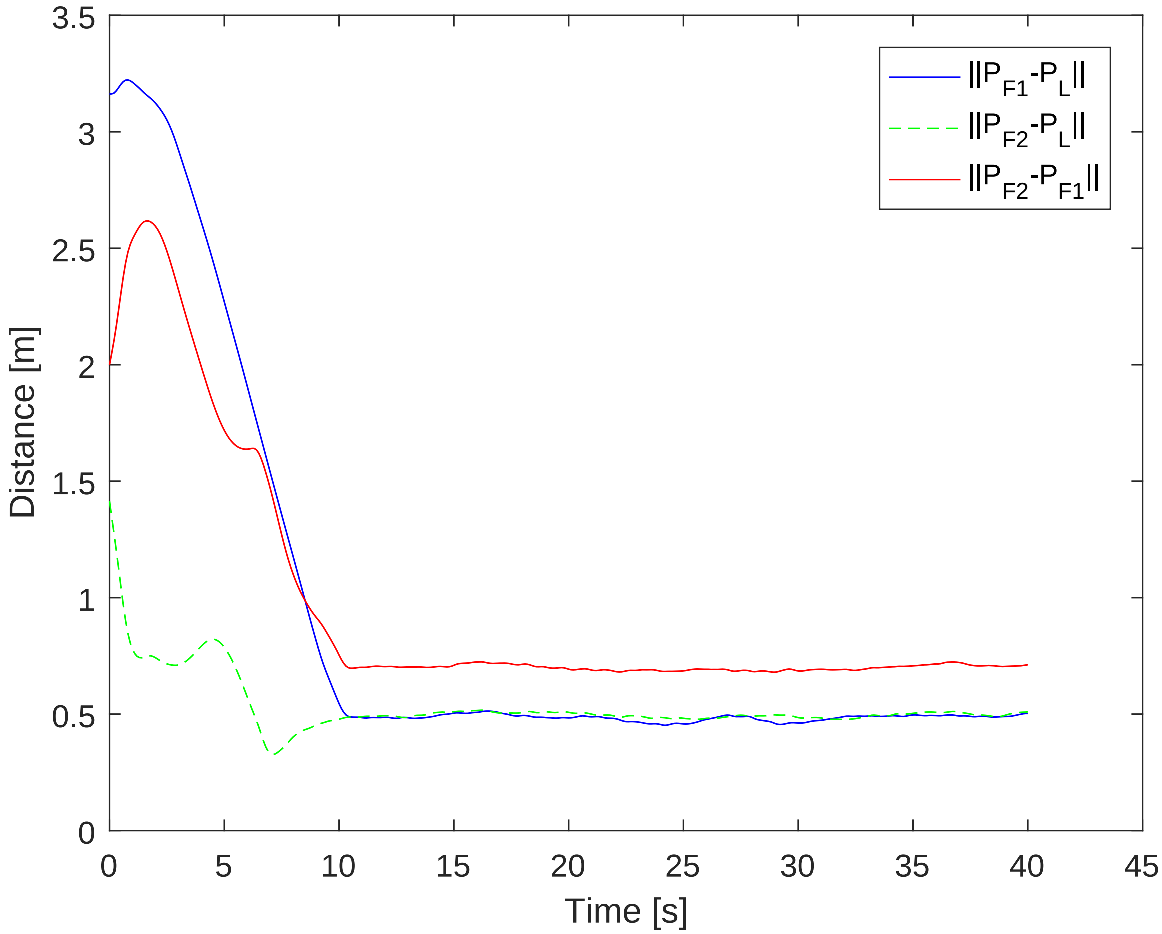

Figure 4.12: Convergence of Planar Distances Between Quadrotor Follower 1 and the Quadrotor Leader (*blue*), Quadrotor Follower 2 and the Quadrotor Leader (*green*), and Quadrotor Follower 2 and Quadrotor Follower 1 (*red*).

Chapter 5

Conclusions and Future Work

A background and motivation for using the leader-follower formation flight scheme was used, along with the importance of the quadrotor application. A model for the quadrotor dynamics was developed for the fully nonlinear rigid body motion, which was to be used as the tracking control plant. These dynamics were then simplified for use in developing the tracking controller. Following this development, an explanation for the use of the Lyapunov analysis was discussed, to show the need for proving stability of the formation trajectory generator. The case where LaSalle's invariance principle is needed was explored.

An asymptotically stable formation trajectory generator was developed based on the integral backstepping process for a simplified dynamic model. The formulation began with planar formation trajectory generation, and then was extended to include the vertical direction. The Lyapunov analysis was used to show stability of the trajectory generator. A further step was taken to show asymptotic stability of the system, through the use of LaSalle's invariance principle. A simulation including one virtual leader and two virtual followers was used to show the effectiveness of the trajectory generator. The trajectory generator was able to successfully allow both virtual followers to achieve a three-dimensional desired distance vector from the virtual leader, as the virtual leader moved on a predefined path. Although only two followers were used here, any number of followers could have been used.

A quadrotor trajectory tracking control law was developed, as well. This was done using sliding mode control, with relationships between the angles and desired states as the sliding surfaces. Using the generated trajectories of the followers and the predefined trajectory of the virtual leader, a leader quadrotor and two follower quadrotors were able to track the desired paths and achieve leader-follower formation flight.

There is future work to be done on this topic. Simulations of large quantities of follower quadrotors are desired. The incorporation of active relative position feedback for the trajectory generation could prove useful. Also, the performance of other application vehicles would be interesting to study. In the same spirit, simulations of multiple types of vehicles should be explored. Despite the importance of these additional research topics, perhaps the most desirable is the experimental application. This would involve a number of quadrotors, with one designated as the leader and the others as followers. Various methods could be used for implementation including dead-reckoning, where each quadrotor knows its generated desired trajectory and tracks that without knowledge of the other quadrotors. This could be done with the current development. Additionally, an external motion capture system could be used to give relative position feedback between the leader and the followers. Finally, on a global scale, sophisticated GPS and communcation techniques could also be used to give relative position feedback. Practical uses for these research topics are evident with the expansion of quadrotor, and more generally UAV, interest, research, and development.

Bibliography

T. Balch and R. C. Arkin. Behavior-based formation control for multirobot teams. *IEEE Transations on Robotics and Automation*, 14(6):926–939, 1998. doi: 10.1109/70.736776.

A. Bürkle, F. Segor, and M. Kollmann. Towards Autonomous Micro UAV Swarms. *Journal of Intelligent & Robotic Systems*, 61(1):339–353, 2011. doi: 10.1007/s10846-010-9492-x.

R. Cui, S. Sam Ge, B. Voon Ee How, and Y. Sang Choo. Leader-follower formation control of underactuated autonomous underwater vehicles. *Ocean Engineering*, 37(17):1491–1502, 2010. doi: 10.1016/j.oceaneng.2010.07.006.

A. Franchi, P. Stegagno, and G. Oriolo. Decentralized Multi-Robot Encirclement of a 3D Target with Guaranteed Collision Avoidance. *Autonomous Robots*, 40(2):245–265, 2016. doi: 10.1007/s10514-015-9450-3.

F. Goodarzi and T. Lee. Stabilization of a Rigid Body Payload With Multiple Cooperative Quadrotors. *Journal of Dynamic Systems, Measurements, and Control*, 138(12):121001–121017, 2016. doi: 10.1115/1.4033945.

E. I. Grøtli and T. A. Johansen. Path Planning for UAVs Under Communication Constraints Using SPLAT! and MILP. *Journal of Intelligent & Robotic Systems*, 65(1):265–282, 2012. doi: 10.1007/s10846-011-9619-8.

L. Hongxia and F. Qi. Application of UAV in the Field Management of Construction Project. *Revista Ibérica de Sistemas e Tecnologias de Informação*, (18A):235–243, 2016. doi: 10.17013/risti.18A.235-243.

J. Hu and G. Feng. Distributed tracking control of leader-follower multi-agent systems under noisy measurement. *Automatica*, 46(8):1382–1387, 2010. doi: 10.1016/j.automatica.2010.05.020.

J. P. LaSalle. Some extensions of Liapunov's second method. *IRE Transactions on Circuit Theory*, CT-7:520–527, 1960.

J. R. T. Lawton, R. W. Beard, and B. J. Young. A decentralized approach to formation maneuvers. *IEEE Transations on Robotics and Automation*, 19(6):933–941, 2003. doi: 10.1109/TRA.2003.819598.

T. Lee. Geometric Control of Quadrotor UAVs Transporting a Cable-Suspended Rigid Body. *IEEE Transactions on Control Systems Technology*, pages 1–10, 2017. doi: 10.1109/TCST.2017.2656060.

M. A. Lewis and K. Tan. High Precision Formation Control of Mobile Robots. *Autonomous robots*, 4(4):387–403, 1997. doi: 10.1023/A:1008814708459.

A. M. Lyapunov. *The General Problem of the Stability of Motion*. PhD thesis, University Kharkov, 1892. Translated into English under the title [Stability of Motion].

A. Mahmood and Y. Kim. Leader-following formation control of quadcopters with heading synchronization. *Aerospace Science and Technology*, 47:68–74, 2015. doi: 10.1016/j.ast.2015.09.009.

V. P. McConnell. Military UAVs claiming the skies with fuel cell power. *Fuel Cells Bulletin*, 2007(12):12–15, 2007. doi: 10.1016/S1464-2859(07)70438-8.

H. Mehrjerdi, J. Ghommam, and M. Saad. Nonlinear coordination control for a group of mobile robots using a virtual structure. *Mechatronics*, 21(7):1147–1155, 2011. doi: 10.1016/j.mechatronics.2011.06.006.

M. Mortazavi, H. A. Talebi, and A. Askari. UAV Formation Control via the Virtual Structure Approach. *Journal of Aerospace Engineering*, 28(1):040140471–040140479, 2015. doi: 10.1061/(ASCE)AS.1943-5525.0000351.

M. H. Nadjim and D. Karim. Model Predictive Lateral Control of a UAV Formation Using a Virtual Structure Approach. *International Journal of Unmanned Systems Engineering*, 2(3):1–11, 2014. doi: 10.14323/ijuseng.2014.8.

A. Nedjati, B. Vizvari, and G. Izbirak. Post-earthquake response by small UAV helicopters. *Natural Hazards*, 80(3):1669–1688, 2016. doi: 10.1007/s11069-015-2046-6.

D. Orfanus, E. P. de Freitas, and F. Eliassen. Self-Organization as a Supporting Paradigm for Military UAV Relay Networks. *IEEE Communication Letters*, 20(4):804–807, 2016. doi: 10.1109/LCOMM.2016.2524405.

L. E. Parker. ALLIANCE: an architecture for fault tolerant multirobot cooperation. *IEEE Transations on Robotics and Automation*, 14(2):220–240, 1998. doi: 10.1109/70.681242.

P. O. Pereira, R. Cunha, D. Cabecinhas, C. Silvestre, and P. Oliveira. Leader following trajectory planning: A trailer-like approach. *Automatica*, 75:77–87, 2017. doi: 10.1016/j.automatica.2016.09.001.

J. Qi, D. Song, H. Shang, N. Wang, C. Hua, C. Wu, X. Qi, and J. Han. Search and Rescue Rotary-Wing UAV and Its Application to the Lushan Ms 7.0 Earthquake. *Journal of Field Robotics*, 33(3):290–321, 2016. doi: 10.1002/rob.21615.

A. Rabah and W. Qinghe. Improved Leader Follower Formation Control for Multiple Quadrotors Based AFSA. *TELKOMNIKA*, 13(1):85–92, 2015. doi: 10.12928/TELKOMNIKA.v13i1.994.

W. Ren and R. Beard. Decentralized Scheme for Spacecraft Formation Flying via the Virtual Structure Approach. *Journal of Guidance, Control, and Dynamics*, 27(1):73–82, 2004. doi: 10.2514/1.9287.

V. Roldão, R. Cunha, D. Cabecinhas, C. Silvestre, and P. Oliveira. A leader-following trajectory generator with application to quadrotor formation flight. *Robotics and Autonomous Systems*, 62(10):1597–1609, 2014. doi: 10.1016/j.robot.2014.05.002.

M. Schneider-Fontan and M. J. Mataric. Territorial multi-robot task division. *IEEE Transations on Robotics and Automation*, 14(5):815–822, 1998. doi: 10.1109/70.720357.

M. Schwager, B. J. Julian, M. Angermann, and D. Rus. Eyes in the Sky: Decentralized Control for the Deployment of Robotic Camera Networks. *Proceedings of the IEEE*, 99 (9):1541–1561, 2011. doi: 10.1109/JPROC.2011.2158377.

Z. Sun and Y. Xia. Receding horizon tracking control of unicycle-type robots based on virtual structure. *International Journal of Robust and Nonlinear Control*, 26(17):3900–3918, 2016. doi: 10.1002/rnc.3555.

J. Torrés-Sanchez, J. M. Peña, A. I. de Castro, and F. López. Multi-temporal mapping of the vegetation fraction in earl-season wheat fields using images from UAV. *Computers and Electronics in Agriculture*, 103:104–113, 2014. doi: 10.1016/j.compag.2014.02.009.

J. A. Vargas-Jacob, J. J. Corona-Sánchez, and H. Rodríguez-Cortés. Experimental Implementation of a Leader-Follower Strategy for Quadrotors Using a Distributed Architecture. *Journal of Intelligent & Robotic Systems*, 84(1):435–452, 2016. doi: 10.1007/s10846-015-0327-7.

M. Veloso, P. Stone, and K. Han. The CMUnited-97 robotic soccer team: Perception and multi-agent control. *Robotics and Autonomous Systems*, 29(2):133–143, 1999. doi: 10.1016/S0921-8890(99)00048-2.

B. Wie. *Space Vehicle Dynamics and Control*. American Institute of Aeronautics and Astronautics, 2 edition, 2008.

D. Xu, X. Zhang, Z. Zhu, C. Chen, and P. Yang. Behavior-Based Formation Control of Swarm Robots. *Mathematical Problems in Engineering*, 2014. doi: 10.1155/2014/205759.

Appendix

MATLAB Code

Main Code

```
%% Leader-Follower Swarm Control

clc
clear variables
close all

%% State Initialization
global m h sim_t d1 d2 k1 k2 k3 b K eps kz1 kz2 dz_f1 dz_f2
   k_1 k_2 k_3 k_4 kz_1 kz_2 alpha_1 alpha_2 beta gamma J_rx
    I_yzx I_zxy  xd yd zd g lamda_1 lamda_2 l_1 l_2



%% Gains
k_1 = 6.5;
k_2 = 0.5;
k_3 = 6.5;
k_4 = 0.5;

kz_1 = 2.5;
kz_2 = 4.5;

```

```
l_1 = 2;
l_2 = 2;

lamda_1 = 4;
lamda_2 = 4;

alpha_1 = 3;
alpha_2 = 3;

b = 0.5;
eps = 10000;

k1 = 0.4;
k2 = 2;
k3 = 0.005;
kz1 = 7500;
kz2 = 7500;
K = 5;

%% Initial Conditions
pL0 = [2;0];
pldot0 = [0; 0.5];

plz_0 = 3;
plzdot_0 = 0;

psi_f1_0 = 3*pi/2;
psi_f2_0 = 0;

p_f1_0 = [3;3];
p_f2_0 = [3;1];

pfz_f1_0 = 0;
pfzdot_f1_0 = 0;
```

```
pfz_f2_0 = 0;
pfzdot_f2_0 = 0;

d1 = [0.35; 0.35];
d2 = [0.35; -0.35];
dz_f1 = 0.35;
dz_f2 = -0.35;

R_f1 = fn_R(psi_f1_0);
R_f2 = fn_R(psi_f2_0);

r_f1_0 = -0.5;
r_f2_0 = 0.5;

S_f1 = fn_S(r_f1_0);
S_f2 = fn_S(r_f2_0);

u_f1_0 = 0.5;
u_f2_0 = 0;

e1_f1_0        = (R_f1'*(pL0-p_f1_0))-d1;
e2_f1_0        = K*sign_fn(e1_f1_0,eps) + (1/k1)*(-S_f1*d1 +
    R_f1'*pldot0 - [u_f1_0; 0]);
zeta_pr_f1_0   = [0;0];

ez1_f1_0 = plz_0-pfz_f1_0+dz_f1;
ez2_f1_0 = plzdot_0-pfzdot_f1_0;

e1_f2_0        = R_f2'*(pL0-p_f2_0)-d2;
e2_f2_0        = K*sign_fn(e1_f2_0,eps) + (1/k1)*(-S_f2*d1 +
    R_f2'*pldot0 - [u_f2_0; 0]);
zeta_pr_f2_0   = [0;0];
```

```
ez1_f2_0 = plz_0-pfz_f2_0+dz_f2;
ez2_f2_0 = plzdot_0-pfzdot_f2_0;
%% Simulation Duration
sim_t = 40;

%% Prepare initial conditions vector

z0=[e1_f1_0; e2_f1_0; zeta_pr_f1_0; r_f1_0;psi_f1_0;ez1_f1_0
    ;ez2_f1_0;e1_f2_0; e2_f2_0; zeta_pr_f2_0; r_f2_0;psi_f2_0
    ;ez1_f2_0;ez2_f2_0];

%% solve ODE using RK4

h=0.1;
t0=0;
t=[t0];

Q=z0';

while t(end)<sim_t
      t0=t(end);
      x0=(Q(end,:))';
      X=rk4('fn_formationcontrol',t0,x0,h);
      t=[t;t0+h];
      Q=[Q;X'];
end

%% Pull states from rk4 output

e1_f1_1 = Q(:,1);
e1_f1_2 = Q(:,2);

e1_f1 = Q(:,1:2);

```

```
e2_f1_1 = Q(:,3);
e2_f1_2 = Q(:,4);

e2_f1 = Q(:,3:4);
zeta_pr_f1 = Q(:,5:6);
r_f_1 = Q(:,7);
psi_f_1 = Q(:,8);

ez1_f1 = Q(:,9);
ez2_f1 = Q(:,10);

e1_f2_1 = Q(:,11);
e1_f2_2 = Q(:,12);

e1_f2 = Q(:,11:12);

e2_f2_1 = Q(:,13);
e2_f2_2 = Q(:,14);

e2_f2 = Q(:,13:14);
zeta_pr_f2 = Q(:,15:16);

r_f_2 = Q(:,17);
psi_f_2 = Q(:,18);

ez1_f2 = Q(:,19);
ez2_f2 = Q(:,20);

%% Reconstruct all important quantities

P_L = [];
P_F_1 = [];
P_F_2 = [];
P_Lz = [];
```

```
P_Fz_1 = [];
P_Fz_2 = [];

for i = 1:length(t)
P_l = [2*cos(0.25*t(i));sin(0.5*t(i))];
P_L = [P_L;P_l'];
P_lz = 3;
P_Lz = [P_Lz;P_lz'];

%%
R_f1 = fn_R(psi_f_1(i));
S_f1 = fn_S(r_f_1(i));

P_f_1 = P_l-R_f1*(e1_f1(i,1:2)'+d1);
P_F_1 = [P_F_1;P_f_1'];

P_fz_1 = P_lz-ez1_f1(i,1)+dz_f1;
P_Fz_1 = [P_Fz_1;P_fz_1'];

%%

R_f2 = fn_R(psi_f_2(i));
S_f2 = fn_S(r_f_2(i));

P_f_2 = P_l-R_f2*(e1_f2(i,1:2)'+d2);
P_F_2 = [P_F_2;P_f_2'];

P_fz_2 = P_lz-ez1_f2(i,1)+dz_f2;
P_Fz_2 = [P_Fz_2;P_fz_2'];
end

%% Trajectory Generator Plotting

pink = (1/255)*[255,51,153];
```

```
purple = (1/255)*[102,0,204];

figure(1)
plot(P_L(:,1),P_L(:,2),'b');hold on;
plot(P_F_1(:,1),P_F_1(:,2),'g');hold on;
plot(P_F_2(:,1),P_F_2(:,2),'r');
xlabel('x [m]');
ylabel('y [m]');
legend('Leader','Follower 1','Follower 2','Location','
    Northwest');
print('C:\Users\campo\Desktop\Thesis\Correct_Format\
    TwoDTrajGen','-depsc');

figure(2)
plot(t,e1_f1(:,1),'g'); hold on;
plot(t,e1_f2(:,1),'--r'); hold on;
plot(t,e1_f1(:,2),'b'); hold on;
plot(t,e1_f2(:,2),'--k'); hold on;
plot(t,ez1_f1,'Color',pink); hold on;
plot(t,ez1_f2,'--c');
axis([0,40,-1.5,3.5]);
xlabel('Time [s]');
ylabel('Position Error [m]');
legend('e1x_{F1}','e1x_{F2}','e1y_{F1}','e1y_{F2}','e1z_{F1}
    ','e1z_{F2}');
print('C:\Users\campo\Desktop\Thesis\Correct_Format\
    PosErrors','-depsc');

figure(3)
plot(t,e2_f1(:,1),'g'); hold on;
plot(t,e2_f2(:,1),'--r'); hold on;
plot(t,e2_f1(:,2),'b'); hold on;
plot(t,e2_f2(:,2),'--k'); hold on;
plot(t,ez2_f1,'Color',pink); hold on;
```

```
plot(t,ez2_f2,'--c');
axis([0,40,-3,2.5]);
xlabel('Time [s]');
ylabel('Velocity Error [m/s]');
legend('e2x_{F1}','e2x_{F2}','e2y_{F1}','e2y_{F2}','e2z_{F1}
    ','e2z_{F2}');
print('C:\Users\campo\Desktop\Thesis\Correct_Format\
    VelErrors','-depsc');

figure(4)
plot(t,psi_f_1,'g'); hold on;
plot(t,psi_f_2,'--r'); hold on;
plot(t,r_f_1,'b'); hold on;
plot(t,r_f_2,'--k');
axis([0,40,-1.5,6.5]);
xlabel('Time [s]');
ylabel('Angular Distance [rad] and Angular Speed [rad/s]');
legend('\Psi_{F1}','\Psi_{F2}','r_{F1}','r_{F2}');
print('C:\Users\campo\Desktop\Thesis\Correct_Format\
    TwoDAngVelandPos','-depsc');

figure(5)
norm_f_1 = sqrt((P_F_1(:,1)-P_L(:,1)).^2+(P_F_1(:,2)-P_L
    (:,2)).^2);
norm_f_2 = sqrt((P_F_2(:,1)-P_L(:,1)).^2+(P_F_2(:,2)-P_L
    (:,2)).^2);
norm_f1_f2 = sqrt((P_F_2(:,1)-P_F_1(:,1)).^2+(P_F_2(:,2)-
    P_F_1(:,2)).^2);
plot(t,norm_f_1,'b');hold on;
plot(t,norm_f_2,'--g');hold on;
plot(t,norm_f1_f2,'r')
axis([0,40,0,3.5]);
xlabel('Time [s]');
ylabel('Distance [m]');
```

```
legend('||P_{F1}-P_L||','||P_{F2}-P_L||','||P_{F2}-P_{F1}||'
    );
print('C:\Users\campo\Desktop\Thesis\Correct_Format\
    TwoDDistances','-depsc');

figure(6)
plot(t,P_Fz_1(:,1)-P_Lz(:,1),'b'); hold on;
plot(t,P_Fz_2(:,1)-P_Lz(:,1),'g'); hold on;
plot(t,P_Fz_2(:,1)-P_Fz_1(:,1),'r');
axis([0,40,-3,0.5]);
xlabel('Time [s]');
ylabel('z [m]');
legend('||Pz_{F1}-Pz_L||','||Pz_{F2}-Pz_L||','||Pz_{F2}-Pz_{
    F1}||','Location','Southeast');
print('C:\Users\campo\Desktop\Thesis\Correct_Format\
    ZDistances','-depsc');

figure(7)
plot3(P_L(:,1),P_L(:,2),P_Lz(:,1),'b');grid on;hold on;
plot3(P_F_1(:,1),P_F_1(:,2),P_Fz_1(:,1),'g');grid on;hold on
    ;
plot3(P_F_2(:,1),P_F_2(:,2),P_Fz_2(:,1),'r');grid on;hold on
    ;
view([335,25]);
xlabel('x [m]');
ylabel('y [m]');
zlabel('z [m]');
legend('Leader','Follower 1','Follower 2')
print('C:\Users\campo\Desktop\Thesis\Correct_Format\
    ThreeDTrajGen','-depsc');

%% Quadrotor Tracking

%% Leader
```

```
dplz1dot = (0.*t);
pldot = [-0.5.*sin(0.25.*t), 0.5.*cos(0.5.*t)];
psi_l= acot(pldot(:,1)./pldot(:,2));
plddot = [-0.125.*cos(0.25.*t),-0.25.*sin(0.5.*t)];
V_l = pldot(:,2)./sin(psi_l);
dpsi_l = ((plddot(:,2)./sin(psi_l))-plddot(:,1))./(((pldot
    (:,2).*cos(psi_l))./((sin(psi_l).^2)))-(V_l.*sin(psi_l)))
    ;
timer = 1;
sat = 0.3;
for timer = 1:1:length(dpsi_l)
    if dpsi_l(timer,1) > sat
        dpsi_l(timer,1) = sat;
    elseif dpsi_l(timer,1) < -sat
        dpsi_l(timer,1) = -sat;
    end
end

phi_l_0 = 0;
theta_l_0 = 0;
dphi_l_0 = 0;
dtheta_l_0 = 0;
P_Lz_0 = 0;
P_Lx_0 = pL0(1,1);
P_Ly_0 = pL0(2,1);

pldddot = [0.0313.*sin(0.25.*t),-0.125.*cos(0.5.*t)];
desired_leader = [P_L(:,1),pldot(:,1),P_L(:,2),pldot(:,2),
    P_Lz(:,1),dplz1dot(:,1), psi_l,dpsi_l,plddot(:,1),plddot
    (:,2)];%,pldddot(:,1),pldddot(:,2)];
initial_leader = [P_Lx_0;P_Ly_0;P_Lz_0;pldot(1,1);pldot(1,2)
    ;dplz1dot(1,1);phi_l_0; theta_l_0; psi_l(1,1); dphi_l_0;
    dtheta_l_0; dpsi_l(1,1)];
```

```
states_leader = f_quadtracking(t,desired_leader,
    initial_leader);

figure(8);
plot(states_leader(:,1),states_leader(:,2),'Color',pink);
    hold on;
plot(P_L(:,1),P_L(:,2),'--b');
axis([-4,4,-4,4]);
xlabel('x [m]');
ylabel('y [m]');
legend('Tracking_{L}','Virtual_{L}','Location','Southeast');
print('C:\Users\campo\Desktop\Thesis\Correct_Format\
    TwoDLeaderTrack','-depsc');

%% Follower 1

u_f1 = (V_l(:,1).*cos(psi_l(:,1)-psi_f_1(:,1)))+((V_l(:,1).*
    d1(2,1)./d1(1,1)).*sin(psi_l(:,1)-psi_f_1(:,1)));

pf1dot(:,1) = u_f1(:,1).*cos(psi_f_1(:,1));
pf1dot(:,2) = u_f1(:,1).*sin(psi_f_1(:,1));

dpsi_d_f1 = (V_l./d1(1,1)).*(sin(psi_l(:,1)-psi_f_1(:,1)));

Gamma_f1 = [1 -d1(2,1); 0 d1(1,1)];

Delta_f1(:,1) = -r_f_1(:,1).*pldot(:,1).*sin(psi_f_1(:,1))+
    r_f_1(:,1).*pldot(:,2).*cos(psi_f_1(:,1))+plddot(:,1).*
    cos(psi_f_1(:,1))+plddot(:,2).*sin(psi_f_1(:,1));
Delta_f1(:,2) = -r_f_1(:,1).*pldot(:,1).*cos(psi_f_1(:,1))-
    r_f_1(:,1).*pldot(:,2).*sin(psi_f_1(:,1))-plddot(:,1).*
    sin(psi_f_1(:,1))+plddot(:,2).*cos(psi_f_1(:,1));

sigmadot_f1 = K*dsign_fn(e1_f1,eps,e2_f1);
```

```

zeta_f1 = zeta_pr_f1 + (1/k3)*b;

invGamma = inv(Gamma_f1);

mu_f1(:,1) = invGamma(1,1).*(Delta_f1(:,1) + k1.*sigmadot_f1(:,1) + k1.*k2.*e2_f1(:,1) + k1.*k3.*zeta_f1(:,1) + k1.*k2.*e1_f1(:,1))+invGamma(1,2).*(Delta_f1(:,2) + k1.*sigmadot_f1(:,2) + k1.*k2.*e2_f1(:,2) + k1.*k3.*zeta_f1(:,2) + k1.*k2.*e1_f1(:,2));
mu_f1(:,2) = invGamma(2,1).*(Delta_f1(:,1) + k1.*sigmadot_f1(:,1) + k1.*k2.*e2_f1(:,1) + k1.*k3.*zeta_f1(:,1) + k1.*k2.*e1_f1(:,1))+invGamma(2,2).*(Delta_f1(:,2) + k1.*sigmadot_f1(:,2) + k1.*k2.*e2_f1(:,2) + k1.*k3.*zeta_f1(:,2) + k1.*k2.*e1_f1(:,2));

udot_f1(:,1) = mu_f1(:,1);
pf1ddot(:,1) = -r_f_1(:,1).*u_f1(:,1).*sin(psi_f_1(:,1))+udot_f1(:,1).*cos(psi_f_1(:,1));
pf1ddot(:,2) = r_f_1(:,1).*u_f1(:,1).*cos(psi_f_1(:,1))+udot_f1(:,1).*sin(psi_f_1(:,1));
pf1dddot = [0,0];
dpfz1dot = -ez2_f1(:,1);

sat_f1 = 0.3;
for timer_f1 = 1:1:length(dpsi_d_f1)
    if dpsi_d_f1(timer_f1,1) > sat_f1
        dpsi_d_f1(timer_f1,1) = sat_f1;
    elseif dpsi_d_f1(timer_f1,1) < -sat_f1
        dpsi_d_f1(timer_f1,1) = -sat_f1;
    end
end

phi_f1_0 = 0;
```

```
theta_f1_0 = 0;
dphi_f1_0 = 0;
dtheta_f1_0 = 0;

desired_follower1 = [P_F_1(:,1),pf1dot(:,1),P_F_1(:,2),
    pf1dot(:,2),P_Fz_1(:,1),dpfz1dot(:,1),psi_f_1,dpsi_d_f1,
    pf1ddot(:,1),pf1ddot(:,2)];%,pf1dddot(:,1),pf1dddot(:,2)
    ];
initial_f1 = [P_F_1(1,1);P_F_1(1,2);P_Fz_1(1,1);pf1dot(1,1);
    pf1dot(1,2);dpfz1dot(1,1);phi_f1_0; theta_f1_0; psi_f_1
    (1,1); dphi_f1_0; dtheta_f1_0; dpsi_d_f1(1,1)];
states_follower1 = f_quadtracking(t,desired_follower1,
    initial_f1);

figure(9)
plot(states_follower1(:,1),states_follower1(:,2),'Color',
    purple); hold on;
plot(P_F_1(:,1),P_F_1(:,2),'--g');
axis([-4,4,-4,4]);
xlabel('x [m]');
ylabel('y [m]');

legend('Tracking_{F1}','Virtual_{F1}','Location','Southeast'
    );

print('C:\Users\campo\Desktop\Thesis\Correct_Format\
    TwoDFollower1Track','-depsc');

%% Follower 2

u_f2 = (V_l(:,1).*cos(psi_l(:,1)-psi_f_2(:,1)))+((V_l(:,1).*
    d2(2,1)./d2(1,1)).*sin(psi_l(:,1)-psi_f_2(:,1)));

pf2dot(:,1) = u_f2(:,1).*cos(psi_f_2(:,1));
```

```
pf2dot(:,2) = u_f2(:,1).*sin(psi_f_2(:,1));

dpsi_d_f2 = (V_l./d2(1,1)).*(sin(psi_l(:,1)-psi_f_2(:,1)));

Gamma_f2 = [1 -d2(2,1); 0 d2(1,1)];

Delta_f2(:,1) = -r_f_2(:,1).*pldot(:,1).*sin(psi_f_2(:,1))+
    r_f_2(:,1).*pldot(:,2).*cos(psi_f_2(:,1))+plddot(:,1).*
    cos(psi_f_2(:,1))+plddot(:,2).*sin(psi_f_2(:,1));
Delta_f2(:,2) = -r_f_2(:,1).*pldot(:,1).*cos(psi_f_2(:,1))-
    r_f_2(:,1).*pldot(:,2).*sin(psi_f_2(:,1))-plddot(:,1).*
    sin(psi_f_2(:,1))+plddot(:,2).*cos(psi_f_2(:,1));

sigmadot_f2 = K*dsign_fn(e1_f2,eps,e2_f2);

zeta_f2 = zeta_pr_f2 + (1/k3)*b;

invGamma = inv(Gamma_f2);

mu_f2(:,1) = invGamma(1,1).*(Delta_f2(:,1) + k1.*sigmadot_f2
    (:,1) + k1.*k2.*e2_f2(:,1) + k1.*k3.*zeta_f2(:,1) + k1.*
    k2.*e1_f2(:,1))+invGamma(1,2).*(Delta_f2(:,2) + k1.*
    sigmadot_f2(:,2) + k1.*k2.*e2_f2(:,2) + k1.*k3.*zeta_f2
    (:,2) + k1.*k2.*e1_f2(:,2));
mu_f2(:,2) = invGamma(2,1).*(Delta_f2(:,1) + k1.*sigmadot_f2
    (:,1) + k1.*k2.*e2_f2(:,1) + k1.*k3.*zeta_f2(:,1) + k1.*
    k2.*e1_f2(:,1))+invGamma(2,2).*(Delta_f2(:,2) + k1.*
    sigmadot_f2(:,2) + k1.*k2.*e2_f2(:,2) + k1.*k3.*zeta_f2
    (:,2) + k1.*k2.*e1_f2(:,2));

udot_f2(:,1) = mu_f2(:,1);
pf2ddot(:,1) = -r_f_2(:,1).*u_f2(:,1).*sin(psi_f_2(:,1))+
    udot_f2(:,1).*cos(psi_f_2(:,1));
```

```
pf2ddot(:,2) = r_f_2(:,1).*u_f2(:,1).*cos(psi_f_2(:,1))+
    udot_f2(:,1).*sin(psi_f_2(:,1));
pf2dddot = [0,0];
dpfz2dot = -ez2_f2(:,1);

timer_f2 = 1;
sat_f2 = 0.3;
for timer_f2 = 1:1:length(dpsi_d_f2)
    if dpsi_d_f2(timer_f2,1) > sat_f2
        dpsi_d_f2(timer_f2,1) = sat_f2;
    elseif dpsi_d_f2(timer_f2,1) < -sat_f2
        dpsi_d_f2(timer_f2,1) = -sat_f2;
    end
end

phi_f2_0 = 0;
theta_f2_0 = 0;
dphi_f2_0 = 0;
dtheta_f2_0 = 0;

desired_follower2 = [P_F_2(:,1),pf2dot(:,1),P_F_2(:,2),
    pf2dot(:,2),P_Fz_2(:,1),dpfz2dot(:,1),psi_f_2,dpsi_d_f2,
    pf2ddot(:,1),pf2ddot(:,2)];%,pf1dddot(:,1),pf1dddot(:,2)
    ];
initial_f2 = [P_F_2(1,1);P_F_2(1,2);P_Fz_2(1,1);pf2dot(1,1);
    pf2dot(1,2);dpfz2dot(1,1);phi_f2_0; theta_f2_0; psi_f_2
    (1,1); dphi_f2_0; dtheta_f2_0; dpsi_d_f2(1,1)];

states_follower2 = f_quadtracking(t,desired_follower2,
    initial_f2);

figure(10)
plot(states_follower2(:,1),states_follower2(:,2),'k'); hold
    on;
```

```
plot(P_F_2(:,1),P_F_2(:,2),'--r');
axis([-4,4,-4,4]);
xlabel('x [m]');
ylabel('y [m]');
legend('Tracking_{F2}','Virtual_{F2}','Location','Southeast');
print('C:\Users\campo\Desktop\Thesis\Correct_Format\TwoDFollower2Track','-depsc');

%% Tracking Plotting
figure(11)
plot3(P_L(:,1),P_L(:,2),P_Lz(:,1),'--b');grid on;hold on;
plot3(P_F_1(:,1),P_F_1(:,2),P_Fz_1(:,1),'--g');grid on;hold on;
plot3(P_F_2(:,1),P_F_2(:,2),P_Fz_2(:,1),'--r');grid on;hold on;
plot3(states_leader(:,1),states_leader(:,2),states_leader(:,3),'Color',pink);grid on;hold on;
plot3(states_follower1(:,1),states_follower1(:,2),states_follower1(:,3),'Color',purple);grid on; hold on;
plot3(states_follower2(:,1),states_follower2(:,2),states_follower2(:,3),'k');grid on; hold on;
view([335,25]);
xlabel('x [m]');
ylabel('y [m]');
zlabel('z [m]');
legend('Virtual_{L}','Virtual_{F1}','Virtual_{F2}','Tracking_{L}','Tracking_{F1}','Tracking_{F2}')
print('C:\Users\campo\Desktop\Thesis\Correct_Format\ThreeDTrack','-depsc');

figure(12)
norm_l_track = sqrt((P_L(:,1)-states_leader(:,1)).^2+(P_L(:,2)-states_leader(:,2)).^2);
```

```
norm_f1_track = sqrt((P_F_1(:,1)-states_follower1(:,1)).^2+(
    P_F_1(:,2)-states_follower1(:,2)).^2);
norm_f2_track = sqrt((P_F_2(:,1)-states_follower2(:,1)).^2+(
    P_F_2(:,2)-states_follower2(:,2)).^2);
plot(t,norm_l_track,'Color',pink);hold on;
plot(t,norm_f1_track,'Color',purple);hold on;
plot(t,norm_f2_track,'k')
xlabel('Time [s]');
ylabel('Distance [m]');
legend('||Virtual Position_{L} - Tracking Position_{L}||','
    ||Virtual Position_{F1} - Tracking Position_{F1}||','||
    Virtual Position_{F2} - Tracking Position_{F2}||');
print('C:\Users\campo\Desktop\Thesis\Correct_Format\
    TwoDTrackDistances','-depsc');

figure(13)
plot(t,states_leader(:,1),'Color',pink); hold on;
plot(t,P_L(:,1),'--b'); hold on;
plot(t,states_follower1(:,1),'Color',purple); hold on;
plot(t,P_F_1(:,1),'--g'); hold on;
plot(t,states_follower2(:,1),'k'); hold on;
plot(t,P_F_2(:,1),'--r');
xlabel('t [s]');
ylabel('x [m]');
legend('Tracking_{L}','Virtual_{L}','Tracking_{F1}','
    Virtual_{F1}','Tracking_{F2}','Virtual_{F2}');
print('C:\Users\campo\Desktop\Thesis\Correct_Format\
    CompareXCon','-depsc');

figure(14)
plot(t,states_leader(:,2),'Color',pink); hold on;
plot(t,P_L(:,2),'--b'); hold on;
plot(t,states_follower1(:,2),'Color',purple); hold on;
plot(t,P_F_1(:,2),'--g'); hold on;
```

```
plot(t,states_follower2(:,2),'k'); hold on;
plot(t,P_F_2(:,2),'--r');
xlabel('t [s]');
ylabel('y [m]');
legend('Tracking_{L}','Virtual_{L}','Tracking_{F1}','Virtual_{F1}','Tracking_{F2}','Virtual_{F2}');
print('C:\Users\campo\Desktop\Thesis\Correct_Format\CompareYCon','-depsc');

figure(15)
plot(t,states_leader(:,3),'Color',pink); hold on;
plot(t,P_Lz(:,1),'--b'); hold on;
plot(t,states_follower1(:,3),'Color',purple); hold on;
plot(t,P_Fz_1(:,1),'--g'); hold on;
plot(t,states_follower2(:,3),'k'); hold on;
plot(t,P_Fz_2(:,1),'--r');
xlabel('t [s]');
ylabel('z [m]');
legend('Tracking_{L}','Virtual_{L}','Tracking_{F1}','Virtual_{F1}','Tracking_{F2}','Virtual_{F2}');
print('C:\Users\campo\Desktop\Thesis\Correct_Format\CompareZCon','-depsc');

figure(16)
plot(t,states_leader(:,4),'Color',pink); hold on;
plot(t,pldot(:,1),'--b'); hold on;
plot(t,states_follower1(:,4),'Color',purple); hold on;
plot(t,pf1dot(:,1),'--g'); hold on;
plot(t,states_follower2(:,4),'k'); hold on;
plot(t,pf2dot(:,1),'--r');
xlabel('t [s]');
ylabel('dx [m/s]');
legend('Tracking_{L}','Virtual_{L}','Tracking_{F1}','Virtual_{F1}','Tracking_{F2}','Virtual_{F2}');
```

```
print('C:\Users\campo\Desktop\Thesis\Correct_Format\CompareDXCon','-depsc');

figure(17)
plot(t,states_leader(:,5),'Color',pink); hold on;
plot(t,pldot(:,2),'--b'); hold on;
plot(t,states_follower1(:,5),'Color',purple); hold on;
plot(t,pf1dot(:,2),'--g'); hold on;
plot(t,states_follower2(:,5),'k'); hold on;
plot(t,pf2dot(:,2),'--r');
xlabel('t [s]');
ylabel('dy [m/s]');
legend('Tracking_{L}','Virtual_{L}','Tracking_{F1}','Virtual_{F1}','Tracking_{F2}','Virtual_{F2}');
print('C:\Users\campo\Desktop\Thesis\Correct_Format\CompareDYCon','-depsc');

figure(18)
plot(t,states_leader(:,6),'Color',pink); hold on;
plot(t,dplz1dot(:,1),'--b'); hold on;
plot(t,states_follower1(:,6),'Color',purple); hold on;
plot(t,dpfz1dot(:,1),'--g'); hold on;
plot(t,states_follower2(:,6),'k'); hold on;
plot(t,dpfz2dot(:,1),'--r');
xlabel('t [s]');
ylabel('dz [m/s]');
legend('Tracking_{L}','Virtual_{L}','Tracking_{F1}','Virtual_{F1}','Tracking_{F2}','Virtual_{F2}');
print('C:\Users\campo\Desktop\Thesis\Correct_Format\CompareDZCon','-depsc');

figure(19)
```

```
norm_f1_l_formation = sqrt((states_follower1(:,1)-states_leader(:,1)).^2+(states_follower1(:,2)-states_leader(:,2)).^2);
norm_f2_l_formation = sqrt((states_follower2(:,1)-states_leader(:,1)).^2+(states_follower2(:,2)-states_leader(:,2)).^2);
norm_f2_f1_formation = sqrt((states_follower2(:,1)-states_follower1(:,1)).^2+(states_follower2(:,2)-states_follower1(:,2)).^2);
plot(t,norm_f1_l_formation,'b');hold on;
plot(t,norm_f2_l_formation,'--g');hold on;
plot(t,norm_f2_f1_formation,'r')
xlabel('Time [s]');
ylabel('Distance [m]');
legend('||P_{F1}-P_L||','||P_{F2}-P_L||','||P_{F2}-P_{F1}||');
print('C:\Users\campo\Desktop\Thesis\Correct_Format\TwoDFormationDistances','-depsc');
```

Trajectory Generation Function

```
%Formation Trajectory Generation
function dx = fn_formationcontrol(t,xx)

global d1 d2 k1 k2 k3 b K eps kz1 kz2 dz_f1 dz_f2
%% States assignment

e1_f1       = xx(1:2);
e2_f1       = xx(3:4);
zeta_pr_f1  = xx(5:6);
r_f1        = xx(7);
psi_f1      = xx(8);
ez1_f1      = xx(9);
ez2_f1      = xx(10);

```

```
e1_f2         = xx(11:12);
e2_f2         = xx(13:14);
zeta_pr_f2    = xx(15:16);
r_f2          = xx(17);
psi_f2        = xx(18);
ez1_f2        = xx(19);
ez2_f2        = xx(20);

%% Leader
pl = [2*cos(0.25*t);sin(0.5*t)];
pldot = [-0.5*sin(0.25*t); 0.5*cos(0.5*t)];
plddot = [-0.125*cos(0.25*t);-0.25*sin(0.5*t)];

plz = 3;
plzdot = 0;
plzddot = 0;
%% Follower 1

R_f1 = fn_R(psi_f1);
S_f1 = fn_S(r_f1);

Gamma_f1 = [1 -d1(2,1); 0 d1(1,1)];
Delta_f1 = -S_f1*R_f1'*pldot+R_f1'*plddot;

sigmadot_f1 = K*dsign_fn(e1_f1,eps,e2_f1);

zeta_f1 = zeta_pr_f1 + (1/k3)*b;

mu_f1 = inv(Gamma_f1)*(Delta_f1 + k1*sigmadot_f1 + k1*k2*
    e2_f1 + k1*k3*zeta_f1 + k1*k2*e1_f1);

e1d_f1 = -S_f1*e1_f1+k1*e2_f1-k1*K*sign_fn(e1_f1,eps);

e2d_f1 = -k2*e2_f1 - k3*zeta_pr_f1 - k2*e1_f1;
```

```
zetadot_pr_f1 = e2_f1;

rdot_f1 = mu_f1(2,1);

psi_l = acot(pldot(1)/pldot(2));
V_l = pldot(2)/sin(psi_l);

psid_f1 = (V_l/d1(1))*sin(psi_l-psi_f1);

ez1d_f1 = ez2_f1;
ez2d_f1 = plzddot-K*sign_fn(plzddot,eps)-K*sign_fn(kz1*
    ez1_f1+kz2*ez2_f1,eps);

%% Follower 2

R_f2 = fn_R(psi_f2);
S_f2 = fn_S(r_f2);

Gamma_f2 = [1 -d2(2,1); 0 d2(1,1)];

Delta_f2 = -S_f2*R_f2'*pldot+R_f2'*plddot;

sigmadot_f2 = K*dsign_fn(e1_f2,eps,e2_f2);

zeta_f2 = zeta_pr_f2 + (1/k3)*b;

mu_f2 = inv(Gamma_f2)*(Delta_f2 + k1*sigmadot_f2 + k1*k2*
    e2_f2 + k1*k3*zeta_f2 + k1*k2*e1_f2);

e1d_f2 = -S_f2*e1_f2+k1*e2_f2-k1*K*sign_fn(e1_f2,eps);

e2d_f2 = -k2*e2_f2 - k3*zeta_pr_f2 - k2*e1_f2;
```

```
zetadot_pr_f2 = e2_f2;

rdot_f2 = mu_f2(2,1);

psid_f2 = (V_l/d1(1))*sin(psi_l-psi_f2);

ez1d_f2 = ez2_f2;
ez2d_f2 = plzddot-K*sign_fn(plzddot,eps)-K*sign_fn(kz1*
    ez1_f2+kz2*ez2_f2,eps);

%% Solve Diff Eq

dx       = [e1d_f1;e2d_f1;zetadot_pr_f1;rdot_f1;psid_f1;
    ez1d_f1;ez2d_f1;e1d_f2;e2d_f2;zetadot_pr_f2;rdot_f2;
    psid_f2;ez1d_f2;ez2d_f2];

end
```

Runge Kutta for the Trajectory Generation Function

```
function x=rk4(name,t0,q0,h)
t1=t0+h/2;
t2=t0+h;
f0=feval(name,t0,q0);
x1=q0+h*f0/2;
f1=feval(name,t1,x1);
x2=q0+h*f1/2;
f2=feval(name,t1,x2);
x3=q0+h*f2;
f3=feval(name,t2,x3);
x=q0+h*(f0+2*f1+2*f2+f3)/6;
```

Rotation Matrix Function

```
function R = fn_R(psi)
R = [cos(psi) -sin(psi); sin(psi) cos(psi)];
```

```
end
```

Skew Symmetric Matrix Function

```
function S = fn_S(r)

S = [0 -r; r 0];

end
```

Saturation Function

```
function ret = sign_fn(arg,eps)

ret = arg./(sqrt(arg.*arg)+eps);

end
```

Time-Derivative of the Saturation Function

```
function ret = dsign_fn(arg,eps,dotx)

ret = (eps./(sqrt(arg.*arg)+eps).^2).*dotx;

end
```

Quadrotor Tracking Function

```
function states = f_quadtracking(t,desired,initial)

global dzd ddxd ddyd dpsi_d dxd dyd counter sim_t psi_d k_1
    k_2 k_3 k_4 kz_1 kz_2 alpha_1 alpha_2 beta gamma J_rx
    I_yzx I_zxy xd yd zd g lamda_1 lamda_2 l_1 l_2 K_v m

%% Parameter Definitions

K_v         = 54.945;         % [rad s/V]
```

```
J_r         = 6e-5;          % [kg m^2]
I_xx        = 0.001;         % [kg m^2]
I_yy        = 0.001;         % [kg m^2]
I_zz        = 0.002;         % [kg m^2]
b           = 3.935139e-6;   % [N/V]
d           = 1.192464e-7;   % [Nm/V]
l           = 0.1969;        % [m]
m           = 2.85;          % [kg]
g           = 9.81;          % [m/s^2]

J_rx        = J_r/I_xx;
I_yzx       = (I_yy - I_zz)/I_xx;
I_zxy       = (I_zz - I_xx)/I_yy;

m = 100;

%% Declare Desired States

xd = desired(:,1);
dxd = desired(:,2);
yd = desired(:,3);
dyd = desired(:,4);
zd = desired(:,5);
dzd = desired(:,6);
psi_d = desired(:,7);
dpsi_d = desired(:,8);
ddxd = desired(:,9);
ddyd = desired(:,10);

%% Initial conditions:
% X = [x y z dx dy dz phi theta psi wx wy wz]

x_0       = initial(1,1);          % m
y_0       = initial(2,1);          % m
```

```
z_0      = initial(3,1);          % m

dx_0     = initial(4,1);          % m/s
dy_0     = initial(5,1);          % m/s
dz_0     = initial(6,1);          % m/s

phi_0    = initial(7,1);          % rad
theta_0  = initial(8,1);          % rad
psi_0    = initial(9,1);          % rad

wx_0     = initial(10,1);         % rad/s
wy_0     = initial(11,1);         % rad/s
wz_0     = initial(12,1);         % rad/s

X0 = [x_0; y_0; z_0; dx_0; dy_0; dz_0; phi_0; theta_0; psi_0; wx_0; wy_0; wz_0];

%% Simulate Dynamics

h=0.1;
t0=0;
t=[t0];

Z=X0';
counter = 1;
while t(end)<sim_t
    t0=t(end);
    Z0=(Z(end,:))';
    states=rk4_track('f_tracking_controller',t0,Z0,h);
    t=[t;t0+h];

    sat_phi = 0.3;
    if states(7,:) > sat_phi
        states(7,:) = sat_phi;
```

```
    elseif states(7,:) < -sat_phi
        states(7,:) = -sat_phi;
    end

sat_theta = 0.3;
    if states(8,:) > sat_theta
        states(8,:) = sat_theta;
    elseif states(8,:) < -sat_theta
        states(8,:) = -sat_theta;
    end

sat_psi = 0.3;
    if states(9,:) > sat_psi
        states(9,:) = sat_psi;
    elseif states(9,:) < -sat_psi
        states(9,:) = -sat_psi;
    end

sat_wz = 1.5;
    if states(12,:) > sat_wz
        states(12,:) = sat_wz;
    elseif states(12,:) < -sat_wz
        states(12,:) = -sat_wz;
    end

    Z=[Z;states'];
    counter = counter + 1;
end

%% Reconstruct states

x        = Z(:,1);
y        = Z(:,2);
z        = Z(:,3);
```

```

dx        = Z(:,4);
dy        = Z(:,5);
dz        = Z(:,6);

phi       = Z(:,7);
theta     = Z(:,8);
psi       = Z(:,9);

wx        = Z(:,10);
wy        = Z(:,11);
wz        = Z(:,12);

states = [x,y,z,dx,dy,dz,phi,theta,psi,wx,wy,wz];

end
```

Runge Kutta for the Quadrotor Tracking Function

```
function x_track=rk4_track(name_track,t0_track,q0_track,h)
t1_track=t0_track+h/2;
t2_track=t0_track+h;
f0_track=feval(name_track,t0_track,q0_track);
x1_track=q0_track+h*f0_track/2;
f1_track=feval(name_track,t1_track,x1_track);
x2_track=q0_track+h*f1_track/2;
f2_track=feval(name_track,t1_track,x2_track);
x3_track=q0_track+h*f2_track;
f3_track=feval(name_track,t2_track,x3_track);
x_track=q0_track+h*(f0_track+2*f1_track+2*f2_track+f3_track)
    /6;
```

Tracking Controller Function

```
function dY = f_tracking_controller(t_track,Y)

```

```
global dzd ddxd ddyd h dpsi_d dxd dyd counter psi_d k_1 k_2
    k_3 k_4 kz_1 kz_2 alpha_1 alpha_2 J_rx I_yzx I_zxy  xd yd
     zd g lamda_1 lamda_2 l_1 l_2 m

%% Retrieve States
x       = Y(1);
y       = Y(2);
z       = Y(3);

dx      = Y(4);
dy      = Y(5);
dz      = Y(6);

phi     = Y(7);
theta   = Y(8);
psi     = Y(9);

sat_phi = 0.3;
    if phi > sat_phi
        phi = sat_phi;
    elseif phi < -sat_phi
        phi = -sat_phi;
    end

sat_theta = 0.3;
    if theta > sat_theta
        theta = sat_theta;
    elseif theta < -sat_theta
        theta = -sat_theta;
    end

sat_psi = 0.3;
    if psi > sat_psi
        psi = sat_psi;
```

```
    elseif psi < -sat_psi
        psi = -sat_psi;
    end

wx      = Y(10);
wy      = Y(11);
wz      = Y(12);

sat_wz = 1.5;
    if wz > sat_wz
        wz = sat_wz;
    elseif wz < -sat_wz
        wz = -sat_wz;
    end

%% Calculate angular rates

dphi     = wx+wy*sin(phi)*tan(theta)+wz*cos(phi)*tan(theta);
dtheta   = wy*cos(phi)-wz*sin(phi);
dpsi     = wy*sin(phi)/cos(theta)+wz*cos(phi)/cos(theta);

%% Control law

s(1) = dtheta+(k_2*g+alpha_1)*theta+(k_1+alpha_1*k_2)*(dx-
    dxd(counter,1))+alpha_1*k_1*(x-xd(counter,1))-k_2*ddxd(
    counter,1);
s(2) = dphi-(k_3+alpha_2*k_4)*(dy-dyd(counter,1))+(k_4*g+
    alpha_2)*phi-alpha_2*k_3*(y-yd(counter,1))+k_4*ddyd(
    counter,1);

u(1) = g-kz_1*(z-zd(counter,1))-kz_2*(dz-dzd(counter,1));
u(2) = -lamda_2*tanh(m*s(2))-I_yzx*dtheta*(dpsi-dpsi_d(
    counter,1))-(k_3+alpha_2*k_4)*(g*phi+ddyd(counter,1))-(
    k_4*g+alpha_2)*dphi+alpha_2*k_3*(dy-dyd(counter,1));
```

```
u(3)  = -lamda_1*tanh(m*s(1))+I_zxy*(dpsi-dpsi_d(counter,1))*dphi-(k_2*g+alpha_1)*dtheta-(k_1+alpha_1*k_2)*(g*theta-ddxd(counter,1))-alpha_1*k_1*(dx-dxd(counter,1));
u(4)  = -l_1*(psi-psi_d(counter,1))-l_2*(dpsi-dpsi_d(counter,1));

u_g      = 0;          % Gyroscopic effect is ignored

%% Test controller on nonlinear dynamics

dY   = zeros(12,1);

dY(1)       = dx;
dY(2)       = dy;
dY(3)       = dz;

dY(4)       = (cos(phi)*sin(theta)*cos(psi)+sin(phi)*sin(psi))*u(1);
dY(5)       = (cos(phi)*sin(theta)*sin(psi)-sin(phi)*cos(psi))*u(1);
dY(6)       = cos(phi)*cos(theta)*u(1)-g;

dY(7)       = wx +  wy*sin(phi)*tan(theta) + wz*cos(phi)*tan(theta);
dY(8)       = wy*cos(phi) - wz*sin(phi);
dY(9)       = wy*sin(phi)/cos(theta) + wz*cos(phi)/cos(theta);

dY(10)      = I_yzx*wy*wz + J_rx*u_g*wy + u(2);
dY(11)      = I_zxy*wx*wz - J_rx*u_g*wx + u(3);
dY(12)      = u(4);

end
```