DISTRIBUTED MODEL PREDICTIVE CONTROL FOR COLLISION AND OBSTACLE AVOIDANCE OF MULTIPLE QUADCOPTERS

by

Shaundell Dubay

Submitted in partial fulfillment of the requirements
for the degree of Master of Applied Science

at

Dalhousie University
Halifax, Nova Scotia
July 2018

This thesis is dedicated to my parents, Rickey and Molly, and my brother, Stefan.

Table of Contents

List of Tables

List of Figures

Abstract

As the cost to manufacture quadcopters decrease, multi-agent applications for civilian tasks, such as large-scale surveying, search and rescue missions and fire fighting, are becoming increasingly realizable. However, a multi-agent system of fast moving quadcopters has a high risk of collisions with neighbouring quadcopters or obstacles. The objective of this work is to develop a control strategy for collision and obstacle avoidance of multiple quadcopters. In this thesis, the problem of distributed model predictive control (MPC) for collision avoidance among a team of multiple quadcopters attempting to reach consensus is investigated. Violations of a predetermined safety radius generates output constraints on the MPC optimization function. In addition, logarithmic barrier functions are implemented as input rate constraints on the control actions. Extensive simulation studies for a team of four quadcopters illustrate promising results of the proposed control strategy and case variations. In addition, distributed MPC parameter effects on the system performance are studied and a successful isolated study for obstacle avoidance of static objects is presented.

List of Abbreviations and Symbols Used

α	Setpoint tuning parameter
$\mathcal{A}$	Adjacency matrix
B	Logarithmic barrier function
Δu	Optimal control move(s)
Δu_{max}	Maximum control rate
Δu_{min}	Minimum control rate
η	Euler angles
e	Error
G	Dynamic matrix
g	Gravitational constant
J	Cost function
K	State feedback gains
λ	Move suppression
μ	Logarithmic barrier function penalty parameter
m	Mass of quadcopter and payload
n_a	Number of agents
n_o	Number of obstacles
n_p	Prediction horizon
n_u	Control horizon
r	Relative distance
r_a	Relative avoidance distance
r_p	Relative predicted distance
ϕ	Roll angle
Φ	Prediction correction
ψ	Yaw angle
$\overline{T}$	Resultant thrust
θ	Pitch angle
τ_θ	Pitching torque, torque about the y-axis

τ_ϕ	Rolling torque, torque about the x-axis
τ_ψ	Yawing torque, torque about the z-axis
ξ	Inertial frame
x	Position along the x-axis
y	Position along the y-axis
z	Position along the z-axis

DMC	Dynamic matrix control
GCS	Global coordinate system
LCS	Local coordinate system
MAS	Multi-agent system
MPC	Model predictive control
UAV	Unmanned aerial vehicle
UGV	Unmanned ground vehicle
UUV	Unmanned underwater vehicle

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Dr. Ya-Jun Pan, whose careful guidance and patience aided in the successful completion of this thesis. Deep appreciation to the members of the Supervisory Committee, Dr. Robert Bauer and Dr. Guy Kember, for their thoughtful feedback and time.

My profound gratitude goes to my parents and my brother, whose continuous encouragement and insights are immeasurable.

Special thanks to the Advanced Controls and Mechantronics research group for their generosity and enthusiasm over the years. Finally, I would like to acknowledge the enormous efforts of Ms. Kate Hide, Ms. Donna Laffin and Ms. Jascinth Butterfield during my time at Dalhousie University.

Chapter 1

Introduction

Multiple agents are often used for the advantages of cooperative behaviour, where they can interact and solve complex problems that are beyond the capacity of a single agent. A multi-agent system (MAS) is a network of software agents or computationally capable physical (electro-mechanical) devices, that exchange information to make decisions or determine actuations in order to progress towards an objective. The main advantage of an unmanned MAS is that they are useful for missions/tasks that are beyond human limitations. They are not subject to human conditions such as fatigue and endurance, and they are expendable and recoverable; removing the risk to humans. Planetary explorations, factory floor operations and transportation can be carried out by cooperative unmanned ground vehicles (UGVs) and a team of unmanned underwater vehicles (UUVs) could be deployed for large-scale surveying, seismic monitoring, underwater explorations and searches. As the cost to manufacture an unmanned aerial vehicle (UAV) decreases, multi-agent applications can aid with civilian tasks such as search and rescue missions, surveillance, fire fighting, crop monitoring and explorations in extreme environments. In addition, multi-agent systems are not limited to homogeneous compositions. A MAS makeup can include a mix (various structures) of one type of unmanned vehicle or combinations of UGVs, UUVs and UAVs.

Due to the operational nature of MASs, it is suitable that centralized control approaches are restricted to small groups like a leader follower configuration. However, for large sizes of MASs the centralized approach fails since centralized information gathering is not sustainable and the computational load becomes impractical. Therefore, a distributed methodology is more effective.

In a multiple quadcopter system, information can be efficiently collected from spatially distributed agents. The structure of a quadcopter allows for versatile movement, making it a popular choice for the applications listed previously. However, a

quadcopter's primary restriction is its battery life. Developments in sensor performance also allow quadcopters to be designed with different characteristics. This is advantageous because various capabilities can be distributed and assigned to agents in the system.

1.1 Research Motivation

A quadcopter can be considered a challenging system, and to an even greater extent with multiple quadcopters. This premise can be highlighted through comparisons with a UGV. A UGV travels on a 2-dimensional plane, it is slow moving, has low-lying sensors that can give feedback on objects nearby, and it has the ability to make a full stop. On the other hand, a quadcopter operates in 3-dimensions, is fast moving, and therefore has a higher risk of collisions.

Many control schemes have been successfully developed for formation control [2], consensus control [3], communication constraints [4] and trajectory tracking [5] for multiple quadcopters. However, certain aspects are lacking in the control for collision and obstacle avoidance of multiple quadcopters. Research to fill these gaps is especially important in enhancing the operational capabilities of multiple quadcopter technologies, allowing important applications like rescue missions or explorations to progress to a functional reality in society.

1.2 Literature Review

Collision avoidance is the maintenance of a strategy designed to prevent agents colliding with other agents. Generally, agents that observe a relative safety distance with other agents can achieve collision avoidance. In addition, collision avoidance applies to inter-agent behaviours of the MAS or with agents from an outside MAS. Evasive movement with static objects is considered obstacle avoidance.

1.2.1 Multi-Agent Collision Avoidance

For a team of multiple quadcopters, a successful performance depends on the ability to fly without collision. In literature, many control strategies have been developed to deal with collision avoidance among multi-agent systems. Methods include trajectory

generation [6] [7] of a collision free path, and gradient methods [8] [9] to determine control gains that fulfill collision avoidance criteria. In [10], fuzzy logic was used to implement trajectory shapes (example: S-shape) that would steer the agents out of the 2-dimensional collision territory. The fuzzy logic control was successful in a leader follower structure, and with inherent knowledge that these shapes would provide sufficient evasive movement for collision avoidance. Potential functions can be used for collision avoidance as in [11] [12] and an adaptive potential function in [13], where repulsive forces were bounded to maintain a desired communication topology. The work in [6] also used collision avoidance potential functions, but designed various regions (communication, maintenance and avoidance) for control. However, potential forces alone may drive agents to a deadlock situation; the repulsive forces cancel with the attractive forces leaving agents unable to advance from their current positions. A unique solution [14] to this problem was to introduce gyroscopic forces (these forces act perpendicular to the direction of motion) to swing the agents out and break free of deadlock situations. A deep learned collision avoidance strategy in [15] provides a method for dealing with noisy sensor measurements. Noise in sensor data or inaccurate data can adversely affect the situational awareness of an agent and can lead to a misguided strategy or poor actuation. The differential game approach [16] allows access to a deduced performance of the control strategy to aid with collision avoidance. However, in order to exploit this *a priori* performance, a centralized framework is necessary.

1.2.2 Model Predictive Control Collision Avoidance

Collision avoidance can also be achieved with optimization based approaches. One scheme with recent developments is model predictive control (MPC), which has the ability to handle hard constraints on the control action and states [17]. While many types of MPC schemes exist, most research practice the state-space formulation.

In [18], a decentralized linear time-varying hierarchical MPC is implemented for control. The top layer is a hybrid MPC that generates online desired positions to reach a known target position and avoid collisions; the middle layer contains a real-time linear MPC for tracking; in the bottom layer, a nonlinear MPC is used for quadcopter control. In [19], decentralized MPC is used for evasive action in the

vertical direction by applying a penalty term, which requires less computation time. However, while evasive movement in the vertical direction is effective, more freedom to avoid other quadcopters could be gained if the x, y-directions were considered. In addition, the decentralized approach is vulnerable should the leader lapse. The converse is true in [20] because distributed MPC is implemented. A distributed methodology is more effective since agents are coupled only with their neighbours [21]. In [20], evasive action between two teams of unmanned aerial vehicles occurs on the horizontal plane by implementing a Kalman filter to estimate the positions of the other team. This method is useful for multiple MAS already in formation (steady state positions) and could be extended for static obstacle avoidance. Following a similar 2-dimensional distributed MPC approach, [22] proposes terminal elements on the controller. Terminal constraints are applied for future predictions and are implemented to guarantee stability [23]. The drawback to this strategy is that final desired positions are needed. The work presented in [24], achieves collision avoidance by separating pairs of vehicles by a shared hyperplane; however only 2-dimensions are considered. The synthesis approach [25] of distributed MPC is unique because this method can utilize past predicted states. This allows for less communication among agents, however the computation times are quite large (between 1-9 seconds).

It is observed that the number of MPC strategies for collision avoidance is limited. Most literature only employ evasive movement in the z-direction, or only consider 2-dimensional movement on the x, y-plane. In addition, some simulations begin when the agents are already in motion or that final desired positions are needed, which is not always practicable.

1.3 Thesis Contributions

The main contribution of this thesis is the development of a new distributed MPC strategy for collision and obstacle avoidance of multiple quadcopters. This work contains the following contributions:

- A consensus algorithm is employed to determine desired setpoints online.
- While other MPC schemes have been applied for collision avoidance, a thorough

search of relevant literature yielded no implementation of dynamic matrix control (DMC). In this work, DMC is firstly used to generate conditional output constraints on quadcopter positions for collision and obstacle avoidance.

- Input rate constraints are implemented in the form of a logarithmic barrier function to limit control actions. Input rate constraints remove the need for hard input constraints such as saturation. In addition, the logarithmic barrier function is a penalty formulation, which helps reduce the overall number of hard constraints on the MPC optimization function.

- In order to observe the influence of constraints on system performance, case studies for collision and obstacle avoidance are formulated by modifying the criteria for output constraint generation.

- In this work, collision and obstacle avoidance in all 3-dimensions (x, y, z-directions) is successful. Unlike many literature, simulations begin with quadcopters at rest and on the ground.

- Extensive simulation studies are conducted and discussions are given regarding system performance and parameter effects.

1.4 Thesis Organization

The organization of this thesis is as follows. Chapter 2 describes the relevant background theories, specifically, graph terminology, the consensus algorithm and a multi-agent control structure comparison. Chapter 3 gives a detailed overview of quadcopter dynamics and the mathematical model used in this work. Model predictive control for consensus of multiple quadcopters is shown in Chapter 4. Chapter 5 presents the formulation of input constraints, details the output constraints for collision and obstacle avoidance and highlights the differences between case variations. Chapter 6 discusses the simulation studies for the collision avoidance algorithms. Chapter 7 presents the simulation results for obstacle avoidance. Chapter 8 summarizes the conclusions of this work and potential areas for future research.

Chapter 2

Background Theory

2.1 Graph Theory

In this section, basic graph theory concepts that are essential in the study of multi-agent systems are presented. A network structure is used to describe the communication topology among agents and can be mathematically modelled as an algebraic graph. With reference to a multi-agent system, the agents are represented by nodes and a communication link from agent j to agent i is represented by an edge.

Directed Graph: A directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ for n_a agents, has node set $\mathcal{V} = \{v_1, ..., v_{n_a}\}$ that is finite and nonempty and an edge set $\mathcal{E}$ of ordered pairs called edges. For a directed graph, an edge e_{ij} of $\mathcal{E}$ shows that information is transmitted from node v_j to v_i. Therefore, it is considered that v_i is the parent node and v_j is the child node. If v_i can receive from agent v_j and vice versa, then this case is specified as a bidirectional edge. Self edges, defined as when a node has the ability to send information to itself, are not allowed unless specifically indicated [26]. An example of a directed graph with four agents is shown in Fig.2.1, where $\{e_{21}, e_{32}, e_{43}\}$ are directed edges and e_{13} is a bidirectional edge.

Undirected Graph: In an undirected graph, agent i and agent j can receive information from each other, indicating that they have bidirectional edges. This is considered a special case, where in an undirected graph e_{ij} corresponds to e_{ij} and e_{ji} in a directed graph. An example of an undirected graph with four agents is shown in

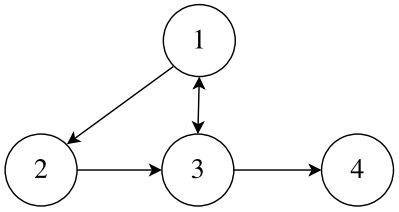

Figure 2.1: Directed graph

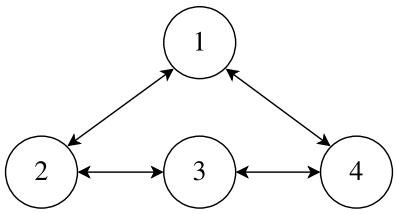

Figure 2.2: Undirected graph

Fig.2.2, where $\{e_{12}, e_{23}, e_{34}, e_{41}\}$ are bidirectional edges.

Weighted Graph: A weighted graph is when an edge has an associated weight or penalty; often used to indicate the reliability or cost of information from a specific node. However, in undirected graphs the weights of e_{ij} and e_{ji} must be the same.

An adjacency matrix,

$$\mathcal{A} = \begin{bmatrix} a_{11} & \cdots & a_{1j} \\ \vdots & & \vdots \\ a_{i1} & \cdots & a_{ij} \end{bmatrix} , \tag{2.1}$$

describes the communication topology or channels for information exchange among agents. It is developed by the following rules:

$$a_{ij} = \begin{cases} 1 , & \text{if } e_{ij} \in \mathcal{E} \\ 0 , & \text{otherwise} \ . \end{cases} \tag{2.2}$$

In (2.2), if agent i can receive from agent j, then element $a_{ij} = 1$, otherwise $a_{ij} = 0$. The associated adjacency matrices for the directed graph in Fig.2.1 and the undirected graph in Fig.2.2 are respectively,

$$\mathcal{A} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \text{and} \quad \mathcal{A} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} . \tag{2.3}$$

Graph Connectivity: The in-degree of a node refers to the number of edges flowing towards the node and can be mathematically determined as the row sum of the adjacency matrix,

$$d_{i,in} = \sum_{j=1}^{n_a} a_{ij} \ . \tag{2.4}$$

The out-degree is the number of edges flowing outwards from a node and can be determined from the adjacency matrix using column sum,

$$d_{i,out} = \sum_{j=1}^{n_a} a_{ji} \ . \tag{2.5}$$

A graph is referred to as balanced if the in-degree and out-degree are equal. Therefore, it can be said that an undirected graph is always balanced.

A directed tree is a directed graph where a sequence of ordered edges exists from one node to another. Multiple directed trees can exist in a single directed graph. A directed spanning tree occurs if a node does not have a parent node (in-degree of zero) and a subset of edges forms a directed tree. Accordingly, all other nodes are reachable from a single node called the root. An example of a directed spanning tree with four agents is presented in Fig.2.3, where agent 1 is the root node.

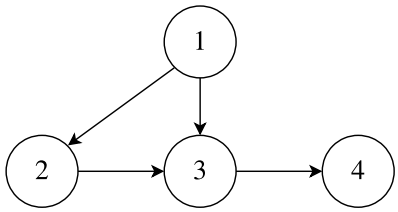

Figure 2.3: Directed spanning tree

A directed graph is strongly connected if distinct nodes pairs can be connected by following a directed path of edges [27]. Consider Fig.2.4, it is strongly connected because an agent can reach every other agent. From a different perspective, the directed graph is strongly connected because each agent is a root node with an associated spanning tree. A directed graph is weakly connected if when its edges are replaced with bidirectional edges, all nodes can reach one another.

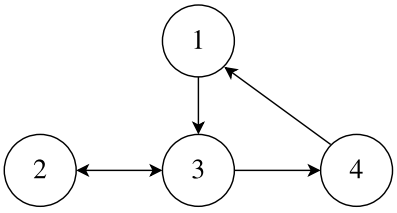

Figure 2.4: Strongly connected graph with multiple spanning trees

2.2 Consensus Control Algorithms

In a network of agents, consensus means that an agreement is reached regarding a certain variable of interest that depends on the states of all agents. Consensus guarantees that agents communicating information according to a network topology is consistent. This is critical to any type of coordination task such as rendezvous [28], swarm [29] or formation [30] [31]. Characterization of formation control schemes are often based on sensing capability and the communication topology of agents. The three main types of formation control schemes identified by [32] are described below.

One type of control is distance based, where inter-agent distances are actively controlled for desired formation. Each agent can sense relative positions of neighbours with respect to their local coordinate system (LCS). It is important to note that LCS orientations are not always aligned with each other, and due to the neighbour dependency, more interactions are crucial.

In displacement based control, agents control displacements of neighbouring agents for desired formation. In this case, each agent only senses the relative positions of neighbour agents with respect to the global coordinate system (GCS). However, each agent does not require their positions with respect to the GCS, only knowledge of the GCS orientation. This type of control, has a moderate tradeoff between sensing capability and interactions.

The type of consensus control used in this work is position based, where agents sense their own positions with respect to a GCS. Each agent must control their own positions to achieve the overall desired formation. With this type of control, a greater sensing capability among agents is required.

A consensus algorithm (or protocol) is the method of negotiation or the interaction rule used to reach consensus. The main component of a consensus algorithm is the update law, which is designed such that the information state of all agents converges to a common value. For MASs with single integrator as

$$\dot{\xi}_i = u_i \ , \tag{2.6}$$

a common linear consensus algorithm is,

$$u_i = \sum_{i=1}^{n_a} a_{ij}(\xi_j - \xi_i) \ , \tag{2.7}$$

where (2.7) is the control input and ξ_i is the i^{th} agent's information state. The objective of (2.7) is to guarantee consensus for any initial condition $\xi_i(0)$ as $t \to \infty$.

If a formation is desired, the input can be designed with relative distance ξ_{r_i} between agents,

$$u_i = \sum_{i=1}^{n_a} a_{ij}(\xi_j - \xi_i - \xi_{r_i}) \ . \tag{2.8}$$

Consider the calculation for relative distances ξ_{r_i} used to achieve a ring formation in the x-y plane. First define the desired distance to the center of the ring as r_d. The relative x-y distance between an agent and the ring's center is calculated as

$$r_{c_i} = r_d R_i r_0 \ , \tag{2.9}$$

where $r_0 = [1 \ 0]^\top$ is used to select matrix elements and R_i is the rotation matrix shown below:

$$R_i = \begin{bmatrix} \cos(\gamma_i) & -\sin(\gamma_i) \\ \sin(\gamma_i) & \cos(\gamma_i) \end{bmatrix} \ . \tag{2.10}$$

In (2.10), γ_i is the accumulative angle for each agent around the ring; the angle separation with respect to agent 1.

The vector of relative distances between agent 1 and the remaining agents can be extracted using

$$r_{i1} = r_{c_i} - r_{c_1} \ . \tag{2.11}$$

Then the final relative distances between agents is determined as,

$$\xi_{r_i} = r_{j1} - r_{i1}, \quad \text{for } i, j = 1, ..., n_a \ . \tag{2.12}$$

An example of consensus with a ring formation is presented, where $r_d = 10$ meters, the number of agents is six and the communication topology is undirected according to

$$\mathcal{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} .$$

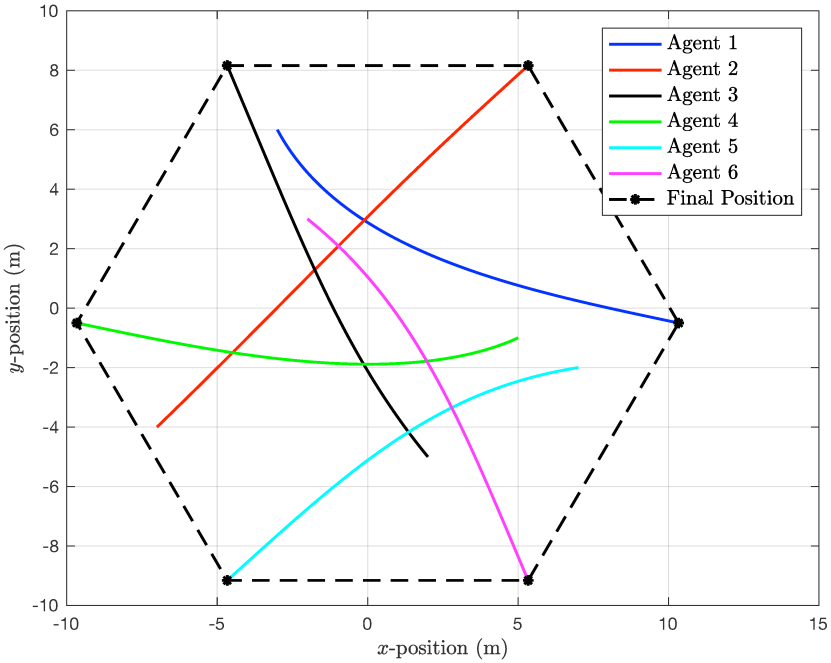

Figure 2.5: Consensus with ring formation

In this case,

$$r_{11} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, r_{21} = \begin{bmatrix} -5 \\ 8.6 \end{bmatrix}, r_{31} = \begin{bmatrix} -15 \\ 8.6 \end{bmatrix}, r_{41} = \begin{bmatrix} -20 \\ 0 \end{bmatrix}, r_{51} = \begin{bmatrix} -15 \\ -8.6 \end{bmatrix}, r_{61} = \begin{bmatrix} -5 \\ -8.6 \end{bmatrix}.$$

The 2-dimensional result of the simulation is shown in Fig.2.5.

2.3 Distributed Control vs. Centralized Control

There are typically two features to consider when developing a control approach for a multi-agent system. One feature is the distribution of information, which usually refers to associated communication costs. The other feature is the complexity of the problem, which directly relates to the computation time.

In a centralized approach, available information about all agents is collected at a single location, as shown in Fig.2.6. With the substantial advances in computer electronics, specifically memory and processing power, the centralized control structure can be advantageous because it can produce system wide solutions based on consistent system wide information. This is because a global optimal control problem must be solved with respect to all agent actuators, given the entire set of information states.

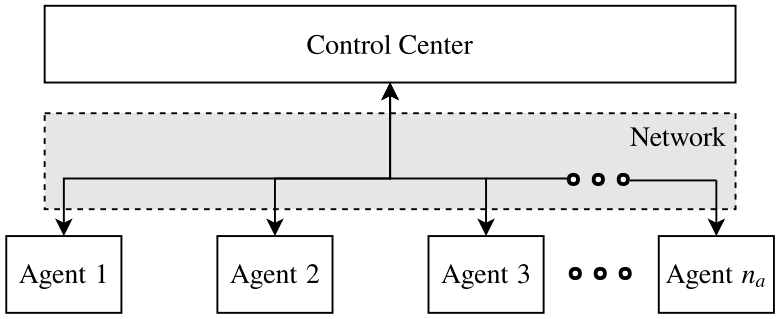

Figure 2.6: Centralized schematic diagram

However, distributing locally collected information will typically result in a larger communication cost and can be disadvantageous because the centralized approach depends on the reliability of inter-agent communication links. The complexity of the centralized approach requires significant computation to determine a solution. In this regard, feasibility depends on the computational capability at the single centralized location. A numerical simulation study was performed in [21] to compare computation times for a team of UAVs under distributed and centralized control. The results showed that the computation time required for distributed control was significantly lower than the centralized approach as the number of agents increased. Therefore, it can be reasoned that the centralized approach will fail for large sizes of MASs, or if the agent responsible for system management/processing is disabled.

A distributed approach is based on local information states communicated by the agent's neighbours, as shown in Fig.2.7. Therefore, the control problem can be divided into a set of smaller local optimization problems. This allows for a small computation time for each agent. However, since the optimization problem is based

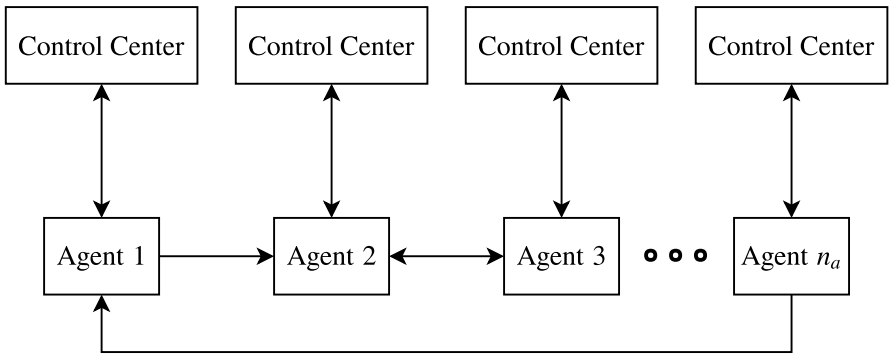

Figure 2.7: Distributed schematic diagram

on local information states, it results in a suboptimal solution of control actions [33]; this is the main disadvantage of a distributed control structure. The scale of suboptimality depends on the communication topology (degree of interaction) of the multi-agent system [34]. The problem of local information is also significant if an agent is attempting to predict a neighbour's or the group's behaviour. A distributed control scheme can also be viewed as modular, as in each agent is responsible for itself. This is advantageous because possible failures will not necessarily affect the overall multi-agent system [35].

Chapter 3

System Modelling

A type of UAV is a quadcopter, also known as a quadrotor. A quadcopter has four rotors directed upwards on each arm and the four arms are arranged in square formation. A quadcopter is capable of vertical takeoff and landing and its inherent dynamic nature allows for maneuverability [36]. For modelling purposes, it is assumed that the quadcopter is manufactured with identical arm lengths and symmetric mass distribution. This chapter presents general quadcopter dynamics and a linearized model for multiple quadcopter control from a Euler-Lagrange formulation.

3.1 Quadcopter Dynamics

Movement of a quadcopter is defined using two reference frames as shown in Fig.3.1. The inertial frame (3.1), which is considered earth-fixed, is made up of x, y, z-axes and is used to reference the absolute linear position. The angular position (3.2) of the quadcopter in the inertial frame is defined using the Euler angles.

$$\xi = \begin{bmatrix} x & y & z \end{bmatrix}^\top , \tag{3.1}$$

$$\eta = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^\top . \tag{3.2}$$

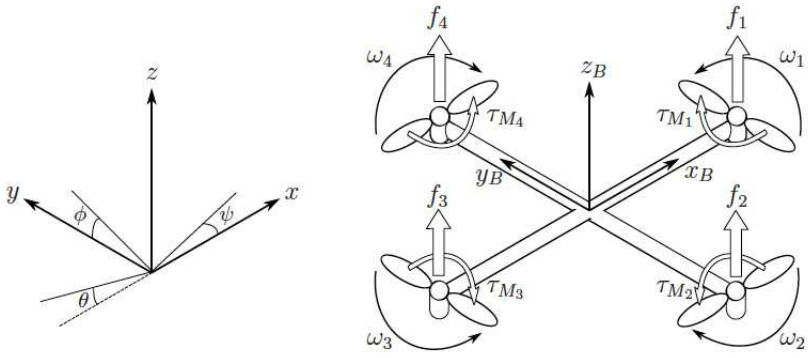

Figure 3.1: Quadcopter frame of reference [1]

The roll angle ϕ signifies the rotation about the x-axis, the pitch angle θ is the rotation about the y-axis, and the yaw angle refers to the rotation around the z-axis.

The body frame is made up of x_B, y_B, z_B-axes and its origin occurs at the center of mass of the quadcopter. It is assumed that the body frame is attached to the quadcopter and is therefore, a moving coordinate system. The linear and angular velocities with respect to the body frame are defined as (3.3) and (3.4), respectively.

$$V = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}^\top \tag{3.3}$$

$$\nu = \begin{bmatrix} p & q & r \end{bmatrix}^\top \tag{3.4}$$

The rotation matrix can be used to move from the body frame to the inertial frame and is defined as

$$R_i^b = \begin{bmatrix} C_\psi C_\theta & C_\psi S_\theta S_\phi - S_\psi C_\phi & C_\psi S_\theta C_\phi + S_\psi S_\phi \\ S_\psi C_\theta & S_\psi S_\theta S_\phi + C_\psi C_\phi & S_\psi S_\theta C_\phi - C_\psi S_\phi \\ -S_\theta & C_\theta S_\phi & C_\theta C_\phi \end{bmatrix}, \tag{3.5}$$

where $C_x = \cos(x)$ and $S_x = \sin(x)$. Since the rotation matrix is orthogonal, the rotation matrix from the inertial frame to the body frame is $\left(R_i^b\right)^{-1} = \left(R_i^b\right)^\top$.

The angular velocities ν in the body frame can be transformed to the Euler angular velocities $\dot{\eta}$ in the inertial frame and vice versa using

$$\nu = W_\eta \dot{\eta} \ , \tag{3.6}$$

and

$$\dot{\eta} = W_\eta^{-1} \nu \ , \tag{3.7}$$

respectively, where the transformation matrix is given as

$$W_\eta = \begin{bmatrix} 1 & 0 & -S_\theta \\ 0 & C_\phi & C_\theta S_\phi \\ 0 & -S_\phi & C_\theta C_\phi \end{bmatrix}. \tag{3.8}$$

The thrust produced by each rotor is proportional to the square angular velocity [1] and is given by

$$f_i = k\omega_i^2 \ , \qquad i = 1, 2, 3, 4, \tag{3.9}$$

where k is a lift constant assumed to be the same for all rotors. The total thrust in z_B-direction is the sum of forces produced by the rotors as

$$T = \sum_{i=1}^{4} f_i \ . \tag{3.10}$$

Therefore, the force input vector in the body frame directions is $f_B = [0 \ \ 0 \ \ T]^\top$.

The torque around each rotor axis is found using $\tau_{M_i} = b\omega_i^2$, where b is a drag constant. Therefore, the torque input vector can be calculated as

$$\tau = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} l(f_4 - f_2) \\ l(f_3 - f_1) \\ \sum_{i=1}^{4} \tau_{M_i} \end{bmatrix}, \tag{3.11}$$

where l is the length from the center of mass of the quadcopter to the rotor.

3.2 Quadcopter Model

The linear quadcopter model is developed using the Lagrangian approach and linearization about an equilibrium point [37].

In general, the Lagrangian is the sum of kinetic energy minus the potential energy of the system. For the quadcopter dynamics, the linear and angular components are separate and cane therefore be studied independently; the Lagrangian has translational and rotational kinetic energies and the gravitational potential energy. Define $q = [\xi^\top \ \ \eta^\top]^\top$, then the Lagrangian is,

$$\begin{aligned} \mathcal{L}(q, \dot{q}) &= E_{trans} + E_{rot} - E_{pot} \\ &= \tfrac{1}{2} m \dot{\xi}^\top \dot{\xi} + \tfrac{1}{2} \dot{\eta}^\top \underbrace{W_\eta^\top I W_\eta}_{\mathcal{J}} \dot{\eta} - mgz \ , \end{aligned} \tag{3.12}$$

where m denotes the mass of the quadcopter and payload, g is the gravitational constant and $I = diag(I_{xx}, I_{yy}, I_{zz})$ is the inertia matrix for the quadcopter. Since it is assumed that the quadcopter is symmetric, then $I_{xx} = I_{yy}$.

The full quadcopter dynamics is obtained from the Euler-Lagrange equations with

external forces and torques. The Euler-Lagrange translational equation is

$$\frac{d}{dt}\left(\frac{\partial E_{trans}}{\partial \dot{\xi}}\right) - \frac{\partial E_{trans}}{\partial \xi} = R_i^b \begin{bmatrix} 0 & 0 & T \end{bmatrix}^\top ,$$

$$m\ddot{\xi} + mg \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\top = R_i^b \begin{bmatrix} 0 & 0 & T \end{bmatrix}^\top . \tag{3.13}$$

The Euler-Lagrange rotational equation is

$$\frac{d}{dt}\left(\frac{\partial E_{rot}}{\partial \dot{\eta}}\right) - \frac{\partial E_{rot}}{\partial \eta} = \tau$$

$$\mathcal{J}\ddot{\eta} + \left(\dot{\mathcal{J}} - \frac{1}{2}\frac{\partial}{\partial \eta}(\dot{\eta}^\top \mathcal{J})\right)\dot{\eta} = \tau$$

$$\mathcal{J}\ddot{\eta} + C(\eta, \dot{\eta})\dot{\eta} = \tau , \tag{3.14}$$

where $C(\eta, \dot{\eta})$ is the Coriolis matrix; refer to Appendix A for the terms in the Coriolis matrix.

The Euler-Lagrange equations (3.13) and (3.14) can be rearranged and written as

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \dot{z} \end{bmatrix} = \frac{T}{m} \begin{bmatrix} -S_\theta \\ C_\theta S_\phi \\ C_\theta C_\phi \end{bmatrix} - g \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} , \tag{3.15}$$

and

$$\mathcal{J}\ddot{\eta} = \tau - C(\eta, \dot{\eta})\dot{\eta} . \tag{3.16}$$

To simplify the rotational dynamics in (3.16), a new input variable [38] can be defined as

$$\tau = C(\eta, \dot{\eta})\dot{\eta} + \mathcal{J}\bar{\tau} . \tag{3.17}$$

Substituting (3.17) into (3.16), results in $\mathcal{J}\ddot{\eta} = \mathcal{J}\bar{\tau}$. Similar to [39], the full

quadcopter dynamics from (3.15) and (3.16) are rewritten as,

$$m\ddot{x} = -u\sin(\theta)\ , \tag{3.18}$$

$$m\ddot{y} = u\cos(\theta)\sin(\phi)\ , \tag{3.19}$$

$$m\ddot{z} = u\cos(\theta)\cos(\phi) - mg\ , \tag{3.20}$$

$$\ddot{\theta} = \tau_\theta\ , \tag{3.21}$$

$$\ddot{\phi} = \tau_\phi\ , \tag{3.22}$$

$$\ddot{\psi} = \tau_\psi\ . \tag{3.23}$$

Define the following states and input variables,

$$\begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} & x_{12} \end{bmatrix}^\top = \begin{bmatrix} x & \dot{x} & y & \dot{y} & z & \dot{z} & \psi & \dot{\psi} & \theta & \dot{\theta} & \phi & \dot{\phi} \end{bmatrix}^\top , \tag{3.24}$$

$$\begin{bmatrix} u_1 & u_2 & u_3 & u_4 \end{bmatrix}^\top = \begin{bmatrix} T - mg & \tau_\psi & \tau_\theta & \tau_\phi \end{bmatrix}^\top , \tag{3.25}$$

then equations (3.18)-(3.23) can be written in a state space form as

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \\ \dot{x}_7 \\ \dot{x}_8 \\ \dot{x}_9 \\ \dot{x}_{10} \\ \dot{x}_{11} \\ \dot{x}_{12} \end{bmatrix} = \begin{bmatrix} x_2 \\ -\frac{u_1}{m}\sin x_9 - g\sin x_9 \\ x_4 \\ \frac{u_1}{m}\cos x_9 \sin x_{11} + g\cos x_9 \sin x_{11} \\ x_6 \\ \frac{u_1}{m}\cos x_9 \cos x_{11} + g\cos x_9 \cos x_{11} - g \\ x_8 \\ u_2 \\ x_{10} \\ u_3 \\ x_{12} \\ u_4 \end{bmatrix} . \tag{3.26}$$

The solution of nonlinear quadcopter dynamics is complicated because of the dependency on aerodynamics forces and moments due to flight conditions, such as, altitude, speed and weight. A simpler approach is based on a linearized model that

describes the quadcopter's motion provided that the perturbations from a known equilibrium state is small.

From the nonlinear quadcopter dynamics produced in (3.26), linearization by Taylor series expansion about an equilibrium point equal to the origin, with $f(0,0) = 0$, produces the quadcopter dynamics modelled by the following linear state equations; similarly used in [40] [41],

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -g & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tau_\theta \ , \tag{3.27}$$

$$\begin{bmatrix} \dot{y} \\ \ddot{y} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & g & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} y \\ \dot{y} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tau_\phi \ , \tag{3.28}$$

$$\begin{bmatrix} \dot{z} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} z \\ \dot{z} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} \overline{T} \ , \tag{3.29}$$

$$\begin{bmatrix} \dot{\psi} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \psi \\ \dot{\psi} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tau_\psi \ . \tag{3.30}$$

The linearized model is decoupled into four sets of equations and is valid given small angle values for θ and ϕ. The flight dynamics in the x, y, ψ-directions are subject to control moments that generate torques around the y, x, z-axes, respectively. The motion in the z-direction is generated via a thrust force, which is the sum of the thrusts from all motors minus the force due to gravity.

Chapter 4

Model Predictive Control for Consensus of Multiple Quadcopters

A brief overview of MPC and its components are presented in this chapter. The general approach in MPC schemes is to utilize a mathematical model to predict the system future output given a set of optimized control actions. The future outputs (prediction) are determined across the prediction horizon n_p, as shown in Fig.4.1, using the system model, past outputs, past inputs and the future inputs (control moves) to be sent to the system. The future control moves are solved by applying

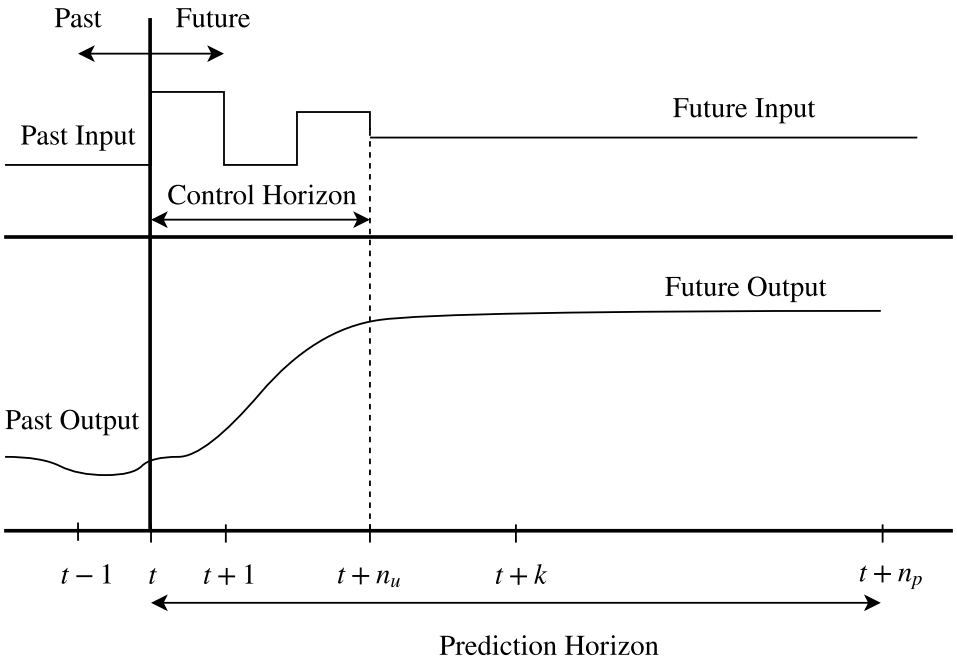

Figure 4.1: Model predictive control strategy

optimization over a finite control horizon. The objective of the cost function is for the system output to be as close as possible to the desired setpoints [42]. This is achieved by formulating the cost function to minimize the error between the prediction and the setpoints; as illustrated in Fig.4.2 where the future errors and cost function is

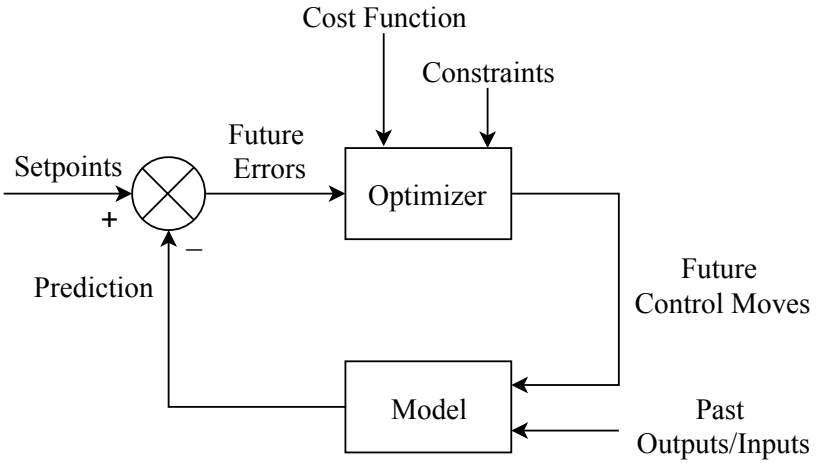

Figure 4.2: Basic model predictive control block diagram

sent to the MPC optimizer. If the cost function is quadratic and unconstrained, and the model is linear, then an explicit solution can be determined analytically. Otherwise, if the cost function is subject to constraints, then a numerical solver must be implemented. According to the receding horizon policy, the first control move of the optimal control sequence is applied. This is because new output values are obtained at the next sampling instant and the prediction is updated based on these output measurements [43]. The receding horizon policy is especially recommended if setpoint changes are expected.

The MPC algorithm implemented in this work is a dynamic matrix control (DMC) method. DMC is a mature technique that was first developed in the seventies by Cutler and Ramaker [44]. It utilizes a dynamic step response model, developed from experimental data, to predict the future control moves. DMC has the ability to handle hard constraints and can correct for modelling mismatch and nonlinearities.

4.1 Dynamic Matrix Control

The MPC algorithm implemented requires a dynamic matrix to solve for the predicted response of the process output variable. The DMC algorithm cannot handle open loop unstable systems because the system will never reach steady state [45]. The linear quadcopter model (3.27)-(3.30) presented in Section 3.2 is marginally stable [46]. Therefore, feedback gains are used to obtain a stable step response; similar to [47]. Full state feedback gains are designed by selecting desired closed loop poles and solved

using Ackermann's formula [48]; feedback gains are implemented for all directions,

$$\begin{aligned} u_{fx} &= -K_x \begin{bmatrix} x & \dot{x} & \theta & \dot{\theta} \end{bmatrix}^\top, \\ u_{fy} &= -K_y \begin{bmatrix} y & \dot{y} & \phi & \dot{\phi} \end{bmatrix}^\top, \\ u_{fz} &= -K_z \begin{bmatrix} z & \dot{z} \end{bmatrix}^\top, \\ u_{f\psi} &= -K_\psi \begin{bmatrix} \psi & \dot{\psi} \end{bmatrix}^\top. \end{aligned} \tag{4.1}$$

The dynamic matrix G is developed with normalized step response coefficients as shown in (4.2), where n_u represents the number of control moves to be evaluated (length of the optimal control sequence), also know as the control horizon. Note that the prediction horizon, n_p should ideally contain the number of discrete intervals needed to reach at least 95% of the steady state.

$$G = \begin{bmatrix} g_1 & 0 & \cdots & 0 \\ g_2 & g_1 & \cdots & 0 \\ g_3 & g_2 & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots \\ g_{n_p} & g_{n_p-1} & \cdots & g_{n_p-n_u+1} \end{bmatrix}_{n_p \times n_u} \tag{4.2}$$

It is desired to control positions in the x, y, z, ψ-directions, therefore, dynamic matrices G_x, G_y, G_z and G_ψ are developed for each direction. It is assumed that the system dynamics do not change, therefore, the dynamic matrices are calculated once and remain constant.

In this work, the discrete form of the linear consensus algorithm in (2.7) is implemented as

$$\begin{cases} \tau_\phi &= \delta \sum_{j=1}^{n_a} a_{ij}(y_j - y_i - y_{r_{ij}}) \\ \tau_\theta &= \delta \sum_{j=1}^{n_a} a_{ij}(x_j - x_i - x_{r_{ij}}) \\ \bar{T} &= \delta \sum_{j=1}^{n_a} a_{ij}(z_j - z_i - z_{r_{ij}}) \\ \tau_\psi &= \delta \sum_{j=1}^{n_a} a_{ij}(\psi_j - \psi_i - \psi_{r_{ij}}), \end{cases} \tag{4.3}$$

for $i = 1, ..., n_a$, where x_i, y_i, z_i, ψ_i are the i^{th} agent's information states, and $x_{r_{ij}}$, $y_{r_{ij}}$, $z_{r_{ij}}$, $\psi_{r_{ij}}$ are the predetermined relative state from agent i to j and time step δ. This consensus algorithm is used to determine the desired positions x_d, y_d, z_d, ψ_d.

A setpoint tuning parameter α, can be applied to reduce the aggressiveness of the control action by creating a smooth first order approach to the desired position. This is expressed in (4.4), where t is the time and k is an index from 1 to n_p,

$$\begin{aligned} x_{sp}(t+k) &= \alpha x_{sp}(t+k-1) + (1-\alpha)x_d \\ y_{sp}(t+k) &= \alpha y_{sp}(t+k-1) + (1-\alpha)y_d \\ z_{sp}(t+k) &= \alpha z_{sp}(t+k-1) + (1-\alpha)z_d \\ \psi_{sp}(t+k) &= \alpha \psi_{sp}(t+k-1) + (1-\alpha)\psi_d \ . \end{aligned} \tag{4.4}$$

The prediction error is determined as the difference between the setpoints and the future prediction as,

$$\begin{aligned} e_x &= x_{sp} - \hat{x} \\ e_y &= y_{sp} - \hat{y} \\ e_z &= z_{sp} - \hat{z} \\ e_\psi &= \psi_{sp} - \hat{\psi} \ . \end{aligned} \tag{4.5}$$

Note, α is a number between 0 and 1 and if applied, reduces the prediction error by modifying the setpoints. This intuitive modification affects the transient performance of the controller: as α increases, the closed loop response is slower.

The optimization function of the MPC controller to be minimized is shown in (4.6), where $\Delta u_x, \Delta u_y, \Delta u_\psi, \Delta u_z$ are the control moves based on $[u_x \ u_y \ u_\psi \ u_z] = [\tau_\theta \ \tau_\phi \ \tau_\psi \overline{T}]$.

$$\begin{aligned} J_x &= (\hat{x} - x_{sp})^2 + \Delta u_x^\top R_x \Delta u_x \ , \\ J_y &= (\hat{y} - y_{sp})^2 + \Delta u_y^\top R_y \Delta u_y \ , \\ J_z &= (\hat{z} - z_{sp})^2 + \Delta u_z^\top R_z \Delta u_z \ , \\ J_\psi &= (\hat{\psi} - \psi_{sp})^2 + \Delta u_\psi^\top R_\psi \Delta u_\psi \ . \end{aligned} \tag{4.6}$$

In general, $R_{(\cdot)} = diag(\lambda_{(\cdot)}, ..., \lambda_{(\cdot)})$ with tuning parameter $\lambda_{(\cdot)}$ for each direction x, y, z, ψ. The tuning parameter λ is implemented as move suppression to account for the aggressiveness of the closed-loop response by reducing the magnitude of $\Delta u_{(\cdot)}$.

Note, that the general form of the optimization problem in (4.6) can also be written as,

$$J = (G\Delta u - e)^2 + \Delta u^\top R \Delta u \ . \tag{4.7}$$

By taking the partial derivative and setting it equal to zero,

$$\frac{\partial J_x}{\partial \Delta u_x} = 0 \ , \quad \frac{\partial J_y}{\partial \Delta u_y} = 0 \ , \quad \frac{\partial J_z}{\partial \Delta u_z} = 0 \ , \quad \frac{\partial J_\psi}{\partial \Delta u_\psi} = 0 \ , \tag{4.8}$$

the unconstrained solution of the optimization functions are obtained as,

$$\begin{aligned}
\Delta u_x &= (G_x^\top G_x + R_x)^{-1} G_x^\top (x_{sp} - \hat{x}), \\
\Delta u_y &= (G_y^\top G_y + R_y)^{-1} G_y^\top (y_{sp} - \hat{y}), \\
\Delta u_z &= (G_z^\top G_z + R_z)^{-1} G_z^\top (z_{sp} - \hat{z}), \\
\Delta u_\psi &= (G_\psi^\top G_\psi + R_\psi)^{-1} G_\psi^\top (\psi_{sp} - \hat{\psi}).
\end{aligned} \tag{4.9}$$

As observed in (4.9), it is often difficult to estimate a small value for λ. Instead, the move suppression can be applied by multiplying the diagonals of the square matrix $G^\top G$, where λ is a number slightly larger than 1.

In general, when the optimal solution of (4.6) is determined, the control input is designed as

$$u = \Delta u + u^- \ , \tag{4.10}$$

where u^- represents the previous control input.

The generalized form for the future prediction $\hat{y}$ of the system response is shown in (4.11) as an example. The equation consists of three distinct terms that account for I) past, II) current and III) future control moves,

$$\begin{aligned}
\hat{y}(t+k\big|_t) = \overbrace{\hat{y}(t)}^{\text{II}} + \overbrace{\sum_{i=k-n_u-1}^{k} G(i)\Delta u(t+k-i\big|_t)}^{\text{II and III}} \\
+ \overbrace{\sum_{i=k+1}^{k+n_p-1} G(i) - G(i-k)\Delta u(t-k-i)}^{\text{I}} \ .
\end{aligned} \tag{4.11}$$

The future prediction is calculated according to the number of discrete intervals and updated by shifting the predictions forward:

$$\hat{x} = \begin{bmatrix} \hat{x}(t+1) \\ \vdots \\ \hat{x}(t+n_p) \end{bmatrix}, \quad \hat{y} = \begin{bmatrix} \hat{y}(t+1) \\ \vdots \\ \hat{y}(t+n_p) \end{bmatrix}, \quad \hat{z} = \begin{bmatrix} \hat{z}(t+1) \\ \vdots \\ \hat{z}(t+n_p) \end{bmatrix}, \quad \hat{\psi} = \begin{bmatrix} \hat{\psi}(t+1) \\ \vdots \\ \hat{\psi}(t+n_p) \end{bmatrix}. \tag{4.12}$$

With the new control input the future prediction is evaluated as

$$\begin{aligned}
\hat{x} &= \hat{x} + G_x \Delta u_x \ , \\
\hat{y} &= \hat{y} + G_y \Delta u_y \ , \\
\hat{z} &= \hat{z} + G_z \Delta u_z \ , \\
\hat{\psi} &= \hat{\psi} + G_\psi \Delta u_\psi \ ,
\end{aligned} \tag{4.13}$$

where according to the receding horizon policy only the first optimal control move in the sequence is applied.

From the current measured value of the response, (4.14) is used to adjust the future system dynamic prediction to account for modelling mismatch or nonlinearities in the model,

$$\begin{aligned}
\Phi_x\big|_t &= x_m\big|_t - \hat{x}(t+1)\big|_t \,, \\
\Phi_y\big|_t &= y_m\big|_t - \hat{y}(t+1)\big|_t \,, \\
\Phi_z\big|_t &= z_m\big|_t - \hat{z}(t+1)\big|_t \,, \\
\Phi_\psi\big|_t &= \psi_m\big|_t - \hat{\psi}(t+1)\big|_t \,.
\end{aligned} \tag{4.14}$$

4.2 General Effects of Prediction and Control Horizon

It is often difficult to determine the prediction horizon or control horizon. If a desired sampling time is known, then the prediction horizon can be determined based on an accurate reflection of the step response. Otherwise, some consequential control characteristics should be consider.

A short prediction horizon deprives the control optimization of important information. This short-sighted control produces aggressive control actions and instability can result. Theoretically, a prediction horizon equal to infinity will capture the full effects of the control action, however, this is not computationally feasible. In addition, if the matrix resulting from the least squares algorithm ($G^\top G$) is ill-conditioned, difficulty in controlling outputs can occur. This can be moderately corrected using substantive values of move suppression, which increases the magnitude of the diagonal elements of the matrix inverted in the least squares calculation. Note that an ill-conditioned matrix can lead to numerical issues, specifically, inversion of a nearly singular matrix. Large control horizons can introduce more zeros into the dynamic matrix, especially if deadtime is present; inversion of such a matrix should also be considered. In general, a small control horizon can produce cautious control actions.

Chapter 5

Optimization in Collision and Obstacle Avoidance

To ensure a successful group performance, a safe relative distance between agents should be enforced. Collision and obstacle avoidance among agents is achieved by applying output constraints on the MPC optimization function of each controlled position variable.

5.1 Output Constraints

The generation of output constraints necessary for collision and obstacle avoidance require a few parameters to first be defined. First, the actual relative distance is calculated using the current positions of agent i and neighbour agents j,

$$r = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}\ . \tag{5.1}$$

The predicted relative distance is calculated using the predicted positions of agent i $(\hat{x}_i, \hat{y}_i, \hat{z}_i)$ and the current position of neighbour agents j,

$$r_p = \sqrt{(\hat{x}_i - x_j)^2 + (\hat{y}_i - y_j)^2 + (\hat{z}_i - z_j)^2}\ . \tag{5.2}$$

As shown in Fig.5.1, a spherical safety region around each agent is the space that no object should enter and is defined by a radius r_s. Therefore, the relative safety distance between any two agents is $2r_s$. The avoidance distance r_a is used to apply output constraints before reaching the relative safety distance. With $0 < 2r_s < r_a$,

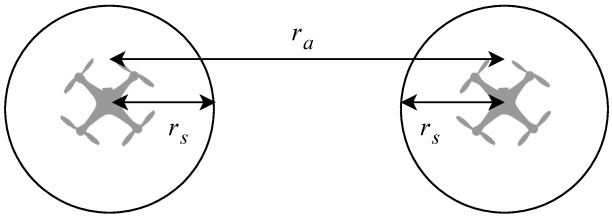

Figure 5.1: Safety and avoidance distances

output constraints are applied when the condition (5.3) is valid:

$$r_p \leq r_a \ . \tag{5.3}$$

The avoidance distance cannot be greater than the smallest relative distance between agents according to the desired geometric formation. Otherwise, output constraints will be generated and affected agents will be unable to reach the final desired formation.

To properly formulate constraints, the shape of the future prediction must be evaluated. As a general example, consider Fig.5.2; the general constraint value σ in Prediction I is a maximum constraint while in Prediction II is a minimum constraint.

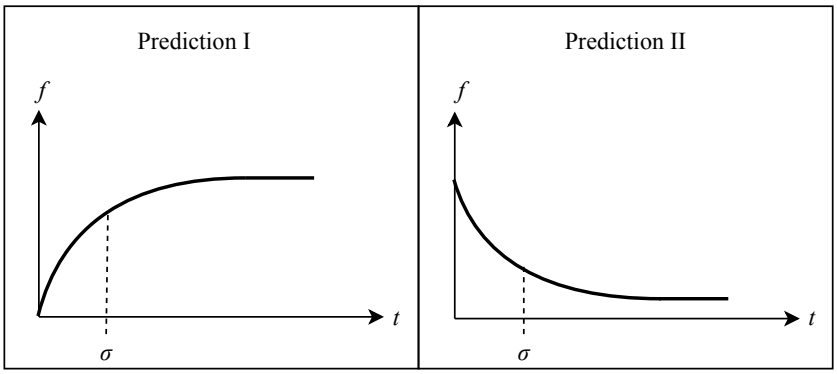

Figure 5.2: Constraint generation considering shape of future prediction with f being the general variable

In this work eight shapes were included, as shown in Fig.5.3, when formulating constraints. This is because in simulation, positions can be positive or negative. The additional shapes are needed, since minimum or maximum constraints change depending on a negative or positive position value. Identification of the prediction shape is guaranteed by determining the relationship between σ_1 and σ_2. Let σ_1 be the constraint value in question and σ_2 be a value at the end of the prediction horizon. Note, it is important to determine the constraint value σ_1, according to (5.3), as the earliest occurrence along the prediction horizon. A summary of the identification and constraints according to shape in the x, y, z-directions are presented in Table 5.1.

The constraint equations for the controlled variables (except ψ) are formulated

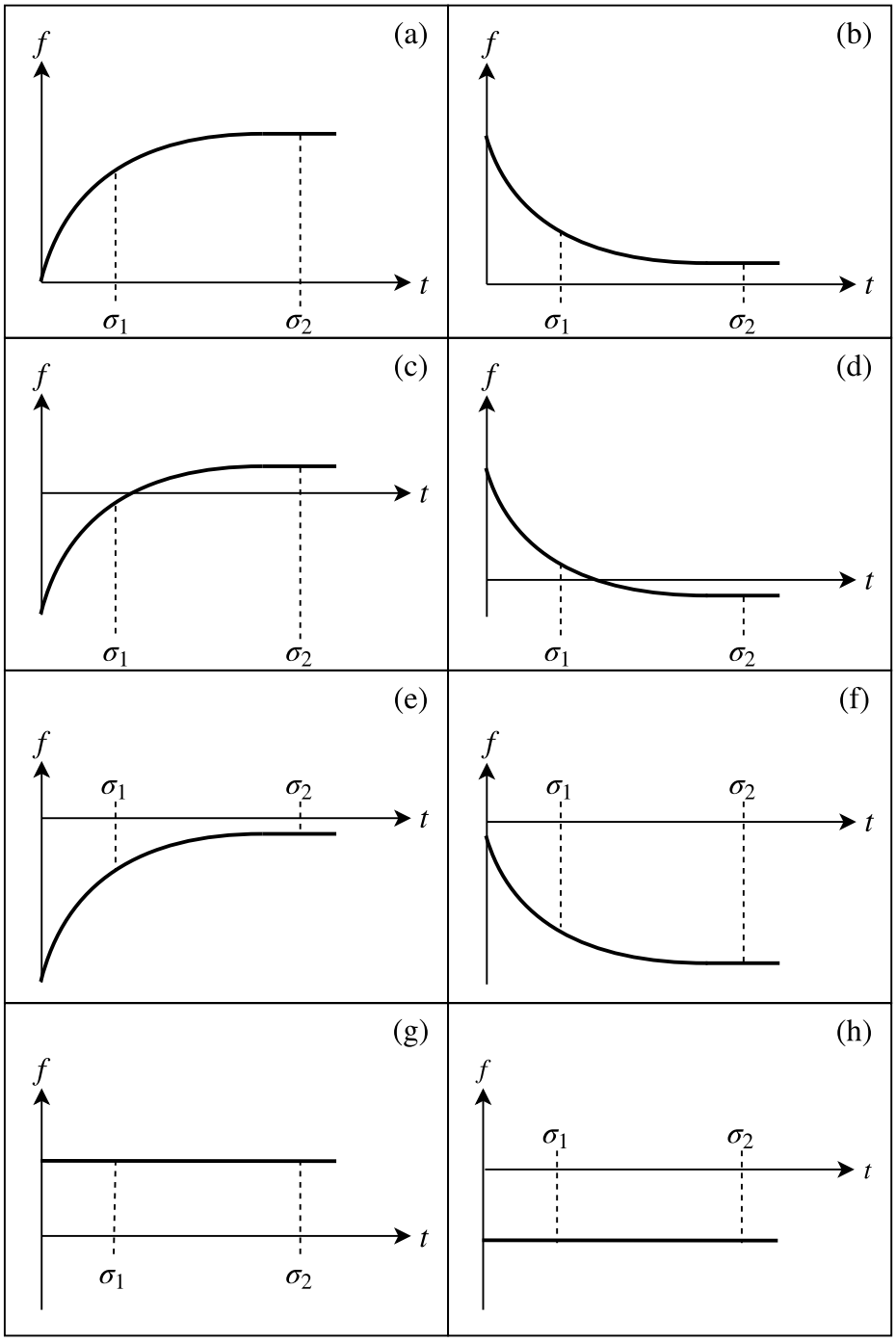

Figure 5.3: Future prediction shapes used for determining constraints

Table 5.1: Identification and prediction shapes for constraint generation

Shape	Identification	x constraints	y constraints	z constraints
Fig.5.3 (a)	$\sigma_1 < \sigma_2$ $\sigma_1 > 0$ $\sigma_2 > 0$	$x_{max} = \sigma_1$ $x_{min} = -\text{inf}$	$y_{max} = \sigma_1$ $y_{min} = -\text{inf}$	$z_{max} = \sigma_1$ $z_{min} = 0$
Fig.5.3 (b)	$\sigma_1 > \sigma_2$ $\sigma_1 \geq 0$ $\sigma_2 > 0$	$x_{max} = +\text{inf}$ $x_{min} = \sigma_1$	$y_{max} = +\text{inf}$ $y_{min} = \sigma_1$	$z_{max} = +\text{inf}$ $z_{min} = \sigma_1$
Fig.5.3 (c)	$\sigma_1 < \sigma_2$ $\sigma_1 \leq 0$ $\sigma_2 > 0$	$x_{max} = \sigma_1$ $x_{min} = -\text{inf}$	$y_{max} = \sigma_1$ $y_{min} = -\text{inf}$	$z_{max} = +\text{inf}$ $z_{min} = 0$
Fig.5.3 (d)	$\sigma_1 > \sigma_2$ $\sigma_1 > 0$ $\sigma_2 < 0$	$x_{max} = +\text{inf}$ $x_{min} = \sigma_1$	$y_{max} = +\text{inf}$ $y_{min} = \sigma_1$	$z_{max} = +\text{inf}$ $z_{min} = \sigma_1$
Fig.5.3 (e)	$\sigma_1 < \sigma_2$ $\sigma_1 < 0$ $\sigma_2 < 0$	$x_{max} = \sigma_1$ $x_{min} = -\text{inf}$	$y_{max} = \sigma_1$ $y_{min} = -\text{inf}$	$z_{max} = +\text{inf}$ $z_{min} = 0$
Fig.5.3 (f)	$\sigma_1 > \sigma_2$ $\sigma_1 \leq 0$ $\sigma_2 < 0$	$x_{max} = +\text{inf}$ $x_{min} = \sigma_1$	$y_{max} = +\text{inf}$ $y_{min} = \sigma_1$	$z_{max} = +\text{inf}$ $z_{min} = 0$
Fig.5.3 (g)	$\sigma_1 = \sigma_2$ $\sigma_1 > 0$	$x_{max} = \sigma_1$ $x_{min} = -\text{inf}$	$y_{max} = \sigma_1$ $y_{min} = -\text{inf}$	$z_{max} = \sigma_1$ $z_{min} = 0$
Fig.5.3 (h)	$\sigma_1 = \sigma_2$ $\sigma_1 \leq 0$	$x_{max} = \sigma_1$ $x_{min} = +\text{inf}$	$y_{max} = \sigma_1$ $y_{min} = +\text{inf}$	$z_{max} = +\text{inf}$ $z_{min} = 0$

below for maximum

$$\begin{aligned} G_x \Delta u_x &\leq x_{max} - x_{sp} + e_x \ , \\ G_y \Delta u_y &\leq y_{max} - y_{sp} + e_y \ , \\ G_z \Delta u_z &\leq z_{max} - z_{sp} + e_z \ , \end{aligned} \tag{5.4}$$

and minimum constraints

$$\begin{aligned} G_x \Delta u_x &\geq x_{min} - x_{sp} + e_x \ , \\ G_y \Delta u_y &\geq y_{min} - y_{sp} + e_y \ , \\ G_z \Delta u_z &\geq z_{min} - z_{sp} + e_z \ . \end{aligned} \tag{5.5}$$

5.2 Input Constraints

Limits on the control action are applied directly to the optimization function in the form of a logarithmic barrier function (5.6), similar components presented in [49] [50]. From the control action limit, the maximum and minimum rates of control are $\Delta u_{(\cdot),max}$ and $\Delta u_{(\cdot),min}$, respectively.

$$B(\Delta u_{(\cdot)}) = \mu \left(- \sum_{q=1}^{n_u} \ln \left\{ \Delta u_{(\cdot),max}(q) - \Delta u_{(\cdot)}(q) \right\} - \sum_{q=1}^{n_u} \ln \left\{ \Delta u_{(\cdot)}(q) - \Delta u_{(\cdot),min}(q) \right\} \right) . \tag{5.6}$$

The maximum control rate,

$$\begin{aligned} \Delta u_{x,max} &= u_{x,max} - u_x^- + u_{fx} \ , \\ \Delta u_{y,max} &= u_{y,max} - u_y^- + u_{fy} \ , \\ \Delta u_{z,max} &= u_{z,max} - u_z^- + u_{fz} \ , \\ \Delta u_{\psi,max} &= u_{\psi,max} - u_\psi^- + u_{f\psi} \ , \end{aligned} \tag{5.7}$$

and the minimum control rate,

$$\begin{aligned} \Delta u_{x,min} &= u_{x,min} - u_x^- + u_{fx} \ , \\ \Delta u_{y,min} &= u_{y,min} - u_y^- + u_{fy} \ , \\ \Delta u_{z,min} &= u_{z,min} - u_z^- + u_{fz} \ , \\ \Delta u_{\psi,min} &= u_{\psi,min} - u_\psi^- + u_{f\psi} \ , \end{aligned} \tag{5.8}$$

are calculated at each sampling instant, where $u_{(\cdot),max}$ and $u_{(\cdot),min}$ are the system's maximum and minimum control limit.

The penalty from the barrier increases and decreases smoothly as points move near to far from the limits. A positive factor μ can also be introduced to manipulate the severity of the penalty as boundaries on the control action are approached. Fig.5.4 illustrates the effects of μ on a logarithmic barrier function with limits $[\Delta u_{min}, \Delta u_{max}] = [-1,\ 1]$. If $\mu = 1$ is the point of reference, then $\mu > 1$ results in a severe penalty as the limits are approached and $\mu < 1$ gives a less severe penalty.

Constraints that cannot be violated are called hard constraints and the converse are referred to as soft constraints. Note that soft constraints do not necessarily violate

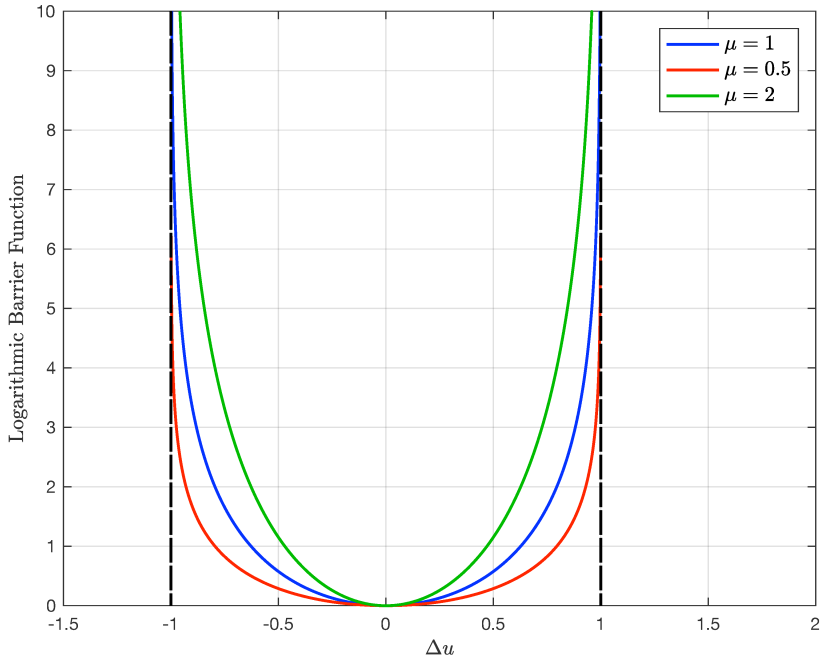

Figure 5.4: Logarithmic barrier function with varying μ

limits, rather they can be considered constraints that are not explicitly applied to the cost function. The logarithmic barrier function is used to reduce the number of active hard constraints on the MPC optimization function, which allows for a quicker computation. In addition, implementing constraints on input control rates means hard limits, like saturation, are not needed.

5.3 MPC Optimization Function

The optimization function of the x, y, z-direction controllers to be minimized, with input constraints and subject to output constraints are

$$\begin{aligned} J_x &= \tfrac{1}{2}(G_x\Delta u_x - e_x)^2 + \tfrac{1}{2}\Delta u_x^\top R_x \Delta u_x + B(\Delta u_x)\ , \\ J_y &= \tfrac{1}{2}(G_y\Delta u_y - e_y)^2 + \tfrac{1}{2}\Delta u_y^\top R_y \Delta u_y + B(\Delta u_y)\ , \\ J_z &= \tfrac{1}{2}(G_z\Delta u_z - e_z)^2 + \tfrac{1}{2}\Delta u_z^\top R_z \Delta u_z + B(\Delta u_z)\ . \end{aligned} \tag{5.9}$$

The optimization function to be minimized with only an input constraint is for the ψ-direction control,

$$J_\psi = \frac{1}{2}(G_\psi\Delta u_\psi - e_\psi)^2 + \frac{1}{2}\Delta u_\psi^\top R_\psi \Delta u_\psi + B(\Delta u_\psi)\ . \tag{5.10}$$

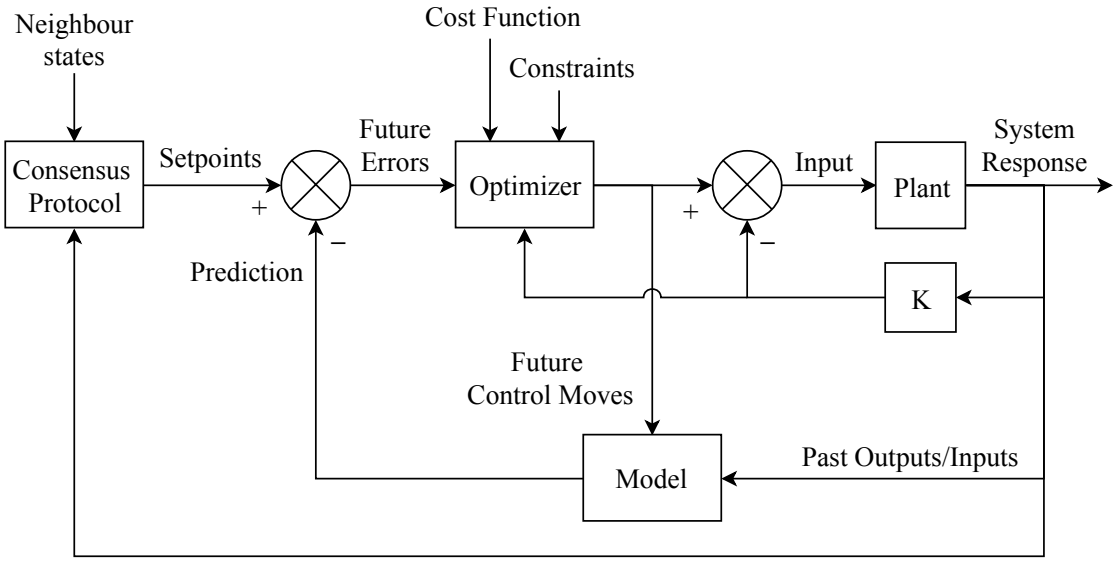

Figure 5.5: Block diagram of the proposed control strategy

The block diagram of the control strategy is shown in Fig.5.5. It depicts the general procedure as: neighbour states and agent states are received and used to determine desired setpoints. The prediction error enters the MPC subject to input and output constraints. The optimal solution calculated by the MPC is added with the feedback control, and this summation is the input to the quadcopter; this is done simultaneously for all agents in all x, y, z, ψ-directions.

5.4 Algorithm for Case Variations

An algorithm structure for the control strategy and variations will be presented in this section. An algorithm structure is used to effectively highlight the differences among the control variations, which are referred to as cases. The main difference among cases is the constraint generation for collision avoidance. Note that the positions referenced in the following algorithms are with respect to the absolute reference frame.

Case 1: In this case, output constraints are engaged if the actual relative distances are less than or equal to the avoidance distance. In addition, evasive action is only in the z-direction and is a calculated value. This evasive value is

$$z_e = r_a - r \ , \tag{5.11}$$

which is the distance required for the relative distance to be greater than the avoidance

distance. The evasive value is subtracted from the current z-position as

$$z_{max} = z_i - z_e \ , \tag{5.12}$$

to then be used to formulate a maximum constraint according to (5.4). The process evaluated by each agent can be found in Algorithm 1.

Algorithm 1 Collision avoidance

Require: All agents execute simultaneously.

1: **repeat**
2: Obtain current position and neighbour positions.
3: Determine desired positions using (4.3) and generate setpoints (4.4).
4: Adjust prediction with (4.14) and calculate the actual relative distances (5.1).
5: **if** $r \leq r_a$ **then**
6: Determine z_{max} according to (5.12). Generate output constraints (5.4) and (5.5).
7: Find solution to optimization functions (5.9) and (5.10) via numerical solver.
8: Apply the sum of MPC control input (4.10) and feedback gains (4.1).
9: Update prediction (4.13).
10: **until** Consensus = true

Case 2: In this case, output constraints are generated for only one (x, y or z) predetermined direction. The output constraints (for the particular direction) are engaged if the predicted relative distances are less than or equal to the avoidance distance. The process for Case 2 is outlined in Algorithm 2.

Case 3: In this case, output constraints are applied for all directions (x, y and z) and evasive movement occurs only in the z-direction. First, the actual relative distance between neighbours is checked using current positions. If the actual relative distance is less than or equal to the avoidance distance then an output constraint is generated to restrict the agent to its current position; this is done for x,y-directions.

The x and y constraints are generated as a minimum or maximum according the relationship between the current and previous position. For example, consider Fig.5.6, where x_1 represents the current position and x_0 represents the previous position of the agent. In event (a) x_1 would be a minimum, while in event (b) x_1 would be

Algorithm 2 Collision avoidance

Require: All agents execute simultaneously.

1: **repeat**
2: Obtain current position and neighbour positions.
3: Determine desired positions using (4.3) and generate setpoints (4.4).
4: Adjust prediction with (4.14) and calculate the predicted relative distances (5.2).
5: **if** $r_p \leq r_a$ **then**
6: Generate output constraints (5.4) and (5.5) in the predetermined direction, according to Table 5.1.
7: Find solution to optimization functions (5.9) and (5.10) via numerical solver.
8: Apply the sum of MPC control input (4.10) and feedback gains (4.1).
9: Update prediction (4.13).
10: **until** Consensus = true

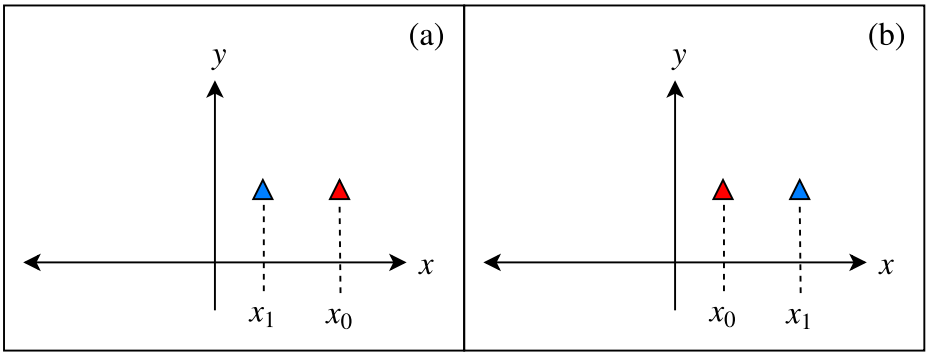

Figure 5.6: Previous and current x-positions for constraint generation

a maximum. All possible combinations of current and previous positions for x and y constraint generation are illustrated in Fig.B.1 and Fig.B.2, respectively; refer to Appendix B. A summary of the event identification and constraint generation for Case 3 can be found in Table 5.2.

Similar to Case 1, the evasive value in the z-direction is calculated using (5.11). The evasive value is either added to the current z-position

$$z_{min} = z_i + z_e \ , \tag{5.13}$$

or subtracted from the current z-position as

$$z_{max} = z_i - z_e \ . \tag{5.14}$$

Table 5.2: Event identification and constraint generation

Event	Identification	Max. constraints	Min. constraints
Fig.B.1 (a)	$x_0 > x_1,\ x_1 \geq 0,\ x_0 \geq 0$	$x_{max} = +\text{inf}$	$x_{min} = x_1$
Fig.B.2 (a)	$y_0 > y_1,\ y_1 \geq 0,\ y_0 \geq 0$	$y_{max} = +\text{inf}$	$y_{min} = y_1$
Fig.B.1 (b)	$x_0 < x_1,\ x_1 \geq 0,\ x_0 \geq 0$	$x_{max} = x_1$	$x_{min} = -\text{inf}$
Fig.B.2 (b)	$y_0 < y_1,\ y_1 \geq 0,\ y_0 \geq 0$	$y_{max} = y_1$	$y_{min} = -\text{inf}$
Fig.B.1 (c)	$x_0 < x_1,\ x_1 \geq 0,\ x_0 \leq 0$	$x_{max} = x_1$	$x_{min} = -\text{inf}$
Fig.B.2 (c)	$y_0 < y_1,\ y_1 \geq 0,\ y_0 \leq 0$	$y_{max} = y_1$	$y_{min} = -\text{inf}$
Fig.B.1 (d)	$x_0 > x_1,\ x_1 \leq 0,\ x_0 \geq 0$	$x_{max} = +\text{inf}$	$x_{min} = x_1$
Fig.B.2 (d)	$y_0 > y_1,\ y_1 \leq 0,\ y_0 \geq 0$	$y_{max} = +\text{inf}$	$y_{min} = y_1$
Fig.B.1 (e)	$x_0 > x_1,\ x_1 \leq 0,\ x_0 \leq 0$	$x_{max} = +\text{inf}$	$x_{min} = x_1$
Fig.B.2 (e)	$y_0 > y_1,\ y_1 \leq 0,\ y_0 \leq 0$	$y_{max} = +\text{inf}$	$y_{min} = y_1$
Fig.B.1 (e)	$x_0 < x_1,\ x_1 \leq 0,\ x_0 \leq 0$	$x_{max} = x_1$	$x_{min} = -\text{inf}$
Fig.B.2 (e)	$y_0 < y_1,\ y_1 \leq 0,\ y_0 \leq 0$	$y_{max} = y_1$	$y_{min} = -\text{inf}$

Whether the evasive value is added or subtracted, is predetermined for each agent. This feature allows some agents to have evasive action by increasing their z-position, while others have evasive action by decreasing their z-position. Note that in all cases, the complementary constraint for (5.13) and (5.14) is $z_{max} = +\text{inf}$ and $z_{min} = 0$, respectively. The structure of this collision avoidance strategy is shown in Algorithm 3.

Case 4: This case is the control strategy for obstacle avoidance. Output constraints in all directions are engaged if the predicted relative distances (between the agent and obstacles) are less than or equal to the avoidance distance. The process for the obstacle avoidance can be found in Algorithm 4.

Algorithm 3 Collision avoidance

Require: All agents execute simultaneously.

1: **repeat**
2: Obtain current position and neighbour positions.
3: Determine desired positions using (4.3) and generate setpoints (4.4).
4: Adjust prediction with (4.14) and calculate the actual relative distances (5.1).
5: **if** $r \leq r_a$ **then**
6: Determine z_{max} according to (**??**). Generate output constraints (5.4) and (5.5) using current positions for x,y-directions and z_{max} for the z-direction.
7: Find solution to optimization functions (5.9) and (5.10) via numerical solver.
8: Apply the sum of MPC control input (4.10) and feedback gains (4.1).
9: Update prediction (4.13).
10: **until** Consensus = true

Algorithm 4 Obstacle avoidance

Require: All agents execute simultaneously.

1: **repeat**
2: Obtain current position and obstacle positions.
3: Determine desired positions using (4.3) and generate setpoints (4.4).
4: Adjust prediction with (4.14) and calculate the predicted relative distances (5.2).
5: **if** $r_p \leq r_a$ **then**
6: Generate output constraints (5.4) and (5.5) according to Table 5.1
7: Find solution to optimization functions (5.9) and (5.10) via numerical solver.
8: Apply the sum of MPC control input (4.10) and feedback gains (4.1).
9: Update prediction (4.13).
10: **until** Consensus = true

Chapter 6

Simulation Studies on Collision Avoidance

This chapter presents the simulation results for the proposed control strategies outlined in Algorithms 1-3 from Chapter 5. Note that in order to better view simulation details some plots only display a portion of the full simulation time.

The simulation consists of a leaderless 4 agent team of quadcopters at rest and spatially distributed on the ground. The quadcopters are required to form a ring with a relative distance of 7.5 meters to the circle's center at a height of 15 meters. The initial quadcopter positions (x, y, z, ψ) are $Q1 = (2, 1, 0, 0)$, $Q2 = (-3, -7, 0, 0.1)$, $Q3 = (5, 5, 0, 0.4)$ and $Q4 = (8, 0, 0, 0.5)$; all remaining states are zero. The team of quadcopters exchange information with their direct neighbours; the resulting adjacency matrix is undirected

$$\mathcal{A} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}.$$

The radius of the safety region was chosen to be $r_s = 0.75$ meters; this is based on a quadcopter with arm lengths of 0.3 meters. Therefore, the relative safety distance between quadcopters is calculated as 1.5 meters. The avoidance distance is equal to 6 meters (Case 1), 7 meters (Case 2) and 5 meters (Case 3). Based on motor thrust details in [51], the control limits in simulation are ± 12 Newton-meters and ± 20 Newtons. Parameter selection is summarized in Table 6.1. Simulations were performed on Matlab 2017b and MPC optimization functions were solved using "fmincon" solver.

The full state feedback gains used to stabilize the quadcopter open loop response were designed by selecting desired closed loop poles $\begin{bmatrix} -1 & -1 & -1 & -1 \end{bmatrix}$; gain values

Table 6.1: Simulation Parameters

Definition	Value
mass (kg)	$m = 2$
sampling time (s)	$\delta = 0.05$
state feedback gains	$\begin{cases} K_x = \begin{bmatrix} -0.1019 & -0.4077 & 6 & 4 \end{bmatrix} \\ K_y = \begin{bmatrix} 0.1019 & 0.4077 & 6 & 4 \end{bmatrix} \\ K_z = \begin{bmatrix} 2 & 4 \end{bmatrix} \\ K_s = \begin{bmatrix} 1 & 2 \end{bmatrix} \end{cases}$
control horizon	$n_u = 2$
prediction horizon	$n_p = 180$
move suppression	$\lambda = 1.001$
setpoint tuning parameter	$\alpha = 0.6$
logarithmic barrier function parameter	$\mu = 1$

are presented in in Table 6.1. The normalized step response for the quadcopter dynamics in x, y, z, ψ-directions can be see in Fig.6.1. The normalized step responses are sampled to obtain the position coefficients for the dynamic matrices.

The 2-dimensional and 3-dimensional views of the results without a collision avoidance strategy are shown in Fig.6.2 and Fig.6.3 respectively. It is observed that the quadcopters can reach consensus and achieve the desired formation at the requested height of 15 meters. Note that the red circles represent points where the relative distance between two quadcopters was below the relative safety distance, and the black star is the final quadcopter position.

The relative distances between all combinations of quadcopters are presented in Fig.6.4, where Qij for $i, j = 1, ..., n_a$ and $i \neq j$ in the figure legends refers to quadcopter i and quadcopter j. Between 3 and 6 seconds, all four quadcopters violate the relative safety distance with another quadcopter (Q14 and Q23). Quadcopters 2 and 3 have the smallest violating relative distance of 0.0573 meters while Quadcopters 1 and 4 have a relative distance of 0.6462 meters.

Although collisions occurred, it is observed in Fig.6.5 that the input was successfully constrained using the logarithmic barrier function.

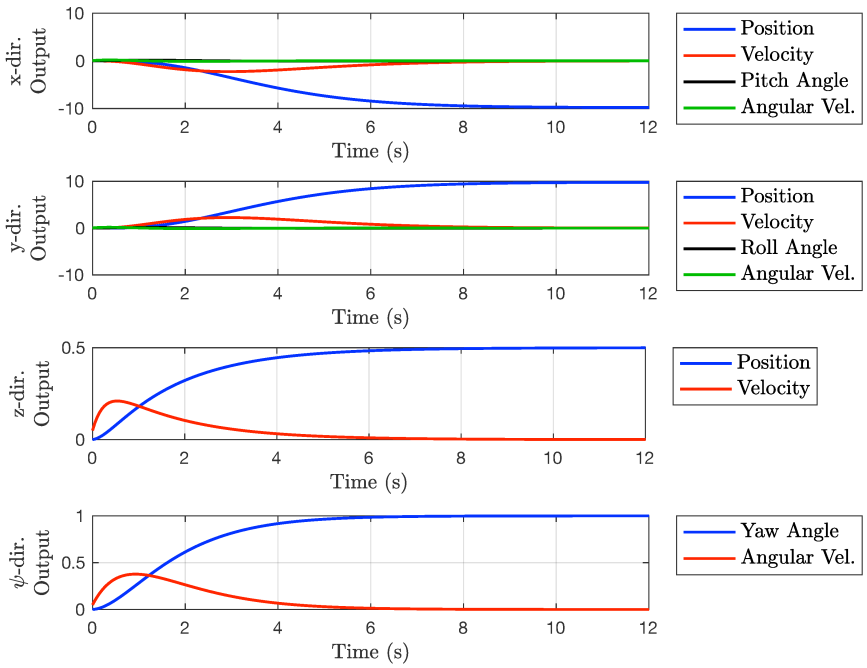

Figure 6.1: Normalized step response for quadcopter dynamics

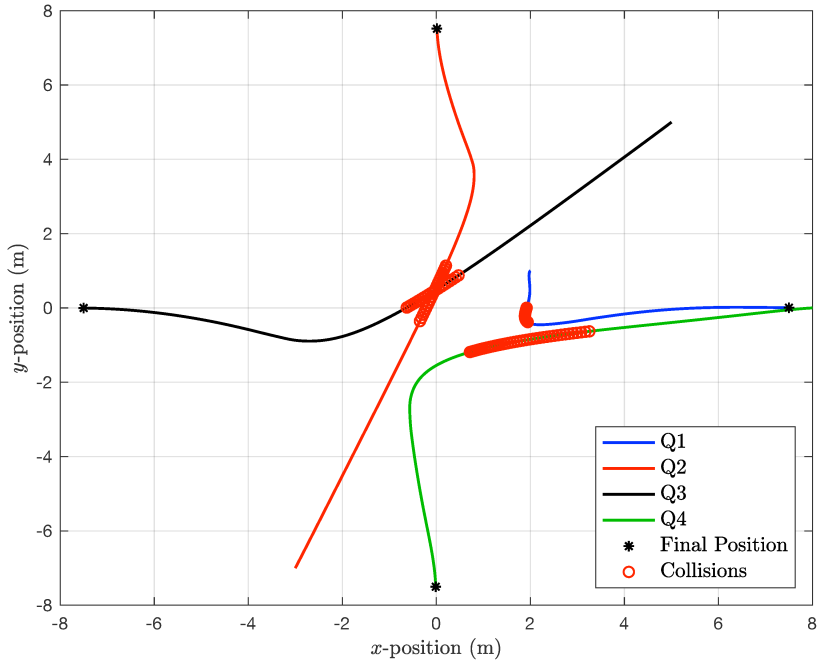

Figure 6.2: 2-dimensional view without collision avoidance

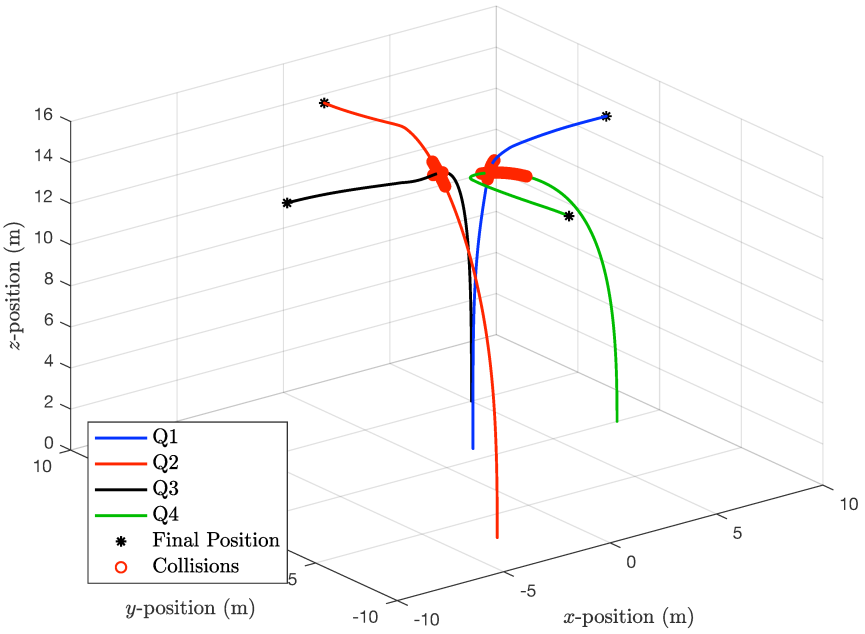

Figure 6.3: 3-dimensional view without collision avoidance

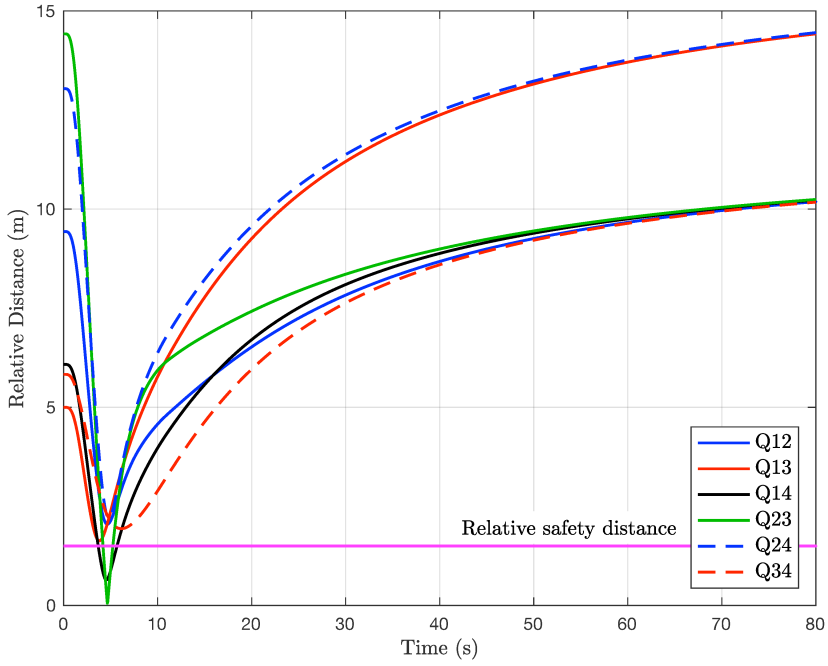

Figure 6.4: Relative distances without collision avoidance

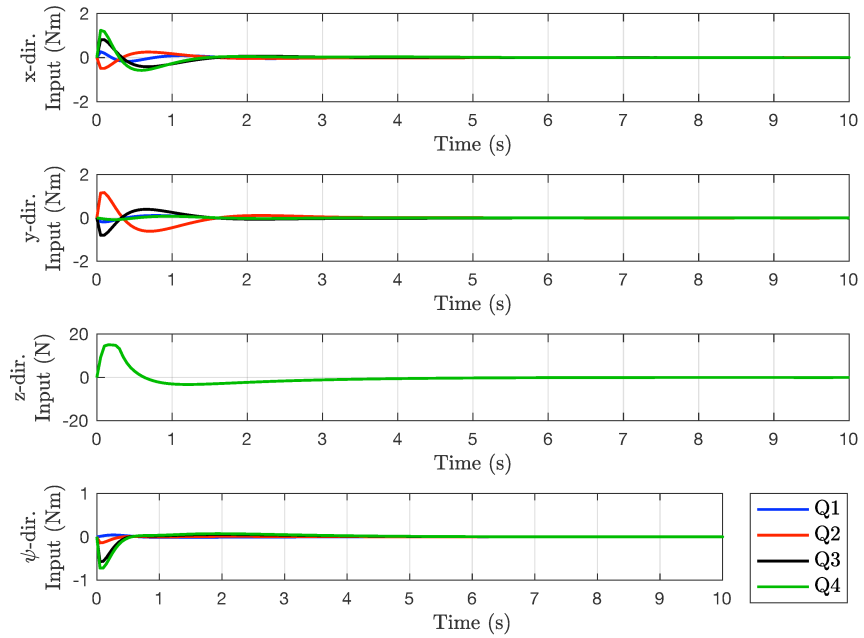

Figure 6.5: Control actions without collision avoidance

6.1 Case 1 - Collision avoidance with z-direction constraints

This section presents the collision avoidance results of Algorithm 1. The 3-dimensional view of the Case 1 collision avoidance results is found in Fig.6.6. It is observed that Quadcopter 3 and Quadcopter 4 both remain close to the ground in order to avoid collisions with Quadcopter 1 and Quadcopter 2. This is expected since the collision avoidance strategy applies constraints only in the z-direction. All quadcopters reach the desired height of 15 meters and the desired geometric formation.

The relative distances between all combinations of quadcopters can be seen in Fig.6.7. It can be observed from this plot that the quadcopters do not violate the relative safety distance of 1.5 meters. Notice that the large relative distance decrease that appears around 20 seconds, only occurs with quadcopter pairs Q13, Q14, Q23 and Q24. This is because Quadcopters 3 and 4 make a steep altitude gain, which brings them closer to Quadcopters 1 and 2

The control action results for Case 1 is presented in Fig.6.8. In the figure, it can be verified that the logarithmic barrier function was able to limit the control actions successfully, specifically in the z-direction. In addition, the control actions are quite smooth.

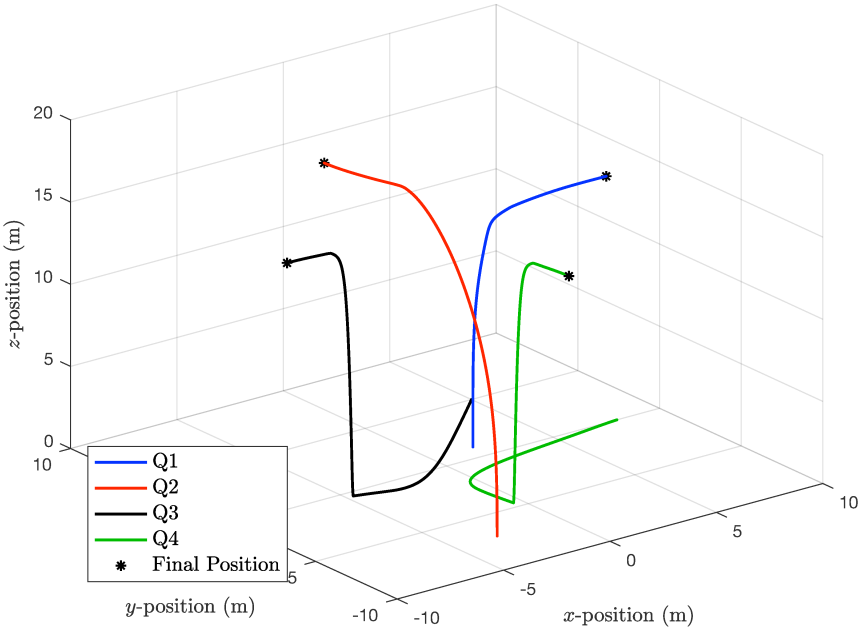

Figure 6.6: 3-dimensional view with collision avoidance (Case 1)

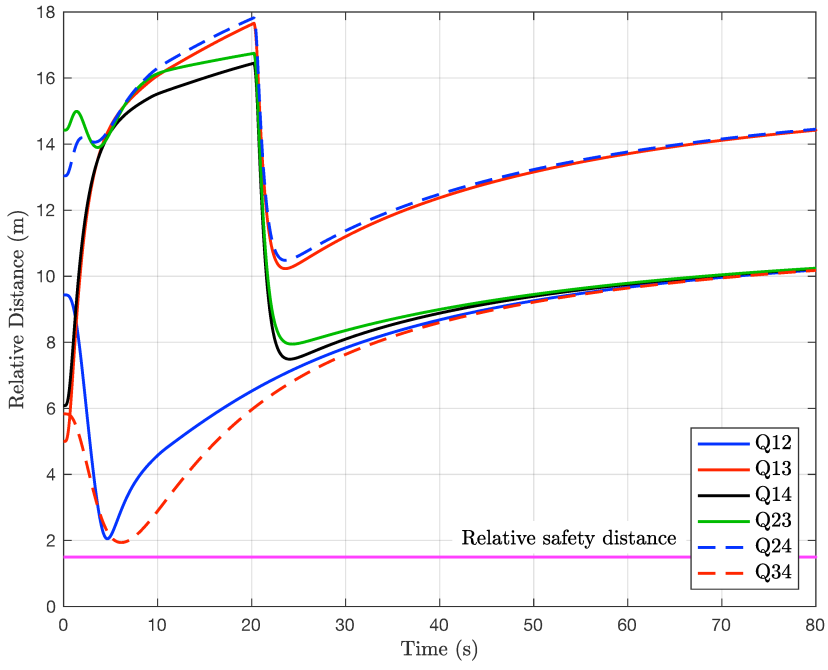

Figure 6.7: Relative distances with collision avoidance (Case 1)

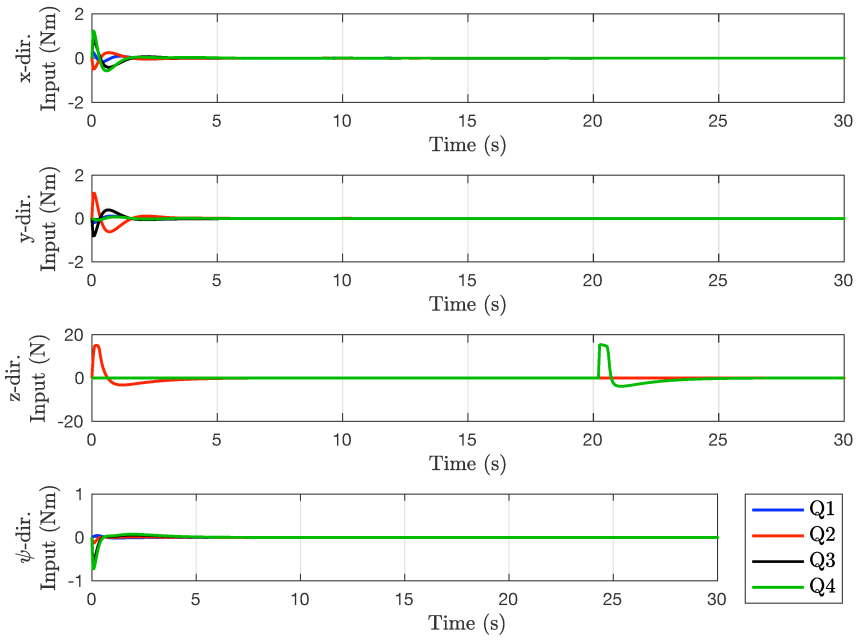

Figure 6.8: Control actions with collision avoidance (Case 1)

As presented in Fig.6.9, the team of quadcopters also achieved consensus in the ψ-direction and converged to a value of 0 radians. Among the quadcopters, the largest value for the pitch angle is 0.1401 radians and the largest value for the roll angle is 0.1806 radians. Both of these angles are relatively small and abide by the linearized quadcopter model condition. Consider that the percent error of small angle approximations exceed 1% at about 0.245 radians for sine, and 0.663 radians for cosine; refer to Appendix C.

6.2 Case 2 - Collision avoidance in one predetermined direction

This section presents the collision avoidance results of Algorithm 2. In this case, Quadcopters 1, 2, 3 and 4 have constraints applied in the x-direction, z-direction, y-direction and z-direction, respectively.

The 3-dimensional plot in Fig.6.10 illustrates the effects of the output constraints as motion in the z-direction dips down and up for Quadcopters 2 and 4.

The relative distance between all quadcopter combinations is presented in Fig.6.11. Quadcopters 1 and 3 reach the closest to the relative safety distance, at a distance of 1.6 meters but remain above the mark.

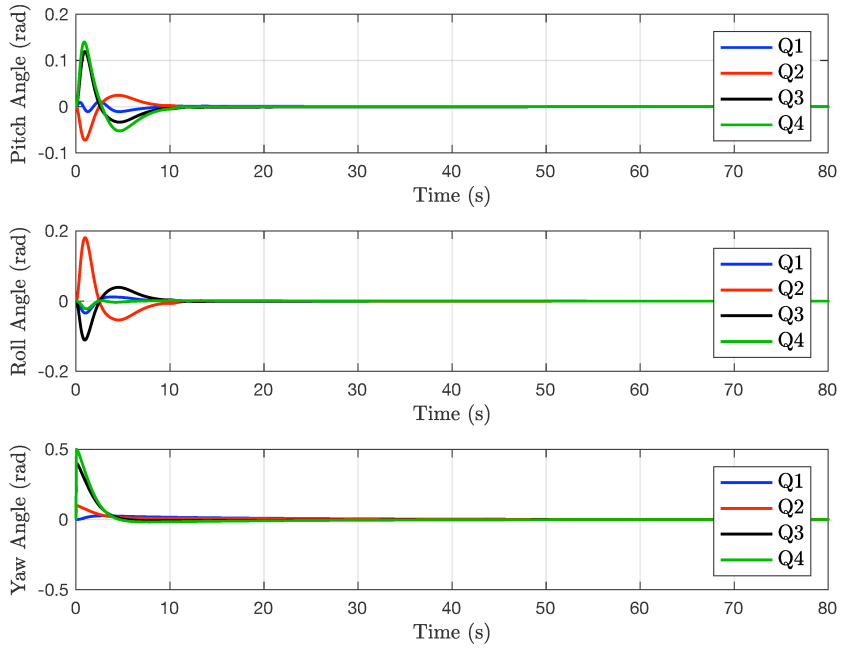

Figure 6.9: Roll, pitch, yaw angles with collision avoidance (Case 1)

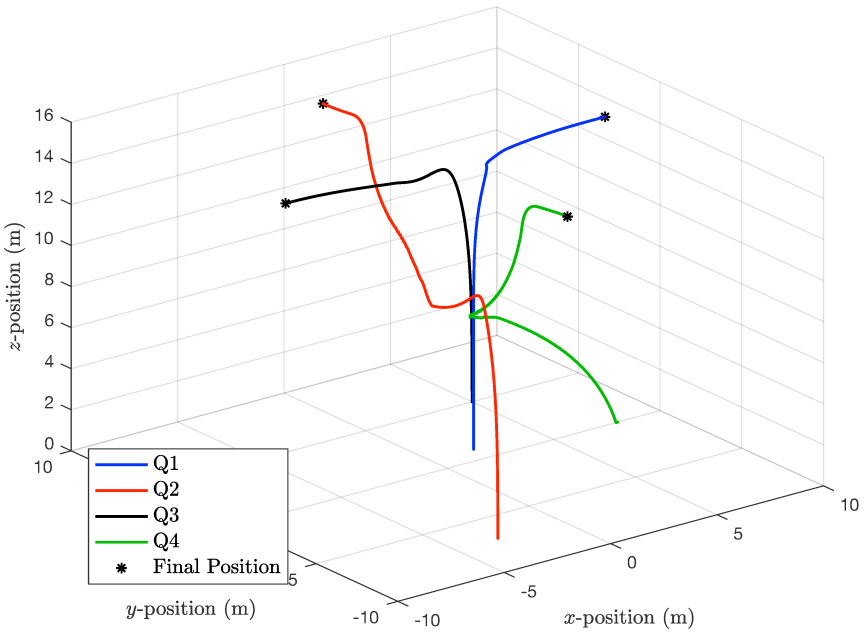

Figure 6.10: 3-dimensional view with collision avoidance (Case 2)

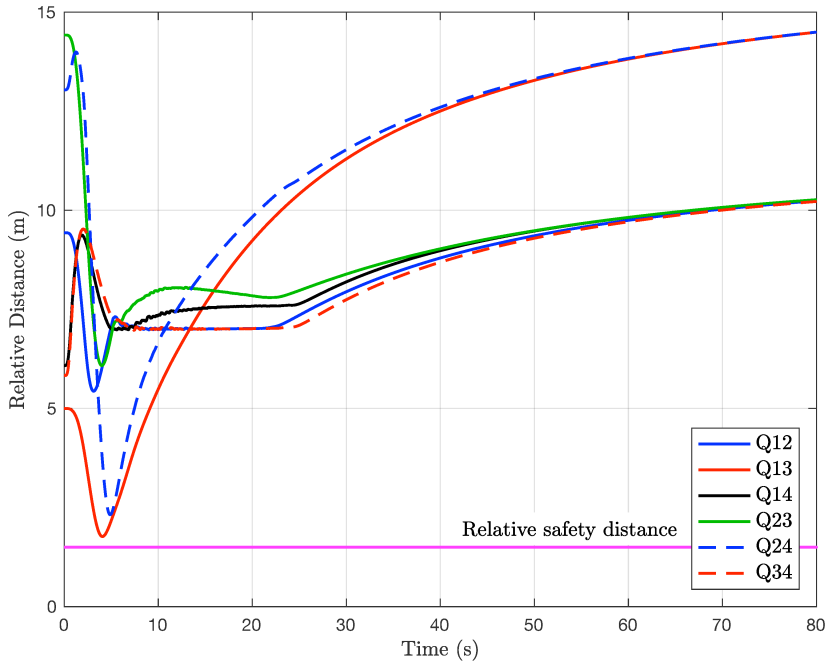

Figure 6.11: Relative distances with collision avoidance (Case 2)

Fig.6.12 shows that the control inputs were successfully limited, however, there is chatter in the z thrust for Quadcopter 4. The effects of the chatter on the z-positions of Quadcopter 2 and Quadcopter 4 is difficult to observe in Fig.6.10. The first 4.65 seconds of chatter (where the control does not dip too far past zero) may be attributed to the constraints turning on/off every third or fourth sampling instant. Slightly larger values of move suppression can reduce the chatter , however, it also affects the overall result of the obstacle avoidance.

In Case 2, the quadcopter team achieve consensus in the ψ-direction as shown in Fig.6.13. The largest value for the quadcopter pitch angle and roll angle is 0.1806 radians and 0.1794 radians, respectively. These relatively small angle values satisfy the linearized quadcopter model condition; refer to Appendix C.

In this control strategy, it is observed that 2 quadcopters must be assigned z-direction constraints, otherwise collision avoidance is not achieved. This is not unexpected, since constraints only limit movement in the direction the quadcopter is already headed according to setpoints. The relative distance between setpoints actually reveal that quadcopters will collide with each other. Since the desired z-position is reached quite quickly, the quadcopters are all on the same x-y plane; Therefore, x

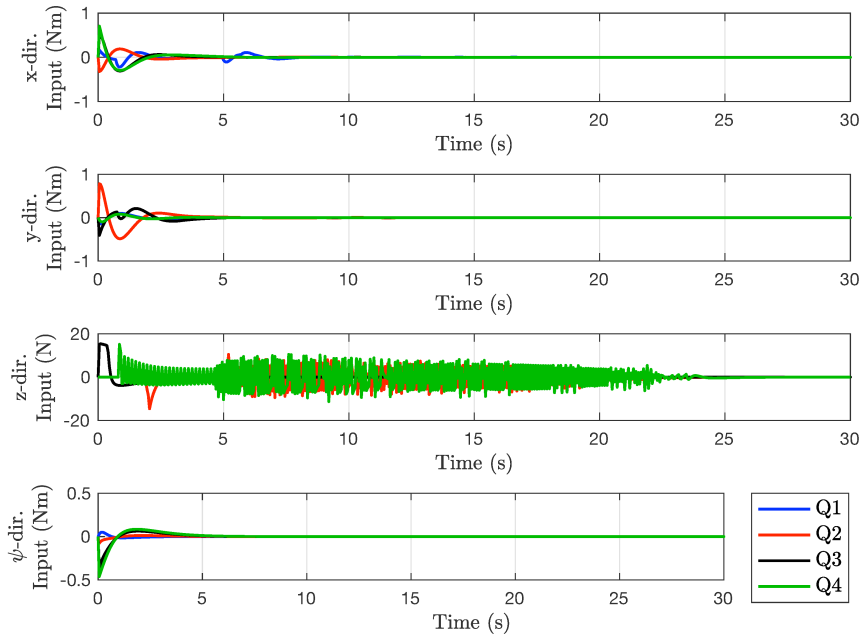

Figure 6.12: Control actions with collision avoidance (Case 2)

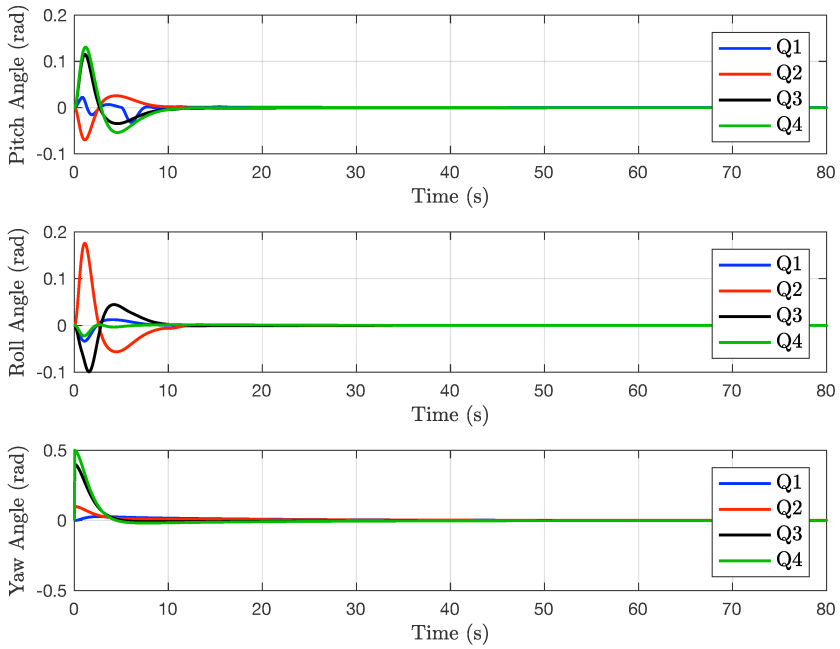

Figure 6.13: Roll, pitch, yaw angles with collision avoidance (Case 2)

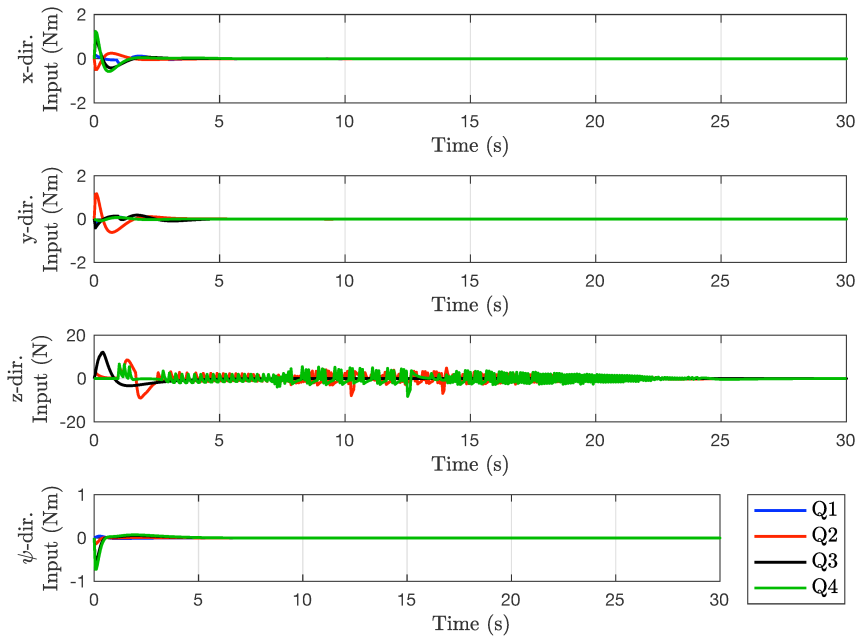

Figure 6.14: Control actions with collision avoidance (Case 2, $\lambda = 1.100$)

and y constraints are not enough for collision avoidance.

6.2.1 Effects of Tuning Parameters

Move Suppression

Recall that the move suppression is used to manipulate the aggressiveness of the closed loop response. As mentioned previously, the chatter that occurred in Fig.6.12, can be reduced by increasing the move suppression. By increasing the move suppression in the z-direction to $\lambda_z = 1.100$, it is observed in Fig.6.14 that the magnitude of the chatter is reduced. Similar to the original case, 2.85 seconds to 7.70 seconds (where the control does not dip too far past zero) may be attributed to the constraints turning on/off every fifth or sixth sampling instant.

However, the move suppression affects the control actions, which in turn affects the overall collision avoidance performance. For example, consider the 3-dimensional view in Fig.6.15. It is evident that Quadcopter 2 lowers less severely to avoid Quadcopter 3, compared to Fig.6.10 where $\lambda = 1.001$; less aggressive control actions results in less aggressive movements.

A comparison of relative distances is presented in Fig.6.16, where (a) is when

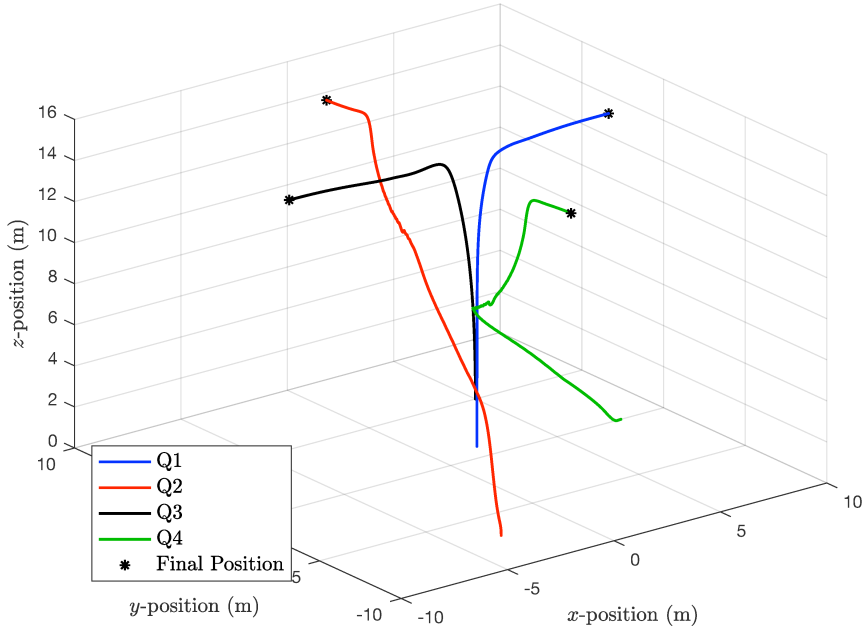

Figure 6.15: 3-dimensional view with collision avoidance (Case 2, $\lambda = 1.100$)

$\lambda_z = 1.100$ and (b) is the original Case 2 with $\lambda_z = 1.001$. It is observed when $\lambda_z = 1.100$, the relative distances of Q12, Q23, Q34, Q14 jump higher than when $\lambda_z = 1.001$ in the first 5 seconds. On the other hand, when $\lambda_z = 1.001$, Q24 jumps and remains higher in the first 5 seconds. The lowest relative distance was 1.873 meters and 1.769 meters for $\lambda_z = 1.100$ and $\lambda_z = 1.001$, respectively.

Setpoint Tuning Parameter

Recall that the setpoint tuning parameter α can be implemented to also reduce the aggressiveness of the control actions. As a comparison two simulations were done: a case where $\alpha = 0.9$ and a case where the parameter was removed ($\alpha = 0$). The control actions for the z-direction can be seen in Fig.6.17. Note that as an isolated study, λ was not increased to reduce the chatter.

However, the effects of the setpoint tuning parameter also affects the collision avoidance performance. For example, The 3-dimensional view of when $\alpha = 0$ and $\alpha = 0.9$ is shown in Fig.6.18 and Fig.6.19, respectively. When observing Quadcopter 2, it is apparent that $\alpha = 0$ results in more aggressive movements compared to $\alpha = 0.9$.

A comparison of relative distances is presented in Fig.6.20. It seems that in

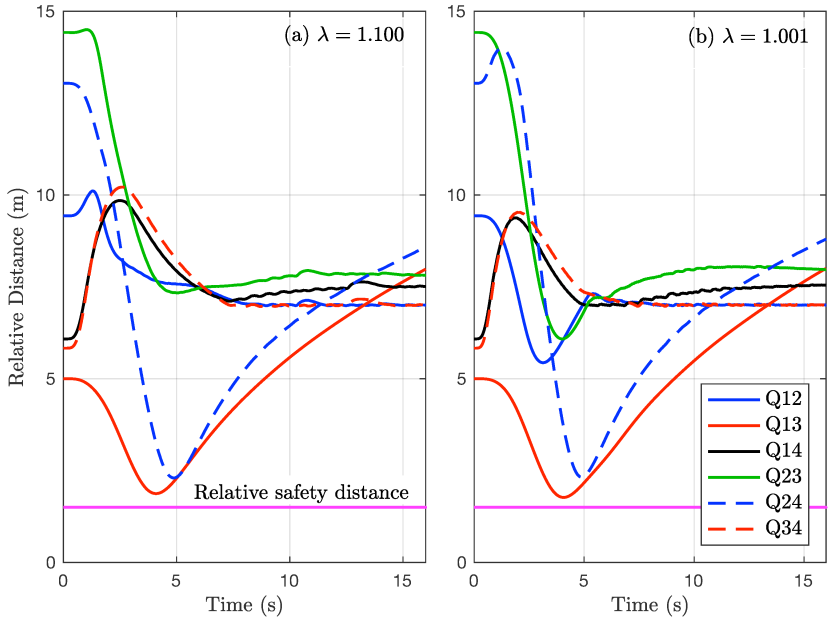

Figure 6.16: Relative distance with collision avoidance comparison (Case 2, $\lambda = 1.100$ and $\lambda = 1.001$)

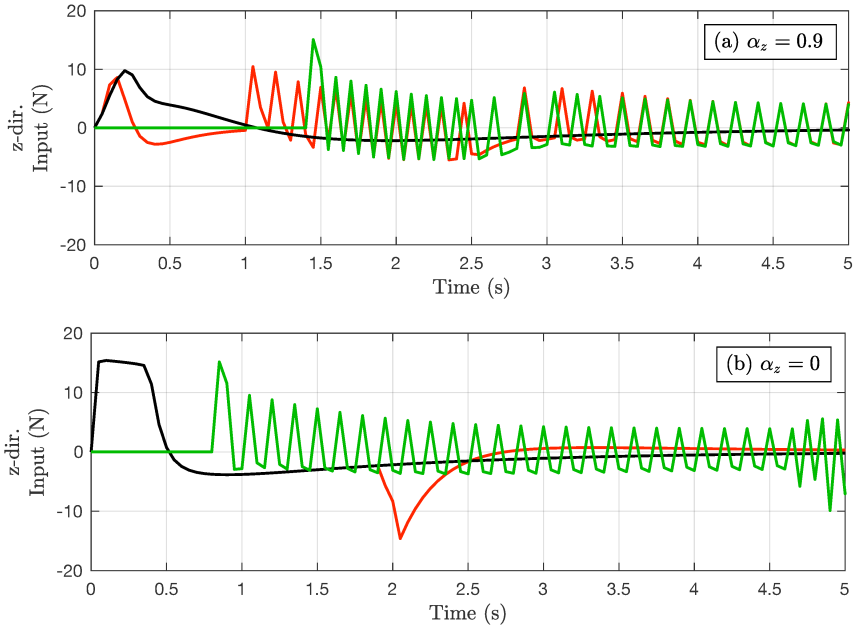

Figure 6.17: Control actions with collision avoidance (Case 2, $\alpha = 0.9$ and $\alpha = 0$)

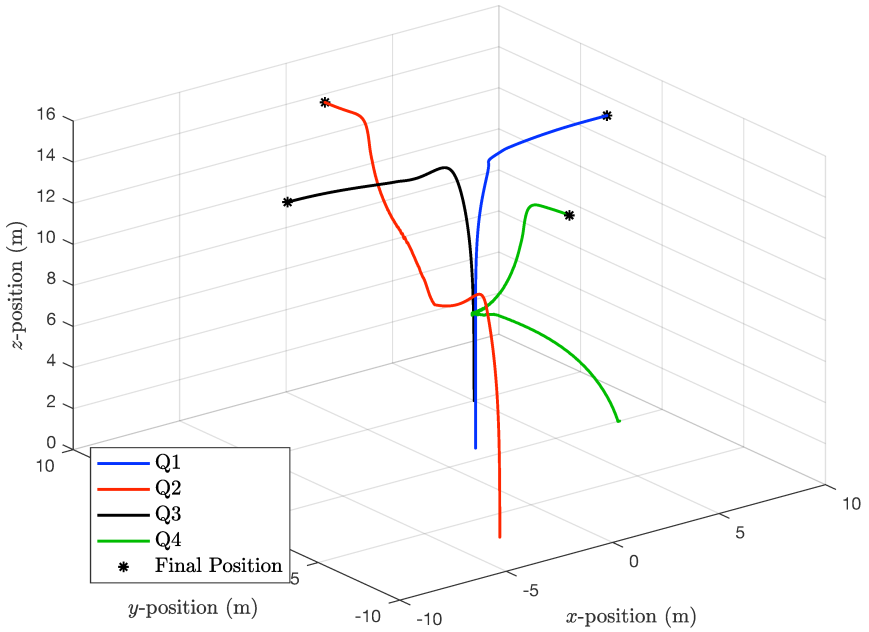

Figure 6.18: 3-dimensional view with collision avoidance (Case 2, $\alpha = 0$)

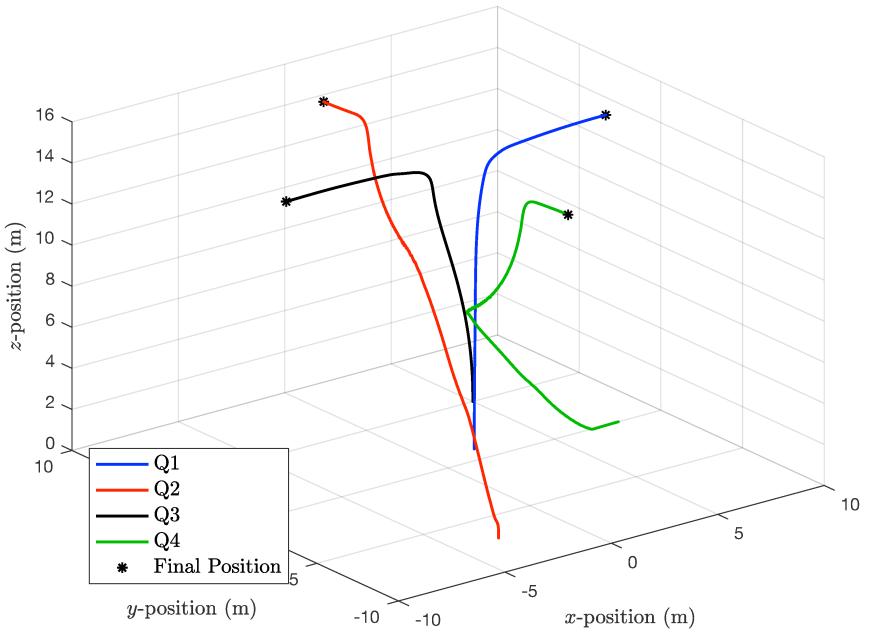

Figure 6.19: 3-dimensional view with collision avoidance (Case 2, $\alpha = 0.9$)

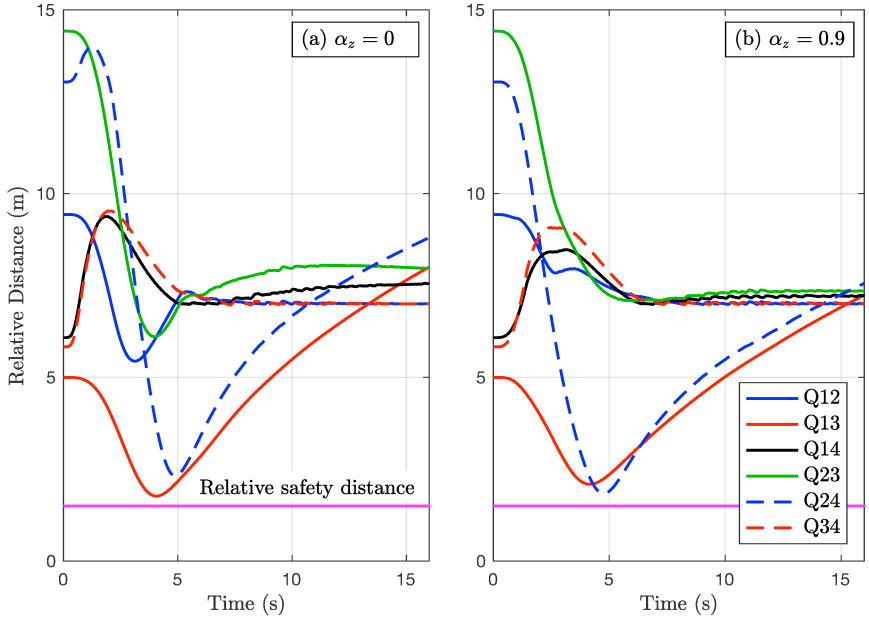

Figure 6.20: Relative distance with collision avoidance comparison (Case 2, $\alpha = 0$ and $\alpha = 0.9$)

Fig.6.20 (a), the more aggressive movements can result in positive and negative gains. For example, when $\alpha = 0$, Q14, Q24, and Q34 maintain a greater relative distance than when $\alpha = 0.9$. Conversely, when $\alpha = 0.9$, Q13, Q23, and Q12 maintain a greater relative distance than when $\alpha = 0$. This comparison could be considered inconclusive.

6.2.2 Effects of Number of Agents

The current form of the control strategy is unable to successfully achieve collision avoidance with a team of more than 4 quadcopters. Some trials of 5 quadcopter teams were successful with a strongly connected communication topology, however, it appeared to be circumstantial. Not all strongly connected topologies achieved collision avoidance, and no pattern or characteristic of the successful/unsuccessful topologies could be identified.

The successful and unsuccessful 5 agent trial had communication topologies according to Fig.6.21. The difference between the successful and unsuccessful trial is that Quadcopter 5 additionally receives information from Quadcopter 1.

The 3-dimensional view of topology (a) and (b) can be found in Fig.6.22 and

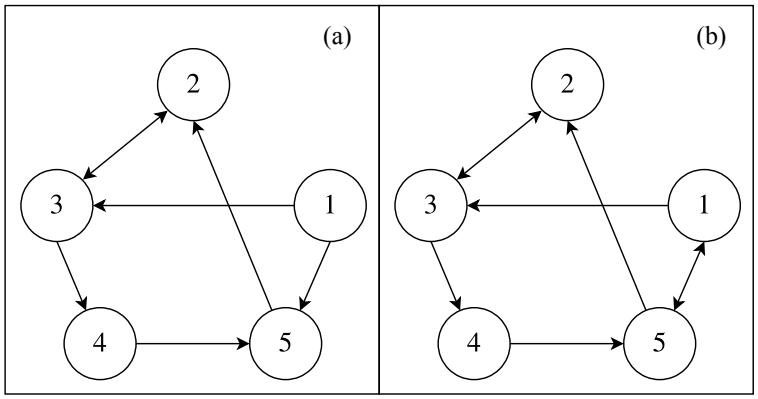

Figure 6.21: Communication topology (a) successful 5 agent topology (b) unsuccessful 5 agent topology

Fig.6.23, respectively. Despite the occurrence of collisions in topology (b), some avoidance can be seen.

The proposed control strategy achieves consensus by limiting the quadcopter's positions in the direction it was already headed in. The consensus algorithm with a ring formation (along with the specified initial positions) develops setpoints that bring quadcopters towards the center of the ring, before moving outwards. Therefore, some quadcopters are unaware as they approach or are being approached by non-neighbour quadcopters. An apparent answer to this problem is to include more communication links among the quadcopters, however, the argument of centralized versus distributed control resurfaces. A possible solution would be to have an underlying distributed communication topology used to develop desired setpoints and achieve consensus, and have quadcopters receive broadcasted positions as they are within range of non-network quadcopters.

6.2.3 Effects of Initial Conditions

In this section, the effects of different x, y initial conditions were observed. The relative distances between quadcopters is shown in Fig.6.24 as a comparison for the investigated variations. Variation (a) places the team closer together so their relative distances are smaller, while Variation (b) creates larger relative distances between the quadcopters. The original variation contains the initial conditions according to the simulations shown for Case 2, and has relative distances inclusive to Variation (a)

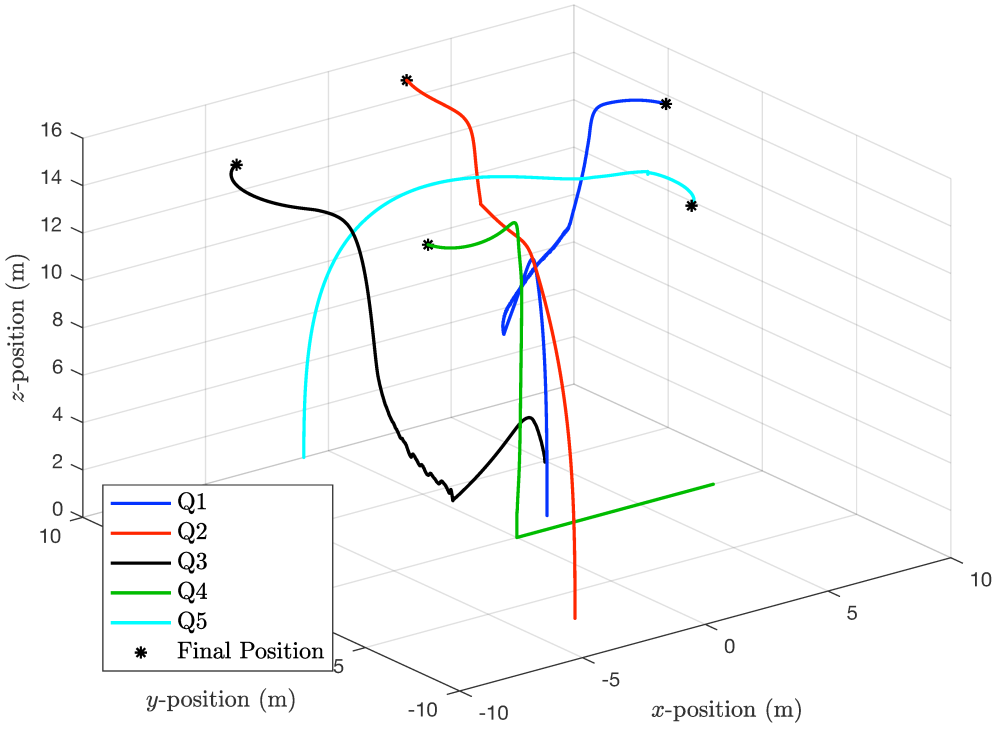

Figure 6.22: 3-dimensional view with collision avoidance (Case 2, topology (a))

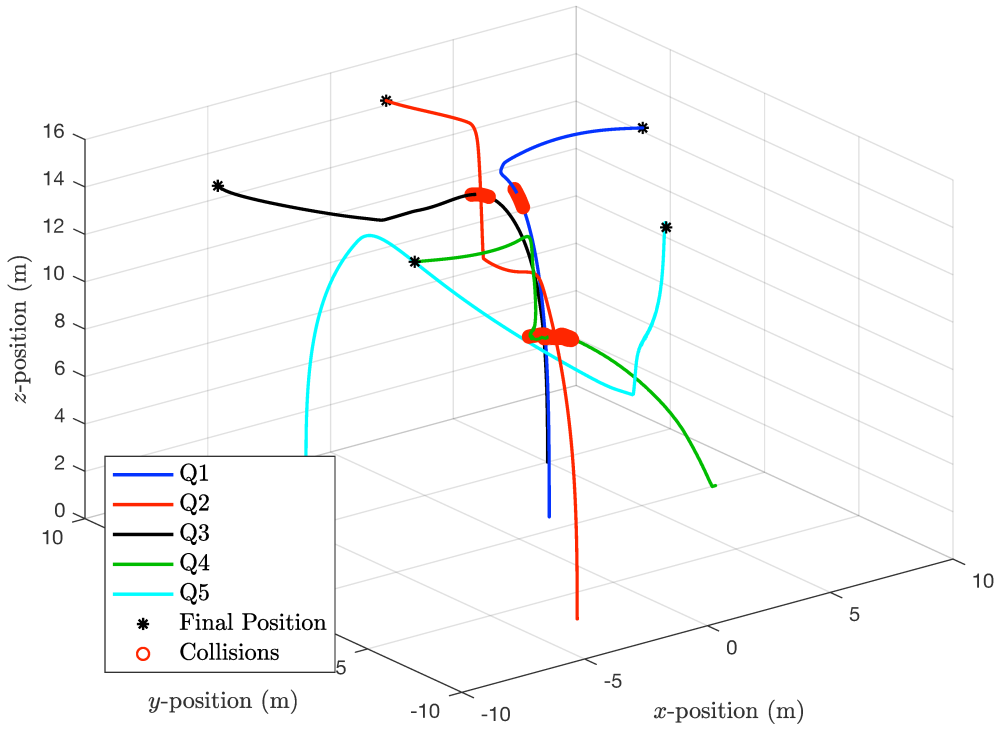

Figure 6.23: 3-dimensional view with collision avoidance (Case 2, topology (b))

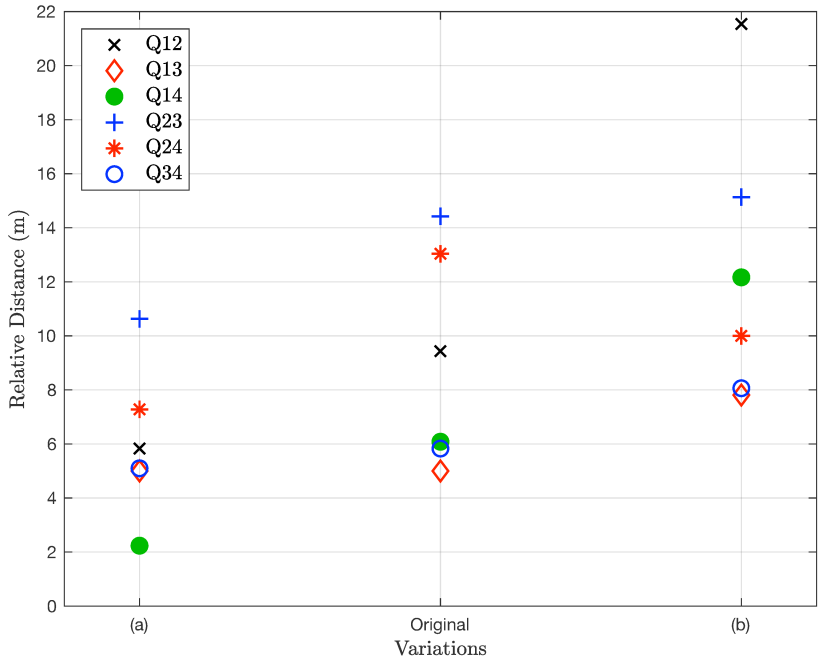

Figure 6.24: Relative distance comparison according to initial condition variations

and (b).

The 3-dimensional result of Variation (a) without and with collision avoidance is presented in Fig.6.25 and Fig.6.26, respectively. It is easy to observe that the quadcopters are closer.

The 3-dimensional result of Variation (b) without and with collision avoidance is presented in Fig.6.27 and Fig.6.28, respectively. Compared to Variation (a), the initial positions of the quadcopter team appear to be farther apart.

The variations in initial conditions show effects on the points of collision, and therefore, the collision avoidance path. In addition, the initial positions affect the setpoints developed by the consensus algorithm, which in turn influences the pairing and amount of colliding quadcopters. For example, the quadcopters that exceed the relative safety distance in Variation (a) are Q23 and Q24, while in Variation (b) they are Q23, Q14 and Q34.

6.2.4 Effects of Different Communication Topologies

A different communication topology has effects on the setpoints developed by the consensus protocol and the information used by quadcopters for collision avoidance.

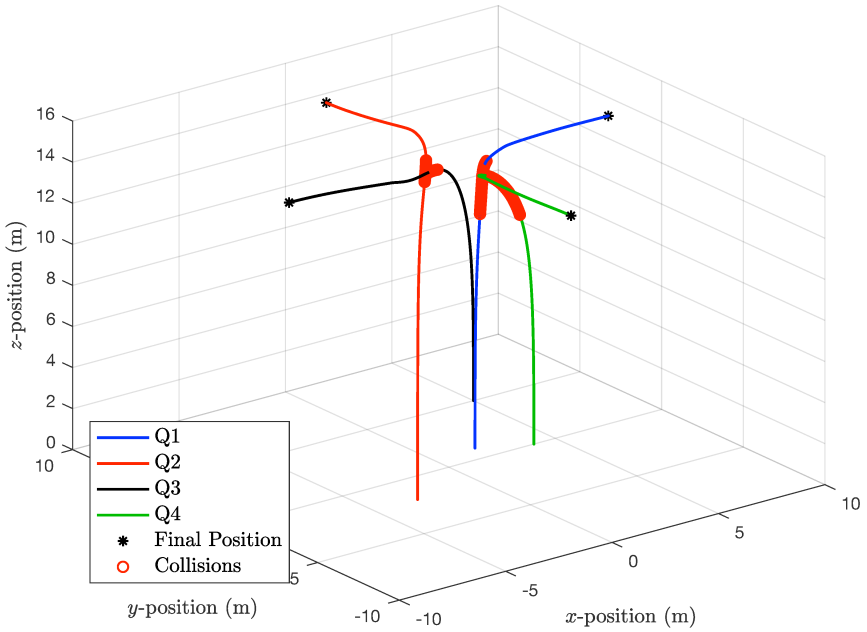

Figure 6.25: 3-dimensional view without collision avoidance (Variation (a), Case 2)

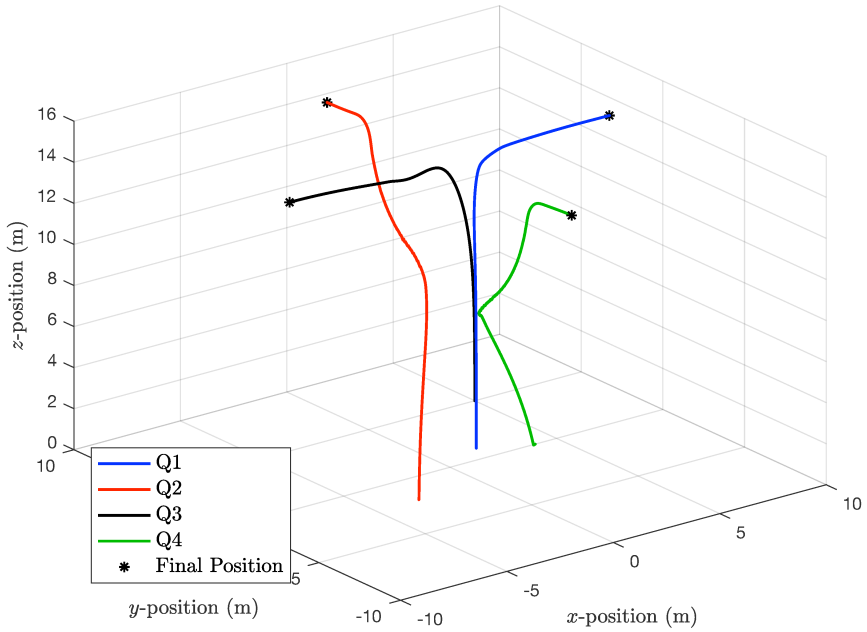

Figure 6.26: 3-dimensional view with collision avoidance (Variation (a), Case 2)

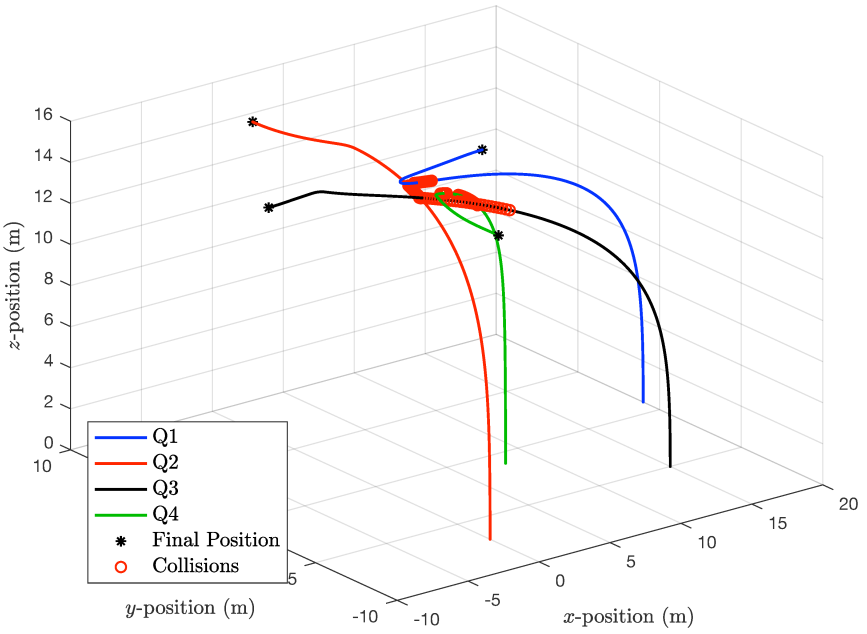

Figure 6.27: 3-dimensional view without collision avoidance (Variation (b), Case 2)

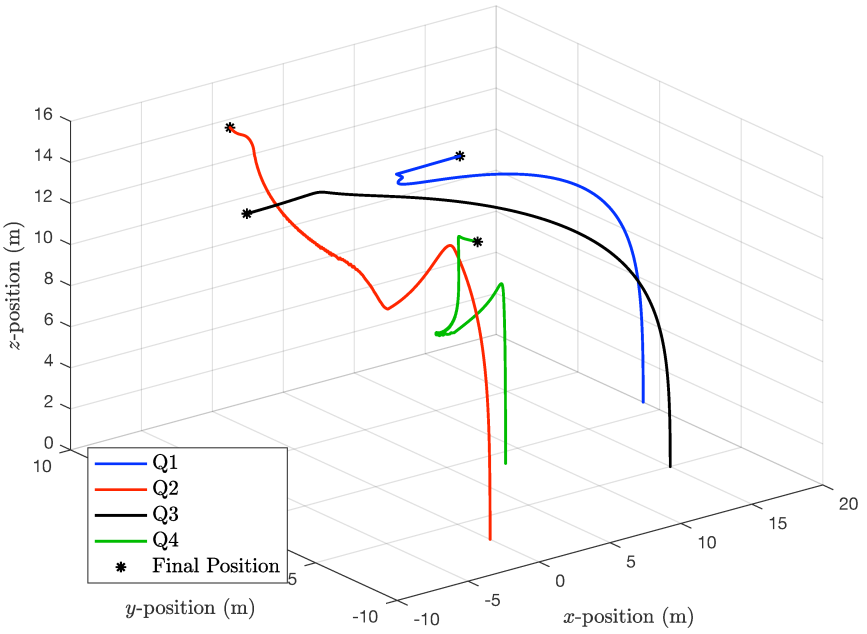

Figure 6.28: 3-dimensional view with collision avoidance (Variation (b), Case 2)

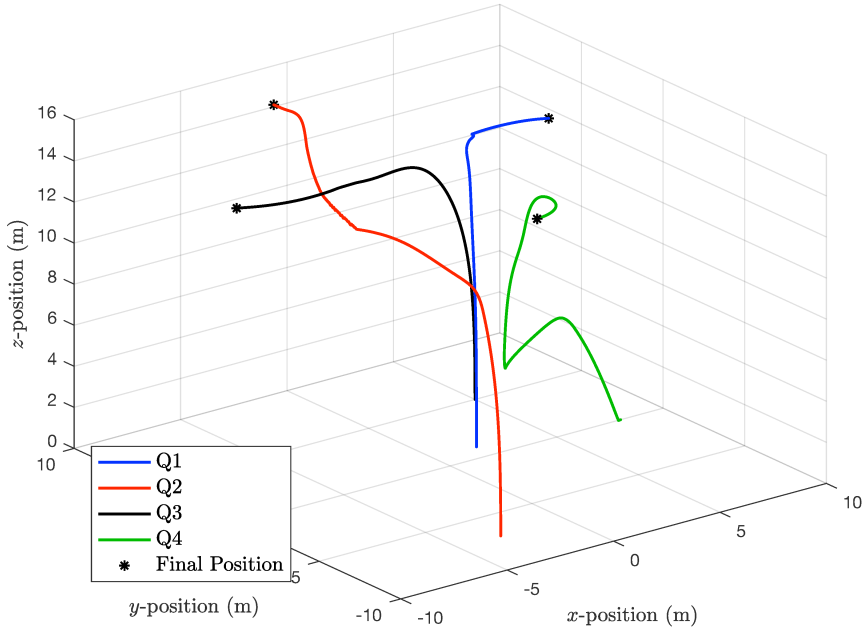

Figure 6.29: 3-dimensional view with collision avoidance (Case 2, strongly connected topology)

A strongly connected communication topology and a spanning tree topology was investigated, with respective adjacency matrices

$$\mathcal{A} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}, \quad \text{and} \quad \mathcal{A} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

Both topology cases successfully achieved consensus and reached the desired altitude; Fig.6.29 presents the 3-dimensional view of the strongly connected topology case.

A performance contrast occurred with the spanning tree topology; as expected, all quadcopters except the root moved in order to accomplish the desired formation; this can be seen in Fig.6.30. Minimal movement from the root (Q1) increased the consensus time to 173.15 seconds from 87 seconds in the strongly connected topology case.

A comparison of relative distances for the topologies investigated is shown in Fig.6.31. Consider Fig.6.31 (b), in this topology Quadcopter 4 only receives from

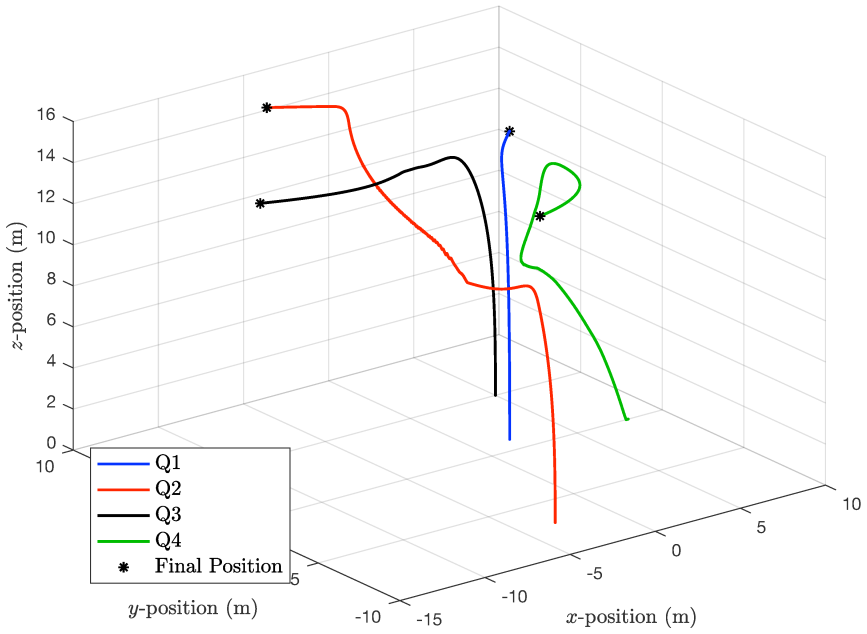

Figure 6.30: 3-dimensional view with collision avoidance (Case 2, spanning tree topology)

Quadcopter 3 and does not send any information. This could be a reason why the relative distance between Quadcopters 1 and 4 dip in Fig.6.31 (b), compared to (a) and (c).

Other than consensus time, which will be addressed in Section 6.4, no distinctions in performance were apparent between the original undirected topology, the spanning tree topology and the strongly connected topology. It is difficult to make conclusions based on the communication topology of the MAS because the topology affects the setpoints, which affects the collision avoidance movements. This suggests that each topology case is unique.

6.2.5 3-Dimensional Formation

A case was considered where the quadcopter formation did not occur only on the x-y-plane, but also included a z-plane formation. The quadcopters were still required to reach the desired ring formation and in addition, have a relative altitude of 2 meters with their (increasing numbered) neighbour. Without and with the collision avoidance strategy, the quadcopter team did not have any collisions. The 3-dimensional

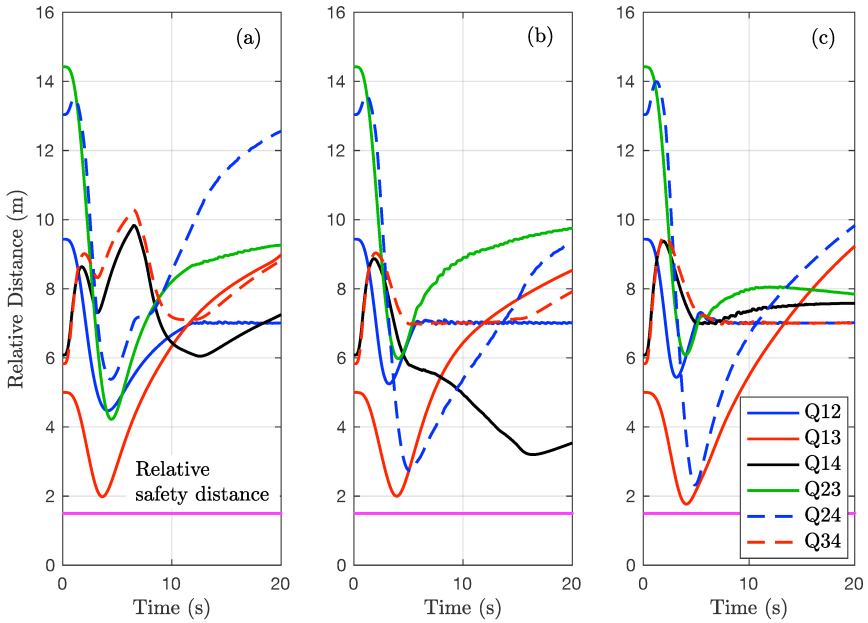

Figure 6.31: Relative distance comparison with collision avoidance (Case 2, (a) strongly connected topology (b) spanning tree topology (c) original undirected topology)

view without and with collision avoidance is presented in Fig.6.32 and Fig.6.33, respectively. It is observed, that the collision avoidance constraints were still applied, even though collisions did not occur. This is because the avoidance distance (the distance where the constraints are turned on/off) still triggered the constraints.

In Fig.6.34, it is shown that the relative distances do not violate the relative safety distance. However, the event with collision avoidance maintains larger relative distances between all quadcopters, than without collision avoidance.

6.3 Case 3 - Collision avoidance with x, y and z constraints

The Case 3 control strategy outlined in Algorithm 3, defines quadcopters to have either positive or negative evasive z-movements (in addition to x, y constraints). In this simulation study even numbered quadcopters have negative evasive actions and odd numbered quadcopters have positive evasive actions.

The 3-dimensional view of the collision avoidance according to Case 3, is shown in Fig.6.35. It is observed that the collision avoidance is successful, however, prominent oscillations in the z-direction occur.

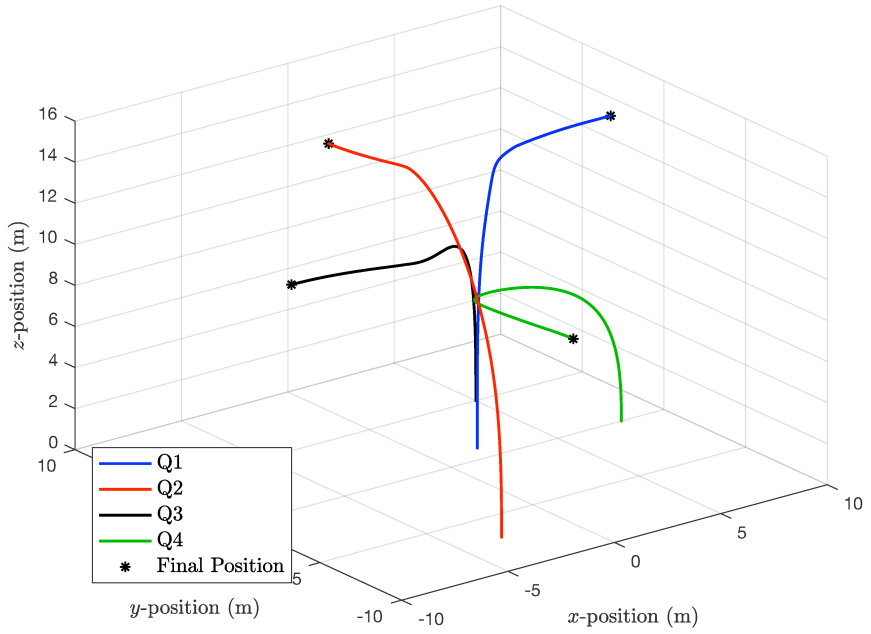

Figure 6.32: 3-dimensional view without collision avoidance (Case 2, 3-dimensional formation

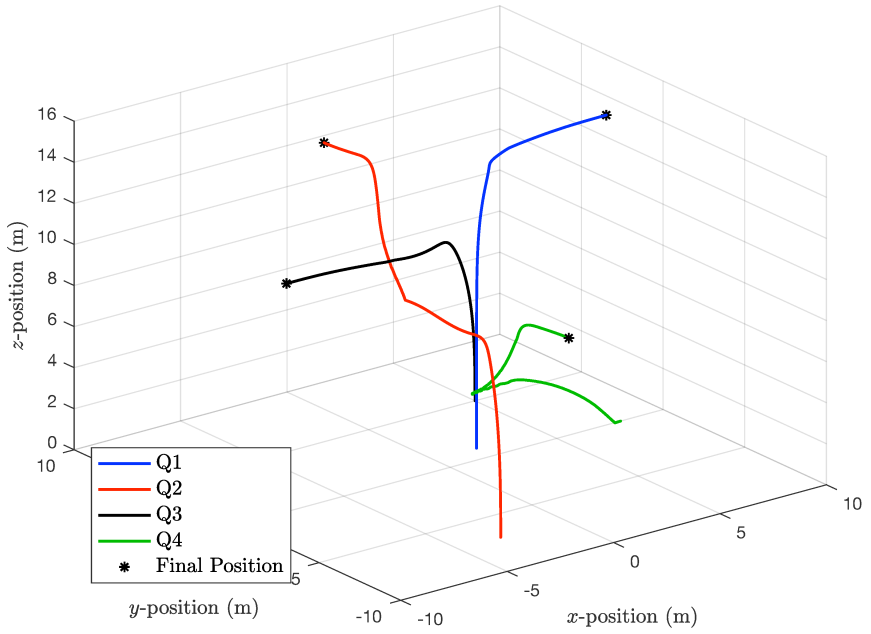

Figure 6.33: 3-dimensional view with collision avoidance (Case 2, 3-dimensional formation

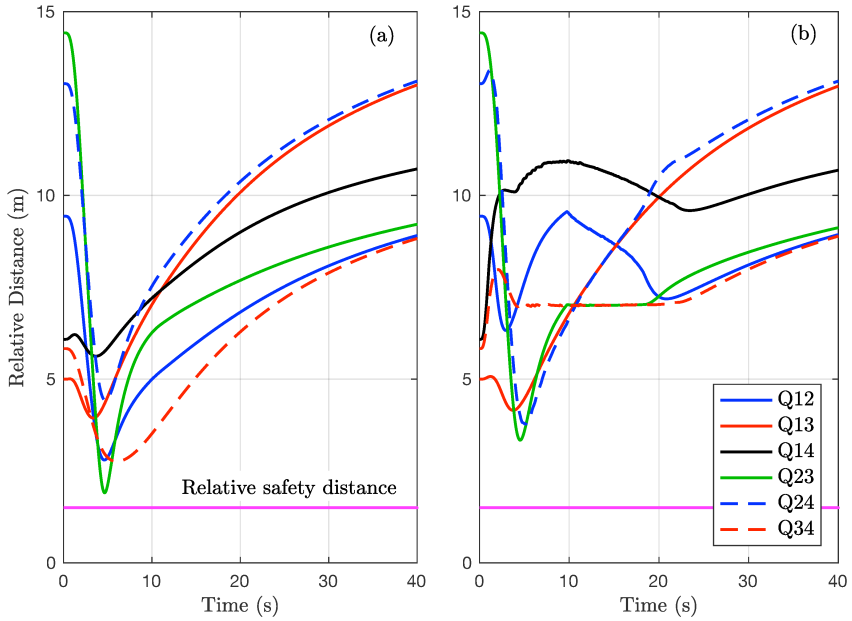

Figure 6.34: Relative distance comparison (Case 2, (a) without collision avoidance (b) with collision avoidance

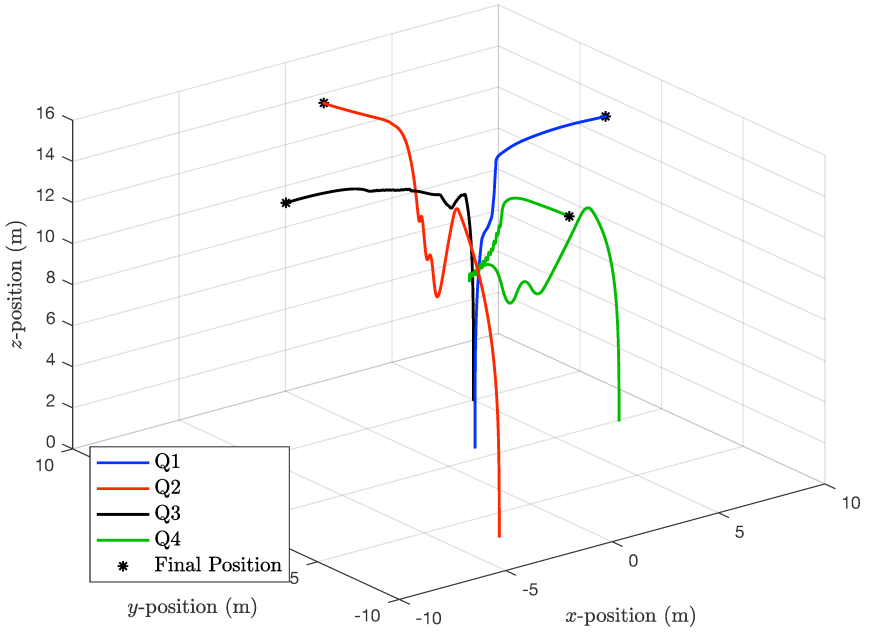

Figure 6.35: 3-dimensional view with collision avoidance (Case 3)

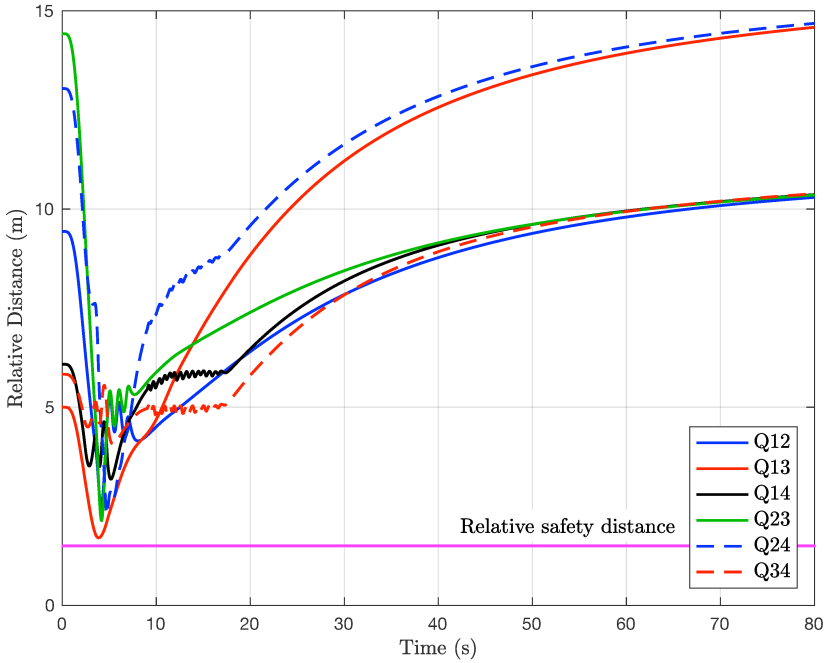

Figure 6.36: Relative distances with collision avoidance (Case 3)

The relative distances between all quadcopters can be seen in Fig.6.36. The relative distances remain above the relative safety distance. The effect of the oscillations is also apparent between 0 seconds and 20 seconds.

The control actions for the collision avoidance case is found in Fig.6.37. The oscillatory control actions for the quadcopter thrust verify that the position oscillations occur only in the z-direction. It is observed that the even numbered quadcopters (Q2 and Q4) have larger oscillations. This is because their evasive movements are subtracted from their current z-position; therefore, they are already in violation when these constraints are applied. In addition, the control actions were successfully limited using the logarithmic barrier function.

In Fig.6.38, it can be seen that the quadcopters reach consensus in the ψ-direction and converge to a value of 0 radians. The largest value among the quadcopters is 0.1309 radians for the pitch angle and 0.1755 radians for the roll angle. Both of these angles are within range (according to Appendix C) to satisfy the small angle condition of the linearized quadcopter model.

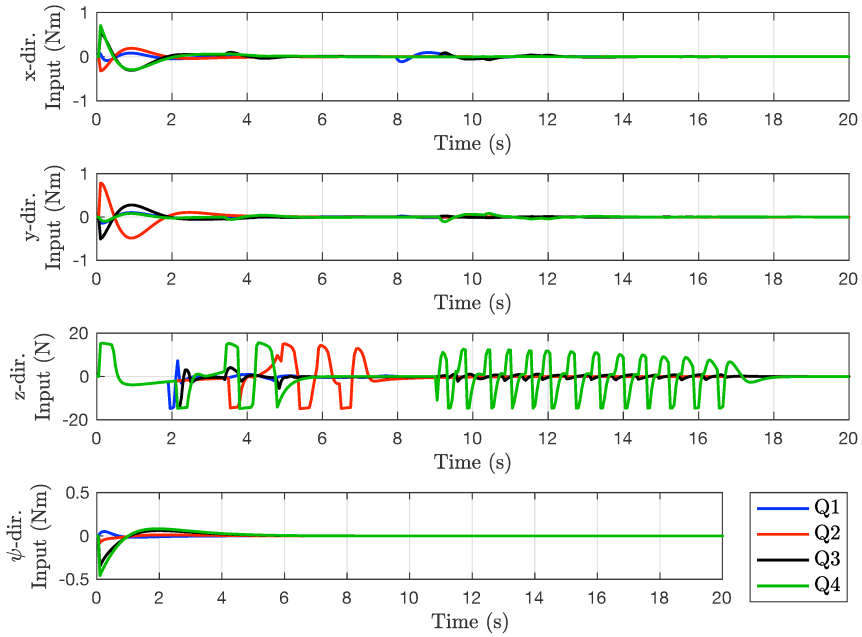

Figure 6.37: Control actions with collision avoidance (Case 3)

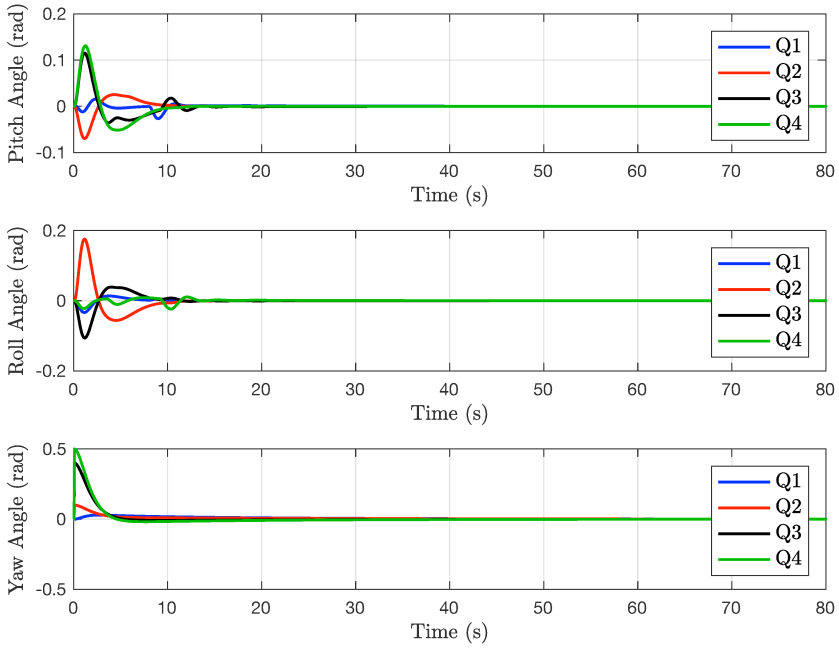

Figure 6.38: Roll, pitch, yaw angles with collision avoidance (Case 3)

Table 6.2: Simulation Times

Case & Variation	Reference	Consensus Time (s)	Computation Time (s)			
			x-dir.	y-dir.	z-dir.	ψ-dir.
1	Section 6.1	110.95	0.0224	0.0229	0.0120	0.0151
2, Original	Section 6.2	107.95	0.0222	0.0232	0.0129	0.0152
2, $\lambda = 1.100$	Section 6.2.1	109.90	0.0230	0.0231	0.0166	0.0155
2, $\alpha = 0$	Section 6.2.1	107.55	0.0228	0.0232	0.0134	0.0156
2, $\alpha = 0.9$	Section 6.2.1	119.85	0.0230	0.0235	0.0133	0.0155
2, 5 agents†	Section 6.2.2	161.50	0.0211	0.0209	0.0128	0.0163
2, 5 agents††	Section 6.2.2	182.20	0.0234	0.0219	0.0122	0.0163
2, Far	Section 6.2.3	124.15	0.0220	0.0226	0.0134	0.0152
2, Close	Section 6.2.3	109.45	0.0225	0.0232	0.0129	0.0152
2, SC‡	Section 6.2.4	87.00	0.0217	0.0222	0.0134	0.0159
2, Spanning tree	Section 6.2.4	173.15	0.0209	0.0213	0.0118	0.0135
2, 3D formation	Section 6.2.5	110.95	0.0222	0.0213	0.0129	0.0152
3	Section 6.3	103.25	0.1084	0.1078	0.1835	0.1020

†successful, ††unsuccessful, ‡strongly connected

6.4 Case Comparison

The time to reach consensus and the computation time of the optimization functions are presented in Table 6.2, and compared for the collision avoidance case studies. Consensus time refers to the time is took for the quadcopters to reach the desired geometric formation in the x-y plane and reached the desired altitude in the z-direction. This was calculated using the number of simulation iterations and the sampling time. Computation time refers to the time it took for the processor (the computer running the simulation) to 1) formulate the optimization function, 2) formulate the constraints, and 3) solve the optimization function. The computation times presented in Table 6.2 are the averaged values among the quadcopters according to the optimization function in each direction. The computation time was measured using Matlab functions: "tic" (records the internal time at execution) and "toc" (determines elapsed time).

In Case 2, it is observed that the when the setpoint tuning parameter increased ($\alpha = 0$ to $\alpha = 0.9$) the consensus time increased. This is because the first order approach to the desired setpoint is less aggressive, which means that the setpoints

across the prediction horizon are more gradual. As a result, the larger α value reduced the control aggressiveness and therefore, the system of quadcopters took longer to achieve consensus.

As expected, the team of quadcopters took longer to reach consensus in Case 2 with far initial conditions versus Case 2 with close initial conditions. This result is attributed to the additional distances covered by the quadcopters.

In Case 2, the strongly connected topology outperformed the spanning tree topology with respect to consensus time. This is because with the spanning tree topology, the root hardly moved forcing the other quadcopters to achieve consensus without its contribution in covering distance. In addition, the strongly connected topology reached consensus quicker than the original undirected topology. As the number of edges increases, the connectivity increases, which allows agreement to be reached quicker [27].

While the consensus time in Case 3 is similar to the other case studies, the computation times are quite large; especially in the z-direction. The large computation times in the x and y-directions, indicate that solving the optimization function numerically to constrain the quadcopter to its current position, may not be practical. Simply applying control that guarantees hovering behaviour may be more reasonable. It is difficult to draw conclusions for the high computation time in the ψ-direction, since no output constraints are applied. All case study computation times, except for Case 3, indicate that a sampling time of 0.05 seconds is reasonable.

Chapter 7

Simulation Studies on Obstacle Avoidance

7.1 Case 4 - Obstacle avoidance with x, y and z constraints

This chapter presents the simulation results for the proposed control strategy outlined in Algorithm 4. This is an isolated study on obstacle avoidance, where collision avoidance is not considered. Simulations are designed such that quadcopters will collide with obstacles and not each other. Note, in order to better view simulation details some plots only display a portion of the full simulation time.

The simulation consists of a leaderless 3 agent team of quadcopters at rest and spatially distributed on the ground. The quadcopters are required to form a ring with a relative distance of 7.5 meters to the ring's center at a height of 15 meters. The initial quadcopter positions (x, y, z, ψ) are Q1 $= (2, 1, 0, 0)$, Q2 $= (-3, -7, 0, 0.1)$, and Q3 $= (5, 0, 0, 0.4)$; all remaining states are zero. There are two spherical obstacles ($n_o = 2$) with center positions and radius (x, y, z, radius) as O1 $= (2, -2, 8, 2)$ and O2 $= (-1, 0, 15, 3)$. In this study, the team of quadcopters have position information of all obstacles.

The radius of the safety region was chosen to be $r_s = 0.75$ meters; this is based on a quadcopter with arm lengths of 0.3 meters. Therefore, the relative safety distance between quadcopters is 1.5 meters. The avoidance distance is 3 meters and the control limits based on motor thrust values [51], are ± 12 Newton-meters and ± 20 Newtons. The parameters are selected as presented in Table 6.1. Simulations were performed in Matlab 2017b and the optimization functions were solved using "fmincon" solver.

The 2-dimensional and 3-dimensional views of the results without the obstacle avoidance strategy are shown in Fig.7.1 and Fig.7.2, respectively. It is observed that the quadcopters can reach consensus and achieve the desired formation at the requested height of 15 meters. Note that the green circles represent points where the relative distance between a quadcopter and an obstacle was below the relative safety distance or inside the obstacle region, and the black star is the final position.

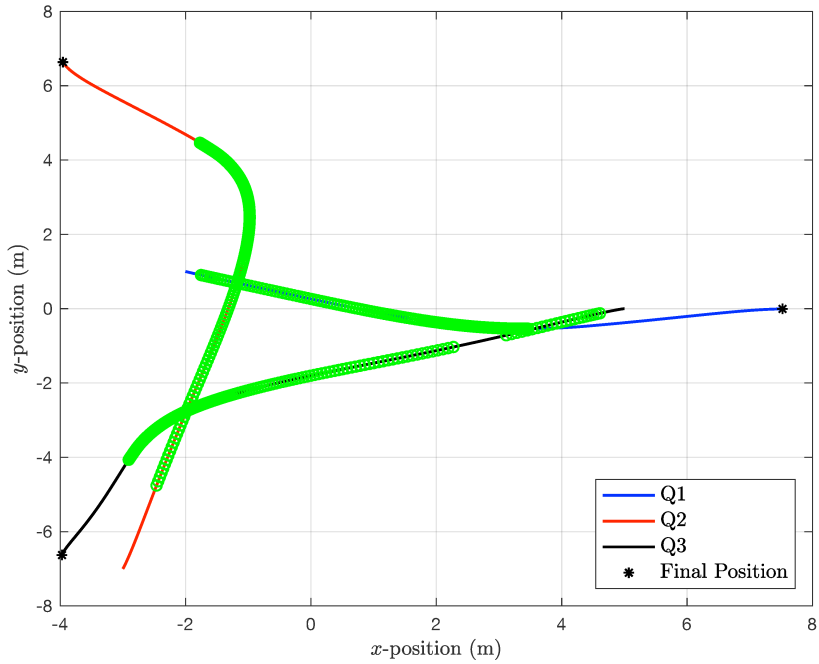

Figure 7.1: 2-dimensional view without obstacle avoidance (Case 4)

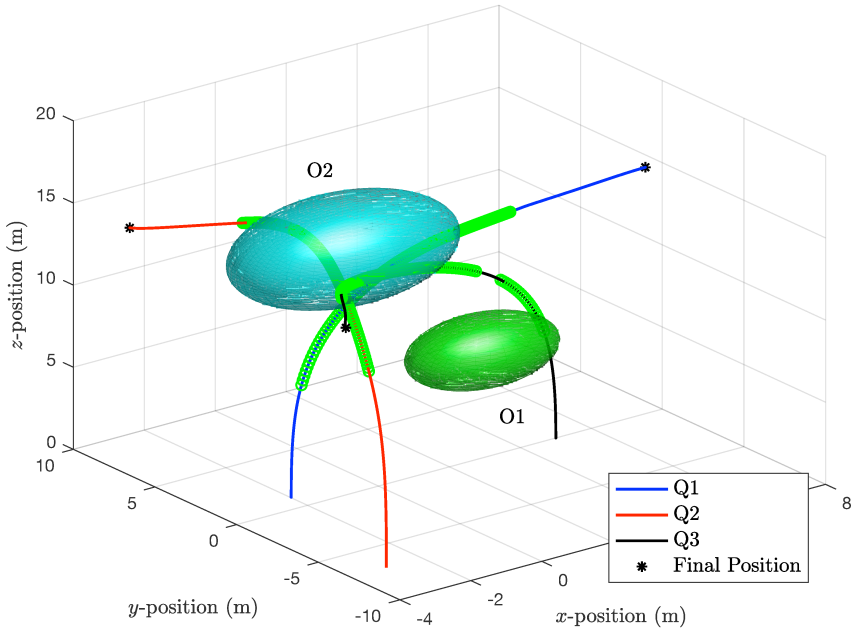

Figure 7.2: 3-dimensional view without obstacle avoidance (Case 4)

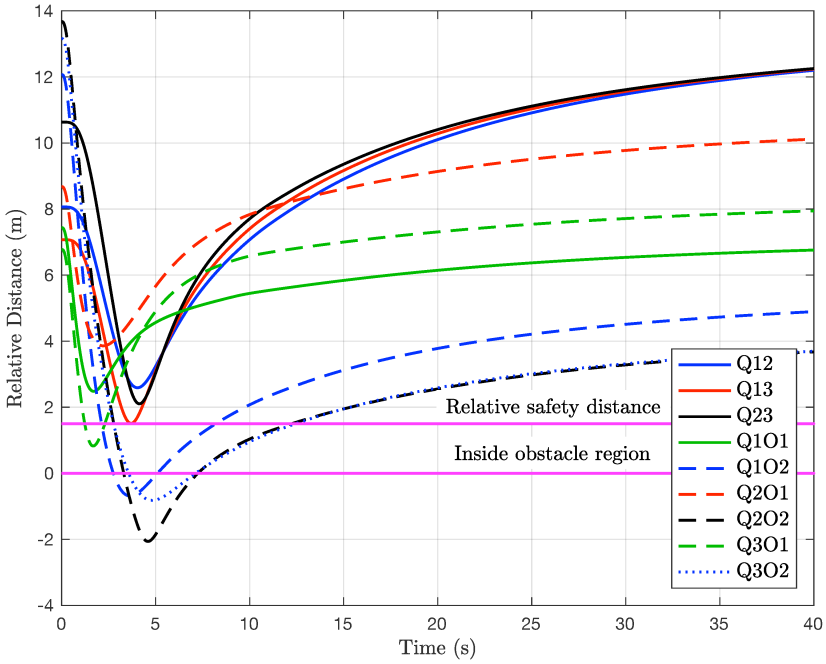

Figure 7.3: Relative distances without obstacle avoidance (Case 4)

The relative distances between all combinations of quadcopters and obstacles are presented in Fig.7.3, where QiOj for $i = 1, ..., n_a$ and $j = 1, ..., n_o$ in figure legends refers to quadcopter i and obstacle j. Between 1 and 13 seconds, all three quadcopters violate the relative safety distance with one or both obstacles. Quadcopters 3 violates the relative safety distance with Obstacle 1, while Quadcopters 1, 2 and 3 go beyond and enter the regions occupied by the obstacles.

Although collisions occur with obstacles, the input was successfully limited using the logarithmic barrier function as shown in Fig.7.4.

With obstacle avoidance, the 3 quadcopters successfully achieve consensus and the desired height, which can be seen in Fig.7.5.

The plot presented in Fig.7.6, shows that the relative distance for all combinations of quadcopters and obstacles do not go below the relative safety distance of 1.5 meters. The closest distance is 1.605 meters and occurs during the simulation at 3.7 seconds. However, very small oscillations can be seen in Fig.7.6; for example Q13 between 10 and 20 seconds.

The cause of the oscillations that were visible in Fig.7.6, is observed in Fig.7.7. In this plot it is apparent that the oscillations are caused by the large chatter from

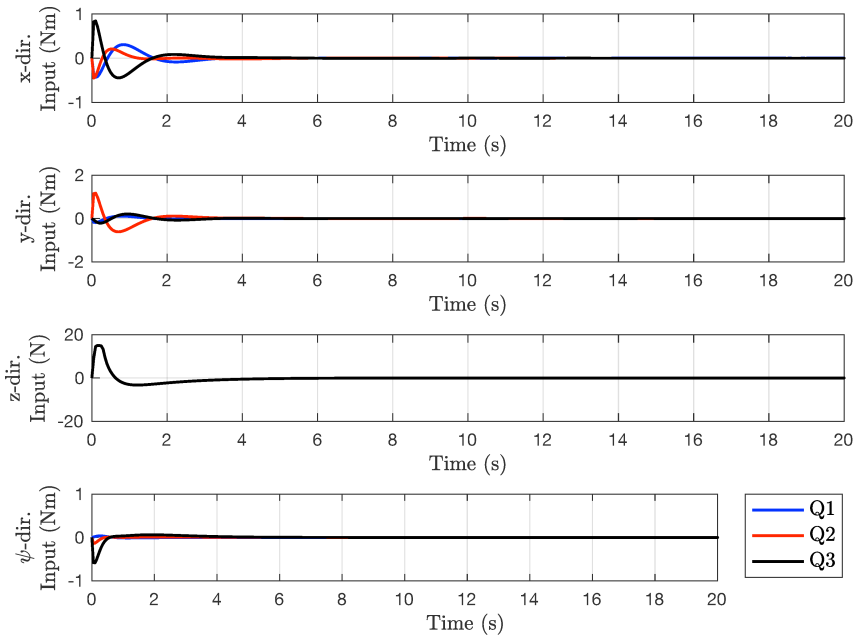

Figure 7.4: Control actions without obstacle avoidance (Case 4)

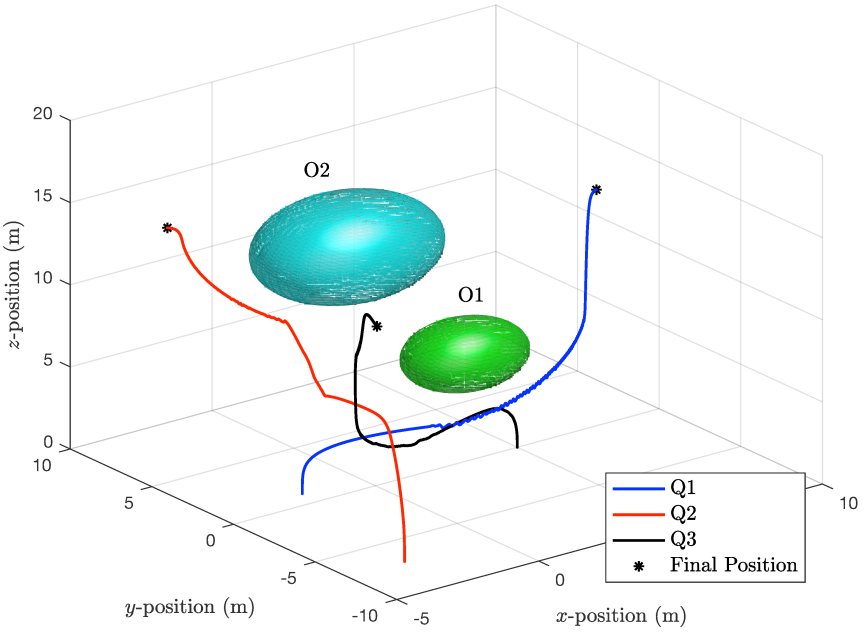

Figure 7.5: 3-dimensional view with obstacle avoidance (Case 4)

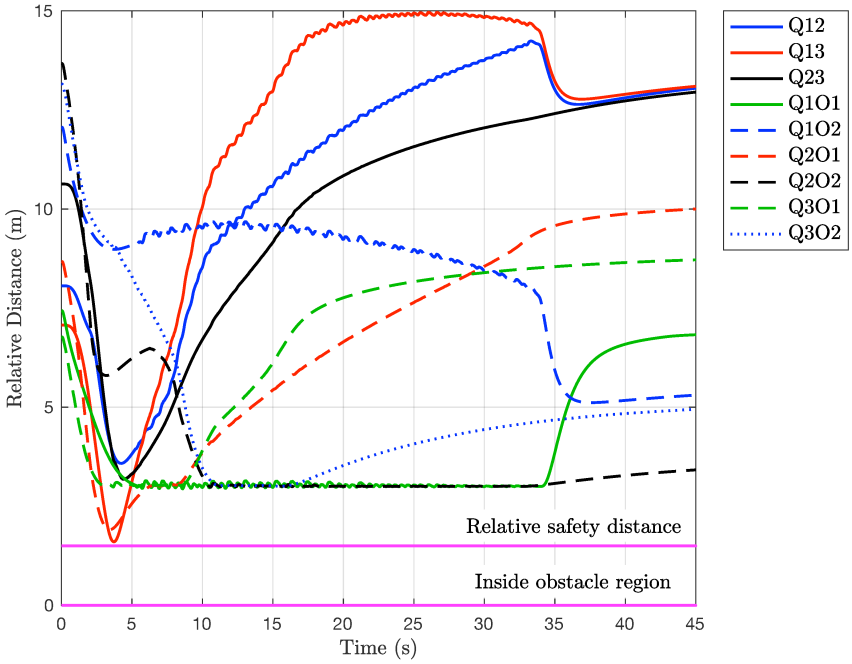

Figure 7.6: Relative distances with obstacle avoidance (Case 4)

the z-direction control action. It is possible that this chatter is the result of the optimization function solver "fmincon". In many sampling instants the solver prints to the Command Window "Local minimum possible", while in other cases it prints "Local minimum found". According to Matlab documentation, this message is an indication that the solver is unsure if the local minimum was found and therefore, may have solved for a suboptimal solution. Otherwise, a larger move suppression could be applied.

In Fig.7.8, it can be seen that the quadcopters reach consensus in the ψ-direction and converge to a value of 0 radians. The largest value among the quadcopters is 0.1353 radians for the pitch angle and quite larger for the roll angle at 0.1870 radians. Both of these angles satisfy the small angle condition according to Appendix C.

7.2 Case Design Comparison

The similarities of the control designs for collision avoidance and obstacle avoidance, include the use of predicted relative distances for constraint generation. This specifically applies (5.3), which is implemented in collision avoidance Case 2 (Algorithm

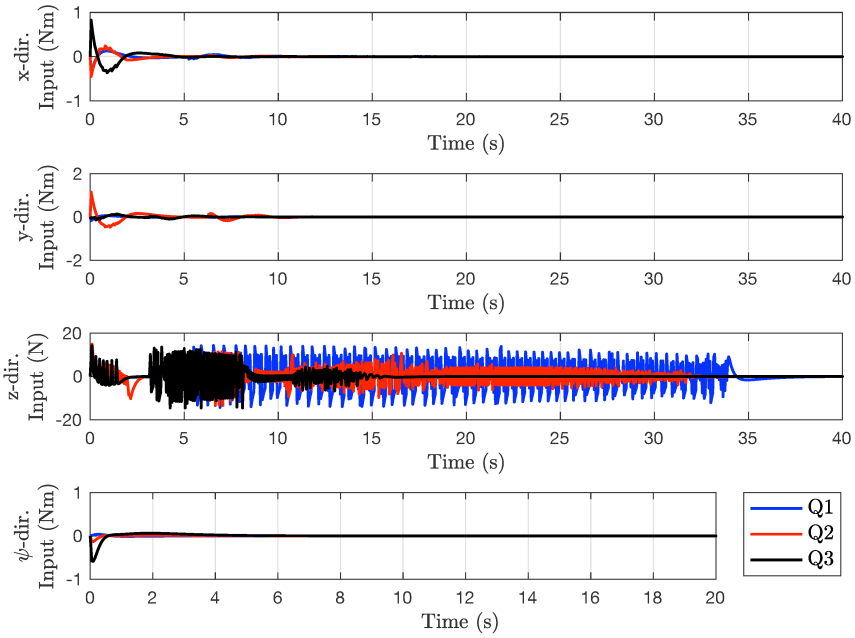

Figure 7.7: Control actions with obstacle avoidance (Case 4)

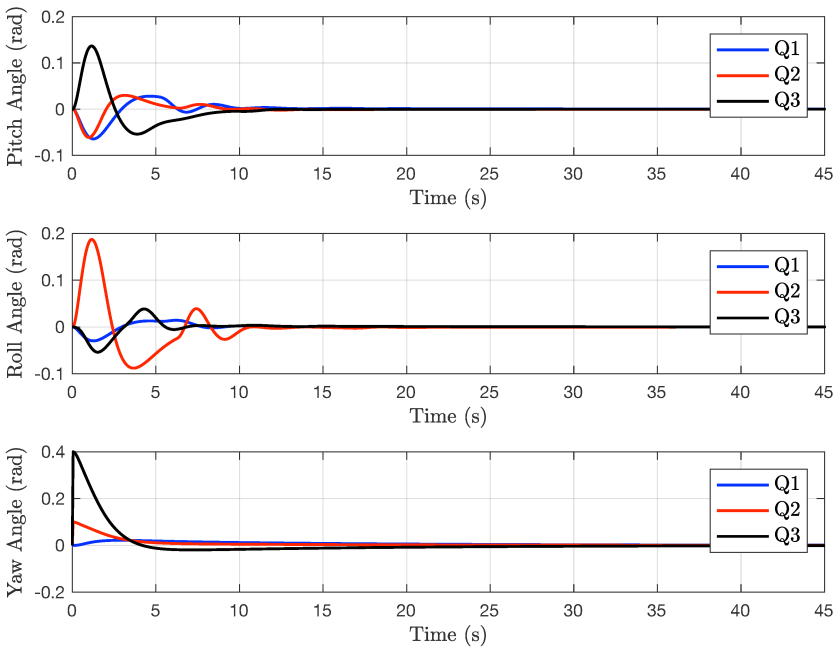

Figure 7.8: Roll, pitch, yaw angles with obstacle avoidance (Case 4)

2). However, the differences between the collision avoidance Algorithm 2 and the obstacle avoidance Algorithm 4, can provide greater insight on the considerations that led to current control designs. The main consideration in obstacle avoidance of static objects, is that the avoidance solely depends on the agent's tactic. Simple evaluation: the obstacle will not move, so the agent must. From this idea, the quadcopter in Algorithm 4 will generate constraints all x, y, z-directions to obtain full evasive movement. The opposite idea was the main consideration in collision avoidance Algorithm 2. In a team of agents communicating with one another, the avoidance depends on all agents involved. For instance, consider two agents approaching each other with the ability of full evasive movement. It is more likely that they will collide during the evasive movement, than if they were restricted to evading in different directions, like in Algorithm 2, or in the positive and negative directions of an axis, like in Algorithm 3.

Chapter 8

Conclusions and Future Work

The objective of this work was to develop a control strategy for collision and obstacle avoidance of multiple quadcopters. The proposed MPC strategy and variations, allow for multiple quadcopters to reach consensus in the x, y, ψ-directions and obtain a preassigned desired altitude in the z-direction. The collision and obstacle avoidance is achieved by applying conditional output constraints on the x, y, z-position of the quadcopter, based on the positions of neighbouring quadcopters. If collision is not imminent, then the output constraints are not applied to the MPC optimization function. This on/off feature aids in reducing the frequency the MPC optimization function is subject to hard constraints. In turn, this promotes minimizing computation times. In addition, implementing logarithmic barrier functions as input rate constraints on the control actions, help to reduce computation times. Simulation studies for a team of four quadcopters illustrated successful trials of the case variations under the proposed condition that all quadcopters start from rest. This is a feature that is practical in real world applications but often overlooked. Extensive simulations were conducted to properly interpret the parameter effects of the distributed MPC on the overall system performance. A general understanding of the MPC parameters and initial condition effects were determined, however, due to mutual parametric dependence, the effects of communication topology has yet to be determined. As an isolated study, a case variation of the proposed control method was successfully applied for obstacle avoidance of static objects.

There are many potential extensions and current challenges to consider for future development of this work. First would be to apply the proposed control strategy and variations to a real system of multiple quadcopters. This would be an interesting experiment since no current collision avoidance methods employ DMC. Recall that DMC can use real experimental data to develop predictions and can correct for modelling mismatch and nonlinearities. Currently, the controllers depend on obtaining

full state feedback to stabilize the marginally stable quadcopter dynamics. Output feedback could be investigated, or another method of stabilizing the system without full state knowledge could be explored. It is also practical to explore collision avoidance control design for quadcopters subject to time delays or communication constraints. Other considerations include hardware limitations such as noisy sensor measurements or onboard processing power. Solutions to these possible extensions will aid in the main goal of improving performance and operational capabilities of multiple quadcopters.

Bibliography

[1] T. Luukkonen, "Modelling and control of quadcopter," tech. rep., Aalto University, 2011.

[2] Q. Ali and S. Montenegro, "Explicit model following distributed control scheme for formation flying of mini uavs," *IEEE Access*, vol. 4, pp. 397–406, 2016.

[3] H. Du, J. Zhang, W. Zhu, and D. Wu, "Finite-time consensus control for a group of quadrotor aircraft," in *2017 Chinese Automation Congress (CAC)*, pp. 1531–1536, Oct 2017.

[4] Y.-J. Pan, H. Werner, Z. Huang, and M. Bartels, "Distributed cooperative control of leader-follower multi-agent systems under packet dropouts for quadcopters," *Systems & Control Letters*, vol. 106, pp. 47 – 57, 2017.

[5] N. T. Nguyen, I. Prodan, and L. Lefevre, "Multi-layer optimization-based control design for quadcopter trajectory tracking," in *2017 25th Mediterranean Conference on Control and Automation (MED)*, pp. 601–606, July 2017.

[6] Y. Lin and S. Saripalli, "Sampling-based path planning for UAV collision avoidance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, pp. 3179–3192, Nov 2017.

[7] S. Vera, J. Cobano, G. Heredia, and A. Ollero, "Collision avoidance for multiple UAVs using rolling-horizon policy," *Journal of Intelligent & Robotic Systems*, vol. 84, 10 2016.

[8] Q. Gong, C. Wang, Z. Qi, and Z. Ding, "Gradient-based collision avoidance algorithm for second-order multi-agent formation control," in *2017 36th Chinese Control Conference (CCC)*, pp. 8183–8188, July 2017.

[9] A. Mondal and L. Behera, "Gradient-based collision free desired formation generation," *IFAC Proceedings Volumes*, vol. 47, no. 1, pp. 448 – 454, 2014. 3rd International Conference on Advances in Control and Optimization of Dynamical Systems (2014).

[10] A. Souliman, A. Joukhadar, H. Alturbeh, and J. F. Whidborne, "Real time control of multi-agent mobile robots with intelligent collision avoidance system," in *2013 Science and Information Conference*, pp. 93–98, Oct 2013.

[11] Y. Xia, X. Na, Z. Sun, and J. Chen, "Formation control and collision avoidance for multi-agent systems based on position estimation," *ISA Transactions*, vol. 61, pp. 287 – 296, 2016.

[12] Y. Kuriki and T. Namerikawa, "Experimental validation of cooperative formation control with collision avoidance for a multi-UAV system," in *2015 6th International Conference on Automation, Robotics and Applications (ICARA)*, pp. 531–536, Feb 2015.

[13] A. Mondal, L. Behera, S. R. Sahoo, and A. Shukla, "A novel multi-agent formation control law with collision avoidance," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 3, pp. 558–568, 2017.

[14] D. E. Chang, S. C. Shadden, J. E. Marsden, and R. Olfati-Saber, "Collision avoidance for multiple agent systems," in *42nd IEEE International Conference on Decision and Control*, vol. 1, pp. 539–543 Vol.1, Dec 2003.

[15] P. Long, W. Liu, and J. Pan, "Deep-learned collision avoidance policy for distributed multiagent navigation," *IEEE Robotics and Automation Letters*, vol. 2, pp. 656–663, April 2017.

[16] T. Mylvaganam, M. Sassano, and A. Astolfi, "A differential game approach to multi-agent collision avoidance," *IEEE Transactions on Automatic Control*, vol. 62, pp. 4229–4235, Aug 2017.

[17] E. F. Camacho and C. Bordons, *Model Predictive Control*. Springer-Verlag London, 1999.

[18] A. Bemporad and C. Rocchi, "Decentralized linear time-varying model predictive control of a formation of unmanned aerial vehicles," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pp. 7488–7493, Dec 2011.

[19] Y. Kuriki and T. Namerikawa, "Formation control with collision avoidance for a multi-UAV system using decentralized MPC and consensus-based control," in *Control Conference (ECC), 2015 European*, pp. 3079–3084, July 2015.

[20] M. Ille and T. Namerikawa, "Collision avoidance between multi-UAV-systems considering formation control using MPC," in *IEEE International Conference on Advanced Intelligent Mechatronics*, July 2017.

[21] A. Richards and J. How, "Decentralized model predictive control of cooperating UAVs," in *2004 43rd IEEE Conference on Decision and Control*, vol. 4, pp. 4286–4291, December 2004.

[22] L. Dai, Q. Cao, Y. Xia, and Y. Gao, "Distributed MPC for formation of multi-agent systems with collision avoidance and obstacle avoidance," *Journal of the Franklin Institute*, vol. 354, no. 4, pp. 2068 – 2085, 2017.

[23] J. Rossiter, *Model-Based Predictive Control: A Practical Approach*. CRC Press, 2003.

[24] R. V. Parys and G. Pipeleers, "Distributed model predictive formation control with inter-vehicle collision avoidance," in *2017 11th Asian Control Conference (ASCC)*, pp. 2399–2404, Dec 2017.

[25] P. Wang and B. Ding, "A synthesis approach of distributed model predictive control for homogeneous multi-agent system with collision avoidance," *International Journal of Control*, vol. 87, no. 1, pp. 52–63, 2014.

[26] F. L. Lewis, H. Zhang, K. Hengster-Movric, and A. Das, *Cooperative Control of Multi-Agent Systems: Optimal and Adaptive Design Approaches*. Springer, 2014. Graph theory Chapter 2.

[27] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, pp. 1520–1533, Sept 2004.

[28] Z. Huang and Y. J. Pan, "Observer based leader following consensus for multi-agent systems with random packet loss," in *2017 IEEE Conference on Control Technology and Applications (CCTA)*, pp. 1698–1703, Aug 2017.

[29] F. Meng, J. Xi, Z. Shi, and Y. Zhong, "Leader-following consensus for singular swarm systems," in *Proceedings of the 31st Chinese Control Conference*, pp. 6357–6362, July 2012.

[30] L. Sheng, Y.-J. Pan, and X. Gong, "Consensus formation control for a class of networked multiple mobile robot systems," *Journal of Control Science and Engineering*, vol. 2012, Jan. 2012.

[31] U. Pilz, A. P. Popov, and H. Werner, "Robust controller design for formation flight of quad-rotor helicopters," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pp. 8322–8327, Dec 2009.

[32] K.-K. Oh, M.-C. Park, and H.-S. Ahn, "A survey of multi-agent formation control," *Automatica*, vol. 53, pp. 424 – 440, 2015.

[33] J. S. Shamma, ed., *Cooperative Control of Distributed Multi-Agent Systems*. John Wiley & Sons Ltd, 2007.

[34] J. M. Maestre and R. R. Negenborn, eds., *Distributed Model Predictive Control Made Easy*. Springer, 2014.

[35] F. Lamnabhi-Lagarrigue, A. Annaswamy, S. Engell, A. Isaksson, P. Khargonekar, R. M. Murray, H. Nijmeijer, T. Samad, D. Tilbury, and P. V. den Hof, "Systems & control for the future of humanity, research agenda: Current and future roles, impact and grand challenges," *Annual Reviews in Control*, vol. 43, pp. 1 – 64, 2017.

[36] P. Castillo, R. Lozano, and A. E. Dzul, *Modelling and Control of Mini-Flying Machines*. Springer, 2005.

[37] J.-J. Slotine and W. Li, *Applied Nonlinear Control*. Prentice-Hall, 1991.

[38] L. R. G. Carrillo, A. Dzul, R. Lozano, and C. Pgard, *Quad Rotorcraft Control: Vision-Based Hovering and Navigation*. Springer, 2012. Euler Langrange formulation.

[39] P. Castillo, A. Dzul, and R. Lozano, "Real-time stabilization and tracking of a four-rotor mini rotorcraft," *IEEE Transactions on Control Systems Technology*, vol. 12, pp. 510–516, July 2004.

[40] D. Lara, A. Sanchez, R. Lozano, and P. Castillo, "Real-time embedded control system for VTOL aircrafts: Application to stabilize a quad-rotor helicopter," in *International Conference on Control Applications*, Oct 2006. Proceedings of the 2006 IEEE.

[41] Y. Kuriki and T. Namerikawa, "Formation control of UAVs with a fourth-order flight dynamics," *52nd IEEE Conference on Decision and Control*, pp. 6706–6711, 2013.

[42] L. Wang, *Model Predictive Control System Design and Implementation Using MATLAB*. Springer, 2009.

[43] W. S. Levine, ed., *The Control Systems Handbook: Control System Advanced Methods*, ch. 28: Linear Model Predictive Control in the Process Industries, pp. 28.1–28.24. CRC Press, 2010.

[44] C. R. Cutler and B. L. Ramaker, "Dynamic matrix control - a computer control algorithm," *Automatic Control Conference*, 1980.

[45] K. Z. Qi and D. G. Fisher, "Model predictive control for open loop unstable process," in *1993 American Control Conference*, pp. 791–795, June 1993.

[46] N. S. Nise, *Control Systems Engineering*. John Wiley & Sons, 6th ed., 2011.

[47] J. K. Lee and S. W. Park, "Model predictive control for multivariable unstable processes with constraints on manipulated variables," *Korean Journal of Chemical Engineering*, vol. 8, pp. 195–202, July 1991.

[48] K. Ogata, *Modern Control Engineering*. Upper Saddle River, NJ, USA: Prentice Hall, 5th ed., 2010.

[49] A. G. Wills and W. P. Heath, "Barrier function based model predictive control," *Automatica*, vol. 40, no. 8, pp. 1415 – 1422, 2004.

[50] R. Dunia and G. Fernandez, "MPC with conditional penalty cost," in *2009 35th Annual Conference of IEEE Industrial Electronics*, pp. 1657–1662, Nov 2009.

[51] A. V. Hystad, "Model, design and control of a quadcopter," Master's thesis, Norwegian University of Science and Technology, 2015. Reference for control limits.

[52] T. Bennison and E. Hall, *A Level Mathematics: A Comprehensive and Supportive Companion to the Unified Curriculum*. Tarquin Group, 2016.

Appendix A

Quadcopter Model - Coriolis Terms

As presented in [1] the Coriolis terms are contained in the matrix

$$C(\eta, \dot{\eta}) = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix} , \tag{A.1}$$

$$\begin{aligned}
C_{11} &= 0 \\
C_{12} &= (I_{yy} - I_{zz})(\dot{\theta}C_{\phi}S_{\phi} + \dot{\psi}S_{\phi}^2C_{\theta}) + (I_{zz} - I_{yy})\dot{\psi}C_{\phi}^2C_{\theta} - I_{xx}\dot{\psi}C_{\theta} \\
C_{13} &= (I_{zz} - I_{yy})\dot{\psi}C_{\phi}S_{\phi}C_{\theta}^2 \\
C_{21} &= (I_{zz} - I_{yy})(\dot{\theta}C_{\phi}S_{\phi} + \dot{\psi}S_{\phi}C_{\theta}) + (I_{yy} - I_{zz})\dot{\psi}C_{\phi}^2C_{\theta} + I_{xx}\dot{\psi}C_{\theta} \\
C_{22} &= (I_{zz} - I_{yy})\dot{\phi}C_{\phi}S_{\phi} \\
C_{23} &= -I_{xx}\dot{\psi}S_{\theta}C_{\theta} + I_{yy}\dot{\psi}S_{\phi}^2S_{\theta}C_{\theta} + I_{zz}\dot{\psi}C_{\phi}^2S_{\theta}C_{\theta} \\
C_{31} &= (I_{yy} - I_{zz})\dot{\psi}C_{\theta}^2S_{\phi}C_{\phi} - I_{xx}\dot{\theta}C_{\theta} \\
C_{32} &= (I_{zz} - I_{yy})(\dot{\theta}C_{\phi}S_{\phi}S_{\theta} + \dot{\phi}S_{\phi}^2C_{\theta}) + (I_{yy} - I_{zz})\dot{\phi}C_{\phi}^2C_{\theta} + I_{xx}\dot{\psi}S_{\theta}C_{\theta} \\
&\quad - I_{yy}\dot{\psi}S_{\phi}^2S_{\theta}C_{\theta} - I_{zz}\dot{\psi}C_{\phi}^2S_{\theta}C_{\theta} \\
C_{33} &= (I_{yy} - I_{zz})\dot{\phi}C_{\phi}S_{\phi}C_{\theta}^2 - I_{yy}\dot{\theta}S_{\phi}^2C_{\theta}S_{\theta} - I_{zz}\dot{\theta}C_{\phi}^2C_{\theta}S_{\theta} + I_{xx}\dot{\theta}C_{\theta}S_{\theta} \, .
\end{aligned}$$

Note: $C_x = \cos(x)$, $S_x = \sin(x)$ and I_{xx}, I_{yy}, I_{zz} are the inertia values for the quadcopter.

Appendix B

Case 3 - Positions for x and y Constraint Generation

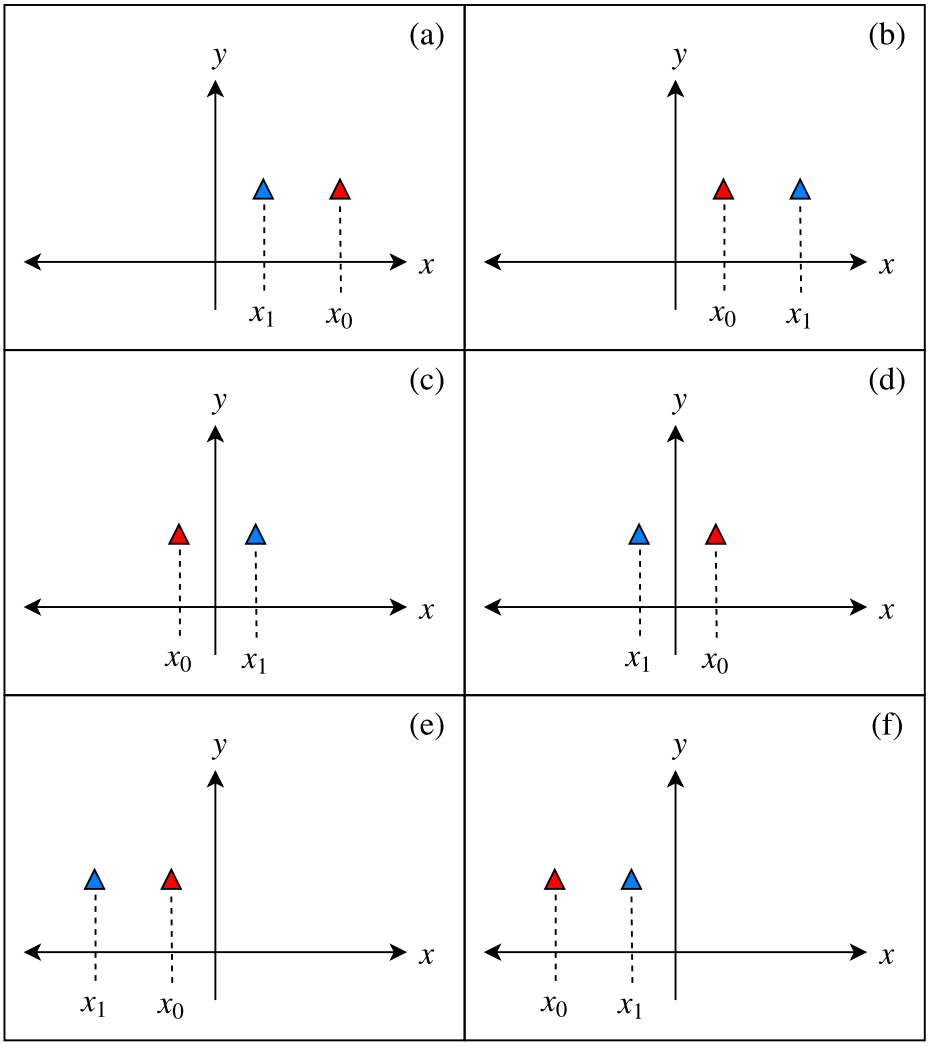

Figure B.1: Previous and current x-positions for constraint generation

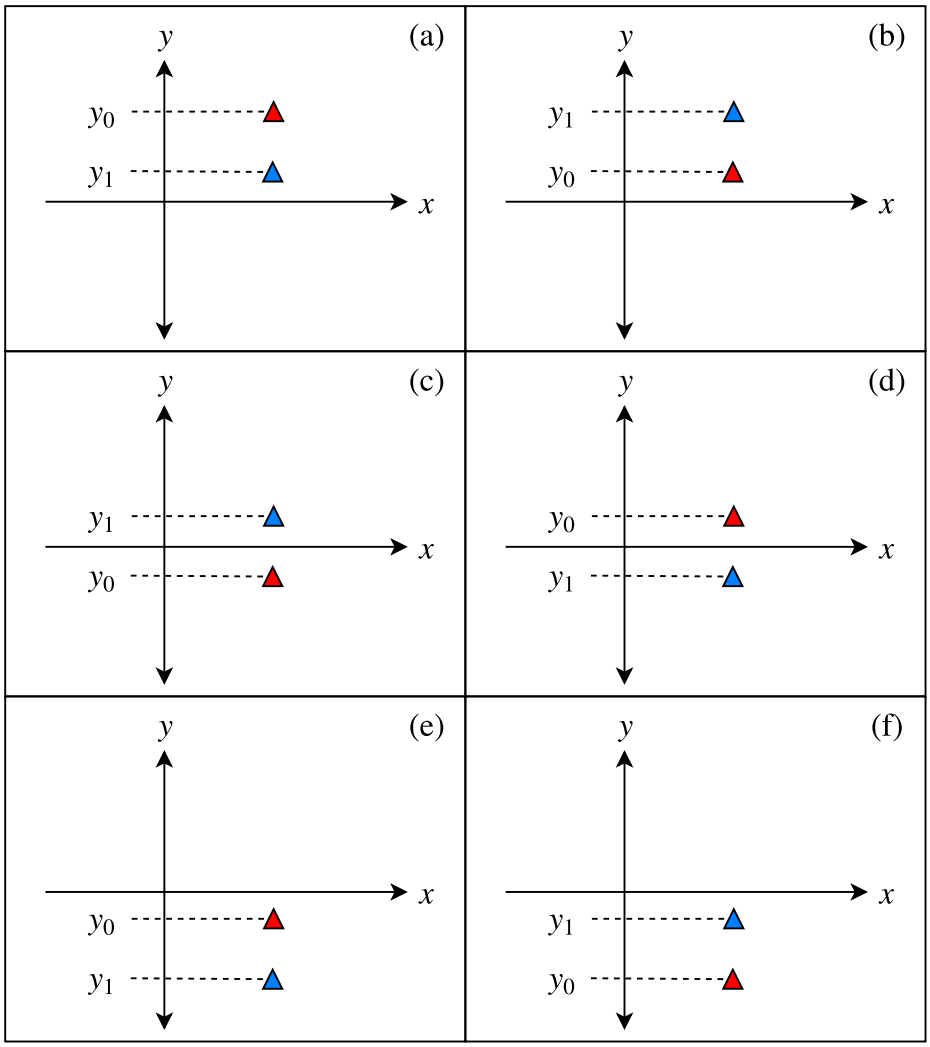

Figure B.2: Previous and current y-positions for constraint generation

Appendix C

Small Angle Approximation

Let the approximate sine value for small angles be represented as,

$$\sin\theta \approx \theta \ . \tag{C.1}$$

Similarly in [52], the approximate cosine value can be determined from a double angle identity as,

$$\begin{aligned} \cos 2\theta &\approx 1 - 2\sin\theta^2 \ , \\ \cos\theta &\approx 1 - 2\sin\tfrac{1}{2}\theta^2 \ . \end{aligned} \tag{C.2}$$

Using (C.1), it can be stated that if θ is small, then $\sin\frac{1}{2}\theta \approx \frac{1}{2}\theta$. Now substituting into (C.2),

$$\cos\theta \approx 1 - 2(\tfrac{1}{2}\theta^2) \ ,$$

$$\cos\theta \approx 1 - \tfrac{\theta^2}{2} \ . \tag{C.3}$$

The percent errors of equations C.1 and C.3, with respect to the actual sine and cosine values is presented in Fig.C.1. It is observed, that a minimal percent error of 1% is obtained when $\theta = 0.245$ radians for sine, and $\theta = 0.663$ radians for cosine. The values determined at 1% error can be used as a benchmark for the quadcopter linearization condition of small ϕ and θ angles. Therefore, it is safe to reason that roll and pitch angle values less than 0.245 radians are sufficient for the quadcopter linearization condition.

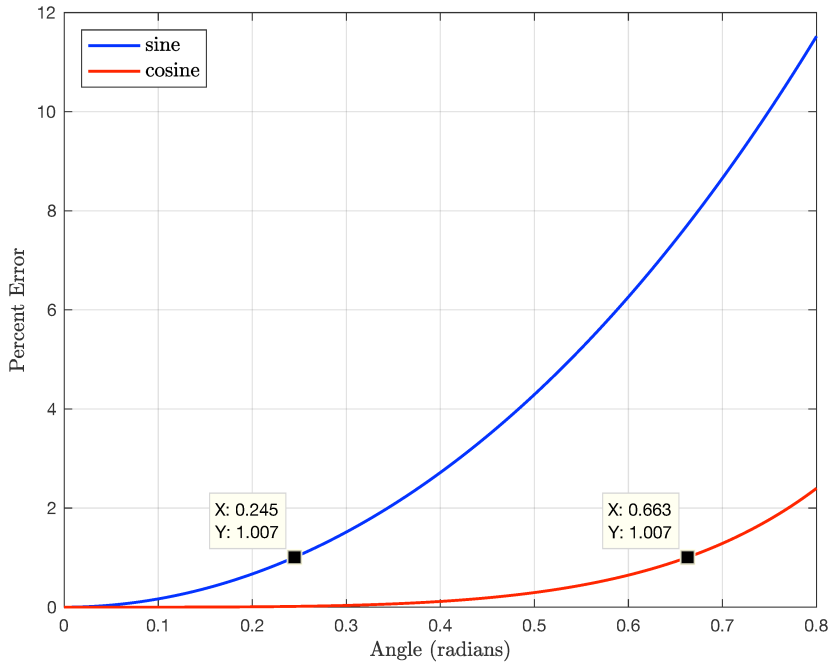

Figure C.1: Percent error between approximation and actual values

Appendix D

Author's Publication List

Peer-Reviewed

S. Dubay and Y.J. Pan,"Distributed MPC based Collision Avoidance Approach for Consensus of Multiple Quadcopters", *Proceedings of the 14th IEEE International Conference on Control and Automation*, 2018.

S. Dubay, Y.J. Pan, M. Charest and D. Shukla,"A Master Follower Device for Demonstrating Concepts of Modelling and Control", *Proceedings of the 2016 CSME International Congress*, 2016.

Other Publications

S. Dubay and Y.J. Pan, "Distributed MPC for Multi-Agent Systems with Applications to Quadcopters", (Presentation) *2017 Dalhousie University Mechanical and Materials Engineering Conference*, 2017.

In Preparation

S. Dubay and Y.J. Pan, "Distributed DMC with Constrained Optimization for Collision and Obstacle Avoidance of Multiple Quadcopters", (Journal).